Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema de videovigilancia de vehículos basado en Kurento

Autor: David Gutiérrez Hermida
Tutor: María Teresa Ariza Gómez

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema de videovigilancia de vehículos basado en Kurento

Autor:
David Gutiérrez Hermida

Tutor:
María Teresa Ariza Gómez
Profesor titular

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Carrera: Sistema de videovigilancia de vehículos basado en Kurento

Autor: David Gutiérrez Hermida

Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis profesores

Agradecimientos

Tras un largo camino lleno de muchos dolores de cabeza, me gustaría dedicarle estas últimas líneas a aquellas personas que me han acompañado durante todo este proceso.

En primer lugar, agradecerle a mi tutora, María Teresa Ariza Gómez su atención y apoyo durante todo el tiempo que he estado realizando el proyecto. Y también darle las gracias por su ayuda en el transcurso de estos años de carrera en las asignaturas que imparte de telemática, donde su actitud comprensiva y buen hacer con todos los alumnos hacen que sea una profesora querida por todos.

A mis amigos y compañeros, con los que he pasado agobios y muchos días de estudio, pero también muchas risas. Sin los mensajes interminables para hacer y resolver dudas durante los meses de exámenes, los mensajes de apoyo y ánimo, las buenas y malas experiencias compartidas en las prácticas y los largos días en la E.T.S.I., este camino no habría sido el mismo.

A mis padres, por brindarme la oportunidad de estudiar esta carrera y hacer todo lo posible para que continuara adelante y nunca me diera por vencido, sin su esfuerzo y apoyo no lo habría conseguido.

A todos por igual, gracias por hacer esto posible.

David Gutiérrez Hermida

Sevilla, 2021

Resumen

En los últimos años se ha visto cómo ha crecido exponencialmente el número de vehículos que cuenta con conexión a internet y múltiples funcionalidades que lo convierten en un “coche conectado”. Por este motivo, en este proyecto se busca encontrar una nueva funcionalidad para poder sacar aún más provecho a los vehículos y su seguridad.

A continuación, se desarrolla una aplicación enfocada a la seguridad de los automóviles con la que será posible evitar intrusiones y robos dentro de los vehículos. El propietario del vehículo será notificado cuando se esté produciendo un robo en el coche y, además, podrá ver el interior del mismo en tiempo real.

Esta herramienta puede ser instalada en el vehículo de diferentes formas. En este caso se ha optado por el uso de una Raspberry pi, donde se alojará el código en Python para notificar y transmitir. Asimismo, el interior del vehículo dispondrá de una webcam para mostrar la imagen en directo.

La herramienta base para desarrollar este proyecto es Kurento, un servidor de medios que permite la comunicación del flujo de vídeo. Además de esta aplicación, otros instrumentos utilizados para emitir la grabación en el dispositivo del cliente han sido Docker o Servidores Spring

Abstract

In recent years, the number of vehicles with an internet connection and multiple functionalities that make it a “connected car” has grown exponentially. For this reason, this project seeks to find a new functionality to be able to get even more out of vehicles and their safety.

Next, an application focused on car safety is developed with which it will be possible to prevent intrusions and theft inside vehicles. The owner of the vehicle will be notified when a theft is taking place in the car and will also be able to see the interior of the car in real time.

This tool can be installed in the vehicle in different ways. In this case, the use of a Raspberry pi has been chosen, where the Python code will be housed to notify and transmit. It will also have a webcam inside the vehicle to show the live image.

The base tool to develop this project is Kurento, a media server that allows the communication of the video stream. In addition to this application, other instruments used to broadcast the recording on the client's device have been Docker or Spring Servers. -translation by google-

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
KMS	Kurento Media Server
RTP	Real Transport Protocol
POM	Project Object Model
RTC	Real Time Communication
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1 INTRODUCCIÓN

Many things are improbable, only a few are impossible.

- Elon Musk -

1.1 Motivación

La principal motivación que me ha llevado a realizar este proyecto ha sido la de crear una plataforma que nos permita en caso de un robo dentro de un vehículo poder ver el interior del coche en tiempo real y desde cualquier dispositivo con el simple hecho de tener internet, para de esta manera poder identificar a los asaltantes.

Este proyecto se ha realizado usando diferentes tecnologías que están presentes en nuestro día a día y pudiendo ser accesible a cualquier persona (OpenSource), así como implementable en cualquier vehículo de la actualidad que cuenten con acceso a Internet. Todo esto es posible gracias al exponencial aumento en los últimos años del internet de los dispositivos (IoT), todos conectados a través de Internet.

Destacar también la importancia de que es un sistema completo, donde su conjunto da la posibilidad de aportar un producto totalmente funcional y aplicable a la realidad, para esto ha sido necesaria la implementación de varias tecnologías que serán descritas en apartados posteriores.

La base donde se sustenta todo el sistema es [1]Kurento, tecnología gracias a la que ha sido posible la transmisión en tiempo real del vídeo, usando para ello un protocolo de comunicación propio donde configurar y ajustar todos los parámetros de esta transmisión.

1.2 Objetivos

El objetivo del presente proyecto ha sido crear una plataforma que nos permita la visualización en tiempo real del interior del vehículo cuando en este se produzca un robo o intento de robo. Para hacer esto posible se ha desarrollado una aplicación que se ha denominado Carprotect, que se instala en el coche y se encarga de trasmitir el vídeo captado por la [2]webcam hacia el Servidor de Medios de Kurento que lo procesa. Además de esto se ha desarrollado un CarProtect Server y un cliente Web para que el usuario final pueda visualizar el interior del vehículo. Destacar también la introducción del sistema de notificaciones por email en la aplicación Carprotect, donde automáticamente, cuando se produzca el intento de un robo se enviará una notificación al correo electrónico previamente configurado desde donde el usuario final podrá entrar en una url para así poder ver el interior del vehículo.

---

[1] Kurento: Kurento es un servidor de medios WebRTC y un conjunto de API de cliente que simplifican el desarrollo de aplicaciones de vídeo avanzadas para plataformas WWW y teléfonos inteligentes. [23]

[2] WebCam: cámara de vídeo portátil, en el caso de este proyecto, será instalada en el vehículo del usuario final.

Uno de los objetivos más importantes del proyecto, es poder aportar una aplicación completa y usable, para ello aparte de la transmisión y tratamiento del vídeo en tiempo real, se ha configurado todo para poder ser instalado en un dispositivo externo como una [3]Raspberry pi y así poder ser incluido en cualquier vehículo con conexión a internet.

1.3 Antecedentes

El principal motivo que me ha llevado a la realización de este proyecto ha sido mi estrecho contacto e interés por el mundo del motor y la tecnología. En los últimos años los automóviles han sufrido una gran evolución tecnológica de forma que cuentan con funcionalidades que hace unos años eran impensables.

Gracias a mi empresa, un taller de electricidad de automóviles, he estado en contacto con las innovaciones automovilísticas. Hace unos años que se han implementado cámaras en los vehículos (principalmente en transporte público y de mercancías), la conocida como [4]Dashcam (se muestra en la Figura 1) usada para grabar continuamente lo que pasa en la vía y gracias a esto se me ocurrió la idea de hacer justamente lo contrario, grabar el interior del coche, y además resolver el problema de los robos, para poder identificar a los ladrones cuando intenten cometer uno de estos.

Figura 1. Ejemplo de Dashcam [1].

Otro de los motivos principales ha sido el poder usar e indagar en tecnologías de transmisión y procesamiento de vídeo, esto ha sido posible gracias a Kurento que me ha brindado la oportunidad de poder trabajar en su tecnología.

1.4 Solución

1.4.1 Características de la solución

En la solución que se ha desarrollado se han usado diferentes tecnologías con el fin de crear una plataforma de videovigilancia de vehículos totalmente funcional. A continuación, se explican las características principales de la solución:

- Sistema hardware: en primer lugar, para poder llevar la aplicación al usuario final y concretamente a su vehículo, se han pensado en dos soluciones, si el vehículo cuenta con un sistema hardware actual, se podría instalar la aplicación Python fácilmente en el [5]ordenador a bordo del vehículo, en el caso de que esto no fuera posible se ha hecho uso de una Raspberry Pi donde poder tener alojado todo lo

[3] Raspberry pi: La Raspberry Pi es una serie de ordenadores de placa reducida, ordenadores de placa única u ordenadores de placa simple (SBC) de bajo coste [24]

[4] Dashcam: La **dashcam** (o **dash cam**) no es más que una cámara convencional que viene específicamente preparada para ser utilizada dentro de un coche. [25]

[5] Ordenador a bordo: sistema de información del vehículo desde donde se controlan todos los parámetros y medidas de la conducción.

necesario para la transmisión del vídeo hacia los servidores de Kurento, por supuesto para esto necesitaremos que la Raspberry o el vehículo tengan continua conexión a Internet.

- Carprotect: aplicación software principal que irá instalada en el vehículo o bien en la Raspberry pi, en esta se realizan varias acciones básicas, en primer lugar, cuando se recibe un aviso de que se está produciendo un robo, este inmediatamente notifica al usuario a través de un email, donde va contenido la url de la página web (Cliente Web) que permite ver el interior del vehículo, para esto ha sido necesario la instalación de un servidor de correo electrónico desde donde enviar los emails. Se puede configurar algunos parámetros como el correo del usuario final a quién notificar gracias al fichero de configuración denominado "Carprotect_ini.json".

  Por otra parte, CarProtect es el encargado de trasmitir el vídeo de la webcam, que estará conectada a la raspberry. Esta aplicación envía el vídeo a Kurento Media Server (KMS) mediante flujo RTP, un detalle a destacar es que el CarProtect Server notifica a Carprotect del puerto donde se encuentra Kurento a la espera de recibir flujo de vídeo, ya que este puerto es elegido de manera aleatoria por Kurento, por esta razón se ha creado un "mecanismo de comunicación" en el que el CarProtect Server le comunica y Carprotect el puerto al que tiene que enviar el vídeo al KMS.

- Simulador de Alarmas: es una pequeña aplicación, con poco interés tecnológico. Esta aplicación que se implementa en la Raspberry se encarga de enviar una alarma simulando un "robo" y de esta forma poder probar la plataforma.

- Kurento Media Server: es el encargado de recibir el vídeo RTP de la webcam, de todo el procesamiento del vídeo y de enviarlo al CarProtect Server para que lo exponga en la página web.

- El CarProtect Server, usando la tecnología Spring, es el encargado de realizar la comunicación tanto con el usuario final, como con Kurento Media Server, a través del cliente Web. Donde se indica que se inicie la retrasmisión del vídeo

- Cliente Web: página web del cliente a la que accederemos mediante la url contenida en la notificación por email enviada por Carprotect, en dicha página web podremos solicitar el inicio de la transmisión, que se comunicará al CarProtect Server y donde posteriormente podremos ver en tiempo real el interior del vehículo, también podremos parar la retransmisión cuando nos parezca oportuna para que Kurento Media Server detenga la trasmisión de vídeo.

1.4.2 Caso de uso

El caso de uso principal de la plataforma desarrollada es la vigilancia y control del interior del vehículo, de esta manera el usuario final podrá desde cualquier lugar, recibir una notificación cuando en su vehículo se haya producido una intrusión, esta intrusión podría ser detectada por los diferentes sensores presentes en los vehículos de hoy en día. En nuestro proyecto esta intrusión es simulada. Una vez que se reciba la notificación de robo, a través de correo electrónico, ésta contendrá una url que el usuario podrá visitar y desde donde podrá ver en tiempo real el interior del vehículo a través de la webcam previamente instalada en el mismo.

1.4.3 Esquema de la arquitectura

En la Figura 2 se muestra el esquema de la arquitectura que se ha desarrollado. En este esquema se representan los tres grandes desarrollos del proyecto. La programación de la parte que va dentro del coche (Carprotect), la configuración, adaptación e instalación de Kurento Media Server y el CarProtect Server, y la creación del Cliente Web.

