UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA

DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Implementación de una *Road Side Unit* de comunicación infraestructura vehículos según el estándar ETSI ITS G5

Autor:

D. Alejandro Lobo González

Tutor:

Dr. D. Juan Carlos Aguado Manzano

Valladolid, 14 de Julio de 2017

TÍTULO:	Implementación de una *Road Side Unit* de comunicación infraestructura vehículos según el estándar ETSI ITS G5
AUTOR:	D. Alejandro Lobo González
TUTOR:	Dr. D. Juan Carlos Aguado Manzano
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Dr. D. Ignacio de Miguel Jiménez
VOCAL:	Dr. D. Juan Carlos Aguado Manzano
SECRETARIO:	Dr. D. Ramón Durán Barroso
SUPLENTE:	Dr. Dña Noemí Merayo Álvarez
SUPLENTE:	Dr. D. Rubén M. Lorenzo Toledo

FECHA: 14 de Julio de 2017

CALIFICACIÓN:

Resumen de trabajo fin de grado

En este proyecto se ha realizado una implementación de un primer prototipo de estación ITS basado en el estándar ETSI ITS G5 para el campo de las comunicaciones V2X. Dicho prototipo está implementado a través del software de código abierto proporcionado por el grupo CSS Labs, de la Universidad de Paderborn, el cual nos permitiría estudiar las funcionalidades que tiene la estación ITS, que en el caso del este proyecto sería una RSU. La mayor aportación que hace este proyecto frente a otros que siguen esta línea de investigación es la amplia y detallada explicación del funcionamiento en global de la capa de *facilities* del estándar ETSI ITS G5, ya que la información de cómo esta capa actúa de una manera conjunta es escasa e incompleta. El funcionamiento conjunto de dicha capa se estudió tanto teóricamente como a través de simulaciones y pruebas reales en donde se obtuvieron datos de un vehículo real, permitiendo ver los diferentes casos en los que las estaciones ITS generan mensajes para comunicarse entre sí.

Abstract

During this project it has been implemented a first prototype ITS station based on the ETSI ITS G5 standard for the field of V2X communications. This prototype uses the open source software provided by CSS Labs group of the University of Paderborn, which would allow us to study the functionalities of a ITS station, that in the case of this project would be a RSU. The main contribution made by this project in contrast to others which follow this research line is the wide and detailed explanation of the overall operation of the ETSI ITS G5 standard layer facilities, since the information about how this layer works in a joint way is in the best of the cases scarce and incomplete. The whole performance of the layer was studied theoretically and through simulations and real tests, in which data were obtained from a real vehicle allowing to see the different cases in which the ITS stations generate messages to communicate with each other.

Palabras clave

OpenC2x, V2X, CAM, DENM, capa de *facilities,* estándar ETSI ITS G5, estándar 802.11p, Intelligent Transport Systems (ITS), OBDII, GPS, tecnología 5G, LDM, DCC, estación ITS

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1

Introducción

1.1 Contextualización y motivación del proyecto fin de grado

La gran demanda que existe hoy en día de movilidad y transporte ha provocado la necesidad de adoptar las medidas necesarias para que viajar utilizando un vehículo particular sea lo menos arriesgado posible. Este hecho unido a las campañas de concienciación realizadas por los gobiernos sobre la seguridad vial ha producido que los accidentes mortales en carretera disminuyan paulatinamente [1], aunque el riesgo de accidente siga siendo elevado.

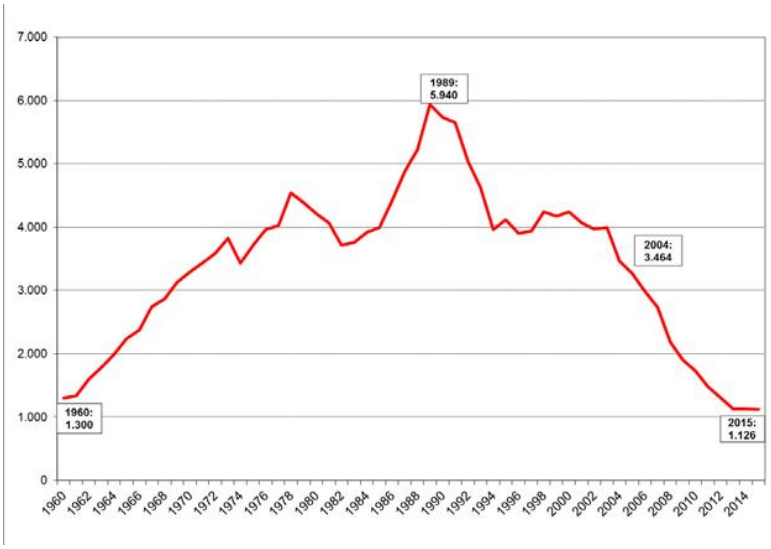

Figura 1. Evolución del nº de accidentes en vías interurbanas en España entre 1960 y 2015 [1].

Dado que el sector tecnológico y de comunicaciones móviles sigue creciendo a grandes pasos, parece que el desarrollo de nuevos sistemas de transporte más eficientes basados en estos dos campos de trabajo se plantea como una solución alternativa [2] para intentar reducir los accidentes de una manera notable. De esta manera surgió una nueva tendencia en investigación en donde se pretendían desarrollar los sistemas de transporte

inteligentes (Intelligent Transport Systems, ITS), definidos como sistemas de transporte a los que se aplican tecnologías avanzadas en información y comunicación, incluyendo Internet, para aumentar la seguridad, eficiencia y comodidad [3]. Estos ITS no están incorporados solamente en vehículos, sino también en todos los elementos que rodean la infraestructura vial.

La comunicación inalámbrica permitirá el uso de estos ITS, cubriendo tanto la comunicación vehículo a vehículo (V2V) como la comunicación vehículo a infraestructura (V2I). Normalmente para referirse a estos tipos de comunicaciones se utiliza el término V2X (Vehicle to everything). En la Figura 2 se puede ver un ejemplo de un entorno cooperativo ITS, donde se representan distintos tipos de comunicación de V2X.

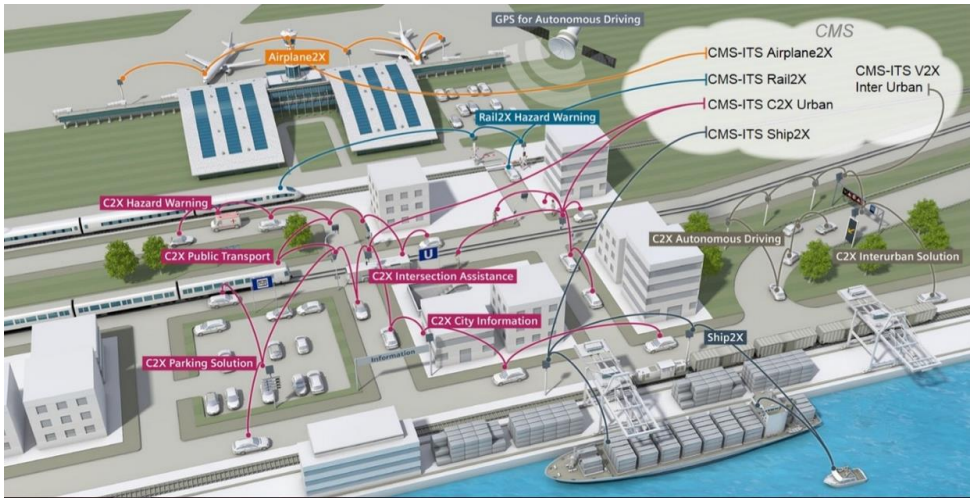

Figura 2. Ejemplo de entorno ITS cooperativos [43].

Para realizar comunicaciones V2X se emplean redes ad-hoc vehiculares (Vehicular ad hoc networks, VANETs), en las que se utilizan los vehículos como nodos de la red y se debe permitir el intercambio rápido de información, no solo para advertir rápidamente al conductor de un posible peligro en la circulación sino también porque al emplear tecnologías de corto alcance DSRC, la conectividad se establece de forma esporádica a través de un enlace ad-hoc [36]. Estas redes tienen sus propias problemáticas asociadas, pero en este trabajo fin de grado no se abordarán, sólo mencionándose en donde sea conveniente de forma sucinta.

De mayor interés para este trabajo final de grado son las aplicaciones que se pueden asociar a los sistemas de transporte inteligente, las cuales han sido clasificadas en cuatro grupos principales, tal y como refleja Tabla 1.

Clase de aplicación	Aplicación	Típicos casos de uso
Seguridad vial activa	Asistencia en carretera: conciencia cooperativa Asistencia en carretera: aviso de peligro en la calzada.	Aviso de vehículo de emergencias Aviso de vehículo en sentido contrario Aviso de obras en la calzada
Eficiencia de tráfico cooperativo	Gestión de velocidad	Notificación de los límites de velocidad.
	Navegación cooperativa	Información sobre el tráfico y recomendación de la ruta a seguir
Servicios locales cooperativos	Servicios básicos locales	Notificación sobre un punto de interés Gestión de parking
Servicios globales de Internet	Servicios comunitarios	Administración de garajes
	Gestión de la vida de dispositivos	Calibración de los datos de las estaciones ITS. Control del hardware/software de las estaciones ITS

Tabla 1. Aplicaciones y casos de uso para la comunicación V2X

Las comunicaciones V2X tienen lugar dentro de entornos ITS cooperativos y se realizan entre estaciones ITS, que pueden ser de dos tipos:

- Road Side Unit (RSU): dispositivo situado en la carretera y que proporciona soporte de conectividad a los vehículos que circulan por ella.
- On Board Unit (OBU): dispositivo instalado en el vehículo que permite el intercambio de información con otras OBUs o RSUs.

Las estaciones ITS son entidades funcionales basadas en la pila de protocolos del estándar ETSI ITS G5 (sección 3.2) que se transmiten datos entre sí a través de aplicaciones ITS utilizando cualquier red de comunicación disponible, definiendo las aplicaciones ITS como instancias de un servicio ITS que implica la asociación entre dos o más procesos de la capa de aplicación de dos estaciones ITS [3].

Empresas como General Motors [4] y Ford [5] en Estados Unidos fueron de las primeras en tener prototipos en funcionamiento y en la actualidad ya existen vehículos que cuentan con esta tecnología basada en la pila de protocolos ETSI ITS G5 embebida en su sistema, aunque utilizados únicamente para fines de investigación y desarrollo, no comerciales. En Europa dicho estándar ha sido testeado durante los últimos 10 años en proyectos como Drive C2X [6] o ITS corridor [7], en el que infraestructuras de Países Bajos,

Alemania y Austria evaluaron el estándar 802.11p para V2I y tecnologías de sistemas centrales en entornos ITS cooperativos.

El estándar 802.11p es utilizado en la capa de acceso de la pila de protocolos del estándar ETSI ITS G5 para la transmisión de datos entre estaciones ITS. En la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid se realizó un trabajo fin de máster a cargo de Javier Fernández Pastrana [8], cuyo objetivo era la implementación del estándar 802.11p en un entorno Linux empleando tarjetas Wi-Fi comerciales de bajo coste y donde se pueden encontrar de forma detalladas las características y funcionalidades de dicho estándar. El proyecto anteriormente citado se empleó como punto de partida en la realización del presente proyecto, con la finalidad de aprovechar las ventajas del estándar 802.11p, relatadas en dicho trabajo fin de máster.

En la línea marcada por las investigaciones anteriormente mencionadas, multitud de empresas han lanzado sus propios proyectos de código abierto (Open Source Projects) con el objetivo de implementar estaciones ITS y después realizar y/o evaluar su proyecto en un entorno real. Para este proyecto se ha utilizado el OpenC2X Source Project, desarrollado por la Heinz NixDorf Institute (Universidad de Paderborn) [9].

Dicho proyecto OpenC2X fue elegido debido a que permitía construir una primera implementación de una estación ITS muy completa, ya que dicho sistema basado en el protocolo ETSI ITS G5 cuenta con una implementación casi total de la pila de protocolos de dicho estándar. Este hecho posibilitaría trabajar con un primer prototipo bastante real de una estación ITS (en el caso del dicho proyecto una RSU), hasta conocer todas las funcionalidades que esta tiene.

El principal aporte llevado a cabo tras la realización de este proyecto es el estudio detallado realizado del estándar, sobretodo en la capa de *facilities,* que es una de las capas de la pila de protocolos del estándar ETSI ITS G5 de las que menos información se tiene (prácticamente solo se manejan los estándares que definen los protocolos de dicha capa), ampliando la información acerca de dicha capa sobretodo desde el punto de vista práctico (mediante la realización de simulaciones en el laboratorio y reales) y colaborando de este modo a la línea de desarrollo que se está siguiendo en materia de comunicaciones entre vehículos en el marco de la seguridad vial.

1.2 Objetivos

El propósito de este proyecto ha sido la implementación de un prototipo de RSU a partir del uso de un proyecto de código abierto, consiguiendo de esta manera establecer comunicaciones V2X como las que se pueden ver en la Figura 3 y realizar el estudio de las mismas.

Figura 3. Ejemplos de casos de uso en comunicaciones entre vehículos [42].

Para llegar a tal fin se partió de un proyecto anterior [8], donde se implementa el estándar necesario a nivel de capa de acceso para la comunicación entre vehículos y sobre la cual se empezó a desarrollar este proyecto. Dicho proyecto permite realizar tanto pruebas simuladas como pruebas en un entorno real, permitiendo observar las similitudes y diferencias de los datos recogidos en cada tipo de prueba. Para el caso de las pruebas de campo, fue necesario emplear los dispositivos Dongle OBDII [10] y GPS [11], coordinándolos con el proyecto para un correcto funcionamiento del mismo.

La lista de objetivos específicos cubiertos en este proyecto es la siguiente:

- **Instalación del kernel en su versión correcta** que será necesario para elaborar el proyecto actual.
- **Comprensión del estándar 802.11p** empleado en las comunicaciones entre vehículos, el cual supondría la base del proyecto y cuya implementación fue desarrollado en un trabajo fin de máster anterior.
- **Búsqueda de información** acerca de las comunicaciones entre vehículos. Una vez que se recopiló toda la información necesaria, se procedió a la elección del proyecto de código abierto que mejor se adecuasen al objetivo final de implementar una RSU bajo el estándar ETSI ITS G5.
- **Instalación del software** correspondiente al proyecto de código abierto elegido y análisis detallado del mismo, tanto a nivel de código, viendo como se relacionaban todos los componentes del proyecto, como a nivel funcional,

mediante simulaciones en el laboratorio, observando los mensajes enviados entre todos esos componentes.

- **Empleo de dispositivos y pruebas en un entorno real.** Dado que el objetivo de dicho proyecto era la implementación de un RSU, estación ITS empleada en comunicaciones V2X, el proyecto utilizado permite recoger datos reales de un vehículo, de tal forma que puede ver en un mapa el seguimiento del vehículo así la información relevante del mismo para hacer llegar la información de una manera sencilla a cualquier usuario que utilice el sistema.

1.3 Fases del proyecto

La realización del proyecto comenzó con una **primera fase** en la que se realizó la instalación del software necesario (S.O Linux y kernel 4.2.8) en un ordenador proporcionado por la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid para conseguir la transmisión bidireccional de datos entre dicho ordenador y otro que ya tenía instalado dicho software.

