UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación de la IoT al ámbito del transporte. Auto-gestión del tráfico de vehículos inteligentes

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Carlos Izquierdo Martínez

Tutor: Joan Fons i Cors

Curso 2016 – 2017

Resum

En aquest projecte es dissenyarà la infraestructura, basada en l'ús de plataformes IoT, per a millorar les comunicacions entre els vehicles intel·ligents i les carreteres per les quals circulen. L'objectiu és permetre que aquests vehicles intel·ligents puguen comunicar-se amb la infraestructura viària (carreteres, senyals de tràfic, semàfors e incidències) i amb altres vehicles de manera autònoma.

D'aquesta forma, s'aconsegueix que les ciutats puguen oferir diferents servicis als usuaris dels vehicles i que puguen autoregular-se per adaptar-se a les diferents necessitats com la de la situació actual del tràfic o emergències e incidències en les carreteres. Es dissenyarà una solució que contemple els requisits i es desenvoluparà un prototip funcional per a validar la proposta.

Com a escenari, es configuraran diferents situacions que permetran mostrar com la ciutat va donant suport a les seues necessitats canviants i s'adapta per a realitzar canvis en la seua infraestructura viària o inclòs imponents restriccions de circulació. En el cas dels vehicles, gestionaran la informació que rebran de les carreteres per a tomar les seues decisions i complir les noves condicions de la ciutat.

Paraules clau: Internet de les cosses, Ciutats intel·ligents, Vehicles autònoms, Gestió autònoma del tràfic, Sistemes auto adaptatius

Resumen

En este proyecto se diseñará la infraestructura, basada en el uso de plataformas IoT, para mejorar las comunicaciones entre los vehículos inteligentes y las carreteras por las que circulan. El objetivo es permitir que estos vehículos inteligentes puedan comunicarse con la infraestructura vial (carreteras, señales de tráfico, semáforos, incidencias) y con otros vehículos de manera autónoma.

De esta forma, se consigue que las ciudades puedan ofrecer diferentes servicios a los usuarios de los vehículos y puedan autorregularse para adaptarse a diferentes necesidades como la situación actual del tráfico o emergencias e incidencias en sus carreteras. Se diseñará una solución que cumpla los requisitos y se desarrollará un prototipo funcional para validar la propuesta.

Como escenario, se configurarán diferentes situaciones que permitan mostrar como la ciudad va dando soporte a sus necesidades cambiantes y se adapta para realizar cambios en su infraestructura vial o imponer restricciones de circulación. En el caso de los vehículos, gestionan la información que reciben de las carreteras para tomar sus decisiones y cumplir las nuevas condiciones de la ciudad.

Palabras clave: Internet de las cosas, Ciudades inteligentes, Vehículos autónomos, Gestión autónoma del tráfico, Sistemas auto adaptativos

Abstract

This project will design the infrastructure, based on the use of IoT platforms, to improve communications between smart vehicles and the roads thought which they circulate. The main goal is to enable these smart vehicles to communicate with the road infrastructure (roads, traffic signals, traffic lights, incidents, etc) and with other vehicles on an autonomous basis.

In this way, we get that cities can offer different services to users and can manage itself to adapt to the different needs such as the current traffic situation or emergencies and incidents in the city. A solution that accomplish the requirements will be designed and a functional prototype will be developed to validate the proposal.

As the scenario, different situations will be configured to show how the city is supporting the needs and adapts itself to make changes in its roads infrastructure or imposing traffic restrictions. In the case of the smart vehicles, they manage the information they receive from the city to make their decisions and accomplish the new conditions imposed by the city.

Key words: Internet of things, Smart cities, Autonomous vehicles, Autonomous traffic management, Self-adaptive systems

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

El crecimiento de la población en las ciudades es un hecho que no ha parado desde que comenzó con la revolución industrial. Actualmente, la mitad de la población ya vive en centros urbanos y se espera que las ciudades crezcan aún más. Este crecimiento demandará una maximización y expansión de las infraestructuras y servicios que actualmente proporcionan las ciudades, incluso más allá de sus capacidades y que obliga a buscar nuevos modelos de gestión.

El concepto ***Smart City*** (en castellano Ciudad Inteligente) surge y se define como aquella ciudad que usa las **tecnologías de la información y las comunicaciones** (TIC) para relacionarse con su entorno y proporcionar nuevos servicios o soluciones tecnológicas avanzadas para facilitar la interacción con sus ciudadanos, haciendo su vida más fácil.

El gran reto que se plantea en el proyecto es el de cómo gestionar el problema de la movilidad en las ciudades que es cada vez más acuciante. Uno de los mayores problemas en el ámbito de la movilidad es la congestión del tráfico, que tiene un impacto negativo muy considerable en la calidad de vida de una ciudad tanto por la disminución de la productividad, como por el empeoramiento de la calidad del aire, así como por la contaminación acústica que conlleva.

Figura 1.1: Simplificación de una Smart City

1.1 Motivación y retos tecnológicos

El proyecto propone desarrollar soluciones auto adaptativas orientadas a mejorar la movilidad y el transporte de una *Smart City* funcionando sobre una plataforma de **Internet de las cosas** (IoT). Esta plataforma materializa la idea de la ciudad inteligente y ofrece la posibilidad de tener una gran cantidad de elementos físicos conectados entre sí como son los vehículos, calles, semáforos, y señalizaciones de tráfico.

La utilización de estas nuevas tecnologías es lo que ha motivado a la realización de un proyecto de estas características y ha supuesto un reto el poder profundizar, aprender y comprobar su funcionamiento y el gran potencial en el futuro para el desarrollo de aplicaciones que mejoren la vida de los seres humanos.

Internet de las cosas

La tendencia en los últimos años de conectar a la red cualquier cosa que se pueda imaginar ha aumentado enormemente y es lo que está garantizando el éxito a la que se llama "Internet del futuro".

La posibilidad de adjudicar de una capacidad informática a objetos cotidianos para que recojan datos de su entorno y se comuniquen entre ellos puede crear numerosas aplicaciones que pueden llegar a cambiar por completo la vida de las personas.

Smart Cities

La principal aplicación de la Internet de las cosas donde cada objeto y elemento físico esta interconectado y genera información a la ciudad. Estos objetos pueden ser entre otros coches, carreteras, señales de tráfico, semáforos o viviendas.

Mediante la implementación de servicios adaptables que monitoricen toda esta información que se genera y su evaluación, se puede conseguir dar cierta autonomía o inteligencia a la ciudad con capacidad para que se auto adapte al entorno cambiante que le rodea.

Comunicación M2M

El concepto de "*machine to machine*" se refiere al intercambio de información o comunicación en forma de datos entre dos máquinas remotas. Este tipo de comunicación reduce el tiempo de procesos, los costes y además, amplían servicios que hasta ahora no se tenían.

Estos equipos son capaces de comunicarse con otras máquinas para recibir o transmitir información y desencadenar una acción.

1.2 Objetivos del proyecto

El objetivo principal es extender la funcionalidad que ya ofrece la *Smart City* e incluir nuevos servicios independientes que sean capaces de auto-gestionarse para que se puedan adaptar dependiendo de la situación en todo momento de su entorno y aplicar reglas de actuación orientadas a mejorar la movilidad.

Se estructura en los siguientes puntos:

- Diseñar e implementar una infraestructura de servicios adaptables que sean capaces de proporcionar una inteligencia o carácter autónomo a varios sistemas de una *Smart City* y orientados a mejorar la movilidad y el transporte utilizando plataformas IoT como fuente de información.
- Proporcionar los mecanismos necesarios para que estos servicios se auto-gestionen mediante bucles de control *(feedback loops)* y actúen de forma complementaria a los otros sistemas, modificando su comportamiento sin alterar su implementación base.
- Desarrollar escenarios personalizados e implementar prototipos que demuestran la factibilidad y la utilidad de llevar a cabo este tipo de soluciones en escenarios reales.

1.3 Declaración de alcance

El proyecto como se ha comentado anteriormente parte de la base de una plataforma IoT que me provee el tutor del proyecto, donde se materializa la *Smart City* y donde se interconectan un conjunto de proyectos que integran las carreteras, la señalización, los semáforos, los incidentes y los vehículos de la ciudad.

El desarrollo se centra en proveer de nuevos sistemas y servicios con un carácter autónomo, inteligente y auto adaptativo sobre la gestión del tráfico real de una ciudad donde los vehículos se comunican con su entorno y son capaces de reaccionar a dicha información y tomar decisiones por sí mismos.

Es importante destacar el factor limitante de no tener acceso a la implementación de los diferentes sistemas base que componen la plataforma IoT y que la solución desarrollada se integra de forma separada sin modificar los sistemas ya existentes.

Estos nuevos servicios desarrollados mantienen su propio contexto y adoptan un comportamiento autónomo mediante la comunicación con otros sistemas. De esta forma, ante la posibilidad de que dejen de funcionar, lo único que se perdería sería el comportamiento autónomo en la ciudad sin afectar al comportamiento normal de los sistemas base.

La solución desarrollada muestra cuatro diferentes escenarios donde se puede observar con claridad los nuevos comportamientos de la ciudad cuando ocurren diferentes eventos que son analizados y evaluados por monitores para aplicar unas reglas de actuación.

Los tipos de eventos que se monitorizan en cada uno de los escenarios son:

- La aparición de accidentes en carreteras que pueden afectar de forma directa a otros sistemas como otras carreteras, vehículos que circulan por ellas o señalizaciones de tráfico.
- El aumento de la densidad de tráfico en una carretera y su posterior colapso o congestión que afecta al comportamiento de todos los sistemas que utilizan sus recursos.
- La aparición de un incidente o emergencia en un vehículo que tiene que detener su trayectoria de forma inesperada.
- El cambio entre el día y la noche en la *Smart City* para poder aplicar configuraciones especiales que permitan mejorar la calidad de los servicios o reducir el gasto público.

1.4 Metodología empleada y plan de trabajo

Debido al alto nivel de complejidad que requiere un proyecto de estas características en cuanto a su innovación y aprendizaje de nuevas tecnologías se ha seguido una metodología ágil que consiste en ir iterando sobre cada punto para obtener unos resultados.

La planificación del proyecto consta en cinco fases que se detallan a continuación.

- **Fase 1. Análisis y necesidades del proyecto:**

  Definición del proyecto, objetivos principales, motivación, declaración de alcance, tecnologías a utilizar y productos entregables.

- **Fase 2. Aprendizaje del caso de estudio y nuevas tecnologías:**

  Estudio de la plataforma IoT que integra todos los sistemas de la *Smart City* y sus comunicaciones.
  Aprendizaje de nuevas tecnologías de la internet de las cosas como comunicaciones M2M y MQTT, API REST de interoperabilidad, bucles de control MAPE-K y desarrollo de plugins mediante Java OSGi.

- **Fase 3. Análisis de requisitos:**

  Desarrollo de las ideas y análisis de requisitos de las soluciones a implementar y escenarios a prototipar.

- **Fase 4. Diseño e Implementación de las soluciones:**

  Desarrollo del diseño y la implementación del código pertinente de los escenarios para la plataforma IoT utilizando Java OSGi.

- **Fase 5. Pruebas de funcionamiento:**

  Pruebas funcionales sobre el desarrollo de los escenarios utilizando todo tipo de condiciones para comprobar su funcionamiento autónomo y adaptativo.

Se puede ver una estimación del tiempo empelado y un desglose de las tareas realizadas en el diagrama de Gantt que se muestra a continuación.

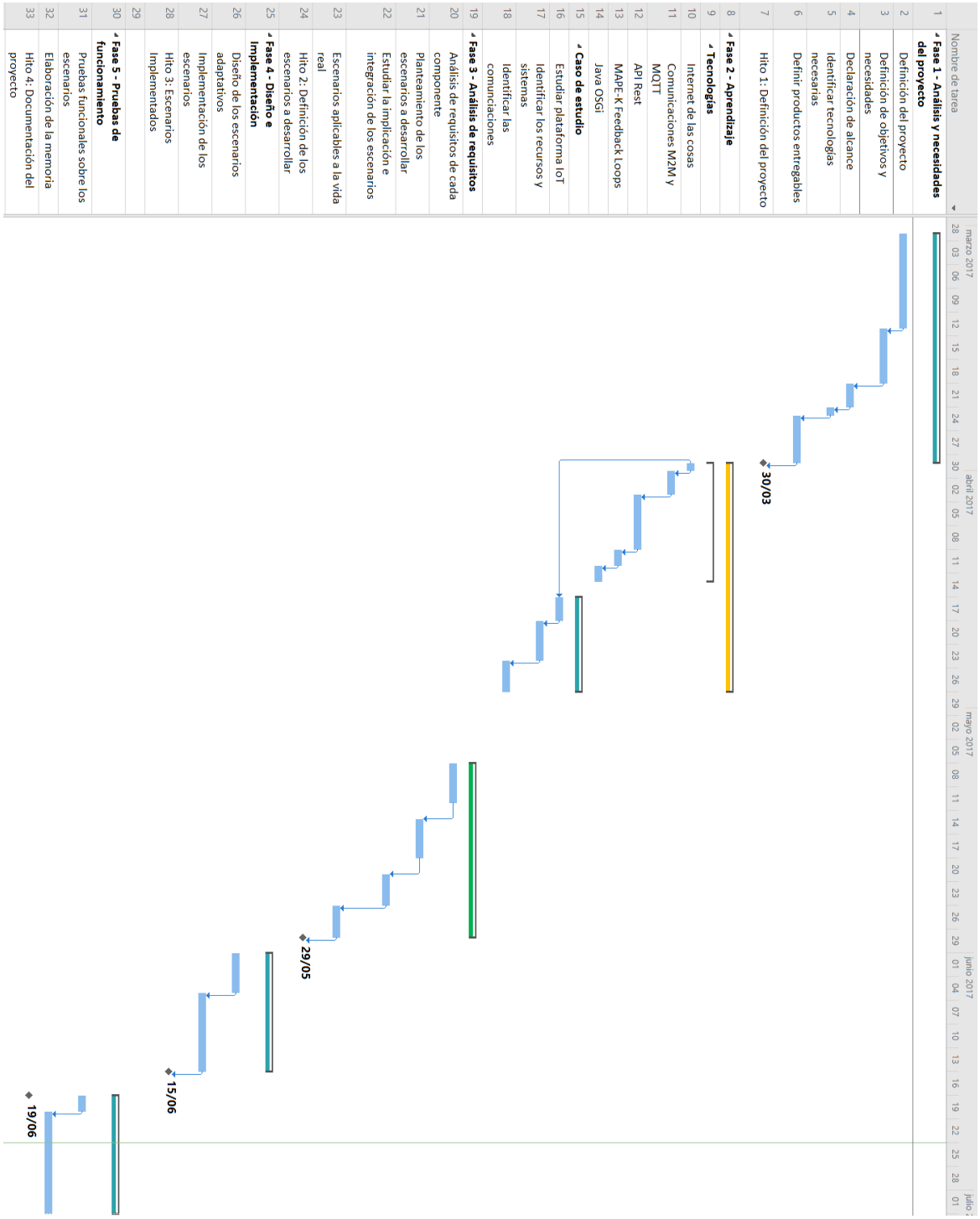

Figura 1.2: Diagrama de Gantt del proyecto

1.5 Organización de la memoria

En este apartado se pretende mostrar cómo se estructura y organiza el contenido de la memoria que se ha dividido en siete capítulos y un apéndice.

En el **capítulo 1 (Introducción)**, se define e introduce el tema del proyecto, se explica la motivación de los objetivos y la metodología que se ha empleado para la realización del trabajo. Se pretende introducir al lector en el contexto del proyecto y que comprenda el tema tratado dando referencias sobre las tecnologías y explicando las soluciones que se van a desarrollar.

En el **capítulo 2 (Contexto tecnológico)**, se centra en explicar detalladamente las tecnologías que se utilizan en el proyecto debido a que, por su complejidad, no son triviales y es necesario explicar su contexto para entenderlas. También se explican las herramientas, entornos y componentes que se han utilizado. Es una de las partes más importantes del trabajo ya que concentra todo el trabajo de investigación que se ha realizado.

En el **capítulo 3 (Caso de estudio)**, se describe un planteamiento inicial del problema y se explica de forma detallada el caso de estudio desde el que se parte para realizar el trabajo. En este caso, el caso de estudio explica el funcionamiento de la plataforma IoT que materializa todos los sistemas que forman la *Smart City* y se detallan los diferentes tipos de comunicaciones que utilizan estos sistemas y que son de utilidad para desarrollar la solución.

En el **capítulo 4 (Análisis del problema)**, se expone el mapa conceptual del problema que se ha abordado mediante un análisis de los diagramas de clases y un borrador de sus interfaces.

En el **capítulo 5 (Diseño de la solución)**, se explica la aplicación de patrones, estilos arquitectónicos y modelos que se utilizan para abordar la solución al problema. En este caso, se expone detalladamente el funcionamiento de la computación autónoma y del modelo de bucles de control (feedback loops) MAPE-K que se utiliza para el desarrollo de los escenarios.

En el **capítulo 6 (Implementación)**, se detalla las estrategias utilizadas y los casos de uso implementados mediante escenarios aplicables a la vida real y como se han desarrollado sus módulos y componentes.

En el último capítulo con **número 7 (Conclusiones)**, se tiene las características del proyecto desarrollado, las conclusiones a las que se ha llegado y los retos tecnológicos que se han conseguido superar. También se comentan los futuros trabajos posibles que se pueden llegar a desarrollar en este ámbito.

Finalmente, se adjunta un **apéndice** en forma de manual de usuario sobre cómo se han probado algunos de los escenarios que se han desarrollado y comprobado que el funcionamiento era el esperado.

CAPÍTULO 2
Contexto Tecnológico

2.1 Internet de las cosas

La internet de las cosas (IoT) es un concepto que está teniendo mucho éxito en la actualidad y todo se debe a las expectativas que genera el poder conectar cualquier objeto a Internet y hacer cualquier cosa que se te ocurra.

La posibilidad de adjudicar de una capacidad informática a objetos cotidianos y dotarles de cierta inteligencia para que recojan datos de su entorno y se comuniquen entre ellos puede crear numerosas aplicaciones que pueden llegar a cambiar por completo la vida de las personas.

Aunque parece que estamos hablando de algo que lleva con nosotros bastante tiempo dada su popularidad en los últimos días, la realidad es que se trata de un concepto muy reciente. No fue hasta el año 2009, cuando el profesor Kevin Ashton utilizó la expresión "*Internet of Things* (IoT)" por primera vez, desde entonces ha sufrido un crecimiento y se ha creado una expectación enrome alrededor del término que ha ido creciendo de forma exponencial.

Para poder entender en su totalidad el concepto hay que retomarse a las distintas evoluciones tecnológicas que han desembocado en lo que conocemos hoy por IoT y que nos han traído a este punto ya que la idea de poder conectar los objetos y hacer que estos fuesen inteligentes ya existía desde hace décadas.

No fue hasta la década de los 60 y, sobre todo, los 70 cuando se crearon los primeros protocolos de comunicaciones que definirían la base de lo que hoy es Internet. Cuando se popularizó Internet no tardó mucho en volver la idea de conectar objetos aprovechando esta nueva red y fue en 1990 cuando John Romkey creo el primer objeto conectado a Internet: una tostadora que se podía encender o apagar en remoto.

A pesar de suponer una revolución esta visión de entender las redes, las comunicaciones que Internet ofrecía en aquellos años eran principalmente cableadas. Esto, y relacionado con que el coste del hardware era aún bastante elevado, hizo que el concepto de implementar objetos conectados pasase inadvertido durante algunos años más.

La revolución y finalmente la popularización de este concepto vino de la mano de la conectividad inalámbrica que permitió aumentar el crecimiento de objetos conectados. Este crecimiento se ha visto especialmente exponenciado por el surgimiento de otras nuevas tecnologías como las *Wireless Sensor Networks* (WSN) o las comunicaciones *Machine to Machine* (M2M).

2.2 Aplicaciones utilizadas del internet de las cosas

Smart Cities

Una de las principales aplicaciones de la Internet de las cosas, que define como cada objeto y elemento físico de una ciudad puede llegar a esta interconectado y generar información de utilidad. Estos objetos pueden ser cualquier elemento cotidiano como coches, carreteras, señales de tráfico, semáforos, viviendas o el alumbrado.

Mediante la implementación de servicios adaptables que monitoricen toda esta información que se genera y su evaluación, se puede conseguir dar cierta autonomía o inteligencia a la ciudad con una capacidad para que se auto adapte al entorno cambiante que le rodea.

Connected Cars

El concepto de coche conectado no es nuevo ya que existen muchos ejemplos de vehículos que están integrados a las redes de comunicación a través de la interconexión con otros dispositivos móviles. Pero aun, uno de los desafíos que está por cumplir es la fabricación de vehículos que utilicen estas nuevas tecnologías para facilitar la conducción y aumentar la seguridad.

El objetivo que se pretende conseguir en el futuro, es la existencia de la conducción autónoma de vehículos sin ninguna intervención del conductor. Esto se conseguiría en gran medida gracias a la conexión directa con la red para producir y consumir información en tiempo real.

Gestión autónoma del tráfico

En cuanto a la gestión autónoma de una ciudad, se necesita disponer primero de una plataforma que integra una *Smart City*, donde todos sus elementos consumen y producen información útil a través de un mismo tipo de comunicación en un lenguaje conocido por todos los demás.

En segundo lugar, de unos vehículos conectados y por lo tanto inteligentes con la capacidad de comunicarse con su infraestructura vial para tomar decisiones y adaptar su comportamiento de forma autónoma dependiendo de su entorno.

Cuando ya se dispone de estas dos aplicaciones, se consigue desarrollar soluciones para las ciudades de forma que puedan ofrecer diferentes servicios a los usuarios y se autorregulen para adaptarse a diferentes necesidades como la situación del tráfico y la aparición de emergencias o incidencias en sus carreteras.

2.3 Tecnologías utilizadas

En el desarrollo del proyecto se ha necesitado realizar una investigación y un aprendizaje sobre las tecnologías que se implementan y se utilizan como base de la plataforma de la que se parte y explica en el caso de estudio y también en el desarrollo de las adaptaciones y los escenarios. Se van a comentar una a una de forma detallada cuáles son.

2.3.1. REST

La tecnología ***RE**presentational **S**tate **T**ransfer* (Tranferencia de Estado Representacional) o comúnmente llamado por su acrónimo **REST**, describe como un sistema puede comunicar su estado a otros sistemas. Como estado se hace referencia a un producto que generalmente se denomina recurso y el acceso a sus características como el nombre o su descripción.

REST se asocia comúnmente con interfaces de servicios web, ya que HTTP es con gran diferencia el protocolo de transporte más utilizado en esta tecnología gracias a su gran facilidad de poder acceder a los recursos.

Una aplicación RESTful, es una aplicación que expone su estado y funcionalidad como un conjunto de recursos que los clientes pueden manipular y modificar de acuerdo a un conjunto de principios:

- Los recursos son inequívocamente identificables y direccionables mediante una dirección normalmente URI, aunque también existen otro tipo de direccionamientos posibles.

- Todos los recursos pueden ser manipulados a través de un conjunto restringido de acciones bien conocidas, generalmente CRUD (crear, leer, actualizar y eliminar) representado normalmente a través de los métodos HTTP: POST, GET, PUT y DELETE.

- No hace falta que se desarrollen todos los métodos HTTP sobre los recursos ya que, si se quiere restringir o limitar el acceso o modificación de un recurso, solo es necesario implementar los métodos que realizan la operación.

- Los datos de los recursos se transfieren a través de cualquiera de las representaciones más conocidas como son XML, JSON o HTML.

La plataforma IoT que se utiliza como base en el proyecto tiene definida por completo una API REST que muestra los recursos de la *Smart City* y sus comunicaciones. Así también, se desarrollan en el proyecto ampliaciones de las soluciones que hacen uso de estas comunicaciones.

2.3.2. MQTT

La tecnología *Message Queuing Telemetry Transport* o comúnmente llamado por su acrónimo MQTT, es un protocolo de mensajería ligero que proporciona a dispositivos llamados clientes que están conectados a una red de comunicaciones y que disponen de recursos limitados, una forma sencilla de distribuir información de telemetría.

El protocolo utiliza un patrón de comunicación *publish/subscribe* que se utiliza también en tecnologías en las que se basa como son las comunicaciones *machine-to-machine* (M2M) y que juegan un papel importantísimo en el Internet de las cosas.

MQTT permite a los dispositivos enviar información mediante la acción *publish* sobre un tema determinado llamado *topic*, a un servidor que funciona como intermediario de mensajes MQTT llamado *broker*. A continuación, el *bróker* envía la información a los clientes que se han suscrito previamente mediante una acción *subscribe* al *topic* del cliente.

Un *topic* o tema, se representa como una ruta hacia un archivo. Los clientes se pueden suscribir a un determinado nivel de jerarquía de la ruta o utilizar caracteres especiales llamados comodines para suscribirse a múltiples niveles.

El comodín de un nivel (+) como indica el nombre es el sustituto para 1 nivel del *topic* y se puede utilizar en cualquier nivel. Se recibirán todos mensajes que sustituya ese nivel exacto.

Figura 2.1: MQTT Comodín de un nivel (+)

El comodín multi nivel (#) requiere que siempre sea el último carácter del *topic* y recibe todos los mensajes de cualquier *topic* con su prefijo.

Figura 2.2: MQTT Comodín multi nivel (#)

La tecnología es una buena opción para redes inalámbricas que experimentan diversos problemas como altos niveles de variabilidad de su latencia debido a las restricciones de ancho de banda o la poca estabilidad de las conexiones. El éxito de la tecnología y el gran atractivo son estas funcionalidades que permiten funcionar sobre redes poco fiables.

El funcionamiento ante una desconexión o interrupción del servicio a uno de los clientes es:

- Cuando la conexión de un suscriptor se rompe o se pierde, el *broker* almacena en un *buffer* todos los mensajes que debería de estar recibiendo el cliente perdido. En el momento en el que éste vuelva a estar operativo se los devolverá en su orden exacto al que fueron enviados.

- Por lo contrario, cuando la conexión con un cliente que publica información se pierde sin ningún aviso, el *broker* puede cerrar también la conexión y enviar a los suscriptores mensajes de cache con instrucciones del publicador.

La plataforma IoT que se utiliza como base en el proyecto hace uso de estas colas de mensajería para las comunicaciones entre los diferentes sistemas. Así también, se desarrollan en el proyecto ampliaciones de las soluciones que hacen uso clientes MQTT que realizan suscripciones y publicaciones en tiempo real de los sistemas que integra la *Smart City*.

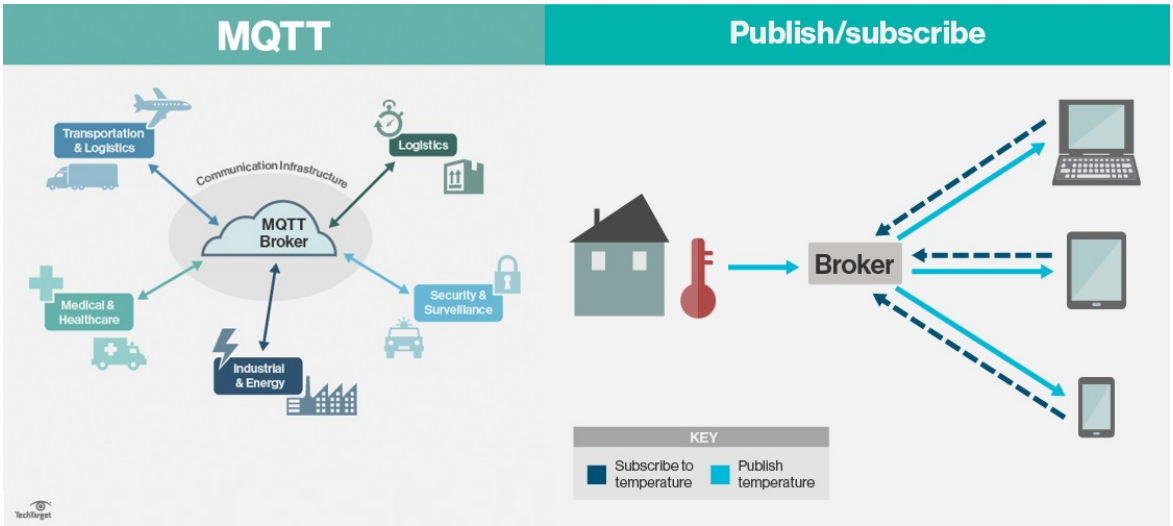

Figura 2.3: Funcionamiento comunicación MQTT

2.3.3. JSON

Se ha visto como las tecnologías REST o MQTT necesitan de una representación en algún formato legible por humanos para sus comunicaciones y tanto en la plataforma desde la que se parte para la realización del proyecto, como las soluciones que se han desarrollado se han optado por utilizar una de las representaciones más populares como es JSON.