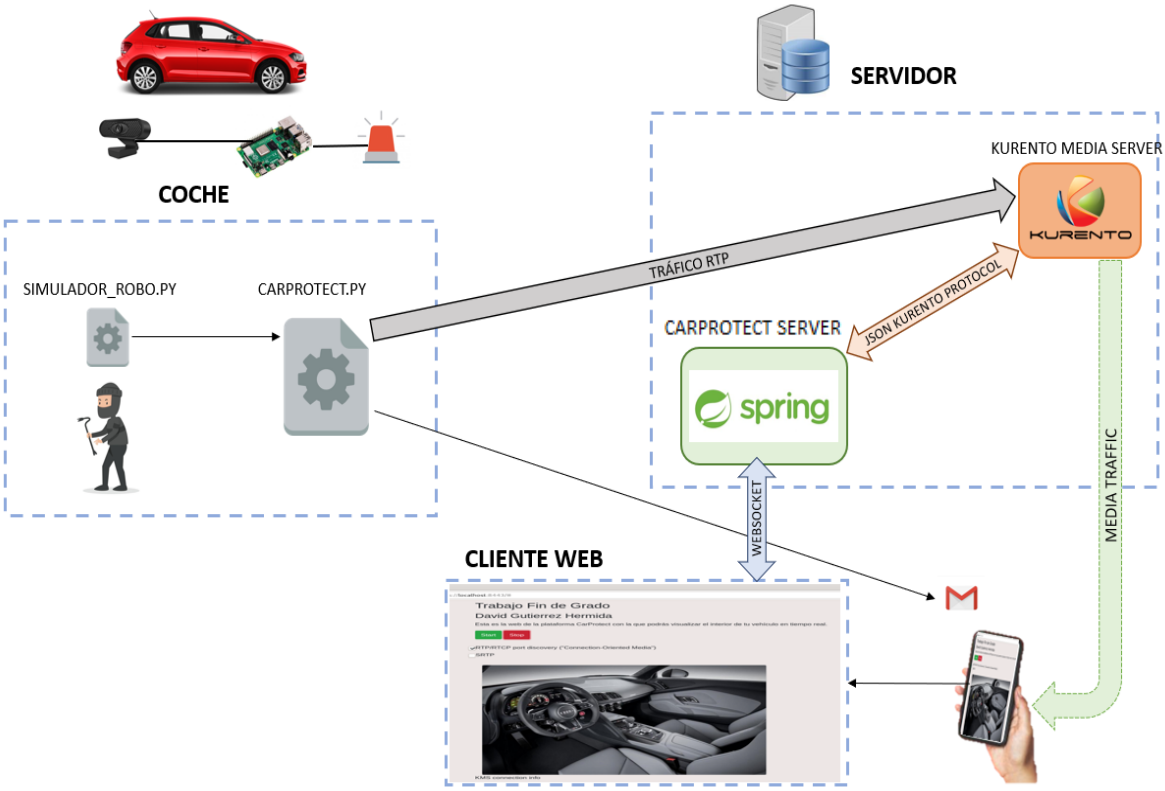

Figura 2. Esquema de la arquitectura.

En el Apartado 5 del presente documento, Desarrollo: Cliente Web, se detalla sobre este mismo esquema el funcionamiento y el paso de mensajes que se lleva a cabo entre las distintas plataformas para poder visualizar en tiempo real el interior del vehículo.

1.5 Tiempo empleado

Nº	ETAPAS	DESCRIPCIÓN	DURACIÓN (horas)
1º	Búsqueda de información	En esta fase se ha realizado una búsqueda de toda la información necesaria para el desarrollo del Proyecto.	80
2º	Desarrollo: Servidores	En esta etapa del Trabajo se ha desarrollado todo lo relativo a los Servidores, tanto la parte del servidor de Kurento Media Server como el CarProtect Server usando Spring.	70
3º	Desarrollo: Carprotect	Comprende el diseño y codificación de la aplicación Carprotect que se instala en los automóviles y es la encargada de notificar al usuario en caso de robo y de comenzar la trasmisión del flujo de vídeo. En esta etapa también se ha desarrollado una pequeña aplicación encargada de simular una alerta de robo.	70
4º	Desarrollo: Cliente Web	En esta parte se desarrolla una página Web donde el usuario puede visualizar en tiempo real el interior de su vehículo.	40
5º	Documentación	La última fase del trabajo engloba la escritura de toda la documentación: memoria, anexos, presentación …	65

Tabla 1. Planificación temporal.

A continuación, se presenta en la Figura 3 y 4 lo anteriormente descrito mediante un diagrama de Gantt:

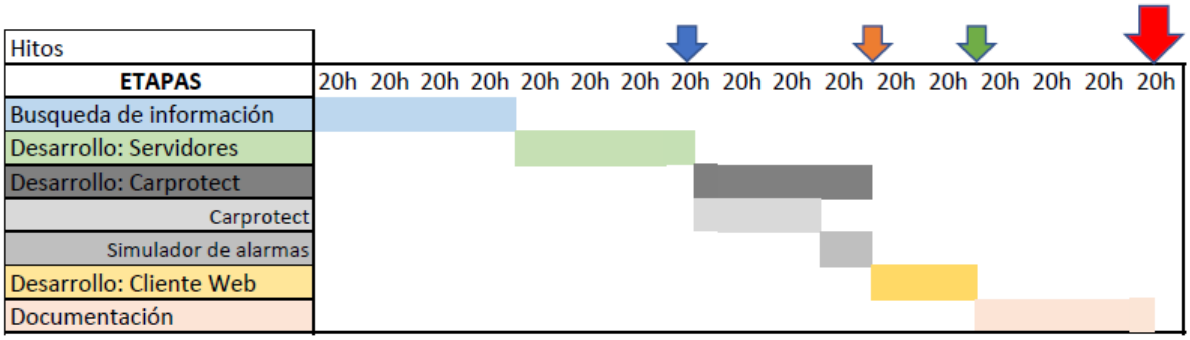

Figura 3. Diagrama de Gantt.

En el Diagrama de Gantt se han marcado los siguientes hitos del proyecto:

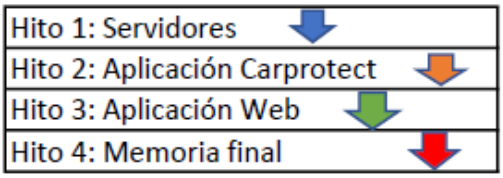

Figura 4. Hitos del proyecto.

1.6 Estructura de la Memoria

El presente documento se ha estructurado de la siguiente forma:

1. Introducción: motivación, objetivos, antecedentes, características y planificación temporal del trabajo realizado.
2. Tecnologías utilizadas: se exponen las tecnologías utilizadas a lo largo del proyecto tanto los elementos hardware como los recursos software.
3. Desarrollo Servidores: análisis del CarProtect Server y explicación del servidor de Kurento Media Server.
4. Desarrollo Carprotect: análisis y desarrollo de la aplicación Carprotect, así como del notificador de alarmas.
5. Desarrollo Cliente Web: análisis y desarrollo de la aplicación web del usuario final.
6. Conclusiones y líneas futuras: conclusiones finales, se exponen los problemas encontrados y líneas de mejoras para la aplicación.

Además, se han incluido al final del documento los Anexos:

- Anexo A: Preparación del entorno. En este anexo se detallan los pasos necesarios para preparar el entorno para ejecutar toda la plataforma creada.
- Anexo B: Instalación y configuración. En esta parte se exponen los comandos necesarios para instalar y configurar los servidores y Carprotect para que se ejecuten correctamente.
- Anexo C: Ejecución. Secuencia de comandos a ejecutar para poner en funcionamiento la plataforma.

2 TECNOLOGÍAS UTILIZADAS

En este apartado se exponen todas las tecnologías que se han usado para la elaboración del Trabajo. Entre estas se encuentran los recursos Software tales como la plataforma Kurento, los distintos lenguajes de programación utilizados, los frameworks… y los recursos hardware.

2.1 Recursos Software

2.1.1 Entorno de Desarrollo

Para construir el entorno de desarrollo sobre el que se ha planteado todo el proyecto se han utilizado:

- [6]Máquina Virtual Debian 10: se ha usado una máquina virtual Debian, para aislar todo el desarrollo del proyecto. En esta máquina virtual con sistema operativo Debian se han instalado todos los servidores tanto el de Kurento como el CarProtect Server.

Figura 5. Logo Debian [2]

- Docker: se trata de un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores software.

Figura 6. Logo Docker [3]

  En este proyecto se han usado los contenedores para desplegar el Servidor de Medios de Kurento. Se ha optado por esta tecnología debido a su facilidad para el despliegue de dicho servidor. En el Anexo A: Preparación del Entorno se detalla el despliegue del servidor usando Docker.

- Spring Boot: es un framework de spring que facilita la creación de servicios Web programados en Java. Se ha usado en el proyecto para la creación del CarProtect Server, que a su vez se comunica con el Servidor de Medios de Kurento.

Figura 7. Logo Spring [4]

- Maven: es una herramienta para la gestión y construcción de proyectos Java. Se basa en un fichero POM (Project Object Model) que describe el proyecto.

[6] Máquina virtual: Una máquina virtual no es más que un software **capaz de cargar en su interior otro sistema operativo** haciéndole creer que es un PC de verdad [26]

Figura 8. Logo Maven [5].

2.1.2 Plataformas utilizadas

- Kurento: es una Plataforma para crear aplicaciones WebRTC para el tratamiento y trasmisión de Vídeo en Tiempo Real (Real-Time Communications).

Figura 9. Logo Kurento [6].

Todo el proyecto se ha basado en esta Plataforma, se ha desplegado un Servidor Multimedia (Kurento Media Server) necesario para el posterior desarrollo de la web y visualizar el interior del vehículo en tiempo real.

- GStreammer: es una herramienta de trasmisión de contenido audiovisual.

Figura 10. Logo GStreamer [7].

Se ha usado esta herramienta para la trasmisión del flujo de vídeo desde la aplicación Carprotect que se ha desarrollado hasta Kurento Media Server. Gracias a esta herramienta se trasmite el flujo de vídeo captado por la Webcam usando el protocolo RTP (Real-Time Transport Protocol).

2.1.3 Lenguajes de programación

Durante la realización del Trabajo se han desarrollado diversas aplicaciones usando diferentes lenguajes dependiendo de la funcionalidad de ésta. Los principales lenguajes de programación usados son los siguientes:

- Python: es un lenguaje de programación interpretado y multiplataforma. Se caracteriza principalmente por su sencillez. En el proyecto las aplicaciones Carprotect y el Simulador de Robos han sido programados usando este Lenguaje.

Figura 11. Logo Python [8].

- Java: es un lenguaje de programación Orientado a Objetos ampliamente utilizado para todo tipo de desarrollos. En el Trabajo que se ha realizado se ha usado para el desarrollo del CarProtect Server.

Figura 12. Logo Java [9].

- HTML, CSS y JavaScript (JS): HTML es un lenguaje de marcado para estructurar las páginas Web. Junto con CSS para la asignación de Estilos y JavaScript se ha desarrollado la Web del Cliente con la que interactuará el usuario final.

Figura 13. Logo HTML, CSS y JS [10].

2.2 Recursos Hardware

Se ha empleado una Rasbpberry pi y una Cámara IP (Webcam) que se instalan en un vehículo y sobre la que se instala la herramienta Carprotect. Las características de estos elementos Software son las siguientes.

- Raspberry Pi 4 Modelo B 4GB (Figura 14) con las características:
  - Procesador: Broadcom BCM2711, SoC de 64 bits Cortex-A72 (ARM v8) de 64 bits a 1,5 GHz.
  - Memoria: SDRAM LPDDR4-2400 de 4GB.
  - Conectividad: IEEE 802.11ac de 2.4 GHz y 5.0 GHz, Bluetooth 5.0, BLE.
  - Gigabit Ethernet.
  - 2 puertos USB 3.0; 2 puertos USB 2.0.
  - Cabezal GPIO Raspberry Pi estándar de 40 pines (totalmente compatible con las placas anteriores)
  - 2 puertos micro-HDMI (hasta 4kp60 compatibles)

Figura 14. Raspberry Pi 4 [11].