Además, en esta primera etapa fue necesario realizar un estudio del estándar 802.11p, debido a que dicho estándar es empleado en la capa de acceso del estándar ETSI ITS G5 [17], utilizado en el proyecto OpenC2X, para permitir el intercambio de información entre todos los vehículos involucrados en la comunicación. Por lo tanto, fue fundamental saber cómo funcionaba dicho estándar y para tal fin se utilizó como base el trabajo fin de máster realizado por Javier Fernández Pastrana [8], donde se explica cómo actúa y cómo se implementa la capa de acceso correspondiente al estándar 802.11p en un entorno Linux.

Para la **segunda fase,** una vez montado todo el sistema necesario, se realizó una búsqueda y comparación de los principales proyectos de código abierto que implementaban una estación ITS utilizando el estándar ETSI ITS G5. A la conclusión de este pequeño estudio, el proyecto elegido fue el OpenC2X.

La **tercera fase** se dividió en dos partes principales:

- En una primera parte se realizó la instalación del software correspondiente al proyecto OpenC2X elegido.
- La segunda parte fue esencialmente teórica, ya que en ella se analizaron todos los módulos y protocolos que implementaban la pila de protocolos del estándar ETSI

ITS G5 y para ello fue necesario una lectura profunda de los estándares que los definían.

Para la **cuarta fase**, ya con todo el software instalado y un conocimiento pleno de las funcionalidades de cada uno de los módulos que conforman el estándar, se realizó un testeo del proyecto. Para esta fase de testeo, se llevaron a cabo una serie de pruebas con el objetivo de mostrar si los mensajes enviados entre dos estaciones ITS se generaban según las condiciones establecidas por los estándares que definían los módulos. Fue de vital importancia la realización de muchas pruebas para comprobar que el software era estable y que todos los módulos del mismo se comunicaban entre ellos correctamente.

En la **quinta y última fase**, se realizaron pruebas reales en vehículos con dos objetivos fundamentales. El primer objetivo fue analizar los mensajes enviados por el vehículo y compararlos con los mensajes generados en las simulaciones realizadas en el laboratorio y el segundo fue recoger datos del vehículo para utilizarlos en el proyecto. Los datos que obtendremos del vehículo serán su velocidad y posición y se introducirán a nuestro proyecto a través de los siguientes elementos respectivamente:

- Dongle OBDII [10].
- GPSlim236 [11].

2

Estado del Arte

2.1 Estándares y proyectos en sistemas ITS

En los últimos años, muchas empresas han llevado a cabo diversas investigaciones para lograr el propósito de crear un entorno ITS cooperativo, incorporando para ello sistemas ITS a vehículos e infraestructuras, consiguiendo de esta manera realizar una comunicación V2X y mejorar tanto el estado del tráfico como la seguridad vial. Algunos de los proyectos llevados a cabo se pueden apreciar en la Tabla 2 [12].

Categoría	Título del proyecto	Objetivo	Consorcio
Proyectos basados en comunicación cooperativa	SAFESPOT	Sistema cooperativo enfocado en seguridad vial para prevenir accidentes	Bosch, Daimler, Volvo, University of Stuttgart, CNRS, …
	DRIVE C2X	Proyecto encargado de desarrollar un sistema cooperativo en Europa	PSA, Renault, HITACHI, Orange, EURECOM, INRIA, IFSTTAR
Seguridad, asistencia para accidentes y gestión de incidentes	iTETRIS	Conseguir intercambio de datos más rápido en comunicaciones V2X	Thales, Innovalia Association, Deutsches Zentrum für Luft, University Miguel Hernández de Elche
	WiSafeCar	Implementar una plataforma para la comunicación inalámbrica que permita mejorar las condiciones de tráfico	CRP Henri Tudor, Taipale Telematics, Sunit, Ubridge, FMI, VTT

Tabla 2. Proyectos internacionales de comunicación V2X [12].

Uno de los primeros proyectos que intentó desarrollar una red ITS en Europa fue el llamado proyecto CONCERT [13], desarrollado desde 1996 a 1998, y cuyo objetivo fue el de establecer comunicación a través de aplicaciones telemáticas con diferentes medios de transporte para el control de la polución, mediante el uso de integrado de pequeñas tarjetas inteligentes y de control de acceso del uso de la vía. Este proyecto se centró en abordar el

control de la demanda, probándose con 12 demostraciones mediante la implementación de la RSU y OBU [14].

Posteriormente, ya en el siglo XXI, surgió la necesidad de crear estructuras más complejas y eficientes. Para ello, la prioridad fue desarrollar sistemas ITS cooperativos, usando la banda de frecuencias 5GHz, como dictamina el estándar ETSI ITS G5 [15]. Este estándar está basado en comunicación WLAN (Wireless Local Area Network), centrado en disminuir el retardo ad-hoc de los datos en las transmisiones vehículo-vehículo y vehículo-infraestructuras. En esta línea, la empresa COMeSafety [16] implementó, entre los años 2006 y 2009, un nuevo algoritmo y especificación para el intercambio de información entre las estaciones ITS. Dicho proyecto estaba basado en el estándar 802.11p, definido como un estándar de comunicaciones de corto alcance o Dedicated short-range communications (DSRC), el cual no necesita autenticación para acceder a la red, y donde la banda de frecuencias utilizada esta delimitada entre 5875 y 5925 GHz [17] empleando los drivers ATH9K y cuyo funcionamiento esta explicado de manera más detallada en el proyecto de Javier Fernández Pastrana [8].

En la actualidad, el estándar ETSI ITS G5 sigue siendo el estándar utilizado en las comunicaciones V2X, aunque su desarrollo no ha sido completado. La capa de acceso sí que tiene totalmente definida su funcionamiento a través del estándar 802.11p, pero las capas superiores, y en especial la capa de *facilities,* no está estandarizada totalmente, ni en su funcionamiento ni en los estándares de los protocolos y servicios que la componen.

Sin embargo, con el auge de las comunicaciones móviles y del aumento del consumo de la tecnología móvil (Figura 4), y en particular con las grandes expectativas generadas por la quinta generación de tecnologías (5G Technology), aún sin estandarizar, cabe preguntarse si es conveniente seguir utilizando dicho estándar, o hay que empezar a evolucionar hacia una comunicación 5G a nivel de capa de acceso que sustituya al estándar 802.11p, o si por el contrario ambas tecnologías convivirán encargándose cada una de aplicaciones diferentes.

Por ello, ya se han realizado estudios y comparativas entre ambas tecnologías, viendo las ventajas y desventajas que tendría es uso de comunicaciones móviles en las comunicaciones intervehiculares. Algunas de las ventajas que tendría la tecnología 5G sobre la 802.11p se enumeran en la Tabla 3.

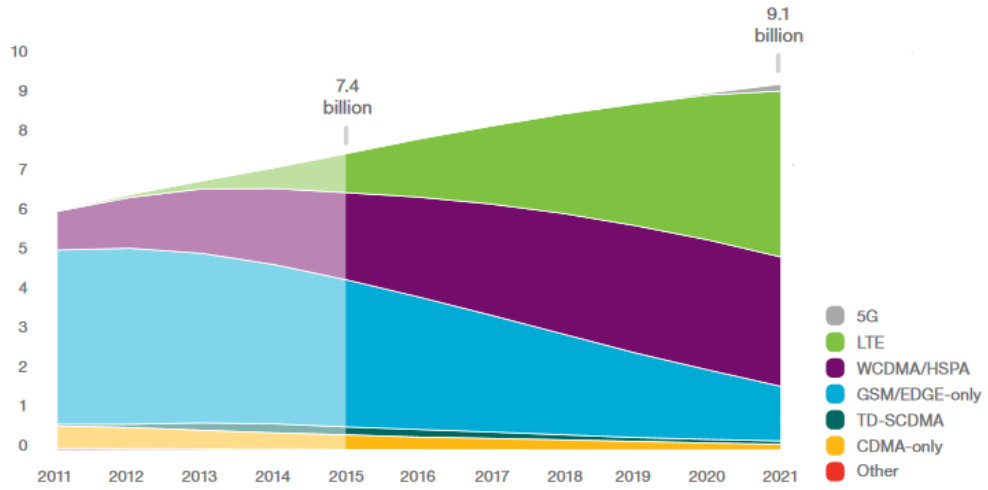

Figura 4. Suscripciones móviles por tecnología [23].

	Tecnología 5G	802.11p	Ventaja de 5G sobre 802.11p
Sincronización	Síncrona	Asíncrona	Aumenta la eficiencia espectral ya que la sincronización permite el uso de multiplexación por división en tiempo (TDM).
Retransmisión	Utiliza la solución de repetición automática híbrida (HARD)	No utiliza HARD	Proporciona un funcionamiento más fiable para un mismo rango de envío.
Forma de onda de transmisión	SC-FDM	OFDM	Permite la una mayor potencia de transmisión para un mismo amplificador de potencia
Tipo de tecnología en la comunicación	Misma tecnología desde el inicio al final de la comunicación	La tecnología 802.11p solo es utilizada a nivel de acceso	Mayor escalabilidad al abordar todas las aplicaciones V2X con una comunicación extremo a extremo
Uso	No sólo para comunicaciones V2X, también móviles e Internet	Desarrollado únicamente para el ámbito de las comunicaciones V2X	Mayor escalabilidad y desarrollo más ampliado
Facilities	Posibilidad de explotar toda la infraestructura móvil existente para comunicarse a través de un medio inalámbrico	Sistema específico para la comunicación, donde las estaciones ITS se conectan a través de una red troncal que proporciona acceso a Internet.	No habría necesidad de realizar obras civiles, ya que se aprovecharían los medios existentes.

Tabla 3. Comparación entre la tecnología 5G y la tecnología 802.11p en las comunicaciones intervehiculares.

Debido al gran número de ventajas que tendría la tecnología 5G, la asociación automovilística del 5G (5GAA), compuesta por un grupo de telecomunicación y la industria automovilística, establecieron el 27 de Septiembre de 2016 [18] un acuerdo para el desarrollo, testeo y producción sistemas 5G para comunicaciones entre vehículos (C-V2x) y ya han empezado a trabajar en la definición de los estándares específicos. Aunque a día de hoy la estandarización del 5G está todavía en el aire y está planeada su publicación para septiembre de 2018 [19], sí que se tiene percepción de que su uso en las comunicaciones relacionadas con los ITS será muy extendido y que proporcionará capacidades de transmisión superiores a las establecidas por la tecnología 802.11p.

Parece claro que la tecnología utilizada para las comunicaciones V2X será la que utiliza 802.11p en la capa de acceso, debido tanto a su amplio conocimiento y a que su funcionamiento ha sido probado en situaciones reales como a las ventajas actuales que tiene sobre las tecnologías móviles, como que los mensajes van directamente al destino final sin necesidad de atravesar canales de subida o bajada o que el coste es inferior al no tener que establecer acuerdos con operadoras móviles.

Pero también existe la necesidad de llegar a conseguir una compatibilidad más amplia entre ambas tecnologías, ya que la tecnología 5G podría coexistir con la 802.11p sin necesidad de desplazarla y de que esta quedase inutilizada, creando una red intervehicular heterogénea que mezcle lo mejor de ambas tecnologías.

Por lo tanto, se podría decir que es importante seguir desarrollando e invirtiendo en 802.11p, tal y como se ha tenido lugar en la realización del presente proyecto mediante el uso del software indicado en la sección 2.2, ya que es una tecnología probada y cuyo uso es inmediato, pero sin olvidarnos de que la irrupción del 5G permitirá mejorar muchos de los aspectos de esta tecnología, permitiendo una mejora tanto en comunicaciones móviles como en seguridad vial.

2.2 Comparativa de los proyectos de código abierto

Los proyectos de código abierto brindan una gran oportunidad para entender el estándar que implementa y poder realizar modificaciones sobre ellos en base al objetivo final del desarrollador. En nuestro caso, la razón de utilizar uno de estos proyectos de código abierto fue la de obtener la implementación de la pila de protocolos del estándar ETSI ITS G5. Como no se pudo encontrar ninguno que implementase todas las capas de la pila, se

sopesaron las ventajas e inconvenientes de los proyectos de código abierto que ya implementaban parte del estándar y en base a los objetivos que se deseaban alcanzar con este trabajo fin de grado se tomó una decisión. Los proyectos más relevantes que se encontraron fueron:

Proyecto Github Vanetza [20]: proyecto implementado a través del lenguaje C++ desarrollado por un grupo de investigación en el instituto de tecnología de Ingolstadt. Cubre la implementación de los protocolos de GeoNetworking y BTP (Basic Transport Protocol) en la capa de red, el protocolo DCC (Descentralized Congestion Control) y algunos mecanismos de la capa de *facilities*, seguridad y administración.

Proyecto Github Geonetworking [21]: proyecto realizado mediante Java y desarrollado dentro del i-Game Project [22] para la competición GCDC. Incluye la implementación básica de los protocolos GeoNetworknig y BTP en lo que se refiere a la capa de red así como los protocolos CAM (Cooperative Awareness Message) y DENM (Descentralized Environmental Notification Message) en la capa de *facilities*. Además utiliza el mecanismo ASN1 (Abstract Syntax Notation One) para codificar y decodificar los diferentes paquetes CAM y DENM enviados en la capa de facilities.

Proyecto OpenC2X [9]: proyecto desarrollo a través del lenguaje C++ por el grupo de investigación de sistemas distribuidos embebidos (CCS Labs) de la Universidad de Paderborn. Dicho proyecto contiene la implementación de las capas de red y transporte (a través de los protocolos GPS, OBDII y DCC), la capa de *facilities* (mediante los protocolos DENM, CAM y LDM (Local Dynamic Map)) y la capa de aplicación (a través de una aplicación web).

ETSI	Estándar	Proyecto OpenC2X	Proyecto GeoNetworking	Proyecto Vanetza
DCC	102 687	✓	×	✓
CAM	302 367-2	✓	✓	×
DENM	302 367-3	✓	✓	×
LDM	302 895	✓	×	×
GeoNet/BTP	102 636-4-2/-5-1	×	✓	✓
Seguridad		×	×	✓
Gestión		✓	✓	✓
GPS		✓	✓	✓
OBDII		✓	×	×

Tabla 4: Comparación entre los proyectos de código abierto [28]

Dado que uno de los objetivos del trabajo fin de grado era conseguir entender las funcionalidades de una RSU en las comunicaciones V2X a través de la implementación de un primer prototipo, se tomó la decisión de elegir el proyecto OpenC2X para llevar a cabo dichas tareas.

Además, una de las principales fortalezas del proyecto OpenC2X sobre los demás es que en dicho proyecto están implementados con bastante detalle tanto los servicios principales de la capa de facilities como el DCC, permitiéndonos analizar en profundidad las relaciones entre estos servicios, en contraposición a los otros proyectos, donde no todos los servicios de la capa de facilities están implementados. Asimismo, la capa de acceso de la pila de protocolos del proyecto OpenC2X puede realizar la transmisión de datos no solo a través de una interfaz cableada (Ethernet) o inalámbrica (Wi-Fi), sino también a través de una interfaz OCB-WLAN, configurada a través del estándar 802.11p, siguiendo los pasos marcados en el proyecto de Javier Fernández Pastrana [8].

