UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema de balizas digitales para entornos de Smart Cities

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Jorge Ros Gil

Tutor: Pietro Manzoni

Curso 2018-2019

Resumen

Las ciudades del futuro se enfrentan a grandes retos debido al aumento exponencial de su población. Estas deben seguir ofreciendo los servicios necesarios para asegurar el bienestar de sus ciudadanos manteniendo el nivel de calidad de estos. Muchas de estas ciudades ya confían en la tecnología para conseguirlo, convirtiéndose de esta manera en ciudades inteligentes o *Smart Cities*. Este trabajo tiene por objetivo proponer un protocolo de comunicación que permita la interacción entre dispositivos del Internet de las Cosas dentro de una ciudad inteligente, siendo este flexible y adaptable a cualquier contexto. Además, se establece un marco teórico que permite introducir al lector en el ámbito de este trabajo.

Palabras clave: ciudad, inteligente, flexible, protocolo, adaptable, Internet de las Cosas.

Abstract

Cities of the future face great challenges due to the exponential growth of their population. They must keep offering the services needed to ensure the wellness of its citizens while assuring their quality. Many of these cities are already using technology to achieve this goal, becoming therefore Smart Cities. This paper aims to propose a communication protocol which allows the interaction between the Internet of Things devices located in a Smart City, being flexible and adaptative to any context. Besides, a theoretical guide is established to introduce the reader to the field in which this paper is involved.

Keywords: city, smart, flexible, protocol, adaptative, Internet of Things.

Tabla de contenidos

Agradecimientos

A mis padres, su esfuerzo y apoyo han hecho posible que hoy me encuentre en el lugar en el que estoy. Sin sus ánimos y confianza en mí nada de esto hubiera sido posible. Cualquier logro que pueda conseguir a partir de ahora también provendrá de todos los esfuerzos que han puesto en mi educación como persona y la forma en que me han transmitido el amor por sus profesiones.

A mi novia, quien desde el momento en que la conocí ha conseguido sorprenderme con su fuerza y ganas de superarse día a día, transmitiéndome esta forma de ser y que ha conseguido hacerme ver mis puntos fuertes, algo no tan sencillo como ver los débiles.

Por supuesto al tutor de este Trabajo de Fin de Máster, Pietro Manzoni, quien me propuso este tema, el cuál me pareció perfecto desde el primer momento. Además, dado que tuve la suerte de tenerlo como profesor en una asignatura del máster, puedo afirmar que ha inspirado e inspirará a muchos ingenieros a desarrollar apego por las materias que imparte.

Y por último, a mis compañeros de trabajo en el sector salud de Everis, quienes me acogieron con los brazos abiertos desde el primer momento. Con su experiencia están consiguiendo hacer de mí un mejor profesional de este bello campo que es la Ingeniería Informática.

1.Introducción

En los últimos años se ha podido observar la gran transformación que ha sufrido la tecnología, tanto en el tamaño de los dispositivos puestos a la venta como en la forma en la que los seres humanos interactuamos con ella. En tan solo unas décadas se ha conseguido reducir el tamaño de los componentes electrónicos, aumentar las capacidades de procesamiento y almacenamiento y mejorar la interacción con el usuario, pieza clave en el desarrollo de la tecnología en esta nueva era digital.

De requerir una habitación de 10x17 metros como hacía el ENIAC en el año 1946 para obtener una capacidad de procesamiento de cinco mil sumas por segundo, treinta y cinco multiplicaciones y tres raíces cuadradas o divisiones, se ha pasado a obtener los mismos e incluso mucho mejores resultados en tan sólo unos nanómetros de silicio.

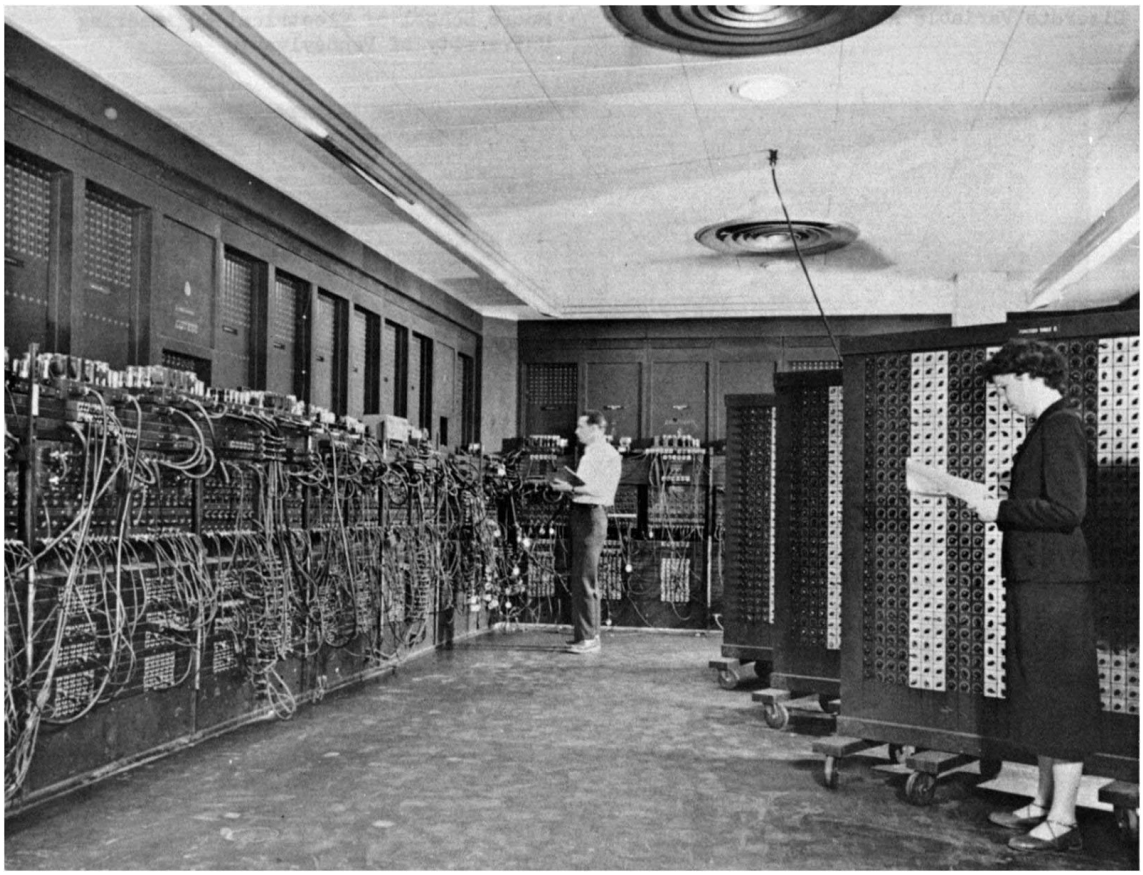

Imagen 1.1 Computador ENIAC

Estos cambios tan evidentes han permitido la popularización y extensión de la tecnología, pudiendo ser encontrada hoy en día en cualquier ámbito, como puede ser la automoción, el comercio, el ocio y un largo etcétera. Toda esta tecnología está siendo utilizada para cambiar la forma en la que vivimos, consiguiendo que sea más ágil, cómoda y sencilla.

Si bien es cierto que este crecimiento es claramente positivo debido a la ayuda que puede ofrecer la tecnología a los problemas contemporáneos, esta debe ser utilizada con cautela, y siempre surgiendo de investigaciones rigurosas que midan el impacto en la sociedad que esta puede ocasionar y determinando la mejor forma de

diseñarla para conseguir el mayor beneficio posible a la hora de implantarla o comercializarla.

Uno de los campos surgidos a partir de esta revolución que aprovecha la disminución del tamaño de los aparatos electrónicos es el **Internet de las Cosas**. Este nicho de investigación se centra en la utilización de pequeños *chips* que acompañan a los objetos de nuestro día a día, convirtiéndolos de esta manera en pequeños computadores conectados a Internet que comparten su estado con el resto del mundo y reciben indicaciones sobre cómo proceder de manera remota.

Toda esta tecnología insertada en los objetos está siendo utilizada para mejorar las tareas cotidianas y repetitivas de los humanos, aumentando el control sobre ellas y delegando en los dispositivos las más aburridas o las que requieren más esfuerzo. De esta manera se consigue una mayor eficiencia en la realización de los quehaceres diarios y el aumento de la calidad en los mismos, ya que los seres humanos solemos perder precisión y capacidad de atención cuando las tareas se vuelven repetitivas y demasiado sencillas.

Este paradigma tecnológico está siendo implantado en la gestión de las ciudades alrededor del mundo, y es uno de los objetivos de este trabajo recoger las ideas enmarcadas en este campo para obtener el desarrollo de un producto relacionado con las ciudades inteligentes.

1.1 Motivación

Este trabajo de fin de máster surge del interés del autor en el ámbito del Internet de las Cosas y en como éste puede transformar radicalmente la forma en la que vivimos. Debido a que las comunicaciones son una de las partes más importantes en este campo y que las ciudades inteligentes son uno de los lugares en los que ahora mismo puede encontrarse una mayor aplicación de tecnologías punteras, la combinación de ambas resulta en un campo de investigación de gran interés y que todavía cuenta con amplios puntos que pueden ser desarrollados.

Se ha creído útil aportar a este campo de investigación un trabajo en el que se diseñara una forma específica de comunicar dispositivos dentro de una ciudad inteligente de manera que esta se realizara eficientemente teniendo en cuenta los recursos de los que se dispone y las necesidades de los ciudadanos.

1.2 Objetivos

1. Diseñar un protocolo de comunicación para entornos de *Smart Cities* en el que un dispositivo inteligente situado en un punto de la ciudad actúe como baliza para proporcionar información a otros dispositivos que realizan consultas sobre ella con el rol de clientes.

2. Obtener una solución independiente del ámbito de uso y flexible para la problemática de las comunicaciones a través de Internet en entornos de ciudades inteligentes.

3. Desarrollar una solución que haga uso del protocolo propuesto y que actúe como prototipo, demostrando la utilidad y viabilidad de este. Para ello se implementa la comunicación entre dos o más dispositivos electrónicos que intercambian mensajes con el formato propuesto.

4. Implementar un prototipo de aplicación web que actúe como sala de control para monitorizar el estado de los dispositivos ubicados en la ciudad y realizar configuraciones sobre ellos remotamente.

5. Reunir las diferentes tendencias en el ámbito del Internet de las Cosas para obtener un manual de referencia breve sobre las mismas.

6. Crear una base teórica y experimental para el desarrollo de futuros trabajos en este ámbito, colaborando en la investigación en la mejora de la integración entre sistemas del Internet de las Cosas.

1.3 Alcance del proyecto

Este proyecto pretende proponer un protocolo de comunicación lo suficientemente flexible e independiente del contexto como para ser utilizado en cualquier caso de uso de ciudades inteligentes en el que se requiera de un descubrimiento de agentes vecinos y afines en la red.

Además, presenta un prototipo en el que se hace uso de este como demostración de la adaptabilidad y viabilidad de su uso en entornos reales de ciudades inteligentes.

1.4 Estructura de la memoria

El presente documento es la memoria resultante del trabajo de fin de máster desarrollado y su estructura se define de la siguiente manera:

En primer lugar se ha realizado una breve introducción al ámbito del Internet de las Cosas, permitiendo introducir al lector en determinada terminología utilizada durante todo el documento e informándole de las tendencias actuales en el mercado. En un punto siguiente se pasa a describir los detalles correspondientes a un subgrupo del Internet de las Cosas: las *smart cities*, ámbito sobre el cual versa la solución propuesta en este trabajo. Además, se expone un ejemplo muy cercano de ciudad inteligente, el de la ciudad de Valencia, reuniendo los distintos servicios que esta ya ofrece y los que ofrecerá en un futuro cercano.

Habiendo introducido al lector de esta memoria en los conceptos necesarios, se pasa a describir el protocolo de comunicación propuesto, objetivo principal del trabajo de fin de máster. Se define el flujo de comunicación que deben seguir los dispositivos que interactúen haciendo uso del protocolo citado y se comenta la estructura que deberán tener los diferentes mensajes empleados en la comunicación.

Una vez descrito teóricamente el protocolo, se pasa a explicar cómo se ha desarrollado el prototipo que hace uso de este, el cuál permite comunicar dos o más dispositivos electrónicos, actuando siempre uno de ellos como punto de acceso o *beacon* o punto de acceso y el resto como clientes. Además, se exponen también los detalles de implementación de la aplicación web cuya función es actuar como sala de control de una ciudad que cuente con dispositivos cuya comunicación se realice mediante la utilización del protocolo propuesto.

Con el protocolo expuesto y el prototipo desarrollado descrito, se pasan a exponer los diferentes retos a los que se enfrentan tanto el Internet de las Cosas como las ciudades inteligentes. También se exponen las posibles continuaciones que podría tener el trabajo desarrollado en el futuro.

Con todos estos puntos tratados, se pasa a establecer las conclusiones obtenidas tras la realización del trabajo y se enumera la bibliografía empleada para documentar esta memoria y obtener la información necesaria para todo el desarrollo del proyecto.

2. Internet de las cosas

El término Internet de las Cosas (en ocasiones conocido como IoT por sus siglas en inglés *Internet of Things*) surge del título de una presentación realizada por **Kevin Ashton** en Procter & Gamble (P&G) en el año 1999. Por aquel entonces Ashton era cofundador y director ejecutivo del Auto-ID center, un grupo de investigación del Massachusetts Institute of Technology que centra su investigación en el campo de la identificación por radiofrecuencia (más conocida como RFID).

En su presentación, Ashton destacó que todos los datos que por aquel momento circulaban por la red, alrededor de 50 petabytes, habían sido introducidos por seres humanos. No solo mediante teclado, si no también escaneando archivos, presionando un botón de grabación o escaneando códigos de barras. Esto suponía un grave problema según su parecer, ya que los humanos cuentan con un tiempo y atención limitados y poca precisión en lo que hacen comparado con las máquinas. Por todo esto, Ashton consideró que los humanos no somos muy buenos a la hora de introducir datos sobre las cosas que nos rodean.

Por ello expresó la necesidad de que fueran los propios objetos ubicados en el mundo real los que crearan los datos sobre su estado, para poder saber si por ejemplo necesitaban reparación o ser sustituidos. De esta manera, se podría reducir el gasto energético, los deshechos, gastos económicos y un largo etcétera.

Ashton propuso utilizar tecnología RFID para:

> *"posibilitar a los ordenadores recoger información por ellos mismos, escuchar y oler el mundo, con toda su gloria aleatoria. La tecnología RFID y los sensores permiten a los computadores observar, identificar y entender el mundo —sin las limitaciones de los datos introducidos por humanos"*

Casi veinte años después de la presentación que marcó el comienzo del término IoT, la definición de este sigue siendo bastante difusa. La Unión Europea, en su documento que establece la hoja de ruta del IoT para 2020 ya lo define de dos maneras distintas según se realice teniendo en cuenta su **significado semántico** "*a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols*" o teniendo en cuenta la **funcionalidad e identidad** *"Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts"*. En la primera, se entiende el IoT como un conjunto de objetos interconectados que se comunican mediante protocolos estándares. Esta definición no se diferencia mucho de la que podríamos entender como una red de computadores, que se encuentran interconectados y comparten mensajes mediante protocolos de red.

Es la segunda la que aporta una gran diferencia. Comenta que el IoT es un conjunto de "cosas" que cuentan con identidades y personalidades virtuales y se

encuentran en un espacio inteligente, es decir, comparten espacio con otros objetos que recogen datos e interactúan con usuarios o con otros objetos.

Con estas dos definiciones podríamos entender el IoT como:

- Un conjunto de objetos interconectados, que interactúan con su entorno y otros objetos inteligentes, aportando datos sobre lo que pueden ver o inferir a partir de los datos recogidos por sus sensores y aquellos recibidos a través de las comunicaciones con otros elementos.

Existen diversas características que definen a un dispositivo del Internet de las Cosas como tal, sin las cuales simplemente sería un dispositivo más conectado a la red.

Una de ellas es que el dispositivo debe estar concebido de manera que sea **invisible**, es decir, debe proporcionar la funcionalidad para la que ha sido diseñado sin entorpecer la rutina diaria de quien interactúa con el. Es por ello que la interacción humano-Dispositivo IoT es un campo que está siendo ampliamente estudiado. Para conseguir esta invisibilidad del dispositivo se utiliza sobre todo el control por voz, siendo esta la más natural para el ser humano. Más allá de la interacción sencilla, podemos encontrar la **no interacción** en absoluto. Tras un tiempo de aprendizaje en el que el dispositivo requiere visualizar las rutinas de funcionamiento de los usuarios, este actuará de manera **proactiva** sin necesidad de recibir las órdenes.

Otra de las características que define a un dispositivo IoT es la **ubicuidad**. Debe encontrarse integrado con su entorno físico, siendo capaz de interactuar con él mediante sensores y actuadores que le permitan encontrarse conectado con el mundo que lo rodea. Además, debe ser accesible desde cualquier punto, por lo que es requisito indispensable que se encuentre conectado a Internet y que la consulta de su estado sea sencilla para cualquiera.

Por último, otra característica interesante para estos dispositivos es que sea **inteligente**. La funcionalidad que desempeña este, debe ser relevante para los usuarios que hacen uso de él y debe ser capaz de adaptarse en función de los cambios en el entorno.

El Internet de las Cosas, concepto que como hemos podido ver cuenta con aproximadamente dos décadas de desarrollo va a contar en los próximos años con un conjunto de **retos y problemáticas** que va a deber superar para poder establecerse como una tecnología útil y próspera. Entre ellos podemos encontrar el reto de asegurar la seguridad y privacidad de los usuarios que interactúen con esta tecnología, que según la definición del propio concepto van a ser todos los seres humanos. En un mundo en el que prácticamente todos los objetos se encuentren conectados entre sí y compartan información de todos los tipos, siendo esta en ocasiones de carácter sensible, es muy importante asegurar que solamente vayan a tener acceso a ella aquellos agentes que deban tenerla.

Otro de los retos con los que va a contar el IoT va a ser asegurar la **interoperabilidad** de todos los sistemas, para lo que se deberán definir estándares y métodos de trabajo de obligatorio cumplimiento para la integración exitosa de los dispositivos ya instalados previamente y de aquellos que lo serán posteriormente.

Estos retos y problemáticas serán desarrollados más extensamente en puntos posteriores de este trabajo.

Debido a que el Internet de las Cosas abarca una cantidad ingente de funcionalidades y dispositivos que pueden ser considerados como pertenecientes a esta corriente, se ha decidido desarrollar la descripción de algunos de los subgrupos más importantes del mismo, ofreciendo los objetivos que persiguen, ejemplos de uso, etc.

2.1 Smart Homes

Las smart homes o casas inteligentes son un subgrupo del Internet de las Cosas, que se refiere a aquellas viviendas que cuentan con alguno o todos sus procesos automatizados mediante componentes electrónicos, entre los que se pueden encontrar sensores, actuadores, cámaras de vigilancia, etc.

Algunos de los objetivos de las casas inteligentes son:

- **Eficiencia energética**: automatizar la utilización de la calefacción o refrigeración dependiendo de la temperatura de la casa y los hábitos de quienes viven en ella o automatizar la iluminación de las estancias, puede ayudar a reducir el gasto energético, con la consiguiente rebaja en las facturas y el menor impacto medioambiental.

- **Comodidad**: una casa inteligente puede ayudar a la comodidad de quien vive en ella automatizando los procesos que se suelen realizar frecuentemente, aprendiendo sus hábitos y actuando en consecuencia.

- **Seguridad**: la monitorización del hogar desde el exterior con sensores de presencia, de humo o cámaras de vigilancia conectadas a Internet permiten aumentar la seguridad y tranquilidad de los que viven en ella.

Dentro de una casa inteligente existen diversas situaciones que pueden ser automatizadas, pudiendo hacer así nuestra vida más fácil haciendo uso de aprendizaje automático sobre nuestras costumbres o de la actuación sobre los elementos a distancia con el uso de una conexión a Internet. Estos son algunos ejemplos de casos de uso en *Smart Homes*:

- **Calefacción inteligente**: supongamos que el usuario de nuestra casa inteligente se levanta todos los días entre semana sobre las siete de la mañana para ir a trabajar. Necesita que la temperatura de su cuarto de baño sea agradable

durante el invierno, ya que necesita ducharse antes de salir de casa. Su casa inteligente puede aprender sobre estos hábitos y encender la calefacción del cuarto de baño media hora antes de que el usuario se levante de la cama, para que así la estancia esté a una temperatura agradable sin necesidad de que él espere a que se caliente o tener que levantarse antes.

- **Iluminación automática**: nuestra casa inteligente puede encender las luces de aquellas estancias en las que nos encontramos de manera automática cuando se considere que la cantidad de luz en el interior no es suficiente. Así si nos encontramos leyendo un libro durante la tarde y la luz baja debido a la puesta de sol, nuestra casa puede entender que necesitamos que se enciendan las luces del salón. También podemos definir diferentes escenarios, por ejemplo, imaginemos que nos gusta apagar las luces cuando vemos una película. Nuestra casa puede aprender que cuando se encienda la televisión y escojamos una película en nuestro servicio de streaming favorito debe apagar las luces.

- **Ventanas o persianas inteligentes**: conociendo la hora a la que nos levantamos por la mañana, la casa puede automatizar la apertura de las persianas para dejar que entre la luz. Y enlazando con el punto anterior, si deseamos ver una película durante el día y nos gusta verlas a oscuras, la casa puede bajar las persianas del salón para crear el ambiente deseado.

Estos posibles casos de uso ya cuentan con productos en el mercado que permiten implementarlos, a continuación se describen algunos de ellos.

- **Termostato Nest**: Nest, que fue adquirido por Google tras su éxito es un fabricante de productos inteligentes para el hogar. Su producto estrella es el termostato, el cual aprende los hábitos del usuario para actuar en consecuencia ajustando la temperatura de las estancias del hogar dependiendo de la hora y época del año. Esto permite, además de aumentar la comodidad, disminuir el consumo eléctrico considerablemente, debido a que memoriza la temperatura en la que nos sentimos cómodos y la baja automáticamente cuando nos vamos de casa.

  También permite controlar la temperatura del hogar a través de Internet estando fuera de casa y en los nuevos modelos se incluye la capacidad de actuar sobre la temperatura de la caldera del agua, aprendiendo sobre nuestros gustos a la hora de ducharnos.

- **Bombilla Philips Hue**: Philips Hue es una familia de productos de iluminación consistente en bombillas, interruptores y sensores que permiten el ajuste de los colores, temporizadores de encendido, cambiar la intensidad, etc. Todo ello desde un dispositivo inteligente como puede ser un teléfono móvil conectado a Internet.

- **Amazon Echo**: Echo son un conjunto de productos diseñados y vendidos por Amazon que permiten la interacción por voz con el asistente inteligente Alexa. Permite reproducir música, consultar datos por Internet e incluso controlar dispositivos inteligentes ubicados en el hogar mediante la voz. Además aprende de nuestros hábitos y gustos, por lo que es capaz de actuar en consecuencia, realizando determinadas acciones de manera automática.

Además de los productos prediseñados y vendidos en las tiendas, existe la posibilidad de crear tu propia casa inteligente desde cero utilizando sistemas de código abierto y *Hardware* como una Raspberry Pi.

Existen sistemas operativos dedicados exclusivamente al control de dispositivos inteligentes dentro de una red domótica, y que debido a su peso ligero no requieren de *Hardware* de altas prestaciones. Algunos de ellos son: MisterHouse, Mycroft o Domoticz.

Si bien es cierto que las posibilidades dentro de las *Smart Homes* son todavía limitadas, cada día aparecen más productos nuevos que permiten actuar sobre los objetos convencionales de los hogares y que además reducen su precio con respecto a dispositivos anteriores.

2.2 Smart Farming

El aumento de la población mundial en los últimos años y el abandono del campo debido a la dureza de trabajar en él, están ocasionando que la producción agrícola, sobre todo en países desarrollados, esté cayendo a ritmos vertiginosos. En concreto, según el Instituto Nacional de Estadística, en 2018 descendió la superficie cultivada de cereales en España aproximadamente un 6% con respecto al mismo período del año anterior. Además, el cambio climático mundial amenaza la disponibilidad de recursos hídricos, con la consiguiente pérdida de cosechas y desprovisión de productos a los consumidores.

Es por ello que es necesario optimizar el uso de los recursos empleados a la hora de producir estas materias primas para combatir la falta de accesibilidad a las mismas y asegurar la rentabilidad de trabajar en el campo en el futuro.

Para poder conseguir esto surge el concepto *Smart Farming* o Agricultura Inteligente, cuyo objetivo es aplicar las tecnologías de la información a la agricultura, consiguiendo de esta manera la monitorización de las plantas, mediciones de humedad y temperatura del suelo o el control de plagas. Gracias a esto, el agricultor puede tener un mayor conocimiento sobre sus plantaciones, pudiendo ahorrar tanto recursos hídricos como esfuerzos dedicados al cuidado de los cultivos.

Además, la agricultura inteligente permite aumentar la producción, reduciendo la necesidad de recurrir a insecticidas perjudiciales para el terreno y optimizando al máximo el uso de agua.

Un ejemplo de casos de uso de agricultura inteligente que ya se encuentran hoy en día en el mercado son las sondas FDR. Estas sondas permiten la monitorización de la humedad y temperatura del suelo a diferentes profundidades, pudiendo así determinar la necesidad de efectuar riegos o no. Diversas empresas ofrecen la instalación de las mismas acompañadas por un software que permite visualizar los datos en tiempo real y actuar sobre el campo si se cree necesario, además de contener histogramas sobre la evolución del cultivo y el terreno.

Podemos encontrar casos de uso más novedosos como la utilización de drones para la evaluación del terreno cultivado o de la madurez de las frutos, para determinar si estos deben ser recogidos o no. El uso de estos vehículos aéreos e incluso de satelites para mejorar la gestión de las parcelas es conocido como **Agricultura de precisión.**

Imagen 2.1. Drone de ayuda a la toma de decisiones en ambientes rurales

2.3 Smart HealthCare

El *Smart HealthCare* o Sanidad Inteligente, hace uso de las tecnologías de la información para mejorar la salud de la ciudadanía. Este concepto puede aplicarse por ejemplo al diseño de dispositivos inteligentes para el cuidado de pacientes con determinadas enfermedades, como puede ser la diabetes. Estas personas pueden beneficiarse de la tecnología mediante el uso de medidores que les notifiquen sobre sus niveles de glucemia y les comuniquen si deben o no inyectarse insulina, pudiendo incluso realizar esta acción por ellos.

Otro ejemplo de dispositivo al servicio de la salud es el de localizadores para personas con Alzheimer. Los pacientes afectados por esta enfermedad neurodegenerativa tienden a desorientarse al salir a la calle, por lo que es útil que sus familiares puedan consultar fácilmente dónde se encuentran en caso de que este se haya perdido.

Pero además de los dispositivos IoT como tal, el *Smart HealthCare* se refiere también al tratamiento de los datos clínicos para su análisis y la disponibilidad de los mismos como servicio para ayudar a mejorar la calidad de vida de los ciudadanos.

Como todos los demás subtipos de IoT, basa sus esfuerzos en establecer que: a mayor calidad de vida, menores costes sanitarios. Es por ello que la monitorización mediante el uso de dispositivos, la mejora de hábitos de vida utilizando el análisis de la rutina del paciente o la detección precoz de enfermedades y pandemias puede aumentar la calidad de vida de los ciudadanos, reduciendo por consiguiente los costes de los servicios de salud tanto pública como privada de los países que hagan uso de la tecnología en este ámbito.

2.4 Smart Industry

La Industria Inteligente, también denominada Industria 4.0 se refiere a la inclusión dentro de la industria convencional, surgida de la tercera revolución industrial, de sistemas ciber-físicos, el Internet de las Cosas y la comunicación no solo entre personas o entre personas y máquinas, si no también la comunicación máquina-máquina (M2M) (Roblek, Meško y Krapež, 2016).

Las revoluciones industriales siempre han venido marcadas por determinadas invenciones que han permitido cambiar radicalmente la forma en la que se producen los bienes. La **primera revolución industrial** surge a partir del empleo de la máquina de vapor, lo que permitió aumentar la producción gracias a la mayor automatización y al aumento de la fuerza que se podía ejercer sobre los materiales en comparación con los métodos anteriores.

La **segunda revolución industrial** se produce con la invención del motor de combustión y la utilización por lo tanto de derivados del petróleo como combustible para las máquinas.

Además de la fuerza motriz que impulsa a las máquinas, la cual está tendiendo a ser generada mediante energías renovables en la **tercera revolución industrial**, esta se diferencia también de las anteriores por el uso intensivo de las tecnologías de comunicación gracias a Internet. El término “tercera revolución industrial” fue propuesto por Jeremy Rifkin, quién en 2006 estableció que se caracteriza por:

- La utilización de energías renovables como fuente de energía
- El desarrollo e investigación en la red de distribución eléctrica inteligente o *smart grid*
- Empleo de las tecnologías de la información para la mejora de la producción y el contacto continuo con clientes y proveedores

Como vemos, esta etapa de la industria sería la más breve de la historia, ya que su sucesora, la **cuarta revolución industrial** empieza a establecer sus bases, teniendo como aspecto más importante la fusión entre el mundo físico y el virtual, además del empleo de tecnologías punteras como la nanotecnología, inteligencia artificial o Internet de las Cosas en los procesos de producción.

Para conseguir esta fusión entre los dos mundos, esta etapa está incluyendo la conexión entre dispositivos de producción para mejorar la calidad de los productos y la velocidad con la que estos se producen, utilizando la posibilidad de autocoordinación entre los diferentes actores que intervienen en el proceso.

