UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DESARROLLO DE PANELES DE CONTROL PARA REDES IOT BASADOS EN NODERED

Salvador García Jiménez

Tutor: Juan Carlos Guerri Cebollada

Cotutor: Pau Arce Vila

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2018-19

Valencia, 3 de diciembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

El presente proyecto abarca la tarea de diseñar, en un entorno IoT (Internet of Things), el envío de contenido multimedia en directo. Para ello se ha escogido como interfaz una Raspberry Pi y un módulo de cámara al cual está conectada. La técnica escogida para la transmisión del contenido es DASH (Dynamic Adaptive Streaming over HTTP). Este proporciona fragmentos de vídeo que permitirán una reproducción de vídeo de bitrate adaptativo, es decir, la calidad del vídeo variará conforme lo haga la red del usuario que consume el servicio. La cámara tendrá la tarea de depositar los fragmentos en un servidor web previamente configurado, desde el cual se accederá a ellos. Todo este servicio se integrará gracias a la herramienta Node-RED, con la cual seremos capaces de diseñar un flujo de datos que nos permita cambiar el vídeo que queremos reproducir. Además, esta herramienta dispone de un espacio para mostrar un Dashboard e interactuar con el flujo de datos previamente creado. Este proyecto se aportará a la comunidad de usuarios de Node-RED con el fin de mejorar y avanzar en el ámbito del IoT y de las Smart Cities.

Resum

El present projecte comprén la tasca de dissenyar, en un entorn IoT (Internet of Things), l'enviament de contingut multimèdia en directe. Per a això s'ha triat com a interfaç una Raspberry Pi i un mòdul de cambra al qual està connectada. La tècnica triada per a la transmissió del contingut és DASH (Dynamic Adaptive Streaming over HTTP). Este proporciona fragments de vídeo que permetran una reproducció de vídeo de bitrate adaptatiu, és a dir, la qualitat del vídeo variarà conforme ho faça la xarxa de l'usuari que consumix el servici. La cambra tindrà la tasca de depositar els fragments en un servidor web prèviament configurat, des del qual s'accedirà a ells. Tot este servici s'integrarà gràcies a la ferramenta Node-RED, amb la qual serem capaços de dissenyar un flux de dades que ens permeta canviar el vídeo que volem reproduir. A més, esta ferramenta disposa d'un espai per a mostrar un Dashboard i interactuar amb el flux de dades prèviament creat. Este projecte s'aportarà a la comunitat d'usuaris de Node-RED a fi de millorar i avançar en l'àmbit del IoT i de les Smart Cities.

Abstract

This project embraces the task of design, into an IoT (Internet of Things) environment, the streaming of live multimedia content. For this task, it has been chosen a Raspberry Pi and a camera module that is connected to it. The chosen format to transmit the content is DASH (Dynamic Adaptive Streaming over HTTP). This transmission technique allows the video to be transferred fragmented in the way that offers bitrate adaptive streaming, this is that video quality will vary with client's network capacity. The camera will capture video and store fragments in a web server that has been previously configured, from which they will be accessible. All this flow of data will be processed with Node-RED, with this tool we will be able to design a data flow that allow us to change what video we want to stream. Furthermore, this tool has the property of adding a Dashboard to the design, this will give the user the possibility to interact with the data flow. This project is intended to be shared with Node-RED community as a contribution with the purpose of improve and go forward in IoT and Smart Cities environments.

Índice

Capítulo 1. Introducción

El IoT o internet de las cosas, a diferencia del internet tradicional, se define como el conjunto de interconexiones por internet de objetos cotidianos. En sus inicios, internet pretendía conectar a usuarios físicos distribuidos por todo el mundo, ahora el internet de las cosas pretende interconectar cualquier objeto que tengamos alrededor con el fin de hacer nuestra vida más fácil. Estas interconexiones pueden realizarlas cualquier elemento eléctrico al que se conecte una tarjeta de red y sus respectivos sensores, los cuales medirán y enviarán los datos pertinentes.

El internet de las cosas, también abreviado como IoT, se ha ganado estar en una posición privilegiada en cuanto a noticias tecnológicas se refiere, y no es para menos, este término es el que está causando la revolución tecnológica que vivimos hoy en día.

Tanto es así que el protocolo de direccionamiento IPv4 (direcciones de 32 bits, limitándola a 2^{32} = 4 294 967 296 direcciones únicas), el cual en un principio se pensó que sería más que suficiente para abastecer a todos los dispositivos del mundo, está llegando a su límite para dejar paso al protocolo IPv6 (direcciones de 128 bits, unas $6{,}7 \cdot 10^{17}$ direcciones por milímetro cuadrado de superficie en la Tierra), el cual promete abastecer las necesidades del IoT.

Desde un termostato inteligente con el cual controlar la temperatura de casa desde el trabajo, hasta un horno inteligente con el cual programar la cocción de la comida, pasando por una nevera que te avisa de los elementos que han caducado o faltan por comprar, se puede apreciar la envergadura que puede tener este internet en ámbitos de la domótica. Este no es el único campo donde es aplicable, sino también en hostelería, comercios de cara al público, flotas de vehículos para logística, gestión de almacenes de logística, agricultura, ganadería, jardinería, aplicaciones para monitorizar constantes de salud en la telemedicina, aplicaciones en ciudades inteligentes (gestión de suministros, gestión ambiental, gestión del tráfico, smart grid…), y un largo etcétera.

El presente proyecto pretende aplicar el internet de las cosas al ámbito de las ciudades inteligentes, diseñando un dashboard que recoge y representa datos generados por sensores, alojados en un servidor Fi-Ware y proponiendo un diseño que utiliza: Node-RED como sistema operativo e interfaz, una Raspberry Pi como sistema de bajo coste donde ejecutar las funciones de este proyecto y además un módulo de cámara compatible con la Raspberry Pi.

En conjunto, la finalidad es aportar a la comunidad de usuarios de Node-RED un flujo de datos que combina el streaming de vídeo en tiempo real con la tecnología de bitrate adaptativo DASH, que permite que cualquier servidor web pueda hacer uso de esta tecnología y cualquier usuario pueda añadir a su flujo de trabajo el conjunto de herramientas y utilidades descritas. Sería un aporte nuevo a la comunidad de usuarios de Node-RED.

1.1 Internet of Things

Internet of Things, o el internet de las cosas, abarca al conjunto de dispositivos que recientemente se han sumado a la gran maraña de dispositivos que conforma internet en sí, en los cuales no se concebía el disponer de conexiones a la red.

Es sin duda la revolución que vive Internet a día de hoy, la cantidad de dispositivos no deja de crecer, y con ella comienzan a aparecer factores que la acompañan como la seguridad de estas redes.

Para entender el impacto que tiene es necesario hablar de número de dispositivos. En 2017 existían 8,4 mil millones de dispositivos conectados a los cuales los catalogaríamos como pertenecientes del IoT [1]. Predecir el futuro es muy complicado, pero varias empresas se han atrevido a dar números. En 2020 se espera que haya 20 mil millones de “cosas” conectadas por internet según Gartner [2].

Esta red tendrá un impacto global a nivel económico también ya que promete transformar el modelo de negocio de la mayoría de empresas en un camino hacia los negocios digitales, los cuales mejoran la eficiencia y aumentarían la tasa de empleo.

El gran objetivo es hacer más fácil la vida de las personas incluyendo nuevos servicios a los objetos cotidianos del día a día.

Fig. 1: Esquema del Internet of Things

1.2 Node-RED

La cantidad de entornos a elegir cuando queremos desarrollar un proyecto en el ámbito del IoT va disminuyendo si queremos ventajas como sencillez, facilidad de uso, una interfaz gráfica intuitiva, disponer de una comunidad de usuarios y soporte. Es por esto que la herramienta a utilizar escogida es Node-RED.

La interfaz gráfica ayuda, y mucho, a entender el flujo de los datos que se manejan. La edición de dichos flujos se lleva a cabo mediante una especie de cajas llamadas nodos conectadas por hilos, las cuales pueden contener cualquier utilidad que se nos ocurra.

Además de poder manejar flujos con las herramientas que nos proporciona la herramienta existe la posibilidad de descargar nodos adicionales, provenientes la mayoría de usuarios que aportan a la comunidad sus creaciones; por lo tanto existe la posibilidad de crear un nodo completamente desde cero, diseñando cada componente al gusto del usuario.

Es una aplicación que se basa en Node.js, escrita en JavaScript y que dispone de un modelo de entradas y salidas para crear aplicaciones. Las opciones son infinitas ya que permite unir dispositivos hardware, APIs y servicios online de maneras muy interesantes.

El estar basada en Node.js aporta la ventaja de ser una aplicación que consume pocos recursos y por tanto pueda utilizarse en dispositivos de la gama más baja de coste como una Raspberry Pi o servicios cloud.

Uno de sus puntos más fuertes es la cantidad de módulos que existen, más de 225.000 [3] dan forma a esta aplicación. Además, de nada serviría tener tantos si no se pudiesen compartir fácilmente, por ello es que se guardan en lenguaje JSON, facilitando la tarea de importar, exportar o compartir dichos módulos.