Por otro lado, la principal carencia del proyecto OpenC2X es que no implementa el servicio de GeoNetworking [24], protocolo de la capa de red que actúa como refuerzo en la comunicación entre las estaciones ITS y en la distribución de los mensajes en las áreas geográficas.

No obstante, en un futuro, se podrían llevar a cabo la realización cambios en dicho proyecto para incluir aquellos servicios que no contiene, y que los proyectos descartados si lo hacen, con el fin de completar la pila de protocolos del estándar ETSI ITS G5.

3

Estudio del proyecto OpenC2X

3.1 Instalación del software

Tal y como se ha argumentado en la sección 2.2, para la realización de este trabajo fin de grado se llevó a cabo una comparación entre varios proyectos de código abierto y consecuentemente se eligió para la realización del mismo el software procedente de la Universidad de Paderborn.

Dicho software fue instalado bajo el sistema operativo Linux Ubuntu [26], en la versión de kernel 4.2.8 [27], ya que fue en este kernel bajo el que se hicieron las modificaciones en los drivers necesarias para conseguir utilizar a nivel de enlace el estándar 802.11p. Dentro de dicho kernel, se realizaron modificaciones para conseguir que el proyecto OpenC2X utilizado tuviese el funcionamiento adecuado y poder así realizar simulaciones y recoger datos para su análisis. Dichas modificaciones fueron:

- Instalación de dependencias software necesarias para poder utilizar programas de usuario y funcionalidades requeridas para el testeo del proyecto. Las dependencias instaladas fueron las indicadas en la Tabla 5, aunque no todas son indispensables para que el software funcione correctamente.

- Generar el .asn necesario para la compilación del proyecto. Este paso se realiza de la siguiente manera:

```
$ cd /path_to_openc2x/OpenC2X/common/asn1/
$ ./generate.sh
```

Dependencias	Uso
libzmq3-dev	Librería de mensajería entre capas.
libboost-all-dev	Proporciona el entorno completo de desarrollo Boost.
protobuf-compiler libprotobuf-dev	Mecanismos para la serialización de estructuras de datos.
libgps-dev gpsd gpsd-clients	Proporcionan los ficheros y funcionalidades necesarias para el uso de sistemas de posicionamiento global (GPS).
libnl-3-200 libnl-3-dev libnl-genl-3-200 libnl-genl-3-dev	Dependencias adecuadas para poder utilizar el programa IW y configurar el modo OCB.
sqlite3 libsqlite3-dev	Librería utilizada por el programa sqlite3 (Base de datos.)
tmux	Comando que permite dividir un terminal en múltiples terminales.
asn1c	Compilador de C++.
build-essential	Librerías comunes para compilación.
cmake	Sistema de compilación de código abierto y multiplataforma.
doxygen	Útil para la generación de documentación desde ficheros de código fuente.

Tabla 5: Dependencias utilizadas en la instalación del software

- Aplicar los parches necesarios para que el mecanismo DCC pueda obtener la carga del canal en función del tráfico de datos enviado entre las distintas estaciones ITS. Estos parches son el 0001-Enable-queueing-in-all-4-ACs-BE-BK-VI-VO y el 0002-Get-hw-queue-pending-stats-from-ath9k-via-netlink. En este punto surgió un problema que no fue capaz de resolverse, y es que no fuimos capaces de medir la carga del canal cuando se utilizaba la interfaz OCB (Outside the Context of a BSS) para la comunicación, empleando tecnología 802.11p. Para obtener la carga del canal, la comunicación entre ambas estaciones ITS (simuladas) debía realizarse a través de la interfaz Wi-Fi o Ethernet.

- Por último, tras realizar la compilación, ya solo quedaría lanzar el proyecto en una terminal, y en ese momento se empezaría la comunicación interactiva entre los diferentes módulos.

3.2 Pila de protocolos del proyecto OpenC2X

Uno de los principales objetivos del proyecto fue buscar un proyecto de código de abierto que reuniese las características necesarias para proporcionarnos una visión global de la arquitectura del estándar ETSI ITS G5 así como de su funcionamiento. Como se especificó anteriormente, se eligió el proyecto desarrollado por el grupo de investigación del Heinz Nixdorf Institut, CCS Labs [9], debido a que ofrecía un mayor número de mecanismos y herramientas para el propósito que nos planteamos. Una de las herramientas fundamentales que nos brindaba era la implementación de una pila de protocolos muy completa orientada la comunicación inalámbrica en los ITS. Como se puede ver en la Figura 5 la arquitectura llevada a cabo por OpenC2X para el estándar ETSI ITS G5 reúne los principales protocolos de las diferentes capas, y además, a nivel de red y transporte ofrece diversas alternativas para la comunicación. En esta figura, los módulos que aparecen punteados son los que no están desarrollados, y los cuales se podrían añadir en líneas futuras del proyecto.

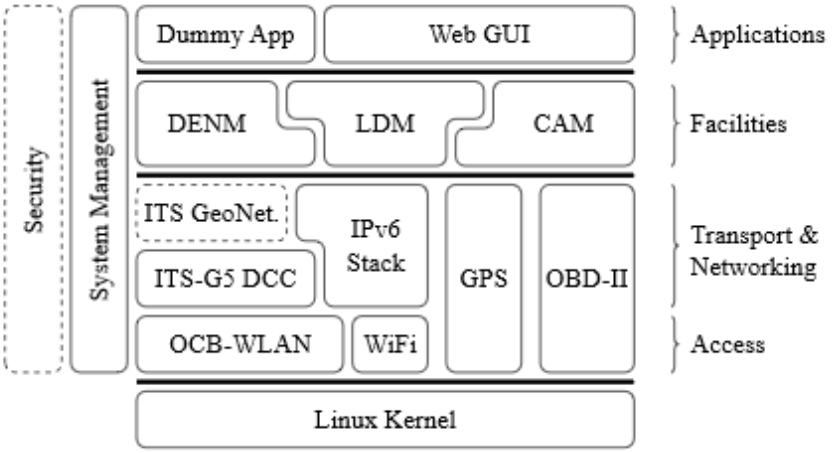

Figura 5. Pila de protocolos del estándar ETSI ITS G5 para el proyecto OpenC2X.
En punteado aparecen los módulos que todavía no están implementados.

En un análisis detallado de las distintas capas, en primer lugar, la pila de protocolos del OpenC2X permite tanto el uso de Wi-Fi, como Ethernet y OCB-WLAN (802.11p) para la comunicación en la capa de acceso. En nuestro caso, dado que el software necesario para la implementación del estándar 802.11p fue desarrollado en el proyecto de Javier Fernández

[8], podemos trabajar con cualquiera de las tres tecnologías. Se puede ver que en el proyecto OpenC2X se habilita un servicio de GPS que no está presente en la arquitectura ETSI ITS G5 [29] como se puede ver en la Figura 6. En realidad el servicio de Geonetworking del estándar requiere de otro servicio que le le proporcione la posición del vehículo. En C2X Geonetworking no ha sido todavía desarrollado, sin embargo ya se incorpora el servicio de GPS, y se añade dentro de la arquitectura, sin que este hecho a nuestro modo de ver repercuta de forma esencial en la compatibilidad con el estándar.

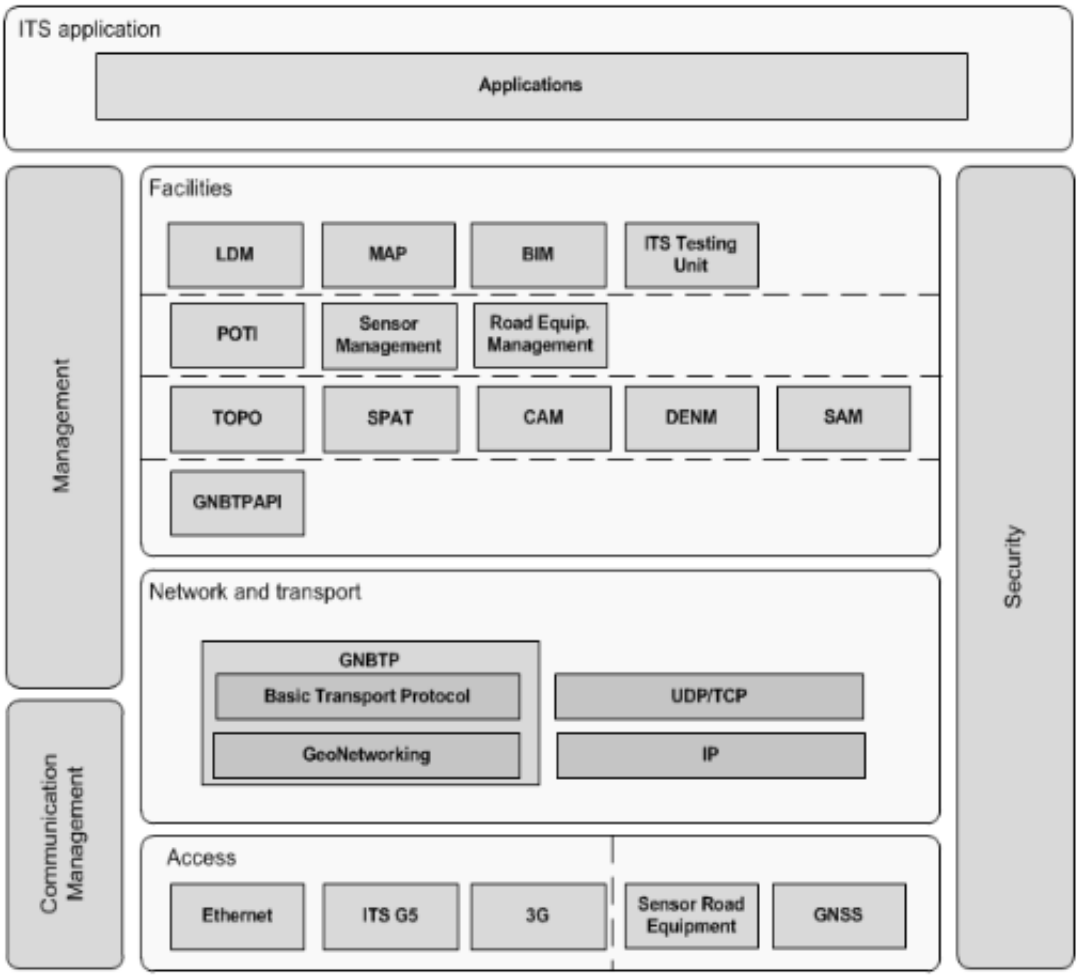

Figura 6 Pila de protocolos del estándar ETSI ITS G5 para una RSU

En lo que se refiere a la capa de *facilities* cuyo estudio es uno de los puntos principales de este proyecto, en la arquitectura general del estándar ETSI ITS G5 vemos que existe una gran variedad de servicios o protocolos. Muchos de estos protocolos tienen a su vez interfaces que permiten la transferencia de datos entre las capas contiguas, como puede ser el caso de POTI (gestión de posición) para el protocolo CAM, explicado en la sección 3.2.3.

En cuanto a nuestro interés, en la arquitectura desarrollada para el proyecto OpenC2X, solo están implementados los servicios CA, DEN y el protocolo LDM en esta capa.

El servicio CA es el encargado de gestionar el protocolo CAM, proporcionando a su vez el servicio de generar, recibir y procesar CAM. Por su parte el servicio DEN administra el protocolo DENM, encargándose de proporcionar los servicios necesarios a las aplicaciones ITS para provocar, actualizar y terminar la transmisión de un DENM. Tanto el servicio CA como el DEN utilizarán los servicios suministrados por las capas de red y transporte para llevar a cabo las funciones anteriormente explicadas. Por último el protocolo LDM es empleado para almacenar dicha información y así tenerla disponible en caso de que sea necesaria.

El resto de protocolos de esta capa tiene una funcionalidad que para nosotros no es demasiado relevante a la hora de realizar un estudio general del estándar ETSI ITS G5, aunque su implementación pueda llevarse a cabo en un futuro.

La capa vertical de gestión (Management) está formalizada mediante el protocolo DCC, que se encarga de gestión de las otras capas y proporciona eficiencia en el rendimiento mediante la asignación de recursos de manera equitativa. Por último, el proyecto OpenC2X consta de una interfaz web asociada correspondiente a la capa de aplicación y sincronizada con el proyecto, la cual nos permite monitorizar los datos y proceder al análisis de los mismos de una manera más intuitiva que en la terminal de Linux.

3.2.1 OBD-II

Este protocolo está definido en el estándar ISO 9141-2 [31], publicado en 1996 y cuyo cometido es el de obtener datos concretos del automóvil que está siendo sometido a estudio. Se desarrolló en California y en un principio fue utilizado para que los servicios públicos pudieran determinar cuánto contaminaba un vehículo y posteriormente, con la evolución de la electrónica en los vehículos, fue empleado entre otras cuestiones para poder realizar una telemetría a la vez que se extrae información de diagnóstico (Diagnostic Trouble Codes, DTC) en el caso de que el vehículo tenga algún fallo, consiguiendo de esta manera que el usuario tenga un control total sobre el vehículo. De esta manera, permitiría obtener información sobre el estilo de conducción, para que un usuario pudiese corregir sus malos hábitos en este aspecto y reducir tanto el impacto medioambiental del vehículo como conseguir un ahorro de combustible [25].

En lo que concierne al proyecto OpenC2X, el objetivo que se persiguió al utilizar dicho es estándar fue acceder a la unidad de control electrónico (Electronic Control Unit, ECU) del vehículo, focalizando el trabajo en obtener los parámetros de velocidad y revoluciones por minuto, aunque este último parámetro solo sería informativo, no se utilizará en el envío de mensajes entre los distintos módulos de la pila de protocolos del ETSI ITS G5. Por otro lado, la velocidad del vehículo puede obtenerse tanto de forma simulada como de forma real. Cuando la velocidad esté obtenida, se enviará a los diferentes módulos periódicamente para que estos la utilicen, como por ejemplo se explica en la sección 3.2.3 con el protocolo CAM, registrando la diferencia de velocidades entre dos mensajes consecutivos para ver si ha tenido lugar algún incidente. Esta sección referente al OBDII está ampliada en el trabajo fin de grado de Pilar Sánchez Martín [30].

3.2.2 GPS

Este servicio es utilizado en nuestro proyecto para recoger los datos que nos permitirán establecer la localización del vehículo.

Con el objetivo de poder analizar las pruebas realizadas con datos obtenidos de diferentes maneras y así testear el funcionamiento del software en diferentes escenarios, en este proyecto OpenC2X existen varias posibilidades en cuanto a la obtención de estos datos GPS, ya que a través de su fichero de configuración (config.xml) se pueden configurar los siguientes parámetros[1]:

- *SimulatedData*: si se encuentra con valor *true* los datos GPS serán simulados y si su valor es *false* los datos serán reales.
- *SimulationMode*: en el caso de que los datos GPS sean simulados, si el valor de este parámetro es 1 los datos se obtienen de un fichero y si es 0 se simulan mediante distribuciones aleatorias.
- *DataFile*: si queremos obtener un fichero con extensión .csv los datos GPS simulados, en este parámetro habrá que indicar la ruta de dicho fichero.