La industria 4.0 o *smart industry* basa su existencia en cuatro principios:

- **Interoperabilidad**: los dispositivos diseñados bajo esta denominación deben tener la capacidad de comunicarse con otros dispositivos, seres humanos, sensores, etc.

- **Transparencia de la información**: enriquecimiento de los modelos digitales de la planta mediante la información generada por los sensores ubicados en ella.

- **Asistencia técnica**: las máquinas deben permitir a las personas visualizar la información generada para permitir la toma de decisiones a corto plazo. Además, deben ayudar a las personas en tareas desagradables o muy forzosas para ellos.

- **Decisiones descentralizadas**: los sistemas deben ser capaces de tomar decisiones por sí mismos siempre y cuando no sean de importancia muy grande, en cuyo caso serán delegadas a un nivel superior.

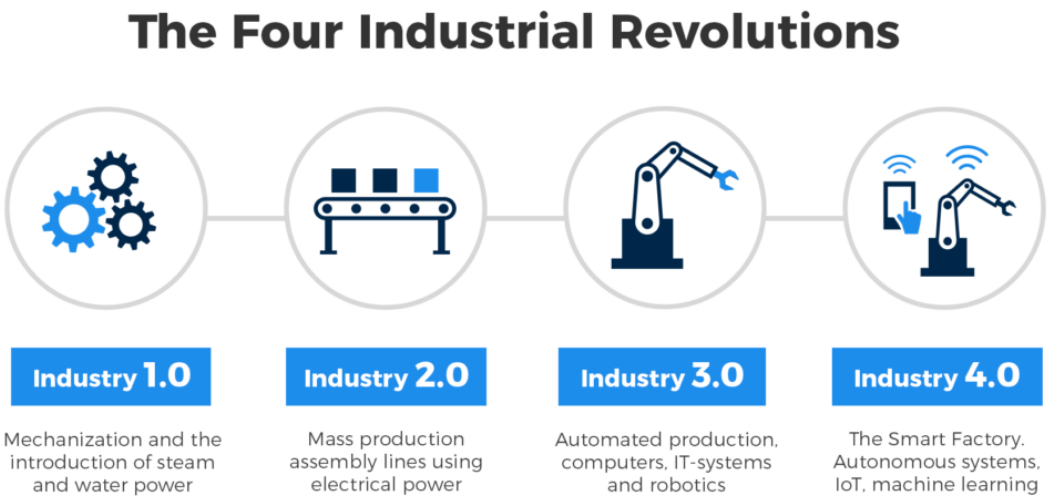

Imagen 2.2. Las cuatro revoluciones industriales. © Spectral Engines

El nexo común entre los subtipos del IoT citados es que dedican sus esfuerzos a mejorar la productividad del ámbito en el que se aplican y reducir costes, haciendo uso de dispositivos ubicados en el entorno, con el cual interactúan y aprovechando la gran cantidad de datos que estos generan para aprender y actuar en consecuencia; permitiendo así la mejora gradual en los objetivos perseguidos.

Se habrá podido observar que no se ha comentado nada sobre un subtipo que se encuentra muy de actualidad y que además es en el que se engloba este trabajo, las *Smart Cities*. Esta falta de mención es intencionada, ya que se pretende dedicar un capítulo de este trabajo solamente a él, para poder describirlo con mayor profundidad, viendo sus objetivos, retos y situación actual.

2.5 Conectividad en IoT

Como ya se ha comentado en puntos anteriores, un dispositivo electrónico no puede ser considerado como perteneciente al Internet de las cosas si no es capaz de comunicarse con el exterior, tanto haciéndolo con su entorno cercano mediante el uso de sensores y actuadores para conocer el lugar donde se encuentra ubicado, como mediante el uso de tecnologías de red para poder comunicarse con otros dispositivos.

La revolución del Internet de las Cosas surge de la posibilidad de enviar información a cualquier región del mundo en cuestión de segundos y de poder recibirla para evitar el aislamiento.

En los comienzos del concepto, se hacía uso de tecnologías que ya se encontraban presentes en la vida cotidiana, tanto en el uso de Internet como de redes de computadores internas. Con el desarrollo de la tecnología, han surgido protocolos de comunicación que han sido diseñados teniendo en cuenta las necesidades del IoT, y que se están haciendo un hueco en la funcionalidad de los productos ofertados en la actualidad en esta materia.

En este punto pasamos a describir las principales características de los protocolos más utilizados para la comunicación entre dispositivos del Internet de las Cosas.

2.5.1 Zigbee

Es un conjunto de protocolos de comunicación inalámbrica de bajo consumo basado en el estándar IEEE 802.15.4, encargado de definir el nivel físico y control de acceso al medio de comunicación en redes inalámbricas de uso personal.

Cuenta con características como el bajo consumo, debido a su baja tasa de transmisión, una topología de red en malla y un alcance de la transmisión de entre 10 y 75 metros dependiendo del entorno. Debido a este corto alcance, ha sido comparado en muchas ocasiones con Bluetooth, aunque cuentan con diversas diferencias, como la

cantidad de nodos posibles en una misma red (65535 para ZigBee y 8 en Bluetooth), el consumo eléctrico, que es menor en ZigBee o la velocidad de transmisión.

Esta última diferencia, la velocidad de transmisión de la información, que en ZigBee es de 250 kbit/s y en Bluetooth de 3000 kbit/s hace que el primero sea utilizado en productos domóticos y aquellos que requieran un consumo de batería muy bajo.

En una red ZigBee pueden existir hasta tres tipos de nodos:

- **Coordinador**: Es el encargado de controlar el tráfico de la red, estableciendo los caminos que deben seguir los dispositivos para conectarse entre ellos.

- **Router**: Realiza la interconexión de dispositivos separados en la topología de la red, y ofrece un nivel de aplicación para la ejecución de código de usuario.

- **Dispositivo final**: Es capaz de comunicarse con el router pero no con otros dispositivos. De esta manera, puede mantenerse en estado de reposo durante largos períodos de tiempo y ahorrar energía.

2.5.2 LoraWan

Al contrario que ZigBee, LoraWan es un protocolo de red que hace uso de la tecnología Lora para realizar envíos de mensajes a largas distancias englobándose en el grupo de redes de área amplia y bajo consumo o *low-power wide area networks (LPWANs* en inglés*)*, operando en frecuencias sub-GHz pudiendo alcanzar así distancias entre 10 y 15 km.

LoraWan se compone de dos tipos de dispositivos, *Gateways* y Nodos:

- ***Gateway***: un *gateway* en la terminología LoraWan es el dispositivo encargado de dirigir mensajes hacia los nodos y de recibir los que estos envían. Los nodos se conectan a él mediante un conjunto de *keys* de seguridad para asegurar la identidad del mismo. Puede soportar hasta 62.500 nodos conectados simultáneamente.

- **Nodo**: Son los dispositivos finales interesados en enviar y recibir mensajes de los *gateways*. Existen tres tipos de nodo en función del modo en el que funcionan:

    - Nodo tipo A: permite el mayor ahorro de batería, ya que sólo se encuentra a la espera de recibir mensajes desde el *gateway* cuando le han enviado datos, pasando el resto del tiempo en modo reposo.

    - Nodo tipo B: además de mantenerse a la espera de recibir un mensaje tras haber enviado uno, este tipo de nodos también abre ventanas de

recepción en determinados momentos, activándose y quedando a la espera de mensajes enviados desde el *gateway*. De momento es el modo de funcionamiento menos extendido.

- Nodo tipo C: este tipo de nodos son los que ofrecen menor ahorro de energía, por lo que suelen ser utilizados en dispositivos que cuentan con una fuente de alimentación ilimitada. Consumen más energía ya que se encuentran siempre en modo escucha, solamente saliendo de él cuando requieren transmitir un mensaje al *gateway.*

Debido a sus características, LoraWan es utilizado normalmente en aplicaciones que requieren de un gran ahorro de energía, una baja tasa de envío de datos y que además necesitan comunicarse con otros dispositivos que se encuentran a distancias grandes.

2.5.3 WiFi

WiFi es el protocolo de red inalámbrica más conocido y extendido con diferencia, ya que es el utilizado para las redes inalámbricas domésticas. Está basado en el protocolo IEEE 802.11 y cuenta con diversas variantes dependiendo del ancho de banda en el que trabajan como IEEE 802.11b, IEEE 802.11g, IEEE 802.11n o IEEE 802.11ac.

Cuenta con varios tipos de dispositivos a través de los cuales realiza las comunicaciones:

- **Puntos de acceso:** los puntos de acceso permiten a los dispositivos conectarse de manera inalámbrica a una red existente.

- **Enrutador inalámbrico:** es el encargado de proveer la conexión a Internet al resto de dispositivos de la red, ya sea de manera inalámbrica o por cable. Suelen contener un punto de acceso y un *switch* en su interior.

- **Repetidor inalámbrico:** permite extender la señal de la red WiFi en aquellos lugares en los que esta ya sea demasiado débil. También pueden funcionar como punto de acceso.

- **Dispositivo terminal:** actual como el cliente del protocolo. Se encuentra conectado a un punto de acceso con el que intercambia mensajes para poder obtener una conexión a la red y enviar y recibir mensajes de Internet.

Debido a su corto alcance, WiFi es utilizado en entornos domésticos y espacios cerrados, pudiendo proveer de altas velocidades de conexión y una cobertura aceptable.

3. Smart Cities

El término *Smart Cities* o ciudades inteligentes se refiere al uso y aplicación de las tecnologías de la información y las comunicaciones para la mejora de la calidad de vida de los habitantes de una ciudad, su eficiencia energética y el fomento de la participación ciudadana. Estas mejoras se consiguen mediante la automatización de diversos procesos y de los servicios ofrecidos por la ciudad.

Esta gestión más eficiente y que tiene en cuenta a las personas que viven en la ciudad es extremadamente importante en el siglo XXI. Julio de 2007 fue una fecha histórica: **por primera vez en la historia, la población que vive en las ciudades superó a aquella que lo hace en zonas rurales**. Además, está previsto que en 2050, el porcentaje de población mundial que habite en una ciudad sea del 70%.

Un término muy en boga actualmente es el de la “España despoblada”, y es que en nuestro país, en el 53% del territorio sólamente vive un 5% de la población, siendo la densidad de población de 12,5 habitantes por kilómetro cuadrado. De estos datos podemos deducir que el 95% de la población vive en el 47% del territorio, con una densidad de población de 91,95 habitantes por kilómetro cuadrado según datos del INE en el 2016.

La gran mayoría de las personas en nuestro país y en todo el mundo habita en las ciudades, lo que supone un gran reto para su administración, provisión de servicios y garantía de la calidad de vida de las personas que viven en ellas.

Las *Smart Cities* suponen un cambio en el paradigma de desarrollo urbanístico de las ciudades del mundo. Durante el siglo XX, este desarrollo se basó en la alta densidad de edificios de gran altura y a la extensión incontrolada de las ciudades destruyendo el territorio a su alrededor, reduciendo de esta manera las superficies cultivables y parajes naturales. El nuevo paradigma urbanístico que presentan las ciudades inteligentes se basa en la utilización de las redes de alta velocidad, sensorización, automatización de servicios y el uso de sistemas informáticos para favorecer la comodidad de los habitantes y la gestión eficiente de los recursos (Hall, 1988).

Mediante el uso de la tecnología, una ciudad puede convertirse en:

- **Interactiva:** permitiendo a los usuarios visualizar su estado actual y en ciertos casos actuar sobre ella.
- **Configurable:** la ciudad permite cambiar algunos de sus aspectos para mejorar su eficiencia en casos excepcionales.

- **Innovadora:** una ciudad inteligente permite que en ella se creen nuevos modelos de negocio, favoreciendo la creación de empleo y el desarrollo de la investigación.

- **Flexible:** es capaz de responder en tiempo real a eventos inesperados y aprender sobre ellos para futuras ocasiones.

- **Eficiente y de calidad:** los servicios que ya se ofrecían pasan a realizarse aprovechando mejor los recursos y aportando una mayor calidad durante el proceso y en el resultado final.

- **Rentable:** el gasto público dedicado a su gestión y la provisión de servicios puede verse reducido considerablemente debido al aumento en la eficiencia y flexibilidad.

Como es comprensible, no solamente es importante ubicar en la ciudad dispositivos que sean capaces de interactuar con ella, si no que es crucial tratar los datos obtenidos mediante sensorización, encuestas ciudadanas u otros medios para aprender sobre ellos. Es por ello que es muy importante en estos casos el concepto de transparencia y datos abiertos. Estos términos se refieren al libre acceso de los datos derivados de la administración y la propia actividad de una ciudad inteligente.

Si la ciudadanía cuenta con acceso a los datos de la ciudad, será capaz de desarrollar nuevas soluciones útiles para la misma, además de sentirse integrada en ella al encontrarse completamente informada sobre las decisiones que se toman.

3.1 Ámbitos de aplicación

Ya en la actualidad, una ciudad cuenta con una serie de servicios que debe ofrecer obligatoriamente a sus ciudadanos dependiendo de la cantidad de habitantes que tenga. Estas obligaciones se recogen en el **Artículo 26 de la Ley Reguladora de las Bases del Régimen Local (detalle de los entes locales municipales)** y especifican el conjunto de servicios que deben ser provistos con la intención de asegurar la calidad de vida de quienes habitan en ella.

De esta manera, todos y cada uno de los municipios españoles, independientemente de su tamaño deben asegurar que sus ciudadanos cuenten con: Alumbrado público, cementerio, recogida de residuos, limpieza viaria, abastecimiento domiciliario de agua potable, alcantarillado, acceso a los núcleos de población, pavimentación de las vías públicas y control sobre alimentos y bebidas.

Un municipio con una población mayor a 5.000 habitantes deberá, además de los servicios citados anteriormente, contar con: parques y bibliotecas públicas, mercados y tratamiento de residuos.

Si el municipio sobrepasa el número de 20.000 habitantes, deberá además contar con servicios de: protección civil, prestación de servicios sociales, medios para la prevención y extinción de incendios e instalaciones deportivas de uso público.

En la última categoría, se encuentran los municipios de más de 50.000 habitantes, los cuales deben además de todos los anteriores, proveer a sus ciudadanos de un servicio de transporte público y asegurar la protección del medio ambiente.

Actualmente en España, el gasto medio asociado a la provisión de los servicios mencionados es de 1.221 € por habitante y año, cantidad que puede ser reducida mediante la gestión eficiente de los recursos y la optimización de procesos a través de la utilización de la tecnología y el análisis de datos.

3.1.1 Transporte público urbano

Como se ha comentado anteriormente, un municipio de más de 50.000 habitantes debe ofrecer a sus ciudadanos un servicio de transporte público según lo dispuesto en **la Ley Reguladora de las Bases del Régimen Local.** Este servicio puede tratarse de líneas de autobús urbano, metro subterráneo, tranvía o cualquier otro sistema que permita el desplazamiento colectivo de los ciudadanos dentro del área urbana de la ciudad.

Hoy en día, no imaginamos las ciudades sin transporte público, mediante el cual se desplazan miles de personas al día y que permite reducir las emisiones de carbono de manera notable, tendiendo este tipo de transporte a la utilización de energías limpias, reduciendo atascos y ofreciendo un servicio con un precio moderado.

Sólamente en Ferrocarriles de la Generalitat Valenciana (FGV), encargada de la gestión de MetroValencia y el TRAM de Alicante, la cifra de usuarios en el año 2017 fue de 74,34 millones de viajeros. Esta cifra permite visualizar la cantidad de desplazamientos en coche que se consiguen evitar mediante el uso del transporte público, teniendo en cuenta que no son los únicos medios de transporte en las ciudades citadas.

El uso de la tecnología en el ámbito del transporte público puede suponer un ahorro tanto económico como energético, además de un aumento en la eficiencia y la calidad de las prestaciones del servicio que pueden derivar por ejemplo en una mayor puntualidad de los transportes, reducción de la energía utilizada tanto en las estaciones o paradas como en los propios vehículos, o la mayor información sobre el estado actual de las líneas a los pasajeros.

Ya es una realidad poder visualizar en pantallas la hora exacta a la que va a llegar el próximo tren o autobús, actualizándose esta información dinámicamente en función del tráfico de la ciudad, imprevistos surgidos en la circulación o el número de personas que se encuentran utilizando ese medio de transporte en ese instante, ya que

su subida y bajada del mismo pueden retrasar los horarios preestablecidos si el número de usuarios es muy alto.

En este último caso, están surgiendo nuevos casos de uso en la ciudad de Barcelona. En ella, desde hace dos años, se está midiendo el nivel de ocupación de los vagones de cada uno de los metros para ajustar la frecuencia de paso de los convoyes, siendo esta mayor en horas punta y menor en horas con baja afluencia de personas. Se prevé que a principios de 2020, además de ser utilizado para calcular la frecuencia con la que deben pasar trenes por las estaciones, esta tecnología será utilizada para informar a los usuarios sobre el nivel de ocupación de cada uno de los vagones, lo que ayudará a distribuir de una forma más eficiente a los viajeros. Este cálculo se realiza mediante sensores ubicados en las amortiguaciones de los vagones, las cuales son capaces de medir el peso soportado en cada momento por los mismos.

Esta información se mostrará en los paneles de las estaciones mediante un código de colores (verde, amarillo y rojo) que indicarán el porcentaje de ocupación de cada uno de los vagones, permitiendo así que los usuarios que esperan el tren se distribuyan en función de estos datos.

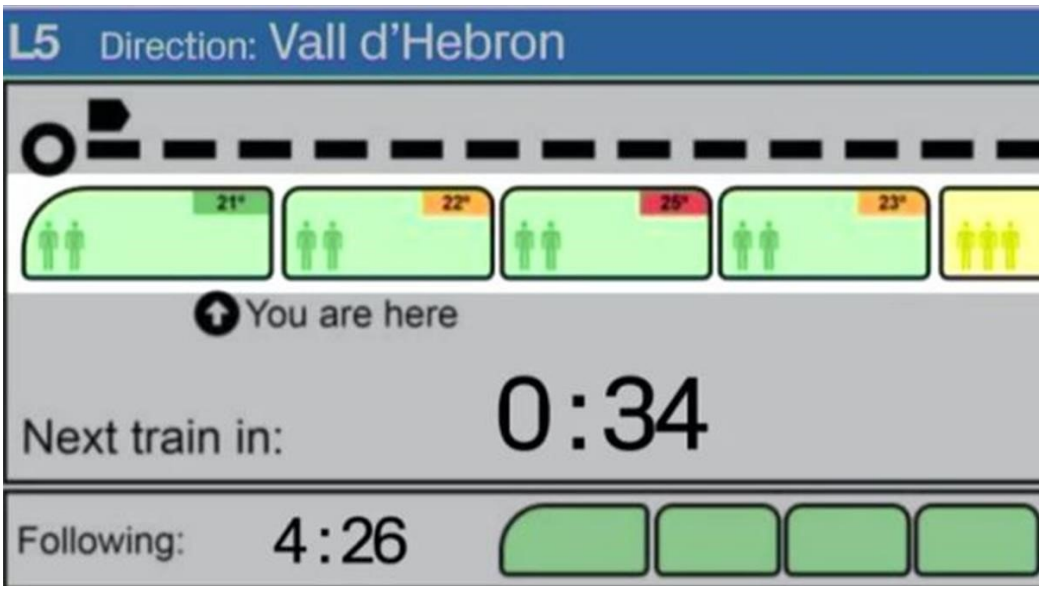

Imagen 3.1. Prototipo del panel de información que muestra el nivel de ocupación de los vagones

El caso más claro de la aplicación de la tecnología en el transporte público es el metro de Singapur, el cual ya mide y muestra la información a los usuarios sobre la carga de los vagones, además de ofrecer datos sobre la humedad y temperatura del interior de cada uno de ellos. Además, parte de sus líneas están compuestas por trenes conducidos automáticamente y ofrece servicios de conexión a Internet 4G y WiFi tanto en las estaciones como en los vagones.

El desarrollo e inclusión de tecnologías específicas para el transporte público puede mejorar la eficiencia del servicio incurriendo en la reducción de costes asociados a su provisión y la mejora en la calidad del mismo en beneficio de los usuarios, haciendo que el número de personas que lo utilizan pueda aumentar considerablemente debido a la comodidad de hacerlo. Es por ello que es un ámbito en el que las administraciones locales están invirtiendo cada vez más recursos tanto materiales como de investigación.

3.1.2 Mejora medioambiental

Uno de los problemas que más preocupan en la actualidad a los habitantes de las ciudades alrededor de todo el mundo es el de la contaminación ambiental. Esta surge mayoritariamente del tráfico que atraviesa la ciudad diariamente. Las emisiones de los vehículos se ven aumentadas en situaciones de tráfico lento y atascos.

Esta preocupación ha derivado en los sistemas de medición de calidad del aire, que se encuentran implantados en la mayoría de las ciudades, y que son los encargados de proporcionar información en tiempo real a la administración local sobre la cantidad de partículas nocivas en suspensión que se encuentran en el aire en un momento determinado.

Estos sistemas permiten conocer la situación en cada momento, así como el histórico recogido desde la implantación del sistema, pudiendo actuar sobre la ciudad para intentar reducir la contaminación en la misma.

Esta gestión del tráfico en función de la contaminación del aire en una ciudad ha sido un tema muy en boga en los últimos meses en ciudades europeas como París o Madrid. En ambas se han aplicado restricciones al tráfico en épocas en las que las mediciones de la calidad del aire ha sido alarmante.

Imagen 3.2. Distintivo ambiental que certifica las bajas emisiones de un vehículo

Para la disminución de las emisiones en algunas ciudades se han tomado las siguientes medidas:

- Restricciones al tráfico permitiendo sólamente el paso a determinadas zonas a vehículos con matrícula par o impar.

- Restricciones al tráfico permitiendo el paso a vehículos de transporte público o eléctricos además de aquellos que prestan algún servicio como los de reparto.

- Sólo permitir el paso a vehículos a partir de un determinado distintivo ambiental, que certifica que sus emisiones son bajas.

Además de la actuación sobre el tráfico de vehículos en la ciudad, las ciudades inteligentes utilizan la tecnología para mejorar la eficiencia en la gestión de los residuos generados por la misma incluyendo su recogida y tratamiento. De esta manera, se puede obtener una forma más eficiente de tratar los residuos, aumentando su porcentaje de reutilización y la prevención de filtraciones y vertidos en fuentes de agua potable.

3.1.3 Participación ciudadana y administración electrónica

Como ya se ha comentado anteriormente, uno de los puntos imprescindibles para considerar una ciudad como inteligente es que sus ciudadanos participen en las decisiones administrativas y políticas que se tomen en ella en el día a día, ya sea mediante consultas electrónicamente, encuestas por Internet o cualquier otro medio disponible que haga uso de las nuevas tecnologías.

Para que los ciudadanos puedan decidir su postura ante las consultas realizadas por el gobierno municipal, deben contar con toda la información posible para poder decidir. De esta necesidad surge el término “transparencia política”, consistente en la disposición de la administración pública a la apertura de la información generada por su gestión durante un periodo de tiempo.

La transparencia puede aplicarse a diferentes aspectos de la actividad política como pueden ser:

- **Inversión financiera:** se refiere a la publicación de la utilización de los recursos económicos públicos, detallando qué producto o servicio ha sido obtenido y la inversión realizada para ello.

- **Retribución cargos políticos:** tablas retributivas de los altos cargos de la presidencia de un estado o región y de los municipios.

- **Obsequios recibidos:** los cargos políticos suelen recibir obsequios durante sus visitas y actos, los cuales son publicados indicando quién ha realizado el obsequio, una descripción y si procede, su valor monetario.

- **Publicación decisiones tomadas:** es habitual en la mayoría de las democracias actuales que las decisiones que toman los equipos de gobierno en el ejercicio de sus funciones sean publicadas en boletines. Con las nuevas tecnologías, estos boletines se publican también en Internet, permitiendo una mayor accesibilidad y disponibilidad para los ciudadanos.

Además de en la publicación de la información relativa a la administración pública y política, la tecnología puede ser utilizada para fomentar la participación ciudadana mediante sistemas de voto electrónico, que permitan consultar a la población su postura ante propuestas realizadas por los gobiernos. También permite la recepción de propuestas y demandas sobre inquietudes de los ciudadanos, pudiendo estas evolucionar en legislación en el futuro.

Otro aspecto ya consolidado actualmente es el de la administración electrónica. Este término se refiere de manera general a la utilización de las tecnologías de la información y las comunicaciones en los procesos desarrollados por las administraciones públicas. Debido a este sentido tan amplio del término, se puede enfocar la administración electrónica en dos subtérminos más diferenciados:

- **Uso intraorganizativo:** se refiere a la digitalización de los procesos internos desarrollados en las administraciones públicas que anteriormente se realizaban en papel. De esta manera se consigue acelerar los mismos, facilitar la trazabilidad de los documentos y el ahorro en materiales de oficina.

- **Relaciones con el exterior:** este aspecto de la administración electrónica es el que afecta directamente a la mayoría de los ciudadanos y empresas. Se refiere a la habilitación de la posibilidad de utilizar las nuevas tecnologías para realizar los trámites burocráticos con la administración pública. De esta manera se consigue aumentar la agilidad y comodidad de las relaciones. El ciudadano suele identificarse ante la administración pública mediante algún tipo de certificado electrónico para asegurar su identidad personal.

Como se puede observar, la utilización de las nuevas tecnologías en el gobierno y administración de los municipios, regiones, países, e incluso a nivel europeo, permite agilizar los procesos de las administraciones públicas. Además, la transparencia en los datos y el fomento de la participación ciudadana, provocan que los habitantes se sientan más cercanos a las decisiones que se toman en el lugar en el que viven, aumentando su interés en las mismas.

Esta dinamización permite el ahorro de recursos económicos y el aumento de satisfacción de todos aquellos ciudadanos que entran en contacto con los gobiernos y administraciones públicas.

3.1.4 Educación electrónica

La educación electrónica, también denominada *e-learning* aprovecha las ventajas proporcionadas por las tecnologías de la información para mejorar la educación y culturización de las personas.

En el sentido más estricto del término, puede referirse al uso de las nuevas tecnologías en las aulas para mejorar el modo en el que interactúan los alumnos entre ellos o con los profesores. De este modo, se complementa la enseñanza habitual a partir de libros de texto, clase magistral y pizarra con elementos de vídeo, audio, infografías o cualquier elemento generado mediante el uso de un ordenador o dispositivo electrónico.

Mediante estos recursos, el profesor puede añadir información sobre el tema que está siendo tratado y presentarlo de manera más visual o descriptiva, reforzando de esta manera el aprendizaje.

Además de en la enseñanza tradicional, las nuevas tecnologías pueden servir para culturizar a la población o turistas de una ciudad en aspectos históricos o monumentales de la misma. De esta manera, mediante aplicaciones de realidad aumentada, audioguías, etc, quien la visita puede recibir información a medida que visita ciertos lugares, sirviendo esta como complemento a los clásicos paneles informativos o guías y folletos impresos en papel.

Imagen 3.3. Aplicación de realidad aumentada que muestra información sobre el Museo del Prado de Madrid

3.2 Caso de estudio: la ciudad de Valencia

Una vez comentados algunos de los aspectos en los que las nuevas tecnologías pueden ser útiles en las ciudades modernas para el aumento de la eficiencia en los procesos y la comodidad de los ciudadanos que habitan en ellas, podemos pasar a ver el ejemplo de una ciudad en concreto que ya utiliza algunos de estos conceptos en su día a día: Valencia.

Con una población de casi 800.000 personas según datos del INE en el año 2018, la ciudad de Valencia se sitúa como la tercera ciudad más grande de España en número de habitantes, sólo por detrás de Madrid y Barcelona. Cuenta con una superficie de 134,65km^2 y una densidad de población de 5.858,78 habitantes por kilómetro cuadrado.

El inicio de la andadura de la ciudad de Valencia como ciudad inteligente puede marcarse en el año 2014, cuando su gobierno municipal decidió lanzar la plataforma VLCi, la cual comenzó a actuar como cuadro de mandos y sistema de gestión de los dispositivos electrónicos y de sensorización que se encontraban ubicados en la ciudad a partir del año 2015. De esta manera, se permitió a usuarios autorizados comenzar a gestionar la ciudad informáticamente y visualizar estadísticas de los datos recogidos. Además, en este mismo año, se creó el Portal de transparencia y datos abiertos, accesible a cualquier ciudadano que lo deseara, permitiendo realizar consultas sobre datos de sensores ubicados en la ciudad como estaciones de medición de ruido o contaminación y datos económicos y de gestión de la administración municipal.

Desde el 2014, la ciudad de Valencia ha ido dotando de “inteligencia” a los diferentes servicios ofrecidos. En el año 2016, integró el servicio de autobuses públicos de la EMT, sensorizándolo en mayor medida y mejorando la información mostrada a los usuarios tanto a través de las marquesinas de las paradas como de su aplicación móvil.

En 2017, se integra también a la plataforma de ciudad inteligente de Valencia el servicio de alumbrado público, el cual comenzó a recoger datos sobre los que aprender para mejorar su eficiencia en el futuro. En este mismo año también se integra el servicio de parkings públicos, el cual publica datos sobre su estado de capacidad tanto en las tradicionales señales ubicadas por la ciudad como en los portales de información web de la plataforma de ciudad inteligente de valencia.

Las últimas incorporaciones a la plataforma de ciudad inteligente han sido las de los servicios de coordinación de obras en la vía pública y mantenimiento de infraestructuras. Además, también se integró el servicio de jardinería municipal, que aporta el registro de árboles monumentales de la ciudad y el registro de árboles y plantas florales para establecer un mapa de afectación para alérgicos al polen.