Para instalar esta aplicación simplemente debemos ejecutar en la línea de comandos la siguiente cadena de texto:

```
sudo npm install -g --unsafe-perm node-red
```

Y para ejecutarla:

```
node-red
```

Automáticamente iniciará el servicio y alojará la interfaz gráfica en *http://localhost:1880*.

Una vez instalada la aplicación y ya funcionando en el servidor local, podemos empezar a utilizar Node-RED.

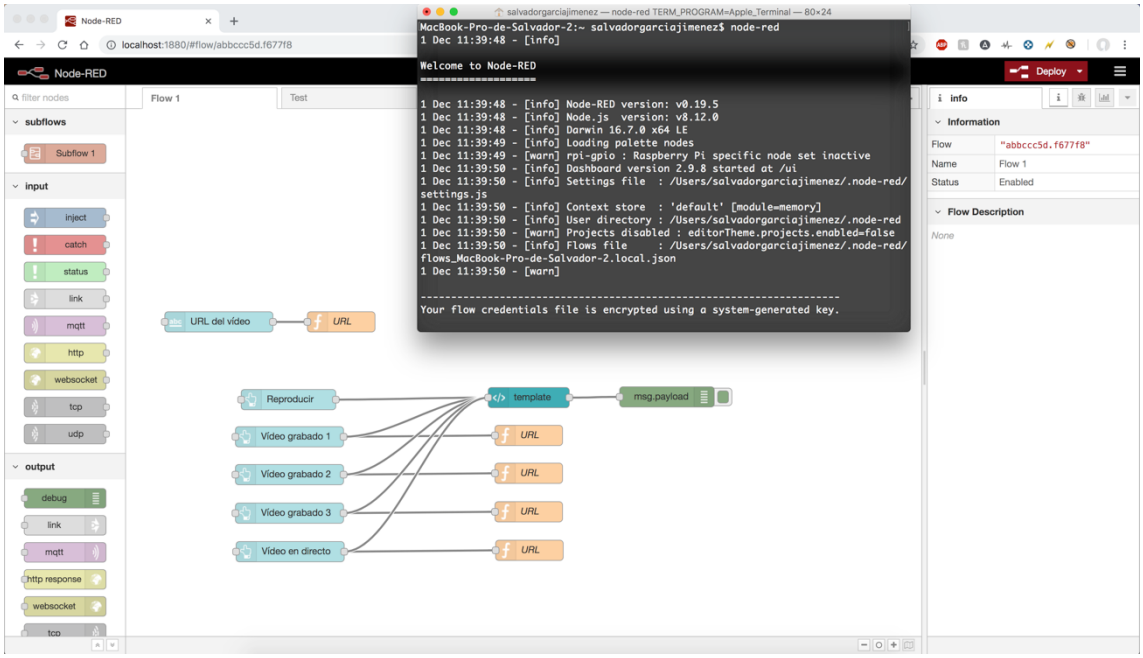

Fig. 2: Interfaz gráfica de Node-RED

En esta interfaz de usuario se pueden arrastrar multitud de nodos, un subconjunto que tiene especial importancia para este proyecto es el de Dashboard.

Este subconjunto de nodos no viene instalado con el paquete de Node-RED "básico". Para ello deberemos instalarlo del siguiente modo:

```
cd ~./node-red
npm i node-red-dashboard
```

El Dashboard de Node-RED se distribuye en una especie de rejilla, cada elemento grupo tiene un tamaño, por defecto se le asigna una anchura de 6 unidades, cada una se compone de 48 píxeles.

En cada grupo existen widgets, los cuales también poseen la propiedad de la anchura, por defecto será automática y ocupará la anchura del elemento grupo.

Existen diversos nodos que ofrecen una gran utilidad a la hora de combinar una interfaz gráfica de visualización de resultados con la interactuación entre el usuario y el flujo de datos diseñado. Hay desde formularios, botones y listas desplegables hasta templates donde programar un fragmento de lenguaje web.

Fig. 3: Ejemplo de Dashboard en Node-RED

1.3 Raspberry Pi

Una Raspberry Pi es una especie de ordenador pequeño, con el cual se pueden hacer infinidad de cosas a un coste realmente bajo. Únicamente habría que conectarla a una fuente de alimentación, además de a un monitor, ratón y teclado para poder utilizarla. El disco duro es una tarjeta microSD, donde se almacena el sistema operativo y los demás ficheros, y las conexiones a internet pueden ser bien por cable, o en las últimas versiones del dispositivo, inalámbricas.

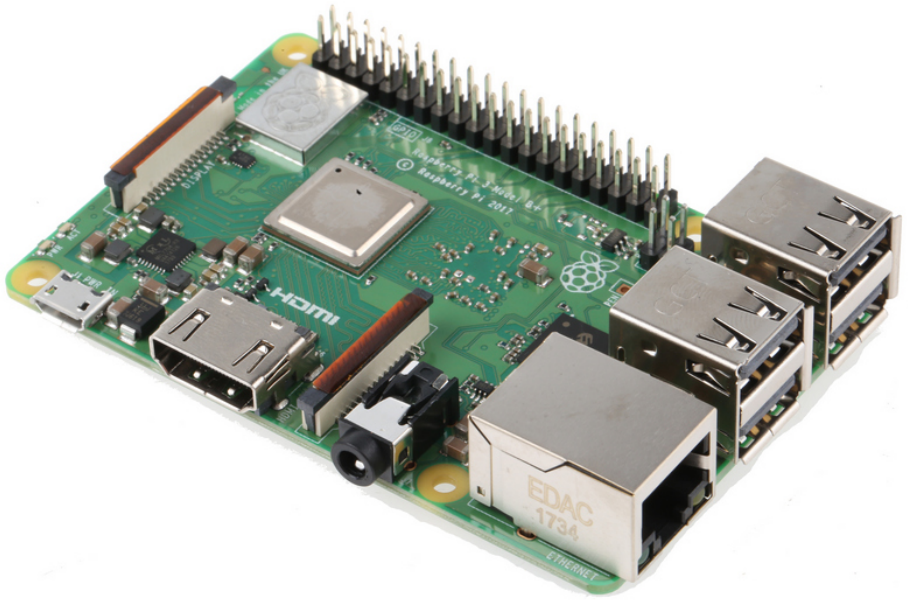

Fig. 4: Raspberry Pi 3 Modelo B+

Sus características principales son:

Procesador: Broadcom BCM2837B0, Cortex-A53 SoC de 64 bits a 1,4 GHz

Memoria: 1GB LPDDR2 SDRAM

Conectividad: 2,4 GHz y 5 GHz IEEE 802.11.b / g / n / ac inalámbrico, LAN, Bluetooth 4.2, BLE, Gigabit Ethernet sobre USB 2.0

Acceso: 4 puertos USB 2.0

Encabezado GPIO de 40 pines

Vídeo y sonido:

- 1 × HDMI de tamaño completo
- Puerto de visualización MIPI DSI
- Puerto de cámara MIPI CSI

Multimedia:

- Salida estéreo de 4 polos y puerto de video compuesto
- Decodificación H.264, MPEG-4 (1080p30); Codificación H.264

Soporte de tarjeta SD: formato Micro SD para cargar el sistema operativo y almacenamiento de datos