- Cámara web (Figura 15) con las características:
    - Webcam Full HD 1080p: Resolución de alta definición panorámica 1920 X 1080P realista a 30 fps.
    - Conexión puerto USB

Figura 15. Cámara Web [12].

3 DESARROLLO: CARPROTECT

Work hard. Have fun. Make history.

- Jeff Bezos -

En este capítulo se detallan los aspectos más relevantes del desarrollo de la aplicación Carprotect que va instalada en el vehículo, y que es la encargada de trasmitir el vídeo de la cámara de seguridad al servidor de Kurento.

3.1 Estructura de la aplicación

Carprotect es una aplicación desarrollada en Python 3, que se instala en la raspberry que está conectada a la cámara de vigilancia del interior del vehículo.

Las principales funcionalidades son las siguientes:

- Simular una alarma de ejemplo para poder comprobar el funcionamiento de la plataforma.
- Notificar al usuario de que se ha producido un robo.
- Trasmitir el vídeo desde la webcam hasta el servidor de medios de Kurento para que este lo procese y esté accesible desde el CarProtect Server.

Para la configuración del correo electrónico, que será particular para cada usuario que use la aplicación se ha usado un fichero de configuración, "*Carprotect_ini.json"* que la aplicación lee al inicio de su ejecución.

Para poder agregar todas las funcionalidades a la aplicación se han hecho uso de las librerías que se muestran en la Figura 16.

```
import smtplib
import ssl
import random
import sys
import socket
import shlex
import subprocess
import json
```

Figura 16. Librerías de Python.

Entre estas destacamos:

- SMTPLIB: librería para el manejo y envío de correo electrónico.
- JSON: necesaria para importar y leer ficheros JSON, en nuestro caso el fichero de configuración.
- SHLEX: librería para ejecutar subprocesos desde el script. Se usa para invocar a la herramienta GS-Streamer.
- SOCKET: librería encargada de la comunicación mediante sockets.

Es importante tener en cuenta que muchas de las librerías no están disponibles para versiones de Python 2.X por lo que se recomienda ejecutar la aplicación exclusivamente con Python3.

3.2 Simulador de Alarmas

En el Código de la aplicación se define una función llamada simuladorAlarmas, esta función de poco interés practico recibe por parámetro el email al que se pretende enviar la notificación del robo.

Esta función pregunta al usuario a través de la consola de comandos si desea generar una alarma, simulando que se está produciendo un robo, una vez que el usuario decide simular la alarma, se enviaría la notificación por correo electrónico.

En otras palabras, es como si estuviéramos generando una alarma de manera aleatoria, para así poder probar el sistema completo y que el flujo de vídeo llegue desde la webcam al usuario final.

3.3 Notificación por correo electrónico

Una vez simulada la alarma, la aplicación procede a notificar por correo electrónico al usuario. Para este envío de correo electrónico se debe configurar las variables definidas en el fichero de configuración carprotect_ini.json que se muestran en la Figura 17.

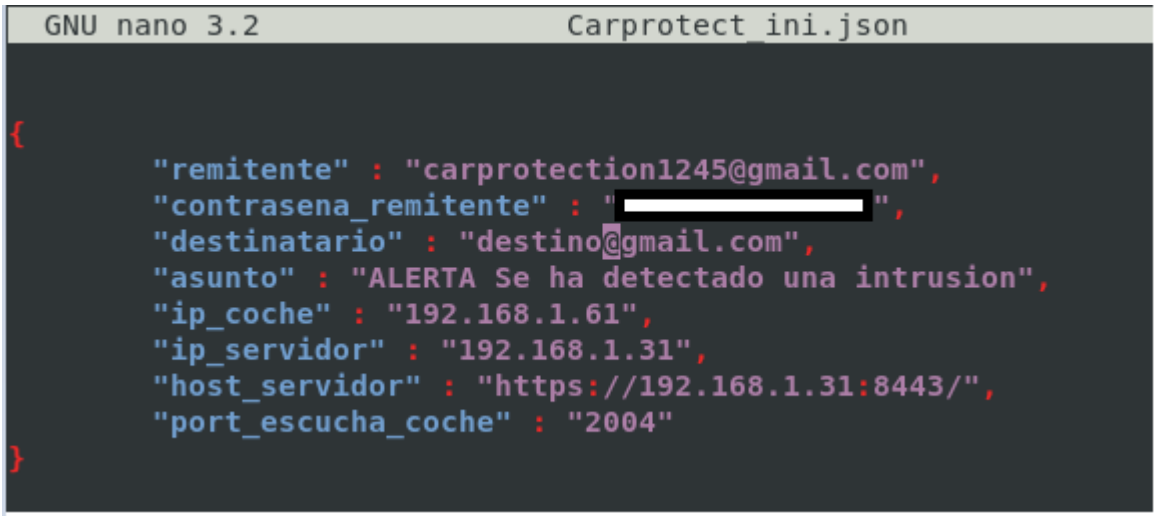

Figura 17. Carprotect_ini.json

En él se definen las siguientes variables:

- Remitente: correo emisor.
- Contraseña_remitente: contraseña del correo emisor.
- Asunto: asunto del mensaje.
- Ip_coche: dirección ip del coche.

- Ip_servidor: dirección ip de la máquina donde se encuentran los servidores.
- Host_servidor: url del CarProtect Server, donde se encuentra publicada la pagina web del cliente.
- Port_escucha_coche: puerto de escucha para la negociación del puerto donde enviar el vídeo de la cámara web.

Además, la aplicación trae configurado un cliente de correo genérico carprotect@gmail.com que es desde el correo que se envía la notificación. Para esto último se hace uso de la librería smtplib y se debe de configurar el servidor de correo en este caso "gmail" y el puerto. En la Figura 18 se muestra el fragmento del código desarrollado para esta tarea.

```
try:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login("carprotection1245@gmail.com", password)
    print ("Login correctamente")
    server.sendmail(remitente, destinatario, email)
    server.quit()
    print ("Correo enviado correctamente")
except:
    print ("""Error: el mensaje no pudo enviarse""")
```

Figura 18. Desarrollo de envío de correo.

Al usuario le llegará un correo electrónico a la dirección indicada, en dicho correo se alerta de que se está produciendo un intento de robo, y también una dirección url que haciendo clic en ella podrá acceder a la página web, desde donde ya podrá visualizar el interior de su vehículo. Los detalles del correo electrónico que se recibe se pueden observar con más detalles en la Figura 19.

Figura 19. Notificación recibida por correo electrónico.

3.4 Envío de Vídeo

Uno de los puntos más importantes en el desarrollo de la aplicación es el envío del vídeo desde Carprotect al Servidor de Medios Kurento.

Para poder enviar el vídeo se ha usado la herramienta Gs-Streamer, invocada desde la aplicación, y que se encarga de capturar el vídeo de la webcam y lo envía usando el protocolo [7]RTP. La herramienta se ejecuta como se muestra en el siguiente fragmento de código.

```
$ comando = 'gst-launch-1.0 -v v4l2src device=/dev/vídeo0 ! image/jpeg,
width=1280, height=720, framerate=30/1 ! jpegdec ! vídeoconvert ! queue
! x264enc ! queue ! rtph264pay ! "application/x-
rtp,payload=(int)103,clock-rate=(int)90000,ssrc=(uint)5004" ! udpsink
host=’ + ip_servidor + ’ port=' + msg +' bind-port=5004'
$ subprocess.call(shlex.split(comando))
```

En la ejecución del comando gst-launch se especifican una serie de parámetros para la captura y el envío del vídeo, a continuación, se detallan los más importantes:

- “Device”: ubicación de la webcam en la máquina, por lo general siempre está en /dev/vídeo0
- Anchura y altura de la imagen (width and heigth)
- “Framerate”: tasa de fotogramas.
- X264enc: Este elemento codifica vídeo sin procesar en datos comprimidos H264, también conocido como MPEG-4 AVC (Códec de vídeo avanzado).
- Ip destino, en este caso la ip del servidor (“ip_servidor”).
- Puerto destino (“msg”): este puerto es negociado anteriormente entre Carprotect y el CarProtect Server.

En el momento que el usuario final decide visualizar el contenido en la aplicación web es cuando el Servidor de Medios de Kurento busca un puerto disponible en la máquina y se pone a la escucha de vídeo RTP. Para que la aplicación Carprotect conozca dicho puerto se ha desarrollado un “Protocolo de comunicación” donde el CarProtect Server le comunica el puerto de Kurento al que tiene que enviar el vídeo.

Esto se ha llevado a cabo habilitando un puerto de escucha en arprotect (servidor de escucha) al que el CarProtect Server notifica el puerto.

En la Figura 20 se muestra el paso de mensajes necesarios para la negociación del puerto.

[7] RTP: significa “Real Time Transport Protocol” (Protocolo de transporte en tiempo real), y define un formato de paquete estándar para el envío de audio y vídeo sobre Internet. Es definido en el RFC1889 [27]

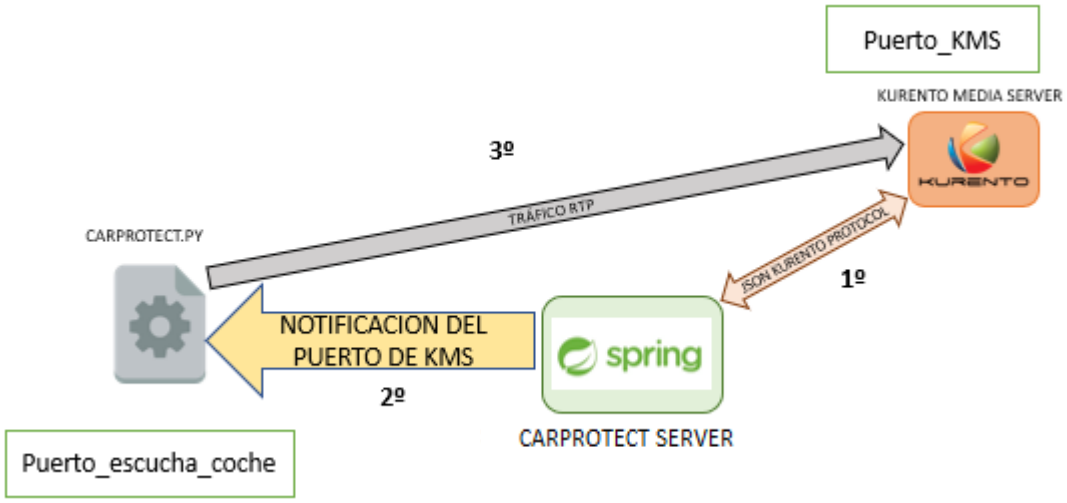

Figura 20. Negociación del puerto.

- **1º.** El CarProtect Server se comunica con Kurento Media Server y este le dice cuál es el puerto de escucha de vídeo. Por ejemplo, supongamos que KMS está a la escucha por el puerto 2222
- **2º.** Carprotect Server se comunica con Carprotect por el Socket establecido (puerto_escucha_coche = 2004) y le informa de cuál es el puerto de KMS, en este ejemplo el 2222.
- **3º.** Una vez que Carprotect conoce el puerto (2222) ejecuta el comando de Gstreamer y empieza la trasmisión del vídeo de la webcam.

Por último, se muestra en la figura 21 el montaje de la webcam, así como la raspberry que contiene a Carprotect.py en un vehículo real.

Figura 21. Montaje de webcam y Raspberry Pi en el vehículo

4 DESARROLLO: SERVIDORES

En el presente capítulo se explica todo lo relativo a los servidores, tanto del CarProtect Server como del Servidor de Medios Kurento. Este apartado detallará como ha sido el desarrollo del CarProtect Server, la comunicación con KMS y como interactúan con el resto de aplicaciones de la plataforma, el cliente web y el Carprotect.

4.1 Kurento Media Server

Kurento Media Server (KMS) es un servidor, de código abierto, multimedia que se utiliza para desarrollar aplicaciones de vídeo avanzadas que utilizan plataformas WebRTC. [15]

"WebRTC es un conjunto de protocolos, mecanismos y [8]API que proporcionan a los navegadores y aplicaciones móviles capacidades de comunicaciones en tiempo real (RTC) a través de conexiones de igual a igual." -translation by google-. [16]

4.1.1 ¿Por qué Kurento Media Server?