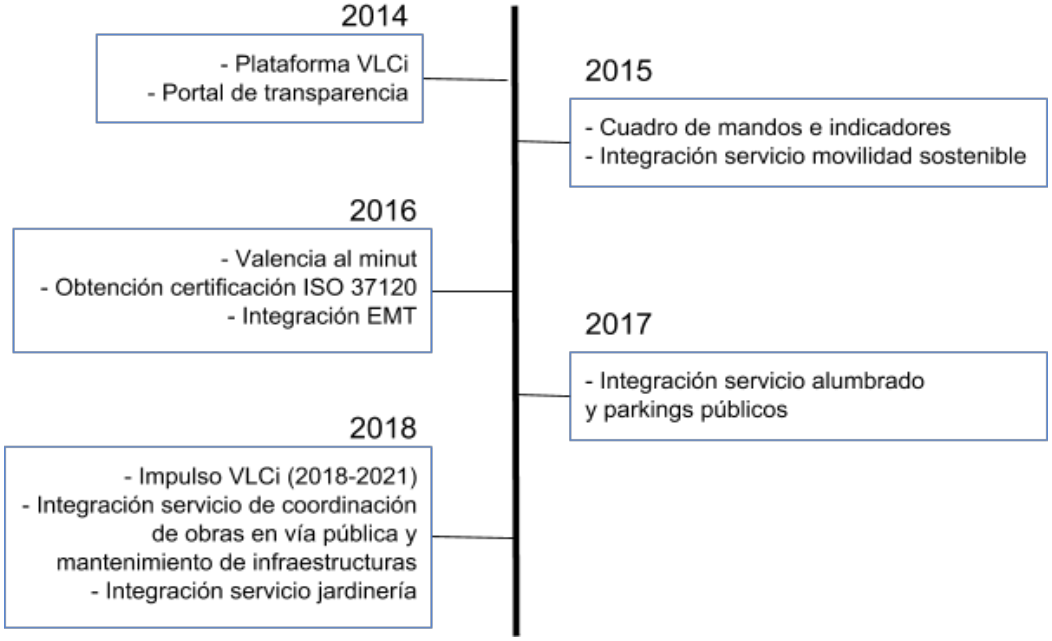

Imagen 3.4. Historia Valencia como ciudad inteligente

3.2.1 València al minut

El proyecto València al minut consiste en un portal de información al servicio de los ciudadanos para su acceso libre que permite visualizar datos sobre el estado del tránsito, de los aparcamientos públicos, el transporte urbano, etc. Toda esta información se muestra en tiempo real para proporcionar al usuario una visión global de lo que está ocurriendo en la ciudad en un determinado momento con sólo unos *clicks*.

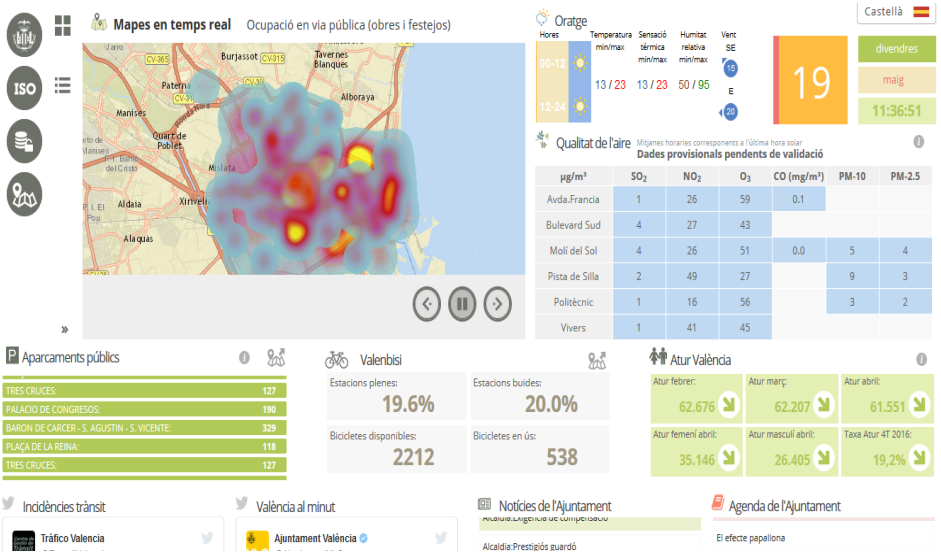

Imagen 3.5. Pantalla de inicio de València al minut

3.2.2 Geoportal

El proyecto Geoportal proporciona información al usuario sobre el estado de la ciudad a través de mapas interactivos. Se incluye información entre los que se encuentran aspectos de movilidad, como pueden ser estado del tráfico en tiempo real, cortes de vías por actos festivos, localización de puntos de interés, zonas tributarias o información administrativa como el catastro o las secciones censales.

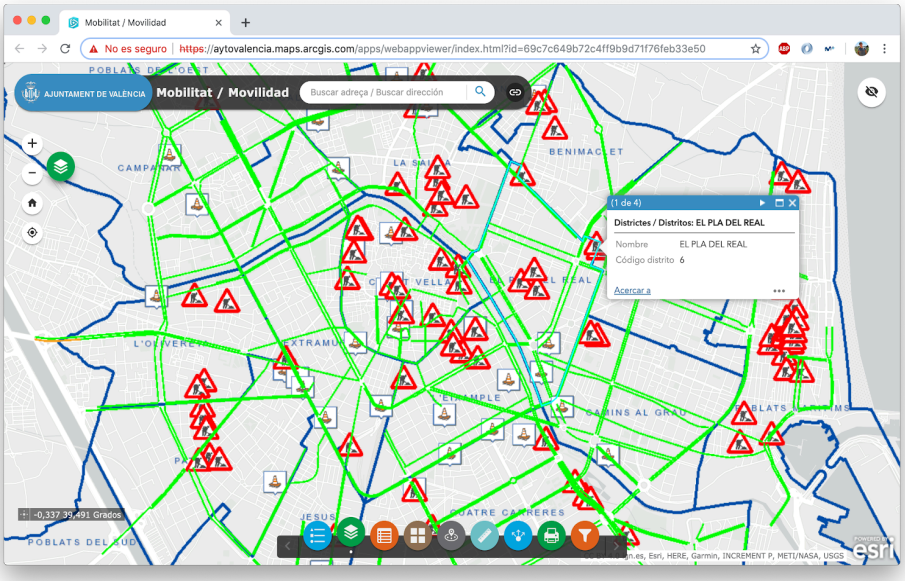

Imagen 3.6. Mapa que muestra las ocupaciones de vía pública por obras en las ciudad en tiempo real

La información mostrada en los mapas puede ser además ampliada pinchando en el elemento deseado, obteniendo de esta manera una breve descripción del punto de interés que varía en función de la información consultada.

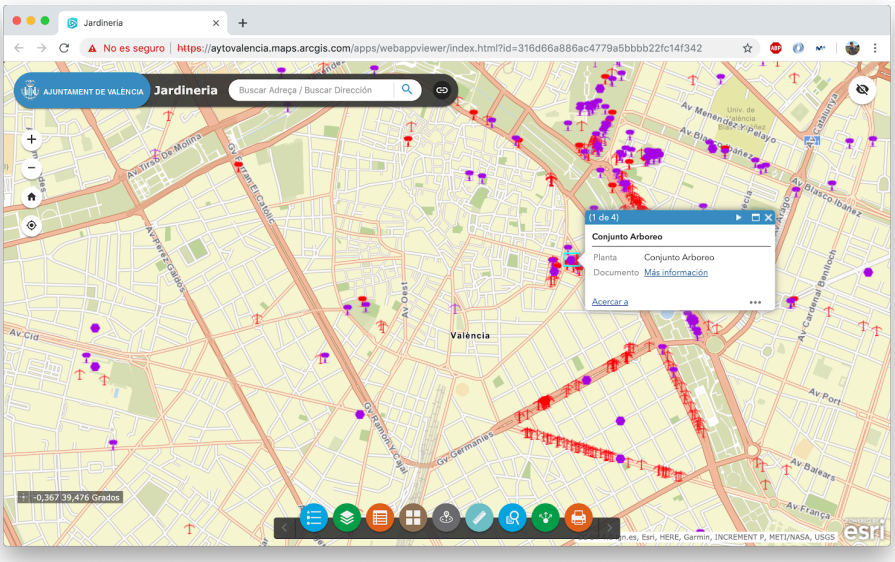

Imagen 3.7. Detalle árboles monumentales Geoportal

3.2.3 AppValencia

AppValencia es una aplicación móvil cuyo objetivo es acercar al ciudadano de Valencia a su ayuntamiento, permitiendo realizar acciones administrativas desde el *smartphone* a través del portal de la Sede Electrónica Municipal, convirtiéndose en una solución móvil de administración electrónica.

Además, la aplicación permite visualizar el estado actual de la ciudad, como los horarios de los servicios de transporte público, y recibir alertas o notificaciones que pueden ser útiles para el usuario.

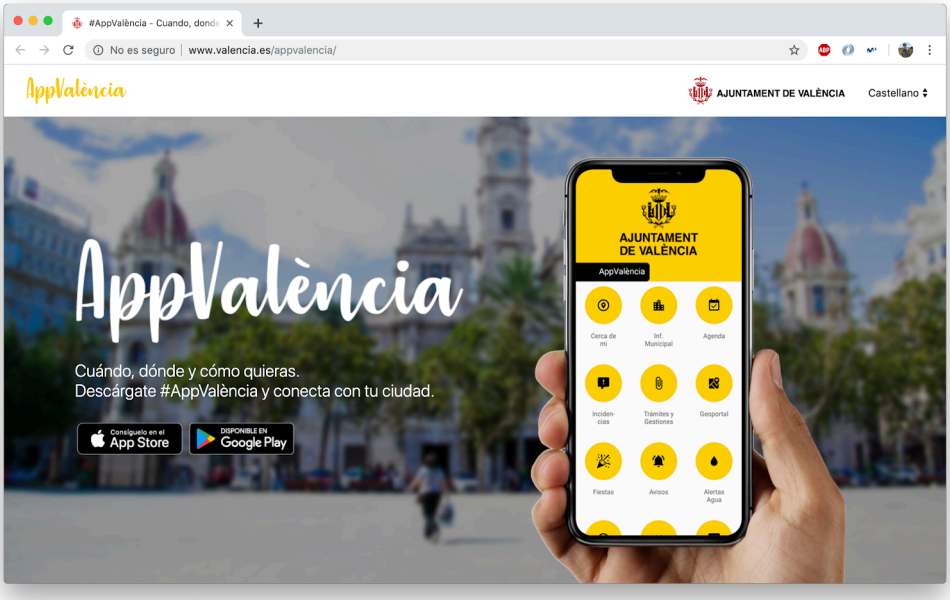

Imagen 3.8. Página web AppValència

Con estas herramientas, la ciudad de Valencia utiliza los datos que generan los sensores ubicados por toda su extensión para mostrarlos a la ciudadanía y mejorar los servicios que le ofrece, permitiendo así la mejora de la calidad y la satisfacción de habitantes y visitantes del municipio.

Además de las soluciones ya implantadas, existen multitud de proyectos que se encuentran en proceso como:

- **Instalación de alumbrado inteligente:** utilizará los datos recogidos desde la integración del alumbrado público a la plataforma VLCi y los sensores ubicados en las farolas para apagar y encender el alumbrado dependiendo de la situación, con el consiguiente ahorro energético y económico generado.

- **Gestión de residuos sólidos y limpieza:** sensorización de los vehículos de limpieza y mantenimiento de la ciudad para una gestión más eficiente de la

misma y la comunicación entre equipos, que permitirá una respuesta más ágil a situaciones de limpieza no programadas.

- **Piloto sensores mercado de Ruzafa:** proyecto de sensorización del histórico mercado de Ruzafa para una gestión más eficiente y moderna del mismo. Se miden factores como la humedad, iluminación, aforo, ruido, etc.

- **Gestión del ruido Plaza del Tossal:** este proyecto piloto relaciona el ruido medido en la plaza con la intensidad lumínica de su alumbrado. De esta manera, si el ruido aumenta se disminuye la intensidad de la luz, permitiendo atenuar así el ruido generado por las personas que se encuentran en ella.

4. Protocolo de comunicación propuesto

En este trabajo se presenta el diseño e implementación de un **protocolo de comunicación** entre dispositivos inteligentes en una Smart City que resuelva la problemática del intercambio de datos. Este protocolo requerirá de al menos **dos agentes**, uno que actuará como Punto de Acceso y otro que lo hará como cliente de los servicios que ofrezca la baliza digital que actúa como servidor.

Se ha diseñado el protocolo de manera que pueda ser utilizado en diferentes dispositivos e implementado en cualquier lenguaje de programación. Además, es **independiente del entorno** y de la tarea que la baliza esté realizando en ese momento, lo que aporta una mayor flexibilidad y la posibilidad de que esta cambie el servicio que ofrece con una simple configuración de su funcionamiento. Es por ello que podría ser aplicado en cualquiera de los ámbitos y aplicaciones de las Smart Cities mencionados anteriormente.

En puntos siguientes de este documento se describe el diseño del protocolo así como las tecnologías utilizadas para implementarlo y poder crear un prototipo usable del mismo.

4.1 Análisis del problema

Antes de pasar a diseñar el protocolo como tal, se debe establecer cuál es la problemática que se desea solucionar con él y la forma de realizarlo.

En una ciudad en la que miles o incluso millones de dispositivos se encuentren conectados a Internet y deseen comunicarse con su entorno, guardar una relación de todos los posibles vecinos con los que podría ser interesante contactar en algún momento es inviable, ya que la cantidad de información podría ser imposible de almacenar teniendo en cuenta la pequeña capacidad de los dispositivos del Internet de las Cosas.

Es por ello que deberemos diseñar un protocolo que permita a los dispositivos **descubrir por su cuenta aquellos vecinos que le pueden ser útiles** para obtener una información determinada. Esto además debe poder realizarse en el momento en el que se vaya a consultar la información, no necesitando por lo tanto almacenar ninguna información sobre otros elementos de la red.

También es importante tener en cuenta que el protocolo deberá ser ligero, haciendo uso de un pequeño conjunto de mensajes y no requiriendo demasiado intercambio de información para lograr su cometido. De esta manera, en ningún

momento se saturará la red ni se impedirá el funcionamiento del resto de funcionalidades que el dispositivo deba desempeñar.

El dispositivo no deberá quedar bloqueado nunca a la espera de recibir confirmaciones ni mensajes de ningún tipo para no entorpecer el resto de funcionalidades. Es por ello que deberán definirse *timeouts* pasados los cuales se abortará el proceso de comunicación.

4.1.1 Fases de la comunicación

Para poder realizar las funciones para las que ha sido diseñado el protocolo correctamente teniendo en cuenta los detalles mencionados anteriormente, se creen necesarias dos fases en el proceso de comunicación:

- **Descubrimiento:** en esta fase, el dispositivo cliente buscará los vecinos más cercanos que presten servicios que le puedan ser útiles para la función que desempeña

- **Obtención de información:** una vez elegido el vecino más óptimo para realizar la comunicación, se construyen mensajes utilizados para consultarle la información requerida.

Por lo tanto, teniendo en cuenta los detalles mencionados anteriormente y las fases que se creen necesarias para poder realizar una comunicación satisfactoria, podemos pasar a describir cómo se ha diseñado el protocolo objetivo principal de este trabajo de fin de máster.

4.2 Diseño del protocolo

Como se ha comentado en la introducción de este capítulo del trabajo, existirá un dispositivo que actuará como servidor en la comunicación. De ahora en adelante este será denominado punto de acceso o AP.

4.2.1 Descubrimiento de vecinos

Al iniciar los dispositivos, el AP quedará en un bucle infinito a la espera de recibir mensajes por parte de algún cliente que se encuentre cerca de él y requiera información.

En el caso de los dispositivos que actúan como cliente, enviarán en intervalos de tiempo configurable un mensaje de descubrimiento denominado IS_AP, cuyo cometido será encontrar el punto de acceso más cercano a él que contenga la información en la que está interesado.

La frecuencia en la que los dispositivos clientes realicen una búsqueda de puntos de acceso será variable dependiendo del caso de uso en el que nos encontremos, así, un coche recorriendo una ciudad deberá contar con una frecuencia de consulta de puntos de acceso cercanos alta, ya que de ello dependerá que por ejemplo se pare en un semáforo o no. Por el contrario, si el caso de uso fuera un punto de acceso que distribuya publicidad en formato electrónico a las personas que pasan cerca de él, los clientes de este servicio deberán tener una frecuencia de consulta menor, ya que normalmente pasarán andando y no será crítico que consulten la información suministrada por el punto de acceso un segundo antes o después.

Este tipo de mensajes serán difundidos mediante un *broadcast* a todos los dispositivos conectados a la misma red que él. Será posteriormente cuando el cliente elija qué punto de acceso le es más útil tanto por proximidad como por la funcionalidad que desempeña.

Un mensaje de tipo IS_AP contará en su estructura con el tipo de mensaje enviado, información del cliente que realiza la consulta y sobre qué temática (*topic*) desea preguntar al AP. De manera visual, un mensaje IS_AP tiene la siguiente estructura:

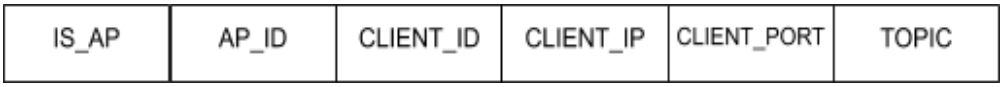

Imagen 4.1. Esquema de un mensaje IS_AP

Los mensajes serán enviados en formato JSON (definido en RFC 4627), útil para que estos sean legibles y fácilmente tratables en el código de la aplicación. Un mensaje de tipo IS_AP tendrá la siguiente estructura JSON una vez enviado desde un cliente:

Imagen 4.2. Estructura JSON de un mensaje IS_AP

Cuando un punto de acceso reciba un mensaje de tipo IS_AP, comprobará si el topic sobre el que quiere preguntar el cliente es igual al que él trata. Si no lo es, descartará el mensaje y no contestará. Si por el contrario el topic solicitado es igual al suyo, este responderá con un mensaje de tipo CONFIRM_AP. También comprobará si el identificador del AP es igual al suyo, ya que si no lo es deberá ignorar el mensaje.

También existe la posibilidad de que este identificador venga vacío, en cuyo caso deberá responder al mensaje.

Un mensaje de tipo CONFIRM_AP contendrá información relativa al emisor del mismo (en este caso el AP), de quien va a recibirlo, el *topic* que trata y dentro de este los *subtopics* sobre los que puede aportar información y las coordenadas en las que se encuentra.

Un *topic* puede contar con uno o varios *subtopics*, así si un AP tiene como función actuar como semáforo, su *topic* podría denominarse "*traffic_lights*". Los *subtopics* serían la información que nos podría ser útil conocer sobre un semáforo, como por ejemplo el estado en el que está, el tiempo que le queda hasta cambiar al siguiente estado o cuantos peatones se encuentran esperando para cruzar la calle. En un ejemplo como este, los *subtopics* podrían denominarse "state", "time_remaining" y "pedestrians".

Con esta información, un mensaje de tipo CONFIRM_AP tendrá la siguiente estructura representada gráficamente:

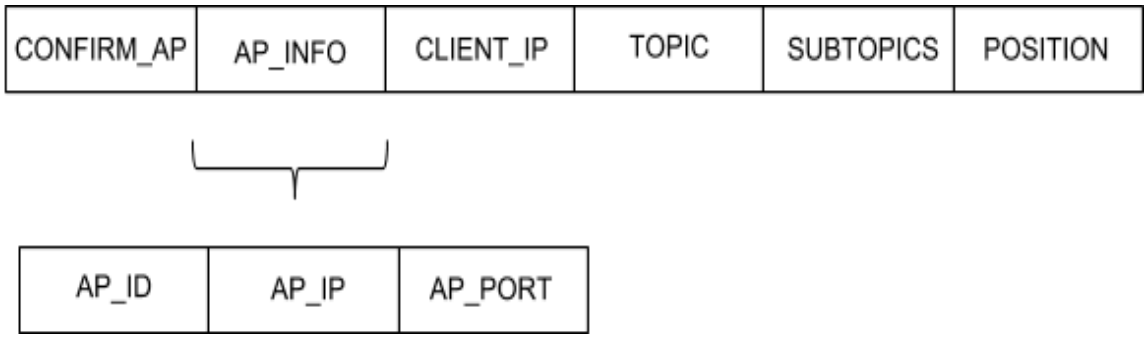

Imagen 4.3. Esquema gráfico de un mensaje CONFIRM_AP

Y esta estructura de mensaje en formato JSON:

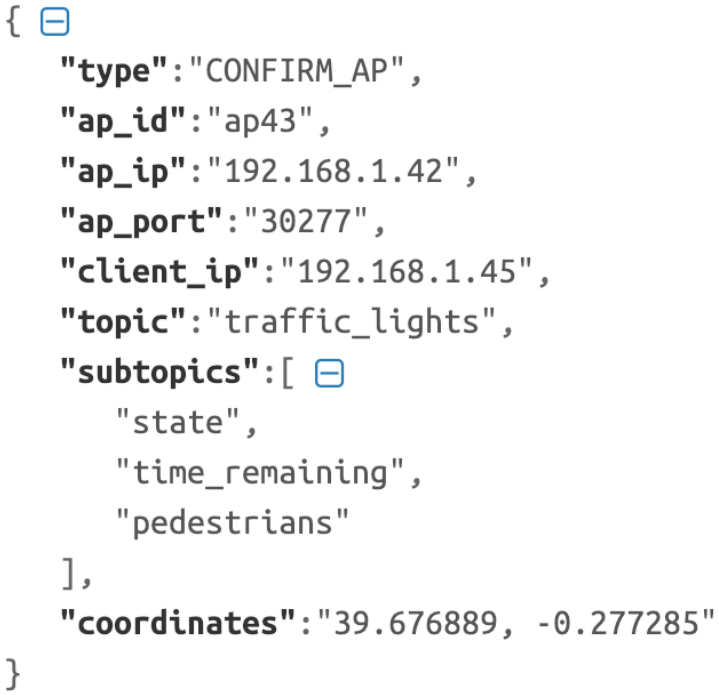

Imagen 4.4. Estructura JSON de un mensaje CONFIRM_AP

Este mensaje ya no requiere ser enviado en *broadcast*, ya que la dirección del cliente interesado es conocida gracias a que se envió a través del mensaje IS_AP del paso anterior. Por lo tanto, este mensaje sólamente será enviado a la dirección del cliente especificada en el campo “client_address”.

Una vez enviado el mensaje CONFIRM_AP, el AP queda a la espera de recibir más mensajes, ya sean de tipo IS_AP de más clientes queriendo descubrir puntos de acceso o peticiones de información por parte de los clientes que ya conocen su dirección y las temáticas que trata.

4.2.2 Petición de información

Una vez el cliente recibe la confirmación de que un dispositivo es un punto de acceso y además trata el topic en el que él está interesado, debe comprobar que la distancia entre ellos no sea demasiado grande, en cuyo caso deberá descartar este AP como viable.

Esto se debe a que en una misma red pueden existir diversos puntos de acceso que reciban el mensaje de descubrimiento de APs y contesten al mismo. Si no se comprueba la distancia a la que están, podríamos estar tratando con un punto de acceso a una gran distancia, de poca utilidad en muchos casos de uso. Por ejemplo, imaginemos el caso en el que un coche quisiera consultar el estado del semáforo de un cruce por el que va a pasar en unos pocos segundos. Si no tenemos en cuenta la

distancia del AP que nos contesta, podríamos estar teniendo en cuenta el estado de un semáforo lejano que ni siquiera corresponde al cruce por el que estamos circulando. Además, esto también podría ser más exacto si existiera un topic para cada tramo de carretera por el que estemos circulando, así sólamente recibiríamos respuesta de los semáforos que se encuentra en nuestra vía, aunque esta solución escapa los dominios del protocolo y entra en los de la implementación del servicio que utiliza el mismo.

Como hemos visto, una forma que tiene el protocolo de asegurar la utilidad de los APs que responden es mediante la comprobación de la distancia entre ellos y el cliente interesado en realizar la conexión. Esto puede realizarse gracias al campo “POSITION” que se encontraba en el mensaje CONFIRM_AP recibido en el paso anterior y la utilización del algoritmo Haversine, que a partir de dos coordenadas cartesianas obtiene la distancia en metros entre ellas.

Una vez comprobado que el AP que ha respondido se encuentra a una distancia prudencial que puede ser útil, el cliente deberá construir un mensaje de tipo GET_INFO cuya utilidad será preguntar sobre los *subtopics* en los que esté interesado.

Un mensaje de tipo GET_INFO cuenta con campos relativos a la identificación de emisor y receptor del mensaje, topic y subtopics sobre los que se quiere preguntar. Así, un mensaje de tipo GET_INFO tendrá la siguiente estructura definida de manera gráfica:

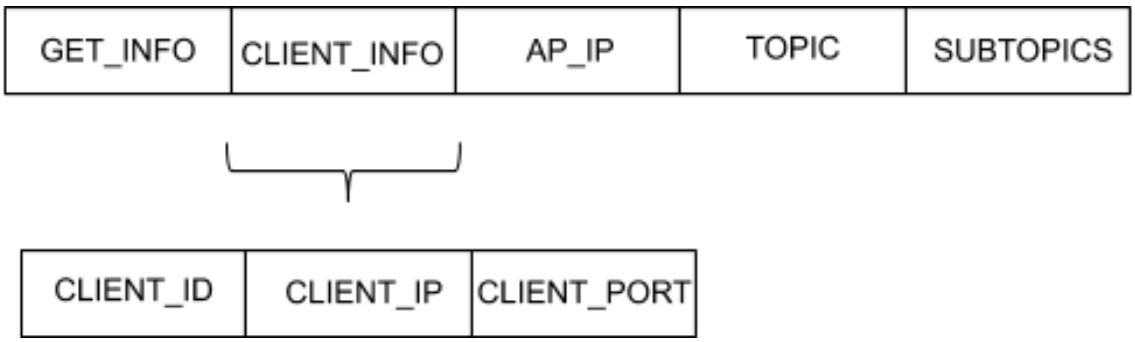

Imagen 4.5. Esquema gráfico de un mensaje GET_INFO

La estructura en formato JSON del mensaje de tipo GET_INFO será:

Imagen 4.6. Estructura JSON de un mensaje GET_INFO

4.2.3 Respuesta a petición de información

Una vez recibido el mensaje de tipo GET_INFO con la petición de información, el AP deberá comprobar que el topic mencionado en el mensaje sea igual al que él trata. Si es así, construirá la respuesta obteniendo la información con la que cuenta sobre los *subtopics* consultados. El mensaje de respuesta, que será de tipo ANS_INFO contará con información relativa a la identificación del receptor y emisor del mensaje, el *topic* sobre el que versa la respuesta y una lista cuyo identificador será el nombre del *subtopic* y los valores serán el estado del subtopic en el momento de la respuesta.

La estructura de un mensaje de tipo ANS_INFO será:

Imagen 4.7. Esquema gráfico de un mensaje ANS_INFO

La estructura en formato JSON del mensaje de tipo ANS_INFO será:


```
{
   "type":"ANS_INFO",
   "ap_id":"ap43",
   "client_id":"lopy327",
   "topic":"traffic_lights",
   "response":{
      "state":"green",
      "time_remaining":20
   }
}
```


Imagen 4.8. Estructura JSON de un mensaje ANS_INFO

En el mensaje mostrado en la Imagen 4.8, todos los *subtopics* por los que ha preguntado el cliente eran correctos y podían ser respondidos por el AP. Pero en el caso en el que el cliente pregunte por un *subtopic* para el que el AP no cuenta con información, el mensaje será respondido igualmente, pero indicando en el resultado de la respuesta que ese *subtopic* no ha podido ser respondido correctamente. Así, si se envía un mensaje de tipo GET_INFO sobre el topic traffic_lights que cuenta con los subtopics: “state” y “time_remaining” en el que se quiere consultar un subtopic de tipo “garbage_load”, se obtendrá una respuesta como la que se muestra a continuación:


```
{
   "type":"ANS_INFO",
   "ap_id":"ap43",
   "client_id":"lopy327",
   "topic":"traffic_lights",
   "response":{
      "state":"red",
      "garbage_load":"Error. Subtopic not available on this context"
   }
}
```


Imagen 4.9. Estructura JSON de un mensaje ANS_INFO con un *subtopic* erróneo

Como se puede observar en la Imagen 4.9, el *subtopic* “state” ha sido respondido pero no así el *subtopic* “garbage_load”, ya que no pertenece al *topic* “traffic_lights”.

Una vez recibido el mensaje ANS_INFO, el cliente ya cuenta con la información que requería del AP, por lo que puede cerrar la conexión. En caso de querer preguntar sobre más topics o hacerlo sobre los mismos de nuevo, deberá volver a realizar el ciclo, volviendo a preguntar por un AP cercano, ya que el protocolo no guarda información alguna sobre identificación de dispositivos.