JSON es la abreviatura de *JavaScript Object Notation* (Notación de objetos JavaScript), y es una forma de almacenar información de forma organizada y de fácil acceso. El éxito de la tecnología viene dado por que nos proporciona que una colección de datos este en un formato que sea fácil de leer, comprender y entender gracias a su composición y que, además, podemos acceder a él de una manera lógica.

Como ejemplo, voy a representar información sobre mi persona escrita en JSON:

```
1  {
2      "name" : "Carlos",
3      "age" : 21,
4      "gender" : "male",
5      "hometown" : "Valencia"
6  }
```

Figura 2.4: Representación de ejemplo de un JSON

Esta declaración y representación crea un objeto persona al que se puede tener acceso. Con la declaración entre llaves se esta indicando que el valor contenido es un objeto y dentro del objeto se puede declarar cualquier numero de propiedades utilizando la estructura "llave": valor separados correctamente por comas. Para acceder a estas propiedades se hace referencia al nombre de la propiedad que se necesita.

A pesar de que JSON lleva el nombre de JavaScript, esta representación puede utilizarse en una gran variedad y multitud de lenguajes de programación como Java, C, C++, Python o PHP.

2.3.4. OSGi

La tecnología OSGi, es un sistema de módulos dinámicos para el lenguaje de programación Java. Con OSGi se pueden definir y desarrollar los medios para instalar, desinstalar, actualizar, iniciar y detener estos módulos.

Estos módulos se llaman *bundles*, pero son, en su forma más simple, unos archivos Java con formato .jar que tienen un manifest especial. Lo atractivo de esta tecnología es que los módulos se pueden instalar, desinstalar y todo lo comentado anteriormente sin detener o reiniciar la máquina virtual de Java sobre la que se está ejecutando.

El framework OSGi gestiona el ciclo de vida de los módulos descritos y además las dependencias entre los paquetes de una manera segura. Un paquete debe indicar que otros paquetes java tiene que exportar e importar. La importación y la exportación de un módulo puede describirse con información sobre la versión utilizada, de modo que se puede tener más de una versión en el mismo paquete ejecutándose en la misma máquina virtual de Java.

La alianza OSGi es la organización que especifica y define el framework de OSGi y todos los servicios que lo acompañan, como por ejemplo, la administración y configuración de datos o el acceso a dispositivos.

2.4 Herramientas software

A continuación, se van a detallar de forma muy breve las herramientas software que no solo han sido imprescindibles para el desarrollo del proyecto, sino que también se utilizan para la presentación y la representación de su funcionamiento.

2.4.1. Eclipse Plug-in Development IDE

El entorno de Eclipse para desarrollo de complementos llamado *Eclipse Plug-In Development Environment* (Eclipse PDE) proporciona herramientas para crear, desarrollar, probar, depurar, crear e implementar plugins Eclipse, fragmentos, características, actualizaciones y productos RCP.

Este ide, tiene de especial que también proporciona herramientas y compatibilidad total con el *framework* OSGi, lo que lo convierte en un entorno ideal no solo para la programación de componentes, sino también de plugins de Eclipse.

Se utiliza para el desarrollo de los módulos adaptativos para la *Smart City*.

2.4.2. Postman

Postman es el *software* que está disponible como complemento para el navegador web Google Chrome y es utilizado en el proyecto gracias a su sencillez y potencial para consumir servicios web REST. Permite ejecutar todo tipo de métodos HTTP hacia un servicio web y personalizar las peticiones con un nivel de fiabilidad y sencillez muy alto.

Se utiliza a lo largo de todo el proyecto para monitorizar cual es el estado de los recursos que están en la *Smart City* y para comprobar que los cambios que realizan las soluciones implementadas funcionan correctamente.

2.4.3. MQTT.fx

La herramienta software MQTT.fx pretende ser una herramienta de escritorio rápida y fácil de usar para la depuración y pruebas de funcionamiento en comunicaciones *machine-to-machine* (M2M) sobre MQTT. Está en continuo desarrollo desde hace ya aproximadamente 2 años y sigue creciendo añadiendo nuevas funcionalidades cada poco tiempo.

Algunas de las funcionalidades que puede hacer son:

- Diferentes perfiles de configuración para conectarse con diferentes MQTT *brokers*.
- *Publish / Subscribe* sobre los topics que se le indiquen como parámetro.
- Soporte para conexiones ad-hoc, seguridades SSL/TLS y autenticación con nombre de usuario y contraseña.

Se utiliza también, a lo largo de todo el proyecto para monitorizar cual es el estado de los recursos que están en la *Smart City* y para comprobar que los cambios que realizan las soluciones implementadas funcionan correctamente.

2.5 Proyectos similares

Por último, se va a mostrar el estudio de un proyecto similar y a la vez complementario aplicado ya a un escenario real y además cercano que desde los últimos meses ya se está implantando.

Se trata de la denominada "Plataforma de Ciudad Inteligente (VLCi)" que es el resultado de aplicar a la ciudad de Valencia una solución tecnológica para convertirla en una *Smart City*. De esta forma, se consigue una mejor gestión de los servicios urbanos y una mejorara de forma exponencial en los flujos de información internos y externos de la ciudad para maximizar la eficiencia de los recursos disponibles.

Además, la ciudad es el referente europeo en utilizar y aplicar por primera vez sobre una ciudad el estándar abierto *FIWARE* de la unión europea. Por ahora se encuentra en fase de implantación en el consistorio y actualmente ya se han integrado a la plataforma un total aproximado de 550 indicadores sobre los cuales ya se ha obtenido la certificación ISO 37120 sobre Desarrollo Sostenible en las Ciudades.

La plataforma toma información constantemente actualizada sobre los sistemas de la ciudad que están funcionando y se la muestra de forma transparente al ciudadano en más de 45 ámbitos entre las que destacan algunas como en este proyecto, sobre la gestión del tráfico.

El proyecto es totalmente complementario con el que de describe en esta plataforma y se podría adaptar sin excesivos problemas utilizando como referencia el mapa de Valencia y que para la toma de medidas se utilicen las que proporciona la plataforma.

Figura 2.5: Esquema de la Plataforma VLCi

CAPÍTULO 3
Caso de estudio

3.1 Introducción a la plataforma *SmartTraffic*

La plataforma *SmartTraffic* es la base que se utiliza en el proyecto y desde la que se parte para desarrollar e implementar los servicios adaptativos que modifican el comportamiento de sus sistemas de forma autónoma y atendiendo a las necesidades de su entorno en todo momento. Esta plataforma se me proporciona ya desarrollada por el tutor del proyecto, Joan Fons, con el objetivo de extender sus capacidades adaptativas.

Propone un prototipo de servicio que integra diferentes recursos y servicios en el ámbito del tráfico de las ciudades inteligentes, ofreciendo una plataforma que utiliza los conceptos de la Internet de las cosas para integrar todos los tipos de recursos que son diferentes entre sí.

Los elementos y recursos (o también llamados *Smart Things*) que ofrece son los siguientes:

- **Red de carreteras inteligentes**: representan las carreteras o calles de un mapa de un área o ciudad inteligente. Se les denomina dentro de la plataforma con el nombre de Roads y dan toda la información sobre su estado en tiempo real como la saturación de la vía, el número de vehículos circulando actualmente, las complicaciones y los incidentes, las restricciones a la circulación o las intersecciones que tienen.

- **Señales de Tráfico**: representados por semáforos, límites de velocidad o incidentes temporales. Cada señalización de tráfico está posicionada en un segmento de carretera llamado RoadSegments, en un punto kilométrico exacto y concreto. Para la presentación de los escenarios se utiliza una maqueta física que representa estas señalizaciones de la carretera y sus diferentes modos de funcionamiento.

- **Vehículos Inteligentes Conectados**: representados por unos vehículos conectados que tienen a lo largo del desarrollo de las soluciones un comportamiento inteligente gracias a que puede conectarse a la infraestructura vial para conocer el estado de las carreteras y a la señalización y adaptarse según sus necesidades. Además, pueden proporcionar información sobre incidentes como accidentes o averías.

- **Incidentes**: que ocurren sobre las carreteras como obras, accidentes o calles cortadas por ejemplo y, además, sobre otros elementos como pueden ser los vehículos con averías técnicas que puedan provocar alguna complicación sobre la correcta

circulación y el flujo de tráfico.

- **Simulador**: que nos proporciona la interfaz y los mecanismos para ayudar a reproducir escenarios realistas y que se podrían aplicar a situaciones de la vida real. El simulador tiene la fuerza para generar información sobre toda la infraestructura, pudiendo y teniendo la capacidad de crear vehículos virtuales que circulan por ella, manejar señalizaciones de tráfico como semáforos o generar incidentes de todo tipo.

Con esto se pretender ver y mostrar que esta es la plataforma desde la que se parte y que se utiliza en el proyecto para diseñar soluciones y escenarios IoT. En el momento de empezar el proyecto, el comportamiento de los sistemas era completamente independiente y no había ningún tipo de servicio autónomo y de auto-gestión funcionando pero si que existían las necesidades que sugerían una adaptación de los sistemas para mejorar su situación.

Para ello, hay que tener en cuenta que el factor limitante del proyecto y el mayor reto al que se enfrenta en su desarrollo, es que no se puede acceder a la implementación de cada uno de los servicios o recursos que forman la *Smart City*. Por lo tanto, la idea es desarrollar módulos independientes mediante el *framework* de OSGi y comunicarlos con cada uno de los sistemas para adaptarlos a sus necesidades a través de los métodos de comunicación que ya disponga la plataforma.

Por último, en los anexos se muestra como configurar el simulador y la puesta en marcha de las soluciones que se explicarán más adelante.

3.2 Escenario y mapa del Parque Tecnológico de Paterna

La infraestructura vial que se está utilizando en el proyecto bajo el simulador de la plataforma *SmartTraffic* es la del Parque Tecnológico de Paterna que en adelante nombraremos como PTPaterna. El diseño está totalmente preparado para permitir tanto a vehículos como a las carreteras mantenerse conectadas e intercambiar información.

Sin duda la elección para trabajar sobre esta red de carreteras y este mapa no es aleatoria ya que esta infraestructura es muy contenida y dispone de todos los elementos que se necesita para la simulación, ya que nos permite mostrar, diseñar e implementar cualquier escenario de una forma más sencilla.

Sin embargo, las soluciones desarrolladas no están limitadas únicamente a este escenario si no que están preparadas para que funcionen bajo cualquier mapa de carreteras sin alterar en ningún momento la implementación necesaria para su funcionamiento.

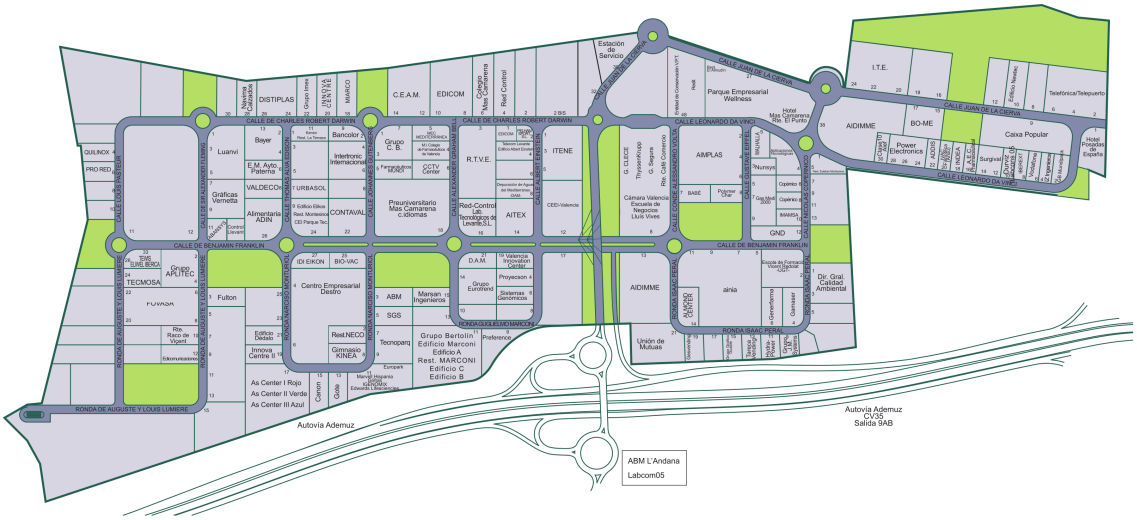

Figura 3.1: Plano del Parque Tecnológico de Paterna

3.3 Sistema de red de carreteras inteligentes

Se muestra a continuación en la siguiente imagen el mapa de carreteras que se va a utilizar en todo el proyecto con la correspondiente identificación que se le ha proporcionado.

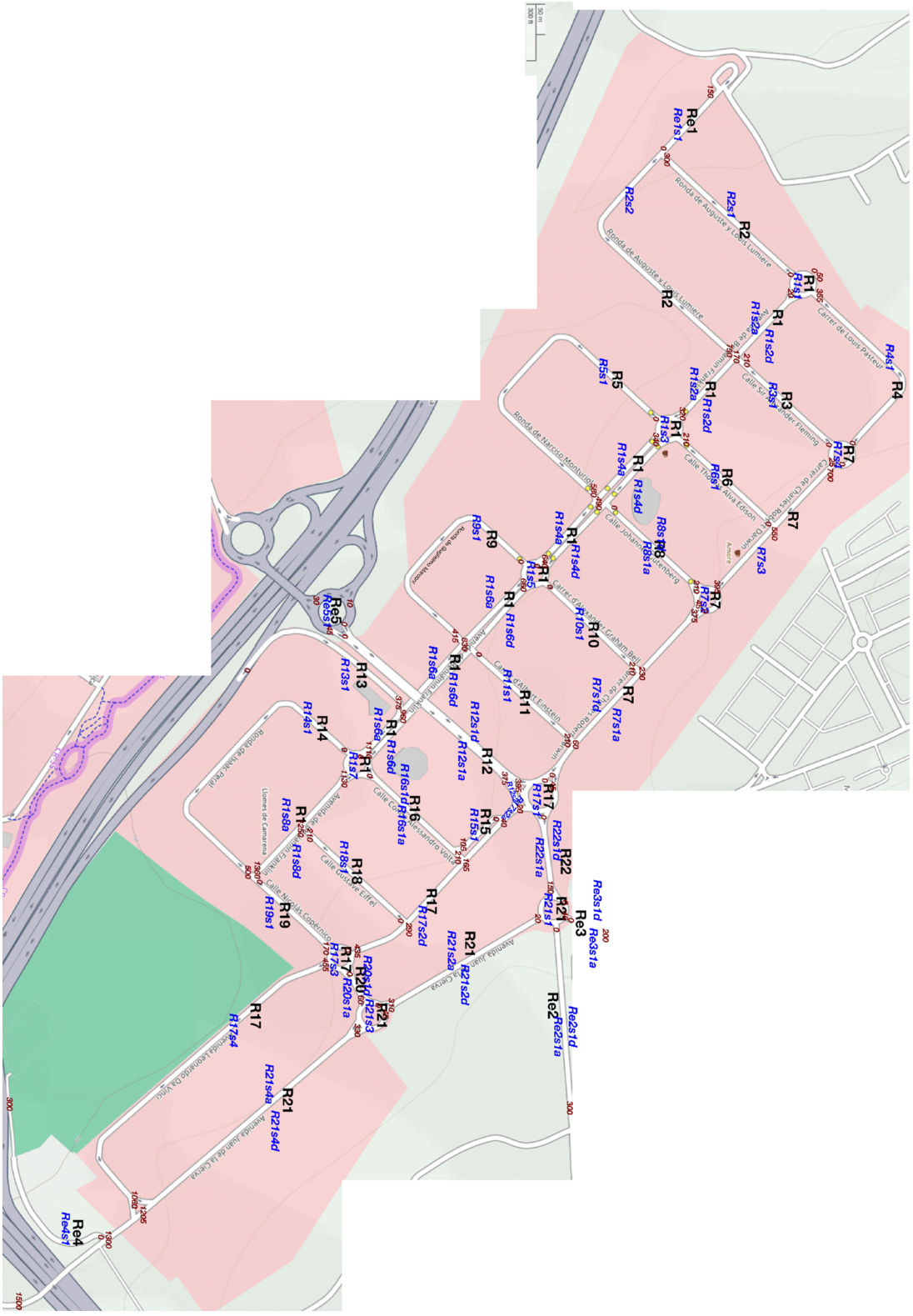

Figura 3.2: Escenario y red de carreteras del Parque Tecnológico de Paterna

3.3.1. Funcionamiento

Las etiquetas de color negro son los nombres e identificadores que se le ha proporcionado a las calles y que van desde el R1 hasta el R22. Así también, las etiquetas de color azul son los diferentes segmentos o tramos de estas calles y las de color rojo indican las distancias (en metros) dentro de cada uno de los segmentos.

Una carretera se estructura en una secuencia de segmentos de los cuales, en cada uno de ellos se tiene un punto de inicio (kilometro inicial) y un punto final (kilometro final). Se puede observar en el siguiente ejemplo de forma más clara:

- La road o carretera R2 se compone de dos segmentos que son R2s1 y R2s2.
- El segmento R2s1 tiene una longitud de 300m y su punto kilométrico inicial empieza en el número 0 y su punto kilométrico final acaba en el número 300.
- El segmento R2s2 tiene una longitud de 450m y su punto kilométrico inicial empieza en el número 300 y su punto kilométrico final acaba en el número 750.
- En cualquier punto de un segmento puede existir una intersección que une con otro segmento. En el caso expuesto hay una intersección para indicar que R2s1 termina y continua con el segmento R2s2.

3.3.2. Propiedades

La representación de un recurso dentro de sistema de carreteras tiene normalmente un gran número de propiedades por lo que las más importantes según su documentación son:

- rt: indica el tipo de recursos que puede ser entre road o road-segment
- name: indica el nombre de la carretera
- id: identificador de la carretera
- segments: tramos que conforman la carretera
- start-kp y end-kp: puntos kilométricos inicial y final
- density: saturación del tramo de carretera
- capacity: capacidad máxima de vehículos en la carretera
- num-vehicles: número de vehículos circulando por el segmento
- max-speed: máxima velocidad permitida en este tipo de vía
- current-max-speed: máxima velocidad permitida 'en estos momentos'
- status: tipo de saturación del segmento (Free Flow, Limited Manouvers, Collapsed, Closed)
- capacity: capacidad (número de vehículos) que 'caben' circulando en la carretera antes de colapsarse

3.4 Sistema de simulación de vehículos inteligentes

Se dispone de un sistema que permite la simulación de vehículos inteligentes que son actualmente capaces de desenvolverse por su entorno y adaptarse a las necesidades de las carreteras, sus señales de tráfico y las incidencias que puedan llegar a encontrarse cuando siguen una ruta de navegación en la plataforma *SmartTraffic*.

Los vehículos son altamente configurables y están diseñados para adaptarse a cualquier tipo de automóvil entre una gran variedad (coche, moto, furgoneta, camión, etc) e incluso se le pueden asignar roles (coche uso privado, transporte público, ambulancia, coche policía, bomberos, etc.) que nos permite asignarles una caracterización y más tarde tener la opción de diferenciarlos y priorizarlos dependiendo de la situación.

Estos vehículos simulados se pueden aplicar a un escenario real e incluso se disponen de maquetas físicas ya existentes que tienen la misma configuración e interactúan de la misma forma que las simulaciones hacen.

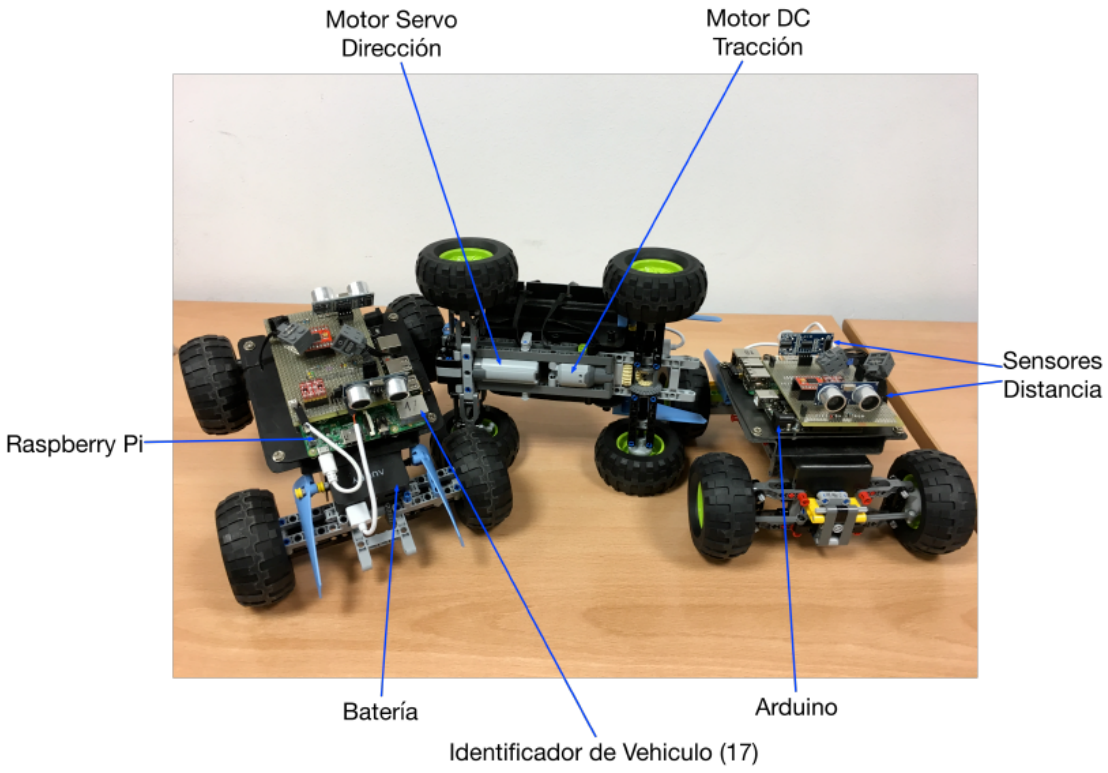

Figura 3.3: Prototipo físico de vehículo inteligente conectado

3.5 Comunicaciones directas mediante API REST

Para la interacción y comunicación entre todos los sistemas que integran y hacen funcionar a la plataforma *SmartTraffic*, existen dos tipos de comunicaciones posibles con las tecnologías que se han comentado en el capítulo anterior. Estos dos tipos de comunicaciones son complementarias y están pensados para diferentes situaciones o escenarios.

En este primer punto y como bien indica el nombre, se trata de una comunicación directa con los sistemas a través de una API REST, que nos da la posibilidad de hacer una consulta sobre la situación de los recursos que la componen.

A continuación, veremos brevemente como hacer uso de la API que ofrecen los diferentes sistemas, con el objetivo de entender su funcionamiento y saber cómo poder acceder a sus diferentes recursos.

3.5.1. Sistema de carreteras inteligentes

Consulta sobre la información y estado actual de todas las carreteras disponibles más sus propiedades en tiempo real.

```
GET Request: http://localhost:8180/roads
```


```
425     },
426     {
427         "rt": "road",
428         "link": "/road/R8",
429         "name": "Carrer Johannes Gutenberg",
430         "id": "R8",
431         "segments": [
432             {...},
448             {...}
464         ]
465     },
466     {
467         "rt": "road",
468         "link": "/road/R9",
469         "name": "Ronda de Guglielmo Marconi",
470         "id": "R9",
471         "segments": [
472             {
473                 "rt": "road-segment",
474                 "end-kp": 415,
475                 "code": "R9s1",
476                 "density": 0,
477                 "link": "/segment/R9s1",
478                 "length": 415,
479                 "max-speed": 60,
480                 "road-segment": "R9s1",
481                 "start-kp": 0,
482                 "capacity": 41,
```


Figura 3.4: Respuesta JSON Array con información sobre todas las carreteras

Consulta sobre la información y estado actual de una carretera indicada (ej. road: Re1) más sus propiedades en tiempo real.

```
GET Request: http://localhost:8180/road/Re1
```


```
{
    "rt": "road",
    "link": "/road/Re1",
    "name": "Re1 Accés Nord",
    "id": "Re1",
    "segments": [
        {
            "rt": "road-segment",
            "end-kp": 150,
            "code": "Re1s1",
            "density": 0,
            "link": "/segment/Re1s1",
            "length": 150,
            "max-speed": 80,
            "road-segment": "Re1s1",
            "start-kp": 0,
            "capacity": 15,
            "current-max-speed": 80,
            "road": "Re1",
            "num-vehicles": 0,
            "status": "Free_Flow"
        }
    ]
}
```


Figura 3.5: Respuesta JSON con información sobre carretera R2 y sus segmentos

Consulta sobre la información y estado actual de un segmento concreto de una carretera (ej. road: Re1s1) más sus propiedades en tiempo real.

```
GET Request: http://localhost:8180/segment/Re1s1
```


```
{
    "rt": "road-segment",
    "end-kp": 150,
    "code": "Re1s1",
    "density": 0,
    "link": "/segment/Re1s1",
    "length": 150,
    "max-speed": 80,
    "road-segment": "Re1s1",
    "start-kp": 0,
    "intersections": [
        {
            "exit-road-point": {
                "road-segment": "Re1s1",
                "position": 0
            },
            "entry-road-point": {
                "road-segment": "R2s2",
                "position": 300
            },
            "status": "Opened"
        }
    ],
    "capacity": 15,
    "current-max-speed": 80,
    "road": "Re1",
    "num-vehicles": 0,
    "status": "Free_Flow"
}
```


Figura 3.6: Respuesta JSON con información sobre segmento Re1s1 y sus intersecciones

3.5.2. Sistema de vehículos inteligentes

Consulta sobre los vehículos que están circulando actualmente por la ciudad en el simulador.

```
GET Request: http://localhost:8180/vehicles
```


```
[
    {
        "navigator": {
            "off-road": false,
            "route": "<No route>",
            "current-position": "(R1s1,300)",
            "destination": "(R1s1,300)",
            "remaining-distance": 0,
            "id": "3240JVM_navigator",
            "status": "REACHED_DESTINATION"
        },
        "id": "3240JVM",
        "characterization": {
            "role": "PublicTransport",
            "type": "Bus"
        },
        "cruiser-speed": 80,
        "speed": 0,
        "status": "Parked"
    }
]
```


Figura 3.7: Respuesta JSON Array con información sobre todos los vehículos disponibles

Consulta sobre la información y el estado actual de un vehículo indicado (ej. vehicle: 1000JBB) más sus propiedades en tiempo real.

```
GET Request: http://localhost:8180/vehicles/1000JBB
```


```
{
    "navigator": {
        "id": "1000JBB_navigator",
        "status": "WAITING"
    },
    "id": "1000JBB",
    "characterization": {
        "role": "PublicTransport",
        "type": "Bus"
    },
    "cruiser-speed": 80,
    "speed": 0,
    "status": "Stopped"
}
```


Figura 3.8: Respuesta JSON con información sobre el vehículo indicado

Creación de un nuevo vehículo en el simulador por ejemplo de tipo "Autobús" y con el rol de "Transporte público" e identificador "1000JBB" con velocidad de 80 km/h.

```
POST Request: http://localhost:8180/vehicles
"Content-Type":"application/json"
"Accept":"application/json"

{ "id":"1000JBB", "speed":80,"type":"Bus","role":"PublicTransport" }
```


```
{
    "navigator": {
        "id": "1000JBB_navigator",
        "status": "WAITING"
    },
    "id": "1000JBB",
    "characterization": {
        "role": "PublicTransport",
        "type": "Bus"
    },
    "cruiser-speed": 80,
    "speed": 0,
    "status": "Stopped"
}
```


Figura 3.9: Respuesta JSON con información del nuevo vehículo creado

Creación de una nueva ruta en el simulador a un vehiculo existente con identificador por ejemplo "1000JBB" que atraviese al menos dos road-segments.

```
PUT Request: http://localhost:8180/vehicles/1000JBB
"Content-Type":"application/json"
"Accept":"application/json"

{"route":[[{"road":"R2s2","point":0},{"road":"R2s2","point":750}],[{"road":"
    R1s1","point":0},{"road":"R1s1","point":300}]]}
```


```
{
    "navigator": {
        "off-road": false,
        "route": "[(R2s2,0)-(R2s2,750)][(R1s1,0)-(R1s1,300)]",
        "current-position": "(R2s2,0)",
        "destination": "(R1s1,300)",
        "remaining-distance": 1050,
        "id": "1000JBB_navigator",
        "status": "WAITING"
    },
    "id": "1000JBB",
    "characterization": {
        "role": "PublicTransport",
        "type": "Bus"
    },
    "cruiser-speed": 80,
    "speed": 0,
    "status": "Stopped"
}
```


Figura 3.10: Respuesta JSON con información del vehiculo con su ruta nueva

3.6 Comunicaciones indirectas mediante cola de mensajes MQTT

En este segundo punto que habla de comunicaciones y como también indica el nombre, se explican las comunicaciones indirectas entre los elementos a través de un sistema de colas de mensajes MQTT. Se trata de otro mecanismo de comunicación en el que los vehículos y las carreteras se intercambian la información de manera no directa y dejan atrás mecanismos donde un recurso interactúa con otros recursos.