Kurento Media Server se puede utilizar para permitir la transmisión, procesamiento, grabación y reproducción de medios. KMS se basa en la fantástica biblioteca multimedia GStreamer y ofrece las siguientes características [17]:

- Protocolos de transmisión en red, incluidos HTTP, RTP y WebRTC .
- Comunicaciones de grupo soportando medios de mezcla y los medios de encaminamiento / despacho.
- Soporte genérico para filtros que implementan algoritmos de Visión por Computador y Realidad Aumentada.
- Almacenamiento multimedia que admite operaciones de escritura para WebM y MP4 y reproducción en todos los formatos admitidos por GStreamer.
- Transcodificación de medios automática entre cualquiera de los códecs compatibles con GStreamer, incluidos VP8, H.264, H.263, AMR, OPUS, Speex, G.711 …

En la Figura 22 se muestra las diferencias entre los servidores de medios comunes y KMS:

[8] API: Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones. [28]

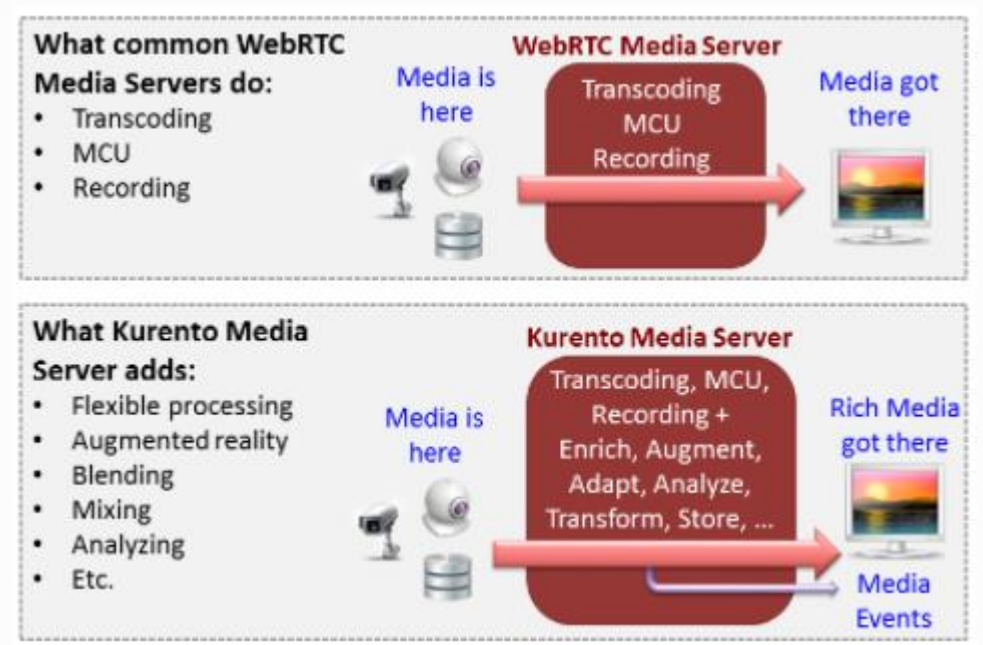

Figura 22. Diferencias entre los servidores de medios [18].

Kurento está diseñado en base a los siguientes principios fundamentales:

- Distribución de medios y servicios de aplicaciones: Kurento media Server está diseñado para poder ser implementado entre diferentes máquinas. De esta forma un KMS puede recibir peticiones de más de una aplicación a la vez y viceversa.
- Adecuado para la nube: está preparado para poder integrarse en la nube y actuar como un componente [9]PaaS.
- Canalizaciones de medios: Kurento es capaz de encadenar elementos multimedia a través de canalizaciones. Gracias a esto se simplifica la complejidad del procesamiento de contenido multimedia.
- Desarrollo de aplicaciones: todas las aplicaciones desarrolladas por los programadores pueden ser diseñadas en cualquier plataforma o tecnología, tal y como prefiera el desarrollador. Esto es gracias a que Kurento se adapta a todas las plataformas, pudiendo el desarrollador excluirse de las complejidades internas de Kurento.
- Capacidad de comunicación de un extremo a otro: Kurento tiene la capacidad de llevar a cabo la comunicación completa de un extremo a otro, por lo que el desarrollador no tendrá que transcodificar, transportar o renderizar medios en el extremo del cliente.
- Flujos de medios totalmente procesables: Kurento permite no solo la comunicación interactiva entre dos usuarios (estilo Skype), sino que puede procesar vídeo de persona a máquina o de máquina a máquina (por ejemplo, en intercambio de datos multisensoriales)
- Procesamiento auditable: Kurento es capaz de generar información usable y legible para el monitoreo, facturación y auditoría de QoS.
- Integración perfecta con IMS: Kurento está diseñado para poder integrarse con la herramienta [10]IMS usada por los operadores de telefonía.
- Capa de adaptación de medios transparente: gracias a la convergencia que proporciona esta capa de adaptación usada por Kurento es usable en dispositivos que tengan requisitos de pantalla diferentes, o

[9] PaaS: PaaS significa Plataforma como servicio. Se trata de un conjunto de servicios basados en la nube que permiten a los usuarios empresariales y desarrolladores crear aplicaciones de forma rápida y rentable. [29]
[10] IMS: Subsistema Multimedia IP (IMS) o (IP Multimedia Subsystem) es un conjunto de especificaciones que describen la arquitectura de las redes de siguiente generación (Next Generation Network, NGN), para soportar telefonía y servicios multimedia a través de IP. [30]

diferencias en la velocidad de transmisión.

Se puede obtener información detallada de KMS en su documentación [19].

4.1.2 Kurento Media Server aplicado a Carprotect

En este proyecto se ha usado el Servidor de Medios para recibir el flujo de vídeo desde la aplicación Carprotect instalada en el coche, y para reproducirlo en la aplicación web.

Se ha usado su implementación en Docker debido a su facilidad de despliegue, en el Anexo B: Instalación y configuración se muestran los pasos a seguir para realizar esta tarea.

La aplicación web desarrollada muestra un flujo RTP simple: un RtpEndpoint está configurado en KMS para escuchar un flujo de vídeo entrante. Este flujo debe ser generado por un programa externo, que en nuestro caso se genera con la aplicación Carprotect, usando Gstreamer. En la página se proporciona información visual, conectando el RtpEndpoint a un WebRtcEndpoint en modo de solo recepción. El servidor Java, que se verá en detalle en el siguiente apartado, se conecta a todos los eventos emitidos desde KMS.

4.2 CarProtect Server

Para el desarrollo del CarProtect Server se ha utilizado la documentación de Kurento “RTP Receiver” [17]. Se ha desarrollado un servidor usando Java en el lado del servidor, basado en el framework de Spring Boot [20] para facilitar el proceso de desarrollo e implementación.

4.2.1 Estructura

En la Figura 23 se muestra la estructura de directorios para la creación del CarProtect Server. Para construir el servidor se ha partido del tutorial oficial de kurento de RTP Receiver [21] modificando solo aquellos ficheros necesarios para conseguir la funcionalidad deseada.

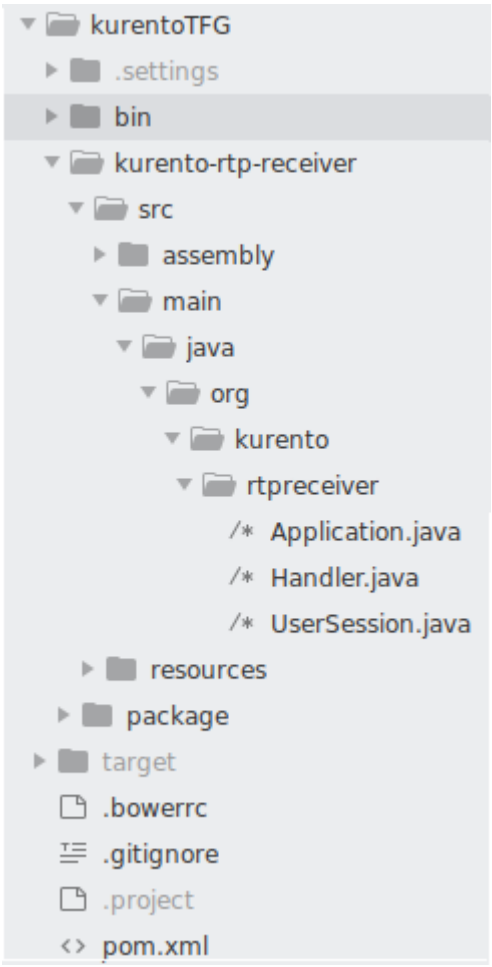

Figura 23. Estructura CarProtect Server.

Entre los ficheros que aparecen en la anterior figura cabe destacar los siguientes:

1. pom.xml: fichero XML que contiene información sobre el proyecto y los detalles de configuración utilizados por Maven para construir el proyecto. Contiene valores predeterminados para la mayoría de los proyectos. Ejemplos de esto es el directorio de compilación, que es target; el directorio de origen, que es src/main/java.

2. Application.java: Es la clase principal, en ésta, KurentoClient se instancia en esta clase como un Spring Bean. Este bean se utiliza para crear Kurento Media Pipelines, que se utilizan para agregar capacidades de medios a sus aplicaciones (Figura 24).

```
@SpringBootApplication
@EnableWebSocket
public class Application implements WebSocketConfigurer
{
  @Bean
  public Handler handler()
  {
    return new Handler();
  }

  @Bean
  public KurentoClient kurentoClient()
  {
    return KurentoClient.create();
  }

  @Bean
  public ServletServerContainerFactoryBean createServletServerContainerFactoryBean() {
    ServletServerContainerFactoryBean container = new ServletServerContainerFactoryBean();
    container.setMaxTextMessageBufferSize(32768);
    return container;
  }

  @Override
  public void registerWebSocketHandlers(WebSocketHandlerRegistry registry)
  {
    registry.addHandler(handler(), "/rtpreceiver");
  }

  public static void main(String[] args) throws Exception
  {
    SpringApplication.run(Application.class, args);
  }
}
```

Figura 24. Aplicación Server.

3. Handler.java: en este fichero se encuentra la clase Handler que se implementa TextWebSocketHandler para manejar solicitudes de texto WebSocket. La pieza central de esta clase es el método handleTextMessage. Este método implementa las acciones para las solicitudes, devolviendo las respuestas a través del WebSocket. En otras palabras, implementa la parte del servidor del protocolo de señalización. Además, es la responsable de notificar a Carprotect, a través de socket de escucha de ésta, del puerto de escucha de Kurento Media Server.

   En el protocolo diseñado existen tres tipos diferentes de mensajes entrantes al servidor: process SDP Offer, stop y addIceCandidates (Figura 25). Estos mensajes se tratan en la cláusula switch, dando los pasos adecuados en cada caso.

```
protected void handleTextMessage(WebSocketSession session,
    TextMessage message) throws Exception
{
  JsonObject jsonMessage = gson.fromJson(message.getPayload(),
      JsonObject.class);
  String sessionId = session.getId();

  log.debug("[Handler::handleTextMessage] {}, sessionId: {}",
      jsonMessage, sessionId);

  try {
    String messageId = jsonMessage.get("id").getAsString();
    switch (messageId) {
      case "PROCESS_SDP_OFFER":
        handleProcessSdpOffer(session, jsonMessage);
        break;
      case "ADD_ICE_CANDIDATE":
        handleAddIceCandidate(session, jsonMessage);
        break;
      case "STOP":
        handleStop(session, jsonMessage);
        break;
      default:
        sendError(session, "Invalid message, id: " + messageId);
        break;
    }
  } catch (Throwable ex) {
    log.error("[Handler::handleTextMessage] Exception: {}, sessionId: {}",
        ex, sessionId);
    sendError(session, "Exception: " + ex.getMessage());
  }
}
```

Figura 25. TextWebSocketHandler.