4.3 Resumen de una comunicación con el protocolo

Una vez comentados tanto el flujo de la comunicación realizada utilizando el protocolo diseñado como la estructura de los mensajes que se utilizarán para ello, se resume gráficamente una comunicación ideal, representando los agentes implicados y los mensajes que intercambian entre ellos:

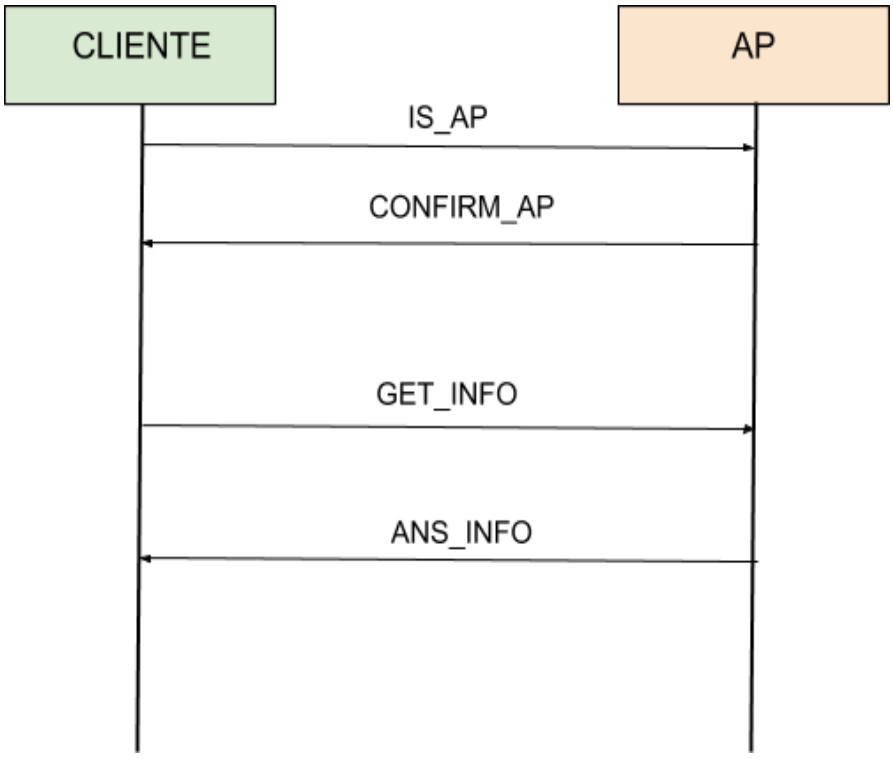

Imagen 4.10. Resumen de una comunicación

4.4 Mensajes de configuración

Además de la comunicación entre los agentes cliente y punto de acceso, se podrán enviar mensajes desde la interfaz web que actuará como cuadro de mandos. Desde ella se podrán configurar cada uno de los AP, haciendo que por ejemplo cambien su estado (un semáforo en verde podría ser cambiado a modo emergencia para que

tuviera preferencia un camión de bomberos). Podrá incluso modificarse también el servicio que el dispositivo brinda a la ciudad.

Un mensaje de configuración será de tipo PUSH_CONFIG, y la identificación del cliente será “web”. Además, el mensaje contará con la dirección a la que enviar la confirmación de que el cambio se ha realizado correctamente (mensaje de tipo CONFIG_OK) y en su cuerpo contará con el topic que se quiere cambiar en el dispositivo y los subtopics con los nuevos valores.


```
{
   "type":"PUSH_CONFIG",
   "client_id":"web",
   "client_ip":"192.168.1.45",
   "client_port":"3223",
   "ap_id":"ap43",
   "topic":"traffic_lights",
   "subtopics":{
      "state":"emergency",
      "time_remaining":100
   }
}
```


Imagen 4.11. Estructura JSON de un mensaje PUSH_CONFIG

Así, este mensaje enviado desde la interfaz web, indicará al AP que se requiere cambiar el estado del semáforo a “emergency” y el tiempo restante en este estado a 100 segundos. Cuando el AP haya realizado los cambios necesarios, deberá notificar a la interfaz web que se ha realizado correctamente mediante un mensaje CONFIG_OK cuya estructura será la siguiente:


```
{
   "type":"CONFIG_OK",
   "ap_id":"ap43",
   "ap_ip":"192.168.1.45",
   "ap_port":"3223",
   "topic":"traffic_lights"
}
```


Imagen 4.12. Estructura JSON de un mensaje CONFIG_OK

4.5 Comparativa con otros protocolos

En resumen, el protocolo propuesto puede considerarse como de descubrimiento de vecinos (*neighbor discovery protocols*) o de descubrimiento de servicios (*service discovery protocols*).

Los protocolos de descubrimiento de vecinos son útiles para que un dispositivo conozca al resto de dispositivos que se encuentran en su misma red, sin necesidad de tener una configuración previa y estática, y permitiendo dejar de almacenar la información sobre alguien en caso de determinar que este ya se encuentra inactivo. Con unos simples mensajes, un nodo puede conocer las direcciones de red del resto de nodos ubicados en ella, encontrar routers y guardar la información relativa al camino que deben seguir los paquetes para llegar hasta ese nodo descubierto.

Por otro lado, como su propio nombre indica, un protocolo de descubrimiento de servicios permite que un dispositivo encuentre un servicio en una red sin la necesidad de conocer su nombre, indicando solamente una descripción del servicio que desean conseguir.

En este punto del trabajo vamos a proceder a describir protocolos de los dos tipos mencionados, comparándolo con el que se ha propuesto en puntos anteriores. Se debe tener en cuenta que aunque se realice una comparación, estos no cuentan con el mismo ámbito de uso, ya que los protocolos que se describirán ahora fueron diseñados para ser utilizados en Internet o redes de área local, mientras que el nuestro es un protocolo enfocado a *Smart Cities* y para un uso concreto no tan generalista.

4.5.1 Neighbor Discovery Protocol para IPv6

El protocolo de descubrimiento de vecinos para la versión de IPv6, fue propuesto en el RFC 4861 en Septiembre de 2007 por T. Narten, E. Nordmark, W. Simpson y H. Soliman, con la intención de proveer a esta versión de IP de un protocolo de descubrimiento de nodos en una red como el creado en diciembre de 1998 para la versión 4.

La función principal del Neighbor Discovery Protocol, NDP en adelante, es permitir a los *hosts* y *routers* de una red conocer las direcciones de red de otros nodos a los cuales puede alcanzar a través de sus conexiones.

Además, NDP también permite a los *hosts* encontrar *routers* a los que enviar mensajes para que estos los redirijan a sus destinos correspondientes como consideren, pudiendo buscar alternativas en caso de que este haya caído y por lo tanto no pueda redirigir los paquetes que se le envíen.

Otra función interesante del protocolo, es que permite a los dispositivos mantener un registro de los elementos en la red que se mantienen activos y con los que por lo

tanto puede contactar. De esta manera, es capaz de detectar si un dispositivo ya no se encuentra disponible, por lo que deja de considerarlo como viable para realizar una comunicación.

Al igual que nuestro protocolo, NDP define diferentes mensajes que permiten la realización de las funciones para las que ha sido diseñado el protocolo:

- **Solicitud de routers:** este tipo de mensaje permite a los dispositivos cuya interfaz de red acaba de despertar, solicitar a los *routers* que generen mensajes de **Anuncio de router** en el momento en lugar de esperar al siguiente mensaje enviado de manera periódica.

  Este mensaje cuenta con los campos típicos de IP, como son la dirección fuente, la destino y el salto límite. Además, añaden campos como el tipo de mensaje (133), el código (0), un checksum para comprobar la integridad del mensaje, un campo reservado, el cuál deberá ser inicializado a 0 por el emisor e ignorado por el receptor y un campo final de opciones.

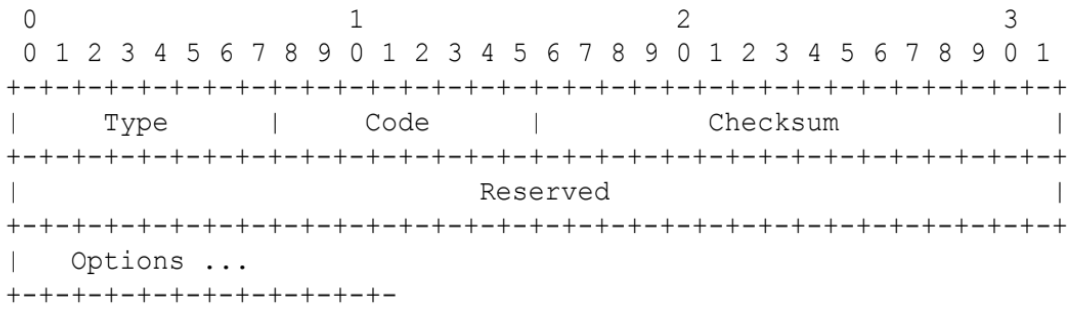

Imagen 4.13. Estructura de un mensaje de solicitud de routers

- **Anuncio de router:** mediante este mensaje, los *routers* anuncian su presencia en la red, o bien de manera periódica o cuando reciben un mensaje de solicitud de router desde un nodo.

  En este caso, el mensaje cuenta también con los campos IP indicando las dirección y el salto límite. Además, como en el mensaje anterior, también cuenta con un campo tipo (134), código (0), un *checksum*, *Cur Hop Limit (*especifica el valor a colocar en el campo *Hop Count)*, “M” que indica si la dirección es alcanzable mediante DHCP y “O”, que indica si existe configuración adicional que se proporciona mediante DHCP. También cuenta con campos sobre información del propio *router,* como su tiempo de vida, el tiempo que considerará que el nodo está activo tras haber recibido un mensaje desde él, etc.

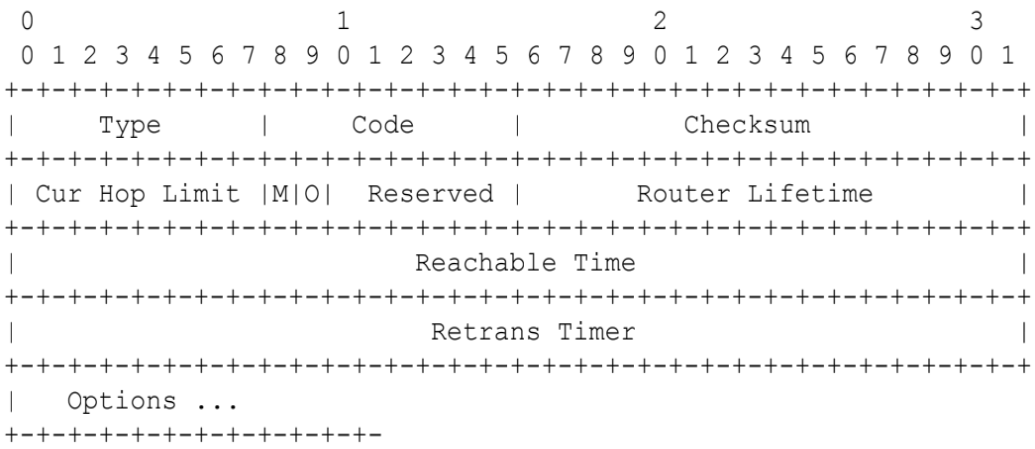

Imagen 4.14. Estructura de un mensaje de anuncio de routers

- **Solicitud de vecinos:** permite a un nodo determinar la dirección de un vecino en la red y asegurarse de que este todavía se encuentra activo para realizar comunicaciones.

  Cuenta con los campos IP usuales, el campo tipo (135), código (0), el checksum de comprobación, un campo reservado que no se utiliza y la dirección del nodo destinatario de la solicitud.

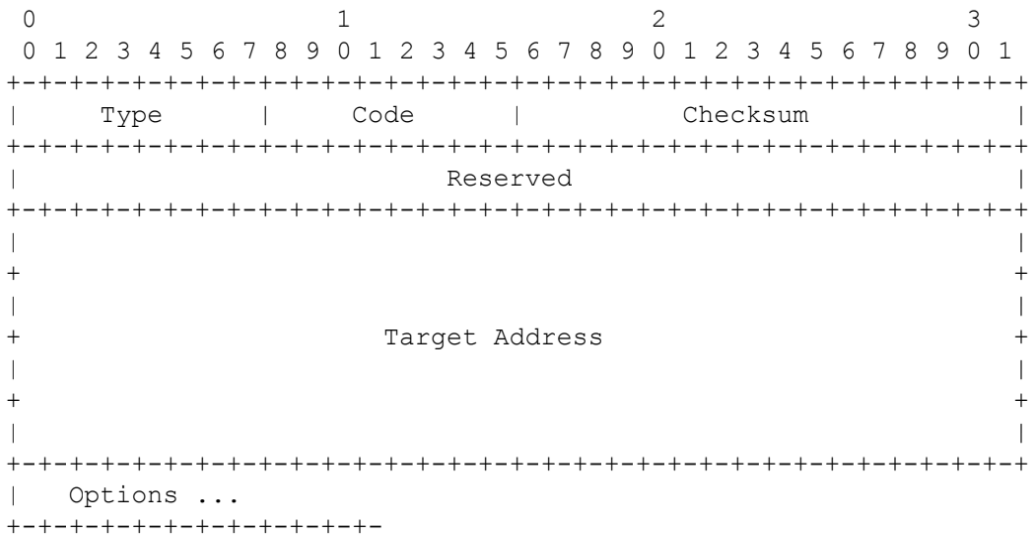

Imagen 4.15. Estructura de un mensaje de solicitud de vecinos

- **Anuncio de vecino:** es el mensaje en respuesta a la solicitud de vecinos. También puede ser enviado sin haber realizado una solicitud cuando se realizan cambios en la dirección de red u otras características del nodo.

  Cuenta con los campos IP habituales y añade a los de tipo (136), código (0) y checksum: “R” campo que indica si el emisor es un *router*, “S” indica si el mensaje se ha generado por una solicitud desde otro nodo o si por el contrario se ha generado automáticamente y “O”, que indica si la información sobre este nodo que tienen los demás nodos almacenada debe ser sobreescrita o no.

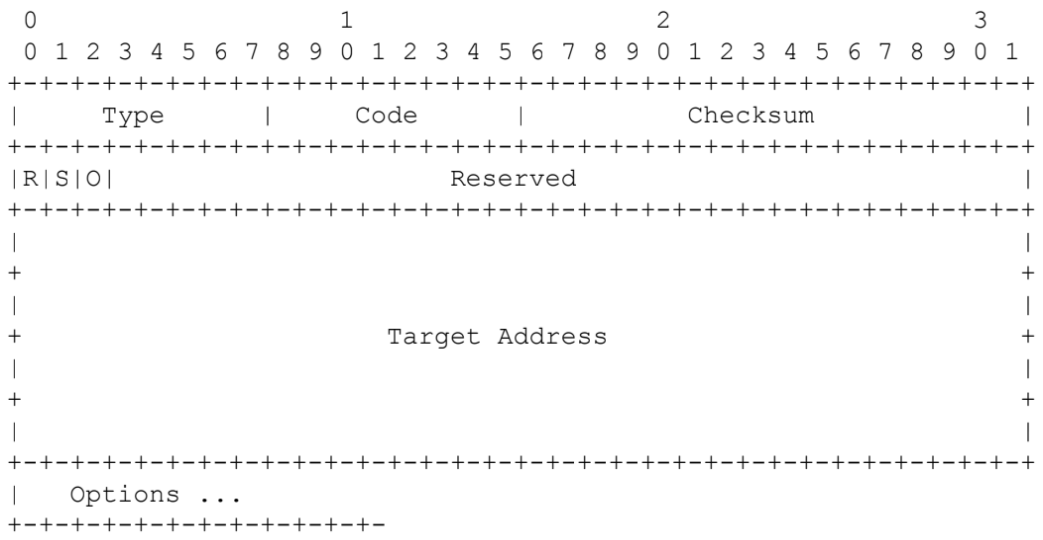

Imagen 4.16. Estructura de un mensaje de anuncio de vecino

- **Redirección:** los mensajes de redirección son enviados por los *routers* para informar a los nodos de que existe un camino mejor a través del cuál pueden enviar los mensajes al destino deseado. También permite indicar al nodo que el destinatario con el que está queriendo contactar es un vecino y por lo tanto no necesita pasar a través de un *router.*

  Este tipo de mensaje cuenta con los campos IP y los ya mencionados campos de tipo (137), código (0), *ckecksum*, la nueva dirección a través de la cual es mejor llegar al destino y la dirección destino en sí con la que se desea contactar.

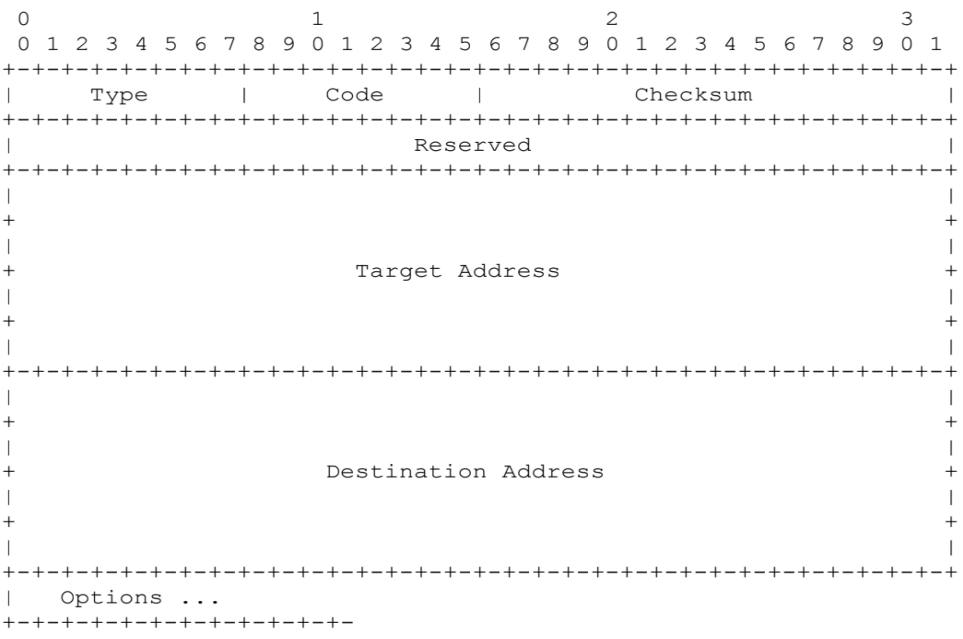

Imagen 4.17. Estructura de un mensaje de redirección

Mediante el uso de estos mensajes, los nodos son capaces de mantener información sobre los dispositivos ubicados en la red a la que ellos se encuentran conectados, pudiendo así conocer sus direcciones, estado y rutas óptimas para realizar comunicaciones en ella.

Una vez analizado este protocolo podemos determinar qué ventajas y desventajas puede tener su uso en una red convencional.

Ventajas:

- Permite a un dispositivo conocer su entorno de red sin necesidad de configuración previa y grandes tablas de datos
- Reconfiguración de la información con la que se cuenta, pudiendo añadir nuevos dispositivos y eliminando los que ya no se encuentran activos fácilmente
- Información sobre rutas óptimas de comunicación con otros nodos

Desventajas:

- Requiere que la red sea capaz de funcionar con mensajes de difusión
- Los mensajes de descubrimiento pueden sobrecargar la red si esta tiene poca capacidad y son utilizados en exceso

Como se habrá podido observar, existen ciertas similitudes entre el protocolo propuesto para este trabajo y este, ya que se utilizan mensajes de descubrimiento de *routers* (equivalente a nuestros mensajes de descubrimiento de puntos de acceso) los cuales son respondidos mediante mensajes que proporcionan información sobre los mismos y la forma en la que trabajan.

Por otro lado, no existe un mensaje equivalente al de descubrimiento de vecinos, ya que no se ha creído necesario de momento que un cliente se comunique con otro en el protocolo propuesto.

4.5.2 Simple Service Location Protocol

Como se ha comentado anteriormente, un protocolo de localización de servicios es empleado en una red para descubrir los servicios ofertados en esta sin la necesidad de contar con tablas o listas de configuración, que requieren configuración previa y pueden quedar obsoletas con tremenda velocidad.

Simple Service Location Protocol, en adelante SSLP, es un protocolo de localización de servicios diseñado para ser utilizado por dispositivos que empleen redes

locales inalámbricas de baja energía (LowPAN, *low-power wireless personal area networks*).

En redes que utilizan el protocolo IP, se utiliza el protocolo Service Location Protocol para encontrar información sobre los servicios ofertados en ellas. Este es útil en aquellas redes en las que no se tengan limitaciones de tamaño de los paquetes o de la energía que debe consumir el dispositivo, pero en LowPAN para IPv6 esto puede suponer un problema.

Dado que el tamaño máximo de un mensaje transmisible para 6LowPAN es de 81 octetos y que ya sólo la cabecera IPv6 ocupa 40 octetos, el espacio disponible para el envío de información restante es demasiado pequeño. Utilizando UDP, cuya cabecera ocupa 8 octetos, el espacio restante para datos sería de 33 octetos. Es por ello que se creyó necesario redefinir el protocolo para su uso en IoT con 6LowPAN.

SSLP reutiliza muchos de los términos de SLP, como los agentes involucrados en el protocolo:

- **Agente Usuario (AU)**: este agente actúa como cliente y es el interesado en establecer conexión con algún servicio de la red. Este cliente obtiene la información relativa a los servicios de los Directorios de Agentes y de los Agentes Servicio.

- **Agente Servicio (AS)**: es el agente situado en los servicios, el cual se comunica con los AU para ofrecer el servicio.

- **Directorio de Agentes (DA):** intermediario situado en zonas intermedias de la red, que guarda información sobre diversos servicios y se la proporciona a los AU cuando estos la requieren, actuando como una especie de servidor de nombres en función de las características solicitadas por el cliente. Su uso es opcional.

SSLP cuenta con varios tipos de mensajes que permiten el descubrimiento de servicios y directorios de agentes en una red:

- **Mensaje de petición de servicio**: este mensaje es utilizado para obtener las URLs asociadas a directorios de agentes y agentes proveedores de servicios.

- **Mensaje de respuesta de servicio:** es el mensaje enviado por un servicio cuyas características corresponden a las solicitadas por el agente usuario en el mensaje de petición de servicio. Contiene las URLs necesarias para poder acceder al servicio.

- **Mensaje de petición de tipos de servicio:** es un mensaje normalmente enviado a un directorio de agentes o mediante *Multicast* para conocer todos los servicios ofertados en una determinada red.

- **Mensaje de respuesta de tipos de servicio:** respuesta a un mensaje de petición de tipos de servicio indicando todos los servicios presentes en la red y sus correspondientes URLs.

- **Mensaje de registro de servicio:** mensajes generados por un agente servicio cuando encuentra un directorio de agentes. En el cuerpo se informan los servicios prestados y la URL a través de la cual se pueden acceder. También puede ser utilizado para eliminar del directorio de agentes un servicio determinado que ha dejado de prestarse.

- **Mensaje de confirmación de registro de servicio:** mensaje generado por el DA para confirmar al AS que un servicio se ha registrado o eliminado correctamente en su base de datos.

- **Mensaje de petición de atributos:** utilizado para obtener los atributos relacionados con un servicio en concreto. Contestado en el **mensaje de respuesta de atributos.**

- **Mensaje de anuncio de directorio de agentes:** es el mensaje utilizado por un DA para darse a conocer a los agentes usuarios y servicios.

Como se puede observar, cuenta con una gran variedad de tipos de mensajes diseñados para realizar las funciones del protocolo. Si bien es cierto que comparte mucha de la forma de trabajar del protocolo original SLP, SSLP ha conseguido reducir el tamaño de los mensajes considerablemente, haciéndolo apto para uso en redes con dispositivos de bajo consumo como pueden ser las de IoT.

5. Evaluación de un prototipo

5.1 Introducción: Tecnologías empleadas

5.1.1 Placas de desarrollo LoPy

LoPy es una placa de desarrollo fabricada por la empresa PyCom, que cuenta con oficinas en Reino Unido y Holanda. Surge de las dificultades con las que sus creadores vieron que contaban los desarrolladores del Internet de las Cosas para desarrollar y desplegar soluciones por la dificultad de integración entre sistemas y la complejidad de ellos debido a la poca abstracción del hardware.

Imagen 5.1 Placa de desarrollo LoPy con PyTrack

PyCom ofrece una plataforma IoT sencilla mediante la provisión de placas de hardware programables mediante el lenguaje de programación MicroPython, una versión reducida del popular lenguaje Python. Entre estas placas se encuentra LoPy, que es la utilizada para la implementación del protocolo descrito en este trabajo.

Además de una pequeña capacidad de cómputo, LoPy ofrece conectividad a Internet mediante protocolos como Bluetooth, LoRa o WiFi (en la versión más actual permite utilizar también SigFox).

PyCom también ofrece placas de expansión, cuya función es ofrecer servicios de sensorización, conectividad NFC o localización GPS.

En este trabajo se ha utilizado la placa de expansión PyTrack, que ha permitido utilizar localización GPS para mostrar la ubicación de un punto de acceso dentro de la ciudad y el filtrado por distancia que realizan los clientes al realizar consultas mediante *broadcast*.

5.1.2 MicroPython

MicroPython es una implementación software del conocido lenguaje de programación Python en su versión 3. Está escrito en C con el objetivo de ser ejecutado en microcontroladores, permitiendo una programación más sencilla de los mismos. Incluye muchas de las librerías *Core* de Python y una consola interactiva que permite ejecutar comandos.

Fue creado originalmente por Damien George, perteneciente al Centro de Investigación en matemáticas de la Universidad de Cambridge, siendo financiado mediante una campaña en Kickstarter en el año 2013. En la actualidad MicroPython es soportado por casi todas las arquitecturas ARM y es utilizado en infinidad de implementaciones de Internet de las Cosas y en Arduino.

Como se ha comentado, está basado en el popular lenguaje de programación Python 3, un lenguaje de propósito general definido comúnmente como **lenguaje de scripting orientado a objetos**. Fue creado a finales de los ochenta por Guido Van Rossum con el fin de ser un lenguaje accesible a personas que se iniciaran en el mundo de la programación pero con suficientes características para que también pudiera ser utilizado por desarrolladores con amplia experiencia. A menudo, muchos de sus usuarios se refieren al término **Filosofía Python**. Dentro de esta filosofía podemos encontrar principios como legibilidad y transparencia de código, elementos muy importantes para los creadores de Python.

Desde la versión 2.1.2 podemos encontrar estos principios filosóficos escribiendo `import this` en una consola Python:

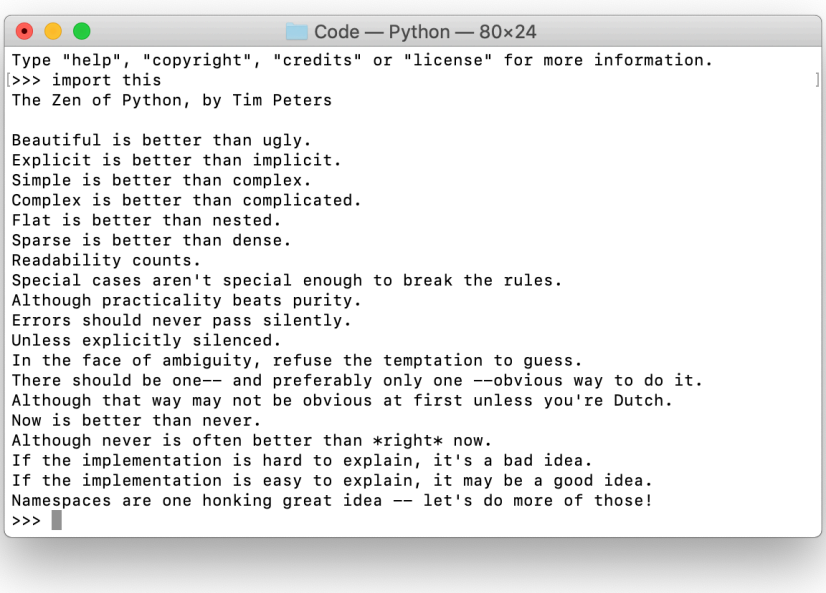

Imagen 5.2 Filosofía Python mostrada tras ejecutar import this en la consola

5.1.3 Node.js

Node JS es un entorno de ejecución en servidor concebido para funcionar de manera asíncrona y proporcionar una alta escalabilidad. Está basado en el lenguaje ECMAScript.

Funciona con un único hilo, aunque su orientación asíncrona permite la ejecución de hasta cientos de miles de eventos de manera concurrente.

Se organiza en módulos, los cuales proveen de diversas funcionalidades para el desarrollo. Algunos vienen previamente compilados en el propio binario de Node JS, como el módulo de red, que proporciona funcionalidad para programación de red asíncrona y otros módulos fundamentales, como por ejemplo Path, FileSystem, Buffer, Timers y Stream. También se pueden incluir módulos desarrollados por terceros, que extienden la funcionalidad de Node o proveen abstracción para desarrollos complejos o repetitivos.

En este proyecto, Node ha sido escogido para implementar el servidor de la web que actúa como sala de control de los dispositivos alojados en una ciudad debido a:

- La facilidad de desarrollo de un servidor que soporte conexiones concurrentes y su fácil escalabilidad.

- Gracias a su cercanía al sistema operativo, con el que interactúa para gestionar las conexiones, Node permite utilizar sockets UDP, necesarios para la comunicación con los dispositivos LoPy.

- Al estar basado en ECMAScript, permite la interacción con Embedded JavaScript, útil en el desarrollo del aspecto de las páginas web debido a que es posible insertar código javascript directamente en el HTML gracias a él.

5.1.4 Embedded JavaScript

Embedded JavaScript (EJS) es un lenguaje de marcado que permite insertar código JavaScript en los documentos HTML, encargados de definir la estructura y contenido de las páginas web.