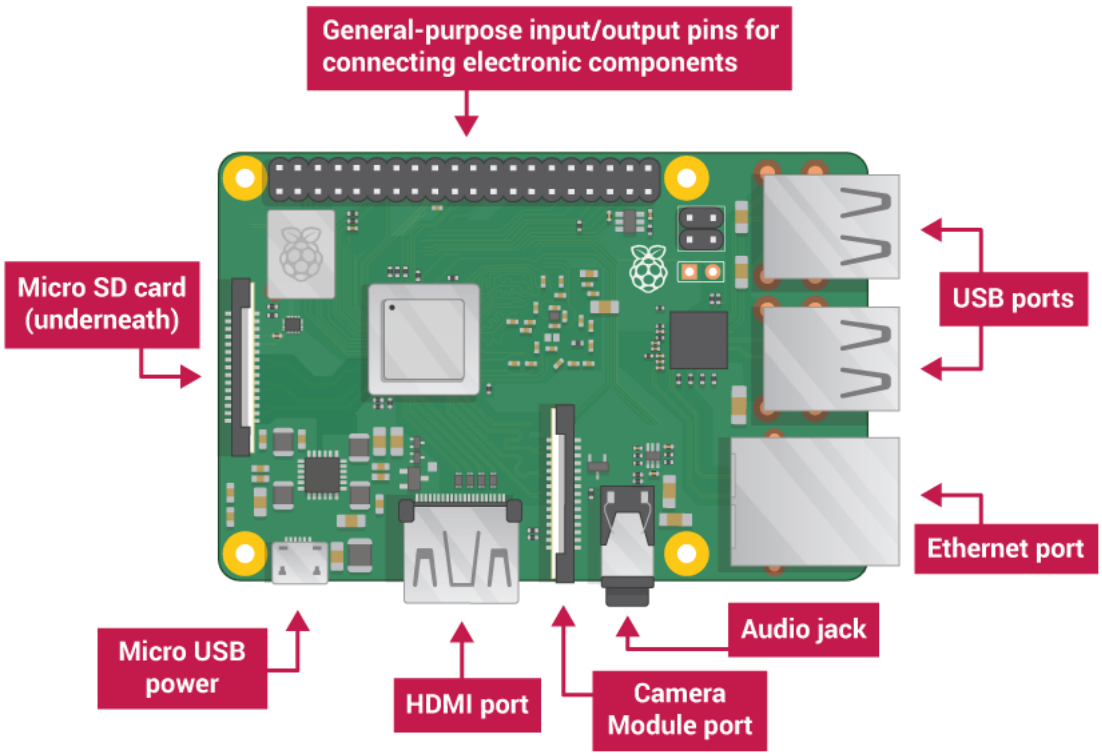

Fig. 5: Mapa de conexiones de la Raspberry Pi

1.4 Módulo de cámara V2 para Raspberry Pi

Para este proyecto se ha escogido la cámara que vende la marca Raspberry Pi por temas de compatibilidad y conexión. Se trata de un módulo externo que se conecta al dispositivo a través del puerto dedicado al módulo de cámara, el cual se encuentra en la parte superior de la misma (Fig. 5).

Dispone de una resolución de 8 megapíxeles, fabricada por Sony, la cual incorpora el sensor IMX219, el cual está diseñado para ser compatible con la Raspberry Pi. Es capaz de capturar imágenes de hasta 3280 x 2464 píxeles y de capturar vídeos en 1080p a 30fps, 720p a 60fps y 640x480 a 60 ó 90 fps.

Es una cámara pequeña (25 x 23 x 9 mm) y de poco peso (3 gramos aproximadamente), lo cual la hace perfecta para proyectos donde aspectos como el tamaño y peso son importantes.

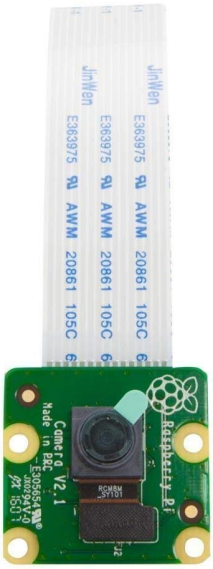

Fig. 6: Módulo de cámara V2 para Raspberry Pi

1.5 Raspbian

A la hora de llevar a cabo el presente proyecto surgió la necesidad de escoger un sistema operativo que proporcionase soporte para todas las herramientas necesarias.

Partiendo del hardware que se va a utilizar, una Raspberry Pi, los sistemas operativos a los cuales se podían optar debían estar basados en GNU/Linux como primer requisito. También se tendrá en cuenta el hecho de hacer uso de una cámara de vídeo, fabricada por la empresa que fabrica la Raspberry.

Raspbian es el sistema operativo oficial para Raspberry Pi. Es el que vamos a utilizar por ser el que más compatibilidad ofrece ya que vamos a utilizar también la cámara del mismo fabricante que la Raspberry Pi.

Es un sistema operativo formado a partir de una distribución del sistema operativo GNU/Linux, por tanto es libre y basado en Debian.

Para instalarlo simplemente hay que descargar la imagen de disco que nos proporciona la propia web de Raspberry Pi [8]. Con ayuda de un programa como Rufus para Windows o Etcher para Mac OS se monta la imagen en nuestra tarjeta microSD. Sencillamente ya sólo habría que introducir la tarjeta en la Raspberry e iniciar el dispositivo.

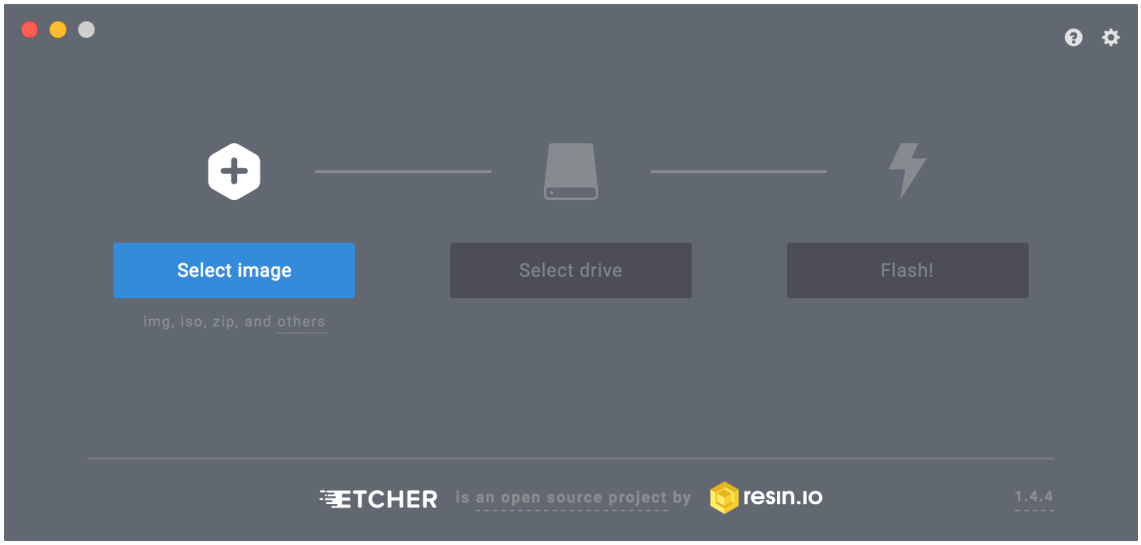

Fig. 7: Interfaz gráfica de Etcher

Está optimizado para el hardware de la Raspberry Pi e incluye instalado un navegador web (Chromium), un escritorio (LXDE), herramientas de desarrollo (IDLE) y muchas más aplicaciones, entre ellas Node-RED.

Fig. 8: Escritorio de Raspbian

1.6 MPEG-DASH

La necesidad de alcanzar múltiples plataformas y dispositivos electrónicos de consumo siempre ha sido un reto constante. En un intento de resolver los problemas que surgen en la distribución de contenido multimedia surge DASH, un estándar común para multitud de dispositivos.

La experiencia que el usuario obtiene al reproducir un vídeo con streaming de un solo bitrate es inferior al del streaming con bitrate adaptativo, ya que la calidad del vídeo variará conforme lo haga la conexión del cliente.

DASH (Dynamic Adaptive Streaming over HTTP) es una técnica de streaming de bitrate adaptativo desarrollada por MPEG, la cual proporciona streaming de contenido multimedia en alta calidad a través de Internet utilizando servidores web tradicionales, ya que funciona sobre HTTP [4].

El funcionamiento se basa en dividir el contenido que se va a transmitir en pequeños segmentos, cada uno de los cuales contendrá una fracción del contenido, que en la mayoría de los casos es una película o vídeo en directo. Además, el protocolo crea un mismo fragmento en diferentes bitrates.

Es una técnica adaptativa, por lo tanto, mientras el usuario final está consumiendo el contenido, es decisión de este el descargarse un fragmento de mayor resolución o menor, todo ello dependiendo del estado de la red.

A cada instante se evalúa la capacidad de la red y el cliente solicita la descarga del siguiente fragmento en la máxima calidad posible siempre que no interrumpa la reproducción.

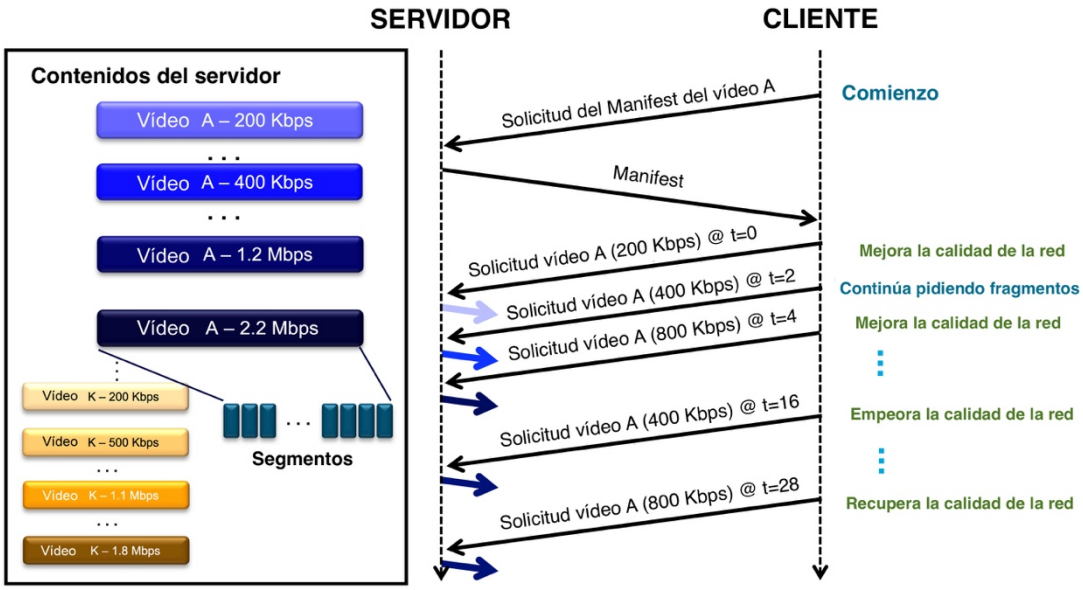

Fig. 9: Flujo de datos entre cliente y servidor utilizando DASH

Esta técnica es la primera solución de streaming de bitrate adaptativo basado en HTTP que se consolida como estándar internacional. No es un protocolo de transporte, sino una técnica, el protocolo de transporte utilizado es TCP.