4. UserSession.java: en este fichero se encuentra la clase de Kurento UserSession, responsable de establecer las sesiones entre el CarProtect Server y el Servidor de Medios de Kurento usando el Kurento Protocol.

Para la ejecución del servidor se utiliza Maven, En el "Anexo C: Ejecución" se enumeran los pasos necesarios para llevarlo a cabo.

4.2.2 Lógica del Servidor

Se sigue una arquitectura cliente-servidor. En el lado del cliente, la lógica se implementa en JavaScript. En el lado del servidor, utilizamos Carprotect Server, basado en Spring-Boot que consume la API del cliente Java de Kurento para controlar las capacidades del servidor de medios de Kurento [17].

Para comunicar estas entidades se utilizan dos canales WebSockets:

1. Se crea un WebSocket entre el servidor creado y el cliente del navegador para implementar un protocolo de señalización personalizado.
2. Otro WebSocket se utiliza para realizar la comunicación entre el Carprotect Server y el servidor de medios de Kurento. Para ello, dicho servidor utiliza la biblioteca Kurento Java Client. Esta comunicación se realiza mediante el Protocolo de Kurento.

A continuación, se muestra el diagrama de clases del Carprotect Server:

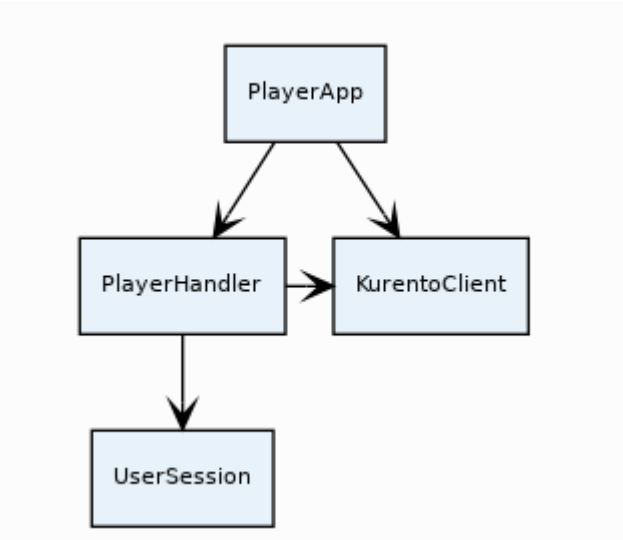

Figura 26. Diagrama de clases CarProtect Server [21].

4.2.3 Lógica del Cliente

El cliente web hace uso del protocolo WebSocket, descrito en la RFC6455 [31], brinda una forma de intercambiar datos entre el navegador y el servidr por medio de una conexión persistente. Los datos pueden ser pasados en ambas direcciones como paquetes "packets", sin cortar la conexión y sin utilizar peticiones HTTP (HTTP-Request). [32]

Se utiliza websocket para este tipo de aplicaciones porque es muy cómodo para servicios que requieren intercambio de información continua, por ejemplo, reproducción de vídeo.

A modo de ejemplo, para entender el funcionamiento de WebSocket, se muestra en la Figura 27 un diagrama ilustrativo de una conexión.

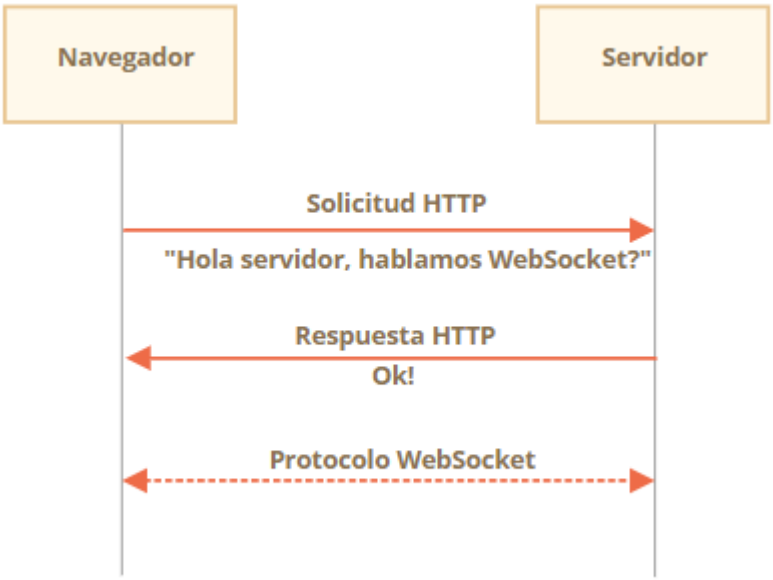

Figura 27. Diagrama ilustrativo Websocket[32].

En esta figura cuando el navegador pregunta si el servidor usa Websocket, en nuestro caso está habilitado para ello, se crea la conexión ("`new Websocket(url)`") y se comunica usando el protocolo Websocket que no es a través de HTTP. En la Figura 28 se muestra el fragmento del código desarrollado (index.js) donde se crea

la conexión Websocket.

```
const ws = new WebSocket('wss://' + location.host + '/rtpreceiver');
```

Figura 28. Línea de código Websocket.

Nuestra aplicación web desarrollada usa una biblioteca JavaScript específica de Kurento llamada kurento-utils.js para simplificar la interacción WebRTC entre el navegador y el Carprotect Server. Esta biblioteca depende de adapter.js, que es una utilidad WebRTC de JavaScript mantenida por Google que abstrae las diferencias del navegador.

Estas bibliotecas están vinculadas en la página index.html y se utilizan en el archivo index.js.

El siguiente apartado amplía la información relativa a la parte del cliente web, y se muestra el proceso de comunicación completo desde que se pulsa “start” para visualizar el contenido hasta que se recibe y se muestra el flujo de vídeo procedente de la webcam del vehículo.

5 DESARROLLO: CLIENTE WEB

El usuario final interacciona con la plataforma a través de un cliente web que le permite visualizar el interior de su vehículo. A continuación, se exponen los detalles relativos a la aplicación web.

5.1 Estructura de la aplicación web

En este proyecto se ha desarrollado una simple página web utilizando HTML, Javascript y CSS. De forma que permita al usuario visualizar el interior de su vehículo en tiempo real.

En la Figura 29 se muestra las estructuras de ficheros desarrolladas, el código estará disponible en Github.

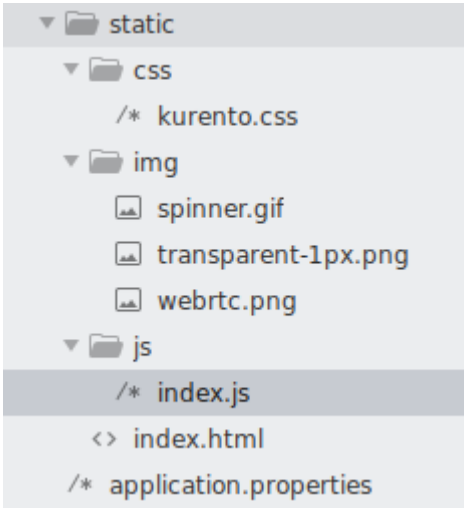

Figura 29. Estructura aplicación web

La documentación de Kurento sobre "RTP Receiver" nos ha servido como modelo tanto en el desarrollo de la aplicación como del CarProtect Server [13].

5.2 Funcionalidad

Para comunicar al cliente con el Carprotect Server Java EE se ha diseñado un protocolo de señalización basado en mensajes JSON sobre WebSocket. La secuencia normal entre el cliente y el servidor es la siguiente:

1. El cliente inicia el vídeo
2. El cliente detiene el vídeo

Se muestra el diagrama de secuencia simplificado en la Figura 30.

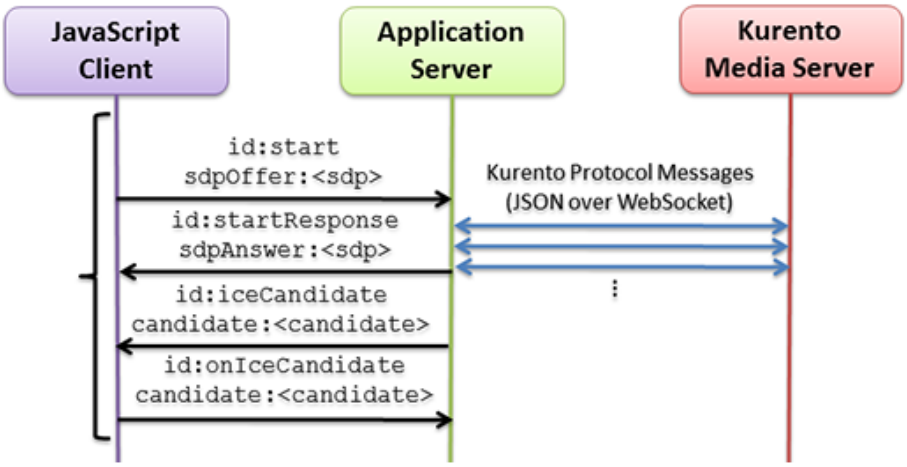

Figura 30. Diagrama de secuencia II [14].

En la Figura 31 se muestran los botones de start and stop con los que el cliente se comunica con el Carprotect Server y a su vez con los que se indican que se inicie o se pare la retrasmisión del vídeo.

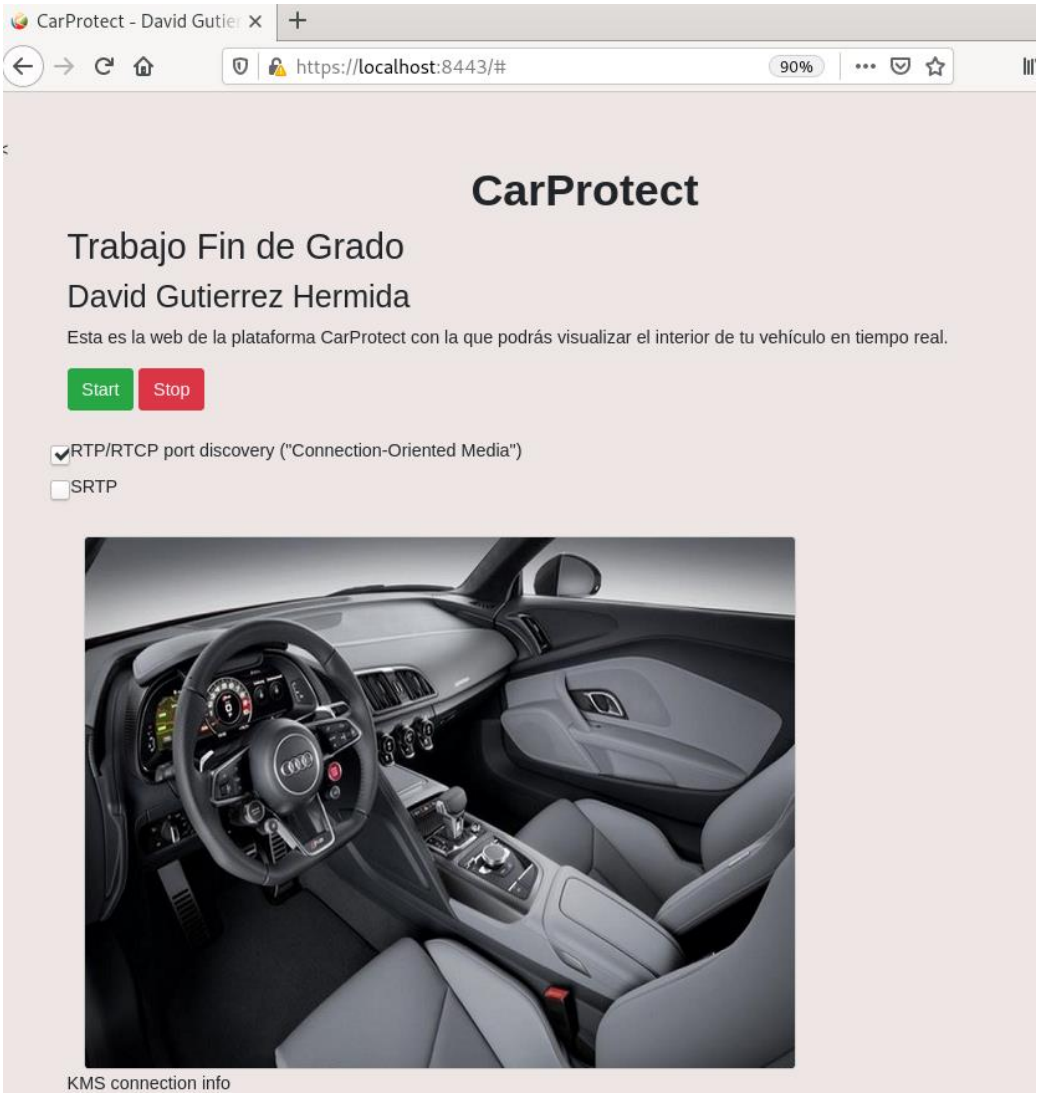

Figura 31. Web Cliente.