Un documento EJS cuenta con las mismas etiquetas utilizadas en el lenguaje de marcado HTML y fragmentos de código JavaScript insertados, lo que permite compartir variables con el servidor, con el consiguiente ahorro de esfuerzo a la hora de por ejemplo mostrar elementos que sólo deben ser visibles en ciertas ocasiones, haciendo uso de cláusulas condicionales JavaScript insertadas en el código HTML.

En este trabajo, el uso de EJS ha permitido compartir los valores de las variables obtenidas directamente mediante el servidor en Node para ser mostradas en la página web y mostrar elementos de la página sólamente bajo ciertas condiciones.

Algunas de las ventajas de la utilización de EJS son:

- Su compilación y renderizado de la página web es extremadamente rápido, ya que el navegador está interpretando código HTML y JavaScript, lenguajes para los que están altamente optimizados

- Inserción sencilla de código JavaScript mediante etiquetas <%%> para la inserción de código simple y etiquetas <%=%> para establecer valores en campos HTML

- Sencillez a la hora de realizar *debug* de los errores, ya que son errores típicos de JavaScript que pueden ser visualizados en la consola del navegador o en el propio servidor

- Es totalmente compatible con cualquier servidor basado en JavaScript, entre ellos el utilizado en este trabajo, Node

5.1.5 BootStrap

BootStrap es un conjunto de librerías multiplataforma que ayuda en la abstracción de código a la hora de realizar diseños de aplicaciones web. Cuenta con diferentes plantillas para el diseño de cualquier elemento HTML, como pueden ser: botones, menús, barras de navegación, etc.

Fue desarrollado por Mark Otto y Jacob Thornton, empleados de Twitter, con el objetivo de crear un *framework* que permitiera la consistencia visual de los herramientas internas que se desarrollaban en la empresa. En agosto del año 2011, Twitter ofreció el código de BootStrap como código abierto, convirtiéndose este en el repositorio más popular en el año 2012 en GitHub.

Está basado en módulos descritos en lenguaje LESS, en los cuales se implementan las modificaciones a los elementos HTML. Estos módulos son totalmente modificables por los usuarios que los utilizan, por lo que la personalización de las páginas web que utilizan BootStrap no se encuentra limitada.

Su utilización es sencilla, los documentos HTML que utilizan BootStrap deben enlazar el archivo CSS a su página y hacer referencia a las clases descritas en este dentro del elemento *class* de los elementos HTML de la página.

Además de los módulos proporcionados en los archivos CSS de BootStrap pueden encontrarse en Internet plantillas desarrolladas por usuarios particulares, siendo algunas gratuitas y otras de pago, y que pueden ser utilizadas para realizar un desarrollo todavía más sencillo de la página web.

En este trabajo, BootStrap ha sido utilizado para permitir que la página web fuera *responsive* para ser utilizada en cualquier dispositivo y navegador compatible y permitir abstraer el diseño de la misma.

5.2. Descripción de la solución

Para implementar en un prototipo el protocolo de comunicación que se planteó para este trabajo de fin de máster, se ha procedido a dividir la funcionalidad en dos partes que serán descritas en este punto del documento.

Podemos diferenciar una primera parte o funcionalidad que es la ejecutada únicamente utilizando las placas de desarrollo LoPy y una segunda funcionalidad, la comunicación entre un punto de acceso y la plataforma web que actúa como sala de control y permite la configuración remota de los puntos de acceso.

5.2.1 Estructura de la solución en las LoPy

En la primera parte desarrollada podemos encontrar la implementación pura del protocolo. En ella, dos placas de desarrollo LoPy hacen uso del protocolo definido para intercambiar datos en tiempo real, actuando una de ellas como cliente del servicio y la otra como proveedor del mismo, respondiendo a las consultas recibidas.

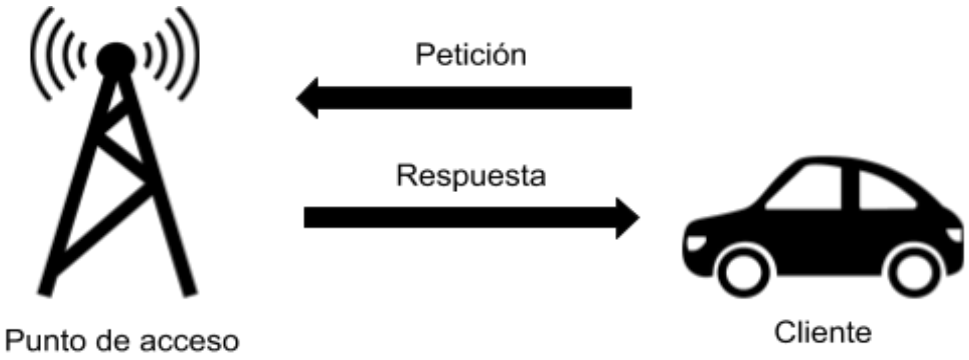

Imagen 5.3. Un cliente situado en un vehículo utiliza el protocolo de comunicación para obtener datos del punto de acceso

Para la implementación del protocolo en las placas de desarrollo LoPy se han generado diversos archivos cuya función se detalla a continuación:

- **boot.py**: Este fichero es el primero que ejecuta una placa de desarrollo LoPy en el arranque. Se ha aprovechado esto para que este archivo sea el encargado de conectar la placa a la red de Internet mediante WiFi. Así conseguimos asegurar la conectividad del dispositivo, requisito indispensable para cualquier objeto del Internet de las Cosas.

- **prot_earth.py:** Incluye la implementación de los métodos generadores de los mensajes de comunicación descritos en el protocolo. En él podemos encontrar métodos para los mensajes: IS_AP, CONFIRM_AP, GET_INFO, ANS_INFO, PUSH_CONFIG, CONFIG_OK.

- **earth_utils.py:** Contiene métodos auxiliares que son utilizados en la implementación de la funcionalidad para permitir una mejor visibilidad del código. Es por ello que podemos encontrar métodos como el cálculo de la proximidad entre dos coordenadas mediante el algoritmo de Haversine o métodos para el tratamiento de los datos recibidos en los mensajes.

- **ap.py:** Implementa la función de servidor desempeñada por el punto de acceso (AP). Este queda a la espera de recibir mensajes y filtra dependiendo del tipo de los mismos para ejecutar un flujo u otro, construyendo una respuesta para el cliente que ha realizado la solicitud.

- **client.py:** Contiene toda la funcionalidad ejecutada por el cliente en la placa de desarrollo LoPy. Este envía el mensaje de reconocimiento de tipo IS_AP y queda a la espera de recibir una contestación por parte de algún punto de acceso. Si la recibe solicita los datos que le sean necesarios según la función que desempeña.

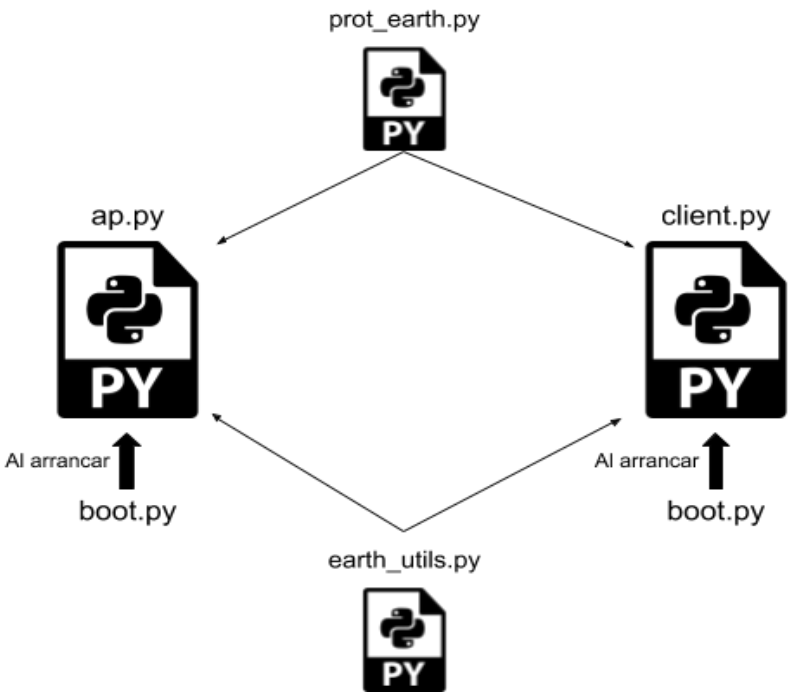

Imagen 5.4. Estructura de archivos creados para la implementación de la solución en las LoPy

5.2.2 Estructura de la solución web

La página web implementada actúa como sala de control para una Smart City que cuente con este protocolo para la comunicación entre puntos de acceso y clientes.

Así, esta permite la comunicación con los puntos de acceso para obtener el estado de los mismos y poder visualizarlos mediante una interfaz amigable. Esto es importante ya que puede servir para monitorizar los puntos de la ciudad que se encuentran activos, y si no lo están o su estado no es el esperado, abrir una incidencia para que acuda un técnico.

Además, permite la configuración remota de los AP, pudiendo cambiar su estado o funcionalidad que desempeñan sin realizar desplazamientos. Puede ser útil en casos excepcionales como festividades, en los que un semáforo no necesite estar encendido ya que la vía está cortada o se requiera mayor cantidad de contenedores de un tipo de basura que de otra.

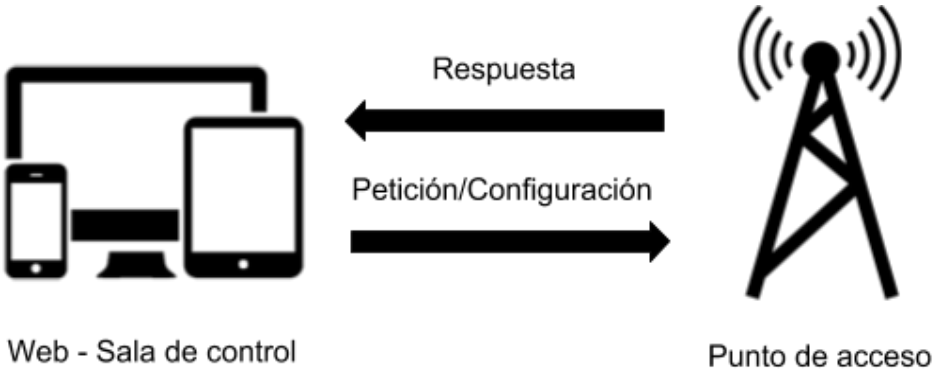

Imagen 5.5. Desde la sala de control web, los usuarios pueden consultar el estado de los puntos de acceso y configurarlos remotamente

Para la implementación de la página web que actúa como sala de control y la interacción que debe realizar con las placas de desarrollo LoPy que realizan la función de punto de acceso, ha sido necesario generar los archivos descritos a continuación:

- **server.js:** escrito en JavaScript utilizando la librería Node, realiza la función de servidor de la aplicación. Es la encargada de tratar los eventos generados desde la página web para recopilar los datos, construir los mensajes enviados a los AP, recibir las respuestas y mostrar la información recogida.

- **index.ejs:** escrito en HTML y utilizando Embedded JavaScript, tiene la función de describir la estructura de la página principal de la web. Su estilo se define a partir de las hojas de estilos de BootStrap. Cuenta con una estructura condicional, en la que se muestra una vista inicial en el primer acceso y tras realizar una consulta a un punto de acceso esta varía en función del topic sobre el que se esté consultando.

- **config.ejs:** escrito en HTML y utilizando Embedded JavaScript, tiene la función de describir la estructura de la página de configuración de los puntos de acceso. Al igual que la página principal, su estilo se ha definido utilizando BootStrap. En este caso, la vista no cambia dependiendo de los datos enviados o recibidos.

- **style.css:** contiene modificaciones menores a los estilos de la página web descritos en BootStrap.

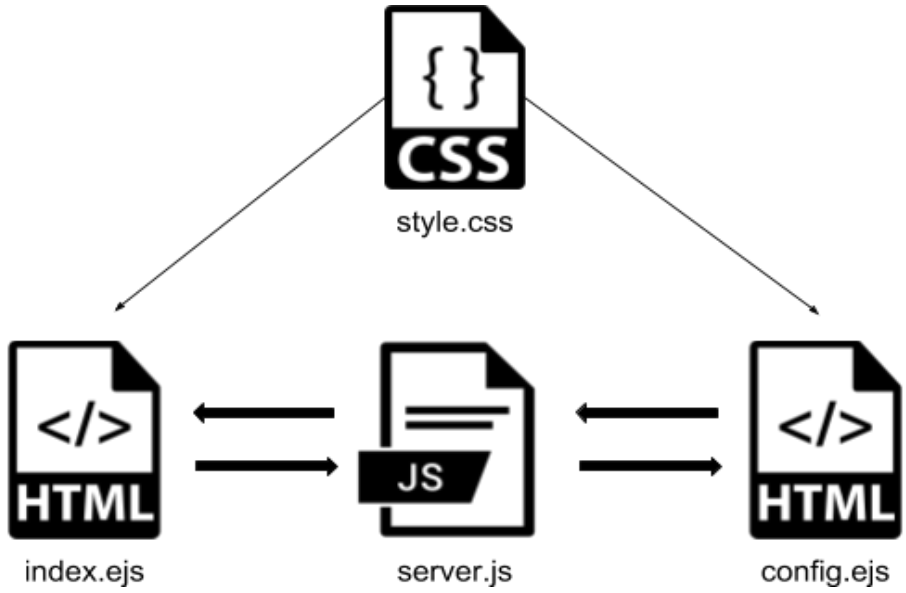

Imagen 5.6. Estructura de archivos creados para la implementación de la solución web

5.3. Casos de uso implementados

Para ilustrar el funcionamiento de la solución implementada para este trabajo, se ha procedido a definir tres posibles casos de uso en los que esta tecnología podría ser útil en una ciudad inteligente. Además, se han elegido estos tres específicamente debido a su facilidad de implantación en las ciudades actuales y los problemas contemporáneos que estos resuelven.

En este apartado se expondrá la motivación de cada una de ellas, una breve descripción y los beneficios que podrían tener en la movilidad y ordenamiento en una ciudad real.

5.3.1 Caso 1: Semáforos configurables

En la actualidad, cualquier persona que circule con un vehículo por una ciudad habrá podido observar que en ocasiones podemos encontrarnos detenidos en un semáforo en rojo sin motivo aparente. No hay peatones cruzando o con intención de hacerlo y la vía perpendicular a la que nosotros nos encontramos está completamente desierta.

Además, cuando nos encontramos en un atasco, los semáforos en verde para nosotros parecen ser más cortos de lo habitual, no nos da tiempo a llegar a ellos y vuelven a cambiar a rojo.

Estos problemas se producen a diario en las grandes ciudades actuales, en las que ir a trabajar puede llegar a demorarse de dos a tres horas para las personas que viven fuera de ella.

Debido a que ciertas zonas en concreto de una ciudad son muy concurridas durante ciertas horas del día, la regulación del tráfico por parte de los semáforos debería ser adaptable a las necesidades de la ciudad. Así, si el anillo interno de la ciudad se encuentra completamente colapsado durante una jornada festiva o una franja horaria determinada, los semáforos dispuestos en el anillo externo deberían tardar más tiempo en ponerse en verde, para permitir que los vehículos que ya se encuentran dentro puedan llegar a sus destinos y no contribuir más al embotellamiento hasta que este se despeje mínimamente.

Una posible solución para estas pérdidas de tiempo ante semáforos en rojo inútiles y grandes aglomeraciones de vehículos podría ser que estos se adaptaran dependiendo de la masificación de la vía, es decir, de los vehículos que circulan por ella en contraposición con los que circulan por la vía que la intersecta. Aunque no solo tenemos que tener en cuenta la cantidad de vehículos que circulan por una vía, si no que también debemos evaluar la cantidad de viandantes que desean cruzarla, ya que a ellos tampoco les gustaría tener que esperar durante media hora a que su semáforo se pusiera en verde para que los atascos de la ciudad se reduzcan.

Estudios como el de Ozan K. Tonguz[1] proponen que sean los propios vehículos los que se comuniquen entre ellos para decidir quién debe tener la preferencia en una intersección permitiendo así que sean los propios vehículos los que regulen el tráfico, sin la necesidad de contar con semáforos realizando esa función. De esta manera, un vehículo que llegue a una intersección en la que sólo él desea pasar, no deberá esperar a un semáforo en rojo que no se lo permita.

Esta decisión de quién tiene la preferencia en una intersección es calculada por cada uno de los vehículos que se aproximan a ella teniendo en cuenta su distancia a la intersección y la de los demás vehículos que pretenden pasar, además de la velocidad de cada uno de ellos y su trayectoria. De esta manera se elige un líder, el cual decide quién debe tener el semáforo en rojo y quién debe tenerlo en verde y pasar. Una vez el líder abandona la intersección, pasa el testigo a un vehículo en la vía perpendicular, que pasa a convertirse en el nuevo líder.

---

[1] Tonguz, O. (2018). Red light, green light - No light. Tomorrow's communicative cars could take turns at intersections. *IEEE Spectrum, 55(10),* 24-29. doi: 10.1109/mspec.2018.8482420

Mediante esta solución, el estado de los semáforos es variable en función de los vehículos que se encuentren en la intersección.

En este trabajo se ha propuesto una solución diferente, en la que el semáforo como tal no exista y sea sustituido por una baliza inteligente configurable que se comunique con los vehículos para comunicarles el estado actual del semáforo que representa en función de la concurrencia de la vía en ese momento.

Este estado puede ser calculado a partir de diferentes factores, como el número de vehículos que se encuentran en la vía, el número de peatones esperando para cruzar si los hubiera, el estado de las calles próximas a la zona y un largo etcétera. En nuestro caso, el estado del semáforo se calcula de forma aleatoria, ya que se ha creído fuera del alcance de este trabajo el investigar y proponer qué valores de los factores expuestos anteriormente deben hacer cambiar de color el semáforo.

Por lo tanto, la solución propuesta para el caso de uso de los semáforos configurables en este trabajo es:

1. Un vehículo que circula por una vía realiza consultas periódicas para encontrar puntos de acceso que traten el *topic* “traffic_lights” para saber el estado de los semáforos en una intersección próxima.

2. Cuando un punto de acceso recibe una de estas consultas, contesta indicando su posición geográfica al vehículo y le confirma que es un semáforo.

3. Recibida la confirmación, el vehículo debe comprobar su proximidad al punto de acceso, ya que los datos sobre el estado de un semáforo no cercano a su posición deben ser ignorados.

4. Confirmando la adecuación del punto de acceso a los intereses del vehículo este solicitará los datos deseados al semáforo, el cual le contestará con ellos. Para este caso se han definido tres *subtopics* que se creen útiles sobre la información de un semáforo para un vehículo circulando por una vía:

   a. Estado: como su propio nombre indica, es el estado en el que se encuentra el semáforo en el momento de la consulta. Este puede ser: “verde”, “naranja”, “rojo” y un último estado excepcional “emergencia”.
   b. Tiempo restante: indica el tiempo que falta para que el semáforo cambie de estado.
   c. Peatones: contador que indica el número de peatones esperando para cruzar el semáforo.

Estos tres *subtopics* pueden ser consultados por un vehículo interesado, utilizándolos para evaluar si debe frenar en la intersección o no.

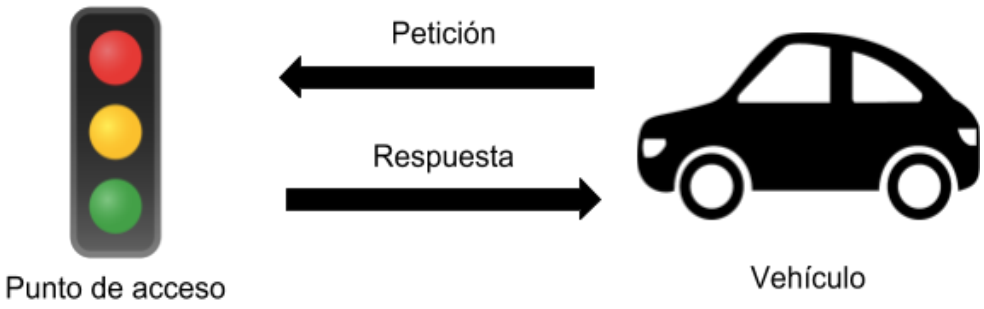

Imagen 5.7. Un cliente situado en un vehículo utiliza el protocolo de comunicación para obtener el estado de un semáforo y decidir si debe parar o no

Además de esta interacción habitual, también podrá contactar con un punto de acceso de tipo semáforo la web. Desde ella, podrá comprobarse el normal funcionamiento de este, realizar revisiones y abrir una incidencia si se cree que podría estar funcionando incorrectamente. También podrá ser configurado, pudiendo cambiar su estado manualmente dependiendo de las necesidades de la zona de la ciudad en la que esté situado.

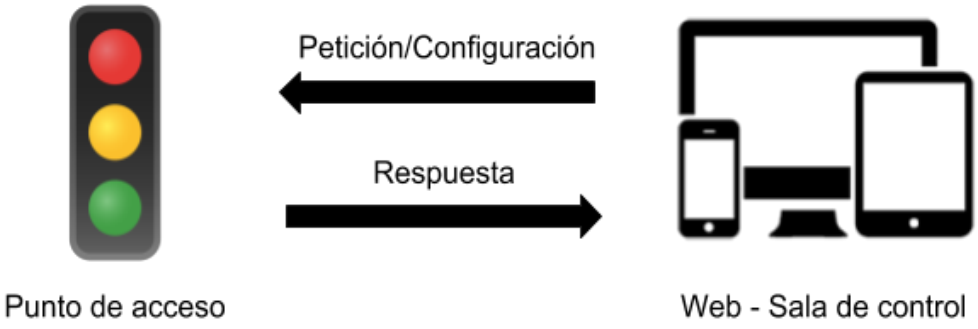

Imagen 5.8. Desde la web es posible consultar el estado del semáforo y realizar configuraciones sobre él de manera remota.

5.3.2 Caso 2: Recogida de residuos eficiente

Además de la movilidad dentro de las grandes ciudades, un problema grave al que estas se están enfrentando es la recogida de los residuos generados por sus habitantes y las industrias dentro de ellas.

Según el último estudio estadístico realizado por el **Instituto Nacional de Estadística** para España en el año 2016, en nuestro país se generaron 129 millones de toneladas de residuos sumando aquellos generados por la ciudadanía general (que representa el 16,8% del total) y los generados por las industrias (siendo un 83,2% del total). Del estudio se puede extraer que los residuos generados por los hogares, que son los mayoritarios dentro de las ciudades, aumentaron un 4,5% en ese año con respecto al año anterior. El estilo de vida del “usar y tirar” al que nos hemos acostumbrado los ciudadanos de las ciudades modernas del siglo XXI supone un gran reto logístico para mantener la limpieza y salubridad de las calles.

Concretamente, en el área metropolitana de Valencia se generaron en el año 2016 más de 750.000 toneladas de residuos, que deben ser recogidas y procesadas para ser reutilizadas o llevadas a vertederos. En la ciudad de Valencia, la empresa encargada de la recogida de residuos y adecuación de las calles es S.A. Agricultores de la Vega de Valencia (SAV). Debido a que participan en el "Registro de huella de carbono, compensación y proyectos de absorción de dióxido de carbono" del Ministerio de Transición Ecológica, podemos conocer que en el año 2016 esta empresa generó 7624 toneladas de CO2 durante la realización de su actividad.

Si bien es cierto que la recogida de residuos no es la única actividad que esta empresa realiza, los datos nos ilustran el gran impacto medioambiental que puede generar en una ciudad mediana como Valencia.

Es por ello que en este trabajo se ha creído importante definir un caso de uso para el protocolo creado relacionado con la gestión de residuos.

Este caso de uso tiene por objetivo una recogida más eficiente de estos residuos teniendo en cuenta la carga del contenedor para decidir si el camión debe detenerse a vaciarlo o no. De esta manera, se reducirán las paradas y arrancadas de estos grandes camiones, momentos en los que consumen una mayor cantidad de combustible.

Por supuesto, además de la reducción del consumo de combustibles, una recogida más eficiente de residuos puede implicar mayor velocidad de recogida y menores complicaciones en la circulación del tráfico de la ciudad debido a camiones de basura parados realizando su función de recogida.

Así, un camión sólo deberá parar en caso de que el porcentaje de llenado del contenedor objetivo sea mayor a un determinado umbral establecido teniendo en cuenta diversos factores:

- **Época del año:** habrá momentos del año en los que los contenedores se llenen con mayor rapidez. Por ejemplo, en verano los contenedores suelen llenarse con menos frecuencia debido a que la población de la ciudad está de vacaciones.

- **Tipo de contenedor:** un contenedor de basura orgánica tiene una frecuencia de llenado mayor que un contenedor de vidrio, ya que la cantidad de basura generada del primer tipo es mucho mayor a la del segundo.

- **Ubicación:** aquellos contenedores situados en zonas de restaurantes o locales de fiesta nocturna suelen contar con mayor generación de residuos de tipo vidrio, mientras que otras zonas pueden tener mayor generación de otro tipo de residuos.

- **Fecha:** además de la época del año, también es relevante en la generación de residuos el día o semana. Podemos poner el ejemplo de las fallas, en las que la cantidad de residuos generados en la ciudad de Valencia se dispara exponencialmente. Otro ejemplo son las fiestas navideñas, en las que la cantidad

de residuos de papel generados por los envoltorios de los regalos aumentan el llenado de los contenedores de papel y cartón.

Estos factores y otros muchos pueden influir en la cantidad de basura generada en una ciudad, aunque realizar este cálculo no se encuentra entre los objetivos de este trabajo.

En este trabajo se propone que un cliente situado en un camión de basura dentro de una ciudad inteligente, interactúe utilizando el protocolo de comunicación diseñado con puntos de acceso incrustados en contenedores para consultar a estos el estado de los mismos, pudiendo así decidir si debe parar a realizar la recogida o si por el contrario debe continuar la marcha.

El funcionamiento del caso de uso aplicando el protocolo sería el siguiente:

1. Un camión realiza una ruta predefinida por la ciudad inteligente realizando consultas para encontrar contenedores dentro de su recorrido. Por lo tanto, en este caso, el camión actuará como cliente del protocolo.

2. Una vez un contenedor reciba un mensaje de descubrimiento, confirmará al camión su condición de contenedor enviando también su posición geográfica.

3. El camión, conociendo la ubicación del contenedor realizará una consulta de su estado y tipo de basura que este recoge.

4. Una vez recibida la información, el primer paso será comprobar si el tipo de basura recogida por el contenedor es el mismo que recoge el camión. Si no lo es, el procesamiento no continuará. Si por el contrario recogen el mismo tipo de residuos, el camión evaluará si debe detenerse teniendo en cuenta la carga actual del contenedor.

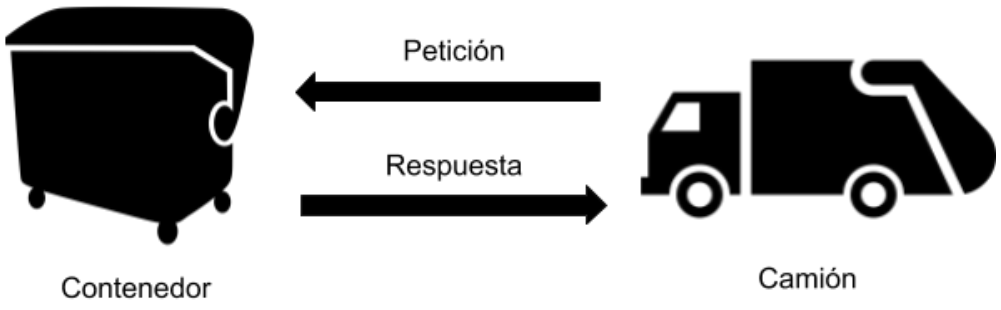

Imagen 5.9. Un cliente situado en un camión de basura utiliza el protocolo de comunicación para obtener el estado y tipo del contenedor y decidir si debe vaciarlo o no

De esta manera, el camión de basura sólo realizará paradas en los casos en los que sea estrictamente necesario teniendo en cuenta la velocidad de llenado de los contenedores.

Desde la web, se podrá consultar el estado de cada uno de los contenedores ubicados en la ciudad y su posición geográfica, pudiendo ser útil esto para por ejemplo enviar servicios excepcionales de recogida en los casos en los que una zona determinada de la ciudad tenga una producción de residuos mayor de lo habitual.

Además, el contenedor podrá ser configurado remotamente desde la web, pudiendo por ejemplo cambiar el tipo de basura que este recoge.

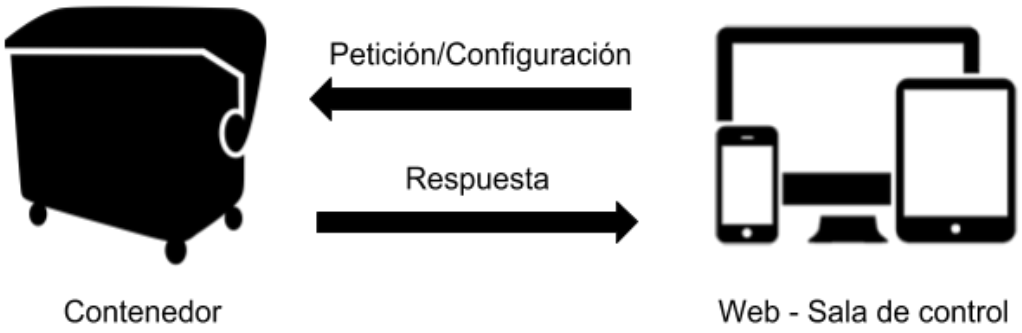

Imagen 5.10. Desde la web es posible consultar el estado del contenedor y realizar configuraciones sobre él de manera remota.