El estar basado en HTTP lo hace internacionalmente compatible con cualquier web, además, esta técnica puede trabajar con contenido codificado con cualquier formato como H.265, H.264… etc.

El MPD (Media Presentation Description) es el fichero que describe la información del segmento tal como la temporización, los enlaces de los que proviene el vídeo, las resoluciones y bitrates.

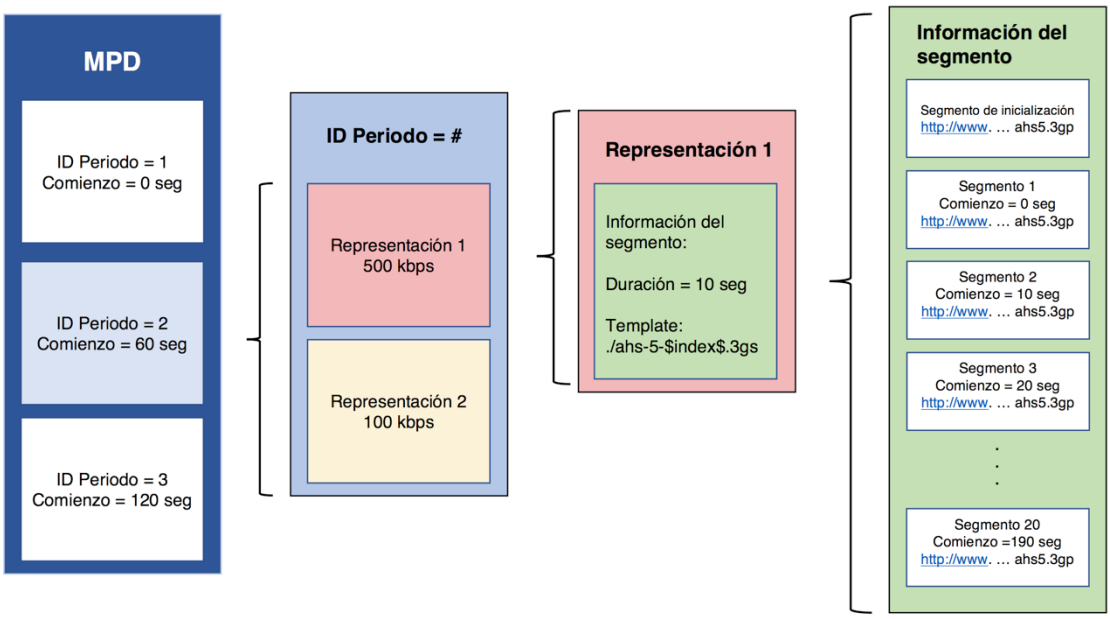

Fig. 10: Estructura de un archivo MPD

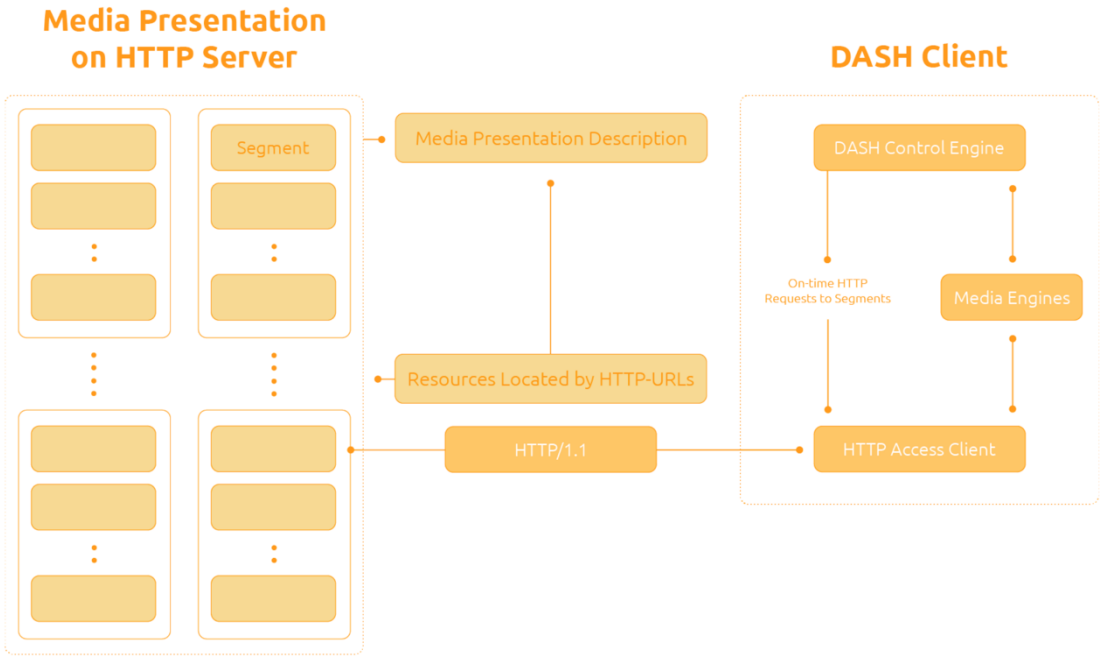

Fig. 11: Esquema genérico del funcionamiento de DASH [6]

Para reproducir el contenido el cliente obtiene el MPD, a través del cual obtiene los tiempos, resoluciones, anchos de banda y demás datos que usará para descargar siguientes segmentos y solicitar el adecuado mediante peticiones HTTP GET. De este modo el cliente irá escogiendo fragmentos de mayor o menor bitrate, para que en conjunto no superen la capacidad de su red.

Es una técnica que se ha desarrollado teniendo un único fin, tener una buena QoE (Quality of Experience). Esta se define como el grado de satisfacción de un cliente al consumir un servicio.

En este caso sería el cliente que consume contenido multimedia y prefiere que el vídeo se vea un poco peor si la conexión empeora, a que el vídeo se pause continuamente.

La implementación de esta tecnología está presente en plataformas mundialmente extendidas como YouTube o Netflix.

No es directamente compatible con HTML5, por lo que existen soluciones en JavaScript que permiten a los buscadores utilizar MPEG-DASH. Una de ellas, la que vamos a utilizar, es Shaka Player.

1.7 Shaka Player

Shaka Player es el reproductor que se ha escogido para este proyecto. Este pertenece al conjunto de software de código abierto de Google, siendo una librería que nos proporciona un reproductor DASH.

Es capaz de reproducir formatos adaptativos de vídeo tales como DASH o HLS en un buscador, sin utilizar plugins ni Flash. En vez de eso, Shaka utiliza estándares web. Esto facilita la tarea de reproducir audio y vídeo de manera adaptativa utilizando buscadores de última generación.

Es una librería que no tiene dependencias fuera de ella, por lo que a la hora de compilar una aplicación, todo lo que necesitas está en el propio código de las librerías.

Soporta cualquier buscador que soporte los estándares web más básicos, y está probado en [5]:

- Chrome
- Chromecast
- Firefox
- Microsoft Edge
- IE 11
- Safari
- Widevine
- PlayReady

Soporta las técnicas de DASH siguientes [5]:

- VOD (Video on Demand), vídeo en directo y grabaciones que son vídeos en directo durante la grabación, pero acaban como VOD al finalizar
- MPD@timeShiftBufferDepth, que se utiliza para volver atrás en vídeos en directo
- Contenido de múltiples periodos (estático y dinámico, o sea, grabado y en directo), que permite fragmentar el contenido del manifiesto DASH, definiendo una hora de inicio y duración
- Elementos Xlink (enlaces en lenguaje XML)
- Todas las formas de indexar información sobre un segmento (p.ej. SegmentBase@indexRange se utiliza cuando tienes un segmento dentro de una representación, si hay más de un segmento se utiliza SegmentList y Segment Template, estos atributos apuntan a las Segment Index Box que contienen información de subsegmentos)
- Manifiestos multi-códec o multi-contenedor
- Contenido encriptado
- Key rotation (se utiliza principalmente en cambios de claves criptográficas durante la emisión de contenido)
- Trick modes (útiles para rebobinar o avanzar un vídeo, seleccionando un subconjunto de imágenes que mostrar)

Capítulo 2. Objetivos

El objetivo principal del presente proyecto es comprender el funcionamiento de un sistema de emisión de vídeo en directo en un entorno que utiliza la tecnología DASH, una Raspberry Pi y un módulo de cámara como hardware y una programación de los requisitos para llevar a cabo dicha emisión. Tras todo lo anterior, el vídeo se representará en un Dashboard compuesto por un flujo de datos, que a su vez se compone de distintos nodos en la plataforma Node-RED.