En esta comunicación debe de existir un intermediario que se encarga de clasificar y distribuir los mensajes a múltiples recursos que se llama *middleware*. De esta forma, se permite que un recurso se comunique (haciendo el rol de publicador) con múltiples recursos a la vez (que hacen de suscriptores).

Como se ha comentado anteriormente, esta tecnología permite que tanto el rol de publicador y suscriptor se intercambie de forma continua y que la actualización de algún recurso se notifique a los demás cuando ocurre sin la necesidad de que haya otros recursos preguntando continuamente si ha habido algún cambio de interés para ellos.

El patrón de comunicación que se utiliza en esta plataforma es útil en algunos de los diferentes escenarios que se pueden plantear. Según la documentación de la plataforma, es útil en casos donde una carretera quiere informar a todos los vehículos que circulan por que ha habido una incidencia, el cambio de información en las señales de tráfico o cuando un vehículo quiere difundir un mensaje como que ha sufrido un accidente.

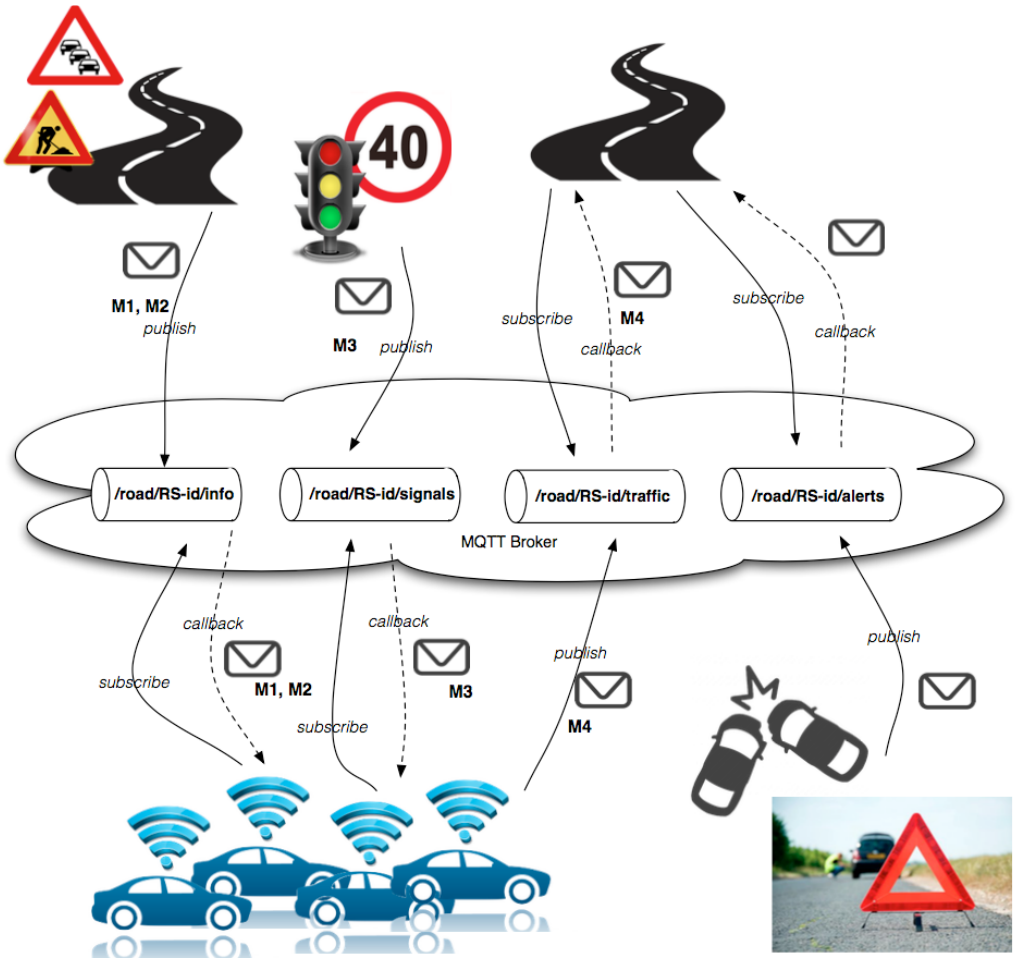

Figura 3.11: Esquema de comunciaciones indirectas mediante colas MQTT

Como se puede ver en el esquema de comunicaciones entre vehículos y carreteras existen 4 colas de mensajería principales de las cuales se van a explicar a continuación los mensajes más importantes para entender el desarrollo que explica posteriormente de las soluciones.

Carreteras difunden mensajes de estado e incidentes a los vehículos.
Mensaje I1: Estado del segmento de una carretera. Topic: road/R1s1/info


```
{
    "msg":{
        "rt":"road-segment",
        "road":"R1",
        "road-segment":"R1s1",
        "start-kp":0,
        "end-kp":50,
        "length":50,
        "capacity":6,
        "num-vehicles":2,
        "status":"Mostly_Free_Flow",
        "max-speed":40,
        "current-max-speed":40,
        "link":"/segment/R1s1"
    },
    "id":"MSG_3287774477254",
    "type":"ROAD_STATUS",
    "timestamp":3287774477254
}
```


Mensaje I2: Incidencia en el segmento de una carretera. Topic: road/R1s1/info


```
{
    "msg":{
        "incident-type":"INCIDENT",
        "id":"612477150",
        "road-segment":"R1s1",
        "starting-position":10,
        "ending-position":20,
        "description":"Working Area",
        "status":"Active",
        "rt":"traffic::incident",
        "link":"/incident/612477150"
    },
    "id":"MSG_3287774477255",
    "type":"ROAD_INCIDENT",
    "timestamp":3287774477255
}
```


En el mensaje I1 se indica que el Road-segment o segmento de la carretera publica periódicamente su estado en la cola info para que los vehículos que están en ella, sepan en todo momento cuál es su estado mientras que en el mensaje I2 se indica la aparición de un accidente desde el punto 1' hasta el 20 por ser una zona de trabajo.

Señales de tráfico que notifican sus cambios de estado.

Como se puede ver en los dos mensajes para el valor de los semáforos se utiliza una nomenclatura especial donde L = Apagado y H = Encendido. Por lo tanto se pueden hacer diferentes combinaciones teniendo en cuanta que la primera luz es de color rojo, la segunda de color amarrillo y la tercera de color verde.

En el mensaje S1 se indica que el semáforo del segmento R1s2a ha cambiado su estado parpadeando en color ámbar gracias a la combinación de "LHL". En el mensaje S2 por contra se puede ver que ahora ha cambiado a luz roja fija gracias a la combinación "HLL".

Mensaje S1: Estado de un semáforo. Topic: road/R1s2a/signals


```
{
    "msg":{
        "signal-type":"TRAFFIC_LIGHT",
        "rt":"traffic-signal",
        "starting-position":170,
        "id":"TL_atR1s2a_170",
        "ending-position":170,
        "road-segment":"R1s2a",
        "value":"LHL"
    },
    "id":"MSG_1499024163951",
    "type":"TRAFFIC_SIGNAL",
    "timestamp":1499024163951
}
```


Mensaje S2: Estado de un semáforo. Topic: road/R1s2a/signals


```
{
    "msg":{
        "signal-type":"TRAFFIC_LIGHT",
        "rt":"traffic-signal",
        "starting-position":170,
        "id":"TL_atR1s2a_170",
        "ending-position":170,
        "road-segment":"R1s2a",
        "value":"HLL"
    },
    "id":"MSG_1499024166952",
    "type":"TRAFFIC_SIGNAL",
    "timestamp":1499024166952
}
```

Los vehículos notifican su posición exacta a través de la ubicación.

Como bien indica el título los vehículos indican su posición exacta en todo momento cuando están atravesando un segmento de carretera. En concreto, realizan tres diferentes acciones entre las que estan registrar su acceso al segmento, indicar su posición actual en todo momento en la vía y registrar su salida al segmento.

Mensaje T1: Entrada de un vehículo a un segmento. Topic: road/R2s2/traffic

```
{
    "msg":{
        "vehicle-role":"PublicTransport",
        "action":"VEHICLE_IN",
        "road-segment":"R2s2",
        "position":0,
        "vehicle-id":"3240JVM"
    },
    "id":"MSG_1499025165760",
    "type":"TRAFFIC",
    "timestamp":1499025165760
}
```

Mensaje T2: Mensaje de actualización de posición de vehículo. Topic: road/R2s2/traffic

```
{
    "msg":{
        "vehicle-role":"PublicTransport",
        "action":"CHECK_IN",
        "road-segment":"R2s2",
        "position":216,
        "vehicle-id":"3240JVM"
    },
    "id":"MSG_1499025174700",
    "type":"TRAFFIC",
    "timestamp":1499025174700
}
```

Mensaje T3: Salida de un vehículo de un segmento. Topic: road/R2s2/traffic

```
{
    "msg":{
        "vehicle-role":"PublicTransport",
        "action":"VEHICLE_OUT",
        "road-segment":"R2s2",
        "position":750,
        "vehicle-id":"3240JVM"
    },
    "id":"MSG_1499025207707",
    "type":"TRAFFIC",
    "timestamp":1499025207707
}
```

CAPÍTULO 4
Análisis del problema

En el caso de estudio se ha visto cómo funciona la plataforma IoT llamada *SmartTraffic* y se han explicado cada uno de los sistemas que se integran en ella de forma detallada como son el sistema de red de carreteras inteligente, las señalizaciones de tráfico, los vehículos conectados e inteligentes, los incidentes y el simulador.

A su vez, todos ellos generan una cantidad de información importante sobre el estado de sus sistemas que se puede ver y analizar a través de las comunicaciones que también se han definido como son la comunicación directa con la utilización de una API REST por cada uno de los sistemas o la comunicación indirecta a través de colas de mensajería MQTT basado en comunicaciones M2M.

Para simplificar la nomenclatura de estos sistemas a partir de ahora se referenciarán con una sola palabra que se define a continuación:

- Sistema de red de carreteras inteligentes: ROADS.
- Sistema de vehículos conectados e inteligentes: VEHICLES.
- Sistema de señalización del tráfico: SIGNALS.
- Sistema de incidentes: INCIDENTS.
- Sistema de simulación: SIMULATOR.

Se puede ver en la siguiente imagen una simplificación de la plataforma con tres de sus sistemas que se han explicado en el capítulo anterior y como se ha dicho, la utilización por una parte de comunicaciones directas y comunicaciones indirectas.

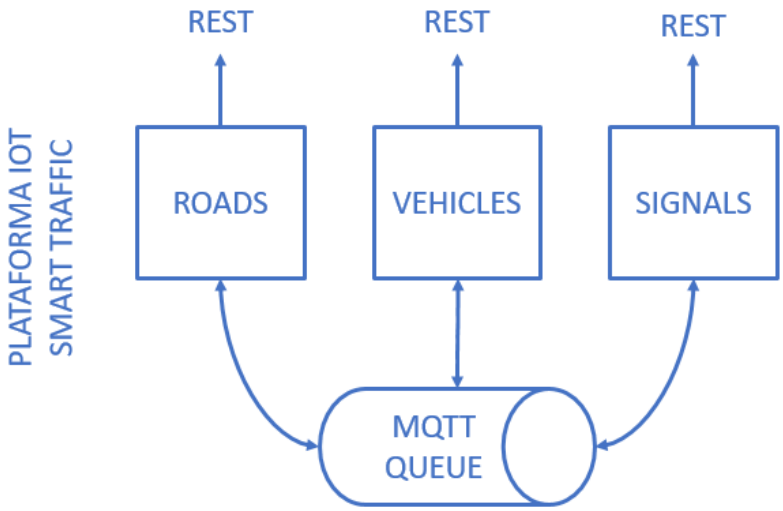

Figura 4.1: Sistema base de la plataforma *SmartTraffic*

Como se ha visto en el esquema, todos los sistemas funcionan de forma independiente y complementaria entre ellos para materializar la idea del funcionamiento de una ciudad a través de una simulación. Pero en la simulación de esta ciudad a través de la plataforma, se puede decir que no se tiene un comportamiento inteligente ya que sus sistemas no son capaces de entenderse y adaptarse en conjunto hacia una auto-gestión autónoma dependiendo de sus necesidades.

Este problema es el que se pretende cambiar y solucionar con el desarrollo del siguiente proyecto y convertir la plataforma de ciudad en inteligente, ya que se dispone de toda la capacidad y potencial suficiente para que tenga cierta autonomía de sus servicios gracias a la gran cantidad de información que ya están generando todos sus sistemas a la plataforma de la Internet de las cosas.

Solo es necesario el desarrollo de unos servicios adaptativos que sean capaces de controlar esta información que se genera por parte de los sistemas y dirigirlos de alguna forma a que cambien a un comportamiento autodirigido dependiendo de las necesidades en conjunto y la demanda de los recursos de la ciudad. Cuando se consiga en todos los aspectos regular y desarrollar estos servicios para cualquier ámbito o recurso de la ciudad entonces se puede llegar a considerar que una ciudad es inteligente y se le puede aplicar el concepto de *Smart City*.

Como no se puede abordar todos los ámbitos y servicios que ofrece una ciudad debido a su gran cantidad y su complejidad, el proyecto se centra en mejorar únicamente un aspecto complejo y variante como es la movilidad y el transporte en las ciudades para desarrollar estos servicios adaptativos y atender en concreto a las necesidades cambiantes del tráfico.

CAPÍTULO 5
Diseño de la solución

5.1 Introducción a los sistemas auto adaptables

Otro de los grandes retos en el desarrollo del proyecto es como diseñar y desarrollar un sistema el cual esté garantizado su funcionamiento en un entorno muy cambiante y responda con precisión y exactitud ante cualquier necesidad por muy imprevista que sea. Es una pregunta que muchos ingenieros de software se han preguntado, cuando se han encontrado en la tesitura de tratar con incertidumbres en los resultados debido a unos conocimientos incompletos en el momento del diseño de un producto.

Algunos ejemplos que pueden ocasionar él encontrarse en la misma situación es cuando tenemos sistemas que:

- Las condiciones de funcionamiento son dinámicas y por lo tanto son muy difíciles de poder predecir.
- Las partes del sistema entran y salen del sistema a voluntad.
- Se produce un cambio en los objetivos en el funcionamiento.

Auto adaptación

La solución en este caso se tiene en la auto adaptación, es decir construir sistemas con la capacidad de adaptarse a sí mismos en entornos cambiantes. Un sistema autoadaptable se define como aquel que mantiene un modelo de sí mismo en tiempo de ejecución. También se le define como el sistema que es capaz de razonar sobre sí mismo y adaptarse para alcanzar las nuevas metas cambiantes.

Una arquitectura de un sistema basado en auto-adaptación es:

- **Separación entre el ámbito de dominio del sistema y del ámbito de adaptación.**

  De esta forma, el dominio del sistema pasa a ser un sistema manejado por el ámbito de adaptación. En el momento que por algún motivo el sistema de adaptación deje de funcionar el dominio del sistema no se verá afectado en absoluto.

- **El ámbito de adaptación es manejado por un bucle de retroalimentación (*feedback loop*).**

  Un bucle de retroalimentación *feedback loop* se define como un conjunto de componentes que comparten un mismo conocimiento sobre el sistema.

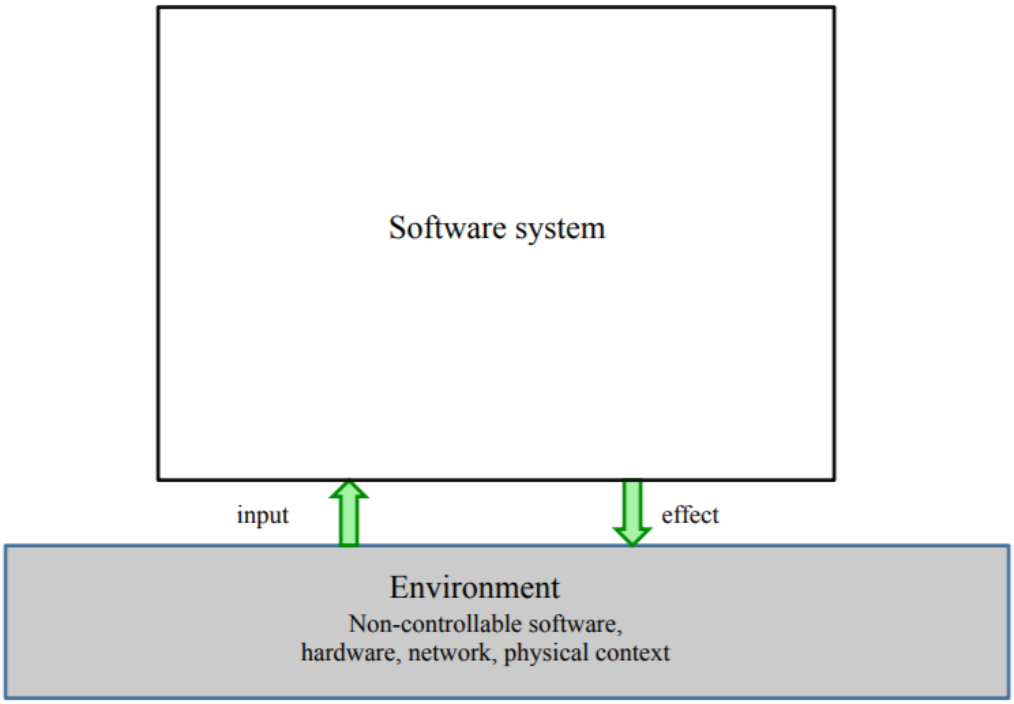

Figura 5.1: Esquema de un sistema no auto adaptable

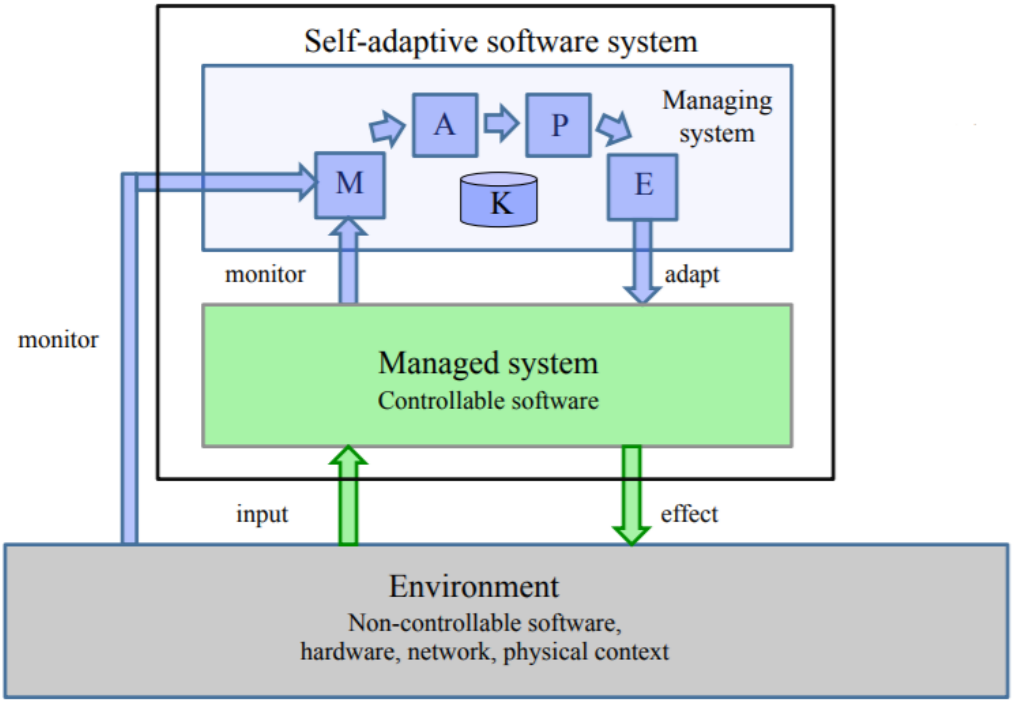

Figura 5.2: Esquema de un sistema auto adaptable

5.2 Computación autónoma y modelo MAPE-K

La computación autónoma propone una estructura con bucles de retroalimentación para tener en cuenta a la hora de desarrollar la parte adaptativa de los sistemas auto adaptables. En este bucle, los sistemas estas especializados con monitores de sensores y con acciones de reconfiguración o actuadores.

Estos dos elementos tienen que tener una estrecha relación con un tercer componente que sea el que los controle y tome la decisión e implemente la estrategia de adaptación dinámica. Se puede ver como referencia de computación autónoma el enfoque del modelo MAPE-K en la siguiente figura.

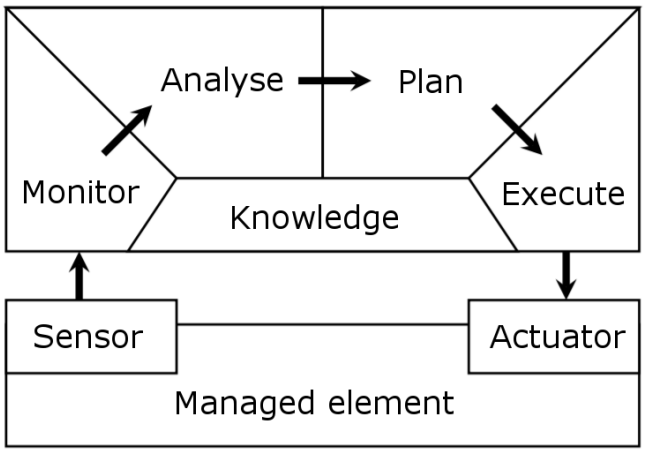

Figura 5.3: Modelo de computación autónoma MAPE-K

Como se puede ver en la figura del modelo MAPE-K este tiene más componentes adicionales que se encargan de controlar a los otros dos descritos anteriormente. Se nombran a continuación todos los componentes del bucle de control:

- Monitor de sensores.
- Análisis de los datos monitorizados.
- Planificación de las acciones de respuesta.
- Ejecución de estas acciones definidas basadas en la representación del sistema bajo administración o control.

Estos bucles pueden diseñarse y desarrollarse de muchas maneras basándose en técnicas de visión artificial, pero es importante destacar que es muy complejo controlar al máximo o incluso dominar el comportamiento de los sistemas automatizados de bucle cerrado con precisión.

5.3 Relación con la teoría de control

La teoría del control ofrece a los diseñadores un marco de métodos y técnicas para construir sistemas automatizados con un comportamiento bien dominado por lo que se pretende que la solución del circuito de retroalimentación para la adaptación, se considere un caso de bucle de control.

Un bucle de control implica tener en el sistema controlado por una serie de sensores y actuadores y a partir de ahí, construir un modelo de comportamiento dinámico sobre el proceso y proporcionar su especificación como el objetivo del control para que se derive el control a estas bases.

Esta metodología no es para nada habitual en ciencias de computación donde por lo general la solución se diseña directamente sin contemplar este tipo de casos.

5.4 Solución hacia un sistema auto adaptable y con bucles de control MAPE-K

Después de haber introducido las tecnologías necesarias para la solución se entiende mejor porque ha sido lo más adecuado aplicarlas todas en el desarrollo del proyecto. Es importante destacar y recordar de nuevo que no se tiene acceso a la implementación de los sistemas base que integra la plataforma IoT por lo que se ha forzado desde un principio a la utilización de esta metodología.

En la solución participan todos y cada uno de los sistemas que forman parte e integran la plataforma *SmartTraffic*, pero con motivo de hacer una simplificación para entender de una mejor forma el modelo de la solución, se nombran de aquí en adelante a los tres principales sistemas que son el de ROADS, VEHICLES y SIGNALS.

Sobre estos principales sistemas (Roads, Vehicles y Signals) y con la intención de ir hacia un sistema auto adaptable se han definido y tratado como sistemas administrados o bajo control de un bucle de control. Para ello, se han puesto diferentes sensores y sondas en cada uno de ellos que un monitor está manejando y que gracias al bucle de control se decidirá aplicar una acción u otra.

Se ha querido desde el principio del proyecto que este tipo de adaptaciones no solo afecten a un único sistema, sino que un único bucle de control pueda afectar según sus directrices a su controlador administrado y este a su vez a otros si es necesario para llevar a cabo la adaptación.

En la siguiente figura se puede observar un esquema de la conversión del sistema base del que se partía hacia el sistema auto adaptativo.

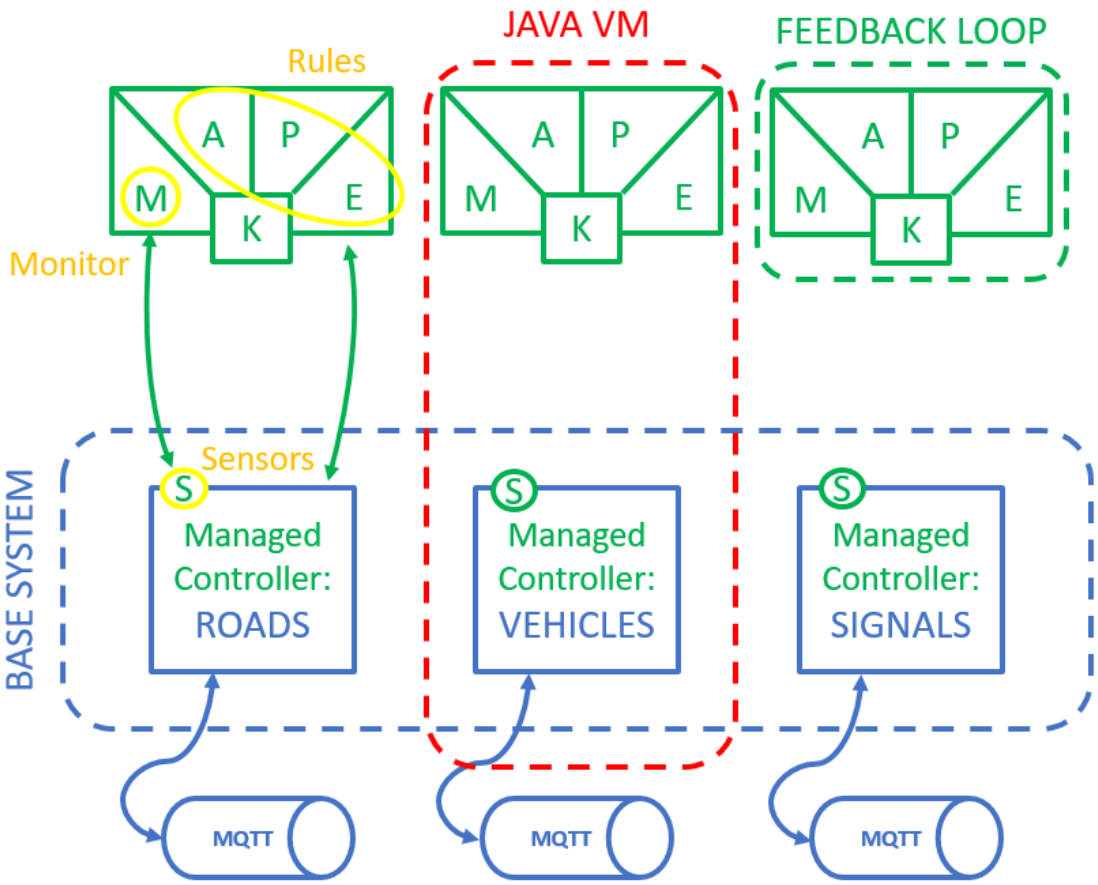

Figura 5.4: Diseño de la solución auto adaptable de *SmartTraffic*

En color azul podemos ver el sistema base desde el que se partía antes de la realización del proyecto, en color verde podemos ver la conversión de hacia el sistema auto adaptativo donde hay un bucle por cada una de las adaptaciones que se quieres hacer sobre los sistemas.