5.3.3 Caso 3: Limitación de velocidad

En los últimos años se han visto aumentados el número de siniestros de circulación causados por los excesos de velocidad. Los vehículos comerciales actuales consiguen alcanzar grandes velocidades sin mucho esfuerzo, lo que ocasiona que los conductores superen los límites de velocidad establecidos en las vías con gran frecuencia.

De un total de 65.000 accidentes con víctimas ocurridos en el año 2017, la DGT establece que 5.682 de ellos fueron causados por excesos de velocidad. Es evidente que los excesos de velocidad son un gran factor de riesgo durante la conducción, pero aún así muchos conductores continúan infringiendo los límites establecidos, pudiendo generar situaciones de peligro e incluso accidentes.

Pudiendo aprovechar las nuevas tecnologías que cada día más se están incluyendo en los vehículos de nueva fabricación, se propone un caso de uso para el protocolo definido en este trabajo que ayuda en la limitación automática de la velocidad de los vehículos que circulan por una vía.

El funcionamiento del caso de uso relativo a la limitación automática de velocidad utilizando el protocolo propuesto sería:

1. Un vehículo circula con normalidad por una vía. Periódicamente envía mensajes de descubrimiento de puntos de acceso que traten el *topic "speed limit"*.

2. Un punto de acceso, que estará situado a la entrada de la vía y reciba un mensaje de descubrimiento por parte de un cliente situado en un vehículo, responderá a este confirmando su existencia y enviando datos relativos a su ubicación, identificación, etc.

3. Tras recibir este mensaje de confirmación, el vehículo consultará al punto de acceso el límite de velocidad establecido para la vía por la que circula o a la que está a punto de entrar.

4. El punto de acceso responderá enviando los datos solicitados al vehículo.

Si bien es cierto que en la actualidad la información recibida por el vehículo sobre la velocidad de la vía no tendría efecto alguno sobre el vehículo, en trabajos futuros podrían realizarse investigaciones para fijar la limitación de velocidad de este utilizando sus ordenadores de a bordo y la información recibida desde el punto de acceso.

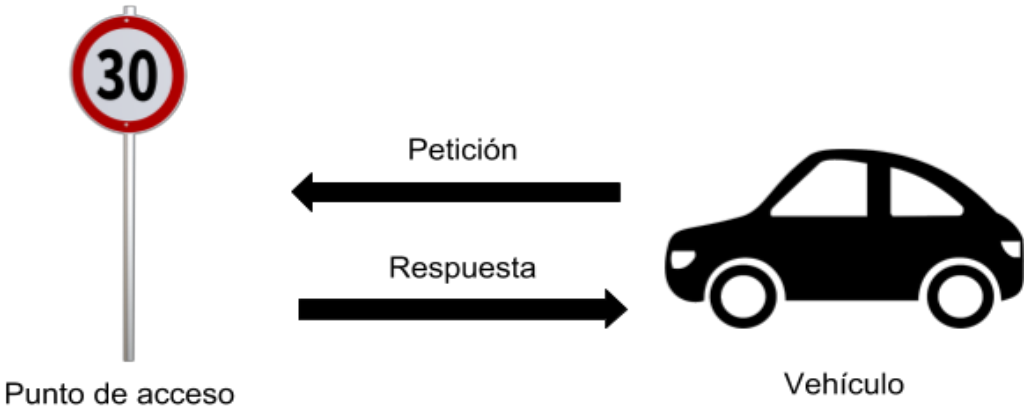

Imagen 5.11. Un vehículo que circula por una vía consulta a un punto de acceso la velocidad límite de la vía

Desde la web, será posible consultar el estado del punto de acceso. Además, este podrá ser configurado, de manera que si por causas excepcionales como obras o mal tiempo el límite de velocidad debe ser reducido, esto podrá ser realizado cómodamente de manera remota.

Esto facilita las tareas de conservación de las vías, ya que en primer lugar se podría prescindir de las señales visuales a las que estamos acostumbrados. La información que estas representan podría ser mostrada al conductor del vehículo en una pantalla en el interior. Además, cambios temporales en los límites de velocidad de las vías como los ocurridos en España durante el año 2011 con la intención de ahorrar

energía, ya no supondrían un gran reto. Ese año, miles de señales ubicadas en las vías tuvieron que ser modificadas una a una para cumplir con la nueva legislación y volver a ser modificadas una vez esta dejó de estar en vigor.

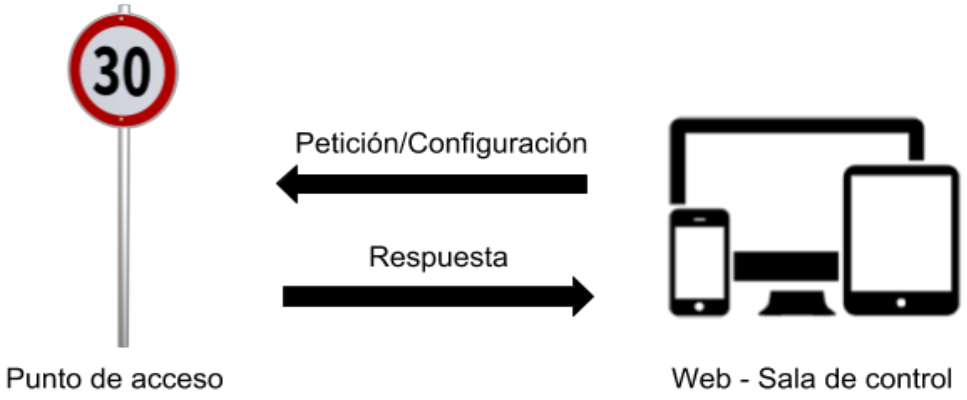

Imagen 5.12. Desde la web es posible consultar el estado del punto de acceso y realizar configuraciones sobre él, pudiendo por ejemplo cambiar la velocidad límite de una vía.

5.4. Implementación de la solución

Tras haber descrito las distintas funcionalidades implementadas en este trabajo con detalle en los puntos anteriores, vamos a pasar a desgranar los detalles de la implementación realizada para poder desarrollar la solución.

En primer lugar se describirán los pasos seguidos para permitir que dos placas de desarrollo LoPy implementaran el protocolo diseñado como solución para comunicaciones dentro de una ciudad inteligente entre dispositivos empotrados. Como ya se ha comentado en varias ocasiones, una de estas placas actúa como cliente y la otra como servidor en la comunicación. En este punto se describirá cómo se ha llegado a realizar esta comunicación.

Después se pasará a detallar la implementación de la página web que simula una sala de control dentro de la ciudad inteligente, encargada de la monitorización y configuración remota de todos los dispositivos ubicados en ella. Para ello, se describirá tanto la implementación del servidor como de la interfaz.

5.4.1 Implementación en las LoPy

La implementación de la solución en las LoPy se ha realizado utilizando el lenguaje MicroPython y solo utilizando librerías propias del lenguaje, sin añadir ninguna mantenida por terceros, que a la larga puede provocar la desaparición del soporte de las mismas en el futuro o que aparezcan fallos graves en el sistema debido a su utilización.

Como ya se comentó en el punto 2.1 de esta memoria, se han desarrollado diversos archivos para asegurar una mínima modularidad en la solución implementada. De esta manera se consigue que las dependencias entre archivos se encuentren más localizadas.

5.4.1.1 Arranque de los dispositivos

Cuando una placa de desarrollo LoPy es conectada a una fuente de energía, esta realiza un proceso de arranque similar a la de cualquier dispositivo *hardware*. Tras realizar el proceso, en el que se ha encendido la unidad de proceso, se han habilitado los dispositivos de entrada/salida, etc., LoPy ejecuta un primer archivo denominado **boot.py**.

Este archivo se encuentra vacío por defecto, pero el programador puede utilizarlo para realizar las primeras acciones tras el arranque del dispositivo. En nuestro caso, se ha aprovechado para conectar las placas a la red de Internet mediante WiFi. De esta manera aseguramos que cualquier acción que queramos realizar posteriormente y que requiera una conexión va a contar con ella.

En primer lugar se ha procedido a realizar las importaciones de las librerías necesarias para implementar la funcionalidad de conexión a la red mediante WiFi. Estas son “WLAN” del paquete “network” y “machine”:

- **Network:** librería de MicroPython que contiene los métodos necesarios para configurar las conexiones WiFi.

- **Machine:** contiene métodos relacionados directamente con la placa que está siendo utilizada. Permiten el control de los distintos elementos *Hardware* como pueden ser la CPU, buses del sistema, relojes o dispositivos de entrada salida. Debido a que ofrece acceso a estos elementos sin restricción alguna debe ser utilizada con precaución ya que se pueden causar daños al sistema.

```
import machine
from network import WLAN

# boot.py -- run on boot-up
SSID = '[redacted]'
AUTH = '[redacted]'

wlan = WLAN(mode=WLAN.STA)

nets = wlan.scan()
for net in nets:
    print(nets)
    if net.ssid == SSID:
        print('Network found!')
        wlan.connect(net.ssid, auth=(net.sec, AUTH), timeout=5000)
        while not wlan.isconnected():
            machine.idle() # save power while waiting
        print('WLAN connection succeeded!',wlan.ifconfig())
        break
```

Imagen 5.13. Código implementado para boot.py

Posteriormente se declaran las variables **SSID** y **AUTH**, que contendrán el nombre de la red WiFi a la que se va a conectar la placa y la contraseña de la misma respectivamente.

Realizado esto, se instancia el objeto wlan, que será creado en modo **WLAN.STA**, lo que quiere decir que actuará como cliente de un punto de acceso WiFi. El modo alternativo es **WLAN.AP**, el cual configura el dispositivo como un punto de acceso WiFi al que otros clientes se conectan.

Hecho esto se realiza un escaneo de la red, lo que permite conocer todos los puntos de acceso WiFi disponibles al alcance del dispositivo. Con esta lista de puntos se puede realizar un bucle que la recorra. En el caso de encontrar la red cuyo SSID coincide con el SSID de la red a la que deseamos conectarnos, se realizará un intento de conexión.

Si la conexión se realiza satisfactoriamente, la ejecución de **boot.py** terminará. Si por el contrario no se consigue conectar con la red deseada, el dispositivo entrará en modo *idle*, el cual permite ahorrar energía hasta que se consiga realizar dicha conexión.

5.4.1.2 Implementación de la mensajería

Los mensajes incluidos en el protocolo de comunicación propuesto en este trabajo han sido implementados en un fichero denominado **prot_earth.py**. Se ha escogido realizar esta implementación en un fichero separado de todo el resto de la funcionalidad para que esta pueda ser fácilmente exportable a cualquier ámbito y a trabajos futuros.

Para una comprensión más sencilla de la implementación del protocolo pasamos a recordarlo brevemente describiendo los pasos seguidos para realizar una comunicación con el:

1. Un cliente envía periódicamente mensajes de tipo IS_AP, que permiten el descubrimiento de puntos de acceso que tratan el mismo *topic*.

2. Un AP que reciba un mensaje de tipo IS_AP evaluará si el *topic* que él trata se corresponde con el incluído en el cuerpo del mensaje recibido. Si es así, responderá al cliente mediante un mensaje de tipo CONFIRM_AP. Si por el contrario no tratan el mismo *topic*, el AP no responderá.

3. Recibido el mensaje de tipo CONFIRM_AP, el cliente deberá comprobar su distancia al punto de acceso, cálculo que realizará usando el algoritmo de Haversine, cuya entrada serán las coordenadas recibidas desde el AP y las suyas. En el caso en el que el AP se encuentre relativamente cerca, enviará un mensaje de tipo GET_INFO solicitando la información que requiera. Si por el contrario el AP se encuentra demasiado lejos, no lo considerará viable y terminará la comunicación con el.

4. Tras recibir el mensaje de tipo GET_INFO, el AP construirá y enviará el mensaje de respuesta ANS_INFO al cliente con la información que este ha solicitado.

5. Recibida la información solicitada, el cliente termina la comunicación y realiza las acciones que desee con los datos obtenidos desde el punto de acceso.

Revisado el flujo seguido por el protocolo de comunicación podemos pasar a analizar cómo se ha implementado la construcción de los mensajes del mismo.

- **Mensaje IS_AP**

El primer mensaje enviado es IS_AP. Debido a su estructura, el método asociado a su construcción en MicroPython deberá contar con las entradas relativas a:

- El **identificador del AP (ap_id)** con el que se desee contactar. Si no se desea uno en concreto, este campo puede estar vacío.

- **Identificador del cliente (client_id)** que realiza la consulta.

- **Dirección de red (client_ip)** del cliente. Esta dirección será necesaria en el siguiente paso para que el AP conteste al cliente con la confirmación.

- **Puerto en el que trabaja el cliente (client_port).** Como en el caso anterior, este será útil para la respuesta por parte del AP

- ***Topic*** sobre el que el cliente desea hablar con el punto de acceso. Este será el campo que el AP evaluará para saber si debe o no contestar al cliente.

```
def is_ap_message(apId,clientId, clientAddress, clientPort, topic):
    msg = '{"type": "IS_AP", "ap_id": "'
    msg+=apId
    msg+='", "client_id": "'
    msg+=clientId
    msg+='", "client_ip": "'
    msg+=clientAddress
    msg+='", "client_port": "'
    msg+=str(clientPort)
    msg+='", "topic": "'
    msg+=topic
    msg+='"}'
    return msg
```

Imagen 5.14. Código constructor del mensaje de tipo IS_AP

Como se puede observar en la imagen, el código no es más que un método que permite construir un String en formato JSON, añadiendo en las posiciones necesarias los datos de entrada del método.

- **Mensaje CONFIRM_AP**

Como en el mensaje de tipo IS_AP, la construcción de un mensaje CONFIRM_AP requerirá unas entradas determinadas a su método asociado. Como algunas de ellas se repiten con respecto al caso anterior, solo se describen a continuación las añadidas en este:

- **Dirección de red del AP (ap_ip):** Será utilizada por el cliente para enviar el mensaje GET_INFO en caso de requerirlo.

- **Puerto en el que trabaja el AP (ap_port).** Como en el caso anterior, este será útil para la petición por parte del cliente.

- ***Subtopics*.** Este campo contiene aquellos subtemas que el AP trata dentro del *topic* general por el que ha sido preguntado por parte del cliente. Este campo sirve para comunicar al cliente qué información puede pedir al AP y cuál no.

- **Coordenadas geográficas (coordinates):** son los datos relativos a la posición geográfica en longitud y latitud en la cual se encuentra el punto de acceso. Esta será utilizada por el cliente para evaluar la conveniencia de realizar la consulta de información al AP o no.

```
def confirm_ap_message(apId, apAddress, apPort, clientAddress, topic, subtopics, coordinates):
    msg = '{"type": "CONFIRM_AP", "ap_id": "'
    msg+=apId
    msg+='", "ap_ip": "'
    msg+=apAddress
    msg+='", "ap_port": "'
    msg+=str(apPort)
    msg+='", "client_ip": "'
    msg+=clientAddress
    msg+='", "topic": "'
    msg+=topic
    msg+='", "subtopics": "'
    msg+=str(subtopics)
    msg+='", "coordinates": "'
    msg+=coordinates
    msg+='"}'
    return msg
```

Imagen 5.15. Código constructor del mensaje de tipo CONFIRM_AP

El método **confirm_ap_message** construye el mensaje retornando un String con formato JSON, que será el enviado al cliente para confirmar la condición del punto de acceso como tal y su disposición a tratar el *topic* mencionado en el mensaje de descubrimiento IS_AP enviado por el cliente.

- **Mensaje GET_INFO**

Tras recibir la confirmación parte del punto de acceso de que trata el *topic* solicitado y evaluar la adecuación de realizar la comunicación con este por su proximidad, el cliente deberá construir el mensaje de tipo GET_INFO para solicitar la información requerida al AP.

```
def get_info_message(apId,apAddress,clientId,clientAddress,clientPort,topic,subtopics):
    msg = '{"type": "GET_INFO", "client_id": "'
    msg+=clientId
    msg+='", "client_ip": "'
    msg+=clientAddress
    msg+='", "client_port": "'
    msg+=str(clientPort)
    msg+='", "ap_id": "'
    msg+=apId
    msg+='", "ap_ip": "'
    msg+=apAddress
    msg+='", "topic": "'
    msg+=topic
    msg+='", "subtopics": "'
    msg+=str(subtopics)
    msg+='"}'
    return msg
```

Imagen 5.16. Código constructor del mensaje de tipo GET_INFO

El String resultante de la ejecución de este método, que contará con formato JSON, será enviado al AP, el cual deberá preparar la respuesta y enviarla en un mensaje de tipo ANS_INFO.

- **Mensaje ANS_INFO**

Tras recibir la petición de información por parte del cliente, el punto de acceso construirá la respuesta utilizando los datos que contiene en su memoria. Los datos de respuesta serán insertados en un diccionario cuya clave será el *subtopic* solicitado y el valor, el contenido en el AP.

Como caso especial, si el punto de acceso es consultado por un *subtopic* que no trata, el valor insertado en el diccionario para este será un mensaje de error de la forma: *"Error. Subtopic not available on this context"*.

```
def ans_info_message(apId, clientId, topic, response):
    msg = '{"type": "ANS_INFO", "ap_id": "'
    msg+=apId
    msg+='", "client_id": "'
    msg+=clientId
    msg+='", "topic": "'
    msg+=topic
    msg+='", "response": "'
    msg+=str(response)
    msg+='"}'
    return msg
```

Imagen 5.17. Código constructor del mensaje de tipo ANS_INFO

El String resultante de la ejecución de este método, contará con formato JSON y será enviado al cliente. Este realizará las acciones que crea conveniente con ellos, cerrándose con este último mensaje la comunicación.

- **Mensajes de configuración desde la web**

Como ya se ha comentado en diversas ocasiones a lo largo de este trabajo, se ha diseñado una web a modo de prototipo que actúa como sala de control de la ciudad inteligente en la cual se ubican los dispositivos que utilizan el protocolo propuesto.

Además de visualizar los datos almacenados en los puntos de acceso, los cuales se obtienen mediante los mensajes cuya implementación se ha descrito en los puntos anteriores, también se pueden realizar labores de configuración remota sobre ellos.

Para realizar configuraciones sobre un AP, se enviará a este un mensaje de tipo PUSH_CONFIG, que contará además de con los datos relativos a la información del AP que va a ser configurado y aquellos relativos a quién está realizando esta configuración, con los datos a cambiar en la memoria de este.

```
def push_config_message(clientId, clientIp, clientPort, apId, topic, subtopics):
    msg = '{"type": "PUSH_CONFIG", "client_id": "'
    msg+=clientId
    msg+='", "client_ip": "'
    msg+=clientIp
    msg+='", "client_port": "'
    msg+=str(clientPort)
    msg+='", "ap_id": "'
    msg+=apId
    msg+='", "topic": "'
    msg+=topic
    msg+='", "subtopics": "'
    msg+=str(subtopics)
    msg+='"}'
    return msg
```

Imagen 5.18. Código constructor del mensaje de tipo PUSH_CONFIG

Realizadas las modificaciones solicitadas en el punto de acceso, este deberá confirmar que ha realizado los cambios con un simple mensaje de confirmación que contará con información relativa a su identificación y *topic* sobre el cual ha realizado las modificaciones.

```
def config_ok_message(apId,apIp,apPort,topic):
    msg = '{"type": "CONFIG_OK", "ap_id": "'
    msg+=apId
    msg+='", "ap_ip": "'
    msg+=apIp
    msg+='", "ap_port": "'
    msg+=str(apPort)
    msg+='", "topic": "'
    msg+=topic
    msg+='"}'
    return msg
```

Imagen 5.19. Código constructor del mensaje de tipo CONFIG_OK

5.4.1.3 Implementación de la lógica en los AP

Como se ha comentado en puntos anteriores de esta memoria, un punto de acceso del protocolo propuesto actúa como una especie de servidor que recibe consultas y responde a ellas de manera acorde.

Es por ello que es un dispositivo pasivo que se encuentra en reposo mientras no reciba ninguna comunicación, y que realizará las tareas para las que está diseñado en el caso en el que se le requiera.

Para implementar esta funcionalidad, que ha sido realizada en MicroPython como en el caso de la mensajería, se ha generado el archivo **ap.py**. En primer lugar se han realizado las importaciones de los paquetes necesarios para la implementación. Entre ellos podemos encontrar:

- **Socket:** Incluye toda la funcionalidad necesaria para crear *sockets UDP* y con ello permite realizar comunicaciones mediante el protocolo TCP, utilizado para enviar y recibir mensajes en esta implementación.

- **Ujson**: Este módulo permite tratar mensajes en formato JSON dentro del lenguaje MicroPython. Contiene métodos que permiten por ejemplo la conversión de un String a un objeto Python en formato JSON y viceversa. Es una reducción del paquete json del lenguaje Python.

- **Prot_earth**: Como se ha comentado anteriormente, este módulo contiene los métodos necesarios para la creación de los mensajes del protocolo propuesto en este trabajo. Es utilizado para construir las respuestas a los mensajes recibidos desde los clientes.

- **Earth_utils:** contiene un conjunto de métodos implementados para asegurar la limpieza del código y la no repetición de sentencias muy frecuentes. Por ejemplo podemos encontrar accesos a elementos de diccionarios, cálculo de números aleatorios o cálculo de distancias entre coordenadas geográficas.

```
import socket, ujson, prot_earth, earth_utils
from network import WLAN
```

Imagen 5.20. Importación de paquetes en ap.py

Una vez realizadas las importaciones de los paquetes necesarios, se procede a crear el *socket UDP* necesario para las comunicaciones con los clientes o en su caso la web.

Este *socket* tiene una particularidad, y es que debemos indicar en la configuración que este puede recibir mensajes que provengan de *broadcasts*, ya que recordemos que los mensajes de descubrimiento enviados por los clientes son transmitidos de esta manera. Para ello en primer lugar, deberemos crear el *socket*, indicando que se trata de un *socket* que utiliza el protocolo UDP y que trabajará con datagramas.

Hecho esto, indicamos en las opciones del *socket* que: a nivel de *socket (socket.SOL_SOCKET)* habilitamos la opción *broadcast* (32) para permitir que se puedan recibir mensajes de tipo difusión.

Además, para poder recibir estos mensajes se debe enlazar el *socket* a una dirección de Internet vacía y sólo indicar el puerto en el que se va a trabajar.

```
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
server.setsockopt(socket.SOL_SOCKET, 32, 1)
server.bind(("", AP_PORT))
```

Imagen 5.21. Creación del socket UDP

Con el *socket* de comunicación creado, el AP entra en un bucle infinito en el que queda a la espera de recibir mensajes desde los clientes.

Una vez recibido un mensaje, el punto de acceso realiza una primera comprobación evaluando si el identificador del punto de acceso al que va dirigido el mensaje se encuentra informado en el cuerpo del mismo o no. Posteriormente, se pasa a comprobar qué tipo de mensaje se ha recibido, separando el tratamiento del mismo en función de esta información.

En el caso en el que se haya recibido un mensaje de descubrimiento del tipo IS_AP, el punto de acceso comprobará si el *topic* que desea hablar el cliente es el mismo que el que él trata. Además, también se comprobará la identidad del punto de acceso al que va dirigido el mensaje. Si esta es igual a la del AP en cuestión o venía vacía y además se cumplen las otras condiciones, el punto de acceso contestará mediante un mensaje de confirmación CONFIRM_AP.

```
while True:
    data, addr = server.recvfrom(1024)
    content = ujson.loads(data)
    if('ap_id' in content):
        ap_id = earth_utils.get_ap_id(content)
    else:
        ap_id = ""
    if (earth_utils.get_msg_type(content) == "IS_AP"
    and earth_utils.get_msg_topic(content) in AP_TOPIC
    and(ap_id==my_id or ap_id==""")):
        #Envia confirmacion porque es AP y tiene entre sus topics el solicitado
        clientAddress = earth_utils.get_client_address(content)
        client_port = earth_utils.get_client_port(content)
        client_topic = earth_utils.get_msg_topic(content)
        print("\n\n=====================================")
        print('Sending confirmation to address: '+ clientAddress)
        message = prot_earth.confirm_ap_message(my_id, address, AP_PORT,
        clientAddress, client_topic, SUBTOPICS, coordinates)
        server.sendto(message, (clientAddress, int(client_port)))
```

Imagen 5.22. Bucle a la espera de recibir mensajes y lógica para la confirmación tras recibir un mensaje de descubrimiento.

Si en lugar de un mensaje de tipo IS_AP, se recibe un mensaje GET_INFO, el punto de acceso deberá construir la respuesta a la información solicitada en función del *topic* y *subtopics* sobre los que el cliente desee obtener datos.

Para ello, en primer lugar el AP obtendrá los datos asociados al cliente que ha realizado la consulta para poder enviarle la respuesta posteriormente, además del *topic* sobre el que la realiza.

```
#Recibimos mensaje de peticion de informacion
elif earth_utils.get_msg_type(content) == "GET_INFO":
    clientAddress = earth_utils.get_client_address(content)
    client_port = earth_utils.get_client_port(content)
    client_id = earth_utils.get_client_id(content)
    client_topic = earth_utils.get_msg_topic(content)
    print("\nInfo acquisition message arrived from: "+clientAddress+" PORT: "+client_port)
```

Imagen 5.23. Tras recibir un mensaje GET_INFO, el AP recoge los datos del cliente para construir la respuesta posteriormente.

En función del *topic* sobre el que el cliente realiza la petición de información, la funcionalidad ejecutada será ligeramente diferente, ya que cada uno de los *topics* cuenta con un conjunto limitado de *subtopics* sobre los que se pueden realizar consultas.

Si tomamos el ejemplo de los semáforos configurables, los *subtopics* sobre los que el cliente puede consultar son “state”, “time_remaining” y “pedestrians”.

```
#Respuesta al topic semaforos
if(client_topic == 'traffic_lights'):
    print('Preguntado por semaforos')
    #Gestionar la informacion pedida para devolver una respuesta
    ans_info = earth_utils.get_msg_subtopics(content)
    ans_info = ans_info.replace(',',' ').replace('[',' ').replace(']',' ').replace("'",' ').split()
    response = {}
    for sub in ans_info:
        if(sub == 'state'):
            state = earth_utils.get_semaphore_state(state)
            print(state)
            response[sub] = state
        elif(sub == "time_remaining"):
            time_remaining = earth_utils.get_semaphore_time_remaining(state)
            response[sub] = time_remaining
        elif(sub == "pedestrians"):
            pedestrians = earth_utils.get_semaphore_pedestrians(state)
            response[sub] = pedestrians
        else:
            print("Asked for a topic not in database: "+sub)
            response[str(sub)] = "Error. Subtopic not available on this context"
    message = prot_earth.ans_info_message(my_id,client_id, 'traffic_lights', str(response))
    server.sendto(message,(clientAddress, int(client_port)))
    print("Information sent to client: "+clientAddress+" PORT: "+client_port)
    print("\n\n====================================")
```

Imagen 5.24. Construcción de la respuesta a la petición de información tras una consulta por el *topic* “traffic_lights”

Como se puede observar en el código de la Imagen 20, el siguiente paso realizado tras obtener la información asociada al mensaje recibido desde el cliente es el filtrado a partir del *topic* sobre el que se consulta. En este caso se usa como ejemplo el fragmento de código dedicado a la construcción de la respuesta tras una petición sobre el *topic* de los semáforos configurables.

Hecho esto, se obtienen los *subtopics* que son de interés para el cliente. Este campo es tratado para ser incluído en una lista de Python que sea iterable mediante un bucle, ya que es recibida en formato String y contiene símbolos como comas y llaves, los cuales hacen difícil su tratamiento. De esta forma, si contáramos con una cadena de entrada de la forma: "['state','time_remaining','pedestrians']", esta sería transformada a ["state","time_remaining","pedestrians"], siendo una lista Python conteniendo elementos de tipo String eliminando los caracteres como llaves comillas y comas mediante la línea de código:

```
ans_info = ans_info.replace(',',' ').replace('[',' ').replace(']',' ').replace("'",' ').split()
```

Obtenidos los *subtopics* para los que hay que enviar una respuesta al cliente solicitante, se pasa a construir el campo *response,* que contendrá un JSON con los datos sobre el estado del punto de acceso para cada uno de los elementos solicitados. Esto se realiza mediante la utilización de métodos auxiliares implementados en **earth_utils.py**. En este trabajo, se han implementado estos métodos de forma que devuelvan estados de manera aleatoria, sin incluir una lógica que sería necesaria para la implantación de este sistema en un entorno real.

Así por ejemplo, el método que indica el tiempo restante para que el semáforo cambie de estado, genera un número aleatorio entre el 1 y el 60 en el caso en el que el estado del mismo no sea "emergency", ya que en este caso el tiempo restante no es relevante y por lo tanto se retorna -1. Este número aleatorio es generado mediante un número de 32 bits generado directamente en la CPU de la placa de desarrollo LoPy al que se le ha realizado un tratamiento para su conversión a un número entero.

```
def get_semaphore_time_remaining(state):
    if(state!='emergency'):
        return round(RandomRange(1,60))
    else:
        return -1
```

Imagen 5.25. Método utilizado para la obtención del tiempo restante en un estado del semáforo

Como se comentó en la definición del protocolo, en el caso en el que el *subtopic* solicitado por el cliente no sea uno de los tratados por el AP, este contará con el mensaje "Error. Subtopic not available on this context" en el cuerpo de la respuesta generada.

Generada correctamente la respuesta, se procede a construir el mensaje que será enviado al cliente mediante el método correspondiente del módulo **prot_earth**.

El último caso posible para un punto de acceso, es que el mensaje recibido no sea de comunicación desde un cliente si no que por el contrario sea generado desde la web para realizar configuraciones. En este caso, el mensaje recibido será de tipo PUSH_CONFIG.