5.3 Caso de uso de la plataforma

En este apartado se va a explicar brevemente el funcionamiento completo de la plataforma desde que un usuario hace click en el botón start hasta que recibe el vídeo en tiempo real. Se va a mostrar el intercambio de los mensajes más relevantes en la comunicación.

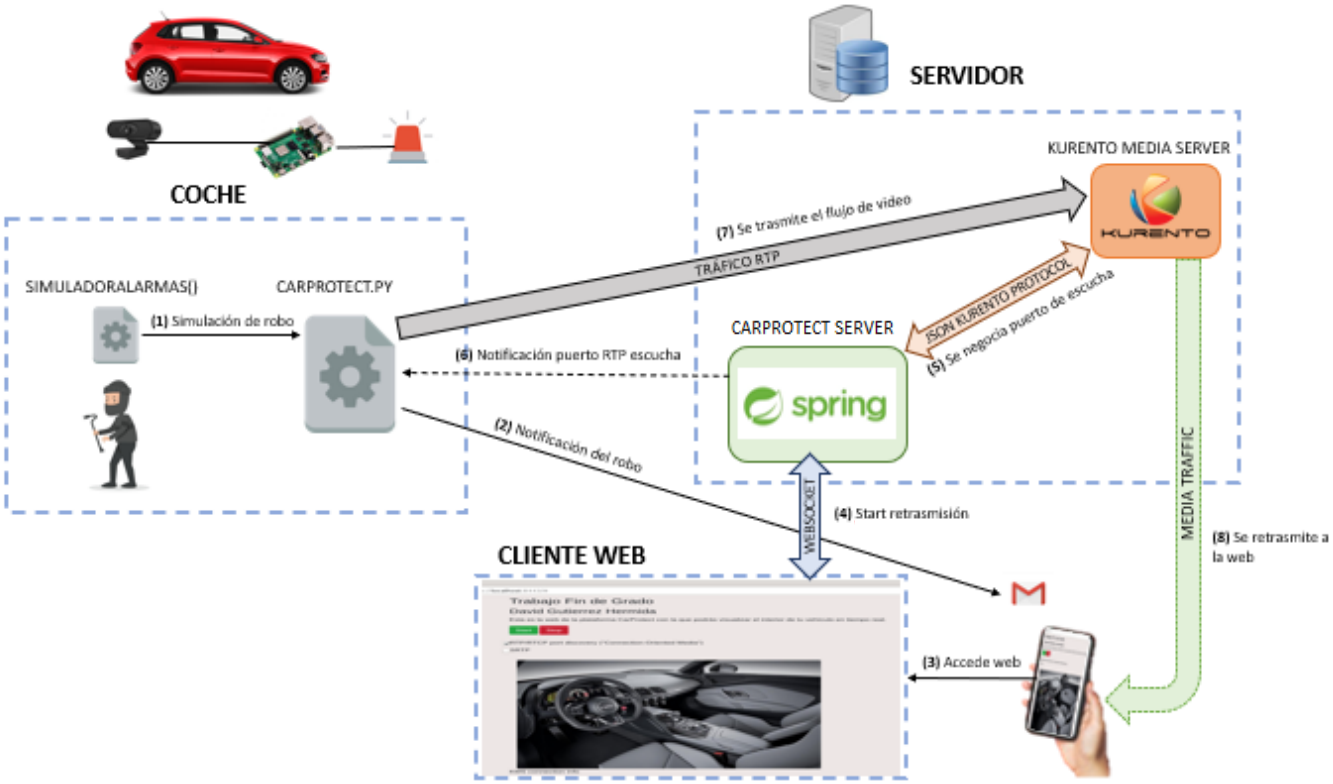

Figura 32. Comunicación completa.

1º **Simulación de robo:** función de Carprotect que permite probar el funcionamiento de la plataforma al simular una intrusión en el vehículo, para así poder notificar al usuario que se está produciendo un robo. Para empezar, se pregunta a través de la consola de comandos si se desea simular un robo, una vez que se seleccione dicha opción se da paso a la notificación del mismo.

2º **Notificación del robo:** Carprotect notifica al usuario mediante correo electrónico, previamente habrá que rellenar un fichero de configuración inicial, donde se pide diversa información como puede ser el correo electrónico al que queremos que llegue la alerta. Desde esta notificación enviada al correo del usuario se podrá acceder a la web con una url indicada en el mensaje. Destacar que para hacer esto posible se ha hecho uso de un servidor de correo electrónico (carprotect@gmail.com) que es desde donde se envía la notificación como se ha explicado en apartados anteriores.

3º **Accede Web:** una vez que el usuario reciba la notificación en uno de sus dispositivos, podrá acceder con un enlace privado, indicado en el mensaje de correo recibido, a la página web.

4º **"Start" Retrasmision:** el usuario pulsa start para comenzar a ver el vídeo. Al pulsar en el botón, internamente se hace uso de la librería kurento-utils.js (concretamente usamos los

métodos generateOffer) importada en nuestro cliente y se genera una petición al Carprotect Server (utilizando Websocket) para que comience a mostrar el vídeo.

5º **Se negocia puerto de escucha**: Carprotect Server y el servidor de medios de kurento se comunican por el protocolo propio de kurento, para así negociar los parámetros de transmisión. De esta manera el KMS buscará un puerto disponible en la máquina para poder comunicarle al servidor Spring cual es el puerto que se queda a la espera de recibir flujo de vídeo.

6º **Notificación puerto RTP escucha:** el servidor Carprotect Server, a través de un "protocolo de comunicación" creado notifica a Carprotect el puerto donde el KMS se ha quedado a la espera de recibir el flujo de vídeo. Para ello, Carprotect cuenta con un puerto a la escucha para poder recibir la información a través del "protocolo de comunicación" establecido y explicado en apartados anteriores. Una vez que en el vehículo (Carprotect) conoce el puerto del KMS donde espera recibir el flujo de vídeo ya está todo listo para empezar a transmitir.

7º **Se transmite el flujo de vídeo:** Carprotect envía el flujo de vídeo con la herramienta GStreamer, que captura el vídeo desde la webcam instalada en el coche. Envía este flujo RTP al servidor de medios de Kurento dirigido al puerto de escucha notificado previamente. Para el correcto funcionamiento de la herramienta GStreamer previamente se deberán configurar varios parámetros relativos al envío del vídeo, como pueden ser las ip's origen y destino o el "framerate" utilizado.

8º **Se retransmite a la web:** a través de Websocket, Carprotect Server muestra el vídeo en el navegador del cliente, este vídeo se lo envía el KMS al Carprotect Server usando el Protocolo de Kurento. Una vez llegados a ese punto, el usuario ya tendría acceso a la imagen en directo, capturada desde su propia webcam.

6 CONCLUSIONES Y LÍNEAS FUTURAS

To win big, you sometimes have to take big risk.

- Bill Gates -

Una vez finalizado el proyecto, se ha conseguido obtener una herramienta que podría mejorar la seguridad de nuestros vehículos. La finalidad es que el usuario pueda ser notificado y pueda visualizar el proceso en tiempo real.

Se han empleado numerosas herramientas para desarrollar el proyecto, sin embargo, la base del trabajo es Kurento.

Es obvio reseñar que este proyecto cuenta con numerosas posibles mejoras e implantaciones, es una primera versión académica de la que poder seguir avanzando y estudiando para no solo mejorarla sino para poder añadir nuevas funcionalidades que hagan que nuestros vehículos estén aún más conectados , ya sea de manera externa a las marcas, con soluciones como la que se muestran en este proyecto o bien sea con soluciones que las propias marcas de vehículos vayan aportando e implementando directamente en los coches para que quede totalmente integrado.

Cabe reseñar que hoy en día los vehículos cuentan con numerosas tecnologías en el ámbito multimedia y de seguridad en la carretera. Pero en mi opinión, creo que donde menos inversión y desarrollo se está realizando es en la propia seguridad del vehículo, ya que la mayoría de los coches actuales únicamente cuentan con alarmas de robo, tal y como era hace 20 años y solo un número muy limitado de vehículos cuentan con aplicaciones que por ejemplo, permitan tener el coche localizado y poder interactuar con él remotamente. Por este motivo se ha decidido realizar este proyecto para poder, quizás, en un futuro poder desarrollar tecnologías parecidas para evitar los temidos robos en el interior de nuestros coches.

Uno de los principales puntos fuertes de la herramienta creada es la facilidad de uso para el usuario final, ya que una vez esté todo instalado en el vehículo y configurado con los datos del usuario, no habría que hacer nada más. Únicamente cuando se produzca un robo será notificado a su correo electrónico y podrá mediante un enlace web acceder a la página donde se podrá observar el interior de su vehículo, desde cualquier lugar del mundo y desde cualquier dispositivo, lo que lo convierte en una herramienta muy interesante para su uso cotidiano para toda la población.

Otro aspecto a tener en cuenta para poder usar la herramienta es la conexión a internet, la Raspberry Pi necesita estar conectada a internet para poder enviar la notificación, así como para poder enviar el flujo de vídeo. Esto puede suponer un problema para el uso de la aplicación en vehículos más antiguos, pero no es un gran problema para vehículos más actuales ya que desde hace 4 o 5 años la mayoría de vehículos traen la posibilidad de contar con internet o vienen preparados para poder incluirles una tarjeta sim, para así contar con conexión a internet en el interior del vehículo.

A continuación, se describen las principales mejoras que podrían hacer de este proyecto una herramienta más

completa y usable en la realidad.

En primer lugar, resaltar que no se ha tenido en cuenta aspectos relacionados con la Seguridad Informática y con la Seguridad de las Comunicaciones. Se podría mejorar el cifrado de las comunicaciones entre Carprotect y el KMS, así como el cifrado de la web, que en nuestra aplicación se utiliza HTTP (Hypertext Transfer Protocol) y no HTTPS (Hypertext Transfer Protocol sobre SSL) que permite encriptar los datos para asegurar una transmisión segura.

Tampoco se han implementado mecanismos de Login en la Web, en nuestro caso cualquier usuario con la URL puede acceder y visualizar el interior del vehículo. Para una posible puesta en producción del aplicativo este sería una mejora a tener en cuenta. Introduciendo un formulario de Login y algún servidor de control de acceso.

Otra de las posibles mejoras, es la integración en el vehículo, como hemos descrito en este documento, en el coche haría falta instalar una cámara webcam y una raspberry pi donde alojar Carprotec.py para poder notificar el robo y poder retransmitir el vídeo. Está claro que a todo el mundo no le gustaría tener que instalar este hardware dentro del habitáculo por motivos de estética, una manera de solucionar esto sería integrar la aplicación Python con el software del propio sistema, la dificultad de esto recae en que cada marca realiza sus sistemas multimedia y de información lo más opacos posibles para evitar posibles modificaciones, pero mediante máquinas de diagnosis y el estudio del software podría implementarse en el propio sistema del vehículo. En el caso de la Webcam podemos buscar una solución más sencilla como “esconder” la cámara en cualquier sitio donde se grabe con buena perspectiva el interior y a la vez este en un lugar discreto.