En color amarillo se han destacado los tres términos más importantes en cuanto al desarrollo de la solución de escenarios que destacaremos más adelante como son los sensores, los monitores y las reglas. Para cada uno de los escenarios que se van a desarrollar para probar el sistema se van a definir de muy diferentes tipos.

En color rojo se destaca la posibilidad de que tanto la parte del sistema que llamamos base como la del sistema adaptativo pueden estar ejecutándose en la misma máquina virtual de Java.

Por lo tanto, por cada uno de los escenarios que se han diseñado para el desarrollo de los servicios adaptativos nos encontramos con un bucle de control manejado por una estrategia MAPE-K.

5.5 Escenarios y servicios adaptativos para mejorar la movilidad de una Smart City

Se han estudiado un gran número de diferentes alternativas para demostrar el funcionamiento, la inteligencia y la capacidad de adaptación y auto gestión de una ciudad en materia de movilidad. Finalmente se han decidido diseñar 4 escenarios aplicables a la realidad con diferente tipo de complejidad que solucionan problemas usuales en las ciudades.

Es importante destacar, que los problemas que se mencionan son problemas reales que ocurren día a día en una ciudad con un alto índice de tráfico por sus calles y que tienen un impacto negativo muy considerable en la calidad de vida de una ciudad tanto por la disminución de la productividad, como por el empeoramiento de la calidad del aire, así como por la contaminación acústica que conlleva.

El desarrollo de otros escenarios no vararía mucho de los que se presentan ya que consistirían en aplicar las mismas técnicas y metodologías que se describen a lo largo del documento.

5.5.1. Diseño del escenario 1: Congestión en una carretera por el aumento del tráfico

Problema Real

Se presenta en este escenario la situación real y problema de la congestión de tráfico en las carreteras de una ciudad. Esta congestión normalmente, se produce cuando existe un gran aumento en el volumen del tráfico en un segmento de la carretera en momentos muy específicos del día (horas punta).

Esto desemboca en la generación de una mayor demanda del espacio disponible por parte de los vehículos que las carreteras no disponen. Una vez se produce no solo la circulación por esta carretera se ve afectada sino también el de las más cercanas que están comunicadas.

Solución Propuesta

Una vez aparece una **congestión del tráfico** en una carretera es muy **difícil controlarla y evitar que se extiendan** a las demás carreteras por lo que su **solución** pasa por **intentar prevenirlas y de esta forma librarse del problema** desde un principio.

La idea consiste en posicionar una sonda en cada uno de los segmentos de las carreteras de la ciudad y que su principal objetivo sea el de controlar la densidad de tráfico en todo momento del segmento. Toda la información generada por las sondas es enviada a un monitor que se encarga de distinguir las situaciones potenciales que pueden desembocar en el problema del embotellamiento.

A medida que la sonda detecta que en un segmento **aumenta su densidad**, el monitor se encarga de aplicar **restricciones progresivas de velocidad y de la circulación** de cierto tipo de vehículos en el segmento. Una vez se resuelve la situación y la densidad va disminuyendo, el monitor aplica las acciones opuestas que restablecen el flujo normal y la circulación del tráfico.

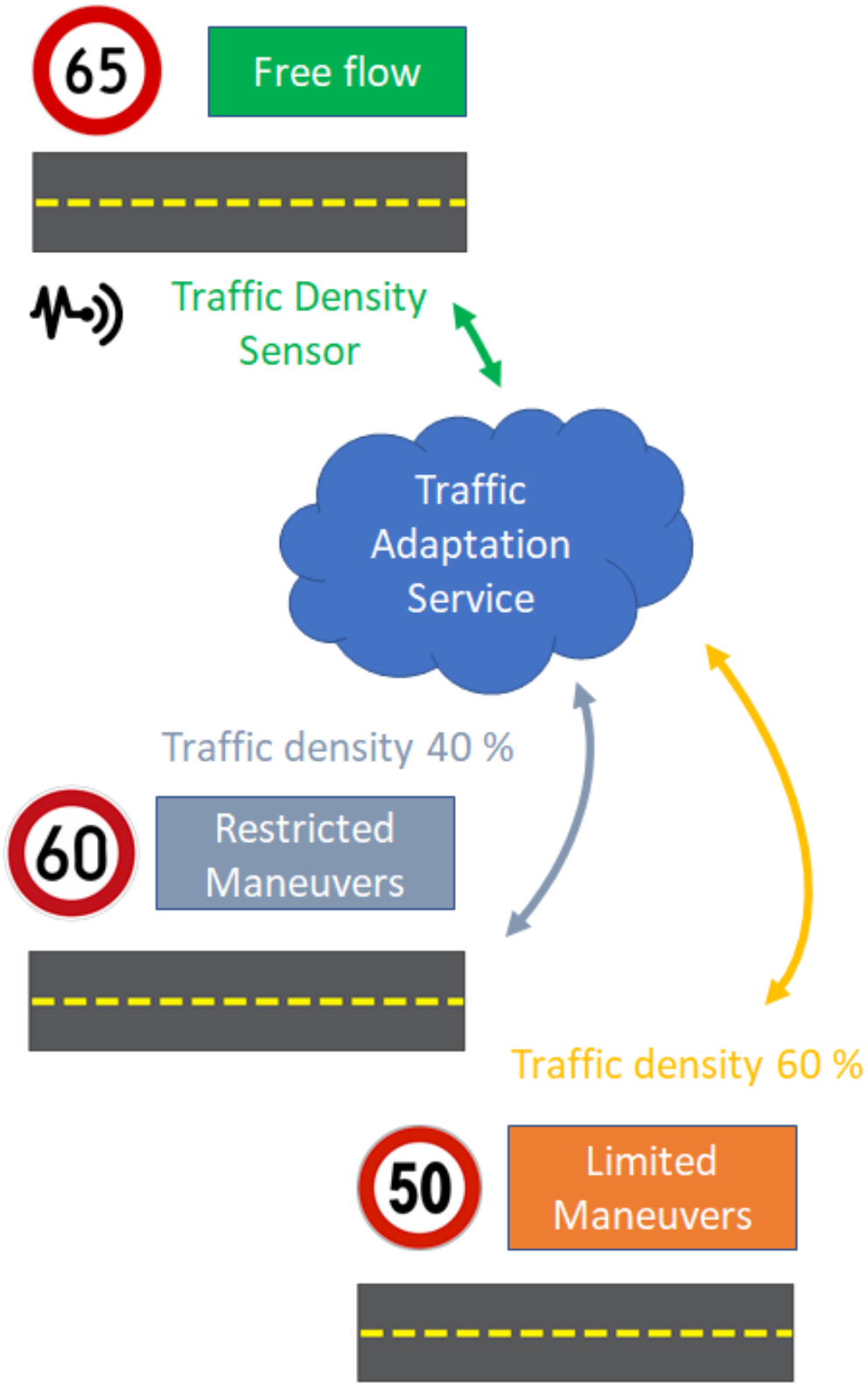

Figura 5.5: Escenario 1: Diseño del servicio adaptable y escenario

Servicio adaptativo a implementar

Escenario 1: Congestión por tráfico en un road-segment		
Sondas	Monitores	Reglas
1. Acciones sobre el sistema de carreteras. (Roads)		
• Detector estado segmento • Sensor densidad de tráfico	• Control de saturación	• Reducción progresiva lím. vel.

Tabla 5.1: Escenario 1: Sondas, Monitores y Reglas

5.5.2. Diseño del escenario 2: Aparición de un accidente en una carretera

Problema Real

Hay situaciones en las que, aunque se intente prevenir como en el escenario anterior la aparición de una congestión en una carretera no es posible por la aparición repentina de accidentes que inevitablemente limitan enormemente la circulación a otros vehículos y el de las carreteras cercanas.

Aquí el problema es distinto al anterior y se agrava porque la congestión no se puede evitar y hay que encontrar opciones de minimizar los daños y además intentar facilitar la atención al accidente.

Solución Propuesta

La solución pasa por minimizar todo lo posible los efectos de la retención y limitar los daños que ocasione solo al segmento donde ocurre. Se plantean por tanto dos posibles estrategias complementarias a abordar, la primera evitar que se extienda a las demás carreteras y la segunda controlar el accidente en el segmento para que no afecte a los demás vehículos.

La idea de evitar que la congestión se extienda a otras calles o segmentos pasa por tener una sonda que controle si está aumentando la densidad de tráfico en el tramo afectado por el accidente, si se supera la mitad de la capacidad del segmento el monitor se encargará de cerrar la carretera para evitar que más coches puedan acceder y agravar la situación. Cuando disminuya la densidad a menos de la mitad de la capacidad y aunque continúe el accidente activo se volverá a abrir al tráfico la carretera.

Para los vehículos que ya estén dentro del segmento afectado por el accidente se verán obligados a cuando crucen el tramo de la incidencia a encender las 4 luces de emergencia y notificárselo a los demás vehículos. Además, si el segmento contiene señalizaciones como semáforos estos se pondrán en modo emergencia para avisar a los demás conductores.

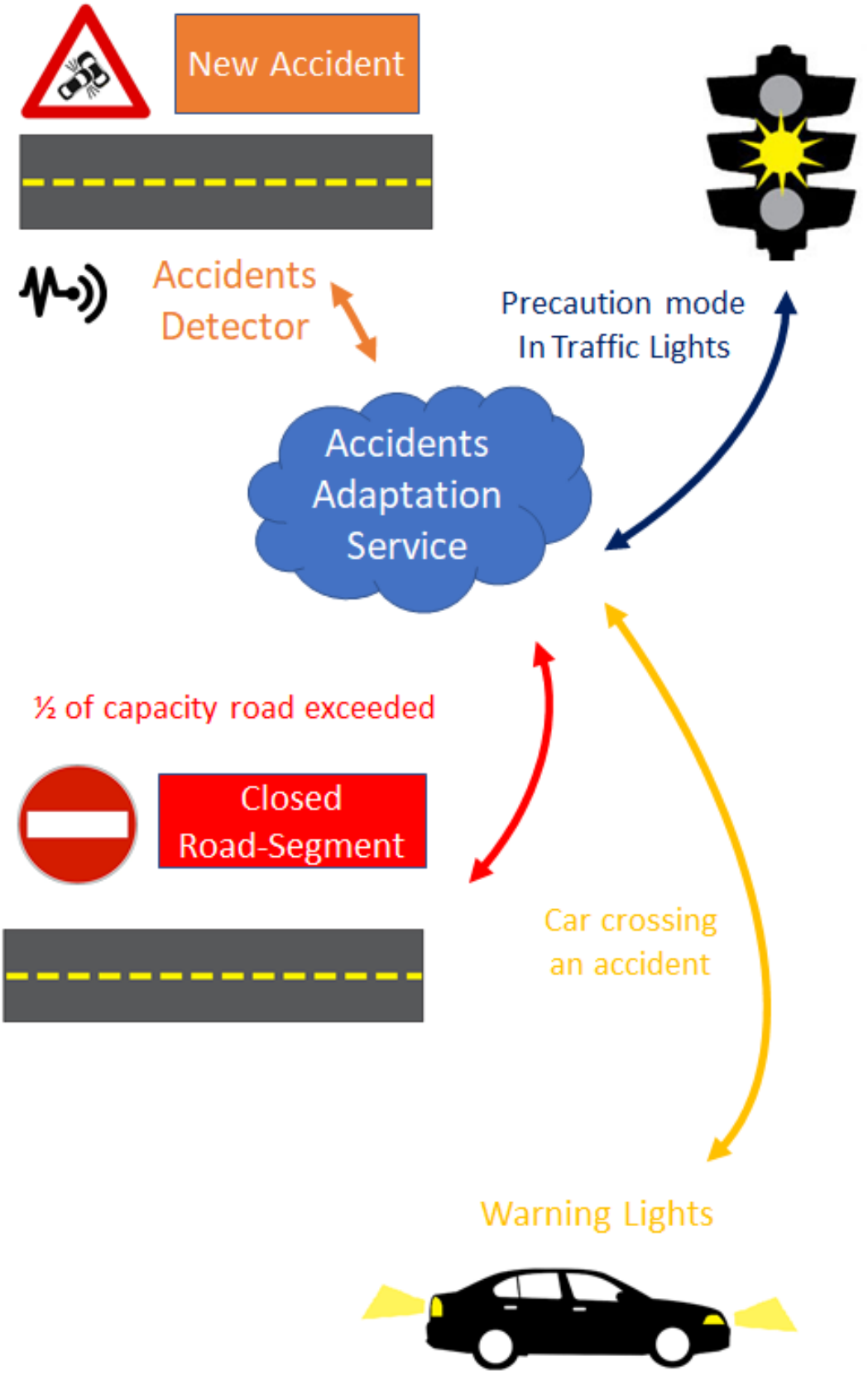

Figura 5.6: Escenario 2: Diseño del servicio adaptable y escenario

Servicio adaptativo a implementar

Al contrario que con el escenario anterior, el diseño de la aparición de un accidente afecta a más de un sistema, en concreto, al sistema de carreteras inteligentes llamado road, a los vehículos inteligentes llamados vehicles, al sistema de señalización del tráfico llamado signals y al sistema de incidentes llamado incidents.

A continuación se puede ver en la siguiente tabla la distinción del servicio en sus diferentes partes y estructurado por el sistema al que afectan.

Escenario 2: Aparición de un accidente en un road-segment		
Sondas	Monitores	Reglas
1. Acciones sobre el sistema de carreteras. (Roads)		
• Detector de accidentes • Sensor densidad de tráfico	• Control de saturación	• Cerrar el road-segment
2. Acciones sobre el sistema de vehículos. (Vehicles)		
• Detector de accidentes • Sensor vehículos en segment	• Coches próx. accidente	• Reducir a la mitad la vel. • Encender 4 intermitentes
3. Acciones sobre el sistema de señalización. (Signals)		
• Detector de accidentes • Sensor semáf. en segment	• Semáf. próx. accidente	• Semáf. precaución emerg.

Tabla 5.2: Escenario 2: Sondas, Monitores y Reglas

5.5.3. Diseño del escenario 3: Aparición de un incidente técnico o emergencia médica en un vehículo

Problema Real

Se presenta otro escenario donde en un momento dado, un vehículo se ve obligado a detener la marcha ya sea porque ha sufrido un accidente, una avería técnica o una emergencia. En este tipo de casos lo primero es la correcta señalización de la incidencia a los demás usuarios de la vía mediante un mecanismo de visibilidad como los triángulos de emergencia o los 4 warnings de emergencia.

Este tipo de situaciones suelen ocurrir en condiciones impredecibles y hay momentos donde no se puede garantizar su realización con las máximas condiciones de seguridad y además ocasionando restricciones a la circulación del tráfico. Mecanismos como los triángulos de emergencia a 50 metros del vehículo han quedado hoy en día del todo obsoletos.

Solución Propuesta

La solución ideada pasa por cuando ocurren este tipo de situaciones, se permanezca el mínimo tiempo posible retenido y obstaculizando el tráfico a los demás usuarios. Además, es importantísimo dar visibilidad a la incidencia y que la comunicación con los demás vehículos, la carretera y la señalización cambie y se produzca de forma instantánea adaptándose a lo ocurrido para evitar que se produzca algún otro accidente.

De esta forma si se tiene una sonda en todos los vehículos que notifican de forma inmediata una parada repentina y no programada, el monitor puede coordinarse con los demás vehículos, carreteras y señalizaciones para comunicar lo ocurrido y publicar un nuevo accidente si es conveniente.

Además, se les da soporte a las emergencias médicas en esta solución adaptativa. Cuando ocurre un accidente o incidencia que requiere de asistencia médica lo primordial es minimizar al máximo el tiempo de respuesta por lo que la sonda puede notificarlo al monitor y este coordinar automáticamente en cuestión de segundo el envío de ambulancia, policía o bomberos.

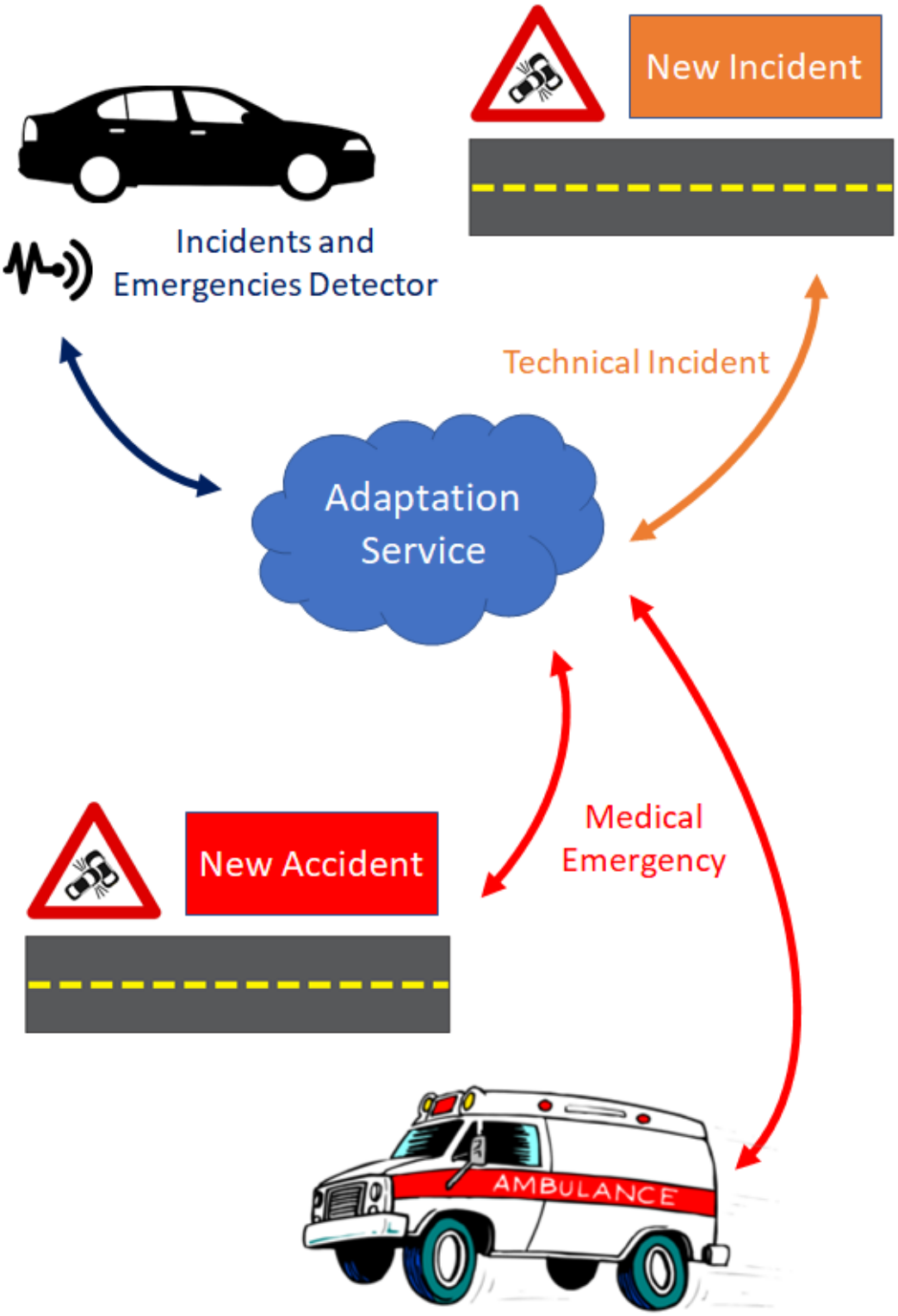

Figura 5.7: Escenario 3: Diseño del servicio adaptable y escenario

Servicio adaptativo a implementar

Este escenario por el contrario de los demás es especial ya que al sistema que afecta es el del simulador. En este caso la detección de un accidente en una carretera es a través de la cola de mensajerías MQTT de la carretera en concreto.

Una vez se recibe se analiza si es necesario asistencia médica y enviar una ambulancia al punto del accidente o si no necesita y solo tiene que publicar el accidente hacia los demás sistemas.

Escenario 3: Incidencias o emergéncias médicas en un vehículo		
Sondas	Monitores	Reglas
1. Acciones sobre varios sistemas y el simulador. (Simulator) Incidencia técnica de un vehículo. (Incident)		
• Detector incidentes coches	• Incidente en roads	• Publicar nuevo accidente en road
Emergencia médica de un vehículo. (Emergency)		
• Detector emerg. en coches • Sensor emerg. médica	• Emergencia en roads	• Publicar nuevo accidente en road • Enviar ambulancia al accidente

Tabla 5.3: Escenario 3: Sondas, Monitores y Reglas

5.5.4. Diseño del escenario 4: Configuración nocturna de la movilidad de una Smart City

Problema Real

En el último escenario desarrollado se plantea la situación de la movilidad nocturna del tráfico en una ciudad inteligente. El gran problema de las ciudades es que se tiene la misma configuración de las carreteras, las mismas normas para los vehículos y la misma señalización por el día y por noche, afectando de manera negativa enormemente a su eficiencia energética.

Es un hecho que el volumen de tráfico no es el mismo por la noche, por lo que en las calles con menos afluencia de coches no tiene sentido que los semáforos sigan su ciclo normal de funcionamiento (verde — ámbar — rojo) dificultando de manera innecesaria la circulación por la ciudad.

También al tratarse de un periodo de descanso la contaminación acústica del tráfico es un problema que se debería de minimizar por la noche y controlarse.

Solución Propuesta

La solución pasa por colocar varios sensores fotoeléctricos alrededor de toda la ciudad y que notifiquen su estado al monitor que controla la adaptación. En el momento en el que establezca la lógica de que se acerca la noche aplicará las diferentes reglas que afectan a diferentes sistemas.

Para todas las carreteras se aplican una restricción del 25 % de su velocidad máxima que se puede sumar a otras restricciones de otros servicios y de esta forma reducir la contaminación acústica que se pudiera producir por parte de los vehículos.

En cuanto a los vehículos se les obliga a que enciendan sus luces de posición como mínimo de forma automática para que puedan desenvolverse mejor en su entorno y se hagan visibles hacia los demás vehículos atendiendo a la nueva condición de oscuridad.

Para solucionar el problema de los semáforos en calles con poca afluencia de coches por la noche y atendiendo siempre al tráfico cambiante, se cambia el ciclo normal de su funcionamiento (verde – ámbar – rojo) a modo precaución (parpadeo de color ámbar) y así eliminar de forma inmediata las esperas innecesarias que producían.

Cuando los sensores detectan la llegada del día y de la luz aplican las acciones opuestas para restablecer la ciudad a su funcionamiento y configuración normal. En el caso de los vehículos ahora pasará a ser opcional tener las luces de posición encendidas.

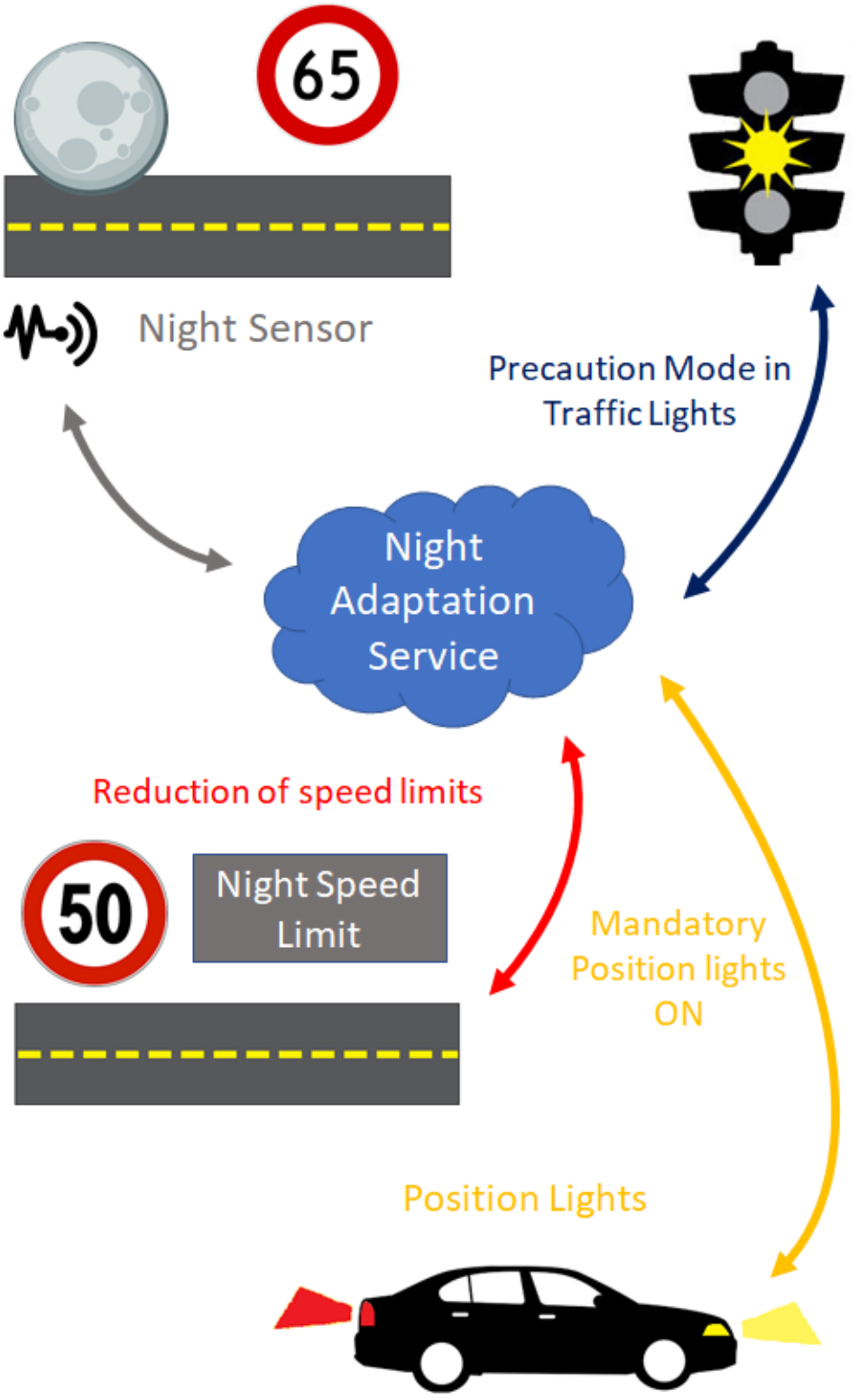

Figura 5.8: Escenario 4: Diseño del servicio adaptable y escenario

Servicio adaptativo a implementar

Escenario 4: Configuración nocturna en la Smart City		
Sondas	Monitores	Reglas
1. Acciones sobre el sistema de carreteras. (Roads)		
• Sensores fotoeléctricos (Oscuridad)	• Roads conf. noche	• Reducir 25 % velocidad de roads
2. Acciones sobre el sistema de vehículos. (Vehicles)		
• Sensores fotoeléctricos (Oscuridad)	• Vehciles conf. noche	• Encender las luces de vehículos
3. Acciones sobre el sistema de señalización. (Signals)		
• Sensores fotoeléctricos (Oscuridad)	• Signals conf. noche	• Semáforos config. precaución (ambar)

Tabla 5.4: Escenario 4: Sondas, Monitores y Reglas

CAPÍTULO 6
Implementación

En el siguiente capítulo, se va a presentar el desarrollo de los cuatro escenarios que se han planteado en el capítulo anterior sobre el diseño de soluciones con el objetivo de mejorar la movilidad de una ciudad inteligente mediante servicios adaptativos.

Para cada escenario se van a introducir de forma detallada los siguientes aspectos:

- Utilización de patrones de diseño y metodología a la hora de implementar el código de los servicios adaptativos.
- Diagramas de flujo que explican el comportamiento del sistema desde que se reciben las muestras por parte de los sensores hasta que se aplican las reglas a los sistemas manejados y administrados.
- Implementación del código más relevante y características especiales del escenario.

6.1 Desarrollo del escenario 1: Congestión en una carretera por el aumento del tráfico

6.1.1. Patrón de implementación

Según las directrices y el diseño que se ha definido en el capítulo anterior sobre el diseño de las soluciones, este primer escenario intenta solucionar el problema de las congestiones en las carreteras a través de métodos que pretenden evitarlas desde un principio en la medida de lo posible.