En primer lugar, se obtienen los datos relativos a la dirección del cliente desde el que se ha recibido el mensaje, para posteriormente poder responder mediante un mensaje indicando que los cambios se han realizado correctamente.

Posteriormente, se obtiene el campo *topic* y se comprueba que este sea igual al tratado por el punto de acceso. Además, se obtienen los *subtopics* sobre los que se desea realizar modificaciones. Estos son recorridos y modificados con los nuevos valores indicados desde la configuración de la web.

```
elif(earth_utils.get_msg_type(content)=='PUSH_CONFIG' and ap_id==my_id:
    print("Recibido un push config")
    clientAddress = earth_utils.get_client_address(content)
    client_port = earth_utils.get_client_port(content)
    topic = earth_utils.get_msg_topic(content)
    subtopics = earth_utils.get_msg_subtopics(content)
    subtopics = subtopics.replace('{','').replace('}','').split(',')
    if(topic=='traffic_lights'):
        for sub in subtopics:
            if(sub=='state'):
                state=subtopics[subtopics.index('state')+1]
            if(sub=='time_remaining'):
                time_remaining=subtopics[subtopics.index('time_remaining')+1]
            if(sub=='pedestrians'):
                pedestrians=subtopics[subtopics.index('pedestrians')+1]
```

Imagen 5.26. Tratamiento de un mensaje de configuración recibido desde la web

Realizados los cambios solicitados, el AP pasará a construir un mensaje de confirmación de tipo CONFIG_OK que indicará a la web que las modificaciones indicadas han sido realizadas correctamente.

```
message = prot_earth.config_ok_message(my_id,address,AP_PORT,topic)
server.sendto(message,(clientAddress, int(client_port)))
```

Imagen 5.27. Envío del mensaje de confirmación de configuración

5.4.1.4 Implementación de la lógica en los clientes

Descrita la funcionalidad implementada para los dispositivos que actuarán como punto de acceso en el protocolo de comunicación propuesto, podemos pasar a analizar cómo se ha desarrollado la funcionalidad de los clientes del mismo.

Como ya sabemos, los clientes se encontrarán ubicados en diferentes elementos de la ciudad, como pueden ser vehículos particulares, camiones de basura o incluso en dispositivos que las personas lleven consigo. Estos lanzarán mensajes de descubrimiento de puntos de acceso periódicamente, con el fin de obtener la información que les interesa para realizar las funciones para las que han sido diseñados.

La funcionalidad de los clientes ha sido desarrollada en el fichero **client.py**, el cual es el encargado de realizar las consultas periódicas de descubrimiento de APs y enviar las peticiones de información a los mismos para obtener los datos necesarios en el ejercicio de sus funciones.

En primer lugar, como en casos anteriores se realizan las importaciones de paquetes auxiliares necesarios para la solución. En este caso se han utilizado **socket, machine, ujson** y **time** como paquetes de terceros y **prot_earth** y **earth_utils** como paquetes propios. No pasamos a describirlos ya que se ha realizado en puntos anteriores.

Hecho esto y tras declarar variables globales necesarias para el desarrollo, se procede, como en el caso del punto de acceso, a declarar el *socket* de comunicación UDP necesario para enviar y recibir mensajes utilizando el protocolo TCP. Este se declara de manera que permita enviar o recibir mensajes de difusión o *broadcast,* útiles en el descubrimiento de puntos de acceso. Además, a diferencia del caso de la implementación del AP, se establece un *timeout* de un segundo, útil para que el cliente no quede bloqueado a la espera de recibir mensajes de respuesta desde puntos de acceso en caso de que no exista alguno cercano o que ninguno trate el mismo *topic* que él. De esta manera, si no se recibe una respuesta, el cliente volverá a lanzar otra consulta transcurrido este segundo de espera.

```
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
client.setsockopt(socket.SOL_SOCKET,32, 1)
# Timeout para que el cliente no se bloquee
# esperando una respuesta
client.settimeout(1.0)
client.bind(("", CLI_PORT))
```

Imagen 5.28. Creación del socket de comunicación y configuración del timeout

Una vez creado el *socket* de comunicación, el cliente entra en un bucle en el que en primer lugar realiza un envío de mensaje de descubrimiento de puntos de acceso IS_AP. Este mensaje será enviado siempre y cuando no exista una comunicación en trámite para evitar sobrecarga de las comunicaciones, debido a que los dispositivos

empotrados no suelen contar con grandes capacidades de cómputo o de hilos disponibles.

Esto será controlado mediante la variable booleana SENT, la cual se encontrará a True si se ha enviado un mensaje de descubrimiento y todavía no se ha terminado la comunicación, siendo False en caso contrario.

```
while True:
    if(not SENT):
        #Envio mensaje busqueda APs
        message = prot_earth.is_ap_message(CLI_ID, address, CLI_PORT, TOPIC)
        client.sendto(message, ('255.255.255.255', 37020))
        print("\n\n====================================")
        print("Sent IS_AP message")
        SENT = True
```

Imagen 5.29. Bucle infinito y envío de mensaje de descubrimiento IS_AP en el caso en el que no exista uno en trámite

Enviado el mensaje de descubrimiento, el cliente quedará a la espera de recibir confirmación por parte de de algún punto de acceso cercano. Si un mensaje de tipo CONFIRM_AP es recibido por su parte, este deberá comprobar si este se encuentra en una posición cercana a él. En este caso, esta distancia mínima se ha establecido en 500 metros como ejemplo, pero deberá ser ajustada en función del caso de uso en el que esté siendo utilizado el cliente.

```
data, addr = client.recvfrom(1024)
#Nos llega un mensaje
if(data):
    content = ujson.loads(data)
    ap_position = earth_utils.get_ap_coordinates(content)
    #CONFIRM_AP recibido desde el AP
    if(earth_utils.get_msg_type(content) == 'CONFIRM_AP'
       and earth_utils.haversine(MY_POSITION, ap_position)<=500.0):

        ap_address = earth_utils.get_ap_address(content)
        ap_port = earth_utils.get_ap_port(content)
        print("\nCONFIRM_AP received from: "+ap_address+" PORT: "+ap_port)
```

Imagen 5.30. Recepción de un mensaje de confirmación desde un punto de acceso y comprobación de la distancia entre el AP y el cliente

Por lo tanto, si el punto de acceso del cual hemos recibido confirmación es válido, el cliente procederá a construir el mensaje de petición de información. Esto será realizado teniendo en cuenta los *subtopics* que tienen en común el punto de acceso y el cliente. Así, si el cliente está interesado en obtener información sobre los *subtopics* “state”, “time_remaining” y “pedestrians” del *topic* “traffic_lights” pero el punto de acceso sólo trata el *subtopic* “state”, únicamente se realizará la consulta sobre este. Esta

obtención de los *subtopics* en común se realiza mediante el método **in_common** de **earth_utils.py,** cuya funcionalidad es simplemente obtener la intersección entre dos listas de Python.

```
#Valores en comun entre dos listas
def in_common(lst1, lst2):
    lst3 = []
    for i in lst1:
        if i in lst2:
            lst3.append(i)
    return lst3
```

Imagen 5.31. Método in_common de earth_utils.py

Obtenidos los *subtopics* sobre los que se realizará la petición de información, se puede pasar a construir el mensaje de consulta GET_INFO y proceder al envío directamente al AP, ya que su dirección de Internet ya es conocida, por lo que no es necesario un mensaje de difusión.

```
#subtopics que trata el ap
subtopics = earth_utils.get_msg_subtopics(content)
#Subtopics en comun entre intereses del cliente y los del AP
our_interests = earth_utils.in_common(INTERESTS,subtopics)

#Sabemos los sutopics, ahora podemos preguntar sobre ellos
message = prot_earth.get_info_message(earth_utils.get_ap_id(content),
                                      ap_address,
                                      CLI_ID, address, CLI_PORT,
                                      TOPIC, our_interests)
client.sendto(message, (ap_address, int(ap_port)))
print("\nSent GET_INFO message to: " + ap_address+" PORT: "+ap_port)
```

Imagen 5.32. Obtención de subtopics en común y envío de la petición de información al AP

Implementado el envío de la petición de información al punto de acceso, sólamente nos queda por describir la manera en la que se ha implementado la recepción de la respuesta al mismo.

Como en casos anteriores, en caso de recibir un mensaje de tipo ANS_INFO, el cliente obtendrá los datos por los que ha realizado la consulta del campo “response”. Con ellos disponibles, el cliente podrá realizar la funcionalidad para la que haya sido diseñado, como puede ser: actuar sobre los frenos del vehículo, decidir si el camión de basura debe parar a vaciar un contenedor o no, o cualquier otra acción posible.

```
if(earth_utils.get_msg_type(content) == 'ANS_INFO'):
    ap_id = earth_utils.get_ap_id(content)
    response = earth_utils.get_msg_response(content)
    print("\nGot response to GET_INFO from: "+ap_id)
    print("With content: "+str(response))
    print("\n\n====================================")
SENT = False
```

Imagen 5.33. Recepción de información en un mensaje de tipo ANS_INFO enviado por el AP

5.4.1.5 Resultado comunicación entre cliente y AP

Descrita la funcionalidad, podemos observar en el *log* generado en las consolas de los dispositivos LoPy el funcionamiento del protocolo de comunicación para un cliente que pregunta al punto de acceso por el *topic* “traffic_lights” y los *subtopics* “state” y “pedestrians”.

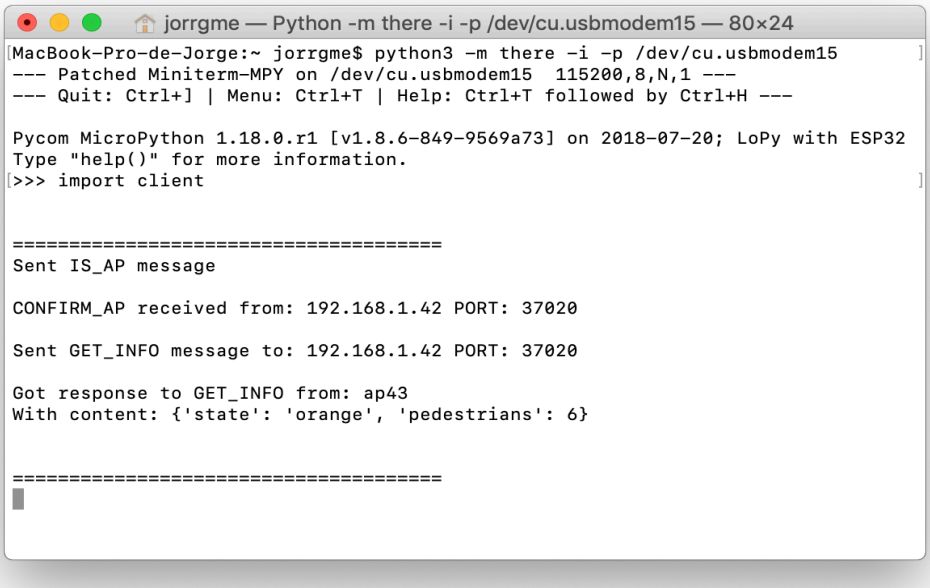

Imagen 5.34. Resultado log comunicación en el cliente

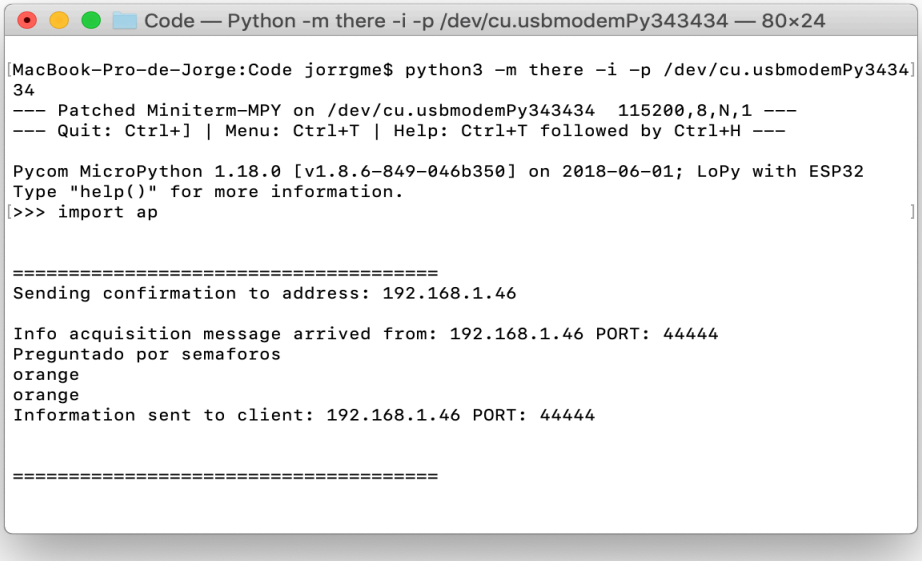

Imagen 5.35. Resultado log comunicación en el AP

5.4.2 Implementación de la web

Una vez terminada la descripción de los pasos que se han seguido para realizar una implementación posible del protocolo de comunicación dentro de una *smart city* de este trabajo, entre un agente cliente y otro punto de acceso convencionales, podemos pasar a describir el segundo tipo de comunicación posible: la realizada entre un **agente web y un punto de acceso**.

Este tipo de comunicación cuenta con una particularidad con respecto a la descrita anteriormente, y es que se realiza entre dispositivos cuya arquitectura no es la misma. Como en el caso anterior, el punto de acceso se encuentra ubicado en un dispositivo LoPy y su funcionalidad se ha desarrollado en lenguaje MicroPython, mientras que la funcionalidad de la web se encuentra alojada en un servidor Node.js y ha sido desarrollada en lenguaje JavaScript. La posibilidad de realizar la comunicación satisfactoriamente entre dispositivos tan diferentes entre sí demuestra la flexibilidad y capacidad de adaptación del protocolo propuesto, ya que es capaz de ser utilizado en entornos muy diferentes siendo aún así los dispositivos que lo utilizan compatibles entre ellos.

Como ya se comentó en el punto 2.2, en el que se analizó la estructura de la web, esta actúa como una especie de sala de control cuya función es monitorizar el estado de los puntos de acceso ubicados en una ciudad y permitir su configuración de manera remota, pudiendo así cambiar el estado en el que están con acciones simples y una interfaz amigable.

En el punto citado se comentó que se ha requerido crear cuatro archivos para realizar la implementación de la web, los cuales serán analizados a continuación, explicando los detalles de la implementación, mostrando ejemplos del resultado obtenido al utilizar la página.

5.4.2.1 Interfaz web

La interfaz web desarrollada para la sala de control, es decir, el diseño que visualizarán los usuarios de la aplicación, consta de dos pantallas bien diferenciadas:

- **Home:** es la primera pantalla a la que se accede al entrar a la página web. Su función es realizar las consultas sobre el estado de los puntos de acceso y mostrar la información recibida desde los mismos.

- **Configuración:** como su propio nombre indica, esta es la pantalla que permite la configuración de los dispositivos de manera remota. De esta manera, se podrá modificar el valor de cada uno de los *subtopics* que trata el punto de acceso sobre el que se realizan los cambios.

La pantalla **Home** cuenta en su primera carga con el formulario necesario para realizar la consulta a los puntos de acceso sobre su estado. Este formulario contiene dos campos, un desplegable que permite elegir el *topic* sobre el que se va a preguntar y un campo de texto que permite introducir el identificador del punto de acceso sobre el que se desea realizar la consulta. Si este campo se encuentra vacío, la consulta será respondida por el punto de acceso más cercano.

De esta manera, el diseño de la pantalla cuando es cargada por primera vez es el siguiente:

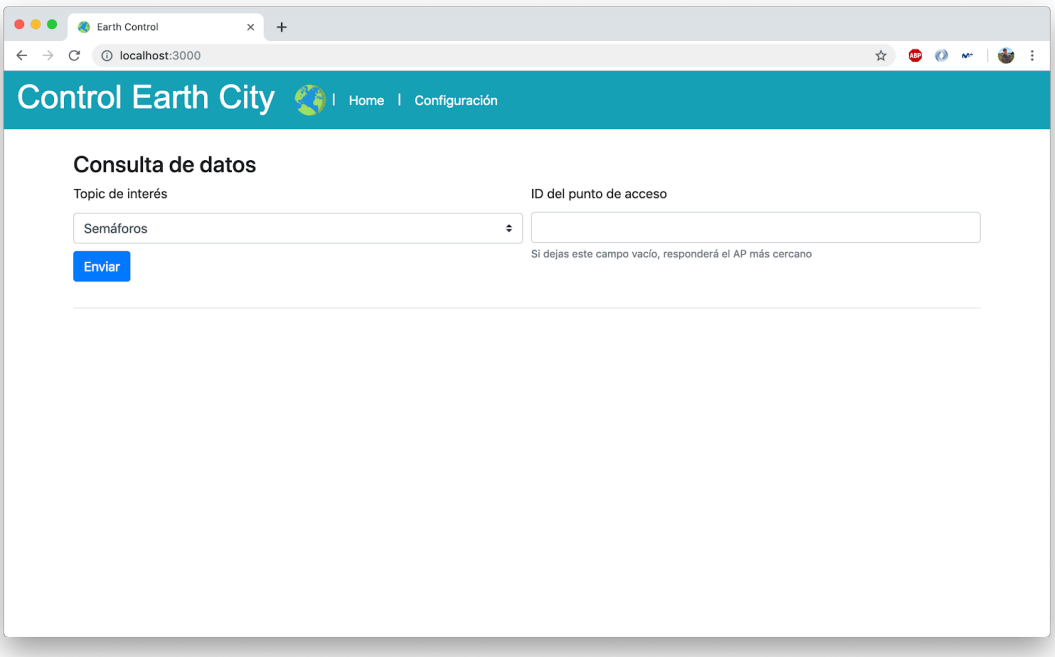

Imagen 5.36. Pantalla inicial de la sala de control

Una vez rellenados los campos y realizada la consulta sobre algún punto de acceso, el contenido de la página variará en función del *topic* sobre el que se haya realizado la consulta.

Como se comentó en puntos anteriores, para este trabajo se han implementado tres casos de uso para permitir probar el funcionamiento del protocolo y proporcionar ejemplos de uso que podrían ser útiles a la hora de implantar esta solución en una ciudad real. Estos tres casos de uso se encuentran disponibles en la web, los cuales recordamos que eran: Semáforos Configurables, Recogida de Basura y Límite de velocidad.

Así, si por ejemplo se ha realizado la consulta sobre el *topic* de semáforos, se mostrará información relativa a sus tres *subtopics (state, time_remaining y pedestrians),* siempre y cuando se encuentren disponibles para ser consultados en el punto de acceso:

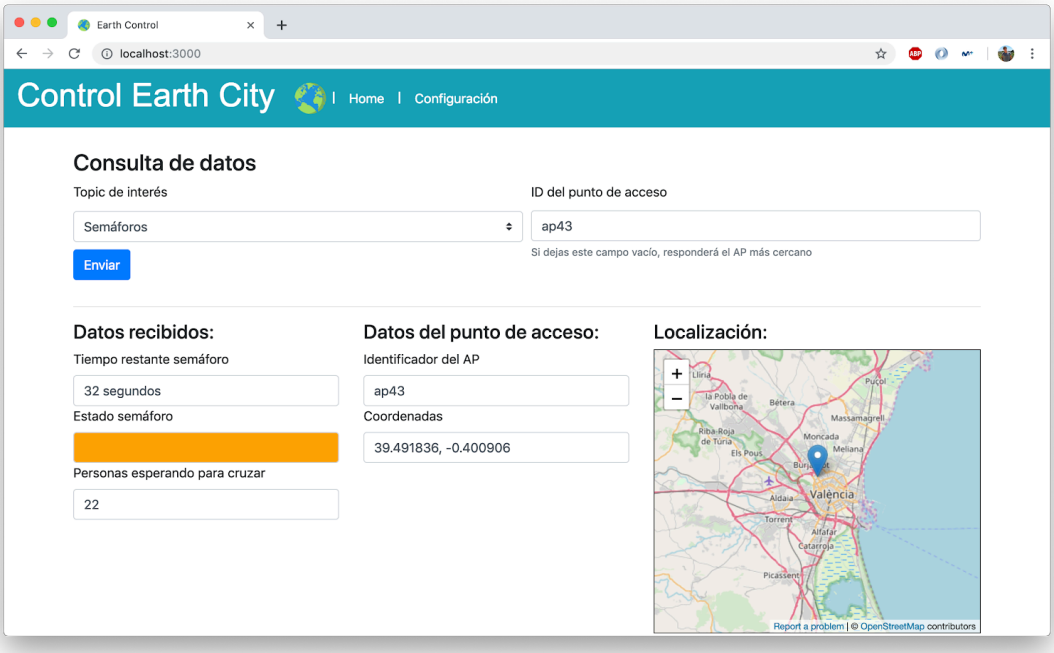

Imagen 5.37. Información mostrada en la pantalla Home tras realizar una consulta sobre el *topic* Semáforos

Como se puede observar en la imagen, se ha realizado una consulta sobre el punto de acceso con identificación “ap43”, al que se le ha preguntado sobre el *topic* Semáforos. En este caso, el punto de acceso cuenta con los tres *subtopics* disponibles, por lo que podemos ver el estado del semáforo, que en este caso es ámbar, el tiempo restante hasta realizar el cambio de estado, que son 32 segundos y las personas que se encuentran esperando para cruzar, siendo en este ejemplo 22.

Además de la información relativa al estado del punto de acceso, se obtienen metadatos de él, como pueden ser su identificación y las coordenadas en las que se encuentra situado. Estas coordenadas se utilizan también para embeber un mapa

gracias al API de la web de mapas gratuitos **OpenStreetMap** que indica el punto exacto en el que se encuentra el punto de acceso consultado pasándole las coordenadas del mismo. Este mapa es completamente interactivo, pudiendo desplazarse por él y mover el centro de atención a un punto de interés determinado.

Se ha elegido el API de **OpenStreetMap** ya que esta es de uso libre y soportada por su comunidad de colaboradores. Se barajaron diversas posibilidades, como hacer uso del API de Google Maps, pero esta no es gratuita, si no que se realizan cobros por cada llamada que se realiza sobre ella.

De igual forma, si la consulta es realizada sobre el *topic* Recogida de Basura, se podrán observar modificaciones en la vista dentro de la columna de **Datos Recibidos**, pero no así en las de **Datos del Punto de Acceso** o de **Localización,** las cuales solo cambiarán su contenido (si es necesario) pero no su estructura:

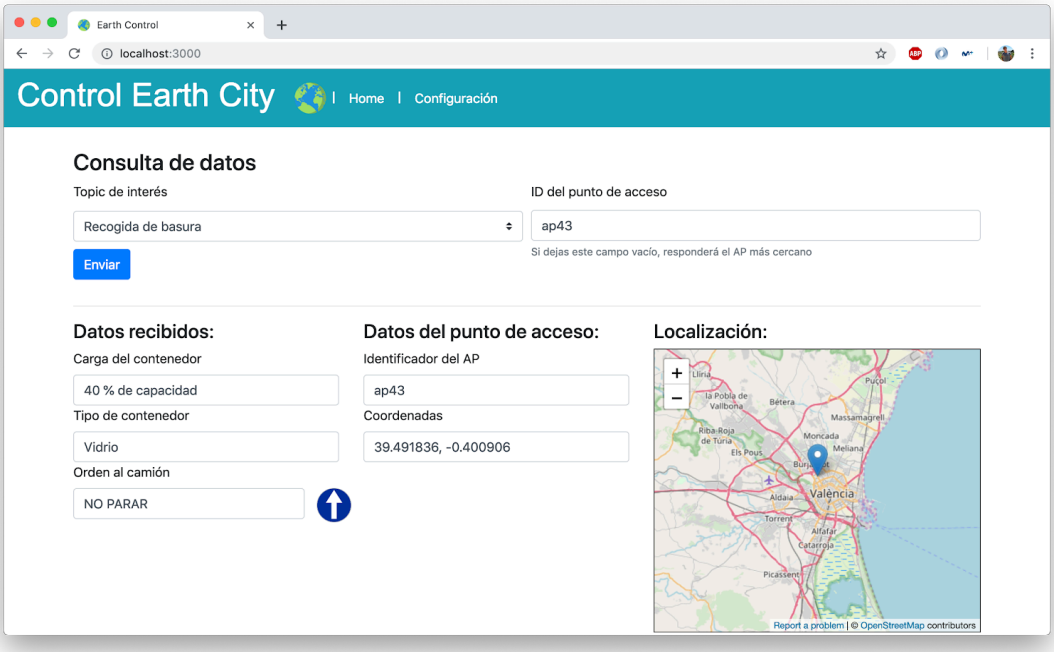

Imagen 5.38. Información mostrada en la pantalla Home tras realizar una consulta sobre el *topic* Recogida de Basura

La particularidad del caso de recogida de basura, es que los *subtopics* disponibles para consultar son “load” y “type”, pero además de la información relativa a estos llegada desde el punto de acceso, su muestra una más, **Orden al camión**. Esta información es calculada a partir de la carga que se haya recibido desde el punto de acceso, siendo el umbral a partir del cual el camión debe parar o no configurable.

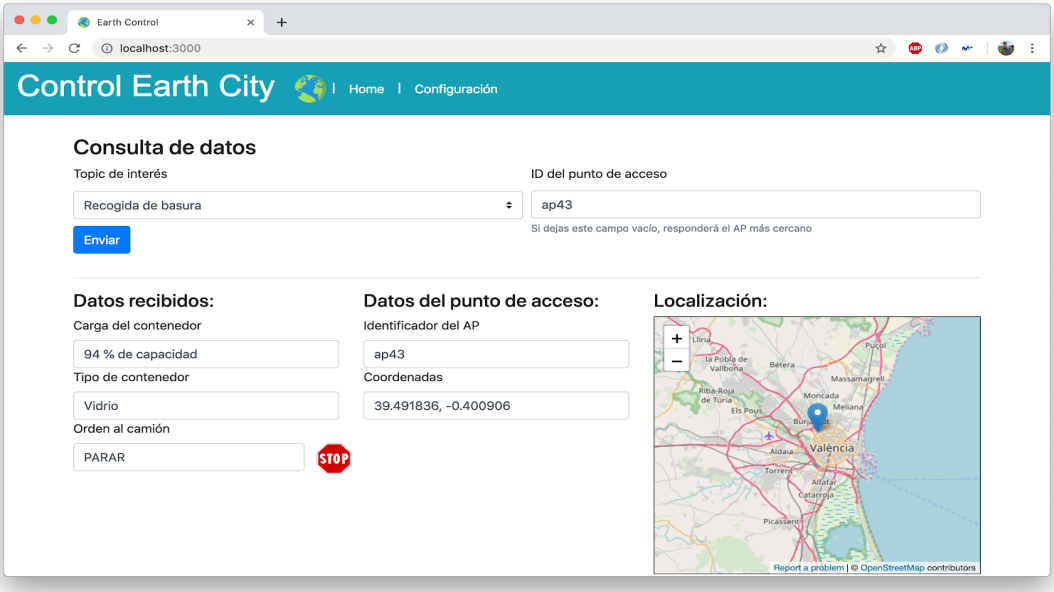

Imagen 5.39. Ejemplo de consulta sobre el *topic* Recogida de basura en el cual se decide que el camión debe parar

El caso en el que se realice la consulta sobre el *topic* Límite de velocidad será similar a los anteriores, mostrando la información relativa al tipo de vía en el que se encuentra situado el punto de acceso y el límite de velocidad de la misma:

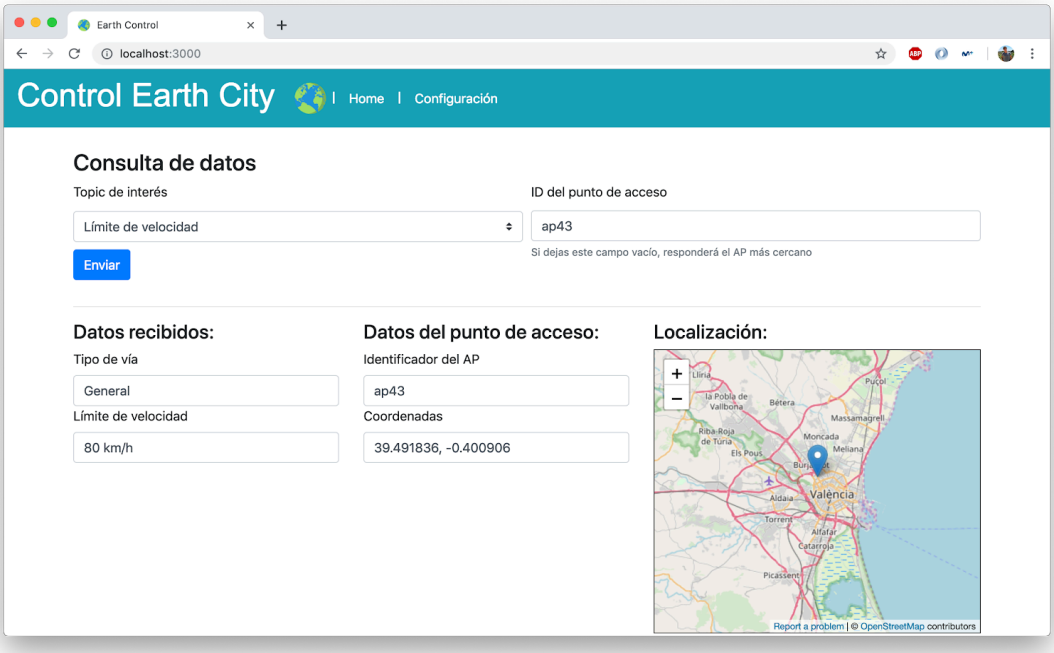

Imagen 5.40. Información mostrada en la pantalla Home tras realizar una consulta sobre el *topic* Límite de velocidad

Descrito el diseño y funcionalidad de la pantalla **Home** de la web desarrollada, podemos pasar a describir el funcionamiento de la pantalla **Configuración.** Como se ha comentado en ocasiones anteriores, esta pantalla permite configurar remotamente el estado de un punto de acceso dentro de una ciudad inteligente conociendo su identificador.