[1] Los valores que se le dan a los parámetros de configuración no tienen porque coincidir con el valor de estos en el fichero gpsService.h. Siempre se utilizará el valor proporcionado a los parámetros en el fichero config.xml.

Una vez que el modelo de la simulación se ha configurado, entonces comienza la ejecución del script. Inicialmente, se obtienen y se serializan[2] los datos GPS, ya sean simulados (a través de las distribuciones aleatorias, especificadas en el anexo 1, o del fichero .csv) o reales, y posteriormente estos datos serializados se envían a los distintos servicios indicando el tipo de dato (GPS) y los datos del mensaje para su uso y almacenamiento.

También esta implementada la opción de recibir datos GPS desde otro dispositivo. Para ello, se recogerán los datos GPS tomados a través de un bucle, que está continuamente esperando datos, y se comparará el dato recién llegado con el anterior para ver si este es válido, es decir, la diferencia de tiempo de llegada entre ellos es menor de un segundo. Si los datos recibidos son válidos, se almacenarán en un buffer y se enviarán a los servicios CA y DEN.

3.2.3 CAM

Este protocolo está definido en el estándar ETSI 302 637-2 [32] y su función es la de permitir el intercambio de mensajes entre las estaciones ITS (ya sean vehículos móviles de cualquier tipo o RSU) con el objetivo de conocer en cada momento el comportamiento de los vehículos que están utilizando la vía y determinar en qué estado de tráfico se encuentra dicha vía. La comunicación puede ser entre vehículos móviles (V2V) o entre vehículo e infraestructura (V2I) y dependiendo del tipo de estación que genere y envíe el CAM, el formato de este será diferente, pues los campos de las tramas CAM serán diferentes, tal y como puede verse en la Figura 7.

Los campos de la cabecera ITS (ITS PDU header), el contenedor básico (Basic Container) y el contener de alta frecuenta (HF Container) siempre estarán presentes en cualquier CAM enviado por todo tipo de estación ITS. Sin embargo, tal y como se especifica en el anexo B del ETSI 302 637-2 [32], el contenido tanto del contenedor de alta frecuencia como del contenedor de baja frecuencia (LF Container), cuyo uso no es obligatorio en todas los CAM, dependerán del tipo de estación ITS que envíe el CAM. Por último, otro campo opcional será el contenedor de vehículo especial (Special vehicle Container), cuyo contenido,

[2] La serialización de los datos consiste en un proceso mediante el cual una estructura de datos es codificada de una determinada manera para su posterior almacenamiento, en nuestro caso, en un buffer de memoria. Esta serialización se suele llevar a cabo mediante el uso de instancias de objetos. La explicación de cómo se realiza esta serialización, así como aspectos relevantes del software, como el uso de temporizadores y buffers está explicado más detalladamente en el proyecto de Pilar Sánchez Martín [30].

explicado en la sección 7.4 del estándar ETSI 302 637-2, dependerá del tipo de vehículo utilizado. Todos estos campos serán explicados más adelante dentro de esta sección.

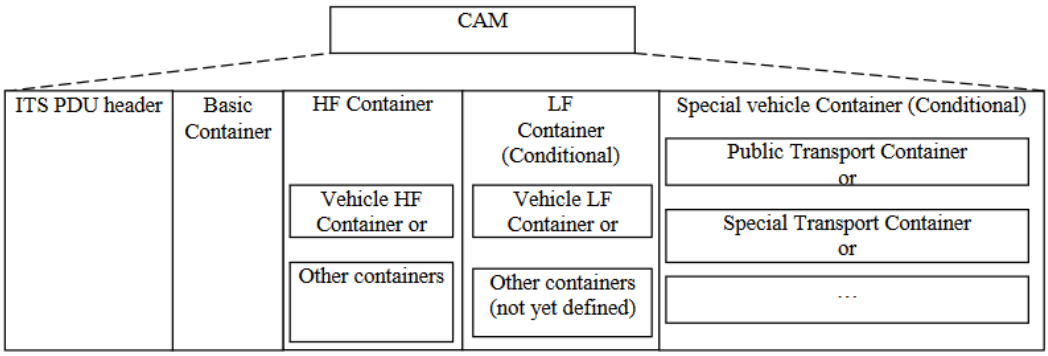

Figura 7. Estructura general de un CAM

Además, en el caso de los CAM enviados por los vehículos, estos pueden ser enviados por diferentes motivos, como se explicará a continuación.

Los mensajes CAM son generados y enviados por el propio protocolo CAM. Estos mensajes son enviados periódicamente por cada vehículo a todas las estaciones que se encuentren dentro de su rango de comunicación, ya sean vehículos o RSU.

Como dijimos en la sección 3.2.3, el CA Basic service es el que se encarga de gestionar el protocolo CAM y para ello hace uso de un conjunto de interfaces y entidades ITS para la comunicación con el resto de capas y servicios de la arquitectura ETSI ITS 5G, tal y como puede observarse en la Figura 8. A través de las interfaces FA-SAP, MF-SAP, SF-SAP y NF-SAP, definidas en el estándar ETSI 302 637-2 [32], realiza el intercambio de datos con el resto de servicios de la capas con las que tiene contacto y mediante las entidades de proveedor de datos de vehículos (VDP) y gestión de posición y tiempo (POTI) se recogerán los datos de las estaciones ITS que posteriormente se almacenarán en la base de datos del LDM.

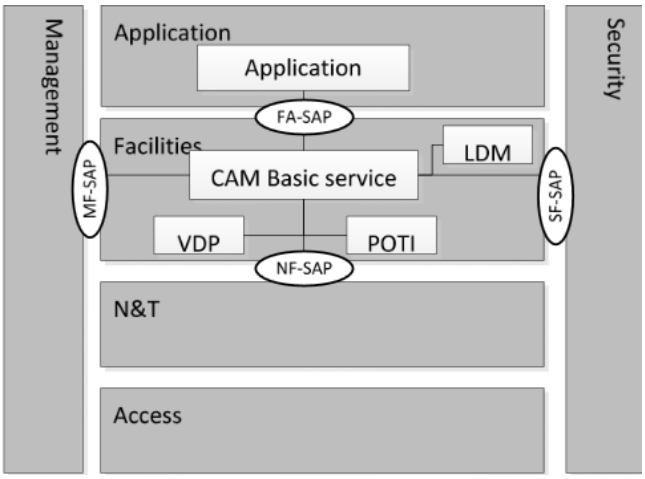

Figura 8. Disposición del servicio CA en la arquitectura del estándar ETSI ITS G5.

En lo que respecta al proyecto OpenC2X, el protocolo CAM tiene asociado un fichero de configuración, donde se pueden configurar los siguientes parámetros:

- *GenerateMsgs*: si este parámetro se encuentra a *true*, se generarán periódicamente mensajes llamando a la función alarm(), que es la encargada de enviar dichos mensajes en función de la causa por la cual se generaron.
- *ExpirationTime*: utilizado por la función send() para indicar que los datos CAM referentes a un mensaje solo serán válidos desde el *currentTime* hasta *currentTime* + *expirationTime* (en el estándar está definido como unsegundo).
- *MaxGpsAge* y *MaxObd2Age*: utilizados para indicar el máximo tiempo de validez de los datos GPS y OBDII respectivamente.
- *ThresholdRadiusForHeading*: indica la distancia en metros mínima que un vehículo se debe haber desplazado respecto de la posición en la que envió el último CAM para considerar realizar el cálculo del cambio de rumbo (heading), que es una de las condiciones por las que se puede generar un CAM y que se explicará a continuación. Esta diferencia en la posición se registra mediante la latitud y longitud.
- *isRSU*: parámetro que sirve para indicar si el emisor del mensaje CAM es una RSU(1) o un vehículo(0). Es necesario diferenciar entre ambos porque, como se especificó antes, la información que se inserta en los mensajes es diferente.

Como hemos apuntado anteriormente, los mensajes CAM son enviados a los módulos LDM y DCC periódicamente, en intervalos no menores de 100 ms, y pueden ser lanzados por diferentes motivos. En primer lugar, un mensaje CAM se enviará si la

diferencia de tiempo con el último mensaje CAM es igual o mayor a 1 segundo. En ese caso, se indicaría que el motivo por el cual se envió el CAM fue por sobrepasar el tiempo de un segundo desde la generación del CAM anterior y también la diferencia de tiempo entre las dos tiempos registrados a las que fueron generados dichos mensajes, para posteriormente generarse el mensaje CAM y enviarse.

Otros motivos por los que un vehículo enviará un mensaje CAM son:

- Cuando el tiempo desde el último mensaje CAM generado es mayor o igual que T_GenCam_Dcc3 y además se dan las condiciones dinámicas siguientes:

  - La diferencia en valor absoluto de posición entre los dos últimos mensajes CAM es mayor que 5 m, aunque en el estándar este valor está marcado a 4 m.
  - La diferencia de *heading* entre los dos últimos CAM es mayor de 4º, tal y como se especifica en el estándar. El parámetro *heading* marca, en grados, la dirección que sigue el vehículo con respecto al punto cardinal norte. Por lo tanto, si, por ejemplo, en un vehículo se desvía del norte 3º y la siguiente vez que se genera un CAM, el vehículo se ha desviado del norte otros 10º, entonces se generaría un CAM debido a que la diferencia de dirección respecto al norte sería de 6º)
  - La diferencia en valor absoluto de velocidad entre los dos últimos mensajes CAM es mayor de 1m/s (el estándar dictamina una diferencia de velocidad de 0.5m/s para enviar el mensaje).

- El tiempo desde que se generó el último mensaje CAM es mayor o igual que T_GenCam_Dcc3 y mayor o igual que T_GenCam3. Además, cabe destacar que en el estándar la generación de los mensajes CAM depende también del estado de congestión del canal y esto es controlado a través de la variable temporal T_GenCam_Dcc, que debería estar siempre entre el valor de 100 ms y 1000 ms, con el objetivo de reducir la generación de CAM y así cumplir los requerimientos del canal marcados por el DCC.

Estas dos últimas condiciones explicadas en el párrafo anterior si que vienen indicadas en el estándar que define el protocolo CAM, pero en nuestro proyecto no están

implementadas y únicamente se define que el intervalo de tiempo entre la generación de dos CAM consecutivos debe estar entre 100 ms y 1000 ms y en lo que respecta a la congestión y a la generación de los CAM, en el software del proyecto OpenC2X solo establece que en caso de congestión del canal y envío simultáneo de CAM y DENM, los DENM tendrían mayor prioridad y se enviarían antes que los CAM.

Todas las condiciones anteriores se comprueban a través de las funciones del proyecto indicadas en el anexo 2, y cuando alguna de las condiciones se cumpla provocará que se genere un mensaje CAM en el que se indicará cuál de estas condiciones es la responsable de la generación y envío de dicho mensaje CAM.

Al mensaje CAM se le asigna el tamaño dinámicamente. Posteriormente, se rellenan los campos del mensaje especificados en el estándar (ver Figura 7), que son la ITS PDU Header, donde se incluyen la versión del protocolo utilizado, el identificador de la estación ITS y el identificador del CAM; el contenedor básico, donde se indica el tipo de estación ITS y su última posición geográfica; y los contenedores que contendrán los datos de estado dinámicos (alta frecuencia), como la velocidad y *heading* , y estáticos (baja frecuencia), como el estado de las luces del vehículo, comprobando previamente si los datos tanto de GPS como los proporcionados por el OBDII son válidos. Además se generará una marca de tiempo, para registrar en el momento en el cual se ha generado el mensaje CAM y así ver la diferencia de tiempo con el CAM anterior y posterior.

Por último, se serializa el CAM generado y se envía al LDM y al DCC a través de la función entrada el tipo de mensaje que es (CAM en este caso) y el mensaje CAM serializado.

Este proyecto también ofrece la posibilidad de recibir mensajes CAM desde otras estaciones ITS. En este caso, se decodifica la información procedente del CAM enviado por el DCC, se serializa y se manda al LDM, mostrando un mensaje por pantalla que indica que un CAM ha sido recibido, tal y como puede verse al final de la sección 4.2.

3.2.4 DENM

En esta sección se va a describir el protocolo DENM definido en el estándar ETSI 302 637-3 [33]. Dicho protocolo es controlado a través del servicio DEN y se encarga principalmente del envío y gestión de mensajes DENM que son generados por las aplicaciones ITS dentro de las estaciones ITS para avisar de un evento de tráfico o medioambiental que tiene un impacto potencialmente peligroso en la seguridad vial (Road

Hazard Warning) al resto de estaciones ITS que se encuentran dentro del área de comunicación, que se define como el área geográfica en el cual la información correspondiente a un evento es identificada como relevante para su uso o distribución. Cada estación ITS procesará el DENM para extraer información y brindársela al usuario final (conductor) para que este pueda tomar la acción que considere más adecuada. La tabla en la que se especifican los eventos que tienen que ocurrir para que se lance un DENM viene detallada en el proyecto de Pilar Sánchez [30].

Si la estación que generó el DENM detecta una evolución o cambio en el evento enviado, el gestor del DEN construirá un nuevo DENM incluyendo la nueva información y notificará que se trata de una modificación de un evento anterior utilizando en el nuevo DENM generado el mismo identificador del mensaje DENM original que avisaba del evento inicial.

Por otro lado, la finalización del evento puede ser indicado de dos maneras posibles: mediante la transmisión de un DENM de cancelación enviado por la estación emisora del mensaje original cuando el evento ha finalizado o cuando una de las primeras estaciones que ha recibido el mensaje se da cuenta de que ese evento ya no existe. En este último caso enviará un DENM de negación marcado con una *flag* específica.

3.2.5 LDM

Este protocolo está definido en el estándar ETSI EN 302 895 [34]. Se define como una base de datos conceptual de información para las capas de aplicación y *facilities* que se sitúa dentro de la estación ITS y cuya función principal es la de almacenar información importante tanto para el funcionamiento de las aplicaciones ITS como para el estudio de la seguridad en carretera y eficiencia de tráfico.

El LDM está dividido principalmente en dos componentes, que se encuentran separados:

- LDM Service: encargado de proporcionar las funcionalidades necesarias a las aplicaciones o *facilities* autorizadas para proveer de datos al LDM (LDM Data Provider) para la manipulación de dichos datos y suministrar mecanismos de acceso a los datos para los consumidores de los mismos (LDM Data Consumer).

- LDM Maintenance: responsable del almacenamiento y mantenimiento de los datos así como de la limpieza de datos obsoletos dentro del LDM.

Los datos pueden ser recibidos desde cualquier tipo de aplicación dentro de una estación ITS y después de su almacenamiento se puede acceder a ellos a través de varias herramientas para analizarlos. En el estándar [34, viene especificado que tanto los LDM Data Consumer como los LDM Data Provider tienen asignada una prioridad a la hora de acceder a los datos. Este aspecto no se refleja en el proyecto OpenC2X, donde se atiende a la primera aplicación o *facility* que lanza una petición.