El desarrollo del código que implementa el servicio como se comentará más adelante se ha estructurado en tres proyectos donde cada uno maneja una parte distinta de la adaptación y utilizando la tecnología de módulos y el entorno de OSGi para Java.

A continuación, se muestra una figura donde se puede ver de forma gráfica la estructura de la adaptación en el entorno de desarrollo Eclipse plug-in development IDE.

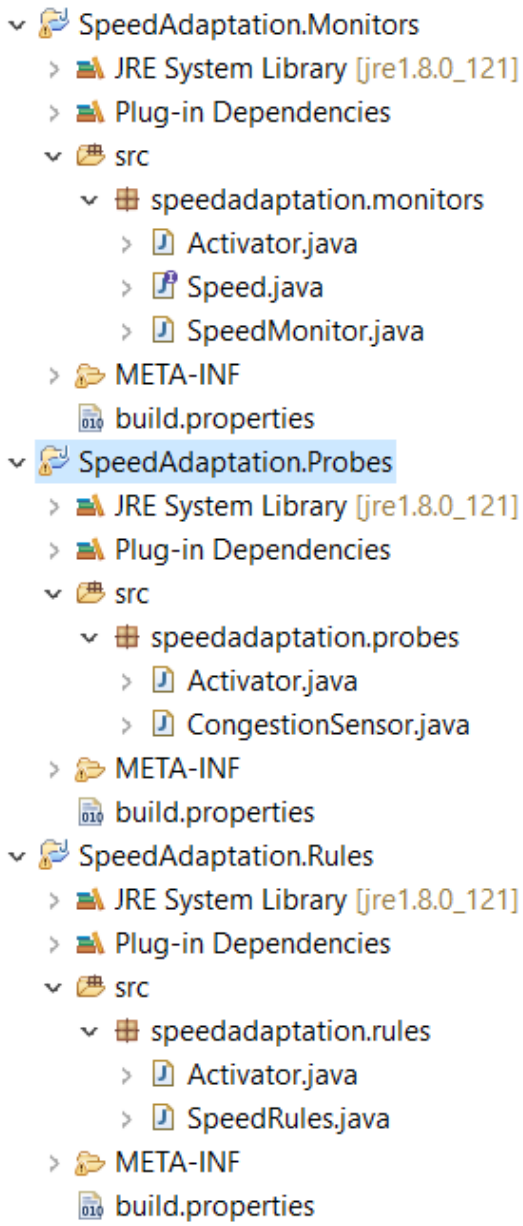

Figura 6.1: Escenario 1: Proyectos eclipse

El codename utilizado para el desarrollo del código del servicio es el de "SpeedAdaptation" y su estructura de proyectos que se observa se describe a continuación.

- El proyecto "SpeedAdaptation.Probes" define la parte de las sondas, detectores y sensores que se incluyen dentro del sistema manejado por la adaptación para la recolección de datos.

  En este caso y con el escenario propuesto, el único sistema controlado por el servicio es el de las carreteras inteligentes.

  La clase "CongestionSensor.java" es la que se encarga de implementar el sensor que toma muestras de las densidades de tráfico de las carreteras.

La clase Activator.java es la que se encarga de poner en marcha el sensor de congestión una vez se decide activar la adaptación.

- El proyecto "SpeedAdaptation.Monitors" define la parte de los monitores que reciben la información por parte de las sondas y aplican una lógica autónoma para decidir que tipo de reglas se deberían de aplicar para adaptarse de la mejor forma posible al contexto de la situación.

  En este caso y con el escenario propuesto, solo existe un monitor que controla al sensor de congestión que hemos definido en el punto anterior.

  La interfaz "Speed.java" y la clase que lo implementa "SpeedMonitor.java" es la que se encarga de implementar el servicio monitor de congestión.

  La clase Activator.java es la que se encarga de registrar el servicio del monitor de congestión dentro del entorno de ejecución de OSGi para que pueda ser llamada una única instancia del mismo monitor desde otros proyectos como el de las sondas o las reglas.

- El proyecto "SpeedAdaptation.Rules" define la parte de las reglas que el monitor se encarga de comunicarle que las aplique en el sistema controlado para que se ejecuten los cambios.

  La clase "SpeedRules.java" es la que se encarga de implementar las ordenes que cambiaran el comportamiento del sistema final manejado.

  La clase Activator.java es la que se encarga de poner en marcha las reglas actuales a aplicar una vez se decide activar la adaptación.

6.1.2. Diagramas de flujo

El diagrama de flujo que se muestra en la figura 6.2 se puede ver el proceso que conlleva desde que los sensores capan la información en cada una de los segmentos de las carreteras y hasta que se aplica una normativa que regula su comportamiento.

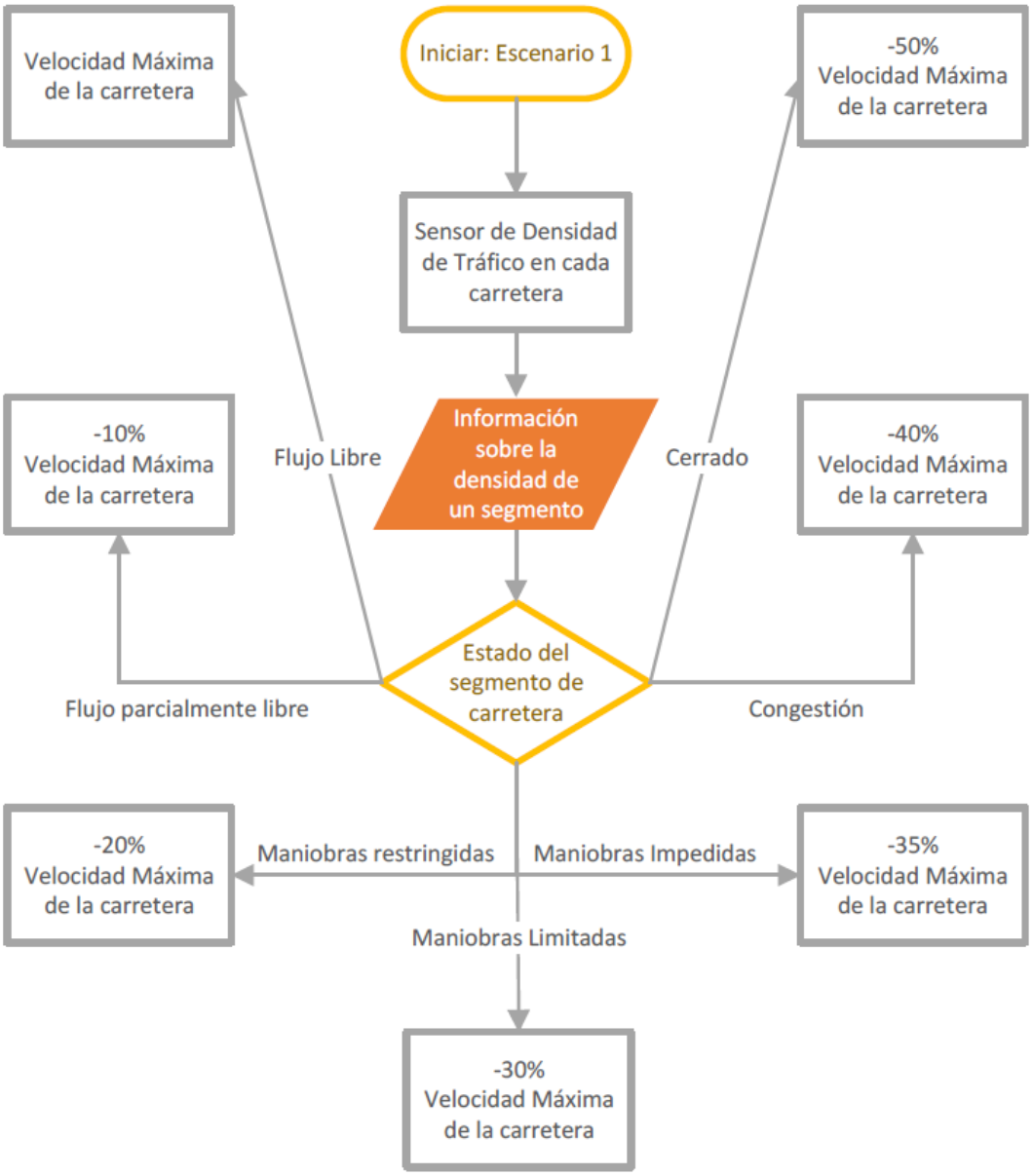

Figura 6.2: Escenario 1: Diagrama de flujo

Se explica que, en primer lugar, hay un sensor de la densidad de tráfico ubicado en cada una de las carreteras de la ciudad inteligente. Este sensor va tomando medidas y se las va transmitiendo a su monitor que se encargará de la valoración y reacción si lo considera oportuno.

En el caso de que el monitor decida que se puede producir una congestión de tráfico debido a que los valores de la densidad en una carretera en concreto están aumentando, entonces decidirá aplicar medidas restrictivas en cuanto a la velocidad para intentar en la medida de lo posible evitar el atasco. Estas medidas restrictivas son progresivas, esto quiere decir que, si el monitor ve que no están surtiendo efecto empezará a aplicar de forma más agresiva y restrictiva.

Aun así, el servicio adaptativo no puede controlar otros factores externos que no sean el aumento del tráfico en una hora punta. Como se comentaba, factores externos como la aparición de un accidente que no se puede prevenir son casos en los que no es posible evitar la congestión de una carretera en concreto. En escenarios posteriores se verá cómo se puede reducir esta situación y como tratarla de la mejor manera posible.

Lo lógica del monitor consiste en relacionar la densidad de vehículos con el estado de la carretera. Por norma general, el sistema seguirá el siguiente comportamiento:

- Con un estado de “Flujo Libre” en un segmento, el límite de la carretera será el máximo.
- Con un estado de “Flujo Parcialmente Libre” en un segmento, el límite de la carretera se reducirá un 10 %.
- Con un estado de “Maniobras restringidas” en un segmento, el límite de la carretera se reducirá un 20 %.
- Con un estado de “Maniobras limitadas” en un segmento, el límite de la carretera se reducirá un 30 %.
- Con un estado de “Maniobras impedidas” en un segmento, el límite de la carretera se reducirá un 35 %.
- Con un estado de “Congestión” en un segmento, el límite de la carretera se reducirá un 40 %.
- Con un estado de “Segmento cerrado” en un segmento, el límite de la carretera se reducirá un 50 %.

6.1.3. Implementación destacable del código

Seguidamente, se va a exponer la implementación del código en Java de las partes más claves del escenario y se explica de forma detallada su funcionamiento mediante comentarios en las lineas del código.

Sensor sobre el sistema de carreteras: CongestionSensor.java

```
1
2  //El sensor implementa el servicio Listener que administra al sistema de
       carrteras
3  public class CongestionSensor implements ServiceListener{
4
5    protected BundleContext context;
6    RoadSegment rs;
7    SpeedMonitor sm;
8
9      //Constructor de la clase Sensor
10   public CongestionSensor(BundleContext context){
11     this.context = context;
12
13         //Se obtiene la referencia del servicio Monitor que administra al
               sistema de carreteras
14     ServiceReference<?> serviceReference = context.getServiceReference(Speed.
           class.getName());
15     sm = (SpeedMonitor) context.getService(serviceReference);
16
17         //Se anade un listener sobre todas las carreteras de la ciudad
18     try{
19       context.addServiceListener(this, "(" + Constants.OBJECTCLASS + "=" +
             IRoadSegment.class.getName() + ")");
20     } catch (InvalidSyntaxException e) {
21       // TODO: handle exception
22       e.printStackTrace();
23     }
24   }
25
26   //Cuando ocurre algun cambio en alguna carretera, se notifica el estado al
         monitor
27   @Override
28   public void serviceChanged(ServiceEvent event) {
29     rs = (RoadSegment) context.getService(event.getServiceReference());
30
31     switch (event.getType()) {
32     case ServiceEvent.MODIFIED:
33         sm.receiveMeasure(rs, rs.getStatus().toString());
34     }
35   }
36
37 }
```

Monitor sobre la adaptación al sistema de carreteras: SpeedMonitor.java

```
1
2  //El monitor implementa la interfaz Speed
3  public class SpeedMonitor implements Speed{
4
5    protected BundleContext context;
6    protected SpeedRules srul;
7
8      //Constructor de la clase monitor
9    public SpeedMonitor(BundleContext context) {
10     super();
11     this.context = context;
12     srul = new SpeedRules();
13   }
14
15   //Desde este metodo recibe la informacion desde el sensor CongestionSensor y
         le envia a la clase que implementa las reglas para que aplique las
         limitaciones
16   @Override
17   public void receiveMeasure(RoadSegment rs, String status) {
18     switch(status){
19       case "Free_Flow": srul.updateRoadSpeed(rs, rs.getRoadSegmentMaxSpeed());
20         break;
21       case "Mostly_Free_Flow": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.9)));
22         break;
23       case "Restricted_Manouvers": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.8)));
24         break;
25       case "Limited_Manouvers": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.7)));
26         break;
27       case "No_Manouvers": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.65)));
28         break;
29       case "Collapsed": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.6 )));
30         break;
31       case "Closed": srul.updateRoadSpeed(rs, ( (int) (rs.
             getRoadSegmentMaxSpeed() * 0.5 )));
32         break;
33       default:
34         break;
35     }
36
37   }
38
39 }
```

6.2 Desarrollo del escenario 2: Aparición de un accidente en una carretera

6.2.1. Patrón de implementación

Según las directrices y el diseño que se ha definido en el capítulo anterior sobre el diseño de las soluciones, este segundo escenario intenta solucionar el problema de las congestiones cuando ya han ocurrido debido a la aparición de un accidente. En este tipo de escenarios es muy difícil e improbable predecir cuándo va ocurrir un accidente por lo que se intenta es minimizar los daños producidos por una congestión que, como consecuencia, no se ha podido evitar.

El desarrollo del código que implementa el servicio, se ha estructurado en tres proyectos donde cada uno maneja una parte distinta de la adaptación y utilizando la tecnología de módulos y el entorno de OSGi para Java.

A continuación, se muestra una figura 6.3 donde se puede ver de forma gráfica la estructura de la adaptación en el entorno de desarrollo Eclipse plug-in development IDE.

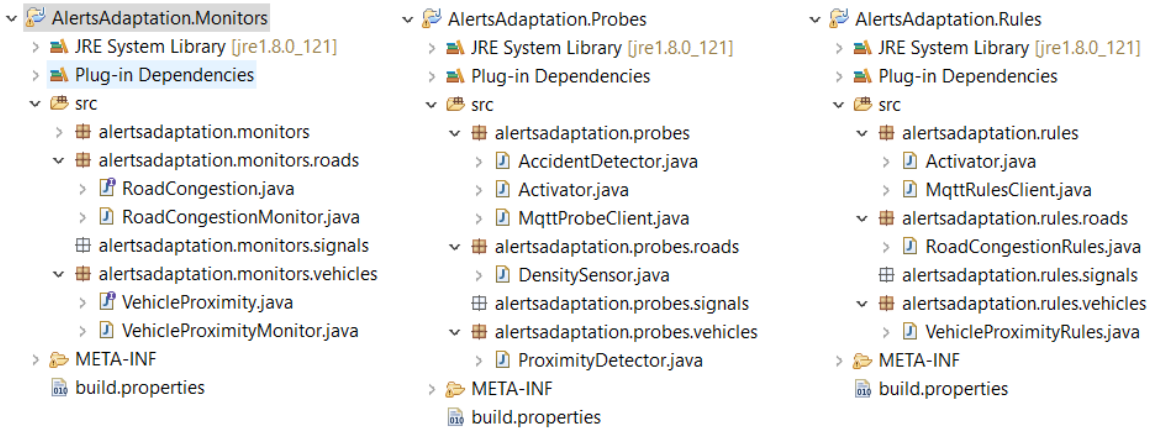

Figura 6.3: Escenario 2: Proyectos eclipse

El codename utilizado para el desarrollo del código del servicio es el de "AlertsAdaptation" y su estructura de proyectos que se observa se describe a continuación.

- El proyecto "AlertsAdaptation.Probes" define la parte de todas las sondas, detectores y sensores que se incluyen dentro de los sistemas manejados por la adaptación para la recolección de datos. En este caso y con el escenario propuesto, los sistemas controlados por las sondas de este proyecto son el de las carreteras inteligentes, los vehículos conectados y las señalizaciones de tráfico.

  Al ser varios sistemas los que son controlados, dentro del proyecto se han definido diferentes paquetes, uno para cada implementación referente a un sistema y uno que utiliza elementos comunes de todos. Estos son:

  - Alertsadaptation.probes (común)
  - Alertsadaptation.probes.roads (específico del sistema de calles)
  - Alertsadaptation.probes.signals (específico del sistema de señales)
  - Alertsadaptation.probes.vehicles (específico del sistema de vehículos)

La clase "AccidentsDetector.java" es la que se encarga de implementar el detector que escucha activamente a través de las comunicaciones indirectas de la plataforma si ha ocurrido alguna incidencia o accidente en una carretera.

Como se explicará más adelante, cuando ocurre una incidencia entonces se activan los diferentes sensores restantes que afectan a los sistemas. Si no hay ninguna incidencia entonces estos sensores permanecen desactivados/inactivos y por lo tanto no se aplican reglas de actuación.

La clase Activator.java es la que se encarga de poner en marcha el detector de accidentes.

- El proyecto "AlertsAdaptation.Monitors" define la parte de los monitores que reciben la información por parte de las sondas y aplican una lógica autónoma para decidir cuál es el próximo movimiento de actuación.

  Este escenario afecta a diferentes sistemas a diferencia del primer como son el de sistemas de carreteras inteligentes, el de vehículos inteligentes conectados y la señalización. Nos encontramos que en el proyecto hay más de una implementación de un monitor que controlará donde cada monitor controlará un sistema independiente.

  Estos monitores son ligeramente distintos al que nos podemos encontrar en el primer escenario, ya que antes de aplicar las reglas de actuación necesita más información de otros sensores, detectores y sus monitores complementarios para llevar a cabo una valoración final.

  Como ya se explicará más adelante, cuando recibe información de un accidente de gestionar la activación o puesta en marcha de otros sensores para recolectar más datos y aplicar restricciones.

  La interfaz "RoadCongestion.java" y la clase que lo implementa "RoadCongestionMonitor.java" es la que se encarga de implementar el servicio monitor de congestión para las carreteras.

  La interfaz "VehicleProximity.java" y la clase que lo implementa "VehicleProximityMonitor.java" es la que se encarga de implementar el servicio monitor de vehículos cercanos a un accidente.

  La clase Activator.java es la que se encarga de registrar los servicios de los monitores dentro del entorno de ejecución de OSGi para que puedan ser llamadas una única instancia del mismo monitor desde otros proyectos como el de las sondas o las reglas.

- El proyecto "AlertsAdaptation.Rules" define la parte de las reglas que los monitores se encargan de comunicarle que las aplique en los sistemas controlados para que se ejecuten los cambios en conjunto.

  La clase "RoadCongestionRules.java" es la que se encarga de implementar las ordenes que cambiaran el comportamiento del sistema de carreteras inteligentes.

  Por otro lado, la clase "VehicleProximityRules.java" es la que se encarga de implementar las ordenes que cambiaran el comportamiento del sistema de vehículos conectados.

  La clase Activator.java es la que se encarga de poner en marcha las reglas actuales a aplicar una vez se decide activar la adaptación.

6.2.2. Diagramas de flujo

En los diagramas de flujo que se van a mostrar en las siguientes figuras, se puede ver el proceso que conlleva desde que el detector de accidentes capta la información desde cada uno de los segmentos de las carreteras y hasta que se aplica una normativa que regula su comportamiento y minimización de daños.

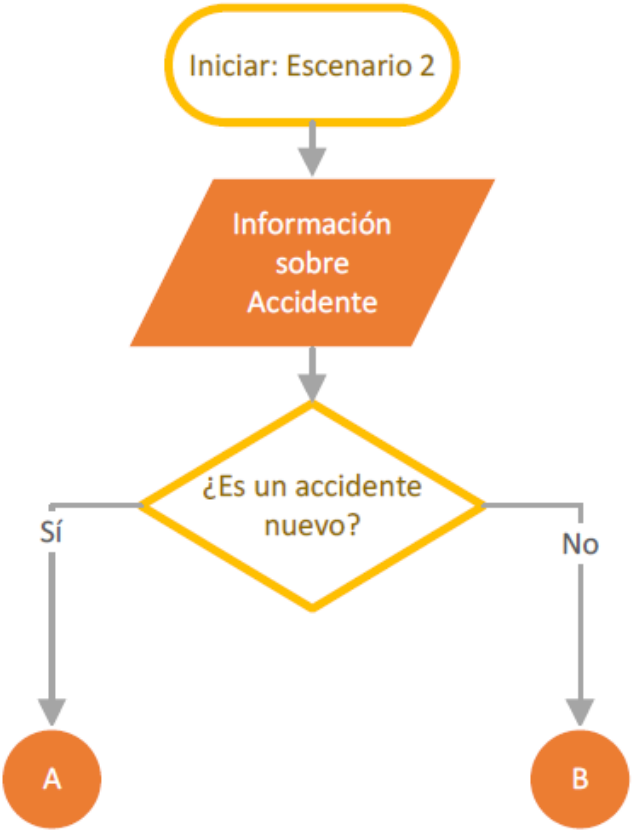

Figura 6.4: Escenario 2: Diagrama de flujo inicial

Mensaje recibido:

```
{
  "msg" : {
    "incident-type" : "OTHER",
    "rt" : "traffic::incident",
    "starting-position" : 300,
    "link" : "/incident/1152445151",
    "description" : "Accidente de Trafico con colision multiple",
    "id" : "1152445151",
    "road-segment" : "R2s2",
    "ending-position" : 400,
    "status" : "Active"
  },
  "id" : "MSG_1495888770000",
  "type" : "ROAD_INCIDENT",
  "timestamp" : 1495888770000
}
```

En este primer diagrama de la figura 6.4 se muestra el primer paso que es cuando el detector de accidentes recibe un mensaje nuevo que contiene información sobre un accidente. Esta información contiene datos como la carretera, el segmento, el punto o tramo donde está la incidencia y si esta es nueva o una actualización de una antigua alerta ya registrada.

Se puede apreciar que cuando se recibe información sobre un accidente nuevo, dependiendo de si es un accidente nuevo (que no ha sido antes notificado por esta via) se escoge la rama A y si es una actualización como que el accidente ya se ha resuelto se escoge la rama B.

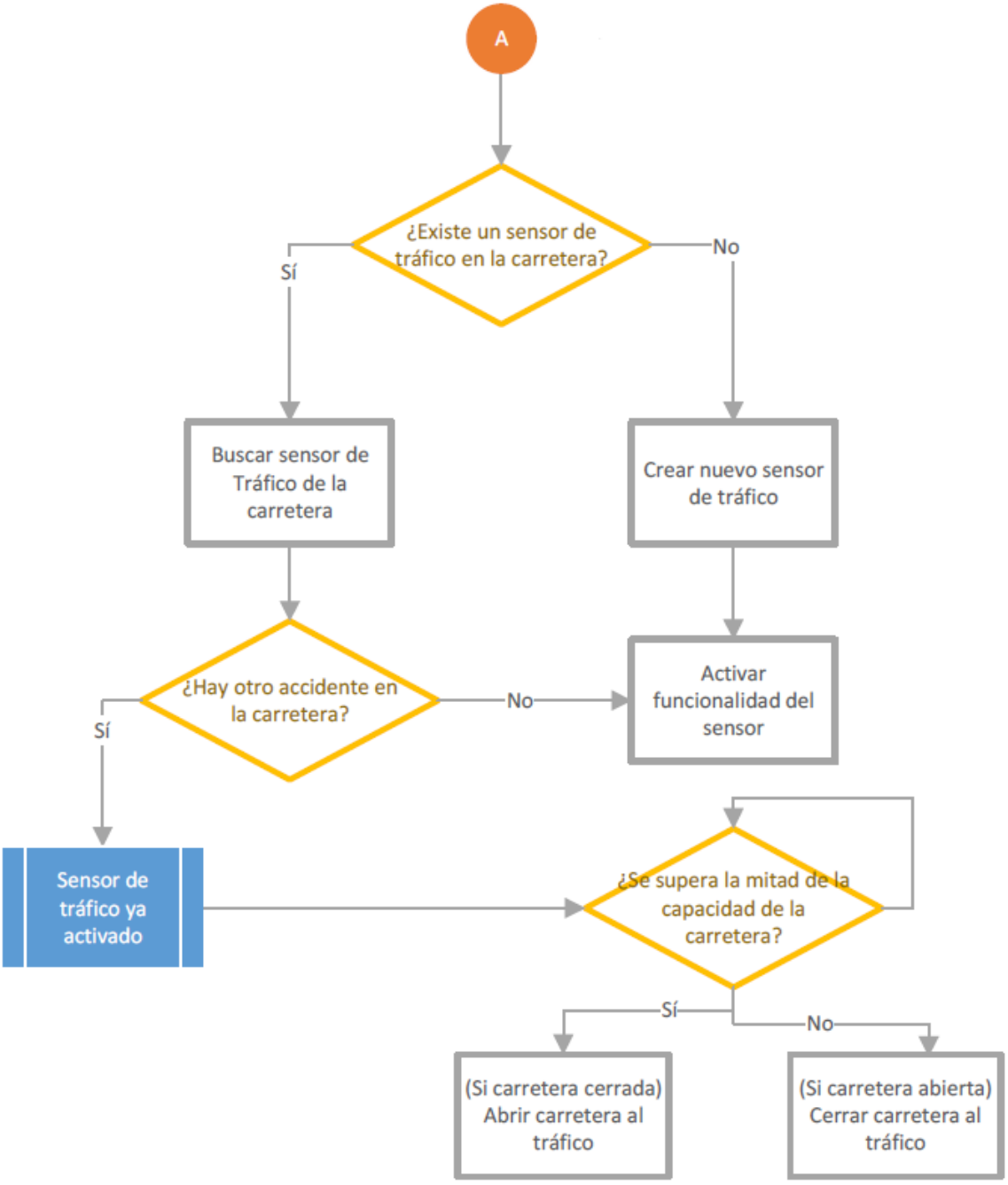

Figura 6.5: Escenario 2: Diagrama de flujo A para carreteras

En la figura 6.5 la información recibida ya se sabe que es sobre un accidente nuevo por lo que el primer paso es analizar de nuevo en que carretera ha ocurrido y averiguar si ya existía un sensor de densidad de tráfico creado para ese segmento de la carretera. Este primer diagrama muestra la acción sobre el sistema de carreteras.

En el caso de que no existiera ningún sensor para ese segmento se creará y se pondrá en funcionamiento. En el caso contrario de que ya existiera un sensor, el servicio buscará su referencia y tratara de localizar el servicio para saber si está en funcionamiento.

El sensor ya puede estar activado bien porque haya otro accidente en el mismo segmento que este controlando (entonces se añadirá que controle el nuevo también) o bien porque anteriormente hubo uno, pero esta desactivado (en este caso se pondrá en funcionamiento). Una vez activado el sensor el sistema se autorregulará y cada vez que se supere la mitad de la capacidad del segmento de carretera se cerrará, cuando disminuya a menos de la mitad se volverá a abrir.

Mientras no se reciba alguna novedad sobre el accidente y se consideré que sigue abierto o en proceso por el sistema, el servicio seguirá autorregulándose y ante los cambios de densidad de la carretera se seguirá abriendo y cerrando.