Otra mejora importante, quizás la que más, es la implementación en el sistema de la notificación del robo, como se ha descrito, la notificación se hace de manera simulada para poder así probar el sistema, pero sería interesante estudiar la manera de que la propia herramienta recibiera la notificación real, esta podría ser la mejora más complicada de implementar, se podría hacer mediante la instalación de sensores en el vehículo, como por ejemplo sensores en los cristales que envíen una notificación cuando estos sean rotos o con sensores de sonido, con la dificultad de que habría que añadir más hardware externo al vehículo. Y la otra opción, quizás más sencilla, sería la unión de la herramienta creada con la propia alarma del coche, ya que todos los vehículos de la actualidad poseen una alarma de robo, podría integrarse dicha alarma con la aplicación para que cuando detecte un robo en el vehículo (mediante los propios acelerómetros y sensores de impacto del vehículo) se notifique a carprotect.

Una vez creada la herramienta y probada, me dispuse a llevar a cabo una prueba real en un vehículo, donde instalamos la Raspberry pi y la cámara WebCam, de esta forma se pudo realizar diferentes pruebas reales donde se realizaba la grabación dentro de un vehículo real, a continuación, se muestran imágenes de la instalación en el coche de la webcam y de la raspberry pi y una captura de pantalla donde se puede observar desde la página web creada la imagen en directo del interior del coche (Figura 33 y Figura 34).

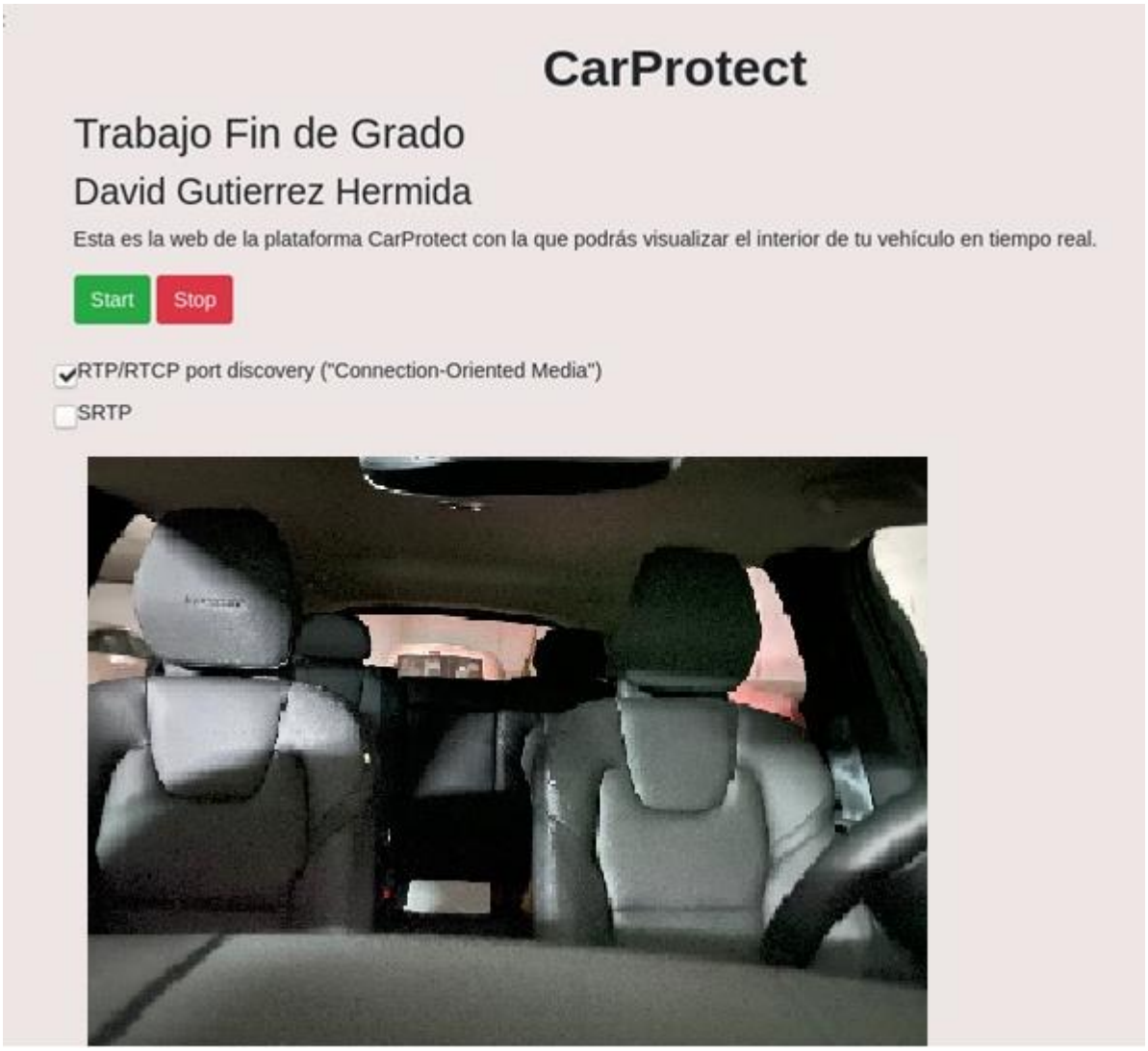

Figura 33. Imagen captada desde la webcam dentro del vehículo.

Figura 34. Montaje completo en el interior del coche

Después de finalizar el proyecto y revisar los objetivos propuestos, puedo decir que estoy satisfecho con el trabajo realizado. Se ha usado y estudiado una plataforma de tratamiento y envío de flujo de vídeo, Kurento, que ha sido la base del proyecto desde el principio. Se ha estudiado y profundizado en esta plataforma para poder luego, implementarla en un sistema de vigilancia en un vehículo. Y, por último, se han usado y profundizado en multitud de tecnologías ya conocidas y usadas y otras menos conocidas para mí. Decir que este proyecto está enfocado a la implementación en un vehículo, pero podría usarse en cualquier otro ámbito, como por ejemplo para sistemas de vigilancias de casas, negocios, etc.

Por último, decir que lo más laborioso y complicado del proyecto a mi entender, ha sido la conexión de todas las tecnologías usadas para así poder construir un módulo completo y usable, para ello ha sido necesaria en principio conocer y entender muy bien todas estas tecnologías por separado para posteriormente crear el sistema completo, donde todas ellas se comunican entre sí y consiguen lograr el objetivo marcado. Me parece un proyecto muy completo donde he plasmado multitud de conocimientos adquiridos en la carrera, donde he podido observar que todo lo aprendido tiene relación y me ha servido para crear una herramienta útil y completa.

REFERENCIAS

[1] MKM publicaciones, Imagen Dashcam, [En línea]. Avaliable: https://revistabyte.es/gadgets-smart-home/dashcam-en-tu-coche/

[2] Software in the Public Interest. Inc, Debian, Logo Debian, [En línea]. Avaliable: https://www.debian.org/index.es.html

[3] Docker, Logo Docker, [En línea]. Avaliable: https://www.docker.com/

[4] VMware. Inc, Spring, Logo Spring, [En línea]. Avaliable: https://spring.io/projects/spring-boot

[5] The Apache Software Foundation, Maven, Logo Maven [En línea]. Avaliable: https://maven.apache.org/download.cgi

[6] Kurento, Logo Kurento, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/glossary.html#term-WebRTC

[7] GStreamer, Logo Gstreamer, [En línea]. Avaliable: https://gstreamer.freedesktop.org/

[8] Python Software Foundation, Logo Python, [En línea]. Avaliable: https://www.python.org/community/logos/

[9] Oracle, Java, Logo Java, [En línea]. Avaliable: https://www.java.com/es/

[10] Html-css-js, Logo HTML, CSS y JS, [En línea]. Avaliable: https://html-css-js.com/

[11] PC Componentes, Imagen Raspberry, [En línea]. https://www.pccomponentes.com/raspberry-pi-4-modelo-b-4gb?gclsrc=aw.ds&gclid=CjwKCAjwz_WGBhA1EiwAUAxIcWcuSa6ED-YxW3-FyeRVgqBITHLW-VgCF51TOw1BmOy0ISBtFm_2ShoC4EgQAvD_BwE

[12] Amazon, Imagen Webcam, [En línea]. Avaliable: https://www.amazon.es/BCMASTER

[13] Kurento, Kurento Java Tutorial - RTP Receiver, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/tutorials/java/tutorial-rtp-receiver.html

[14] Kurento, Java - WebRTC magic mirror, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/tutorials/java/tutorial-magicmirror.html

[15] Kurento, Welcome to Kurento, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/

[16] Kurento, Glosario, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/glossary.html#term-WebRTC

[17] Kurento, Why a WebRTC media server? , [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/user/intro.html#why-a-webrtc-media-server

[18] Kurento, Kurento Media Server capabilities Image, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/user/intro.html#id6

[19] Kurento, Kurento Design Principles, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/user/intro.html#kurento-design-principles

[20] Kurento, Kurento Spring Boot, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/glossary.html#term-Spring-Boot

[21] Kurento, Server-side class diagram of the Application Servet, [En línea]. Avaliable: https://doc-kurento.readthedocs.io/en/latest/tutorials/java/tutorial-rtp-receiver.html#id1

[21] Kurento, Kurento Java tutorials, [En línea]. Avaliable: https://github.com/Kurento/kurento-tutorial-java

[22] Github, CarprotectTFG, [En línea]. Avaliable: https://github.com/carprotection/CarprotectTFG

[23] Kurento, Whats Kurento, [En línea]. Avaliable: https://www.kurento.org/whats-kurento

[24] Wikipedia, Raspberry Pi, [En línea]. Avaliable: https://es.wikipedia.org/wiki/Raspberry_Pi

[25] ComputerHoy, DashCam, que son? , [En línea]. Avaliable: https://computerhoy.com/listas/apps/dashcam-que-son-que-sirven-cuales-son-mejores-39591

[26] Xataka, Máquinas vistuales, [En línea]. Avaliable: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas

[27] 3CX, ¿Qué es RTP? , [En línea]. Avaliable: https://www.3cx.es/voip-sip/rtp/

[28] Redhat, Que son las APIs y para que sirven , [En línea]. Avaliable: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

[29] Salesforce, ¿Qué es PaaS? , [En línea]. Avaliable: https://www.salesforce.com/es/learning-centre/tech/paas/

[30] Wikipedia, Subsistema multimedia IP, [En línea]. Avaliable: https://es.wikipedia.org/wiki/Subsistema_Multimedia_IP

[31] IETF,RFC6455, [En línea]. Avaliable: https://datatracker.ietf.org/doc/html/rfc6455

[32] Javascript.info, WebSocket, [En línea]. Avaliable: https://es.javascript.info/websocket

ANEXO A: PREPARACIÓN DEL ENTORNO

En este Anexo se detallan los componentes del entorno tecnológico necesario para la correcta ejecución de los servidores y de las aplicaciones creadas. En este anexo se explica el entorno de prueba empleado tanto en el coche usando la Raspberry Pi como en la máquina virtual utilizada para alojar los servidores.

4.2 Entorno Raspberry Pi

El primer paso es la instalación del hardware, es decir, se debe conectar la webcam con la raspberry y esta última conectarse a Internet (SIM). En la Figura 35 se muestra la conexión de la webcam.