En el caso del proyecto de código abierto utilizado, la primera acción que se realiza es la de abrir la base de datos donde se almacenarán los mensajes que provienen de las distintas estaciones ITS y a continuación se crearán las tablas donde se almacenen los mensajes enviados por cada servicio si no han sido creadas anteriormente. Tanto la apertura de la base de datos, cuya apariencia puede observarse en la Figura 9, como la inserción y selección de los datos de cada una de las tablas es posible debido al uso de la librería sqlite [35], cuya instalación previa se realizó tal y como se indicó en la sección 3.1.

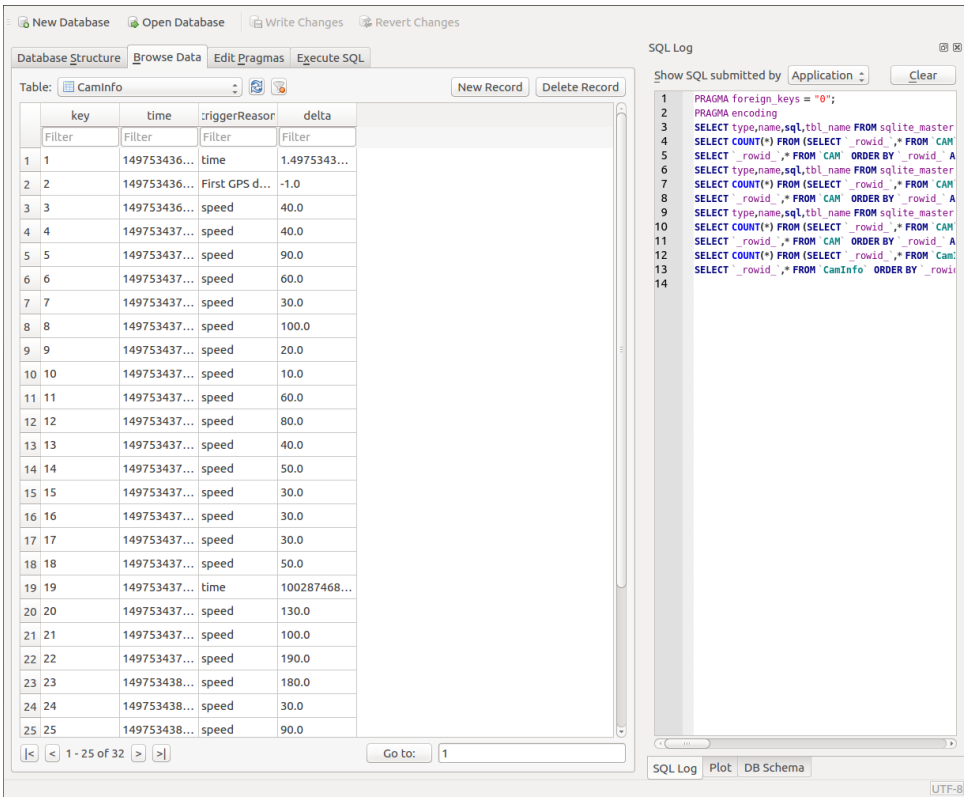

Figura 9. Apariencia de la base de datos asociada al proyecto.

En el estándar ETSI EN 302 895 [34] se establece que al menos se debe almacenar la información referente a los CAM y DENM enviados y recibidos. Sin embargo, dado que la programación de la base de datos es libre, en el proyecto OpenC2X se utilizarán tablas para almacenar información sobre los protocolos CAM y DENM, para almacenar la razón por la que ha generado un CAM (tabla CAMInfo) y para registrar información sobre el mecanismo de gestión DCC (tabla DCCInfo). Aunque también se crean las tablas referentes a los protocolos GPS y OBDII, no está realizada todavía la función que reciba los datos de GPS y OBDII insertándolos posteriormente en la base de datos. Sin embargo, esta funcionalidad sí que se realiza en el CAM y en el DENM, donde sí se reciben los datos de GPS y OBDII, mostrando los datos referentes a los dos módulos anteriores en cada mensaje CAM y DENM enviado.

Todas las tablas mencionadas anteriormente se rellenarán a través de las funciones `insert()` referentes a cada servicio, donde en función de una serie de condiciones, como el tipo de estación ITS que envió el mensaje, se rellenará el mensaje con un tipo de información proveniente de un contenedor específico. Los datos con los que se rellenan las tablas se actualizan cada vez que se ejecuta el proyecto y se lleva a cabo una simulación, sobrescribiéndose la base de datos de la simulación anterior.

Dentro de base de datos se podrán llevar a cabo peticiones, con el objetivo de poder seleccionar o filtrar una serie determinada de paquetes de acuerdo con las condiciones prefijadas en la petición. Esto puede ser muy útil para, por ejemplo, obtener solo información relevante a una serie de instantes de tiempo en los cuales la información ha cambiado sustancialmente o visualizar los mensajes enviados por una única ITS.

Otra ventaja del proyecto OpenC2X junto con el uso de sqlite, en lo referido al empleo de la base de datos, es la posibilidad de realizar gráficas de los parámetros de cada mensaje que consideremos más relevantes y poder tener de esta manera una visión más gráfica e intuitiva de los datos, tal y como se puede ver en la Figura 10, donde se representa en el eje de abscisas el identificador del mensaje y en el eje de ordenadas la velocidad que llevaba el vehículo (m/s * 100) cuando generó el mensaje.

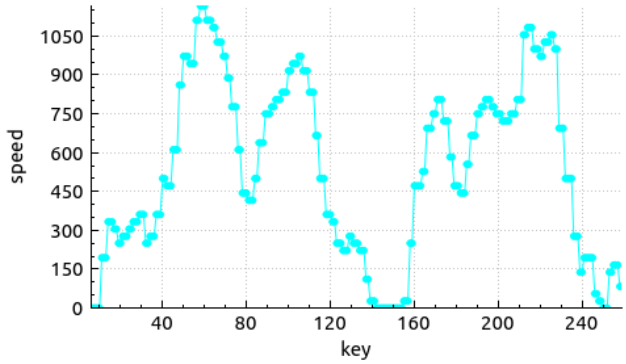

Figura 10. Ejemplo de gráfica (número mensaje – velocidad) en la base de datos.

3.2.6 DCC.

Para que la comunicación entre vehículos utilizando VANETs (Vehicular ad-hoc Networks) sea factible, se ha de asegurar que el DCC funcione correctamente. El DCC es un mecanismo basado en el estándar ETSI 102 687 [37] presente en varias capas de la arquitectura de una estación ITS y cuya función principal es la de mantener la estabilidad de la red ITS, a través de un gestión eficiente de los recursos localizados en cada estación ITS.

En este proyecto, el DCC se centra en la capa de acceso (DCC_access), donde se controlará tanto la potencia a la que se transmiten los mensajes entre las estaciones que se encuentran dentro del área concreta como la tasa de transmisión de dichos mensajes. Este mecanismo se basa en el conocimiento del canal de transmisión, y para obtener información sobre él se realizan sondeos, como por ejemplo sobre la intensidad de la señal recibida. De esta manera, el canal de transmisión se podrá encontrar en tres estados dependiendo de la ocupación del canal: activo (ocupación entre 15% y 40 %), relajado (ocupación menor del 15%) y restrictivo (ocupación mayor del 40%).

Por todo lo descrito anteriormente, el mecanismo DCC es la pieza clave y la más compleja de la arquitectura del estándar ETSI ITS G5, ya que necesita que muchos componentes de las otras capas de la pila de protocolos trabajen juntos para cumplir los requerimientos operacionales que este mecanismo debe de proporcionar. La explicación de estos requerimientos y una descripción exhaustiva del modo de operación del DCC a más bajo nivel se encuentra en el trabajo fin de grado de Pilar Sánchez [30].

4

Testeo del proyecto OpenC2X

Para terminar de tener una perfecta comprensión del software instalado, lo ideal es llevar a cabo el mayor número de simulaciones posibles para obtener una gran variedad de datos en diferentes entornos de simulación para poder afirmar si los resultados obtenidos en dichas simulaciones son los esperados o se ha producido algún evento inesperado, como puede ser un error de compilación al modificar el código del proyecto o alguna situación imprevista en la generación de los mensajes.

4.1 Entorno de simulación

Los pasos para la ejecución del proyecto OpenC2X resultan ser bastante sencillos y están claramente explicados en la documentación proporcionada por el grupo de investigación CCS-Labs.

En primer lugar, siempre que se realice un cambio en alguno de los ficheros de código fuente del proyecto, hay que realizar la compilación del mismo. Para ello habrá que utilizar los comandos *cmake* y *make all* dentro de la carpeta *build*. Seguidamente, se procederá a la ejecución del proyecto para realizar una simulación. En este caso, tendremos que situarnos en la carpeta donde se encuentran los *scripts* de lanzamiento y ejecutar los mismos (si estos *scripts* no tienen permiso de ejecución, habría que proporcionárselos):

```
$ cd path_to_openc2x/OpenC2X/scripts/
$ ./runOpenC2X.sh
```

El resultado de la ejecución del proyecto aparecerá en el terminal de Linux, donde se podrán observar los diferentes CAM y DENM enviados entre los módulos, la velocidad y posición de la estación ITS y el funcionamiento y gestión del DCC. El aspecto que presentará la terminal se puede ver en la Figura 11.

Figura 11. Entorno de simulación de el terminal de Linux.

Tal y como puede apreciarse en la Figura 11, aparecen siete pantallas que dividen el terminal. En dichas divisiones aparecen los CAM tanto enviados como recibidos gestionados por el servicio CA, la información enviada entre la plataforma web y el protocolo LDM, un mensaje informando de si se ha accedido correctamente o no a la base de datos, los DENM enviados y recibidos gestionados por el servicio DE, la información que el protocolo DCC necesita para controlar el canal de transmisión así como los mensajes que tiene en cola, la velocidad proporcionada por el OBDII y los parámetro de latitud, longitud y altitud que proporciona el GPS.

4.2 Plataforma web

Aparte del entorno de simulación que se ha mostrado en la sección 4.1, dicho proyecto tiene una plataforma web local asociada y sincronizada al mismo, que permite analizar de una manera visual y muy intuitiva los mensajes intercambiados por los diferentes módulos que conforman la pila de protocolos del estándar. La apariencia de la plataforma web es la mostrada en la Figura 12.

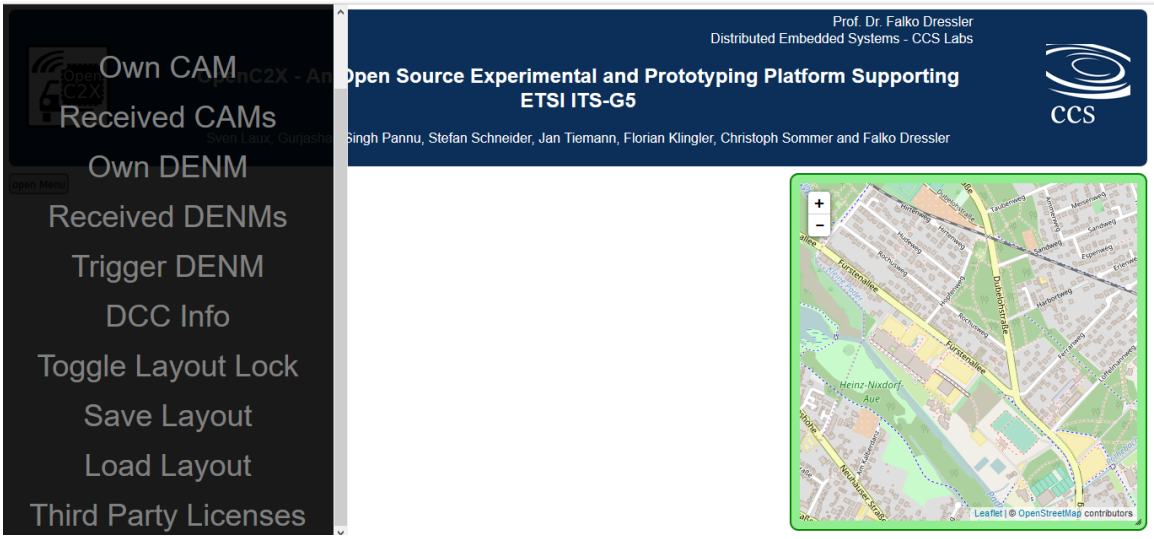

Figura 12. Apariencia de la plataforma web local.

Tal y como se puede apreciar en la imagen anterior, la plataforma web consta de un menú en el cual se puede seleccionar la información referente a la transmisión de datos entre estaciones ITS. También se pueden visualizar las características de los CAM enviados y recibidos, las características de los DENM enviados y recibidos, además de poder lanzarlos desde la plataforma web e información referente al DCC. Además, mediante las opciones de *Layout* se controla la apariencia de la página.

Otro de los aspectos que hacen que la plataforma web sea útil y visual es que consta de un mapa en el cual se pueden localizar los vehículos que están transmitiendo datos dentro del rango de comunicación así como sus movimientos según van cambiando de posición. En la Figura 13 se muestra el aspecto de la plataforma web cuando se está realizando la transmisión entre dos vehículos (con IDs 1994 y 1995), cuando la comunicación es bidireccional y todos los datos están siendo simulados por el ordenador.

También se pueden observar los mensajes enviados y recibidos por ambas estaciones en el entorno de simulación de la terminal de Linux. Los mensajes recibidos procedentes de otra estación aparecerán tanto en el caso del servicio DEN como el CA como "forward incoming", tal y como puede verse en la Figura 14, donde se pueden apreciar CAM y DENM procedentes de la estación ITS con ID 1995 a la estación con ID 1994.