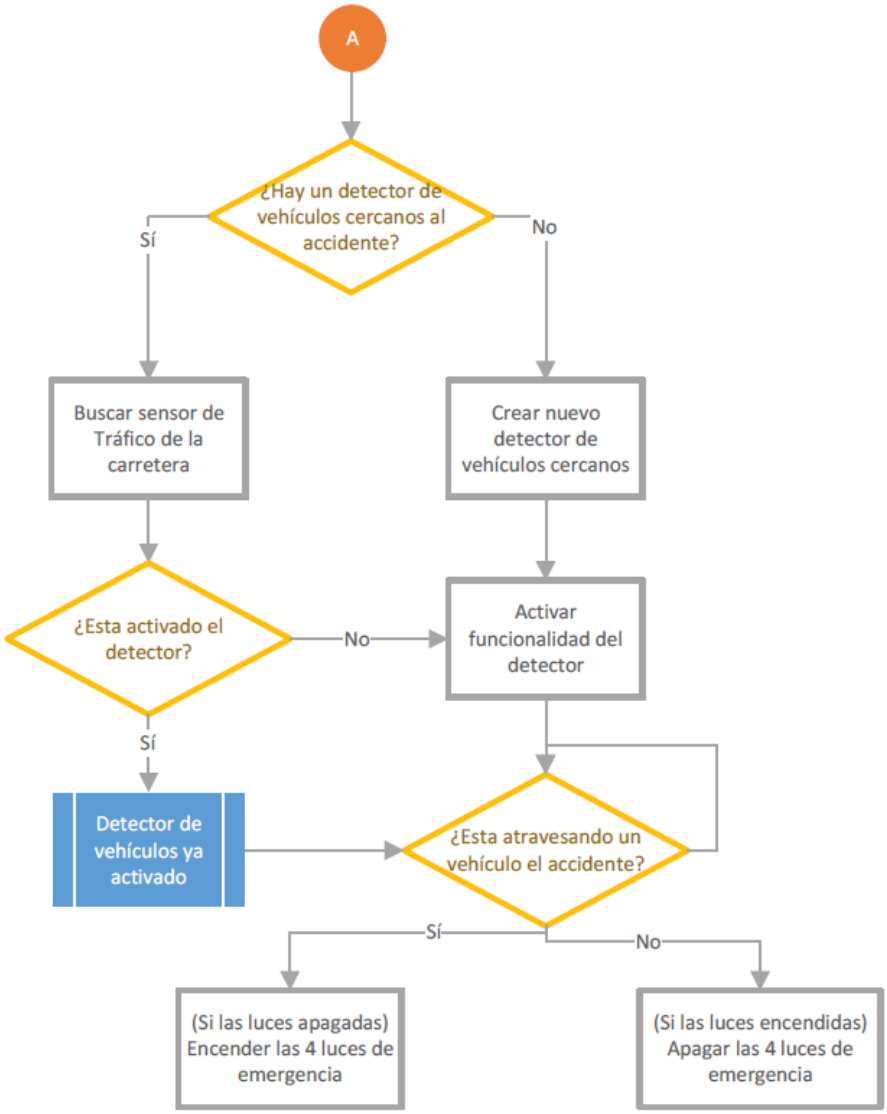

Figura 6.6: Escenario 2: Diagrama de flujo A para vehículos

En la figura 6.6 la información recibida también se sabe ya que es sobre un accidente nuevo, pero con la diferencia de que esta vez cuando se recibe el mensaje, la actuación ya es directamente sobre el sistema de vehículos conectados.

Una vez analizado el mensaje por el monitor se busca el sensor de proximidad de vehículos para ese tramo, aunque es posible que no exista. En el caso de que no exista se crea y activa una nueva instancia del sensor y se posiciona en el segmento y el tramo afectado por el accidente.

Si el sensor ya existía y estaba activo entonces es posible que el accidente ya se hubiera reportado mediante otros sistemas y por tanto ya está funcionando. Por el contrario, si no estaba activo puede ser que ya hubiera un accidente en ese tramo anteriormente y por lo tanto solo hay que volverlo a activar.

Cuando el sensor ya está plenamente operativo se encarga de reportar al monitor los coches que están 100 metros antes del accidente, cruzando el accidente o 100 metros después del tramo del accidente. El monitor en este caso se encargará de obligar al vehículo a encender sus 4 luces de emergencia y que notifique a través de comunicación indirecta por colas mqtt a los demás vehículos de su situación.

Se muestra a continuación, un mensaje de ejemplo que envia un coche a los demás cuando esta atravesando un accidente:

```
{
    "msg":{
     "description" : "Atravesando accidente cercano en carretera",
     "restricted-velocity":30,
       "position":418,
       "vehicle-id":"3240JVM"
    },
    "type":"WARNING_LIGHTS"
}
```

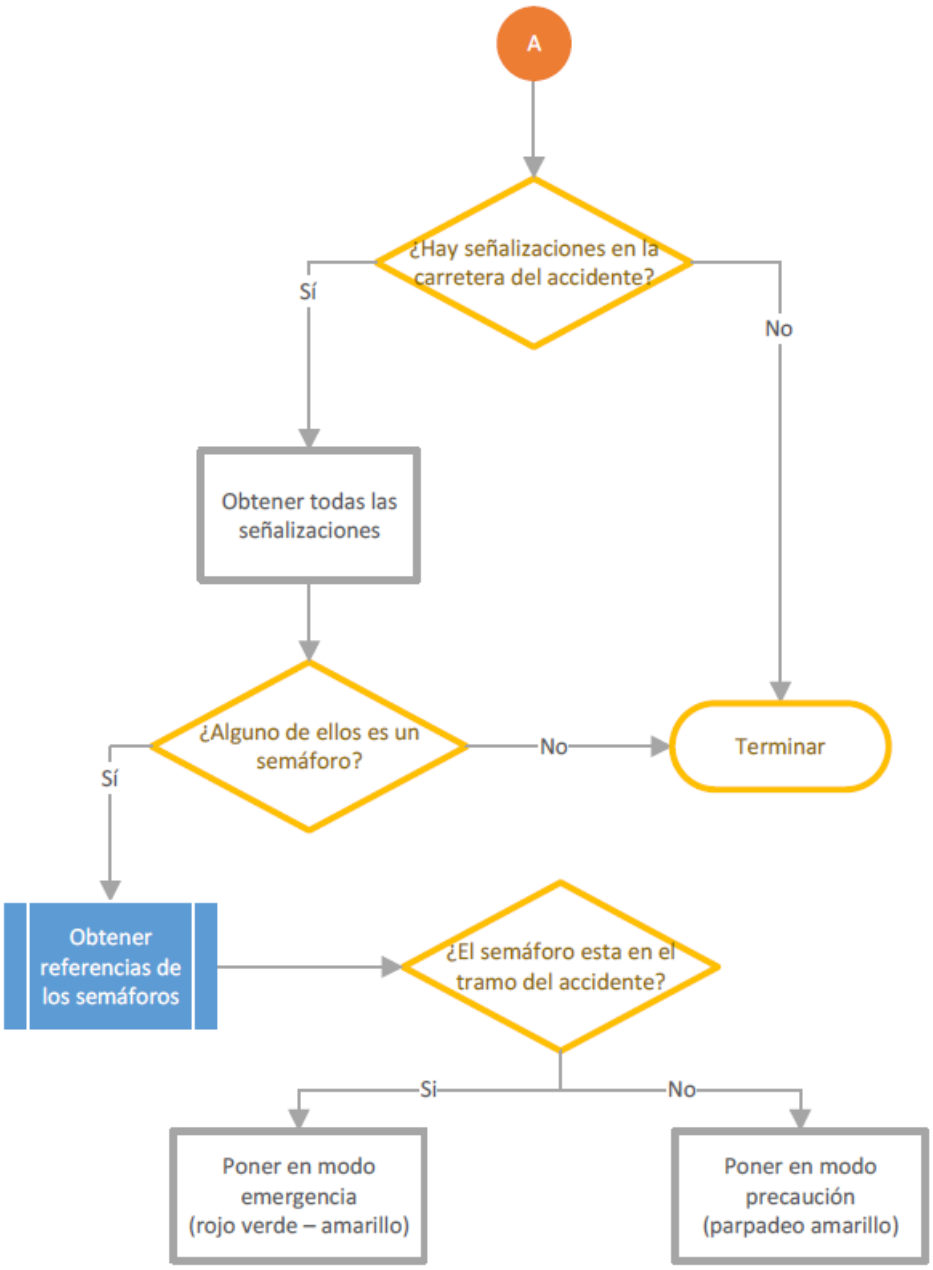

Figura 6.7: Escenario 2: Diagrama de flujo A para señalizaciones

En la figura X.X la información recibida también se sabe ya que es sobre un accidente nuevo como los dos anteriores, pero en este caso cuando se recibe el mensaje, la actuación ya es directamente sobre el sistema de señalización que controla entre otras cosas los semáforos de la ciudad.

Recibida la información, el sensor detectará si hay señalización en el segmento de la carretera donde ha ocurrido el accidente. En la situación de que no haya ninguna señalización, el sistema adaptativo no puede hacer nada y el accidente ya se notificará mediante otras vías a los demás sistemas y usuarios de la ciudad.

En el caso de que, sí que haya, se filtrarán las señalizaciones a solo obtener las referencias de los semáforos ya que es el único elemento del que queremos cambiar su comportamiento. A los semáforos que estén dentro del rango del accidente se le cambiara el ciclo a modo emergencia (parpadeo rojo y verde – parpadeo amarillo) y a los que no estén, pero si dentro del segmento a modo precaución (parpadeo amarillo).

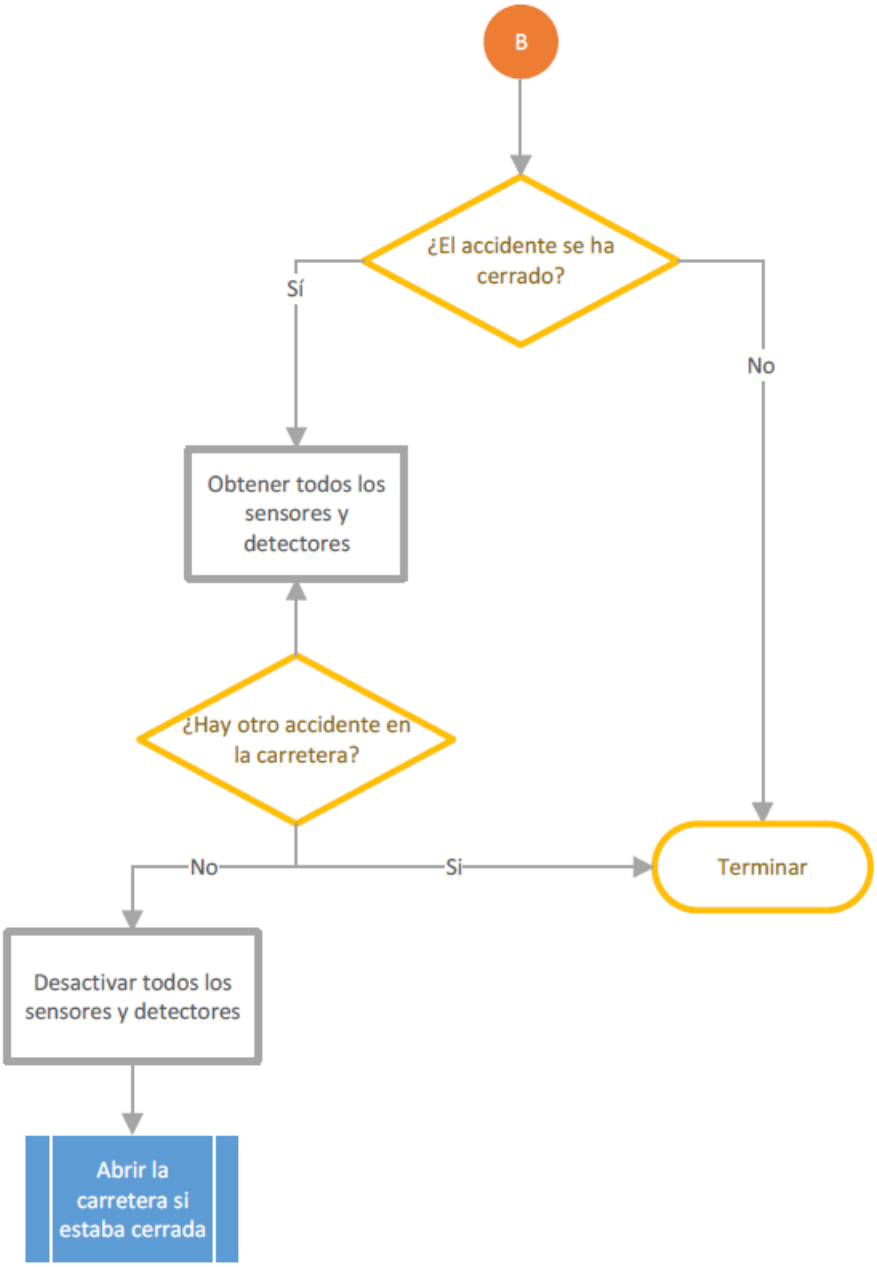

Figura 6.8: Escenario 2: Diagrama de flujo B

En este último diagrama para este escenario de la figura X.X se muestra ya el último paso que es cuando el detector de accidentes recibe un mensaje nuevo que contiene información sobre un accidente que no es nuevo. Este tipo de mensajes puede indicar que el accidente se ha actualizado por ejemplo acotando o expandiendo su tramo o para indicar que ya ha sido resuelto.

El servicio adaptativo solo va a contemplar el caso de que se le envíe un mensaje de actualización que indique que el accidente ha sido resuelto. Antes de volver a restablecer todos los sistemas tiene que tener en cuenta si hubiera aun algún otro accidente activo en el mismo segmento de la carretera (aunque tenga distinto tramo) para en este caso no desactivarlo y esperar que cuando se resuelva este los sistemas vuelvan a la normalidad.

6.2.3. Implementación destacable del código

Seguidamente, se va a exponer la implementación del código en Java de las partes más claves del escenario y se va explicar de forma detallada su funcionamiento mediante comentarios.

Sensor sobre el sistema de vehículos: ProximityDetector.java

```
1  public class ProximityDetector {
2
3    protected MqttProbeClient subscriber = null;
4    protected String id = null;
5    protected VehicleProximityMonitor vpm;
6    protected int startKp;
7    protected int endKp;
8    private int numAccidents;
9
10     //Es el constructor de la clase detector
11   public ProximityDetector(BundleContext context, String id) {
12     this.setId(id);
13     this.numAccidents = 0;
14         //El detector utiliza un cliente mqtt para escuchar los mensajes y
               filtrarlos segun su logica para enviarselos al monitor
15     this.subscriber = new MqttProbeClient(id, this);
16
17         //Se obtiene la referencia del monitor al que se le enviaran los datos
18     ServiceReference<?> serviceReference = context.getServiceReference(
           VehicleProximity.class.getName());
19     vpm = (VehicleProximityMonitor) context.getService(serviceReference);
20   }
21
22     //Metodo utilizado para activar el sensor
23   public void activateSensor(int sKp, int eKp) {
24     this.numAccidents++;
25     this.setStartKp(sKp);
26     this.setEndKp(eKp);
27     this.subscriber.connect();
28     this.subscriber.subscribe(System.getProperty("mqtt.topic.prefix") + "/" +
           System.getProperty("simulator") + "/road/" + id + "/traffic");
29   }
30
31   public boolean isActivated(){
32     return this.numAccidents >0;
33   }
34
35     //Cuando el detector controla mas de un accidente se tiene que alctualizar
           su rango de actuacion
36   public void updateSensorParams(int sKp, int eKp){
37     this.numAccidents++;
38     if(this.startKp > sKp) this.setStartKp(sKp);
39     if(this.endKp < eKp) this.setEndKp(eKp);
40   }
41
42     //Metodo para desactivar el sensor si ya no hay mas accidentes activos que
           este controlando
43   public void deactivateSensor() {
44     this.numAccidents--;
45     if(numAccidents == 0) this.subscriber.disconnect();
46   }
47
48   public String getId() {
49     return id;
50   }
51
```

```
52    private void setId(String id) {
53      this.id = id;
54    }
55
56    public int getStartKp() {
57      return startKp;
58    }
59
60    private void setStartKp(int startKp) {
61      this.startKp = startKp;
62    }
63
64    public int getEndKp() {
65      return endKp;
66    }
67
68    private void setEndKp(int endKp) {
69      this.endKp = endKp;
70    }
71
72      //Metodo que utiliza para filtrar y quedarse con los mensajes que le
          interesan
73      private boolean isValidMeasure(JSONObject msg){
74      try{
75        if(msg.has("msg")){
76          JSONObject aux = msg.getJSONObject("msg");
77          if(aux.has("position")) return true;
78          else return false;
79        } else{
80          return false;
81        }
82      } catch (JSONException e) {
83        e.printStackTrace();
84        return false;
85      }
86    }
87
88    //Metodo utilizado para enviar los mensajes filtrados al monitor que
        administra el sistema
89    public void receiveVehicleClose(JSONObject msg) {
90      if(this.isValidMeasure(msg)){
91        try{
92          JSONObject aux = msg.getJSONObject("msg");
93          vpm.receiveVehicleClose(aux.getString("vehicle-id"), (this.startKp-50
              <= (int) aux.getInt("position") && (int) aux.getInt("position") <=
              this.endKp+50), aux);
94        } catch (JSONException e) {
95          e.printStackTrace();
96        }
97      }
98    }
99  }
```

Monitor de la adaptación sobre el sistema de vehículos: VehicleProximityMonitor.java

```
1
2  //El monitor que administra el sistema de vehiculos implementa la interfaz
       VehicleProximity
3  public class VehicleProximityMonitor implements VehicleProximity {
4
5    protected BundleContext context;
6    protected VehicleProximityRules vpr;
7
8      //Se utiliza un HashMap para que el monitor sepa cuantos detectores, sondas
           o sensores estan creados y posiblemente activos
9    HashMap<String, ProximityDetector> listSensors = new HashMap<String,
       ProximityDetector>();
10
11   public VehicleProximityMonitor(BundleContext context) {
12     super();
13     this.context = context;
14     vpr = new VehicleProximityRules();
15   }
16
17     //Recibe la informacion desde el detector de vehiculos cercanos
18   @Override
19   public void receiveAccident(JSONObject accident) {
20     JSONObject aux;
21     String road_segment, status;
22     int starting_position, ending_position;
23
24     try {
25       if (accident.has("msg")){
26         aux = accident.getJSONObject("msg");
27         if (aux.has("road-segment") && aux.has("status")){
28           road_segment = aux.getString("road-segment");
29           status = aux.getString("status");
30           starting_position = aux.getInt("starting-position");
31           ending_position = aux.getInt("ending-position");
32           //System.out.println("Accidente recibido en " + road_segment + ":" +
                 starting_position + "," + ending_position);
33           System.out.println("-------------------------------------------------
                 ");
34           if(status.equals("Active")){
35             System.out.println("| Accidents Adaptation: New Accident in Road-
                   Segment ");
36             System.out.println("| " );
37             System.out.println("| Status: Active" );
38           } else{
39             System.out.println("| Accidents Adaptation: Closing accident in
                   Road-Segment ");
40             System.out.println("| " );
41             System.out.println("| Status: Closed" );
42           }
43           System.out.println("| Segment " + road_segment + " and position [" +
                 starting_position + "," + ending_position + "]");
44           System.out.println("| Actions: " );
45           if(status.equals("Active")){
46             System.out.println("|   - Activating sensor for congestion control.
                   ");
47             System.out.println("|   - Activating sensor to detect cars near the
                    accident." );
48             System.out.println("|   - Publishing emergency signals in the
                   segment.");
49           } else{
```

```
50              System.out.println("|   - Deactivating sensor for congestion
                    control.");
51              System.out.println("|   - Deactivating sensor to detect cars near
                    the accident." );
52              System.out.println("|   - Removing emergency signals in the segment
                    .");
53            }
54            System.out.println("-------------------------------------------------
                  ");
55            this.updateDensitySensor(road_segment, status, starting_position,
                  ending_position);
56          }
57        }
58      } catch (JSONException e) {
59        e.printStackTrace();
60      }
61
62    }
63
64      //El metodo se encarga de que cuando recibe que un vehiculo esta cerca de
          saber si ya hay un detector para ese tramo activo, si no lo hay lo
          creara y lo metera en el hashmap
65      //Si el detector ya existia pero no estaba activo, se encargara de
          activarlo
66    private void updateDensitySensor(String road_segment, String status, int
        starting_position, int ending_position) {
67      ProximityDetector auxSensor;
68      if(status.equals("Active")){
69        if(this.listSensors.get(road_segment) == null){
70          auxSensor = new ProximityDetector(context, road_segment);
71          this.listSensors.put(road_segment, auxSensor);
72          auxSensor.activateSensor(starting_position, ending_position);
73        } else {
74          auxSensor = listSensors.get(road_segment);
75          if(auxSensor.isActivated()) auxSensor.updateSensorParams(
                starting_position, ending_position);
76          else auxSensor.activateSensor(starting_position, ending_position);
77        }
78      } else if(status.equals("Closed")){
79        auxSensor = listSensors.get(road_segment);
80        if(auxSensor != null) auxSensor.deactivateSensor();
81      }
82
83    }
84
85    //Envia a la clase que implementa las reglas del monitor la informacion
        necesaria para que las aplique y a que vehiculos
86    @Override
87    public void receiveVehicleClose(String vehicle_id, boolean inRange,
        JSONObject msg){
88      vpr.updateRules(context, vehicle_id, inRange, msg);
89    }
90
91  }
```

6.3 Desarrollo del escenario 3: Aparición de un incidente técnico o emergencia médica en un vehículo

6.3.1. Patrón de implementación

Según las directrices y el diseño que se ha definido en el capítulo anterior sobre el diseño de las soluciones, este tercer escenario intenta solucionar un problema sobre cómo reaccionar y de la manera más rápida posible ante una incidencia por parte de un coche en la carretera. En este tipo de escenarios también es muy difícil e improbable predecir cuándo un coche va a detener la marcha y estacionar en la carretera obstaculizando el tráfico porque ha sufrido una avería técnica o una situación de emergencia.

El desarrollo del código que implementa el servicio, se ha estructurado en tres proyectos donde cada uno maneja una parte distinta de la adaptación y utilizando la tecnología de módulos y el entorno de OSGi para Java.

A continuación, se muestra una figura 6.9 donde se puede ver de forma gráfica la estructura de la adaptación en el entorno de desarrollo Eclipse plug-in development IDE.

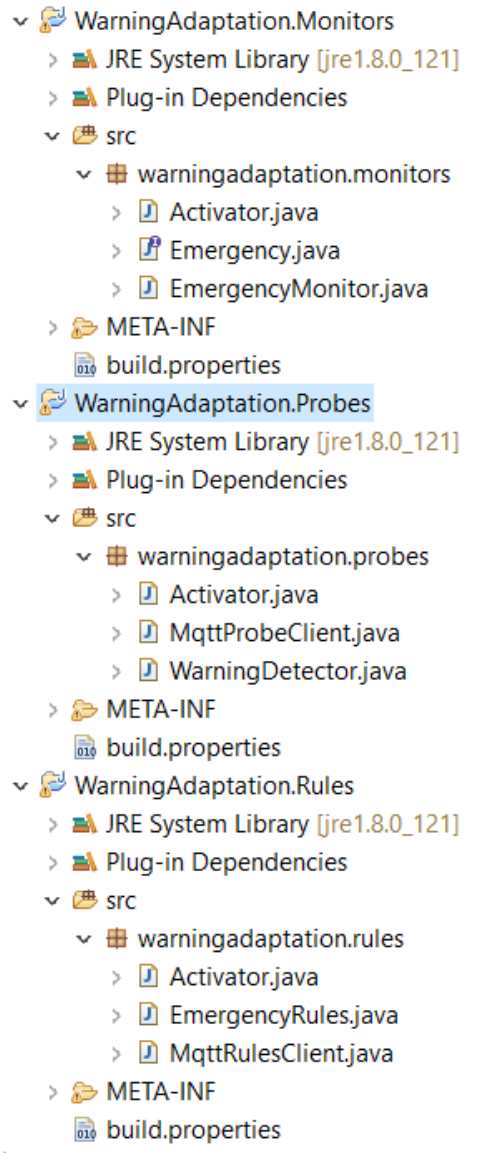

Figura 6.9: Escenario 3: Proyectos eclipse

El codename utilizado para el desarrollo del código del servicio es el de "WarningAdaptation" y su estructura de proyectos que se observa se describe a continuación.

- El proyecto "WarningAdaptation.Probes" define la parte de la sonda, detector o sensor que se incluyen dentro de los sistemas manejados por la adaptación para la recolección de datos. En este caso y con el escenario propuesto, el detector se encarga de escuchar las comunicaciones indirectas gestionadas a través de las colas mqtt para detectar la aparición de un incidente en el simulador.

  El escenario va a afectar al sistema simulador y a las comunicaciones de la plataforma por lo que, dentro del proyecto se ha definido únicamente un solo paquete a diferencia del anterior.

  La clase "WarningDetector.java" es la que se encarga de implementar el detector que escucha activamente a través de las comunicaciones indirectas de la plataforma si ha ocurrido alguna incidencia técnica o emergencia en una carretera.

  La clase Activator.java es la que se encarga de poner en marcha el detector de incidentes.

- El proyecto "WarningAdaptation.Monitors" define la parte del monitor que reciben la información por parte del detector y aplica una lógica autónoma para decidir que tipo de accidente ha de publicar y si es necesario que programe el envío de una ambulancia.

  La interfaz "Emergency.java" y la clase que lo implementa "EmergencyMonitor.java" es la que se encarga de implementar el servicio monitor de incidentes y emergencias para las carreteras.

  La clase Activator.java es la que se encarga de registrar el servicio del monitor dentro del entorno de ejecución de OSGi para que pueda ser llamada una única instancia del mismo monitor desde otros proyectos como el de las sondas o las reglas.

- El proyecto "WarningAdaptation.Rules" define la parte de las reglas que los monitores se encargan de comunicarle que las aplique en el sistema controlado simulador, para que se ejecuten los cambios en conjunto.

  La clase "EmergencyRules.java" es la que se encarga de implementar las ordenes que cambiarán el comportamiento del sistema simulador.

  La clase Activator.java es la que se encarga de poner en marcha las reglas actuales a aplicar una vez se decide activar la adaptación.

6.3.2. Diagramas de flujo

En los diagramas de flujo que se van a mostrar en las siguientes figuras, se puede ver el proceso que conlleva desde que el detector de incidentes capta la información hasta que se aplica una normativa que regula su comportamiento y minimización del tiempo de respuesta.

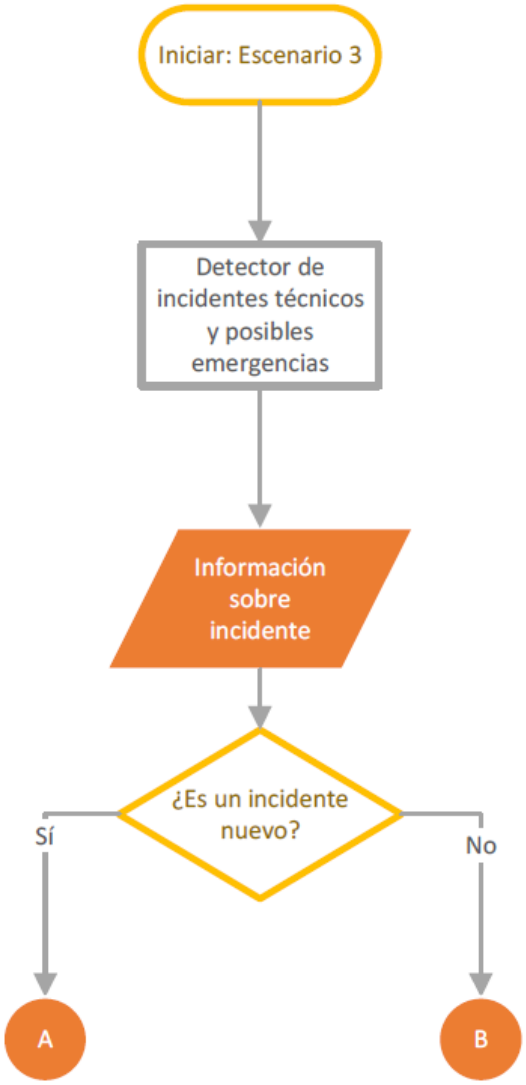

Figura 6.10: Escenario 3: Diagrama de flujo inicial

Mensaje recibido:

```
{
    "msg":{
        "description":"Boton de emergencia pulsado",
        "road-segment":"R2s2",
        "position":400,
        "vehicle-id":"4777DBH",
     "assistance":"yes",
        "status":"Active"
    },
    "type":"WARNING_BUTTON"
}
```

En este primer diagrama de la figura 6.10 se muestra el primer paso que es cuando el detector de incidentes recibe un mensaje nuevo que contiene información sobre un incidente o emergencia. Esta información contiene datos como el tipo de incidente, la carretera, el segmento, el punto o tramo donde está la incidencia, si esta es nueva o una actualización de una antigua alerta ya registrada y si se necesita asistencia médica.

Se puede apreciar que cuando se recibe información sobre un incidente nuevo, dependiendo de si es un incidente nuevo (que no ha sido antes notificado por esta vía) se escoge la rama A y si es una actualización como que el incidente ya se ha resuelto se escoge la rama B.

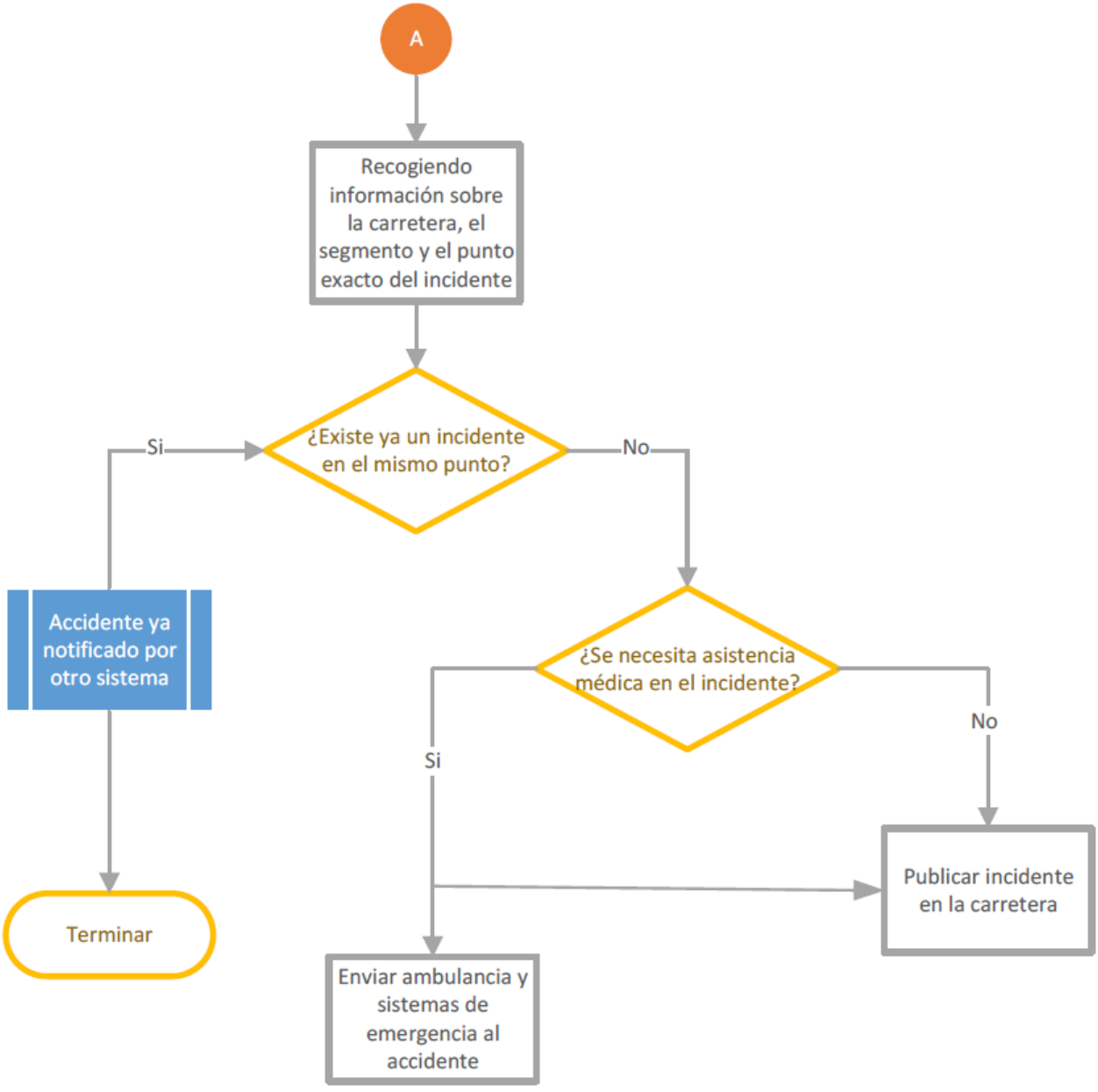

Figura 6.11: Escenario 3: Diagrama de flujo rama A

En la figura 6.11 la información recibida ya se sabe que es sobre un incidente nuevo para el servicio por lo que el primer paso es analizar de nuevo en que carretera ha ocurrido y averiguar si ya existía un incidente creado para ese mismo segmento de la carretera.

En el caso de que no existiera ningún incidente para ese segmento y tramo, entonces el monitor seguirá analizando el mensaje. En el caso contrario de que ya existiera un incidente en ese mismo punto, quiere decir que ya ha sido reportado por otro sistema por lo que no tomará ninguna acción.

Cuando el monitor sigue analizando el mensaje lo primero que intenta es distinguir si se trata de un incidente que se puede solucionar a los pocos minutos o si se trata de una emergencia médica el cual necesita un soporte especial y lo notifica a los demás sistemas. Seguidamente si termina siendo una emergencia se encarga de enviar automáticamente una ambulancia al punto reportado.

La ambulancia se envia mediante una petición a la api rest del recurso del simulador. Se muestra a continuación un mensaje de ejemplo que se envia automáticamente de una ambulancia a un accidente en el punto [R2s2, 400].

```
{
    "id":"Ambulance_1153845111",
    "speed":80,
    "type":"Van",
    "role":"PublicTransport",
    "route":[
        [
            {
                "road":"R2s2",
                "point":0
            },
            {
                "road":"RS1",
                "point":400
            }
        ]
    ]
}
```

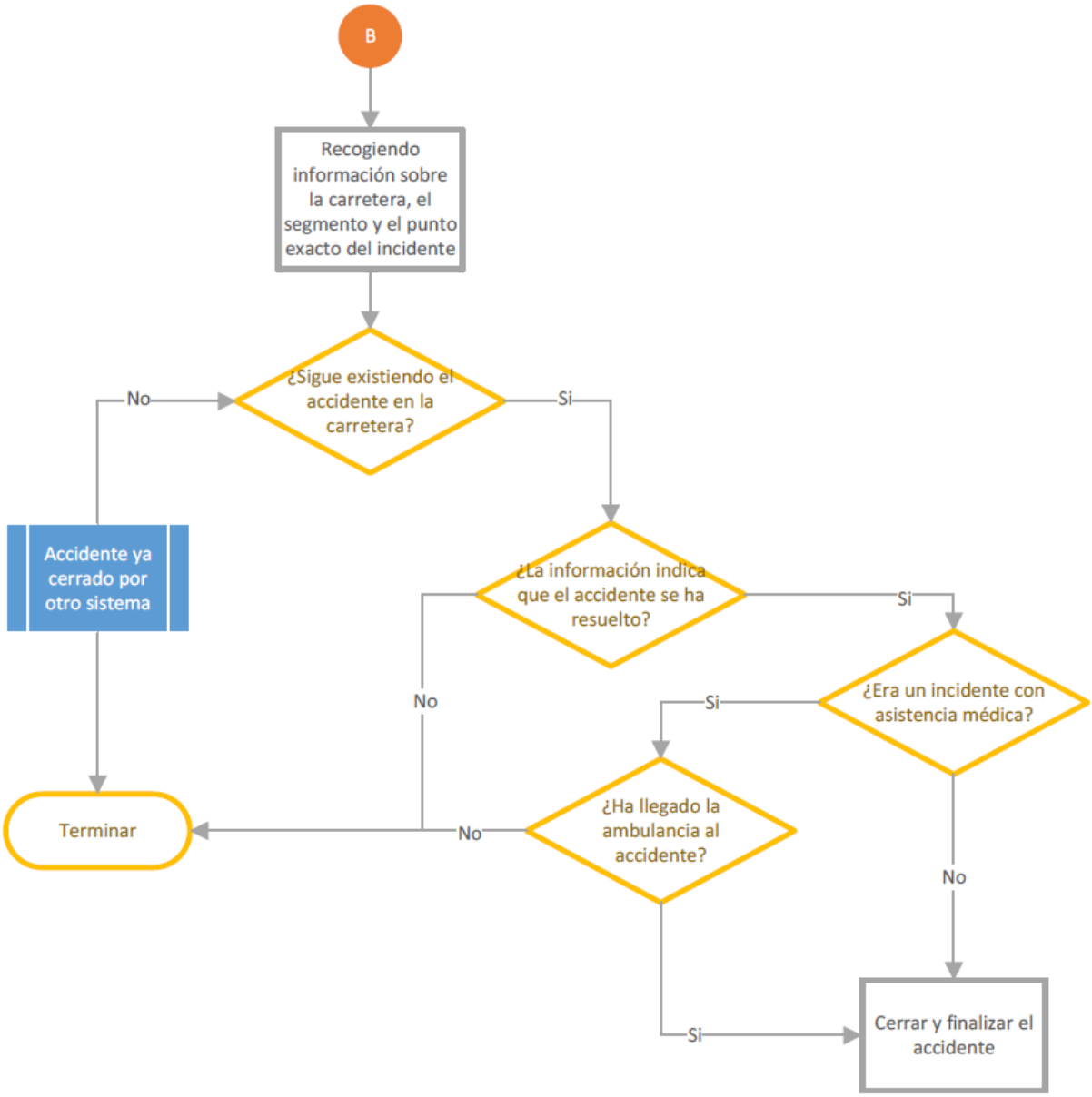

Figura 6.12: Escenario 3: Diagrama de flujo rama B

En este último diagrama para este escenario de la figura 6.12 se muestra ya el último paso que es cuando el detector de incidentes recibe un mensaje nuevo que contiene información de actualización.

El servicio adaptativo solo va a contemplar el caso de que se le envíe un mensaje de actualización que indique que el incidente ha sido resuelto. Antes de volver a restablecer todos los sistemas tiene que tener en cuenta si el accidente requería de un soporte médico y que la ambulancia a llegado al destino correctamente.

En el caso de que la ambulancia no haya llegado al destino, el monitor rechazará la idea de cerrar la emergencia y esperará un nuevo mensaje para volver a comprobarlo. Si no se requería un soporte especial, entonces lo cerrará sin ningún problema.

El siguiente escenario no requiere de una implementación destacable ya que utiliza las mismas técnicas y metodologías que se han utilizado en los anteriores escenarios.

6.4 Desarrollo del escenario 4: Configuración nocturna de la movilidad de una Smart City

6.4.1. Patrón de implementación

El último escenario sigue las mismas directrices y diseño que se han definido en los otros escenarios de forma que el desarrollo del código que implementa el servicio, se ha estructurado en tres proyectos. Cada uno de los tres maneja una parte distinta de la adaptación y utilizan la tecnología de módulos y el entorno de OSGi para Java.

A continuación, se muestra una figura 6.13 donde se puede ver de forma gráfica la estructura de la adaptación en el entorno de desarrollo Eclipse plug-in development IDE.

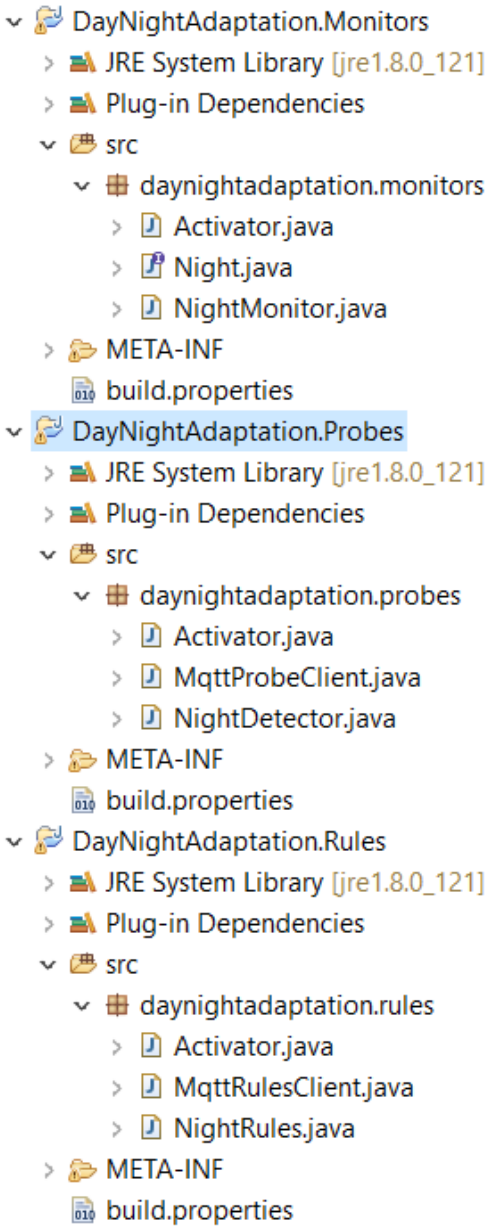

Figura 6.13: Escenario 4: Proyectos eclipse

El codename utilizado para el desarrollo del código del servicio es el de "DayNightAdaptation" y su estructura de proyectos que se observa se describe a continuación.

- El proyecto "DayNightAdaptation.Probes" define la parte de la sonda, detector o sensor que se incluyen dentro de los sistemas manejados por la adaptación para la recolección de datos. En este caso y con el escenario propuesto, el detector se encarga de utilizar las diferentes sondas repartidas por toda la ciudad para detectar si se está en un periodo nocturno.

  El escenario va a afectar al sistema simulador y a todos y cada uno de los sistemas de carreteras inteligentes, vehículos conectados y señalización.

  La clase "NightDetector.java" es la que se encarga de implementar el detector que escucha activamente a través de las comunicaciones indirectas de la plataforma si la ciudad está en un periodo de noche o de día.

  La clase Activator.java es la que se encarga de poner en marcha el detector de incidentes.

- El proyecto "DayNightAdaptation.Monitors" define la parte del monitor que reciben la información por parte del detector y aplica una lógica autónoma para decidir el tipo de restricciones que va a tener que aplicar si se encuentra en un periodo nocturno sobre los demás sistemas.

  La interfaz "Night.java" y la clase que lo implementa "NightMonitor.java" es la que se encarga de implementar el servicio monitor de la configuración nocturna de la ciudad.

  La clase Activator.java es la que se encarga de registrar el servicio del monitor dentro del entorno de ejecución de OSGi para que pueda ser llamada una única instancia del mismo monitor desde otros proyectos como el de las sondas o las reglas.

- El proyecto "DayNightAdaptation.Rules" define la parte de las reglas que los monitores se encargan aplicar a cada uno de los sistemas nombrados anteriormente.

  La clase "NightRules.java" es la que se encarga de implementar las ordenes que cambiarán el comportamiento de los sistemas.

  La clase Activator.java es la que se encarga de poner en marcha las reglas actuales a aplicar una vez se decide activar la adaptación.

6.4.2. Diagramas de flujo

En los diagramas de flujo que se van a mostrar en las siguientes figuras, se puede ver el proceso que conlleva desde que los detectores captan la información desde cada punto de la ciudad coordiandose y hasta que se aplica una normativa que regula su comportamiento de la ciudad por la noche referente a carreteras, vehículos y señalización.

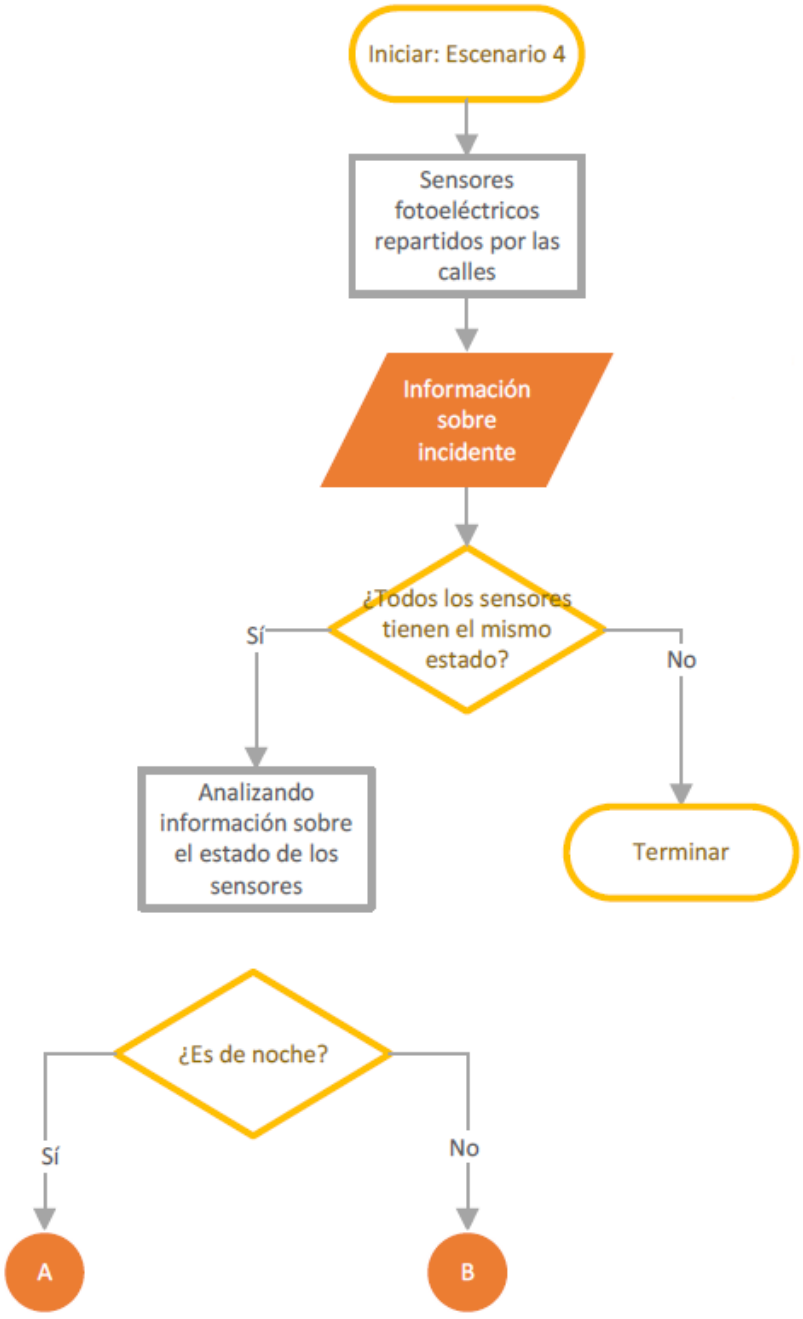

Figura 6.14: Escenario 4: Diagrama de flujo inicial

Mensaje recibido:

```
{
    "msg":{
        "description":"Sensor de luz y oscuridad",
        "status":"Active"
    },
    "type":"NIGHT_CHANGE"
}
```

En este primer diagrama de la figura 6.14 se muestra el primer paso que es cuando el detector recibe un mensaje nuevo que contiene información sobre si es de noche. Esta información tiene que ser coherente en todos los sensores repartidos que lo han recogido por lo que el monitor solo analizará el mensaje si todos tienen la misma medida.

Se puede apreciar que cuando la información ya está validada hay dos tipos de ramas. La rama A es la utilizará el simulador para aplicar la configuración nocturna y la rama B el restablecimiento de todos los sistemas a su funcionamiento normal.

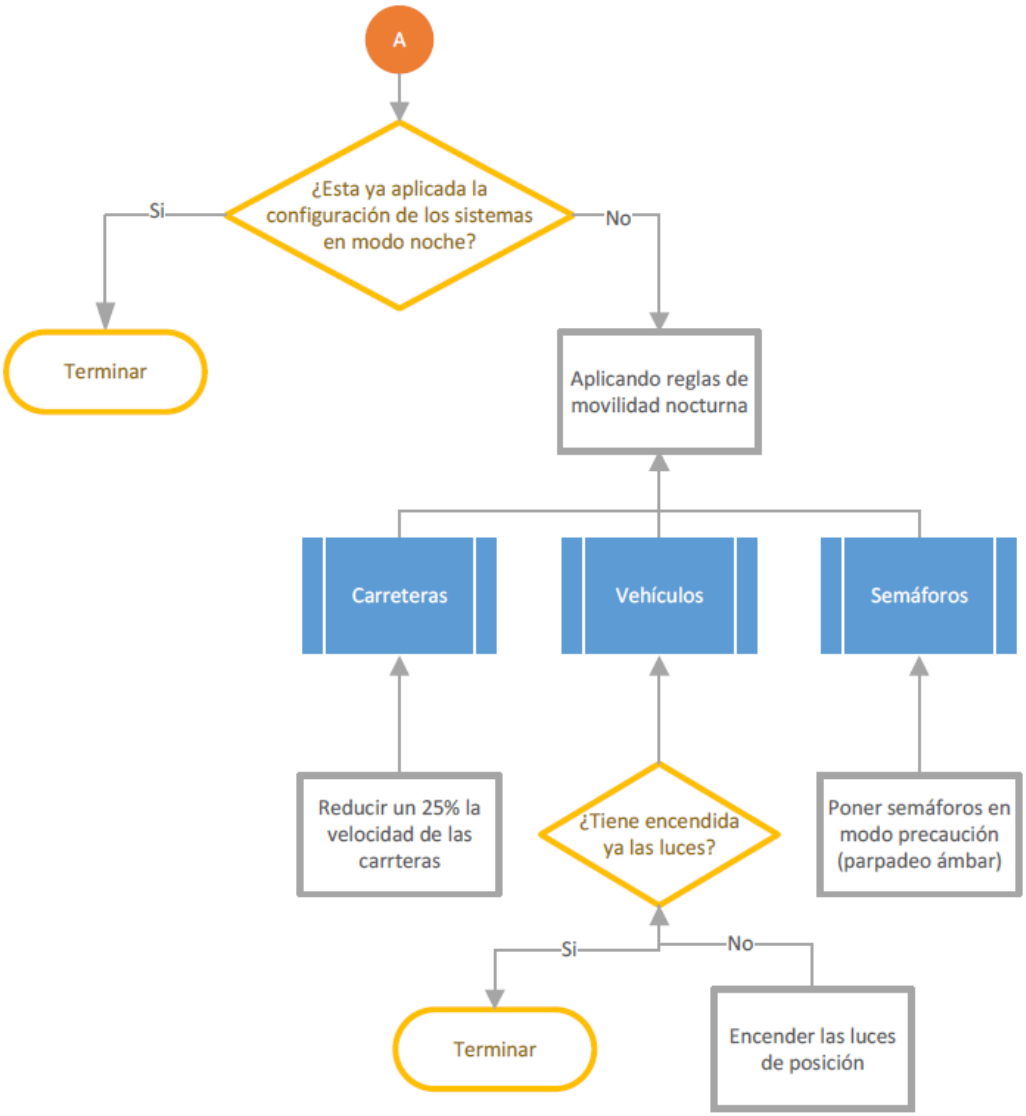

Figura 6.15: Escenario 4: Diagrama de flujo rama A

El monitor se encarga de controlar que no se aplique una configuración que sea la misma que es con la que está funcionando en ese momento en la ciudad. Como en el caso del escenario 2, este caso también afecta a los tres sistemas principales y se aplican las siguientes reglas de actuación siguientes:

- Para el sistema de carreteras inteligente (ROADS), cuando llegué la noche se reducirá en un 25 % la velocidad máxima de los segmentos de carreteras (incluso compatible con otras restricciones aplicadas ya a la velocidad por otros servicios respetándolas)
- Para el sistema de vehículos conectados (VEHICLES), cuando llegué la noche se encenderán como mínimo de manera obligatoria la luz de posición de los vehículos que no la tuvieran encendida y se le notificará al resto de usuarios.
- Para el sistema de señalización (SIGNALS), cuando llegué la noche se cambiará el modo de funcionamiento normal de los semáforos de ciclo normal (verde – amarillo – rojo) al ciclo del modo precaución (parpadeo ámbar) para evitar las esperas innecesarias en los segmentos que no haya tráfico.

El mensaje que envian los coches a los demás sistemas es:

```
{
    "msg":{
     "description" : "El vehiculo enciende las luces",
     "road-segment":"R2s2",
       "position":666,
       "vehicle-id":"3240JVM"
    },
    "type":"LIGHTS_ON"
}
```

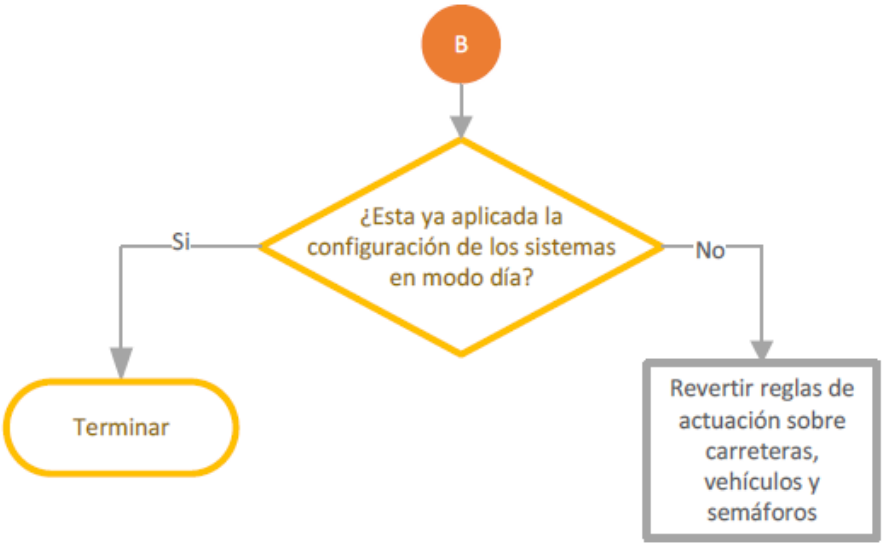

Figura 6.16: Escenario 4: Diagrama de flujo rama B

Cuando los detectores fotoeléctricos capten la luz y todos sean coherentes para llegar a la conclusión de que en la ciudad ya es de día, entonces se restablecerán todas las restricciones a los sistemas y volverá todo a su funcionamiento normal dejando las carreteras con sus límites por defecto, los vehículos ya no tendrán la obligación de tener la luz encendida y los semáforos volverán al ciclo de funcionamiento normal.

El siguiente escenario no requiere de una implementación destacable ya que utiliza las mismas técnicas y metodologías que se han utilizado en los anteriores escenarios.

CAPÍTULO 7
Conclusiones

7.1 Objetivos cumplidos

El objetivo que se ha planteado al inicio del proyecto ha cumplido enormemente con las expectativas ya que la plataforma desde la que se partía no ofrecía ningún tipo de servicio para adaptar las necesidades cambiantes del tráfico de una ciudad y por lo tanto mantenía los mismos problemas que las ciudades actuales. Una vez finalizado el proyecto se puede ver de forma clara como ahora la ciudad si que responde a estas necesidades e intenta solucionarlas de manera autónoma y auto gestionandose sin la intervención de los ciudadanos.

La plataforma IoT ha aumentado enormemente su funcionalidad, aunque aún tiene mucho margen de mejora debido al potencial de las tecnologías que utiliza por lo que en los siguientes puntos se recomendará unas futuras aplicaciones para trabajos posteriores relacionados con la materia. También se comentará la visión global, la experiencia y la valoración personal de lo que ha supuesto el desarrollo de todo el proyecto.

7.2 Valoración personal

El desarrollo de todo el proyecto ha supuesto un gran número de horas de esfuerzo y dedicación que se han visto finalmente recompensados por un gran trabajo innovador, original y con una utilidad asombrosa. Si los tipos de escenarios que se desarrollan e implementan a lo largo del proyecto se trasladaran a escenarios reales como se ha intentado hacer tratando problemas reales que afectan a las personas, sin duda alguna se vería como podría mejorar la vida de las personas en las ciudades exponencialmente.

La parte técnica es la que ha tenido la mayor complejidad sobre las demás, ya que el aprendizaje de estas nuevas y muy diversas tecnologías no estaban en su gran mayoría lo suficientemente bien documentas y en algunos momentos ha podido ser algo frustrante. Pero por esta razón y con mucha ayuda de mi tutor, en los meses que ha supuesto la realización del trabajo en ningún momento se ha dejado de aprender y ha supuesto un gran reto e inspiración para en un futuro enfocarme y continuar con otros proyectos del mismo ámbito.

En cuanto a la parte del caso de estudio y la plataforma IoT que integraba la *Smart City* es sin duda un claro ejemplo del potencial que tienen estas tecnologías aprendidas y cómo es posible plasmar estos conocimientos en algún futuro sobre ciudades reales con un cierto tipo de inteligencia ambiental. El desarrollo de los servicios adaptativos gracias a técnicas de computación autónoma, ha intentado demostrar la viabilidad de

estas tecnologías y junto con el desarrollo de los escenarios sin duda ha sido la parte más motivadora donde ya se veían los frutos del largo aprendizaje de estos meses.

Al finalizar el trabajo, se puede ver que, aunque el aprendizaje haya sido duro y costoso, se han podido desarrollar cuatro servicios adaptativos que mejoran la movilidad de las personas en las ciudades, repartidos en cuatro escenarios realistas. Con estos servicios se ha podido demostrar cómo es posible evitar en la medida de lo posible las congestiones de tráfico en las ciudades y si por algún motivo y por factores externos e imprevisibles ocurrieran, el cómo minimizar sus daños y efectos que producen a los ciudadanos. También se ha visto un escenario que pretende reducir al máximo el tiempo de respuesta cuando un vehículo tiene un accidente en una de las carreteras y si este, además, necesita de soporte médico de emergencia como agilizar los trámites de manera casi inmediata para recibirla. Pero también se enfoca a la mejora del ahorro energético y el medio ambiente con el desarrollo del último escenario, donde se demuestra que es complemente capaz adoptar configuraciones nocturnas en las ciudades que mejoren la vida sustancialmente de los ciudadanos.

7.3 Trabajos futuros y ampliaciones

El desarrollo de este proyecto puede servir de base para que otras personas amplíen todavía más las funcionalidades de la plataforma IoT que integra la *Smart City* y se hagan más escenarios que solucionen los problemas reales de una ciudad. Algunas de las ampliaciones que se proponen fueron ideas que se pensaron para incluir en el siguiente proyecto pero que finalmente tras el estudio y la dedicación de las horas invertidas se tuvieron que descartar por falta de tiempo.

Solucionar problemas de otros ámbitos de las ciudades

El proyecto se centra en mejorar la movilidad de una ciudad proporcionándole una cierta inteligencia en la materia para que se adapte a las necesidades cambiantes del tráfico, pero también se podrían abarcar otros ámbitos con problemas reales como podría ser el de la salud pública y la contaminación ambiental en tiempo real. Un ejemplo para mejorar este ámbito podría la adaptación de las políticas públicas que las regulan. En el ámbito del cambio climatológico se podría crear una red de alertas sobre peligros y adaptaciones ante inundaciones, incendios, tormentas o huracanes.

Plataforma visual que muestre el estado del simulador en tiempo real

Se podría crear una plataforma web que aprovechándose de las comunicaciones tanto directas como indirectas que proporciona la plataforma IoT pueda mostrar en tiempo real el estado de las carreteras en el simulador y sus propiedades como por ejemplo si tiene aplicada alguna restricción y su causa. También sería interesante la visualización de los vehículos a lo largo del mapa y que ruta están siguiendo en todo momento y las señalizaciones cambiantes de cada carretera.

Bibliografía

[1] Ariel, Fundación Telefónica. Smart Cities: un primer paso hacia la internet de las cosas. 2011.

[2] Arquitectura de OSGi, el sistema de modulos dinámicos para Java. Consultado en https://www.osgi.org/developer/architecture/.

[3] OSGi and Gravity Service Binder Tutorial. Consultado en http://oscar-osgi.sourceforge.net/tutorial/.

[4] David Boswarthick, O. Elloumi, and O. Hersent. M2M Communications: A Systems Approach. 2011.

[5] MQTT Documentation and Community wiki. Consultado en https://github.com/mqtt/mqtt.github.io/wiki.

[6] Introducción a JSON. Consultado en http://www.json.org/json-es.html.

[7] Fiware. Consultado en https://www.fiware.org/.

[8] Valencia Ciudad Inteligente (VLCI). Consultado en http://vlci.inndeavalencia.com/?lang=es.

[9] 5 Things to Know About MQTT – The Protocol for Internet of Things. Consultado en https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en.

[10] Smarter Cities. New cognitive approaches to long-standing challenges. Consultado en https://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/.

[11] Joan Fons i Cors. Seminario de INA: Bloque II.1 – Comunicación Asíncrona con MQTT. *Inteligencia Ambiental*, Master MUIINF, 2015.

[12] Joan Fons i Cors. Bloque II.2 – Diseño e Implementación de Servicios RESTful con RESTlet. *Inteligencia Ambiental*, Master MUIINF, 2016.

[13] Joan Fons i Cors. Bloque II.3 – Escenario de Simulación Smart Traffic. *Inteligencia Ambiental*, Master MUIINF, 2017.

[14] Joan Fons i Cors. Bloque II.4 – Infraestructura de Simulación. Inteligencia Ambiental. *Inteligencia Ambiental*, Master MUIINF, 2016.

[15] Danny Weyns, Stephan Shevtsov. Self-Adaptation: MAPE-K and Control Theory. 2014.

[16] Eric Rutten, Nicolas Marchand, Daniel Simon. Feedback Control as MAPE-K loop in Autonomic Computing. [Research Report] RR-8827, INRIA Sophia Antipolis - Méditerranée; INRIA Grenoble - Rhône-Alpes, 2015.

APÉNDICE A
Pruebas funcionales de los escenarios

En este apéndice, se va probar el funcionamiento lógico de alguno de los servicios desarrollados sobre los escenarios que se han implementado durante todo el proyecto. De esta forma, se demostrará esa capacidad adaptativa y comportamiento autónomo del que ahora disponen las ciudades en materia de movilidad.

Para ello, se van a simular emergencias médicas eventuales en alguna carretera de la ciudad y accidentes para ver si la reacción de estos servicios es la esperada.

A.1 Emergencia médica de un vehículo

En esta primera prueba funcional se va a comprobar el funcionamiento del servicio adaptativo de incidencias y emergencias médicas en vehículos inteligentes que se corresponde con el mencionado en la memoria como escenario número tres.

Muchos vehículos que podemos encontrar en la actualidad ya disponen de mecanismos y sistemas para avisar cuando nuestro vehículo ha sufrido un accidente (e incluso algunos de ellos lo hacen forma automática cuando sucede), pero las ciudades no están preparadas para reaccionar antes estos avisos y mensajes y el tiempo de respuesta es excesivamente lento.

Se va a comprobar cuál es el comportamiento de la ciudad ante estos casos mediante la simulación de que un vehículo inteligente que se encuentra en **[Road-Segment: R2s2, Road-Point: 431]** y envía un mensaje indicando que tiene una emergencia médica. Este mensaje se enviará mediante comunicación indirecta a través de colas de mensajes mqtt.

```
MQTT Topic: "es/upv/pros/iot/smartcities/traffic/PTPaterna/road/R2s2/alerts"
MQTT Message:

  {
     "msg":{
        "description":"Boton de emergencia pulsado",
        "road-segment":"R2s2",
        "position":431,
        "vehicle-id":"4777DBH",
        "assistance":"yes",
        "status":"Active"
     },
     "type":"WARNING_BUTTON"
  }
```

Una vez es enviado, los diferentes sistemas que componen la plataforma IoT lo recibirán al instante, pero no serán capaces de entender el contenido del mensaje. Pero aquí es donde entra el servicio adaptativo que se ha desarrollado, el cual al estar integrado en la plataforma también recibe el mensaje como los demás y además es capaz de entenderlo.

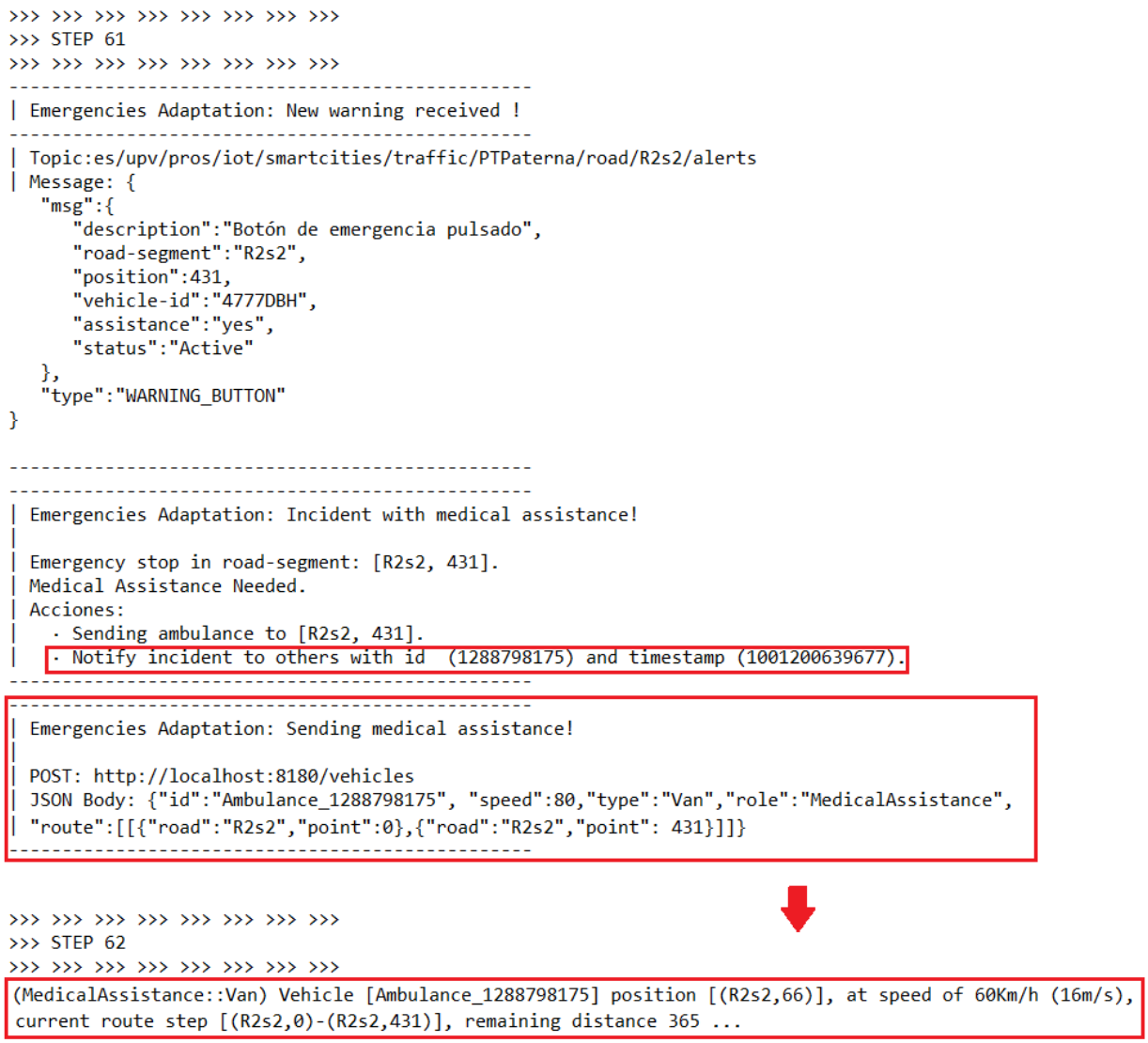

Figura A.1: Reacción del servicio adaptativo de emergencias

La figura A.1 es un extracto de la consola de eclipse donde se ve que el servicio de emergencias ha capturado el mensaje y tras analizarlo ha decidido aplicar 2 reglas de actuación. Las reglas son:

- Publicar un accidente con la información que se ha recibido por parte del vehículo.

  El mensaje recibido se retransmite y adapta para entenderse como un accidente en una carretera. Como se puede ver en la figura A.2 algunos de los cambios más destacados son:

  - El lugar del accidente ha sido en el punto 431.
  - El tramo del accidente comienza 100 metros antes, en el punto 331.
  - El tramo del accidente acaba 100 metros después, en el punto 531.
  - Se le asigna al accidente el identificador 1288798175.
  - La descripción del accidente indica el vehículo que ha tenido el incidente.

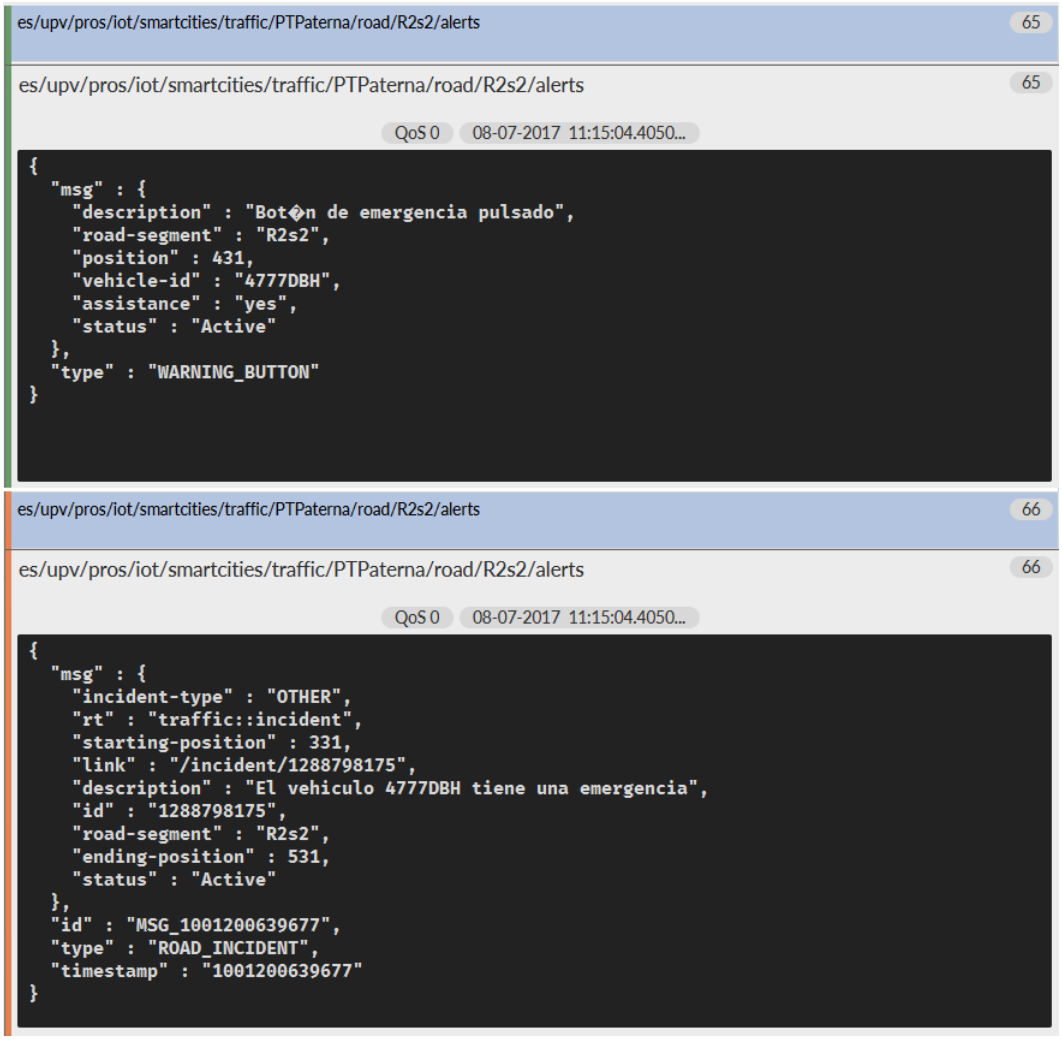

Figura A.2: Retransmisión del mensaje con cambios

Seguidamente, si se consulta el estado de la carretera donde ha ocurrido el accidente se puede comprobar que ya lo ha recibido el sistema de carreteras y procesado.

```
GET Request: http://localhost:8180/segment/R2s2
```


```
{
    "rt": "road-segment",
    "end-kp": 750,
    "code": "R2s2",
    "density": 0,
    "link": "/segment/R2s2",
    "length": 450,
    "max-speed": 60,
    "road-segment": "R2s2",
    "start-kp": 300,
    "capacity": 45,
    "current-max-speed": 60,
    "road": "R2",
    "incidents": [
        {
            "incident-type": "OTHER",
            "rt": "traffic::incident",
            "starting-position": 331,
            "link": "/incident/1288798175",
            "description": "El vehiculo 4777DBH tiene una emergencia",
            "id": "1288798175",
            "road-segment": "R2s2",
            "ending-position": 531,
            "status": "Active"
        }
    ],
    "num-vehicles": 0,
    "status": "Free_Flow"
}
```


Figura A.3: Segmento de la carretera con el accidente notificado

- Enviar una ambulancia al lugar del accidente para dar soporte médico a la solicitud.

  Si se hace una petición REST al sistema de vehículos conectados se podrá comprobar como hay una ambulancia que esta de camino al accidente. Una vez llegué a su destino, el accidente se podrá resolver de manera satisfactoria.

```
GET Request: http://localhost:8180/vehicles
```


```
[
    {
        "navigator": {
            "off-road": false,
            "route": "[(R2s2,0)-(R2s2,431)]",
            "current-position": "(R2s2,66)",
            "destination": "(R2s2,431)",
            "remaining-distance": 365,
            "id": "Ambulance_1288798175_navigator",
            "status": "ROUTING"
        },
        "id": "Ambulance_1288798175",
        "characterization": {
            "role": "MedicalAssistance",
            "type": "Van"
        },
        "cruiser-speed": 80,
        "speed": 60,
        "status": "Moving"
    }
]
```


Figura A.4: Ambulancia creada por el servicio con destino el accidente

A.2 Vehículo inteligente atravesando un accidente

En esta segunda prueba funcional, se va a comprobar una parte del funcionamiento del servicio adaptativo que controla la aparición de accidentes en las carreteras de la ciudad. Este servicio se corresponde con el mencionado en la memoria como escenario número dos y afecta directamente al sistema de vehículos conectados.

Según lo que se ha definido en la memoria, el servicio adaptativo de la ciudad tendría que notificar al coche para que redujera al 50 % su velocidad antes de que llegará al punto del accidente, mientras lo esté atravesando y un poco después de que pase el tramo.

Además de que el vehículo tenga que cumplir esta normativa que se le impone, tendrá la obligación de informar a los demás coches que ha reducido su velocidad porque está atravesando un accidente e informar de su posición.

Se va a comprobar que el comportamiento de la ciudad ante estos casos es el esperado según el párrafo anterior, con un accidente que se encuentra en **[Road-Segment: R2s2, Road-StartPoint: 331, Road-EndPoint: 431]**. Este mismo accidente podría ser compatible con el que se ha provocado en la prueba anterior, ya que como se puede ver, los dos servicios son complementarios y pueden funcionan a la vez.

La estructura del accidente que se toma de ejemplo en la prueba funcional del servicio, se puede ver en el siguiente mensaje:

```
{
    "msg":{
        "incident-type":"OTHER",
        "rt":"traffic::incident",
        "starting-position":331,
        "link":"/incident/1152445151",
        "description":"Accidente de Trafico con colision multiple",
        "id":"1152445151",
        "road-segment":"R2s2",
        "ending-position":431,
        "status":"Active"
    },
    "id":"MSG_1495888770000",
    "type":"ROAD_INCIDENT",
    "timestamp":1495888770000
}
```

A continuación, la figura A.5 muestra como el servicio adaptativo ha capturado el accidente y ha aplicado unas reglas que consisten en activar más sensores para obtener más información.

El comportamiento de este servicio es normal de momento, cuando ocurre un accidente se necesitan tener controlados varios sistemas a la vez para poder mejorar la movilidad.

Los sensores que se han activado por el servicio y que a su vez reportarán a sus monitores son:

- El sensor de congestión de control y su monitor que se encargará de aplicar reglas para cerrar o abrir el segmento si aumenta la densidad de vehículos en la carretera.

- El sensor de señalización que se encargará de buscar si hay algún semáforo cerca para ponerlo en modo emergencia.

- El detector de coches cercas y su monitor que se encargará de aplicar las reglas que se explicarán más adelante, ya que es el que se quiere probar su funcionamiento.

```
>>> >>> >>> >>> >>> >>> >>> >>>
>>> STEP 6
>>> >>> >>> >>> >>> >>> >>> >>>
-------------------------------------------------
| Accidents Adaptation: New accident info received !
-------------------------------------------------
| Topic:es/upv/pros/iot/smartcities/traffic/PTPaterna/road/R2s2/alerts
| Message: {
  "msg" : {
    "incident-type" : "OTHER",
    "rt" : "traffic::incident",
    "starting-position" : 331,
    "link" : "/incident/1152445151",
    "description" : "Accidente de Tráfico con colision multiple",
    "id" : "1152445151",
    "road-segment" : "R2s2",
    "ending-position" : 431,
    "status" : "Active"
  },
  "id" : "MSG_1495888770000",
  "type" : "ROAD_INCIDENT",
  "timestamp" : 1495888770000
}
-------------------------------------------------
-------------------------------------------------
| Accidents Adaptation: New Accident in Road-Segment
|
| Status: Active
| Segment R2s2 and position [331,431]
| Actions:
|    · Activating sensor for congestion control.
|    · Activating sensor to detect cars near the accident.
|    · Publishing emergency signals in the segment.
-------------------------------------------------
```

Figura A.5: Reacción del servicio adaptativo de accidentes

El siguiente paso, es provocar la situación de que un vehículo atraviese ese accidente, por lo que habrá que crear el vehículo y posteriormente asignarle una ruta que atraviese el accidente. Seguidamente, se puede ver el tipo de mensaje que se le envía al sistema de vehículos.

```
POST Request: http://localhost:8180/vehicles
  "Content-Type":"application/json"
  "Accept":"application/json"

  {
     "id":"1000JBB",
     "speed":80,
     "type":"Bus",
     "role":"PublicTransport",
     "route":[
        [
           {
              "road":"R2s2",
              "point":0
           },
           {
              "road":"R2s2",
              "point":750
           }
        ]
     ]
  }
```

Como consecuencia, se pueden ver una serie de figuras donde se muestra que ahora el sensor de vehículos cercanos está detectando que acabamos de crear un vehículo y está atravesando el accidente, por lo que le obliga a reducir su velocidad un 50 % y que notifique a los demás su situación de que tiene encendida las luces de emergencia.

```
>>> >>> >>> >>> >>> >>> >>> >>>
>>> STEP 34
>>> >>> >>> >>> >>> >>> >>> >>>
(SimulationElement vehicle) # VEHICLES: 1
(PublicTransport::Bus) Vehicle [1000JBB] position [(R2s2,266)], at speed of 60Km/h (16m/s),
current route step [(R2s2,0)-(R2s2,750)], remaining distance 484 ...
>>> >>> >>> >>> >>> >>> >>> >>>
>>> STEP 35
>>> >>> >>> >>> >>> >>> >>> >>>
(SimulationElement vehicle) # VEHICLES: 1
(PublicTransport::Bus) Vehicle [1000JBB] position [(R2s2,316)], at speed of 30Km/h (8m/s),
current route step [(R2s2,0)-(R2s2,750)], remaining distance 434 ...
------------------------------------------------------------
| Accidents Adaptation: Car near accident sensor
|
| Car with registration (1000JBB) is going through an accident in [R2s2,316]
| Car Obligations:
|   · Publishing emergency warnings to others :    YES
|   · Reduce speed to half of the road's maximum:  YES
------------------------------------------------------------
------------------------------------------------
| Accidents Adaptation: Car 4 Warning Lights | Restricted Speed !
------------------------------------------------
| Topic:es/upv/pros/iot/smartcities/traffic/PTPaterna/road/R2s2/alerts
| Message: {"msg":{"restricted-speed":30,"description":"Atravesando accidente cercano en carretera",
| "position":316,"vehicle-id":"1000JBB"},"type":"WARNING_LIGHTS"}
------------------------------------------------
```

Figura A.6: Vehículo reduciendo su velocidad y encendido luces de emergencia

Se comprueba a través de herramientas externas, que efectivamente está notificando su situación y que ya, por último, cuando sobrepasa el tramo del accidente vuelve a restablecer su velocidad normal y apaga las luces de emergencia.

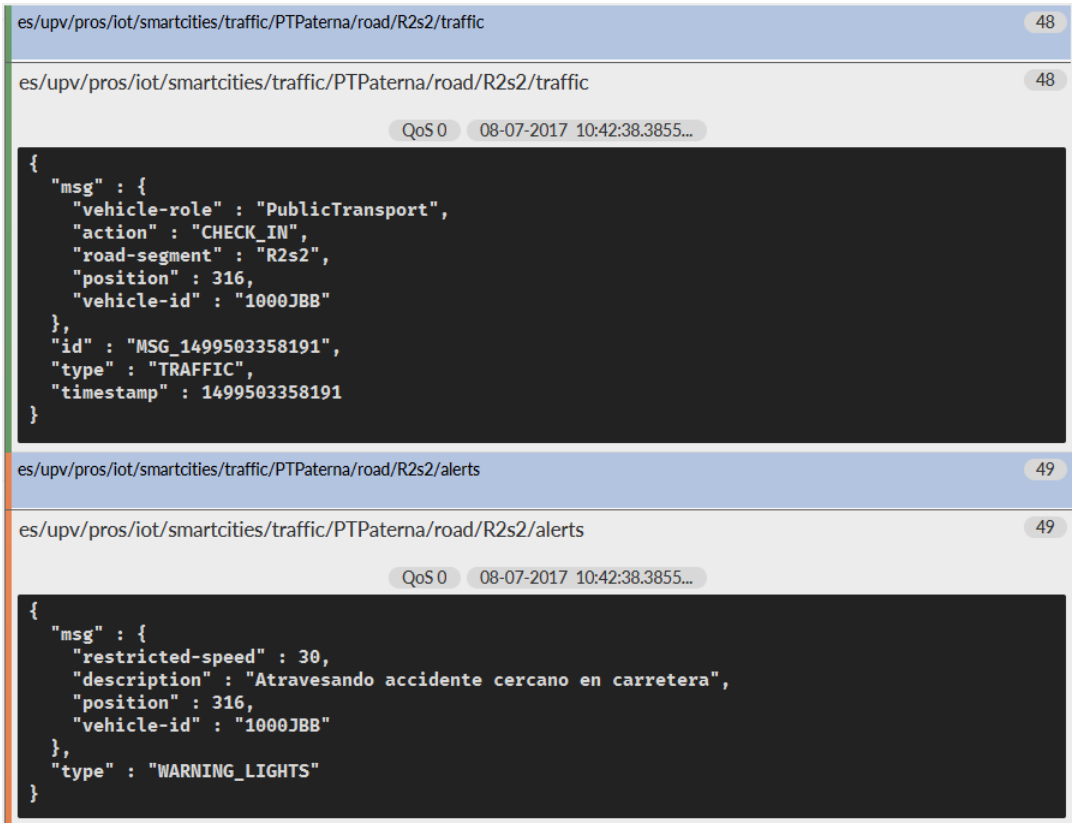

Figura A.7: Vehículo retrasmitiendo su situación a los demás sistemas

```
>>> >>> >>> >>> >>> >>> >>> >>>
>>> STEP 41
>>> >>> >>> >>> >>> >>> >>> >>>
(SimulationElement vehicle) # VEHICLES: 1
(PublicTransport::Bus) Vehicle [1000JBB] position [(R2s2,466)], at speed of 30Km/h (8m/s),
current route step [(R2s2,0)-(R2s2,750)], remaining distance 284 ...
-------------------------------------------------------------
| Accidents Adaptation: Car near accident sensor
|
| Car with registration (1000JBB) is going through an accident in [R2s2,466]
| Car Obligations:
|  · Publishing emergency warnings to others :    YES
|  · Reduce speed to half of the road's maximum:  YES
-------------------------------------------------------------
-------------------------------------------------
| Accidents Adaptation: Car 4 Warning Lights | Restricted Speed !
-------------------------------------------------
| Topic:es/upv/pros/iot/smartcities/traffic/PTPaterna/road/R2s2/alerts
| Message: {"msg":{"restricted-speed":30,"description":"Atravesando accidente cercano en carretera",
| "position":466,"vehicle-id":"1000JBB"},"type":"WARNING_LIGHTS"}
-------------------------------------------------

>>> >>> >>> >>> >>> >>> >>> >>>
>>> STEP 42
>>> >>> >>> >>> >>> >>> >>> >>>
(SimulationElement vehicle) # VEHICLES: 1
(PublicTransport::Bus) Vehicle [1000JBB] position [(R2s2,491)], at speed of 60Km/h (16m/s),
current route step [(R2s2,0)-(R2s2,750)], remaining distance 259 ...
```

Figura A.8: Vehículo restableciendo su velocidad y apagando las luces de emergencia