De este modo se consigue tener una visión global de lo que requiere un servicio de este tipo y se utilizan tecnologías de última generación, presentes en los servicios ofrecidos hoy en día.

En primer lugar, se deberá configurar el hardware a utilizar, en este caso la Raspberry, con la instalación de un sistema operativo con el cual trabajar.

Será necesario un servidor web en el que poder alojar el vídeo en directo, en nuestro caso hemos elegido NGINX.

Deberá configurarse la cámara con el fin de que el sistema operativo la detecte como un dispositivo al que poder llamar.

Se crearán servicios que se ejecutan en el *systemd* de la Raspberry Pi.

Otro requisito para poder reproducir el vídeo es diseñar los nodos en los cuales se va a procesar la reproducción del vídeo, por lo tanto, se progamará el aspecto del vídeo a nivel HTML y JS.

Una vez se tiene el software listo se deberán instalar requisitos como FFmpeg para la codificación hardware de vídeo, en nuestro caso OMX.

Además, se utiliza la plataforma Node-RED, la cual ofrece infinidad de posibilidades a la hora de crear una herramienta. Esta ofrece también un repositorio enorme de aportes de sus usuarios, el cual se realimenta, ya que es muy interesante el mejorar los aportes existentes o incluir nuevas ideas.

Capítulo 3. Metodología

La metodología que se va a seguir en este proyecto es la de ir confeccionando las diferentes partes y requisitos al mismo tiempo. De este modo se van añadiendo características a los resultados que vamos obteniendo. Es decir, partimos de un funcionamiento básico del servicio para ir mejorándolo, adaptándolo a nuestras necesidades y finalmente representarlo gráficamente para facilitar la visualización de este.

3.1 Gestión del proyecto

El presente proyecto se gestiona de manera que al tener un resultado final en el proyecto que funciona, se pasa a la memoria. Tal y como se ha explicado en el apartado anterior este proyecto ha ido cambiando continuamente con el objetivo de seguir mejorando el resultado, por lo tanto, no sería recomendable redactar la memoria al mismo tiempo, ya que estaría en continuo cambio, perdiendo así el trabajo realizado para versiones del proyecto anteriores.

3.2 Distribución en tareas

En primer lugar, la tarea que se tuvo que realizar fue aprender a utilizar la plataforma Node-RED. Gracias a un curso de la web Coursera [7] llamado "A developer's guide to the Internet of Things" se adquirieron los conocimientos básicos de uso de la herramienta, además de comenzar a programar el flujo de datos de diferentes formas.

Una vez hecho esto, lo siguiente era adquirir una Raspberry Pi para empezar a trabajar con ella y la plataforma Node-RED, ya que hasta ahora solo se había utilizado Node-RED en el ordenador personal.

Con la Raspberry Pi disponible para su uso, lo siguiente era comenzar el trabajo dentro de ella, tal como instalación de prerrequisitos, servidores, aplicaciones y la consiguiente programación para que todo funcione.

Una vez programado todo, simplemente quedaba pasar el resultado a una interfaz gráfica que permitiese interactuar con el proyecto de manera visual e intuitiva.

Finalmente, todos los resultados se pasarían a una memoria donde se recoge con detalle cada paso realizado.

3.3 Diagrama temporal

En este proyecto se ha escogido un diagrama que engloba tres periodos bien marcados: el primero consiste en la preparación para la realización de este mediante un curso explicativo, el segundo trata del desarrollo del proyecto dentro del ecosistema Raspberry Pi y programando cada uno de los requisitos del servicio y por último, el tercero sería el desarrollo de la memoria, en el cual se recogería la última versión del trabajo realizado experimentalmente con herramientas software.

Capítulo 4. Desarrollo y resultados

En el apartado de desarrollo y resultados se va a explicar el proceso de desarrollo del proyecto desde el momento en que se empieza a utilizar la Raspberry Pi.

Una vez ya tenemos la Raspberry Pi preparada para su uso, tal y como hemos explicado en el apartado de la introducción, debemos comenzar a añadir a la Raspberry Pi los requisitos necesarios para el funcionamiento del proyecto.

Para hacer más cómoda la tarea del trabajo con la Raspberry Pi, se han utilizado herramientas como conexión SSH o una aplicación llamada VNC Viewer/Server. Ya que sin estas herramientas la Raspberry Pi exige que la utilices con un teclado, ratón, monitor… etc. Esto por lo tanto obliga a no poder trabajar fuera del espacio de trabajo, que habitualmente es nuestra casa.

El hecho de utilizar SSH facilita la tarea de trabajar desde nuestro ordenador personal, sin siquiera estar en casa, con una conexión SSH remota, o en el caso de estarlo, con una conexión SSH local.

Esta conexión debe habilitarse desde el panel de preferencias de la Raspberry Pi que encontramos en Raspbian.

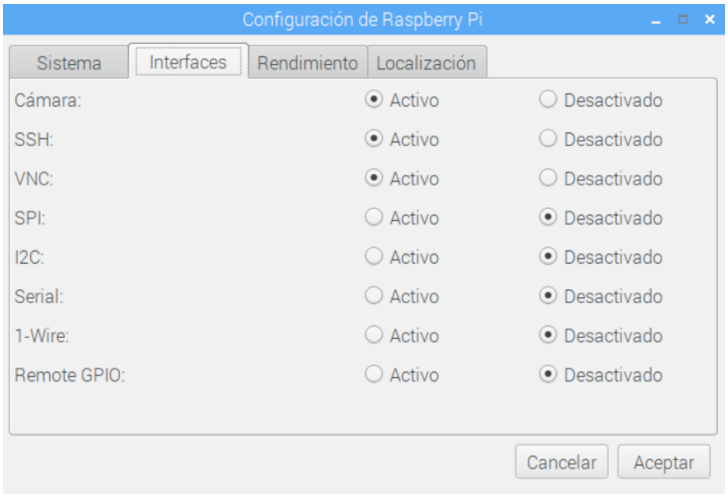

Fig. 12: Configuración de interfaces de Raspberry Pi

Para conectarse simplemente escribimos en nuestro ordenador personal:

```
ssh pi@<dirección IP>
```

Nos pedirá la contraseña de usuario de la Raspberry Pi, al introducirla y pulsar Enter ya estaríamos conectados.

Para finalizar la conexión simplemente ejecutamos el comando *exit*.

```
MacBook-Pro-de-Salvador-2:~ salvadorgarciajimenez$ ssh pi@192.168.1.140
pi@192.168.1.140's password:
Linux raspberrypi 4.14.34-v7+ #1110 SMP Mon Apr 16 15:18:51 BST 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec  2 12:03:40 2018 from 192.168.1.136
pi@raspberrypi:~ $ exit
logout
Connection to 192.168.1.140 closed.
MacBook-Pro-de-Salvador-2:~ salvadorgarciajimenez$
```

Fig. 13: Ejemplo de conexión local a Raspberry Pi

Para la conexión remota se deberán abrir los puertos del router pertinentes que redirijan a la IP de la Raspberry Pi al puerto 22 (SSH). De este modo el NAT sabe que en una conexión a tu IP pública en un puerto X (p.ej. 2022) se deberá redirigir a la IP de la Raspberry Pi en el puerto 22.

Además de esta solución, la más completa es una aplicación que nos proporciona una imagen en tiempo real del escritorio de nuestra Raspberry Pi, es una conexión remota con interfaz gráfica, igual que si estuviésemos utilizando la Raspberry Pi en casa. Esta aplicación es VNC Server en la Raspberry Pi y VNC Viewer en nuestro ordenador personal.

Se trata de una aplicación que mediante internet envía la pantalla de la Raspberry Pi en tiempo real, con el único requisito de registrarse en la plataforma, se inicia sesión en la Raspberry Pi y en el ordenador persona, y listo. Esta aplicación dispone de medidas de seguridad como la introducción de una contraseña para la conexión a la Raspberry Pi, por lo que se comporta de manera parecida al SSH en este sentido.

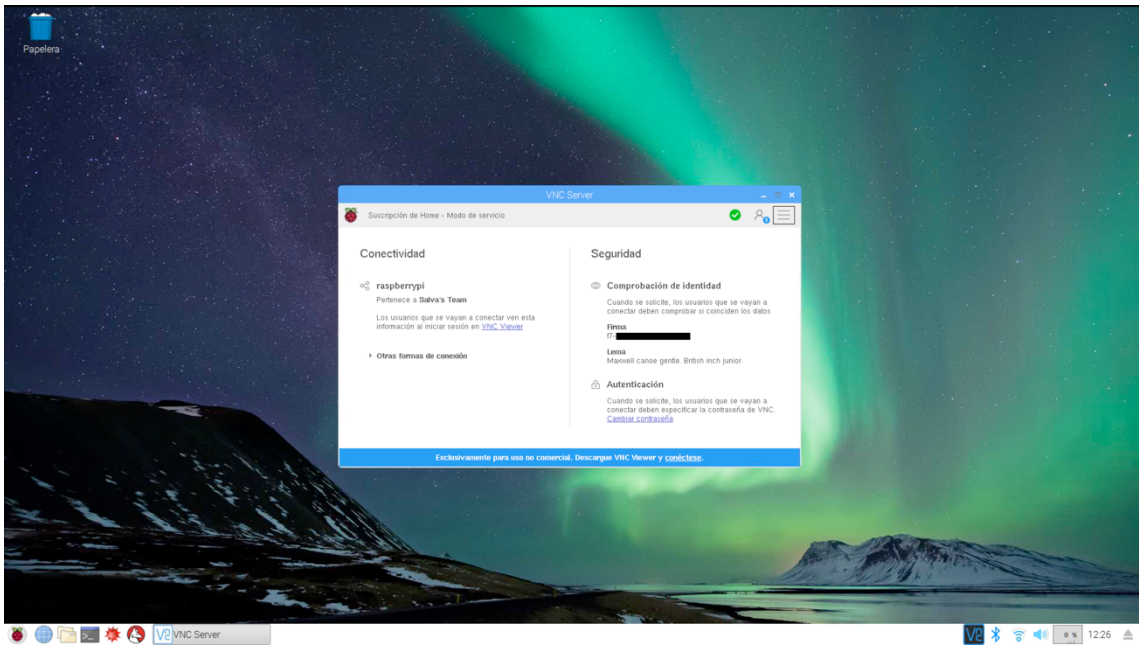

Fig. 14: VNC Server en Raspberry Pi

La captura de pantalla de la Figura 14 se ha tomado a partir de una conexión ya realizada.

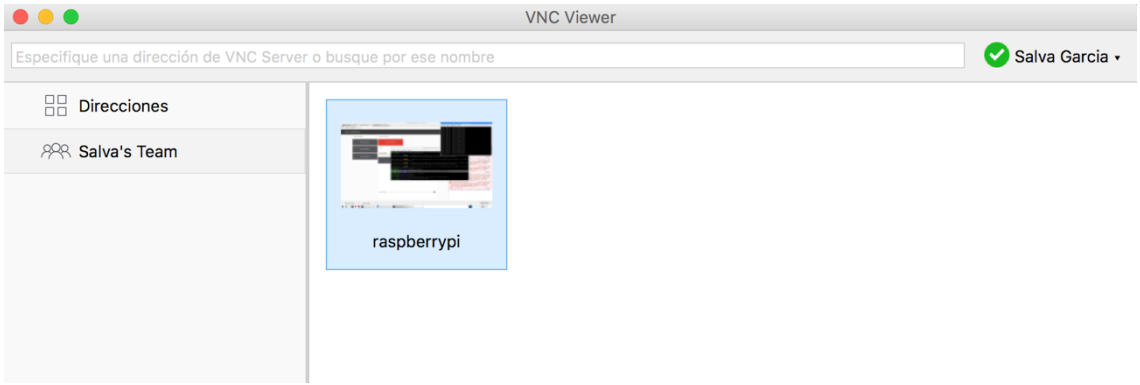

Fig. 15: VNC Viewer en nuestro ordenador personal

4.1 FFmpeg

Para este proyecto si existe una herramienta que no puede ser reemplazada por otra esa es FFmpeg. Es el framework multimedia por excelencia, es capaz de decodificar, codificar, multiplexar, demultiplexar, emitir, filtrar y reproducir prácticamente cualquier formato que exista. Desde el más antiguo hasta el más moderno.

Es una herramienta muy versátil y que funciona a la perfección en cualquier sistema operativo basado en Linux, Mac OS, Microsoft Windows… etc.

Además incluye librerías de códecs, escalado de imagen, multiplexación y demultiplexación, filtros y dispositivos, los cuales pueden ser utilizados por las aplicaciones. No solo esto, sino también los comandos ffmpeg, ffplay y ffprobe, que se pueden utilizar para convertir formatos, reproducirlos y analizarlos, respectivamente [9].

En nuestro caso FFmpeg va a realizar la tarea de codificar el vídeo grabado mediante codificación hardware y convertirlo a un formato compatible con DASH. Para esto antes de nada se debe incluir el servicio FFmpeg en nuestra Raspberry Pi, para ello se instalarán las herramientas que utilizará FFmpeg:

En primer lugar de instalarán las librerías de H.264:

```
cd /<ruta de instalación elegida por el usuario>
sudo git clone git://git.videolan.org/x264
cd x264
sudo ./configure --enable-static
sudo make -j4
sudo make -j4 install
```

Para ello abrimos la carpeta donde queramos instalar el paquete de H.264 y clonamos el repositorio en git de H.264 que contiene todas las librerías que queremos. Una vez clonado, abrimos la carpeta donde se ha clonado y ejecutamos el comando de configuración que habilita las librerías de forma estática. Tras esto se compila y se instala con la configuración proporcionada. La opción `-j4` hace que la Raspberry Pi compile con los 4 núcleos que tiene, en el caso de la Raspbery 3B+ que estamos utilizando, acelera el proceso.

Con H.264 instalado ahora hay que instalar la librería que da soporte para la codificación hardware de OpenMAX. Esta codificación hace que el vídeo se genere gracias a la GPU de nuestra Raspberry Pi, de otro modo no funcionaría correctamente, ya que no codificaría a tiempo real y la emisión de contenido no sería posible. Se necesita ejecutar la librería *bellagio*, la cual es una implementación de OpenMAX de código abierto y que funciona en Linux:

```
sudo apt-get install libomxil-bellagio-dev
```

Una vez se tienen instaladas las herramientas de las que va a hacer uso FFmpeg hay que instalar la propia herramienta, de un modo similar a H.264 se ejecuta:

```
cd /<ruta de instalación elegida por el usuario>
sudo git clone https://github.com/FFmpeg/FFmpeg.git
cd ffmpeg
sudo ./configure --enable-libx264 --enable-mmal --enable-omx --enable-omx-rpi
sudo make -j4
sudo make -j4 install
```

Se abre la carpeta donde queramos instalar el paquete y se clona el contenido de FFmpeg dentro. A continuación, dentro de la carpeta ejecutamos el comando configure para especificar que queremos habilitar la codificación hardware proporcionada por OpenMAX. Finalmente se compila e instala todo conforme a las preferencias especificadas. La opción `-j4` hace que la Raspberry Pi compile con los 4 núcleos que tiene, en este caso, es una tarea de mucha duración por lo que es altamente recomendable, si se dispone de uno de los últimos modelos de Raspberry Pi, utilizarlo.

Para comprobar que está correctamente instalado se puede ejecutar `ffmpeg`:

```
pi@raspberrypi:~ $ ffmpeg
ffmpeg version N-92313-g4a6d5f3cad Copyright (c) 2000-2018 the FFmpeg developers
  built with gcc 6.3.0 (Raspbian 6.3.0-18+rpi1+deb9u1) 20170516
  configuration: --arch=armel --target-os=linux --enable-gpl --enable-libx264 --enable-nonfree --en
able-gpl --enable-nonfree --enable-mmal --enable-omx --enable-omx-rpi
  libavutil      56. 21.100 / 56. 21.100
  libavcodec     58. 34.100 / 58. 34.100
  libavformat    58. 19.102 / 58. 19.102
  libavdevice    58.  4.106 / 58.  4.106
  libavfilter     7. 38.100 /  7. 38.100
  libswscale      5.  2.100 /  5.  2.100
  libswresample   3.  2.100 /  3.  2.100
  libpostproc    55.  2.100 / 55.  2.100
Hyper fast Audio and Video encoder
usage: ffmpeg [options] [[infile options] -i infile]... {[outfile options] outfile}...
```

Fig. 16: Comprobación de la instalación de FFmpeg

Para comprobar si está disponible la codificación hardware se puede ejecutar `ffmpeg -encoders 2>/dev/null | grep h264_omx`:

```
pi@raspberrypi:~ $ ffmpeg -encoders 2>/dev/null | grep h264_omx
 V..... h264_omx             OpenMAX IL H.264 video encoder (codec h264)
```

Fig. 17: Comprobación de la existencia de OMX entre los códecs

Más adelante se demostrará por qué es tan interesante la opción de la codificación hardware.

4.2 NGINX

A la hora de comprobar que los fragmentos se generaban de manera correcta y que son compatibles con un navegador fue necesaria la instalación de un servidor web en la Raspberry Pi.

NGINX es la opción que se ha escogido para tal tarea. Este es un servidor web de código abierto ligero de alto rendimiento y además puede hacer de proxy inverso, cache del HTTP y balanceador de carga. Es famoso por ser fácilmente configurable y por su bajo consumo de recursos del dispositivo que lo ejecuta.