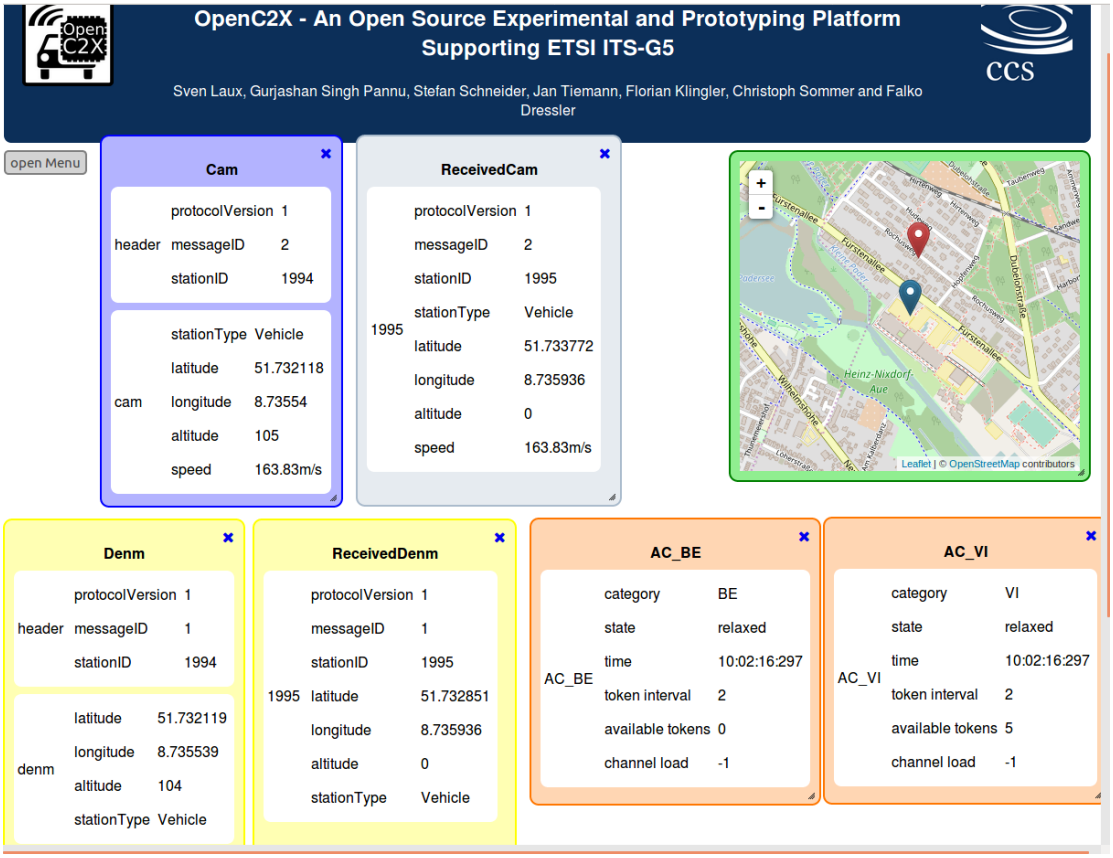

Figura 13. Transmisión de datos bidireccional entre dos vehículos

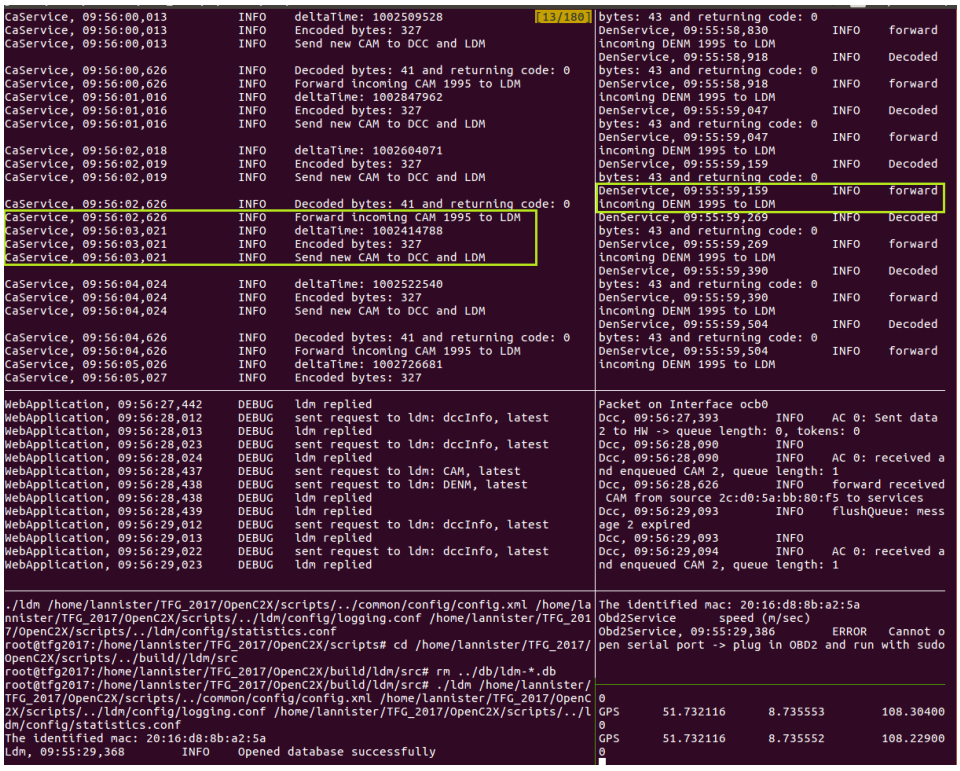

Figura 14. Ejemplo de mensajes entrantes procedentes de otra estación ITS.

4.3 Pruebas realizadas.

Como ya se argumentó al inicio de la memoria, este proyecto está más focalizado a ver el comportamiento del estándar ETSI ITS G5 en la capa de facilities, es decir, el comportamiento y funcionamiento de los servicios CA, DEN y del mecanismo DCC, ya que para el proyecto OpenC2X utilizado se empezó teniendo como base el trabajo final de máster realizado por Javier Fernández sobre el estándar 802.11p [8], donde ya se realizan pruebas en las capas inferiores de la arquitectura ETSI ITS G5. En esta memoria se incluirán las pruebas realizadas para el servicio CA, y las pruebas referentes al servicio DEN y al mecanismo DCC están incluidas y explicadas en la memoria realizada por Pilar Sánchez [30].

4.3.1 Pruebas CAM

Los CAM deberían ser generados y enviados cuando se producen las situaciones descritas en la sección 3.2.3 de la memoria. Para comprobarlo, se realizaron diferentes simulaciones en la que los datos generados y los ficheros de configuración de los módulos fuesen diferentes.

Para ello primero realizemos una descripción de los casos de estudio que se realizarón y después presentaremos los resultados que se obtuvieron al llevar a cabo las simulaciones.

En el primer caso de estudio, al que nos referiremos a partir de ahora como caso 1, se pretendía demostrar que los CAM son generados y enviados por el protocolo CAM a los otros módulos debido a una variación brusca de velocidad o posición y en el segundo caso de estudio, al que nos referiremos como caso 2, el objetivo era comprobar que en situaciones en las que el vehículo se encuentra estático o bien se está moviendo muy lento, es decir, los CAM no serán generados por variaciones en la velocidad o posición, se producirá el vencimiento de un temporizador y entonces se generará un CAM al haber pasado más de un segundo respecto al anterior mensaje.

Por un lado, para el caso 1, la velocidad fue generada por medio de variables aleatorias, tal y como está explicado en el anexo 1, y la posición fue leída de un archivo de datos con extensión .csv donde están almacenadas la latitud, longitud y altitud de la estación ITS en columnas. Las variaciones en ambas variables son muy bruscas en magnitud, asegurando así que la condición para que se genere el cambio se produzca con total seguridad. En un entorno real, la posición no variaría tan rápido aunque si lo puede hacer la

velocidad en caso de que tenga lugar un accidente. El cálculo de la distancia entre vehículos a partir de los parámetros de longitud y latitud se realiza internamente en el software del proyecto a través de funciones trigonométricas.

Los CAM enviados en este caso 1 se recogieron en la base de datos y se filtraron para recoger únicamente los parámetros de posición (latitud, longitud y altitud) y la velocidad en m/s utilizando sqlite a través de la siguiente petición:

```
select key,latitude,longitude,altitude,speed from CAM
```

La razón por la que se generaron los CAM viene especificada en la base de datos, en la tabla CamInfo, donde también aparece la diferencia de magnitudes entre dos CAM consecutivos que provocó que se enviase el siguiente. Los resultados de la simulación del caso 1 pueden apreciarse en la Figura 15[3].

	latitude	longitude	altitude	speed
1	900000001	1800000001	800001	16383
2	617321200	87355420	105	16383
3	517321200	87355420	105	40
4	517421199	87355420	105	0
5	517321200	87355420	105	90
6	517421199	87355420	105	150
7	517321200	287355419	105	120
8	617321200	87355420	105	20
9	617321200	87355420	105	0
10	617321200	87355420	105	10
11	617321200	87355420	105	70
12	617321200	87355420	105	150
13	517321200	87355420	105	110
14	517421199	87355420	105	60
15	517321200	87355420	105	30
16	517421199	87355420	105	60
17	517321200	287355419	105	30

	key	time	triggerReason	delta
1	1	149753...	First GPS data	-1.0
2	2	149753...	distance	1111949.26...
3	3	149753...	distance	1111.94926...
4	4	149753...	distance	1111.94926...
5	5	149753...	distance	1111.94926...
6	6	149753...	distance	1372868.47...
7	7	149753...	distance	1638304.44...
8	8	149753...	speed	20.0
9	9	149753...	speed	10.0
10	10	149753...	speed	60.0
11	11	149753...	speed	80.0
12	12	149753...	distance	1111949.26...
13	13	149753...	distance	1111.94926...
14	14	149753...	distance	1111.94926...
15	15	149753...	distance	1111.94926...
16	16	149753...	distance	1372868.47...
17	17	149753...	distance	1638304.44...

Figura 15. Tabla CAM y CamInfo para el caso 1.

[3] En las tablas que aparecen en la Figura 15 las velocidades que aparecen en la tabla CAM y las direferencias de velocidades que aparecen en la tabla CamInfo aparecen en dm/s, siendo el máximo valor de las velocidades simuladas 15 m/s (150 dm/s).

En la Figura 15 se puede ver en la parte izquierda una tabla que recoge la posición y velocidad de los CAM enviados y en la parte derecha otra con la información acerca de por qué se generaron dichos CAM.

Conviene señalar que en la base de datos los mensajes se empiezan a contabilidad a partir del segundo mensaje, ya que el primero únicamente informa de los valores máximos que pueden tomar los parámetros. En el segundo mensaje se vuelve a enviar la velocidad máxima (hasta la fecha desconocemos la razón para ello, puesto que sería posible introducir en este mensaje la velocidad real), lo que es incorrecto, pero aún así no es problema para enviar el mensaje porque se utiliza como disparador del mismo la diferencia de distancias.

Como se puede en las tablas de la Figura 15, desde los mensajes 2 y 3 hasta los mensajes 7 y 8, los CAM se generan debido a una variación brusca de la posición, debido a que las posiciones entre mensajes consecutivas es mayor de 5 m (se puede observar como los valores de longitud y latitud van cambiando continuamente). Después, los mensajes del 8 al 12 se simularon para que se generasen debido a una variación de velocidad, ya que las posiciones simuladas del vehículo son iguales en todos esos mensajes (mismos valores de latitud y longitud). Tal y como se aprecia en la Figura 15, la razón de la generación de estos mensajes es la que preveíamos, ya que en la tabla CamInfo aparece que la diferencia de velocidades es mayor de 1 m/s. Los restantes mensajes que se aprecian en la Figura 15 se generan como los que se han explicado en primer lugar, por una gran variación en la posición. Por lo tanto el envío de mensaje por variación de posición se deberá a acelaraciones bruscas, mientras que el envío de mensaje por variación de velocidad se deberá tanto a aceleraciones bruscas como a frenazos.

Por otro lado, para el caso 2, simulamos situaciones buscando que se produciese el vencimiento de un temporizador y entonces se generase un CAM al haber pasado más de 1 segundo desde la generación del anterior mensaje.

La tabla CAM y la tabla CamInfo para este caso 2 pueden verse en la Figura 16.

	latitude	longitude	altitude	speed
16	617321200	87355420	105	60
17	617321200	87355420	105	30
18	617321200	87355420	105	80
19	617321200	87355420	105	80
20	617321200	87355420	105	210
21	617321200	87355420	105	110

	key	time	triggerReason	delta
16	16	149786340...	speed	30.0
17	17	149786340...	speed	30.0
18	18	149786340...	speed	50.0
19	19	149786340...	time	100260515...
20	20	149786340...	speed	130.0
21	21	149786340...	speed	100.0

Figura 16. Tabla CAM y CamInfo para el caso 2.

Para el estudio del caso 2, se utilizó para la configuración de las posiciones un fichero local, en el que las posiciones eran las mismas en todo momento, para así provocar que en caso de que las velocidades fuesen las mismas, se generase el CAM debido a que el temporizador de un segundo venciese.

La situación anteriormente comentada se produce al enviar las tramas 18 y 19. Como se puede ver en la Figura 16, los mensajes 18 y 19 tienen los mismos valores de posición y velocidad y como consecuencia de ello la trama 19 se generó debido a que se excedió el temporizador de un segundo, como se puede ver en la tabla CamInfo, donde el triggerReason es time y la delta es de aproximadamente un un segundo (1002'6 ms).

5

Pruebas de campo

Para completar el desarrollo total del proyecto OpenC2X, el último paso es recoger datos reales de un vehículo e incorporarlos a nuestro caso de estudio con fines comerciales o en este caso didácticos. A partir de los datos recogidos, que como ya se explicó anteriormente serán la velocidad y geolocalización (las rpm del motor se obtienen pero no se incluyen en el estándar ETSI ITS G5), se estudiarán diferentes situaciones, tal y como se hacía en el caso simulado, para ver si la generación de mensajes se realiza de forma satisfactoria

5.1 Configuración del GPS

Para la recogida de los datos GPS, antes debemos ejecutar unas instrucciones en nuestro sistema operativo para poder conectar y sincronizar el proyecto OpenC2X con el sistema GPS.

El dispositivo GPS utilizado es Holux GPSlim236 [11], el cual tiene la posibilidad de establecer conexión tanto vía Wi-Fi como vía Bluetooth. En nuestro caso, la conexión entre nuestro software y el dispositivo GPS se llevará a cabo a través de Bluetooth, y para ello se utilizará un puerto serie a través del cual se realiza la transferencia de datos.

El puerto serie se define como una interfaz que permite la transferencia de datos bit a bit entre el dispositivo y el ordenador. En el caso del proyecto OpenC2X utilizaremos un puerto serie virtual [38] (/dev/rfcommX, donde X es el número de canal utilizado), el cual no viene configurado por defecto, por lo que, después de introducir el usb Bluetooth, habrá que ejecutar los siguientes comandos [39] para que dicho interfaz esté disponible:

1. Comprobación de que el usb Bluetooth ha sido detectado.
   hcitool dev
2. Búsqueda de los dispositivos Bluetooth cercanos
   hcitool scan

3. Conexión entre dispositivo GPS y puerto serie 0 asociado al proyecto C2X. El resultado de dicho comando se puede ver en la Figura 17.
   /usr/bin/rfcomm connect 0 00:0B:0D:84:CD:3C

```
Connected /dev/rfcomm0 to 00:0B:0D:84:CD:3C on channel 1
```

Figura 17. Conexión entre el dispositivo GPS y el puerto serie.

4. Por último, en un nuevo terminal, habilitaremos la conexión anterior para establecer la comunicación y de esta forma poder ver todos los datos que están siendo recogidos.
   gpsd –b –N –D 4 /dev/rfcomm1

Estos pasos deberán seguirse siempre que se deseen recoger datos del GPS y serán análalos a los pasos necesarios para realizar la conexión entre el proyecto OpenC2X y el dispositivo OBDII a través del puerto serie 1, los cuales están especificados en el trabajo fin de grado de Pilar Sánchez [30].

Una vez terminada la configuración del puerto serie, se debe comprobar si la conexión se ha realizado correctamente en cuyo caso estaremos recibiendo los datos de posición. Para este cometido, emplearemos el comando xgps de Linux, mediante el cual se nos muestran en una interfaz gráfica los satélites empleados en la obtención de la posición y los datos que se están recibiendo. Un ejemplo del resultado que se obtiene al ejecutar el comando xgps es el mostrado en la Figura 18.

En la figura anterior se pueden apreciar la lista con todos los satélites localizados, así como su posición geográfica. Los satélites están clasificados siguiendo un código de colores para indicar la calidad de la señal, siendo el verde la mejor calidad, amarillo una calidad media y rojo la peor calidad de señal. Justo debajo del gráfico donde aparecen situados los satélites, si la conexión entre el dispositivo GPS y el puerto serie se ha realizado de una manera correcta, deberían aparecer los valores de latitud, longitud y altitud continuamente, que serán los valores que se recogerán automáticamente en el proyecto OpenC2X una vez este se haya ejecutado.

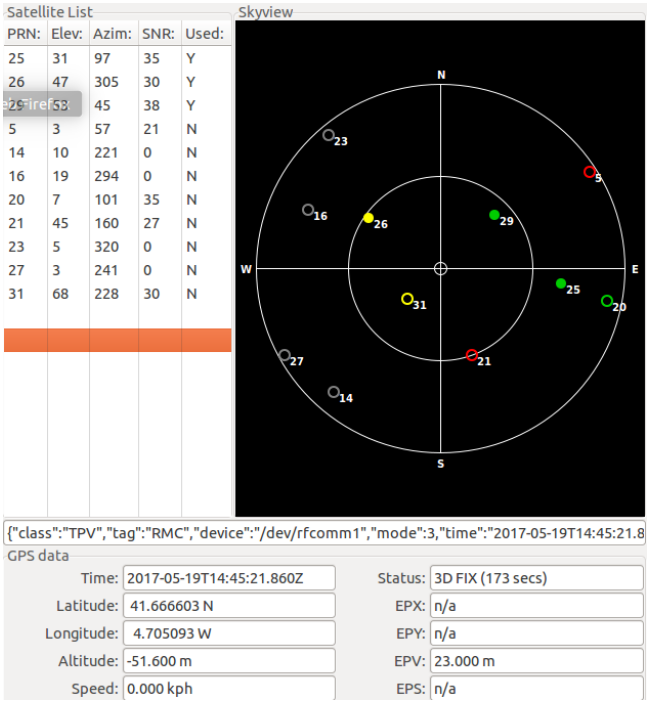

Figura 18. Localización de los satélites y datos recibidos.

5.2 Recogida de datos

El propósito de esta sección es el de demostrar que los CAM, tal y como se especificó en la sección 3.2.3, también se pueden generar cuando la dirección o rumbo (*heading)* que está siguiendo el vehículo cambia bruscamente respecto del punto cardinal norte entre dos mensajes consecutivos.

El motivo de realizar este estudio en esta sección con datos reales y no en la sección 3.2.3 con datos simulados es simplemente que utilizando un vehículo real los cambios de dirección se realizan prácticamente de manera continua, provocando que se cumpla la condición requerida de diferencia de *heading* con mayor facilidad que con datos simulados.

Por lo tanto este caso de estudio, tal y como se ha relatado anteriormente, es el de generar CAM debido a que la diferencia de *heading* entre dos mensajes consecutivos sea mayor de 4º (siempre y cuando también se cumpla la condición de que la distancia que se ha movido el vehículo entre la generación de esos dos mensajes haya sido mayor que el

parámetro de configuración *ThresholdRadiusForHeading*). Los resultados correspondientes a este caso de estudio se muestran en la Figura 19[4].

	key	latitude	longitude	altitude	speed
1	1	900000001	1800000001	800001	16383
2	2	900000001	1800000001	800001	16383
3	3	416649066	-47094100	702	16383
4	4	416649049	-47094116	703	0
5	5	416649049	-47094133	703	0
6	6	416649049	-47094149	703	27
7	7	416649049	-47094149	703	194
8	8	416648966	-47094266	703	194
9	9	416648966	-47094266	703	305
10	10	416648783	-47094516	704	305
11	11	416648783	-47094516	704	500
12	12	416648483	-47094933	704	500
13	13	416648100	-47095416	704	583
14	14	416647699	-47095933	704	583
15	15	416647249	-47096499	704	666
16	16	416647249	-47096499	704	805

	key	time	triggerReason	delta
1	1	1499954272409792004	time	1.4999542724...
2	2	1499954273411957824	time	1001772050.0
3	3	1499954274413966023	time	1001660771.0
4	4	1499954274514799662	First GPS data	-1.0
5	5	1499954275517895676	time	1002758981.0
6	6	1499954276521072948	time	1002901022.0
7	7	1499954277524172424	time	1002827656.0
8	8	1499954278326761978	speed	166.66666666...
9	9	1499954278527904470	heading	-136.0542675...
10	10	1499954279220349357	speed	111.11111111...
11	11	1499954279520822535	heading	-136.8084671...
12	12	1499954280022245192	speed	194.44444444...
13	13	1499954280523868819	heading	-136.2843148...
14	14	1499954281525989466	distance	5.8556432906...
15	15	1499954282528694735	distance	6.1808001925...
16	16	1499954283431345952	distance	6.8699003633...

Figura 19. Tabla CAM y CamInfo con datos recogidos de una prueba real.

Tal y como puede observarse en la tabla CamInfo de la Figura 19, el número de mensajes que son generados debido a un cambio de rumbo en grados es más significativo que en las pruebas simuladas y en este caso, es más difícil registrar la generación de un mensaje CAM debido a un cambio de posición y velocidad, ya que para ello es necesarior acelerar muy bruscamente el vehículo.

La apariencia que tiene la tabla CAM, en comparación con las obtenidas en las pruebas simuladas en el laboratorio (sección 3.2.3), puede hacer pensar que se han recogido los datos de una forma errónea, debido a que la mayoría de las posiciones aparecen duplicadas. Sin embargo, este hecho tiene una explicación razonable.

En el proyecto OpenC2X, a la hora de generar un CAM, el código software realiza la comparación de una manera secuencial de la posición o distancia que ha recorrido el vehículo desde la generación del último CAM, el cambio de *heading* que ha experimentado, la variación de velocidad y que el tiempo entre dos CAM consecutivos no supere un segundo. Esta situación se ilustra en la Figura 20 y es el motivo por el cual, en una misma posición se

[4] Las velocidades mostradas en la tabla CAM y las diferencias de velocidad mostradas en CAMInfo están expresadas en cm/s.

puedan generar varios CAM; dentro de la misma posición (sin variación de la misma) se pueden dar tres situaciones diferentes que generen el mensaje CAM que son si ha cambiado la distancia recorrida más de 5m, si la dirección del vehículo o *heading* ha cambiado más de 4º respecto al norte y si la velocidad ha variado más de 1m/s.

```
//periodically check generation rules for sending to LDM and DCC
void CaService::alarm(const boost::system::error_code &ec) {
     // Check heading and position conditions only if we have valid
GPS data
     if(isGPSdataValid()) {
          if (!mLastSentCamInfo.hasGPS) {
               sendCamInfo("First GPS data", -1);
               mLogger->logInfo("First GPS data");
               trigger();
               return;
          }

          //|current position - last CAM position| > 5 m
          if(isPositionChanged()) {
               trigger();
               return;
          }

          //|current heading (towards North) - last CAM heading| > 4
deg
          if(isHeadingChanged()) {
               trigger();
               return;
          }
     }

     //|current speed - last CAM speed| > 1 m/s
     if(isSpeedChanged()) {
          trigger();
          return;
     }

     //max. time interval 1s
     if(isTimeToTriggerCAM()) {
          trigger();
          return;
     }

     //|Wait 100ms and executes alarm() again
     scheduleNextAlarm();
      }
```

Figura 20. Secuencia de generación de CAM.

Una vez se ha aclarado la interpretación de las tablas CAM y CamInfo anteriores, se va a explicar el procedimiento por el cual se generaron los CAM. La prueba empieza con el vehículo parado, tal y como se puede ver en los mensajes 4 y 5, donde la velocidad es cero. Los mensajes del 4 al 6 se generan debido al vencimiento del temporizador de 1 segundo, ya

que tanto la diferencia de velocidad como de posición entre ellos no es lo suficientemente grande como para generar un CAM.

Después, los mensajes 7 y 8 se generan de la siguiente manera; primero se cumple que ha pasado más de un segundo, pero la velocidad todavía no ha cambiado lo suficiente, sin embargo cuando se genera el mensaje CAM, la velocidad se actualiza a 1,94 m/s. A los 100 ms se vuelve a comprobar el estado, y en este caso se cumple la condición de cambio de velocidad, porque cuando se ha generado el mensaje 7, el software no ha actualizado la variable que controla la última velocidad enviada[5], con lo que en esta variable sigue guardándose la velocidad del mensaje 6, que es 0,27 m/s. Por lo tanto la diferencia de velocidades es 1,66 m/s y se disparará el mensaje CAM por cambio de velocidad superior a 1 m/s. El mensaje 9 tiene la misma posición que el mensaje 8 y se genera por un cambio de *heading* (136,05º) con respecto a la posición del mensaje 7 (misma razón que para los mensajes 7 y 8).

Posteriormente, los mensajes 10 y 11 se generarán por un cambio de velocidad (1,11m/s) y *heading* (136,08º) con respecto al mensaje 9, el cual aunque aparezca con una velocidad de 3,05 m/s, en realidad esa velocidad se alcanzará en el mensaje 10 y no en mennsaje 9. Los mensajes 12 y 13 se generan de la misma manera que los mensajes 10 y 11 y el mensaje 14 se genera por una variación de la distancia de más de 5 m con el mensaje 13.

Los restantes mensajes se generan por circunstancias similares a los explicados anteriormente, ya sea por variaciones en la velocidad de más de 1 m/s, por variaciones en el parámetro *heading* de más de 4 grados o por variaciones de distancia de más de 5 m.

5.3 Resultados gráficos

Para completar el estudio sobre el prototipo de RSU realizado a través del proyecto OpenC2X, utilizaremos los datos recogidos de un vehículo real para registrar su geolocalización en el mapa mediante el API (Application Programming Interface) v3 de Google Maps [40]. Dicha aplicación de usuario, así como la plataforma web local utilizada

[5] Sería conveniente pensar si el sistema tiene que funcionar así realmente, dado que lo lógico quizás sería que dado que esa velocidad ya se ha enviado no se volviera a enviar. Por otro lado, así nos aseguramos que quedan claras las dos razones por las que se envía el mensaje.

para visualizar la información han sido tomadas del proyecto fin de carrera de Javier Fernández Pastrana [41].

Para introducir los datos recogidos en el campo correspondiente se utilizaron las librerías *mySQL* para el lenguaje de programación C++. Para tal fin se creó una base de datos que contaba con una única tabla (*gprmTable*), que nos permitió almacenar las muestras de la localización y cuyos campos son:

- Latitud
- Longitud
- Fecha
- Hora
- Velocidad
- Revoluciones por minuto (RPM).

Para rellenar la base de datos, antes fue necesario realizar la conversión de los parámetros de UCTTime (hora) y velocidad. Para obtener la hora con el formato año/mes/dia hora/min/seg hubo que transformar el tiempo en el que se enviaron los CAM (parámetro *time* en la tabla CamInfo), ya que este tiempo nos da los segundos que pasaron desde 1970 hasta la generación del CAM. Para realizar la conversión empleamos la siguiente instrucción en Excel:

Formato fecha = ((tiempo_a_converir/60)/60)/24)+FECHA(1970;1;1)

Por otra parte, la velocidad la tendremos que convertir de m/s a km/h. Para ello únicamente multiplicaremos por 3'6.

Entonces, una vez realizadas las modificaciones anteriores, se podrá obtener el mapa filtrando las fechas desde el inicio de la ruta hasta el final de la ruta. El resultado obtenido se muestra en la Figura 21, donde se pueden apreciar los *Markers* (puntos en el mapa) y que nos permitirán visualizar la información referida a ese instante de tiempo.

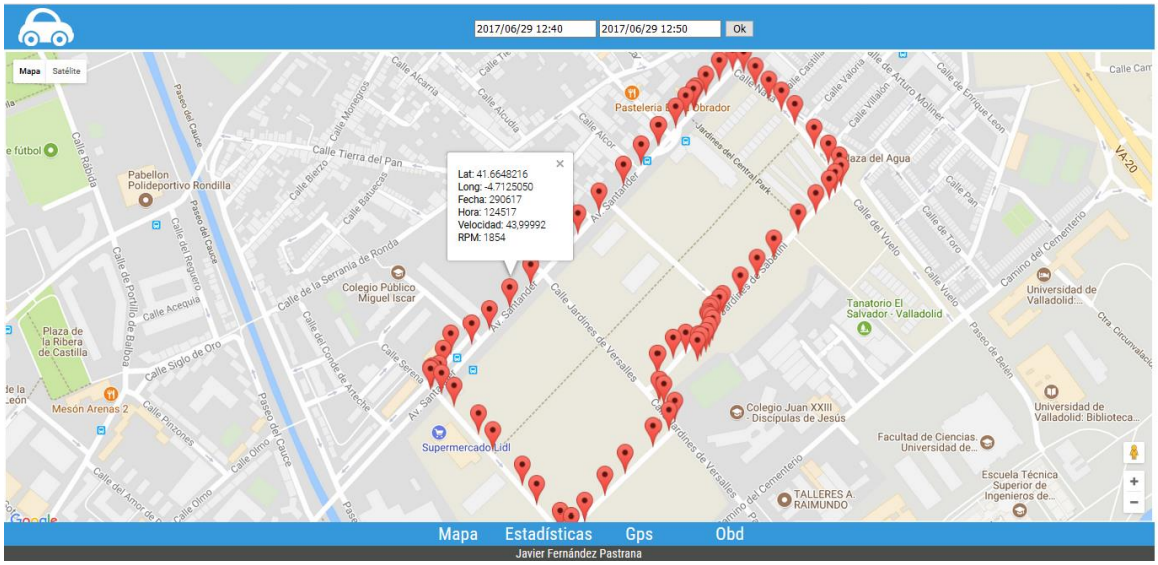

Figura 21. Geolocalización

6

Conclusiones y líneas futuras

6.1 Conclusiones

Teniedo en cuenta los objetivos marcados en la sección 1.2 de este proyecto, se puede concluir que se ha conseguido realizar la implementación de una RSU, utilizando para ello un primer prototipo proporcionado por los CSS-Labs, que buscaba desarrollar un sistema para las comunicaciones V2X, campo que como se ha visto en el estado del arte, está en auge.

En lo que se refiere al desarrollo software, en primer lugar se ha conseguido entender e instalar todos los componentes necesarios que servían de base para la realización de este trabajo fin de grado y que ya habían sido desarrollados, como por ejemplo el estándar 802.11p implementado para la capa de acceso de la pila de protocolos del estándar ETSI ITS G5 sobre el kernel 4.2.8 de Linux. De igual forma se logró instalar sobre dicho kernel el software del proyecto OpenC2X que implementaba la RSU.

A nivel de compresión del estándar ETSI ITS G5, se han cumplido todos los objetivos previstos, en primer lugar estudiando con detalle a nivel teórico el código C++ que implementaba cada uno de los protocolos que forma el estándar junto con los documentos donde estos están definidos, estableciendo de este modo las relaciones entre ellos. Y en segundo lugar, comprobando mediante la realización de simulaciones que todas las partes del proyecto OpenC2X estaban bien coordinadas y la comunicación entre todos los módulos que lo comprendían era la correcta.