Teniendo en cuenta la funcionalidad, se ha diseñado la pantalla para que cuente con los elementos necesarios para poder construir un mensaje de tipo PUSH_CONFIG satisfactoriamente. Así, se ha añadido un desplegable que permite escoger el *topic* sobre el que se desean realizar los cambios en el punto de acceso, un campo de texto para indicar el identificador del punto de acceso sobre el que se realizará la modificación y campos de texto para indicar el *subtopic* a modificar y el nuevo valor que se quiere insertar en el mismo.

Una vez realizado el cambio por el punto de acceso, este construirá el mensaje de confirmación de que ha realizado la configuración, el cual será recibido por el servidor web, que mostrará en pantalla un mensaje que lo indique al usuario.

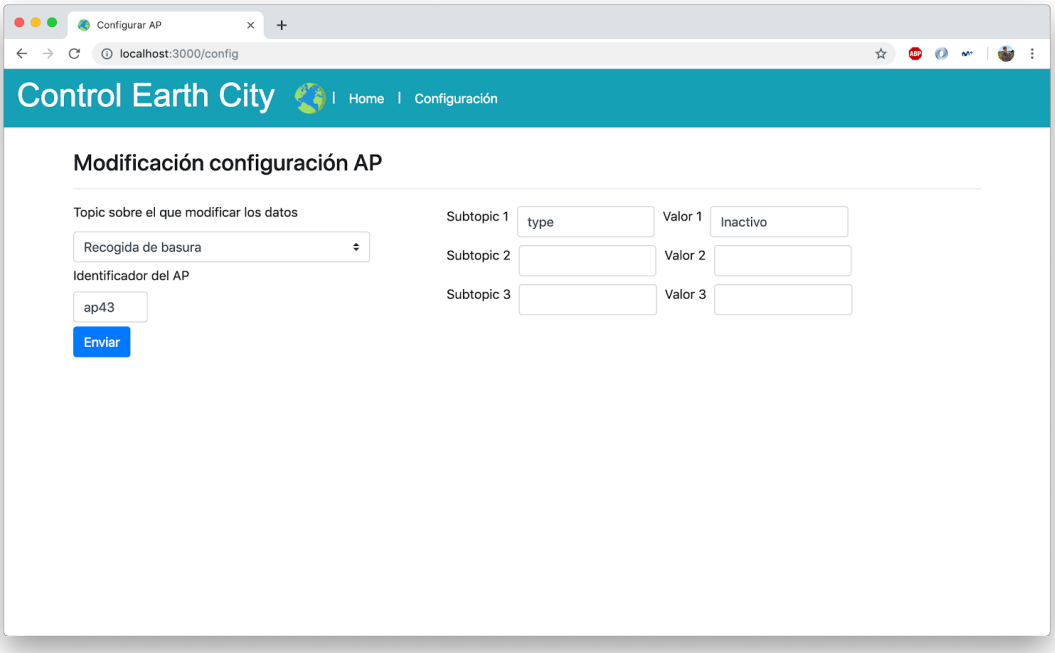

Imagen 5.41. Pantalla de Configuración. Se procede a realizar configuración sobre el ap43 cambiando el tipo de contenedor de basura a inactivo.

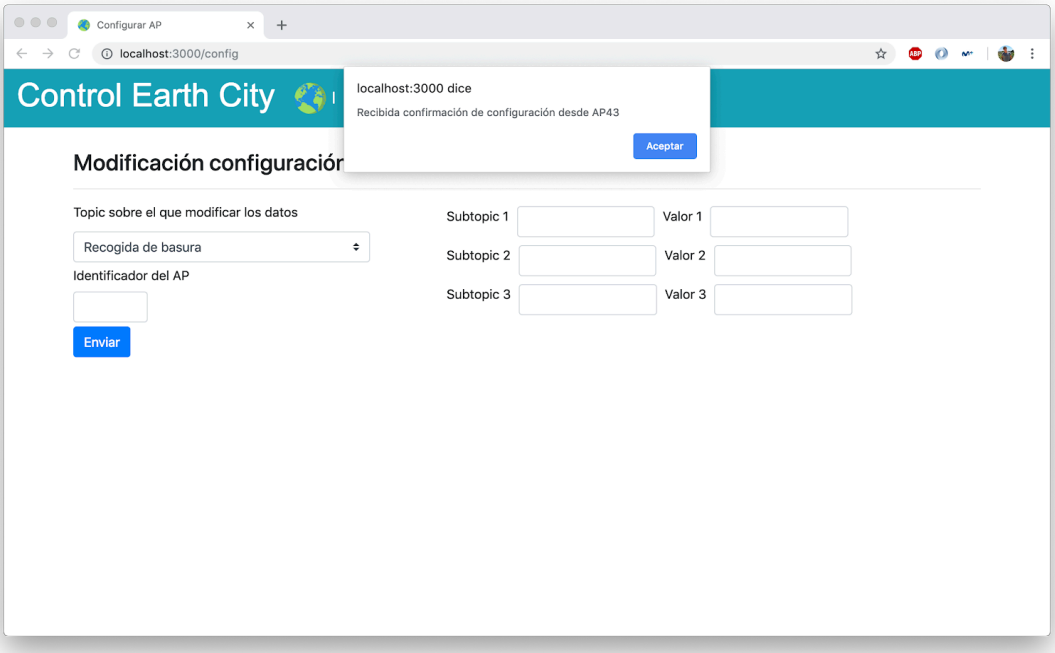

Imagen 5.42. Recepción de la confirmación de configuración desde el AP

5.4.2.2 Lógica del servidor

El servidor actúa como intermediario entre la interfaz web y las LoPy cuya función es ser puntos de acceso. Además, se encarga de escuchar los eventos producidos en la página y tratarlos, así como de mostrar la información obtenida de los AP y cargar y refrescar las páginas mostradas al usuario.

Debido a que su implementación, realizada en lenguaje JavaScript es extensa, se van a mostrar sólamente algunos detalles que permitan comprender su funcionamiento, sin entrar en todo el detalle del desarrollo realizado.

En nuestro servidor se pueden destacar dos acciones diferenciadas: la petición de una página y el envío de un formulario. La petición de la página se refiere a la carga de la misma. Este evento se produce cuando un usuario accede por ejemplo a la pantalla **Home.** Esta se encuentra en la ruta '/' dentro del servidor, por lo que cuando se recibe una petición de carga de esta, se ejecuta el *listener* **app.get('/',...)**. Este *listener* ha sido aprovechado para "dibujar" el contenido de la página mediante el método **render** y enviar el valor de las variables necesarias a la vista, que serán tratadas por EJS.

```
app.get('/', function (req, res) {
  res.render('index', {time_remaining: time_remaining,
                       show_semaphore: show_semaphore,
                       selected_sem: selected_sem,
                       selected_gar: selected_gar,
                       selected_lim: selected_lim,
                       state:state,
                       pedestrians:pedestrians,
                       load: load,
                       type: type,
                       road: road,
                       limit: limit,
                       ap_id: ap_id,
                       coordinates: coordinates,
                       subtopics: subtopics,
                       apID: apID
                       })
})
```

Imagen 5.43. Listener de la petición de carga de la página principal

El otro evento que se puede destacar es el envío de un formulario. Cuando el botón de un formulario es pulsado, se ejecuta la acción en el servidor para la que este fue configurado. En el caso del formulario de la página principal, se ha indicado que al pulsar el botón, la acción ejecutada debe ser '/'.

```
<h3>Consulta de datos</h3>
<div class="form-row">
    <div class="form-group col">
    <form action="/" method="post">
      <label for="topics" id="label_topics" style="display:block">Topic de interés</label>
      <select class="custom-select my-1 mr-sm-2" name="topics">
        <% if(selected_sem){ %>
          <option value="traffic_lights" selected>Semáforos</option>
          <option value="garbage_load">Recogida de basura</option>
          <option value="speed_limit">Límite de velocidad</option>
```

Imagen 5.44. Cabecera formulario de la página Home

De esta manera, el método que escucha los eventos sobre este formulario tiene la forma **app.post('/', …).** Este método es el encargado de recoger la información que se ha proporcionado en los campos del formulario y de construir y enviar el mensaje de petición al AP correspondiente. Además será el encargado de cambiar el valor de las variables lógicas que permiten modificar la vista para mostrar determinados campos con el resultado de la petición u otros en función del *topic* seleccionado.

```
app.post('/', function (req, res) {
  let topic = req.body.topics;
  apID = req.body.ap_ID;
  console.log(topic, apID);
  if(topic == 'traffic_lights'){
    selected_sem = true;
    selected_gar = false;
    selected_lim = false;
  }
  else if(topic == 'garbage_load'){
    selected_sem = false;
    selected_gar = true;
    selected_lim = false;
  }
  else if(topic == 'speed_limit'){
    selected_sem = false;
    selected_gar = false;
    selected_lim = true;
  }
```

Imagen 5.45. Listener para el evento generado por el formulario de la página Home

Un último tipo de evento que cabe destacar es el de recepción de mensajes por parte del servidor. Estos mensajes llegarán desde los puntos de acceso, siendo respuestas a los mensajes enviados por el servidor. Serán recogidos desde el método *listener* **server.on('message', ...).** Este método será el encargado de tratar los mensajes recibidos para obtener la información de los mismos, la cual será mostrada en la próxima petición de carga de las páginas mediante los métodos **app.get('/',...)** o **app.get('/config',...)** en el caso de la pantalla de configuración.

```
server.on('message', function (message, remote) {
  if(remote.address!=ipAddress){
    var obj = JSON.parse(message);
    if(obj['type']=='CONFIRM_AP'){
      apAddress = obj.ap_ip;
      ap_id = obj.ap_id;
      coordinates = obj.coordinates;
      subtopics = obj.subtopics;
```

Imagen 5.46. Listener para el evento de recepción de mensajes desde las LoPy

5.5 Otros aspectos de la implementación

5.5.1 Algoritmo de haversine

El algoritmo de haversine permite el cálculo de la distancia entre dos coordenadas de la forma (latitud, longitud) con una precisión de pocos metros. En este trabajo se ha realizado la implementación en Python del mismo para determinar la distancia en metros entre el punto de acceso y el cliente que solicita información.

```python
#Distancia entre dos coordenadas
def haversine(coord1, coord2):
    R = 6372800  # Radio de la tierra en metros
    lat1, lon1 = coord1
    lat2, lon2 = coord2

    phi1, phi2 = math.radians(lat1), math.radians(lat2)
    dphi       = math.radians(lat2 - lat1)
    dlambda    = math.radians(lon2 - lon1)

    a = math.sin(dphi/2)**2 + \
        math.cos(phi1)*math.cos(phi2)*math.sin(dlambda/2)**2

    return 2*R*math.atan2(math.sqrt(a), math.sqrt(1 - a))
```

Imagen 5.47. Implementación del algoritmo haversine

En primer lugar se realiza la conversión a radianes de las latitudes de las coordenadas una y dos pasadas como argumento. También se realiza la conversión a radianes de las diferencias entre la latitud 2 y la 1 y la longitud 2 y 1.

Se calcula el valor a mediante la fórmula:

$$a = \frac{sin(\frac{dphi}{2})^2}{cos(phi1)*cos(phi2)*sin(\frac{dlambda}{2})^2}$$

De esta manera, la distancia entre dos puntos será:

$$haversine = 2 * R * tan(\sqrt{a}, \sqrt{(1-a)})$$

5.5.2 Obtención de valores aleatorios

Debido a que los dispositivos LoPy con los que se trabaja no son ordenadores de grandes capacidades, ciertas acciones que en los lenguajes convencionales de hoy en día son muy sencillas, no son posibles. Es el caso de la obtención de números aleatorios, la cual se podría realizar en el lenguaje Python mediante una simple llamada al paquete *random*.

En nuestro caso, esta opción no se encuentra disponible, por lo que se ha tenido que recurrir a obtener valores aleatorios mediante el uso directo de la CPU. De esta manera, se han generado valores de 32 bits que han sido pasados a un entero convencional mediante conversiones.

Además, se ha generado el método RandRange para obtener un valor aleatorio dentro de un rango pasado como argumento.

```
def Random():
    r = crypto.getrandbits(32)
    return ((r[0]<<24)+(r[1]<<16)+(r[2]<<8)+r[3])/4294967295.0

def RandomRange(rfrom, rto):
    return Random()*(rto-rfrom)+rfrom
```

Imagen 5.48. Aleatorización y aleatorización dentro de un rango

5.6 Pruebas

Para comprobar el correcto funcionamiento de la funcionalidad desarrollada para este proyecto, se ha procedido a realizar las siguientes pruebas sobre los distintos módulos:

- **Pruebas funcionales:** han permitido verificar el correcto desempeño por parte del código de la funcionalidad para la que ha sido desarrollado. Ayuda a detectar posibles errores cometidos a la hora de implementar la solución
- **Pruebas de rendimiento:** se han utilizado para medir los tiempos necesarios para enviar y recibir mensajes entre dispositivos LoPy haciendo uso del protocolo implementado, evaluar el número de llamadas que puede admitir un punto de acceso simultáneamente o conocer los tiempos de carga de la página web desde que se realiza una solicitud a un punto de acceso hasta que esta muestra la información recibida.

Para realizar las pruebas funcionales se ha optado por realizar tests unitarios manualmente sobre los diferentes métodos desarrollados para implementar la solución descrita en esta memoria. De este modo, se pudieron advertir errores en la estructura de los mensajes, datos devueltos o inconsistencias en algunos resultados tras realizar cálculos en los dispositivos LoPy los cuales fueron solucionados posteriormente. Se pudo comprobar que fueron solucionados realizando los mismos tests unitarios sobre los métodos.

En cuanto a las pruebas de rendimiento se muestra en la tabla a continuación los datos obtenidos en cuanto a medición de tiempos y número máximo de peticiones a un punto de acceso posibles:

Prueba	Resultado
Tiempo de respuesta a mensaje IS_AP	41ms
Tiempo de respuesta a mensaje GET_INFO	103ms
Tiempo total de una comunicación	236ms
Tiempo carga web	890ms
Número máximo peticiones AP	5(*)

Tabla 5.1 Resultados obtenidos en las pruebas de rendimiento y carga

(*) Este dato se ve limitado por la arquitectura del HW, limitada para mejorar el consumo energético

6. Retos y trabajo futuro

Como se ha comentado en diferentes puntos de este trabajo, es sumamente importante que los datos que se manejan en las aplicaciones del Internet de las Cosas y más concretamente en el ámbito de las ciudades inteligentes se encuentren abiertos y disponibles a toda la ciudadanía de forma que sean accesibles, ofreciendo también los administradores del sistema una plataforma que permita visualizarlos de manera sencilla. Esto se consigue mediante herramientas de visualización de datos que los muestren en informes tras realizar análisis estadísticos sobre ellos.

Proveer esta apertura de los datos recogidos supone un gran reto a la hora de asegurar la privacidad, haciendo necesaria la anonimización de los mismos. Además, que todos los objetos estén conectados a Internet supone un gran reto a la hora de asegurar tanto su seguridad física cómo lógica, siendo estos el objetivo de ataques informáticos día tras día.

Un ejemplo de estos ataques es Mirai, un *malware* que utiliza una botnet para infectar *routers* y dispositivos del Internet de las Cosas, principalmente cámaras IP. Su funcionamiento es sencillo, escanea redes periódicamente en busca de dispositivos IoT a los cuales intenta conectarse mediante telnet. Mirai aprovecha que los usuarios que han instalado los dispositivos no suelen cambiar las contraseñas que vienen por defecto en los mismos, pudiendo así conectarse y tomar el control.

Estos dispositivos infectados se unen a la botnet y permiten seguir infectando más dispositivos. El objetivo de estas infecciones es contar con dispositivos que, a pesar de su baja capacidad de cómputo, pueden realizar peticiones masivas a servidores simultáneamente desde diversos puntos del planeta, efectuando así un ataque de denegación de servicio distribuido o DDoS (Distributed Denial of Service).

La botnet creada por Mirai fue utilizada entre otros en el ataque en octubre de 2016 al **servidor de nombres de dominio** Dyn, dejándolo inutilizado mediante diversos ataques de denegación de servicio desde puntos distribuidos por todo el planeta. De esta manera no se pudo acceder a diversos servicios de Internet durante algo más de dos horas. Posteriormente se produjeron dos ataques más el mismo día.

Cámaras como estas son *hackeadas* a diario dejando expuesta la privacidad de las personas que las instalan en sus hogares o de los ciudadanos de aquellas ciudades inteligentes que deciden instalarlas para mejorar la seguridad en sus calles. Basta una simple visita a páginas web como www.insecam.org para visualizar cámaras expuestas por los piratas informáticos.

Como se puede observar, una simple mala configuración de un dispositivo del Internet de las Cosas puede afectar a la seguridad y privacidad de todo aquel que interactúe con él o cuyos datos circulen por la red.

Es por ello por lo que uno de los grandes retos a los que se enfrenta el Internet de las Cosas es estandarizar los requisitos mínimos de seguridad con los que debe contar un dispositivo que es instalado en un hogar, en una ciudad o cualquier lugar

público o privado el cual vaya a tratar con datos sensibles o pueda invadir la privacidad de las personas si es vulnerado por un atacante.

Además, es indispensable concienciar a todos los usuarios del peligro de instalar dispositivos IoT sin ser configurados correctamente, ya que como hemos podido ver esta ha sido una de las causas por las que los atacantes han podido utilizarlos para realizar ataques.

Otro punto importante que tratar es asegurar la privacidad de las personas mediante la anonimización de los datos publicados en la red para asegurar la transparencia de los sistemas sin dejar de lado la seguridad. Sistemas como las plataformas de ciudades inteligentes proveen APIs que permiten consultar los bancos de datos recogidos en la ciudad durante un período de tiempo.

Estos datos pueden ser por ejemplo demográficos, indicando el sexo de las personas, edades, etc. También existen bancos de datos sanitarios, en los que se indica la incidencia de ciertas enfermedades en la población, plagas, etc. Algunos de estos datos se consideran sensibles, por lo que debe asegurarse que no sea posible relacionar un dato con la persona a la que pertenece.

Se entiende por lo tanto que una continuación de este trabajo podría ser securizar el protocolo de comunicaciones propuesto, añadiendo una capa de cifrado que permitiera que los datos enviados por la red pudieran ser incomprensibles para cualquier atacante que escuchara el canal. De esta manera, podría ser utilizado en aplicaciones en las cuales la seguridad es muy importante, como compartición rápida de documentos, intercambio de datos sanitarios y un largo etcétera.

También puede ser útil ampliar este trabajo añadiendo más casos de uso que pudieran ayudar a una ciudad inteligente a ser más eficiente, así como ampliar las funcionalidades desarrolladas para la sala de control web, como mostrar histogramas y gráficas con datos recogidos por el sistema anteriormente.

7. Conclusiones

Habiendo finalizado el contenido de esta memoria se puede concluir que los objetivos que se propusieron para este Trabajo de Fin de Máster han sido completados satisfactoriamente. El primero, diseñar un protocolo de comunicación para *Smart Cities* que permitiera la comunicación entre dispositivos actuando al menos uno de ellos como baliza, ha sido cumplido en el capítulo 4 de esta memoria. En el capítulo citado se expone la estructura de los diferentes mensajes necesarios para permitir la comunicación, reuniendo conocimientos de protocolos populares como los de descubrimiento de vecinos siendo un ejemplo de estos el del RFC 4861.

El segundo objetivo que se planteó fue el de obtener una solución lo suficientemente independiente del ámbito de uso para que el protocolo pudiera ser utilizado en cualquier situación y permitiera la integración con otros dispositivos. Este objetivo se ha cumplido al tener en cuenta estas necesidades en la fase de diseño, permitiendo asegurar que los mensajes creados eran completamente independientes del contexto y tomando como referencia protocolos ya existentes en la actualidad. De esta manera se consigue comunicar dispositivos cuyas arquitecturas son completamente distintas. Esto se considera un logro significativo, ya que como se ha comentado en diversos puntos de este trabajo, uno de los retos que debe afrontar el Internet de las Cosas es conseguir que todos los elementos que lo conforman sean interoperables y compatibles entre sí independientemente de sus arquitecturas.

El siguiente objetivo establecido fue construir un prototipo que permitiera probar el funcionamiento correcto y la utilidad del protocolo propuesto. Este objetivo ha sido logrado, tal y como se muestra en el punto 5.4.1 de esta memoria con la implementación del mismo, haciendo uso de tecnologías diversas que han supuesto un reto a la hora de conseguir la integración entre ellas debido a sus diferencias notables. A pesar de estas diferencias, la robustez y sencillez del protocolo propuesto permitió la comunicación entre dispositivos cuando se superó el aprendizaje necesario sobre las diferentes tecnologías utilizadas.

Además de la comunicación entre los dispositivos electrónicos haciendo uso del protocolo propuesto, también se planteó como objetivo implementar una aplicación web que actuara como sala de control para la monitorización del estado de los dispositivos que se encontraran ubicados en la ciudad. Este objetivo se materializó haciendo uso de tecnologías como Node.js y EJS. Los detalles de esta implementación han sido descritos en el punto 5.4.2 de esta memoria.

Los últimos tres objetivos fueron planteados con la intención de obtener de este trabajo un manual de referencia que proporcionara una base teórica además de práctica sobre el ámbito del Internet de las Cosas y las Ciudades Inteligentes para que pudiera ser utilizado en trabajos futuros sobre esta materia. Para ello se han tomado diferentes artículos científicos y libros que han permitido resumir en pocas páginas la información necesaria para adentrarse en este ámbito de investigación, lográndose de esta manera cumplir estos objetivos.

8. Bibliografía

[1] Ashton, K. (2009). That 'Internet of Things' Thing. RFID Journal

[2] Badirova, A. (2018). Security and Privacy in Internet of Things. Applied Computer Science Georg-August-University Goettingen, Germany

[3] Diccionario Real Academia Española. DEL: Lista de entradas – Diccionario de la lengua española – Edición del Tricentenario. [Varias fechas de consulta]. Disponible en: http://dle.rae.es/

[4] Dirección General de Calidad Ambiental. (2017). Plan Integral de Residuos de la Comunitat Valenciana - MEMORIA DE INFORMACIÓN

[5] Dirección General de tráfico (2018). Estadísticas e indicadores de accidentes con víctimas. Extraído de: http://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/accidentes-30dias/tablas-estadisticas/. Fecha: 12/03/2019

[6] Domoticz. (2019). Extraído de: http://www.domoticz.com/. Fecha: 21/05/2019

[7] EJS - Embedded JavaScript templates. (2019). Extraído de: https://ejs.co/. Fecha: 10/03/2019

[8] El mapa que evidencia cómo España es el país más despoblado del sur de Europa. (2018). Extraído de: http://www.radiocable.com/mapa-esp-mas-despoblado-sur-ue961.html. Fecha: 02/04/2019

[9] Fundación Telefónica (2011). Smart Cities: Un primer paso hacia el internet de las cosas

[10] Generalitat Valenciana - Consellería de Infraestructuras, territorio y medio ambiente. (2016). Residuos Urbanos

[11] Gergely, O. (2018). Living in a Smart Home: A qualitative approach. Sapientia Hungarian University of Transylvania

[12] GVA Oberta - Generalitat Valenciana. (2019). Extraído de: http://www.gvaoberta.gva.es/es. Fecha 08/04/2019

[13] Haciendo IoT con LoRa: Capítulo 1.- ¿Qué es LoRa y LoRaWAN? (2019). Extraído de: https://medium.com/beelan/haciendo-iot-con-lora-cap%C3%ADtulo-1-qu%C3%A9-es-lora-y-lorawan-8c08d44208e8. Fecha: 28/03/2019

[14] Hall, P. (1988). Cities of tomorrow: an intellectual history of urban planning and design. Oxford: Blackwell Publishing.

[15] Hernández, Ó. (2019). El metro de Barcelona informará en los andenes si los vagones llegan más o menos llenos. Extraído de: https://www.elperiodico.com/es/barcelona/20190326/metro-informara-anden-llega-lleno-7375515. Fecha 04/04/2019

[16] Herman, M., Pentek, T., Otto, B (2016). Design principles for industry 4.0. 2016 49th Hawaii International Conference on System Sciences (HICSS). doi: 10.1109/HICSS.2016.488

[17] INFSO D.4 Networked Enterprise RFID INFSO G.2 Micro Nanosystems in Co-operation with the Working Group RFID of the ETP EPOSS. Internet of Things in 2020, Roadmap for the Future, Version 1.1. Technical report, 27 May 2008.

[18] Instituto Nacional de Estadística (INE). (2018). Avances: Superficies y producciones de cultivos. Enero 2018.

[19] Instituto Nacional de Estadística (INE). (2018). Nota de prensa: Estadísticas sobre generación de residuos en España del año 2016.

[20] Kamilaris, A., Pitsillides, A. (2013). "Towards Interoperable and Sustainable Smart Homes", Proceedings, Paul Cunningham and Miriam Cunningham (Edc) IIMC International Information Management Corporation.

[21] Komninos, N. (2018). Smart Cities. In Warf, B. (ed.) The SAGE Encyclopedia of the Internet, 783-789. Sage Publications. DOI: http://dx.doi.org/10.4135/9781473960367.n229

[22] Kumari, D., Pandita, R., & Mittal, M. (2018). An Agricultural Perspective in Internet of Things. International Journal Of Computer Sciences And Engineering, 06(05), 107-110. doi: 10.26438/ijcse/v6si5.107110

[23] Ley 7/1985, de 2 de abril, Reguladora de las Bases del Régimen Local. Boletín Oficial del Estado, 80, de 3 de abril de 1985, 8945 a 8964. Extraído de: https://www.boe.es/boe/dias/1985/04/03/pdfs/A08945-08964.pdf

[24] Mark Otto, A. (2019). BootStrap. Extraído de: https://getbootstrap.com/. Fecha: 10/03/2019

[25] MicroPython. (2019). Extraído de: https://en.wikipedia.org/wiki/MicroPython. Fecha: 09/03/2019

[26] Ministerio para la transición ecológica. (2016). Registro de huella de carbono, compensación y proyectos de absorción de dióxido de carbono. Informe de huella de carbono. Empresa: S.A. AGRICULTORES DE LA VEGA DE VALENCIA

[27] MisterHouse. (2017). Extraído de: http://misterhouse.sourceforge.net/. Fecha: 21/05/2019

[28] Mycroft – Open Source Voice Assistant. (2019). Extraído de: https://mycroft.ai/. Fecha: 21/05/2019

[29] Muñoz-Cañavate, A., & Hípola, P. (2011). Electronic administration in Spain: From its beginnings to the present. Government Information Quarterly, 28(1), 74-90. doi: 10.1016/j.giq.2010.05.008

[30] Narten, T., Nordmark, E., Simpson, W. & Soliman, H. (2007). Neighbor Discovery for IP version 6 (IPv6) - RFC 4861

[31] Node.js Foundation. (2019). Node.js. Extraído de: https://nodejs.org/es/. Fecha: 09/03/2019

[32] Osset, E. (2019). El ENIAC un pionero de los computadores - Museo de Informática ETSINF. Extraído de: http://museo.inf.upv.es/es/eniac/. Fecha 04/04/2019

[33] PAe - Portal de Administración Electrónica. (2019). Extraído de: https://administracionelectronica.gob.es. Fecha 08/04/2019

[34] Piyare, R., Murphy, A., Magno, M., & Benini, L. (2018). On-Demand LoRa: Asynchronous TDMA for Energy Efficient and Low Latency Communication in IoT. Sensors, 18(11), 3718. doi: 10.3390/s18113718

[35] Pycom - Next generation Internet of Things Platform. (2019). Extraído de: https://pycom.io/. Fecha: 09/03/2019

[36] Roblek, V., Meško, M., & Krapež, A. (2016). A Complex View of Industry 4.0. SAGE Open, 6(2), 215824401665398. doi: 10.1177/2158244016653987

[37] Ros Gil, J. (2017). Desarrollo de un caso de estudio para la interacción entre humanos y robots móviles influida por las emociones. (Trabajo de fin de grado). Universidad Politécnica de Valencia.

[38] Salinas, A. (2016). The Internet of Things: Business Applications, Technology Acceptance and Future Prospects. Dissertation of the Julius Maximilian University of Würzburg to obtain the title of Doctor rerum politicarum.

[39] The Fourth Industrial Revolution: what it means and how to respond. (2019). Extraído de: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-
means-and-how-to-respond/. Fecha: 28/03/2019

[40] Tonguz, O. (2018). Red light, green light - No light. Tomorrow's communicative cars could take turns at intersections. IEEE Spectrum, 55(10), 24-29. doi: 10.1109/mspec.2018.8482420

[41] València Ciutat Intel·ligent / Smart City | Ajuntament de València. (2019). Extraído de: http://smartcity.valencia.es. Fecha 09/04/2019

[42] Veizades, J., Perkins, C., Kaplan S. (1997). Service Location Protocol - RFC 2165

[43] Zhang, R. et al. (2018). Virtual Traffic Lights: System Design and Implementation. Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh.