



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica  
Universitat Politècnica de València

## SMARTROAD

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor:** García Ferrando, Germán Adrián

**Tutor:** [Nombre del tutor/es]

[Curso Académico]



Resumen

Resum

Abstract



## TABLA DE CONTENIDO

---

1	Introducción .....	8
1.1	Internet of Things (IoT) .....	8
1.2	SmartCity.....	9
1.3	SmartRoad.....	10
1.4	Arduino .....	10
1.5	Raspberry Pi .....	10
2	Caso de estudio.....	12
2.1	Análisis del estado actual de los servicios de emergencia.....	12
2.2	Motivación .....	15
2.3	Objetivo.....	16
2.4	Visión general de la solución propuesta .....	16
3	Introducción a la simulación .....	19
3.1	Escenario.....	19
3.2	Elementos de la simulación, y sus roles.....	20
3.2.1	SmartCity.....	20
3.2.2	SmartRoad.....	20
3.2.3	Segmento .....	21
3.2.4	Panel.....	22
3.2.5	SmartCar .....	23
3.2.6	Ambulancia .....	23
3.3	Diagrama de clases .....	24
4	Comunicación entre Elementos .....	27
4.1	Formato de los mensajes .....	27
4.1.1	¿Por qué JSON? .....	28
4.2	Identificadores .....	29
4.3	Códigos.....	30
4.3.1	Descripción de cada código .....	30
4.3.2	Atributos de los códigos.....	31
4.4	MQTT & rest.....	34
4.4.1	¿Por qué MQTT? .....	35

4.4.2	¿Por qué REST? .....	35
4.5	Comunicación Mqtt.....	35
4.6	Comunicación rest .....	36
5	Simulación paso a paso .....	38
5.1	Carga del escenario .....	38
5.2	Señal S.O.S.....	38
5.3	Planificación del plan para atender la emergencia.....	39
5.4	Gestión de segmentos con ambulancia en ruta .....	40
5.5	La ambulancia llega a su destino .....	42
6	Referencias.....	44



# 1 INTRODUCCIÓN

---

Este proyecto se sitúa dentro de un entorno donde el Internet de las Cosas (IoT) nos brinda la oportunidad de conectar miles de dispositivos para obtener cantidades de información inimaginables hasta el momento. Con el problema de recopilar información solucionado, el siguiente paso es diseñar sistemas capaces de armonizar dicha información y, en consecuencia, actuar con la mayor precisión posible.

Actualmente, uno de los escenarios de estudio más conocido es el de las SmartCity, ciudades capaces de auto gestionarse por sí solas. La cantidad de eventos que suceden en una gran ciudad diariamente es enorme, y en consecuencia la casuística del sistema que ha de gestionar dicha ciudad, lo es también.

El objetivo de este proyecto es analizar uno de esos casos, darle una solución factible, y a continuación demostrar su viabilidad mediante una simulación, donde estarán representados los elementos fundamentales (relevantes para el caso estudiado) de una SmartCity.

## 1.1 INTERNET OF THINGS (IoT)

Internet of Things nace de la idea de conectar a internet los objetos cotidianos que utilizamos día a día, tostadoras, vehículos, casas... y conectarnos a internet para poder analizar todos los datos que son capaces de recopilar. El concepto IoT se propuso en 1999 por Kevin Ashton, el cual apunta a que a finales de los 90, la mayoría de la información almacenada en internet había sido creada por seres humanos, a base de teclear. Él propone un giro, en el cual se pueda dotar a los computadores de las herramientas necesarias para que generen ellos la información de forma independiente (Ashton, 2009).

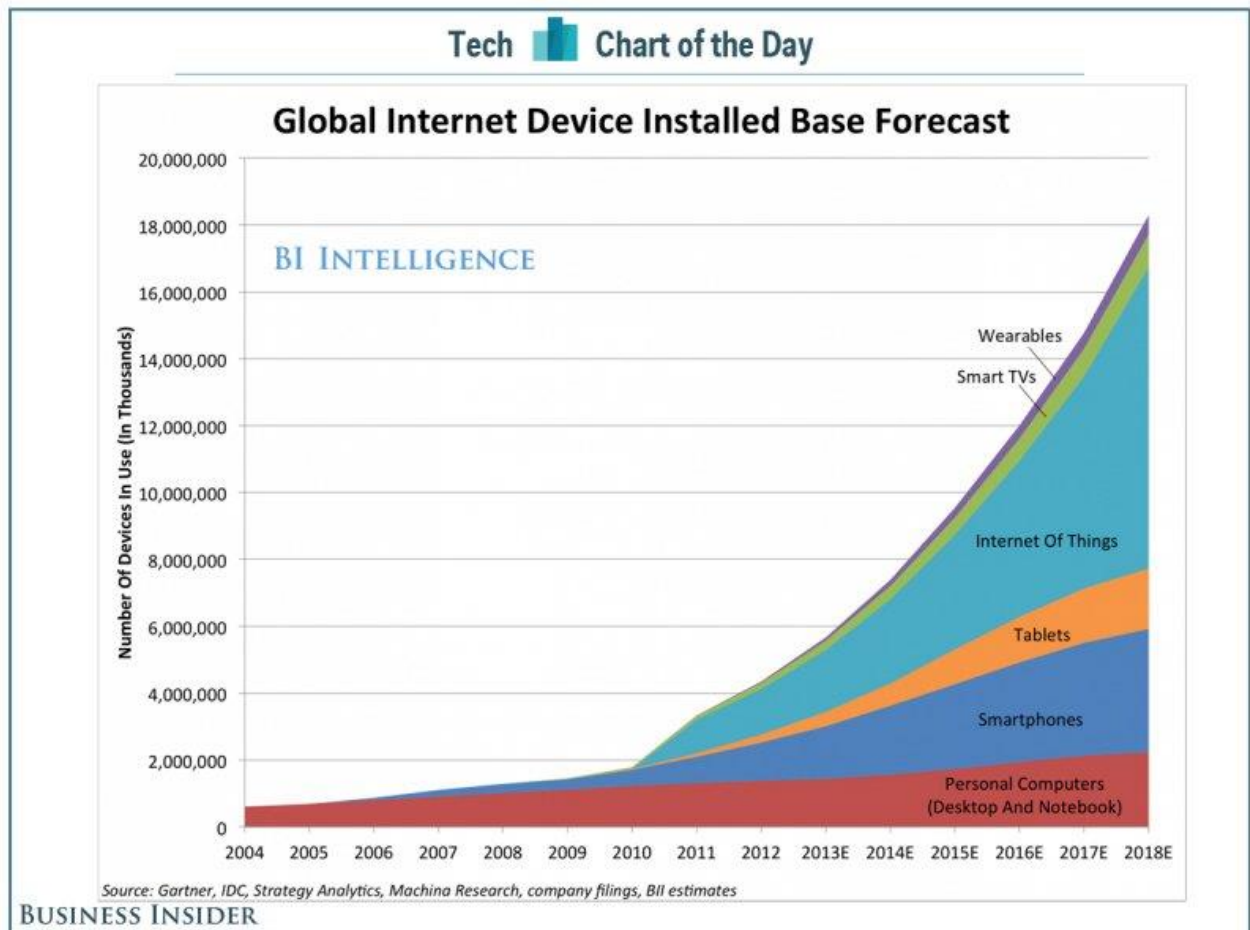
El concepto IoT ya se ha aplicado a varios campos con éxito. En el campo de la medicina, ya se están utilizando pequeños sensores para monitorizar las constantes vitales de un paciente esté donde esté, o simplemente para recordar al paciente cuando es hora de tomarse la medicación. También se han creado “SmartCar”, coches capaces de monitorizar su estado (motor, neumáticos, autonomía...) lo cual ayuda al usuario ya que tiene más información a sobre su vehículo y también ayuda a la industria, ya que puede recopilar la información generada por todos sus vehículos para hacer estudios sobre estos.

La tecnología IoT es algo que no se puede ignorar, está presente en nuestro día a día y cada vez lo va a estar más. En este proyecto IoT representa el punto de unión entre todos los elementos que componen la “SmartCity”. Cada uno de los elementos que interviene, está conectado con el resto de dispositivos siguiendo una arquitectura IoT, lo cual nos permite extrapolar esta



## 1. INTRODUCCIÓN

simulación virtual a una posible simulación real utilizando dispositivos embebidos como Arduino o Raspberry Pi.



*Ilustración 1: Predicción de la tecnología IoT por Business Insider*

### 1.2 SMARTCITY

El concepto de SmartCity aún no está definido con claridad, pero sí que podemos encontrar un patrón entre todos los proyectos que en la actualidad trabajan en este campo. Todos persiguen, a través de la tecnología: **aumentar la calidad de vida de los ciudadanos** mientras se crea una ciudad autosuficiente y sostenible. Por ejemplo, en el proyecto BCN Smart City (BCN, 2016) se habla de utilizar las nuevas tecnologías para hacer una gestión eficiente de los recursos y servicios de la ciudad; en el proyecto *europesmartcities 4.0* dirigido por Vienna University of Technology (WIEN, 2015) se habla de combinar la competitividad de una ciudad con un desarrollo urbano

sostenible centrado en las ciudades de tamaño medio – bajo; también podría nombrarse el PLEEC (Planning for Energy Efficient Cities) que apunta a utilizar la tecnología para crear SmartCities autosuficientes energéticamente, y así un largo etcétera.

El resumen que podemos extraer de este conjunto de proyectos, es que el modo en el que las ciudades han funcionado hasta ahora está cambiando, existe la oportunidad de mejorar el modelo de ciudad que conocemos actualmente. En un futuro, las ciudades serán mucho más interactivas con los ciudadanos, serán ciudades autosuficientes, capaces de auto gestionarse sin la constante supervisión de los seres humanos etc. En definitiva, un lugar mejor donde vivir.

### 1.3 SMARTROAD

Si el concepto de las SmartCity aún no está claro, el de la SmartRoad no es una excepción. A día de hoy hay pocos proyectos que se dediquen exclusivamente al campo de las SmartRoad. Aunque sí que podemos destacar el trabajo que se está realizando en el estado de Virginia, USA. Allí se está construyendo la SmartRoad conocida como “*Virginia Smart Road*”, se trata de una carretera de 3.5km de longitud en la cual se pretende testear varias de las nuevas tecnologías que se están introduciendo en el escenario de las SmartCity. Un ejemplo de estas pruebas fue el cuándo los estudiantes de la Universidad de Washington instalaron en la Virginia Smart Road un pavimento el cual también actuaba como panel solar (UW, 2016).

De todas formas, ya sea aumentando la interacción con los SmartCar que circulen por la carretera, monitorizando el estado de la carretera, haciendo que genere su propia energía... Podemos concluir en que el objetivo básico de las SmartRoad sigue siendo el mismo que en las SmartCities, y es **aumentar la calidad de vida los usuarios**.

### 1.4 ARDUINO

### 1.5 RASPBERRY PI

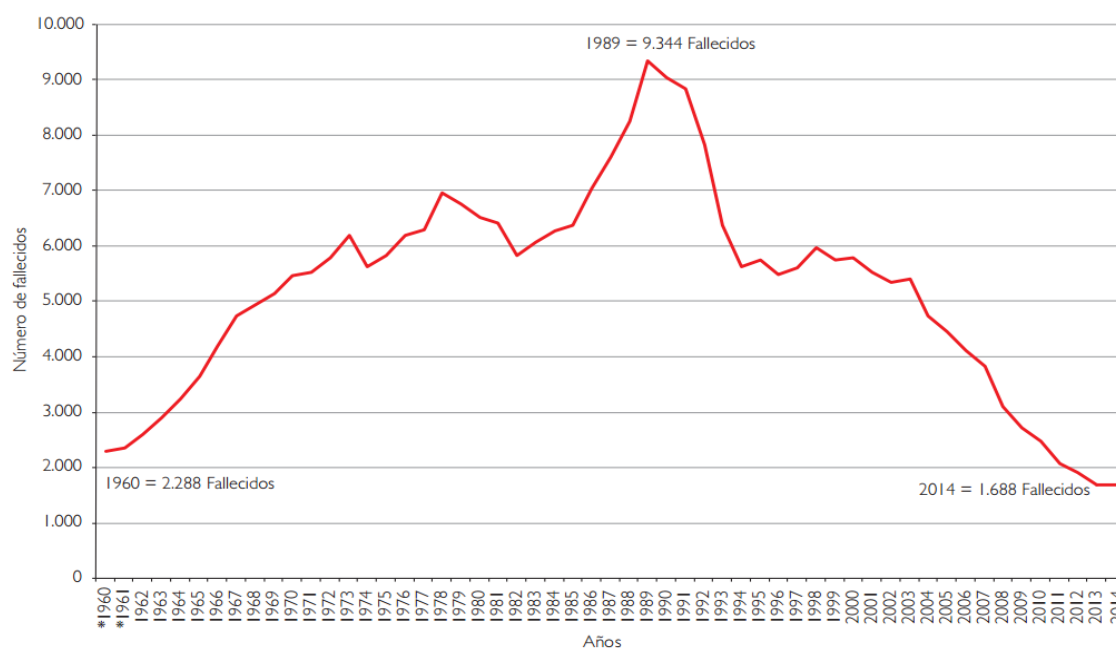
## 2. CASO DE ESTUDIO

## 2 CASO DE ESTUDIO

### 2.1 ANÁLISIS DEL ESTADO ACTUAL DE LOS SERVICIOS DE EMERGENCIA

Con el fin de justificar la necesidad de un mejor método para tratar emergencias dentro de las SmartCity se ha realizado un pequeño estudio sobre el estado actual de los servicios de emergencia en España. Toda la información aquí mostrada ha sido extraída de las páginas oficiales del Ministerio del Interior, en caso contrario se detallará la fuente.

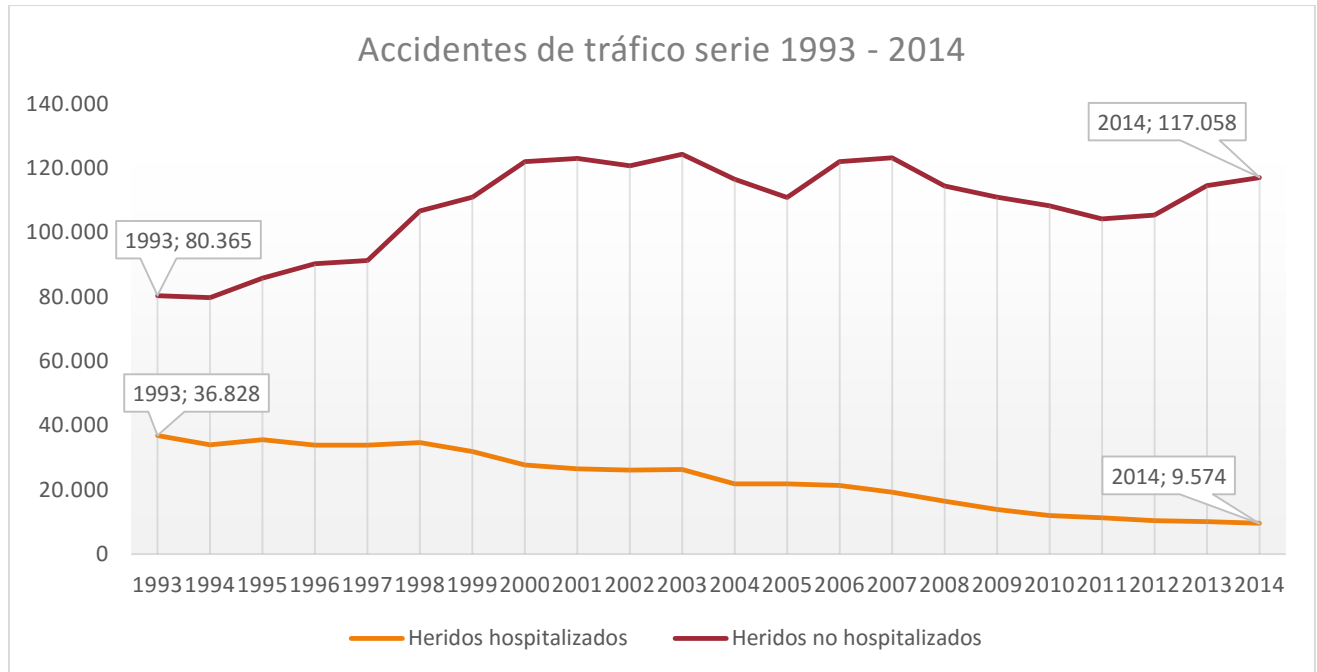
Ante todo, remarcar el buen trabajo que se ha ido haciendo desde un tiempo atrás por los servicios de emergencia del estado español, los fallecidos en accidentes de tráfico han ido descendiendo desde 1989 a un ritmo considerable, aunque como se muestra en la Ilustración 2 el número de accidentes de tráfico que requieren la hospitalización del accidentado es elevada.



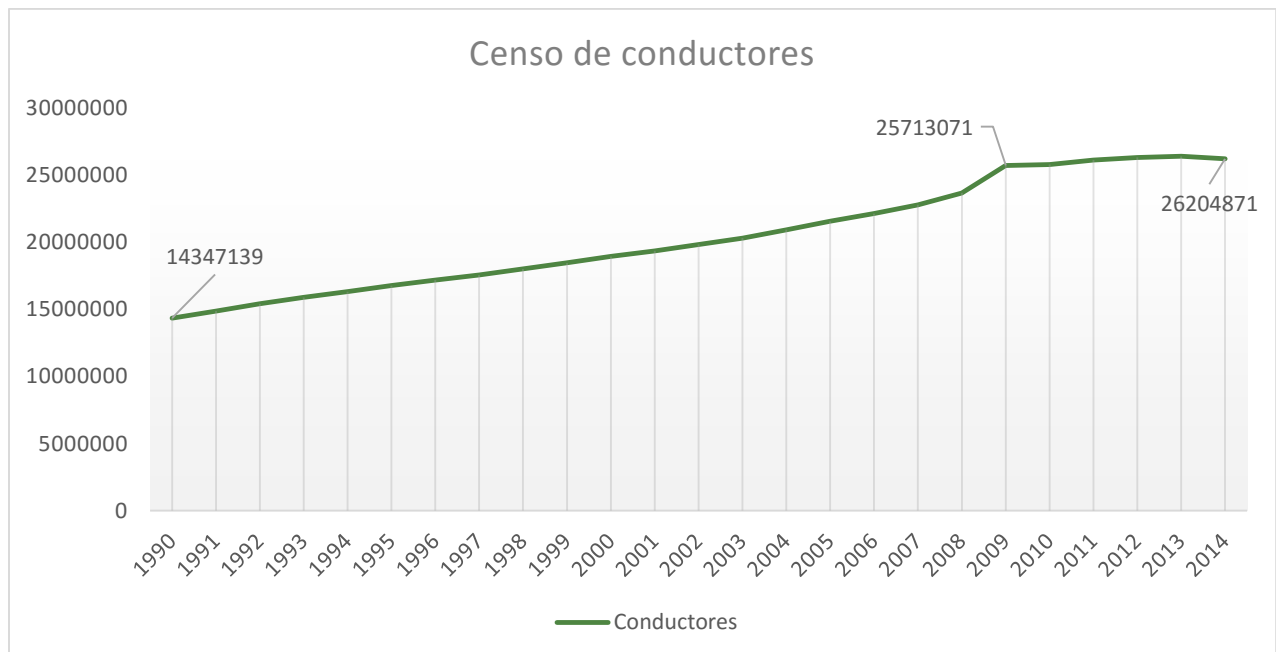
*Ilustración 2: Fallecidos por accidente de tráfico 1960 - 2014*

Un dato a destacar es que desde el año 2011 el número de accidentes ha aumentado, esto podría deberse a un aumento en la cantidad de vehículos en las carreteras, pero como se puede apreciar en el Censo de conductores, la cantidad de conductores se ha estabilizado desde el año 2009.

## 2. CASO DE ESTUDIO



Una de las conclusiones que se podrían extraer de estos datos, es que cada vez hay más usuarios utilizando las carreteras españolas, por lo que invertir tiempo en mejorar los servicios encargados de mejorar la seguridad de los ciudadanos es una causa más que justificada.



Las ambulancias no son los únicos servicios de los cuales dispone una SmartCity para atender las distintas emergencias; a continuación, se hará un breve análisis del estado actual de los bomberos y la policía en el estado español, servicios a los cuales también se podría aplicar la idea general de este proyecto.



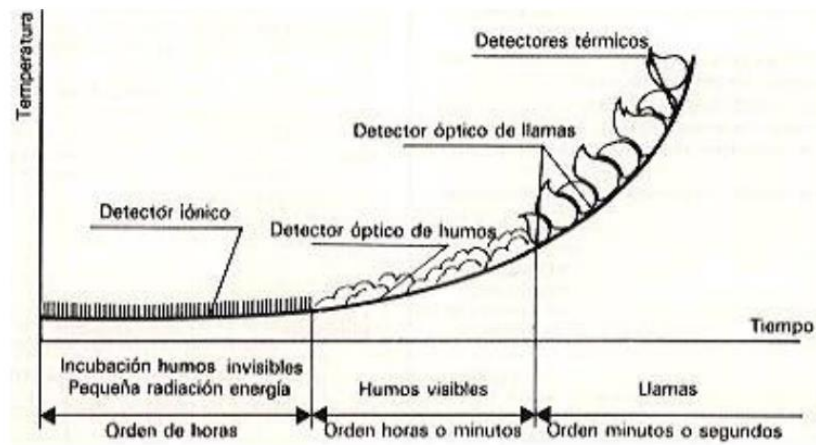
A diferencia de los datos mostrados en la sección de tráfico, el número de incendios no atiende a una función creciente o decreciente, sino que más bien es algo aleatorio. Lo que sí que se puede destacar de estos datos, es que la relación Número de contactos / Número de incendios es bastante elevada, es decir, cuando alguien contacta con el servicio de emergencias para alertar de un posible incendio la probabilidad de que el incendio exista realmente es del 72.32%.

El servicio de bomberos lleva tiempo trabajando en la instalación de sensores de varios tipos para ayudar a la detección de incendios, como se aprecia en la Ilustración 2 hay varios sensores distribuidos por los distintos parques nacionales, cada uno con un distinto tiempo de reacción.

Por desgracia, la detección automática de incendios no es sencilla, por lo que recibir la información a tiempo se convierte en uno de los principales problemas a la hora de tratar estas emergencias. Al igual que en las SmartCity, la tecnología en el contexto de IoT nos trae muchos avances aplicables a este escenario, aunque esto ya no forma parte del alcance del proyecto. En cambio, la propuesta de “generar una respuesta automática una vez se ha detectado la emergencia” es perfectamente aplicable al caso de los incendios, aunque estaría más enfocada a la detección de incendios dentro de una ciudad.

## 2. CASO DE ESTUDIO

*Ejemplo: edificios conectados a la SmartCity que mandan señales de emergencia detectadas por sus sensores de humo.*



*Ilustración 3: Fase de actuación de los sensores*

**TODO: AMPLIAR ESTUDIO O HACER UN PÁRRAFO QUE CIERRE EL TOPIC**

## 2.2 MOTIVACIÓN

Ambulancias en mitad de una emergencia inmóviles en la carretera por atascos; accidentes que no se detectan hasta que un tercero llama al 112; vidas que se podrían salvar, pero no fuimos capaces de hacerlo... Cada día mueren en el mundo sobre 3.400 personas en accidentes de tráfico (OMS, 2016), en España en el año 2014 murieron 340 personas (DGT, 2014), gracias al trabajo de miles de personas el número es cada vez menor, pero, ¿podríamos haberlo hecho mejor?

Desde mi humilde ignorancia creo que algunas situaciones se podrían haber gestionado mejor si se hubiese contado antes con la información necesaria, por lo que mi granito de arena es diseñar una propuesta la cual nos permitiría obtener la información con mayor rapidez y ser capaces de actuar de una forma automática. Esta propuesta se basa en una tecnología ampliamente conocida, y que ya se ha usado con éxito en otros contextos, hablo de IoT junto a un software capaz de gestionar la información generada por los distintos dispositivos.

Creo que la propuesta realizada en este TFG es un boceto más, entre muchos otros, que demuestra la existencia de un camino donde la tecnología nos brinda un mundo mejor; una tecnología que ya se está utilizando en varios proyectos, una tecnología que es viable económicamente y que podríamos empezar a utilizar hoy mismo.

La motivación principal de este proyecto no es superar una última prueba para obtener un título, sino ser capaz de utilizar los conocimientos adquiridos durante mis cuatro años de estudio en el Grado de Ingeniería Informática en pro de la causa más noble que conozco, salvar vidas.

## 2.3 OBJETIVO

El objetivo principal de este proyecto es **reducir el tiempo que un accidente de tráfico pasa sin asistencia sanitaria**.

¿Cómo se va a reducir el tiempo? Bueno, si lo único que necesitamos es llegar antes a un sitio, todos conocemos la solución, solo necesitamos “salir antes” e “ir más rápido”. En base a estos dos conceptos, el proyecto propone cambios en tres puntos clave:

- **Obtener información con mayor rapidez.** No se puede atender un siniestro del cual se desconoce su existencia. La primera propuesta del TFG es dotar a los SmartCar la capacidad de mandar una señal S.O.S cuando detecten que el coche ha sufrido un accidente, o cuando el conductor considere que está en una situación de emergencia.
- **Ser capaces de responder automáticamente.** Una máquina siempre procesará la información con mayor rapidez que un ser humano. Esta solución propone crear un software capaz de actuar en caso de emergencia, con cierta independencia de un supervisor humano.
- **Si no existe un camino adecuado, hay que crearlo.** No se puede tolerar que un vehículo de asistencia sanitaria quede inmovilizado por el tráfico en mitad de una emergencia. Para solucionar esto, se propone que el software tenga la capacidad de gestionar la circulación del tráfico con el fin de evitar el mayor número de obstáculos en ruta.

## 2.4 VISIÓN GENERAL DE LA SOLUCIÓN PROPUESTA

La solución propuesta se puede dividir en tres partes, cada una de estas partes hace referencia a los puntos comentados en el apartado de Objetivo.

**La información del siniestro llega al sistema.** Una vez el SmartCar ha detectado que el vehículo ha sufrido un siniestro, o el conductor detecte que está en una situación de emergencia, se mandará una señal S.O.S la cual será recibida por la infraestructura de la carretera.



## 2. CASO DE ESTUDIO

**Respuesta automática.** Una vez la señal S.O.S ha llegado a la infraestructura del sistema, se inicia un proceso automático el cual dará una respuesta a dicha señal. Primero la carretera debe comunicar a la SmartCity la existencia de una nueva emergencia, una vez hecho esto mandará una señal de “Check S.O.S” al vehículo siniestrado, para que, si este puede recibirla sea consciente de que su emergencia está siendo atendida.

A continuación, la SmartCity debe decidir dos cosas, primero: **¿qué ambulancia es la más indicada para asistir la emergencia?** El criterio de evaluación, a priori, define como mejor ambulancia aquella que se encuentre más próxima a la ubicación del siniestro; y segundo: **¿cuál es la ruta óptima desde la ubicación de la ambulancia hasta el siniestro?** Este es un cálculo complejo, ya que además de la distancia, se han de contemplar factores como el tráfico, estado de la carretera, opciones para abrir o cerrar carriles etc...Una vez decidida qué ambulancia asistirá a la emergencia y qué ruta seguirá, la SmartCity mandará una misión a la ambulancia elegida, la cual debe corroborar que ha recibido la misión y que está en proceso.

Por último, y con el objetivo de que la ambulancia encuentre el **mínimo número de obstáculos en su ruta**, se entra en una etapa en la cual la ambulancia enviará constantemente su ubicación a la SmartCity; con esta información, la SmartCity decidirá si es necesario cerrar o abrir algunos carriles en la ruta de la ambulancia. En caso de ser necesario, se pondrá en contacto con las carreteras, las cuales tienen la autoridad de cerrar carriles al tráfico estándar.

Una vez la ambulancia haya llegado a la ubicación pueden darse dos escenarios:

- El profesional en cuestión considera que necesita más recursos para atender el siniestro, por lo que mandará una nueva petición a la SmartCity con un informe de la situación.
- La emergencia puede ser atendida por el profesional, el cual manda un “check” a la city como que la emergencia ha sido atendida.

Antes de cerrar la emergencia por completo, la SmartCity manda un mensaje a la carretera, la cual comunicará al coche que mandó la señal S.O.S que su emergencia ha sido atendida.



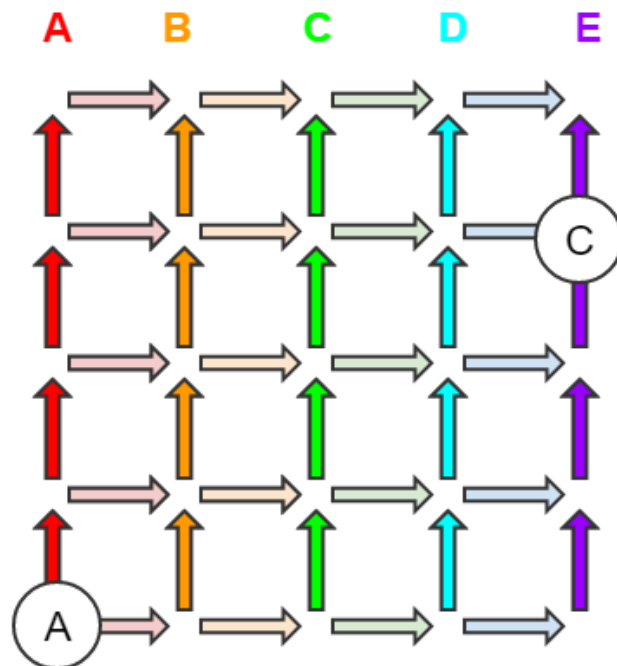
### 3 INTRODUCCIÓN A LA SIMULACIÓN

Antes de poder empezar a explicar paso a paso el funcionamiento de la solución, se ha de explicar el entorno en el que esta se desarrolla. Para poder visualizar la solución se ha creado una simulación en la cual transcurren los eventos, como en toda simulación, el entorno representa una simplificación de la realidad, con el fin de no caer en los detalles y conseguir el objetivo principal que es: demostrar la viabilidad de la solución propuesta.

A continuación, se explicarán los elementos fundamentales que conforman el escenario donde se desarrolla la simulación.

#### 3.1 ESCENARIO

La ciudad ficticia donde se desarrollan los hechos es Atlantis, y su mapa es siguiente:



*Ilustración 4: Escenario de la simulación*

Cada flecha representa un segmento (los segmentos se explicarán en profundidad más adelante) y el color representa a que SmartRoad pertenece dicho segmento (que algunos segmentos tengan un color más claro solo es una ayuda visual).

Se cuenta pues con cinco SmartRoad: A, B, C, D y E. Cada SmartRoad contiene diez segmentos. La forma correcta de interpretar este mapa es como si fuese una matriz de cinco filas y cinco columnas.

Además de los segmentos y las carreteras, podemos encontrar también dos SmartCar. El círculo con la letra A en la posición (1,1) representa una ambulancia; mientras que el círculo con la letra C en la posición (4,5) representa un vehículo estándar. El vehículo C será el encargado de mandar la señal S.O.S y la ambulancia la encargada de ir a socorrerlo. Los vehículos solo pueden desplazarse de nodo a nodo dentro de la matriz, y pueden utilizar un segmento siempre que esté abierto.

Se ha decidido utilizar una ciudad ficticia para poder focalizarse en la idea general. Me atrevería a decir que este mapa representa bastante bien lo que podría ser el esqueleto de una SmartCity; para trabajar en un entorno más realista solo haría falta trabajar con Latitud y Longitud en vez de con los nodos de una matriz, y ampliar lo que se entiende por un “segmento”.

## 3.2 ELEMENTOS DE LA SIMULACIÓN, Y SUS ROLES

### 3.2.1 SmartCity

Como se ha mencionado en la introducción de este documento, la definición de una SmartCity aún no está del todo clara, por lo que antes de empezar a explicar la simulación vamos a definir el rol que ocupa la SmartCity dentro de este proyecto en concreto.

El rol de la SmartCity es actuar como controlador de todos los eventos que ocurren dentro del radio de acción de la ciudad. Atlantis actúa como el núcleo del sistema, es el elemento encargado de armonizar a todos los participantes para que el proceso funcione correctamente. Está constantemente recibiendo información a través de la comunicación establecida con el resto de participantes, y además efectúa un control tanto de las emergencias como de las misiones activas en cada instante de la simulación.

Por último, al ser el elemento con más información de todo el sistema, también es el más capacitado para llevar a cabo la toma de decisiones, es la SmartCity la que decide qué ambulancia debe asistir (en caso de poder elegir), qué ruta ha de seguir y, en consecuencia, que segmentos han de cerrarse o abrirse.

### 3.2.2 SmartRoad

La SmartRoad actúa como un elemento independiente, pero que trabaja junto a la SmartCity para dar respuesta a los eventos que se escapan de su radio de acción (por ejemplo, accidentes). Es un sistema capaz de auto gestionar las carreteras; adaptar los límites de velocidad dependiendo de las condiciones meteorológicas, cerrar o abrir segmentos de la propia carretera en base al

### 3. INTRODUCCIÓN A LA SIMULACIÓN

flujo de, mantener a los usuarios informados de su estado... todo son acciones capaces de realizarse con la información que posee el sistema encargado de la SmartRoad.

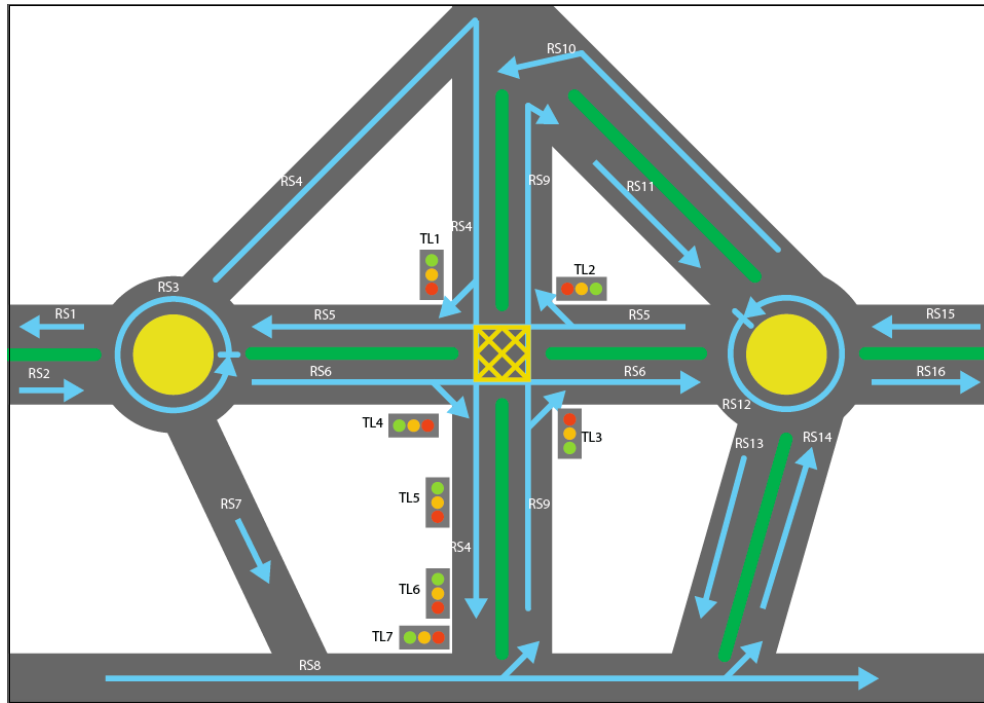
Dentro de este proyecto, la SmartRoad se limita a acatar órdenes de la ciudad (abrir o cerrar segmentos), por lo que no ha sido programada para ser capaz de auto gestionarse. Esto se debe a que, en una situación de emergencia las órdenes de la SmartCity tienen prioridad sobre la autogestión de la SmartRoad.

En resumidas, se podría pensar en las SmartRoad como un filtro entre los usuarios de las carreteras y el núcleo del sistema. Están pensadas como un sistema capaz de gestionar todos los eventos que sucedan dentro de las carreteras, dividiendo así el peso de la autogestión de la ciudad.

#### 3.2.3 Segmento

El segmento representa la unidad en la que se dividen las SmartRoad, exactamente igual que en las carreteras convencionales. Además de actuar como piezas de una carretera, también representan una capa intermediaria entre el software de la SmartRoad y el hardware que está desplegado en la carretera. Los segmentos son los que traducen las órdenes de las SmartRoad a los distintos componentes que se hayan instalado en dicho sector.

En esta simulación los segmentos son un elemento simbólico que representa un estado de la carretera, por lo que no mantienen ninguna comunicación con las SmartRoad. Dentro de la simulación a los segmentos se accede mediante métodos Java.



*Ilustración 5: Segmento realista*

En la ilustración 2 se muestra como sería el diseño de un segmento realista, el cual estaría compuesto por varios carriles y distintos elementos como semáforos, indicadores de velocidad, paneles de texto... En una solución más realista la utilización de estos segmentos nos brindaría más estados que el “segmento abierto/cerrado” utilizado por esta simulación; se podría jugar con los semáforos, reservar carriles específicos, redirigir el flujo de tráfico... con el fin de crear un camino más ligero para los servicios de emergencia.

Como se puede vislumbrar con la especificación de este segmento, diseñar una solución totalmente realista necesita un nivel de detalles muy superior al presentado en este proyecto.

### 3.2.4 Panel

Dentro de la categoría de “Panel” entraría todo lo que conocemos como señales viales. Todo lo que muestre una determinada información a los conductores y forme parte de la infraestructura de la carretera se considera Panel.

La simulación supone que todos los paneles son digitales, por lo que su texto se podría modificar vía software. Esta suposición es inviable hoy en día, por lo que se aconseja distinguir entre paneles con información “dinámica” y paneles con información “estática”; esto se mantendría durante todo el proceso de transición entre las carreteras convencionales y las SmartRoad.

### 3. INTRODUCCIÓN A LA SIMULACIÓN

En un entorno realista los paneles serían la principal vía de comunicación entre el sistema y los usuarios de las carreteras, además de la comunicación directa vía colas de mensajería o API. Dentro de la simulación solo tiene la función de cambiar el texto cuando el segmento al que pertenece ha sido cerrado o abierto, por lo que su función también es puramente simbólica.

#### 3.2.5 SmartCar

Los SmartCar son los principales usuarios de las SmartRoad, y gran parte de los usuarios de la SmartCity. Cuando hablamos de SmartCar hablamos de un coche capaz de interactuar con el entorno que le rodea, además de ser capaz de monitorizar su estado actual, un ejemplo de SmartCar sería el modelo que utiliza IBM en sus demos (IBM, 2016).

La función básica de un SmartCar estándar es hacer uso de los servicios que brinda la SmartCity o SmartRoad. En la simulación solo pueden realizar dos acciones, moverse dentro del mapa o lanzar una señal S.O.S, ya que esto es todo lo necesario para el caso de estudio.

#### 3.2.6 Ambulancia

Las ambulancias están consideradas como SmartCar dentro de la simulación, ya que comparten la funcionalidad básica de moverse en el mapa o mandar una señal S.O.S.

Además de esto, las ambulancias son capaces de comunicarse directamente con la SmartCity, ya que representan uno de los recursos que tiene el sistema para atender las distintas emergencias que se den dentro del rango de acción de la ciudad.

Por ende, las ambulancias amplían las funcionalidades de un SmartCar. Son capaces de “hacer” misiones y atender emergencias. Las misiones son el modo en el que la ciudad mensaja sus recursos para atender las distintas emergencias.

### 3.3 DIAGRAMA DE CLASES

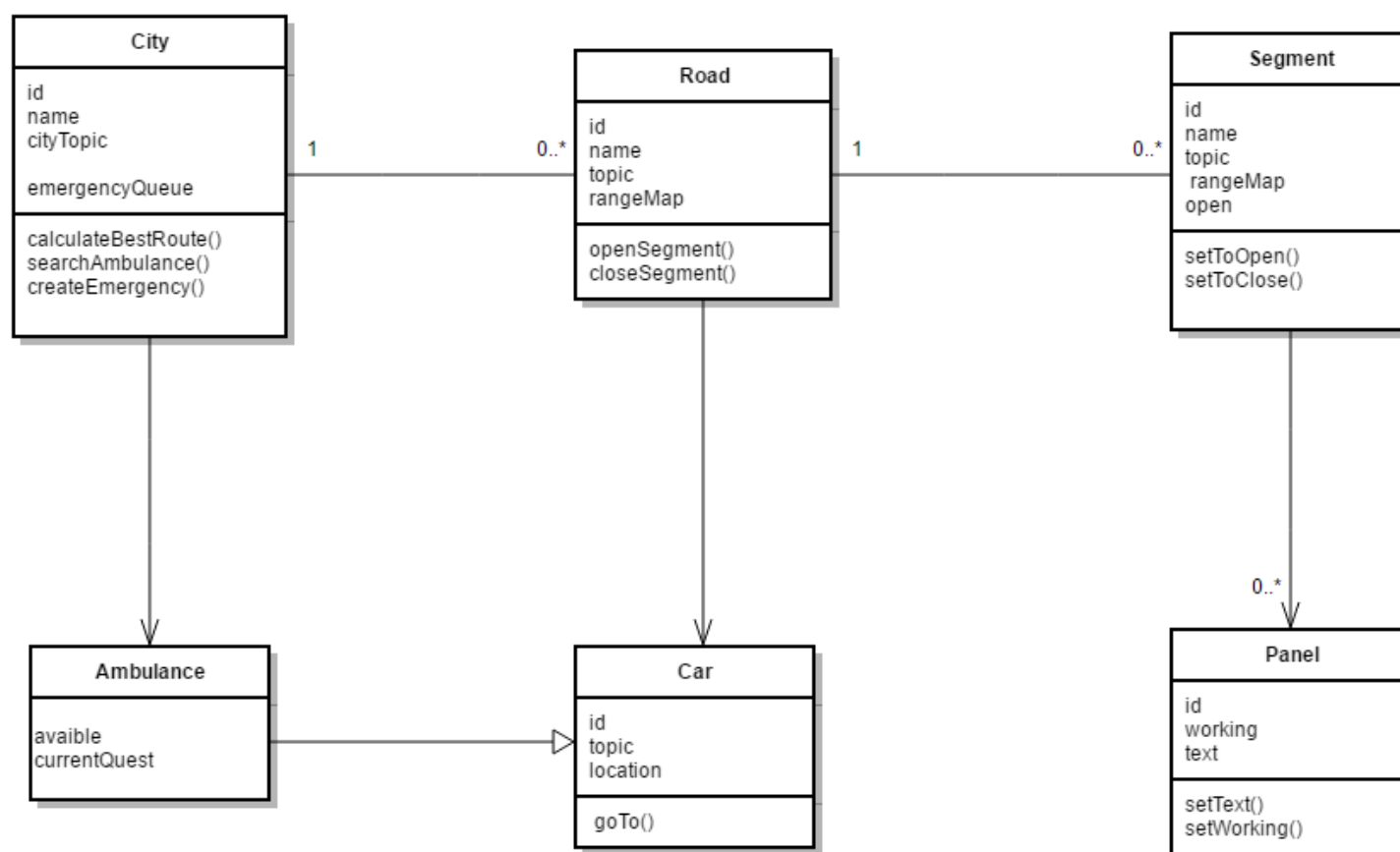


Ilustración 6: Diagrama de clases Diseño



### 3. INTRODUCCIÓN A LA SIMULACIÓN

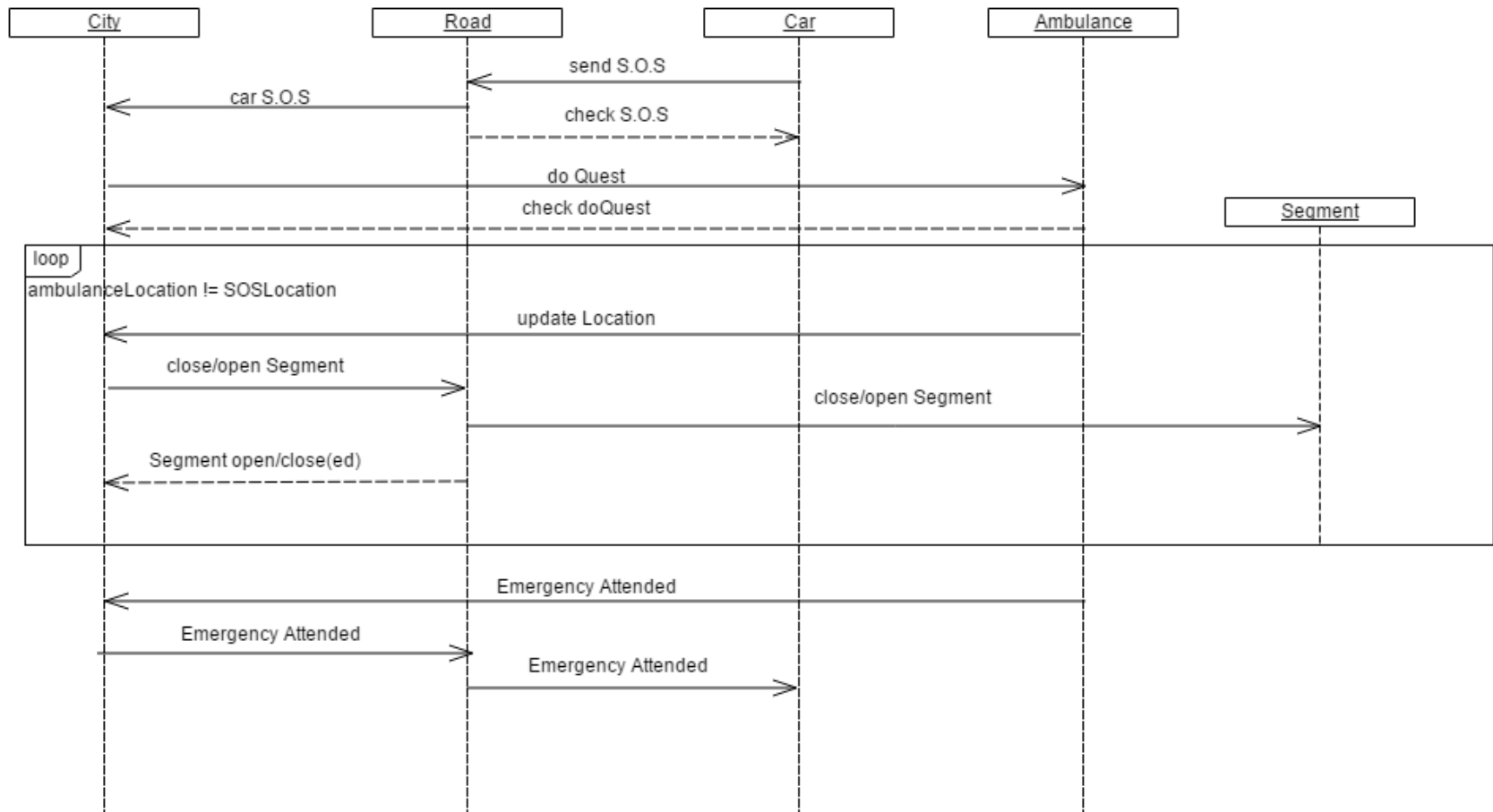


Ilustración 7: Secuencia UML diseño



## 4 COMUNICACIÓN ENTRE ELEMENTOS

---

### 4.1 FORMATO DE LOS MENSAJES

Todos los mensajes dentro de la simulación se estructuran en base al formato JSON. Además de seguir dicho formato, deben contener los siguientes atributos obligatoriamente:

- **Código:** Un código de cuatro dígitos que representa una determinada acción dentro del sistema. Cada elemento del sistema puede responder / mandar unos determinados códigos. La función principal es crear una coherencia y orden entre los mensajes que circulan por el sistema.
- **SenderId:** Identificador del elemento que ha enviado el mensaje.
- **ReceiverId:** Identificador del elemento que debe recibir dicho mensaje. Este campo puede ser *null* en determinadas situaciones.
- **Message:** El mensaje a menudo coincide con la descripción del código. Suele contener una breve descripción en “lenguaje humano” del objetivo del mensaje. El objetivo fundamental es que un profesional pueda leer este campo para poder entender el objetivo del mensaje sin tener que recurrir a la tabla de códigos.
- **Atributos opcionales:** En determinados casos la información básica no será suficiente para gestionar el mensaje, por lo que es posible que se deban añadir nuevos atributos con la información necesaria. Este campo no tiene límite, ya que cada código define los atributos necesarios para tratar una determinada operación.

La idea que subyace detrás de la obligatoriedad de estos atributos es poseer siempre una información básica para que no exista mensaje que no pueda recibir respuesta, ya sea porque el *sender* no esté autorizado, que el *receiver* no pueda atender la solicitud o simplemente que el código del mensaje no exista. En cualquier caso, con estos campos el sistema será capaz de responder en cualquier escenario.

```
JSmessage = {  
    Code: ---;  
    SenderId: ---;  
    ReceiverId: ---;  
    Message: ---;  
    *optional attributes*;  
}
```

*Ilustración 8: Formato Mensajes*

#### 4.1.1 ¿Por qué JSON?

JSON es un formato de mensajería especialmente ligero e independiente del lenguaje de programación. Es fácilmente legible por los humanos, y además fácil de *parsear* por las computadoras. Está basado en un subconjunto de JavaScript, y fue popularizado por Douglas Crockford. Actualmente es el formato más utilizado en comunicaciones asíncronas buscador/servidor, remplazando a XML el cual es utilizado por AJAX.

JSON se ajusta perfectamente a las condiciones de este proyecto, donde los dispositivos no cuentan con una gran capacidad de cómputo, por lo que la sencillez del formato es un gran punto a favor. También se tuvo en cuenta que en el entorno de IoT se intenta armonizar una gran diversidad de dispositivos, y conectarlos con la web, JSON nos permite ajustarnos a lo que actualmente se encuentra "de moda" y nos brinda mayor compatibilidad con futuras aplicaciones que necesiten establecer comunicación con el sistema encargado de gestionar la SmartCity.

Por último, para este proyecto se ha tomado como referencia la iniciativa VLCi, y dentro de su documentación se puede observar que han optado por el formato JSON como estándar de mensajería para sus comunicaciones; por lo que esto es otro punto a favor en vistas a una posible implementación real en el proyecto VLCi.

## 4. COMUNICACIÓN ENTRE ELEMENTOS

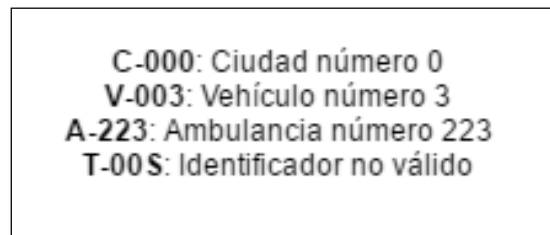
### 4.2 IDENTIFICADORES

Para identificar a los distintos usuarios de la comunicación dentro de la simulación se ha diseñado un formato de identificadores muy sencillo. Como se verá en el apartado de **Mqtt & Rest** solo hay cuatro elementos que se comuniquen: SmartCity, SmartRoad, SmartCar y Ambulance.

A cada uno de estos elementos se les asigna una letra:

- **C**ity (SmartCity)
- **R**oad (SmartRoad)
- **V**ehicle (SmartCar)
- **A**mbulance (Ambulance)

Esta letra representará el primer símbolo del identificador, seguido de un guion y un número de tres dígitos, los cuales representan un contador del número de elementos que se hayan creado. Dicho esto, en la Ilustración TODO se muestran algunos ejemplos.



*Ilustración 9: Ejemplos de Identificadores*

Para finalizar, clarificar que no pueden existir dos identificadores iguales, y que en la simulación no se reciclan identificadores. En una solución más realística los elementos que formasen parte de la infraestructura del sistema contarían con identificadores estáticos, mientras que, en el caso de los vehículos seguirían un esquema como el de las IP dinámicas: el identificador sería solicitado cada vez que el usuario accede al sistema, y cuando deje de utilizarlo dicho identificador se le asignaría a otro vehículo.

### 4.3 CÓDIGOS

En esta implementación los códigos son la forma en la que el software identifica el objetivo que tiene cada mensaje. Se componen de cuatro dígitos (TCCC) donde el primer dígito T representa el “Tema” y los tres últimos representan el “Caso”.

El **Tema** recoge a todos los mensajes que están relacionados con un mismo campo. Ejemplo: el tema número 1 corresponde a “Información Request”, por lo que todos los mensajes que tengan como objetivo solicitar información utilizarán un código que empezará con el número 1.

Cabe destacar que todos los códigos del tema 1 al tema 4 representan “solicitudes”, mientras que los códigos que se sitúan entre el tema 5 hasta el tema 8 representan las “respuestas” a dichas solicitudes.

El **Caso** representa una determinada acción dentro de un tema. Ejemplo: dentro del tema 3 “Misiones” existen varios casos; 000 (Send Quest), 001 (Open Segment), 002 (Close Segment) ...

#### 4.3.1 Descripción de cada código

En la tabla a continuación se muestran todos los códigos que se utilizan en este proyecto.

CÓDIGO	TEMA	CASO	DESCRIPCION
1000	1	000	¿Dónde estoy? (Broadcast)
1001	1	001	Manda la ubicación del SmartCar.
2000	2	000	Llamada de auxilio (S.O.S)
3000	3	000	Envío de una nueva misión
3001	3	001	Orden de abrir un Segmento
3002	3	002	Orden de cerrar un Segmento
4000	4	000	Notificar a la SmartCity que hay un nuevo SpecialVehicle
5000	5	000	Respuesta a la solicitud ¿Dónde estoy?
6000	6	000	Confirmar que la llamada S.O.S ha sido recibida correctamente

#### 4. COMUNICACIÓN ENTRE ELEMENTOS

<b>6001</b>	6	001	Tiempo hasta estimado llegue a la ubicación de la llamada S.O.S (Respuesta a una llamada de auxilio S.O.S)
<b>6002</b>	6	002	La emergencia ha sido atendida (Respuesta a una llamada de auxilio S.O.S)
<b>7000</b>	7	000	Check Quest (Confirmar que se ha recibido la misión correctamente)
<b>7001</b>	7	001	El segmento ha sido abierto correctamente
<b>7002</b>	7	002	El segmento ha sido cerrado correctamente
<b>8000</b>	8	000	Un nuevo SpecialVehicle ha sido añadido a los servicios de la ciudad correctamente.

#### 4.3.2 Atributos de los códigos

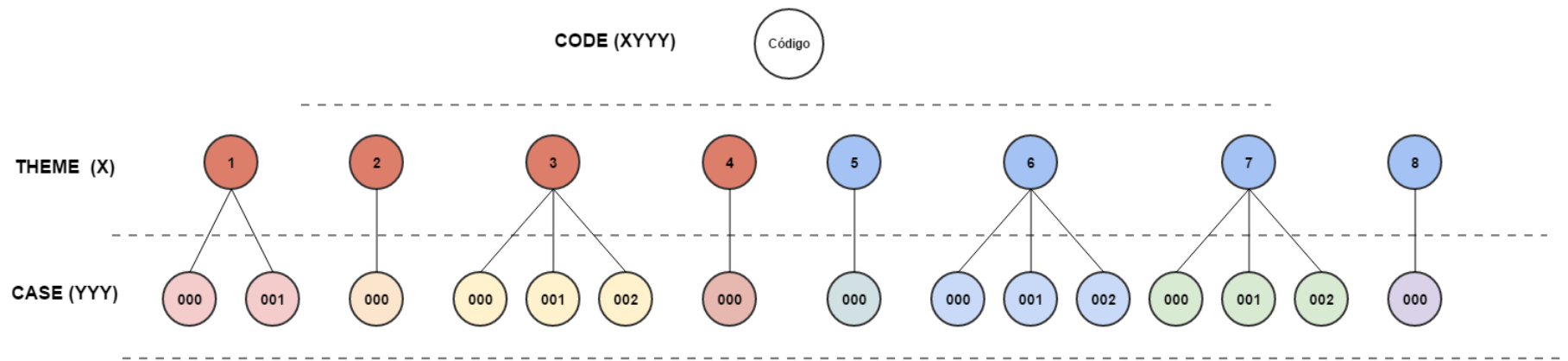
A continuación, y en caso de que sea necesario, se explicarán los atributos opcionales (en caso de que existan) todos los códigos.

<b>CÓDIGO</b>	<b>SenderId</b>	<b>ReceiverId</b>	<b>Opcionales</b>
<b>1000</b>	Vehicle	<i>Null</i>	Location Type
<b>1001</b>	Ambulance	City	LastLocation CurrentLocation NextLocation
<b>2000</b>	Vehicle	<i>Null</i>	Location
<b>3000</b>	City	SpecialVehicle	Quest AmbulanceId

<b>3001</b>	City	Road	RoadName Segment Ini Segment End
<b>3002</b>	City	Road	RoadName Segment Ini Segment End
<b>4000</b>	SpecialVehicle	City	Location Type
<b>5000</b>	City	Vehicle	Topic
<b>6000</b>	Road	Vehicle	-
<b>6001</b>	Road	-	Tiempo
<b>6002</b>	Road	Vehicle	-
<b>7000</b>	Ambulance	City	-
<b>7001</b>	Road	City	-
<b>7002</b>	Road	City	-
<b>8000</b>	City	Special Vehicle	-



#### 4. COMUNICACIÓN ENTRE ELEMENTOS



*Ilustración 10: Esquema de los códigos utilizados en la comunicación*

## 4.4 MQTT & REST

Para este proyecto se ha elegido MQTT y REST como bases de la comunicación entre dispositivos, el uso de uno u otro depende del escenario:

- Se utiliza MQTT cuando se quiere dar a la información un carácter público, es decir, cuando además de los interlocutores haya más interesados en dicha información.

*Ejemplo. Una carretera ha sido cerrada por un accidente, esta información puede ser utilizada por otras carreteras gestionar su flujo de tráfico, por los GPS de los vehículos para rediseñar su ruta...*

- Se utiliza REST para establecer comunicaciones privadas cuando se quiere transmitir información confidencial, o para evitar propagar información innecesaria para el resto de usuarios.

*Ejemplo. Un vehículo acaba de incorporarse a una carretera y desea conocer el estado de esta. Al resto de usuarios no les aporta nada saber que X vehículo solicita saber el estado de la carretera.*

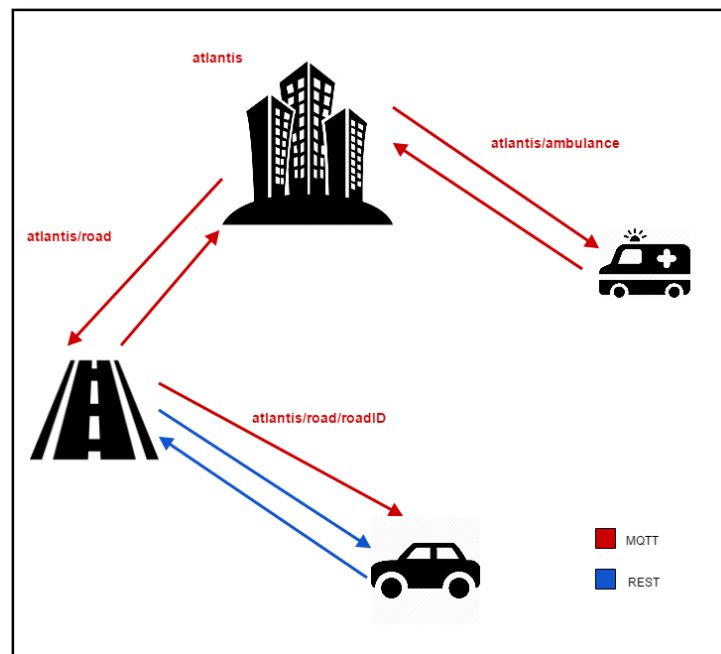


Ilustración 11: Esquema de comunicación

## 4. COMUNICACIÓN ENTRE ELEMENTOS

### 4.4.1 ¿Por qué MQTT?

MQTT es un protocolo de mensajería pub-sub diseñado para funcionar en conexiones con poca banda ancha, alta latencia o en redes poco fiables. La principal característica de MQTT es su sencillez, lo cual le permite minimizar la cantidad de recursos necesarios para hacerlo funcionar.

Fue creado en 1999 por el Dr Andy Stanford-Clark (IBM) y Arlen Nipper (Arcom, actualmente Eurotech). En 2011 IBM y Eurotech hicieron una donación al proyecto “Eclipse Paho”, el cual se utiliza en este proyecto para controlar los clientes MQTT utilizando Java. Además, desde marzo de 2013 MQTT se está sometiendo al proceso de normalización de OASIS, con el fin de convertirse en un estándar.

Los pilares de MQTT le convierten en un protocolo ideal para IoT (Internet of Things) o M2M (Machine-to-Machine), donde los dispositivos no disponen de una gran cantidad de recursos. Dentro de nuestro proyecto, se propone utilizar RaspBerry Pi junto a Arduino a modo de prototipo para empezar a articular una SmartCity, donde el protocolo MQTT encaja perfectamente.

Otro punto a favor para MQTT es que su integración con Java es muy sencilla, gracias a “Paho”, el proyecto mencionado anteriormente, el cual nos permite gestionar los clientes MQTT con gran sencillez, ofrece una buena API y además cuenta con bastante documentación.

Para utilizar MQTT es necesario un *broker*, en este proyecto se ha optado por utilizar Mosquitto. Una de las principales razones fue que Mosquitto es un proyecto incubado por Eclipse, el cual también está dirigiendo “Paho”, por lo que es de esperar que la sinergia entre estos dos proyectos tienda a ser mejor que utilizando otros *brokes* también viables.

Por último, ya se había trabajado antes en un proyecto el cual gestionaba varias comunicaciones entre dispositivos Arduino via MQTT utilizando Mosquitto como *broker*; y como se suele decir, la experiencia es un grado.

### 4.4.2 ¿Por qué REST?

(TODO: Completar cuando se tenga claro que protocolo se va a utilizar )

## 4.5 COMUNICACIÓN MQTT

Existen varios *topics* dentro de la simulación, y cada uno tiene una función definida. Quién puede o no puede suscribirse a cada uno de estos topics es algo que, por ahora, no está definido.

(TODO: información de las city totalmente publica, o habrá restricciones?)

- **atlantis** Este *topic* está reservada para eventos que solo puedan ser atendidos por las SmartCity (emergencias, gestión de misiones...), por lo que, los únicos que pueden hacer

uso de el son los elementos que forman parte de la infraestructura de las SmartCity (SmartRoad, vehículos autorizados...)

- **atlantis/ambulance** Como bien indica el nombre, este *topic* representa el puente comunicativo entre las ambulancias que trabajan para la ciudad, y la misma ciudad. Gestionar misiones, seguimiento de la ubicación de las ambulancias, averías... son algunos de los temas tratados por este *topic*.
- **atlantis/road** *Topic* reservado para la comunicación entre las carreteras y la ciudad. Toda la información de las carreteras es retransmitida por aquí; también se utiliza para comunicar emergencias, cerrar o abrir segmentos...
- **atlantis/road/roadID** Cada carretera tiene su propio *topic* por el cual la carretera se comunica con los vehículos que circulan por esta. Cabe destacar que esta comunicación no es bidireccional, la carretera informa a los suscriptores de información válida para todos, pero si un vehículo necesita comunicarse con la ciudad, lo debe hacer vía REST y no utilizando este canal. La funcionalidad principal de este canal es mantener a los vehículos que circulan por la carretera informados del estado de esta.

## 4.6 COMUNICACIÓN REST

Como se ha mencionado anteriormente, el objetivo de la comunicación REST en esta simulación es mantener comunicaciones ‘privadas’ o evitar propagar información ‘inútil’ para el resto de usuarios.

Dentro del escenario de la ambulancia, la única comunicación REST que se puede encontrar es la que establece el SmartCar con la carretera, para hacer la llamada S.O.S.

(TODO: Explicar detalles de la comunicación REST cuando esté diseñada/implementada)

#### 4. COMUNICACIÓN ENTRE ELEMENTOS

## 5 SIMULACIÓN PASO A PASO

En el siguiente tema se explicará paso a paso como la solución propuesta se desarrolla dentro de la simulación, esto se hará desde una perspectiva más técnica con el fin de demostrar la viabilidad de la solución planteada anteriormente.

### 5.1 CARGA DEL ESCENARIO

El primer paso es realizar la carga del escenario en Java. Esto se hace a través de la clase *EnvironmentOne.java*, todos los elementos de la simulación son objetos Java inicializados en dicha clase. Los constructores de cada objeto Java almacenan toda la información requerida para realizar la simulación, los parámetros que necesitan representan la Input que necesitará el sistema para ubicar dicho elemento dentro de la infraestructura de la SmartCity.

Después de crear la SmartCity y todas las SmartRoad, el siguiente paso es añadir los segments a las SmartRoad con sus respectivos Panels. Una vez hecho esto solo faltará crear los dos SmartCar, la ambulancia y el vehículo estándar que será el encargado de dar la señal S.O.S. Se ha de destacar que una vez se crea la ambulancia, esta se registra como un "SpecialVehicle" dentro de la ciudad a la que pertenezca, lo que le permitirá a la ciudad mantener un seguimiento de los recursos disponibles para atender las emergencias.

```
{ "Code": "4000", "SenderId": "A-000", "ReceiverId": "C-000", "Message": "New special vehicle", "Location": "00", "Type": "ambulance" }
{ "Code": "8000", "SenderId": "C-000", "ReceiverId": "A-000", "Message": "You have been added to Special Vehicles of atlantis city." }
```

*Ilustración 12: Ambulance avisa a la ciudad de que hay un nuevo SpecialVehicle. La ciudad lo añade y le manda un Check.*

Una vez hecho esto, todo el escenario está preparado para empezar la simulación.

### 5.2 SEÑAL S.O.S

(Aclarar como manda la señal de SOS el coche, via Mqtt o via REST...)

## 5. SIMULACIÓN PASO A PASO

El coche manda una señal S.O.S a través de (completar aquí) la cual es recibida por la SmartRoad a la que se ha suscrito. Esta señal se mandará cada “x” tiempo hasta que se reciba un mensaje de confirmación como señal de que la emergencia está siendo atendida.

```
{ "Code": "2000", "SenderId": "V-000", "ReceiverId": "null", "Message": "S.O.S", "Location": "43" }  
{ "Code": "6000", "SenderId": "R-004", "ReceiverId": "V-000", "Message": "Check S.O.S" }
```

*Ilustración 13: Señal S.O.S (JSON)*

*Nota: La simulación supone que el SmartCar es capaz de reconocer en que carretera se encuentra en base a su ubicación GPS.*

Una vez la señal ha sido recibida por la SmartRoad a través del hardware instalado en los segmentos, se activa el protocolo de actuación ante una señal S.O.S. La primera acción que realiza la carretera es mandar un mensaje a la ciudad usando el *topic atlantis*. Hecho esto, mandará un mensaje a modo de “check” al vehículo que hay mandado la señal S.O.S.

Una vez la ciudad haya mandado un mensaje corroborando que ha recibido la información del accidente, la señal habrá llegado al núcleo del sistema, donde dará comienzo una respuesta automática por parte de la ciudad.

### 5.3 PLANIFICACIÓN DEL PLAN PARA ATENDER LA EMERGENCIA

Una vez la señal ha llegado a la ciudad, el primer paso es crear una Emergencia. Dentro de la simulación las Emergencias son objetos java que representan toda la información relacionada con el evento en cuestión, hora de la llamada, ubicación, estado de la emergencia... Una vez se ha creado el objeto, se inserta en una cola de emergencias pendientes, que es el principal mecanismo de la SmartCity para llevar control sobre todas las emergencias que se están gestionando en un determinado momento.

```
{ "Code": "2000", "SenderId": "R-004", "ReceiverId": "C-000", "Message": "{ \"Code\": \"2000\", \"SenderId\": \"V-000\", \"ReceiverId\": \"null\", \"Message\": \"S.O.S\", \"Location\": \"43\" }", "Location": "43", "RequesterId": "V-000" }
```

*Ilustración 14: SmartRoad notifica a la SmartCity de una nueva emergencia*

Hecho esto, se procede a comenzar la búsqueda de la mejor ambulancia para atender el siniestro, la búsqueda se realiza en una lista que contienen todas las ambulancias disponibles a servicio de la ciudad en cuestión. Actualmente el único factor que se tienen en cuenta a la hora de buscar una ambulancia es la distancia a la que se encuentra del siniestro, aunque perfectamente podría hacerse en base a otros factores.

Una vez se ha seleccionado la ambulancia óptima para atender dicha emergencia se procede a calcular la ruta óptima desde la ubicación de la ambulancia hasta la ubicación del siniestro. El cálculo de una ruta óptima es complicado y difícil de definir, existen muchos factores: fluidez del tráfico, disponibilidad de carriles especiales, capacidad para abrir o cerrar segmentos, zonas urbanas por las que hay peligro de accidente... por lo que para evitar emplear tiempo en un método el cual necesitaría del apoyo de varios expertos, dentro de la simulación se ha simplificado a “la distancia mínima entre dos nodos de la matriz”.

Con la información de la emergencia, la ambulancia óptima seleccionada y una ruta calculada la SmartCity posee todos los elementos necesarios para crear una Misión. Dentro de la simulación las misiones son objetos java, al igual que las emergencias, que contienen toda la información necesaria para que un recurso de la ciudad realice las tareas que se le han encomendado de una forma eficiente. Una vez creado el objeto Misión, se mandará a la ambulancia a través del *topic atlantis/ambulance*. Una vez hecho esto, la SmartCity queda a la espera de una confirmación por parte de la ambulancia.

Una vez la ambulancia haya confirmado que puede realizar la misión encomendada, la ciudad marcará la Emergencia como “en progreso” y cambiará el estado de la ambulancia de disponible a “en misión”. Finalmente, la ciudad mantendrá un seguimiento de la emergencia hasta que un profesional confirme que la emergencia ha sido atendida.

## 5.4 GESTIÓN DE SEGMENTOS CON AMBULANCIA EN RUTA

Con el objetivo de ahorrar el máximo tiempo en el camino hacia el siniestro, la SmartCity puede abrir o cerrar segmentos al tráfico estándar. Para ser lo más preciso posible con estas acciones, una vez la ambulancia se pone en ruta, empieza a mandar un mensaje con su ubicación cada que va a abandonar el segmento en el que se encuentra, el mensaje se manda a través del *topic atlantis/ambulance*.



## 5. SIMULACIÓN PASO A PASO

```
{ "Code": "1100", "SenderId": "A-000", "ReceiverId": "C-000", "Message": "My location (last, current and next)", "LastLocation": "00", "CurrentLocation": "00", "NextLocation": "10" }
{ "Code": "1100", "SenderId": "A-000", "ReceiverId": "C-000", "Message": "My location (last, current and next)", "LastLocation": "00", "CurrentLocation": "10", "NextLocation": "20" }
```

*Ilustración 15: Ambulancia informando de su ubicación actual.*

El mensaje es leído por la ciudad, y se considera si es necesario cerrar algún segmento en la ruta de la ambulancia. En la simulación que se ha implementado se utiliza un algoritmo simplista para abrir y cerrar segmentos. Cada vez que la ambulancia manda su ubicación, la smartCity manda cerrar el próximo segmento en la ruta de la ambulancia, y en caso de que sea necesario, manda abrir el segmento por el que acaba de circular la ambulancia.

```
{ "Code": "3001", "SenderId": "C-000", "ReceiverId": "R-000", "Message": "Open the segment", "SegmentIni": "00", "SegmentEnd": "00" }
{ "Code": "7001", "SenderId": "R-000", "ReceiverId": "C-000", "Message": "Segment Opened" }
{ "Code": "3002", "SenderId": "C-000", "ReceiverId": "R-000", "Message": "Close the segment", "SegmentIni": "00", "SegmentEnd": "10" }
{ "Code": "7002", "SenderId": "R-000", "ReceiverId": "C-000", "Message": "Segment Closed" }
```

*Ilustración 16: Comunicación entre la ciudad y las Road mientras se abren y cierran segmentos.*

Este proceso se realiza utilizando el *topic atlantis/road* con el objetivo de que todas las carreteras sean conscientes de los segmentos que se van a cerrar o abrir, y puedan prepararse para un posible flujo de tráfico superior al que esperaban.

Como es habitual, cada vez que se manda una orden se ha de esperar a recibir un mensaje de confirmación, en este caso por parte de las SmartRoad. Cuando la orden llega a las carreteras, el cierre o apertura de los segmentos se hace mediante métodos brindados por los objetos Java. En este caso, las carreteras cuentan con una lista en la cual almacenan todos los segmentos que forman parte de ellas, por lo que se busca el objeto con un identificador idéntico al que viene en el mensaje mandado por la SmartCity, y se procede a utilizar sus métodos para cambiar su estado. El cambio de estado de cualquier Segmento también provoca un cambio en el texto mostrado por los paneles que formen parte de dicho segmento.

Este bucle de abrir/cerrar segmentos se prolonga hasta que la ambulancia haya llegado a su destino. Esto se detectará porque la ambulancia mandará un mensaje donde confirmará que ha llegado a su destino.

## 5.5 LA AMBULANCIA LLEGA A SU DESTINO

Una vez la ambulancia ha llegado a la ubicación del siniestro pueden darse dos situaciones:

- El profesional considera que hacen falta más recursos para atender la emergencia. Esto se comunicará a la ciudad a través del *topic atlantis*, en el mensaje se deberá especificar la situación en la que se encuentra el siniestro, y cuantas son las unidades que el profesional piensa que son necesarias para atender el siniestro. El proceso encargado de gestionar la emergencia inicial deberá redefinir la emergencia con la información recibida, y cambiar su estado a “pendiente”. *Nota: Este escenario no ha sido implementado dentro de la simulación.*
- El profesional se ve capacitado para atender la emergencia y procede a ello. Una vez la emergencia haya sido atendida se le comunicará a la ciudad vía el *topic atlantis/ambulance*. Hecho esto la ciudad borrará la emergencia de la lista de emergencias pendientes y cambiará el estado de la ambulancia de “en misión” a “disponible”.

```
{"Code": "6002", "SenderId": "R-004", "ReceiverId": "V-000", "Message": "Your emergency has been attended"}
```

*Ilustración 17: La SmartRoad comunica a el vehículo de que su emergencia ha sido atendida correctamente.*

En este punto la simulación habría finalizado. La ambulancia se encontraría en la misma ubicación que el coche que envió la señal S.O.S y su estado volvería a ser “disponible”. Todos los segmentos estarían abiertos, y la ciudad no tendría ninguna emergencia pendiente por tratar.

## 5. SIMULACIÓN PASO A PASO

## 6 REFERENCIAS

---

Ashton, K. (22 de 06 de 2009). *RFID Journal*. Obtenido de <http://www.rfidjournal.com/articles/view?4986>

BCN. (15 de 06 de 2016). *BCN Smart City*. Obtenido de <http://smartcity.bcn.cat/>

DGT. (2014). *Dirección General de Tráfico - Ministerio del Interior - Gobierno de España*. Obtenido de <http://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/accidentes-30dias/tablas-estadisticas/2014/>

IBM. (2016). *Connected Car*. Obtenido de <http://m2m.demos.ibm.com/connectedCar.html>

OMS. (14 de 06 de 2016). *World Health Organisation*. Obtenido de [http://www.who.int/violence\\_injury\\_prevention/road\\_traffic/en/](http://www.who.int/violence_injury_prevention/road_traffic/en/)

UW. (10 de 06 de 2016). *Inhabitat - Students Install the World's First Solar Pavement Panels in Virginia*. Obtenido de <http://inhabitat.com/students-install-the-worlds-first-solar-pavement-panels-in-virginia/>

WIEN. (2015). *Technische Universitat WIEN*. Obtenido de <http://www.smart-cities.eu>

¿Por qué Mqtt?

<http://mqtt.org/>

<http://mosquitto.org/>

<http://www.eclipse.org/paho/>

¿Por qué JSON?

<http://www.json.org/>

[http://www.valencia.es/ayuntamiento/DatosAbiertos.nsf/0/2113BD9D1693D7EAC1257C6600449981/\\$FILE/API%20APPCIUDAD%20v3.pdf?OpenElement&lang=1](http://www.valencia.es/ayuntamiento/DatosAbiertos.nsf/0/2113BD9D1693D7EAC1257C6600449981/$FILE/API%20APPCIUDAD%20v3.pdf?OpenElement&lang=1)

NOTAS

## 6. REFERENCIAS

*Nota: Para que esto sea posible hace falta una infraestructura básica, por lo que en el proyecto se da por hecho que todos los elementos cuentan con la tecnología necesaria para llevar a cabo las comunicaciones y cálculos que la solución presentada necesita.*