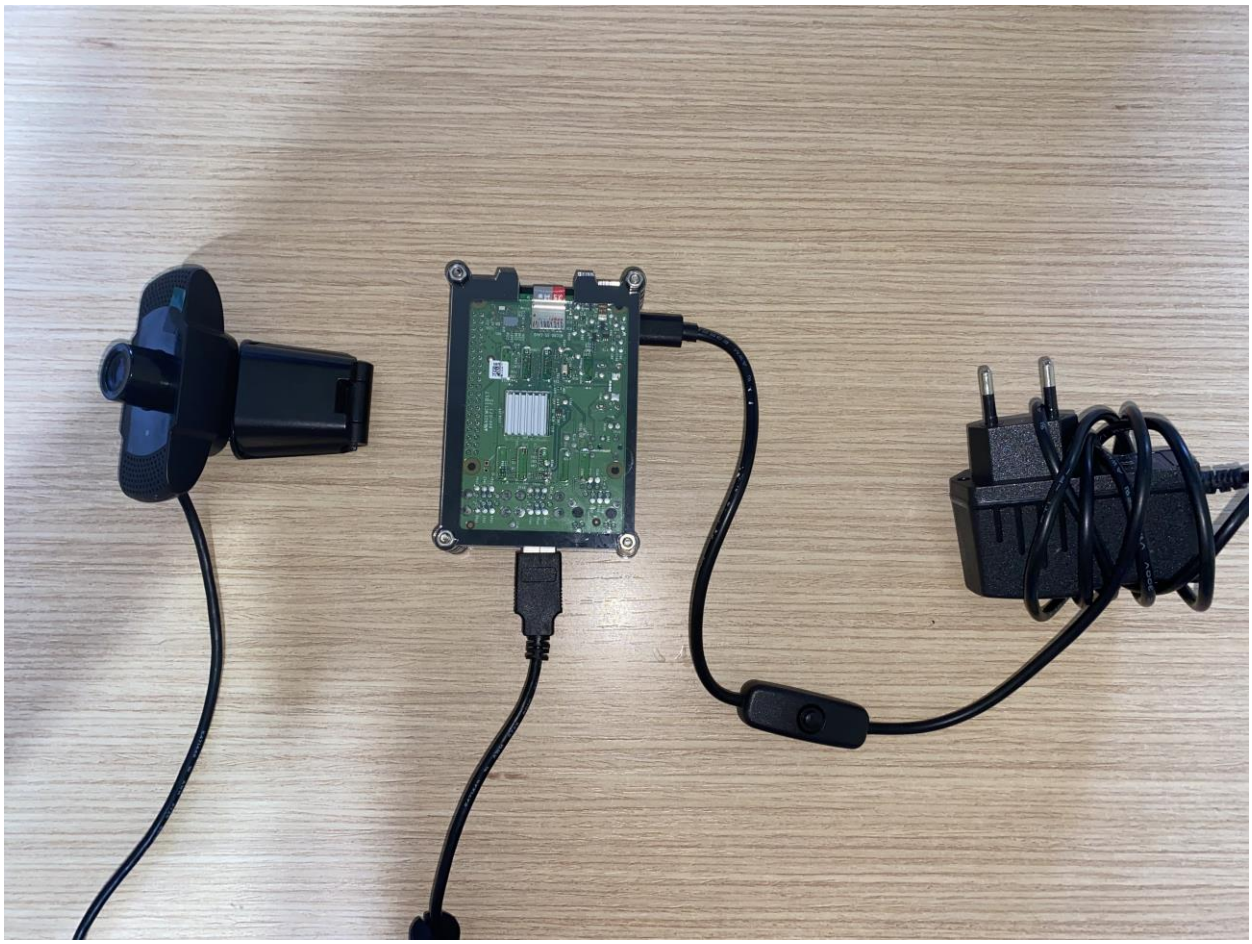

Figura 35. Conexión Webcam

En la Raspberry, con el sistema operativo Raspberry Pi OS es necesario instalar manualmente los siguientes elementos:

- Python 3

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

- Gstreamer

```
$ apt-get install gstreamer1.0-plugins-*
```

4.3 Entorno Máquina Virtual

Como se ha comentado anteriormente los servidores se han alojado en una máquina virtual Debian 10. Como Hipervisor se ha usado VMware Workstation Pro y se ha configurado para que la máquina virtual esté accesible desde la red.

La máquina virtual que se ha usado tiene los siguientes requisitos (Figura 36):

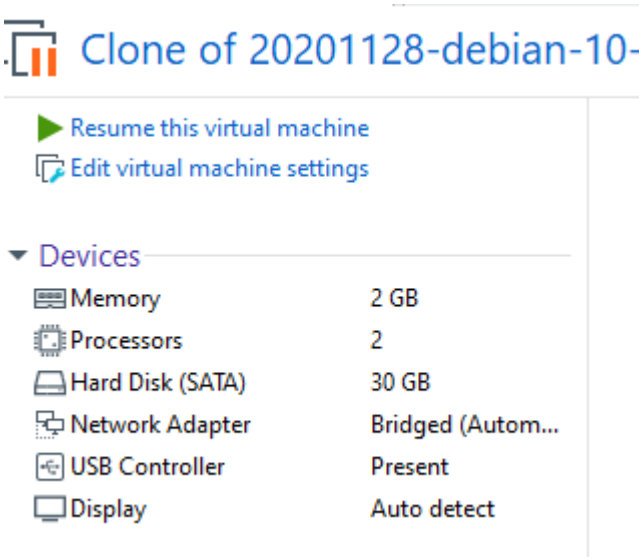

Figura 36. Requisitos de la Máquina Virtual.

Para realizar esta configuración de conectividad en la MV accedemos a lo siguiente:

1. Ajustes de la MV
2. Network Adapter (Adaptador de Red) y lo configuramos en modo Bridge como se muestra en la Figura 37.

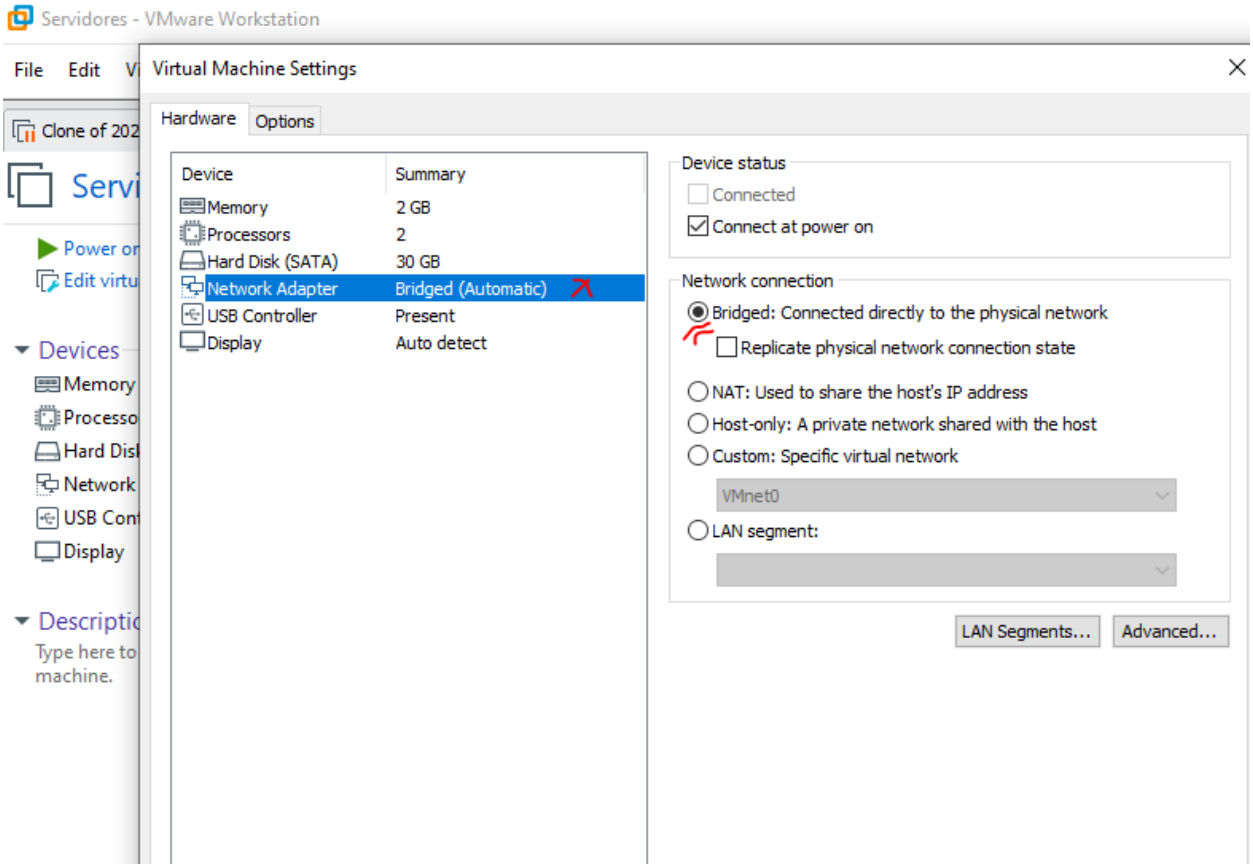

Figura 37. Configuración de red de la Máquina Virtual.

Una vez que tenemos la máquina virtual conectada a la red es necesario instalar el motor de Docker (Docker Engine) sobre el que se ejecutará el servidor de Kurento.

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl
gnupg  lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

ANEXO B: INSTALACIÓN Y CONFIGURACIÓN

En este anexo se enumeran los pasos a seguir para instalar correctamente todos los componentes de la plataforma, tanto el KMS y el CarProtect Server como Carprotect. Usaremos Github, un portal creado para alojar el código de las aplicaciones de cualquier desarrollador [22].

4.4 Instalación de Kurento Media Server

Para poder usar Kurento Media Server solo hace falta instalar la imagen de Docker:

```
$ docker pull kurento/kurento-media-server:latest
```

Desde esta imagen se puede cambiar el puerto de escucha de KMS (5000 UDP), así como el puerto por el que se comunica el Cliente (8888 TCP).

Por ejemplo:

```
$ docker run --rm \
    -p 8888:8888/tcp \
    -p 5000-5050:5000-5050/udp \
    -e KMS_MIN_PORT=5000 \
    -e KMS_MAX_PORT=5050 \
    kurento/kurento-media-server:latest
```

4.5 Instalación de los recursos Web

En este punto se enumeran los comandos tanto para la descarga del código desarrollado, como la instalación de Maven.

- Descarga de código [22]:

```
$ git clone https://github.com/carprotection/CarprotectTFG.git
```

El código se encuentra en el directorio: Kurento TFG.

- Instalación de Maven:

```
$ sudo apt install maven
```

Puede comprobar la versión con el comando:

```
$ mvn -version
```

4.6 Instalación Carprotect

No es necesaria una instalación previa para el uso de Carprotect solo descargar el código "*carprotect.py*" del repositorio. Para esto podemos ejecutar el siguiente comando

```
$ git clone https://github.com/carprotection/CarprotectTFG.git
```

Si la máquina no tiene instalado git, se puede instalar con el siguiente comando:

```
$ apt-get install git
```

O directamente descargar el archivo .zip del repositorio [22].

ANEXO C: EJECUCIÓN

En este último apartado se enumeran los comandos necesarios para ejecutar las diferentes plataformas en el entorno anteriormente descrito. Para que se ejecute de forma correcta se recomienda al lector realizar los pasos previos de Preparación del Entorno (Anexo A) y de Instalación y Configuración (Anexo B).

4.7 Ejecución de Servidores

1. En primer lugar, debemos de ejecutar el contenedor que contiene el Servidor de Medios Kurento.

```
$ docker run -d --name kms --network host \
    kurento/kurento-media-server:latest
```

2. Ejecutamos el CarProtect Server (se debe de ejecutar desde el directorio donde se encuentre los ficheros).

```
$ mvn -U clean spring-boot:run
```

4.8 Ejecución en Raspberry Pi

Una vez que tenemos los servidores en ejecución, en la Raspberry Pi instalada en el coche se debe ejecutar la aplicación Carprotect. Para ello solo debemos de ejecutar lo siguiente:

```
$ cd kurentoTFG
$ python3 carprotect.py
```

Tras la ejecución debemos introducir “SI” para generar una alarma, como se muestra en la Figura 38.

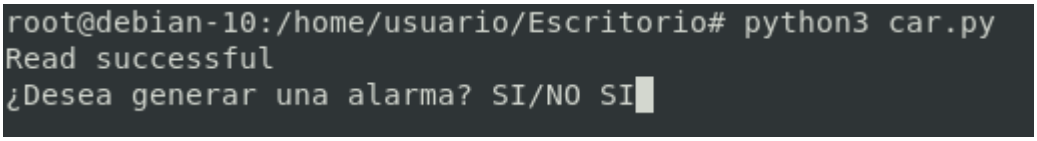

Figura 38. Carprotect ejemplo ejecución