Además del estudio a nivel a teórico, el uso de los dispositivos OBDII y GPS permitió realizar un análisis del proyecto OpenC2X a nivel práctico, ya que hicieron posible la recogida de datos de un vehículo y su inserción en el proyecto utilizado. Esto hizo factible el uso de un entorno real para saber la frecuencia a la que se generaban los mensajes que se intercambian las estaciones ITS. En este apartado del testeo del proyecto OpenC2X a nivel

práctico se logró realizar utilizar las herramientas proporcionadas por Javier Férnandez [41] para dotar a esta investigación de un aspecto más visual e intuitivo en el que analizar los datos recogidos del vehículo.

Por último destacar que durante la realización de este proyecto han sido utilizadas prácticamente casi todas las bases adquiridas durante la carrera además de aspectos nuevos y desconocidos que han tenido que ser aprendidos durante el desarrollo del mismo todo ello aplicado a un sector desconocido pero aun así ha desembocado en la culminación de un prototipo de RSU. Por lo que las conclusiones extraídas a nivel didáctico son más que satisfactorias.

6.2 Líneas futuras

Tal y como se ha dicho durante todo la memoria el proyecto utilizado es muy completo y a la vez muy innovador. Es por ello que todavía consta de alguna de las carencias que permiten que existan ciertos puntos claros para la continuación del proyecto:

Nuevas líneas de desarrollo:

- Implementación de los módulos que le faltan a la pila de protocolos del estándar ETSI ITS G5 para el proyecto OpenC2X, como son el GeoNetworking y la capa vertical de seguridad.

- Buscar alguna solución al fallo al obtener la carga del canal utilizando la tecnología 802.11p

- Comunicación V2V en un entorno real entre dos o más vehículos.

- Llevar a cabo una comunicación V2I empleando una RSU real.

7

Bibliografía

[1] Dirección General de Tráfico (DGT), "Nuevo mínimo histórico en el número de victimas mortales en accidentes desde 1960", 2016. Disponible en: http://www.dgt.es/es/prensa/notas-de-prensa/2016/20160104-nuevo-minimo-historico-numero-victimas-mortales-accidente-desde-1960.shtml (09/07/2017)

[2] Grant-Muller, S. y Usher, M., "Intelligent transport systems: the propensity for environmental and economic benefits," Technological Forecasting and Social Change, 2014, vol. 82, no. 1, pp. 149–166. Disponible en: http://ac.els-cdn.com/S0040162513001364/1-s2.0-S0040162513001364-main.pdf?_tid=b1c029c4-5caa-11e7-997c-00000aacb35f&acdnat=1498727565_bdde63fc6b53fb2366103350d9702d1a (09/07/2017)

[3] ISO/TR 17465-1, "Intelligent Transport System – Terms and Definitions", 2014. Disponible en: https://www.iso.org/obp/ui/#iso:std:iso:tr:17465:-1:ed-1:v1:en (09/07/2017)

[4] General Motors, "Cadillac Builds on V2V Deployment with V2I Development", 2017. Disponible en: https://www.gm.com/mol/m-2017-may-0530-cadillac.html (09/07/2017)

[5] Ford, "Autonomous Vehicles", 2015. Disponible en: http://corporate.ford.com/microsites/sustainability-report-2015-16/mobility-smartmobility-vehicles.html (09/07/2017)

[6] Drive C2X, "Drive C2X successfully completed in July 2014". Disponible en: http://www.drive-c2x.eu/project (09/07/2017)

[7] Cooperative ITS Corridor, "Project details Cooperative ITS Corridor". Disponible en: https://itscorridor.mett.nl/English/Project+details/default.aspx (09/07/2017)

[8] Fernández Pastrana, J "802.11p standard and V2X applications on comercial Wi-Fi cards", Universidad de Valladolid, 2017.

[9] Laux, S., Singh Pannu, G., Schneiden, S., Tiemann, J., Klingler, F., Sommer, C. y Dressler, F., "OPENC2X", CSS-Labs, 2016. Disponible en: http://www.ccs-labs.org/software/openc2x/ (09/07/2017)

[10] *Elm Electronics* Fabricante del chipset ELM237, necesario para el *Dongle* que ofrece la conexión al CAN BUS a través del OBDII. Disponible en: http://www.elmelectronics.com/obdic.html#ELM327 (09/07/2017)

[11] Descripción del Holux GPSlim236. Disponible en: http://www.holux.com/JCore/en/products/products_content.jsp?pno=340 (09/07/2017)

[12] Sassi, A., El Hillali, Y., Revenq, A., Charfi, F. y Kamoun, L., "Enhancing V2X Communication Based on a New Comb-Pilot Estimation Approach", International Journal of Vehicular Technology, 2016. Disponible en: https://www.hindawi.com/journals/ijvt/2016/8341490/ (09/07/2017)

[13] Strandén, L., Ström Chalmers, E. y Uhlemann, E., "Wireless Communications Vehicle-to-Vehicle and Vehicle-to-Infraestructure",SAFER, 2008, pag 24. Disponible en: http://publications.lib.chalmers.se/records/fulltext/137657.pdf (09/07/2017)

[14] CONCERT TR 1013, "*Internal to CONCERT Consortium*",CORDIS, 1998. Disponible en: http://cordis.europa.eu/telematics/tap_transport/research/projects/concert.html

[15] ETSI EN 302 663 "Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band", ETSI, 2012. Disponible en:

http://www.etsi.org/deliver/etsi_en/302600_302699/302663/01.02.00_20/en_302663v010200a.pdf (09/07/2017)

[16] Information Society Technologies, "COMSafety", 2009. Disponible en: http://www.transport-research.info/sites/default/files/project/documents/20130515_125913_29633_COMeSafety_DEL_D31_EuropeanITSCommunicationArchitecture_v2.0_01.pdf (09/07/2017)

[17] ECC, " The armonised use of the 5875-5925 Mhz frequency band for intelligent Transport Systems (ITS)", 2008. Disponible en : http://www.erodocdb.dk/docs/doc98/official/pdf/ECCDec0801.pdf (09/07/2017)

[18] 5GAA, "Telecommunications and automotive players form global cross-industry 5G Automotive Association", 2016. Disponible en: http://5gaa.org/pdfs/5GAA-Press-Release.pdf (09/07/2017)

[19] Wevers, K. y Lu, M., "V2X Communication for ITS- from 802.11p Towards 5G ", IEEE 5G, 2017. Disponible en: http://5g.ieee.org/tech-focus/june-2017/v2x-communication-for-its (09/07/2017)

[20] Rielb, R., "Vanetza", Technische Hochschule Ingolstadt, 2013. Disponible en: https://github.com/riebl/vanetza

[21] Voronov, A., "Geonetworking", 2016. Disponible en: https://github.com/alexvoronov/geonetworking

[22] GCDC, "I-GAME project ",2016. Disponible en: http://gcdc.net/en/i-game (09/07/2017)

[23] Ericsson Mobility Report, "On the pulse of the networked society", 2015. Disponible en: https://www.ericsson.com/assets/local/news/2016/03/ericsson-mobility-report-nov-2015.pdf (09/07/2017)

[24] ETSI TS 102 636-4-1, "Intelligent Transport Systems (ITS); Vehicular communications; Geonetworking: Part 4: Geographical addressing and forwarding for point-to-point and point-yo-multipoint communications; Sub-part1: Media-Independent Functionality", ETSI, 2011. Disponible en: http://www.etsi.org/deliver/etsi_ts/102600_102699/1026360401/01.01.01_60/ts_1026360401v010101p.pdf

[25] E. Meseguer, J., T. Calafate, C., Cano, J.C. y Manzoni, P., "Characterizing the Driving Style Behavior using Artificial Intelligence Techniques", Universitat Politècnica de València. Disponible en: http://www.ieeelcn.org/prior/LCN38/lcn38demos/Demo2-Meseguer.pdf (09/07/2017)

[26] Pagina Web Ubuntu. Disponible en: https://www.ubuntu.com/

[27] Kernel Linux 4.2.8. Disponible en: https://www.kernel.org/pub/linux/kernel/v4.x/

[28] Laux, S., Singh Pannu, G., Schneiden, S., Tiemann, J., Klingler, F., Sommer, C. y Dressler, F., "Demo: OpenC2X – An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5", CSS-Labs, 2016. Disponible en: http://www.ccs-labs.org/bib/laux2016openc2x/laux2016openc2x.pdf

[29] Lin, L., Tomatis, A. y Festag, A., "Global deployment of Car-2-X communication technology", Drive C2X, 2012. http://www.drive-c2x.eu/tl_files/publications/DRIVEC2X_ITSWC2012_SIS68_DRIVEC2X_Lin.pdf (09/07/2017)

[30] Sánchez Martín, P. "Implementación de una unidad de a bordo de comunicación entre vehículos según el estándar ETSI ITS G5", Universidad de Valladolid, 2017.

[31] Estandar OBD_II Estándar OBD-II International Organization for Standardization, ISO 9141-2:1994/Amd 1:1996,1196.

[32] ETSI EN 302 637-2 "Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service", ETSI, 2014. Disponible en: http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf (09/07/2017)

[33] ETSI EN 302 637-3 "Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Part 3: Specification of Descentralized Environmental Notification Basic Service", ETSI, 2014. Disponible en: http://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.01_30/en_30263703v010201v.pdf (09/07/2017)

[34] ETSI EN 302 895 "Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM)", ETSI, 2014. Disponible en: http://www.etsi.org/deliver/etsi_en/302800_302899/302895/01.00.00_20/en_302895v010000a.pdf (09/07/2017)

[35] Definición de la librería SQlite. Disponible en: https://www.sqlite.org/

[36] Standards of Automotive Engineering (SAE), "Dedicated Short Range Communications. Message Set Dictionary", 2009. Disponible en: http://standards.sae.org/j2735_200911/

[37] ETSI EN 102 687 "Intelligent Transport Systems (ITS); Descentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part", ETSI, 2011. Disponible en: http://www.etsi.org/deliver/etsi_ts/102600_102699/102687/01.01.01_60/ts_102687v010101p.pdf (09/07/2017)

[38] Descripción del puerto serie virtual. "Serial Port". Disponible en: https://en.wikipedia.org/wiki/Serial_port#.22Virtual.22_serial_ports

[39] Ebiii, "GPS Parser: GPS set up",2014. Disponible en: https://github.com/LSIR/gsn/wiki/GPS-Parser (09/07/2017)

[40] API v3 Google maps. Disponible en: https://developers.google.com/maps/documentation/javascript/?hl=es (09/07/17).

[41] Fernández Pastrana, J., "Sistema M2M para telemetría y geolocalización de vehículos mediante *Raspberry Pi*", Universidad de Valladolid, 2014.

[42] Nokia, "Vehicle-to-everything", 2017. Disponible en: https://networks.nokia.com/vehicle-to-everything

[43] Filippi, A.,Moerman, K., Daalderop, G., D. Alexander, P., Schober, F. y Pfliegl, W., "Why 802.11p beats LTE and 5G for V2X", eeNewsAutomotive, 2016. Disponible en: http://www.eenewsautomotive.com/design-center/why-80211p-beats-lte-and-5g-v2x

Acrónimos

API: Application Programming Interface

ASN1: Abstract Syntax Notation One

BTP: Basic Transport Protocol.

CAM: Cooperative Awareness Message

CSS: Computer and Communication Systems

DCC: Descentralized Congestion Control

DENM: Descentralized Environmental Notification Message

DSRC: Dedicated short-range communications

DTC: Diagnostic Trouble Code.

ECU: Electronic Control Unit.

GPS: Global Possitioning System

ITS: Intelligent Transport Systems

LDM: Local Dynamic Map

OBU: On Board Unit

OBD: On Board Diagnostic

OCB: Outside the Context of a BSS

POTI: Positioning and Timing Management

RSU: Road Side Unit

VDP: Vehicle Data Providing

V2V: Vehicle to Vehicle

V2I: Vehicle to infraestructure

V2X: Vehicle to everything

VANET: Vehicular Ad-Hod Network

WLAN: Wireless Local Area Network

5GAA: 5G Automotive Association

Anexo 1

En este anexo se muestra con un detalle a más bajo nivel las diferentes funciones del protocolo GPS para una mejor comprensión global del mismo.

En un primer momento se realiza una llamada a la función `sendToServices(gpsPackage::GPS gps)`. Dicha función obtiene los datos GPS. Si son datos reales, únicamente los recoge a través de la función `receivedData()`. Si son simulados, en la función `simulatedData()`, dichos datos GPS se obtendrán a través de las funciones `simulateNewPosition(currentPosition, speed/10, 0)` y `simulateSpeed()`. Estas simulaciones de datos se realizan mediante las siguientes distribuciones:

```
//for simulation only, default values.
mRandNumberGen = default_random_engine(0);
mBernoulli = bernoulli_distribution(0);
mUniform = uniform_real_distribution<double>(-0.01, 0.01);
```

Cambiando los valores que estas funciones toman como argumentos de entrada se obtendrá una mayor o menor rango de valores en la velocidad y posición, que influirán posteriormente en los motivos por los cuales se generarán o no los mensajes CAM.

Por último, la función `gpsDataToBuffer(struct gps_data_t* gpsdata)` será la encargada de almacenar los datos obtenidos o simulados en el buffer para su envío al resto de módulos.

Es conveniente apuntar, que tanto para este como para el resto de protocolos, los objetos que permiten el intercambio de información entre las diferentes partes del proyecto y protocolos se encuentran definidos en la carpeta Common/Utility.

Anexo 2

En este anexo se muestra con un detalle a más bajo nivel las diferentes funciones del protocolo CAM para una mejor comprensión global del mismo.

Después de configurar los parámetros en el fichero config.xml, se realizará una primera comprobación para ver si es necesario generar un nuevo CAM. Para ello se utiliza la función `isTimeToTriggerCAM()`, que verifica si ha pasado el intervalo de 1seg entre los dos últimos mensajes CAM. En ese caso, mediante la función `sendCamInfo()` se enviaría el motivo de porque se ha generado ese CAM., y se generaría el mensaje con la función `trigger()`. Este procedimiento se sigue de igual manera para todas las condiciones explicadas en la sección 3.1.3, mediante las funciones `isPositionChanged()`, `isHeadingChanged()` e `isSpeedChanged()`. En la función `scheduleNextAlarm()` se controla mediante un temporizador que no se comprueben las condiciones de generación de CAM en intervalos inferiores a 100 ms.

Después de que la función `trigger()` se haya ejecutado, se genera el mensaje CAM. Este proceso se realiza en la función `generateCAM()`, función que se encuentra dentro de `send()`, mediante asignación dinámica de memoria, donde se le asigna al mensaje el tamaño dinámicamente con la función calloc en función del tamaño de Cam_t. Tras rellenarse los campos del mensaje, este se enviará a los módulos DCC y LDM.

La acción de recibir mensajes CAM de otras estaciones ITS, mostrarlo por pantalla y reenviarlo al LDM se realiza en la función `receive()`, la cual escucha el medio de transmisión (interfaz utilizada para la comunicación) continuamente mediante un bucle while.

El resto de funciones de este módulo son utilizadas para adecuar los datos a los buffers de almacenamiento y no son relevantes para la comprensión del proyecto.