La instalación de este servidor es tan sencilla como ejecutar estas dos líneas:

```
sudo apt-get update
sudo apt-get install nginx
```

Se editará el fichero de configuración `nginx.conf` ubicado en la ruta de instalación para guardarlo del siguiente modo:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
	worker_connections 1024;
}

http {
	sendfile on;
	tcp_nopush on;
	tcp_nodelay on;
	keepalive_timeout 65;
	types_hash_max_size 2048;
	server {
		listen       80;
		server_name  localhost;

		location / {
			root   /var/www/html;            #ruta del servidor web
			index  index.html index.htm;
		}
		error_page   500 502 503 504  /50x.html;
		location = /50x.html {
		    root   /usr/share/nginx/html;
		}
	}

	include /etc/nginx/mime.types;
	default_type application/octet-stream;

	ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
	ssl_prefer_server_ciphers on;

	access_log /var/log/nginx/access.log;
	error_log /var/log/nginx/error.log;

	gzip on;
	gzip_disable "msie6";

	include /etc/nginx/conf.d/*.conf;
	include /etc/nginx/sites-enabled/*;
}
```

De este modo se le indica la ruta donde se va a alojar la página web.

Mediante los comandos enable y start se le dice al servicio que se habilite o se ponga en marcha:

```
sudo systemctl enable nginx
sudo systemctl start nginx
```

El tener un servidor web añade la funcionalidad de poder acceder a los fragmentos generados por la cámara desde fuera del entorno local.

4.3 Configuración de la cámara

La cámara, al conectarse a la Raspberry Pi por el puerto dedicado a ello, no se reconoce en ella automáticamente. Para ello se debe ejecutar el siguiente comando, el cual activa el módulo *video4linux* para el chipset de la cámara:

```
sudo modprobe bcm2835-v4l2
```

De este modo ya aparecería como dispositivo la cámara, pero tiene un inconveniente, y es que cada vez que se reinicia la Rapsberry Pi hay que volver a ejecutar este comando.

Para solucionar este problema se ha modificado el fichero *modules* que se encuentra en el directorio */etc*:

```
sudo nano /etc/modules
bcm2835-v4l
```

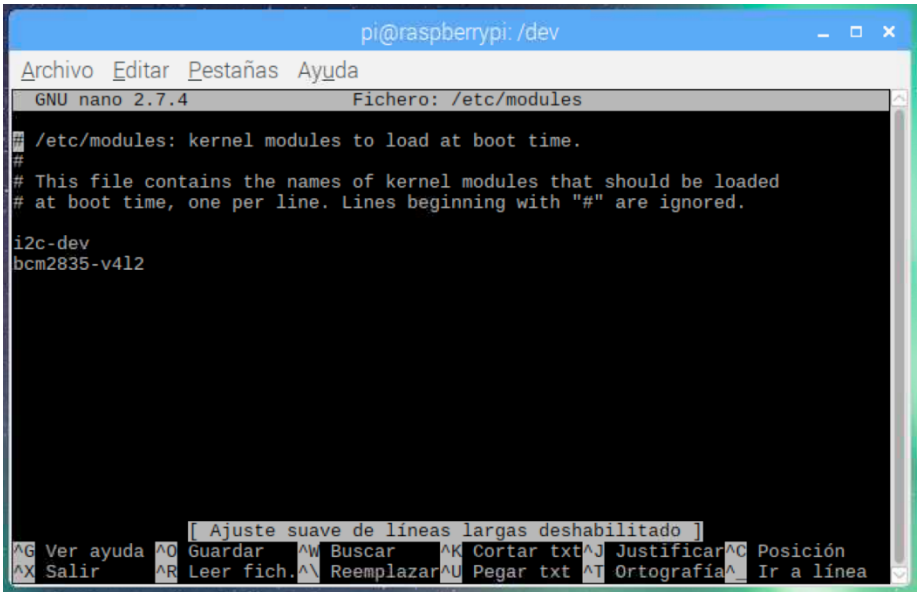

Fig. 18: Modificación de los módulos del kernel

De este modo cada vez que se inicia la Rapsberry Pi se ejecuta el comando que queremos.

4.4 Integración de servicios

Una vez tenemos preparadas las herramientas a utilizar deberemos comenzar a diseñar la solución final al presente proyecto.

En primer lugar se crearán un usuario y grupo de trabajo en la Raspberry Pi, desde los cuales ejecutar los comandos de emisión de vídeo en directo.

Creamos el fichero `camera.rules` dentro del directorio `/usr/lib/sysusers.d/ffmpeg.conf` del siguiente modo:

camera.rules

```
u ffmpeg
m ffmpeg video
```

A continuación se crean los usuarios mediante el siguiente comando:

```
sudo systemd-sysusers /usr/lib/sysusers.d/ffmpeg.conf
```

Con los usuarios ya creados se diseñará un servicio al cual llamar cada vez que queramos comenzar la emisión del vídeo. Este servicio es un servicio `systemd`, y se debe crear en la ruta oportuna: `/etc/systemd/system`. Dentro, se debe guardar un fichero como este:

video0.service

```
[Unit]
Description = Servicio de video utilizando la técnica DASH


[Service]
User = ffmpeg
Group = video
ExecStart = /usr/local/bin/ffmpeg -y -nostdin -f v4l2 -video_size 1280x720 -framerate 25 -i /dev/video0 -vcodec h264_omx -keyint_min 0 -g 100 -map 0:v -b:v 1000k -f dash -seg_duration 4 -use_template 1 -use_timeline 0 -init_seg_name init-video0-'$'RepresentationID'$'.mp4 -media_seg_name video0-'$'RepresentationID'$'-'$'Time'$'.mp4 -remove_at_exit 1 -window_size 20 /var/www/html/video0.mpd


Restart = always
RestartSec = 30s


[Install]
WantedBy = multi-user.target
```

Dentro del fichero existen diferentes apartados, el primero, `Description`, define el servicio.

Los de dentro de `Service` definen quién ejecutará qué:

`User` y `Group` definen quién podrá ejecutar el comando `ExecStart`, el usuario y grupos definidos anteriormente.

En el comando ExecStart está realmente lo que hace que se generen los fragmentos de vídeo, para comprender qué se está ejecutando vamos a analizar uno a uno los parámetros que se le pasan a FFmpeg:

`/usr/local/bin/ffmpeg`	Ruta donde se aloja FFmpeg
`-y`	Sobreescribe los ficheros generados sin preguntar
`-nostdin`	Fuerza a FFmpeg a actuar de manera no interactiva
`-f v4l2`	Formato de entrada (video4linux V2)
`-video_size 1280x720`	Dimensiones del vídeo capturado
`-framerate 25`	Tasa de frames a capturar
`-i /dev/video0`	Nombre del dispositivo / cámara
`-vcodec h264_omx`	Codificación escogida
`-keyint_min 0`	Permitir que cualquier frame sea un key frame
`-g 100`	Al menos cada 100 frames, uno será key frame
`-map 0:v`	Mapea la entrada del vídeo a 0
`-b:v 1000k`	Establece el bitrate a 1000k
`-f dash`	Formato de salida: DASH
`-seg_duration 4`	Duración de los segmentos: 4s
`-use_template 1`	Habilita el uso de SegmentTemplate
`-use_timeline 0`	Inhabilita el uso de SegmentTimeline
`-init_seg_name init-video0-'$'RepresentationID'$'.mp4`	Nombre del segmento utilizado para la inicialización del servicio
`-media_seg_name video0-'$'RepresentationID'$'-'$'Time'$'.mp4`	Nombre de los diferentes segmentos, en este caso los nombres varían con el tiempo
`-remove_at_exit 1`	Indica que se borren los ficheros al terminar el streaming
`-window_size 20`	Indica la cantidad máxima de ficheros a almacenar en el destino (si se supera este número, el fragmento primero se borrará y lo sustituirá el N+1)
`/var/www/html/video0.mpd`	Ruta donde se guardará el manifest

Tabla 1: Explicación de parámetros de FFmpeg

Para generar vídeo en distintos formatos simplemente deberíamos añadir de nuevo una línea a ejecutar prácticamente igual que la anterior. Por ejemplo, podría escalar la resolución de 1280x720 a 720x480 con un bitrate de 500k.

Se cambiaría la línea:

`-b:v 1000k -> -b:v 500k`

Se añadiría la línea:

```
-vf “scale=720:480”
```

Una vez se ha guardado el servicio, se podrá ejecutar del siguiente modo:

```
sudo systemctl enable video0
sudo systemctl start video0
```

Una vez comprendidos los parámetros de FFmpeg se va a demostrar por qué se ha escogido la codificación hardware. Esta, en comparación a la codificación H.264 normal, es mucho más eficiente. Se puede apreciar en el porcentaje de CPU utilizada y la velocidad de codificación de las siguientes figuras, en las cuales se ha ejecutado la misma orden a FFmpeg únicamente cambiando el parámetro del códec:

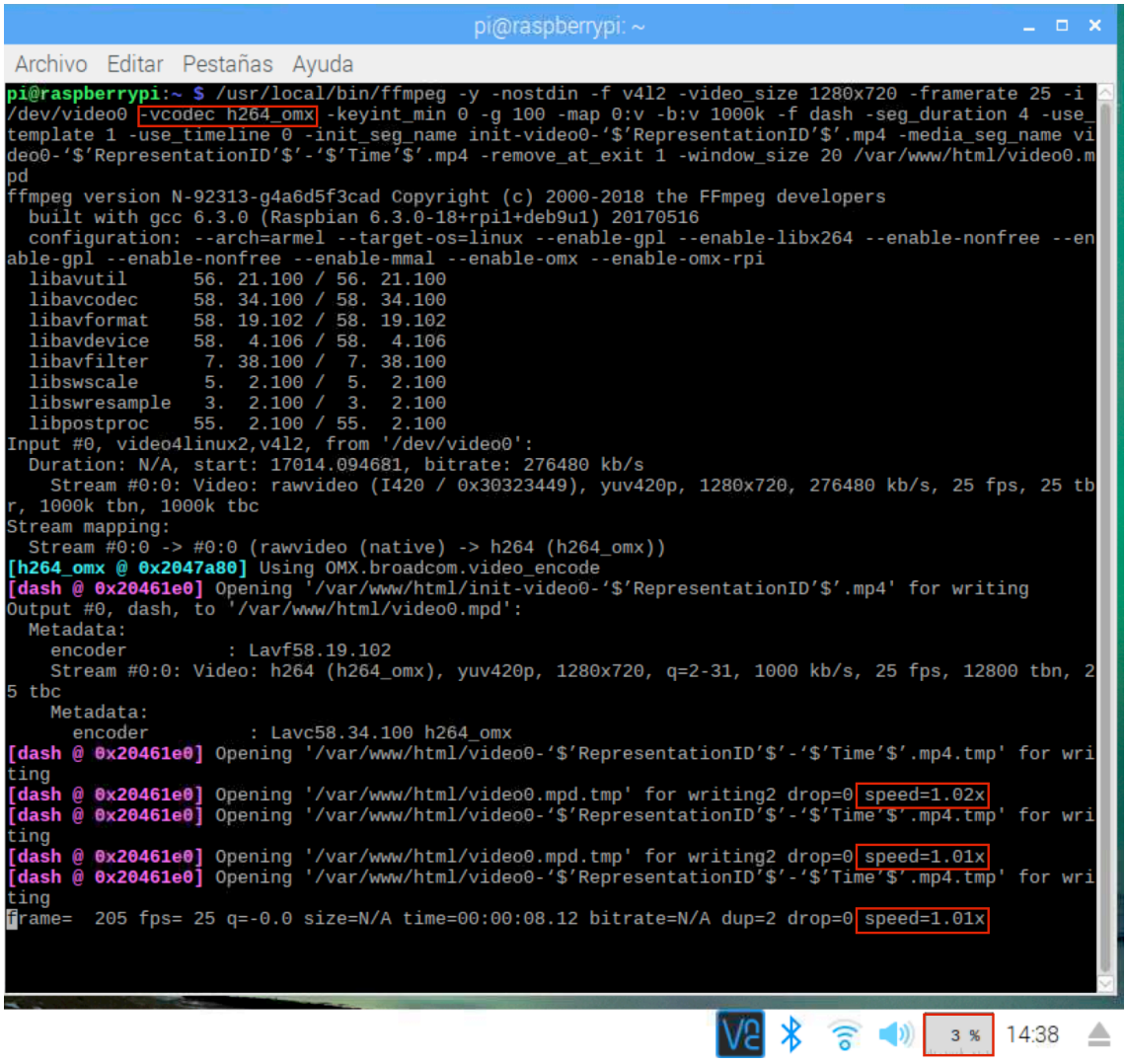

Fig. 19: Codificación utilizando H.264 y OpenMAX

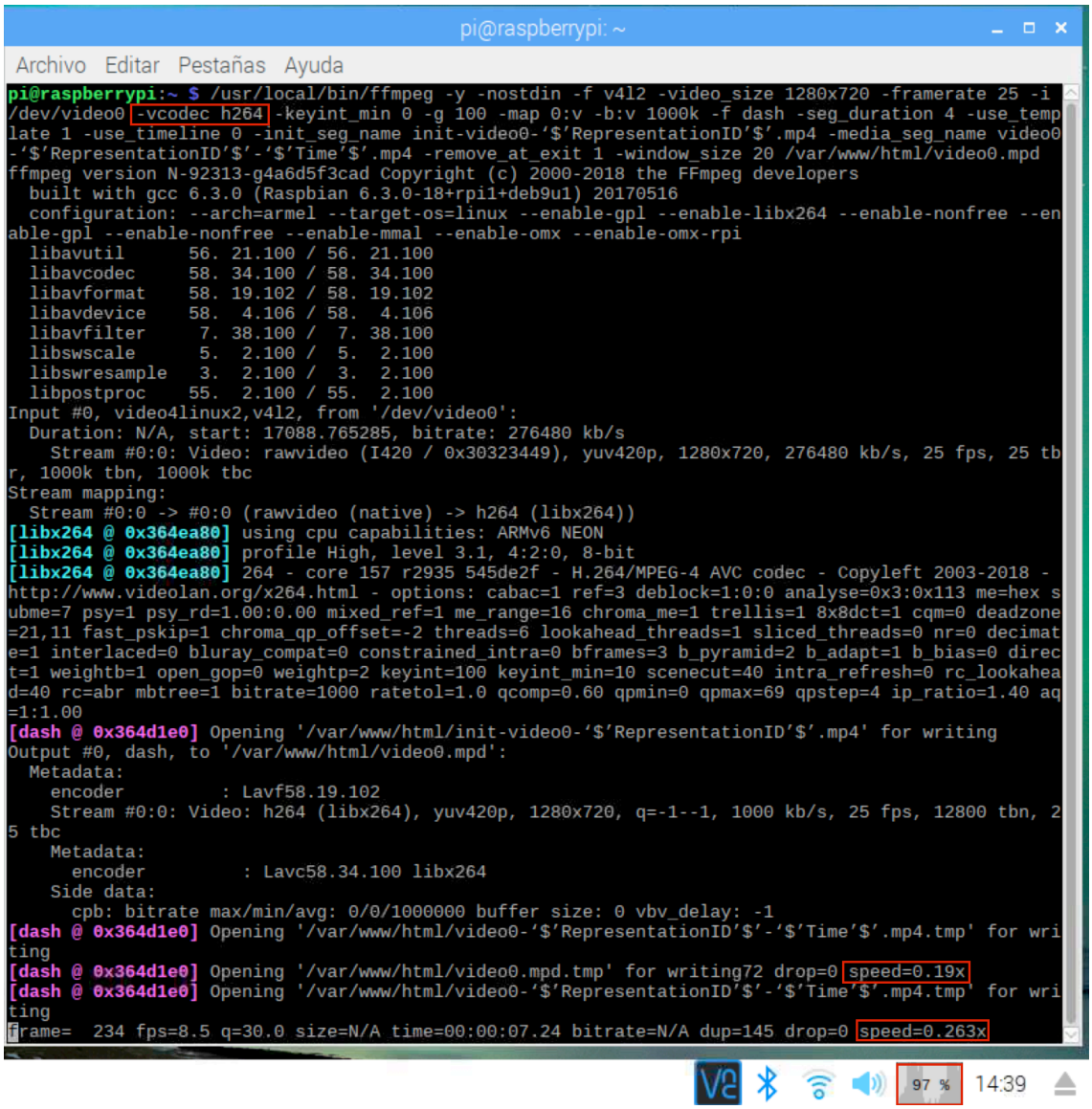

Fig. 20: Codificación utilizando H.264 sin OpenMAX

Como puede apreciarse en ambas figuras 19 y 20, una llama al códec H.264 mediante `–vcodec h264` y la otra llama al códec H.264 que utiliza OpenMAX mediante `–vcodec h264_omx`.

Si observamos el porcentaje de uso de la CPU se aprecia que en el primer caso es de un 6% mientras que en el segundo es de un 97%.

Además, ni siquiera utilizando el 97% de la CPU es capaz de codificar el vídeo a velocidad 1x, es decir, le cuesta más de 1 segundo codificar 1 segundo de vídeo, acumula retraso a medida que se van generando fragmentos, es por esto que es imposible ofrecer un servicio de vídeo en directo.

Una vez generados los fragmentos de vídeo es momento de poder visualizarlo, para ello se deberá confeccionar en Node-RED el flujo de datos pertinente que dirija la información al vídeo a un *template* donde poder visualizarlo y pasarlo a un Dashboard. La idea es poder visualizar, además, otros vídeos en formato DASH, que aunque no sean en directo, sí utilicen esta técnica.

El primer paso será diseñar el Dashboard que se mostrará al usuario.

En primer lugar se debe crear un botón, el cual al pulsarlo guarde en una variable global el valor de la URL del *manifest* o la ruta del vídeo en directo. A partir de este dato, el template deberá ser capaz de reproducir el vídeo.

En el Dashboard que se quiere diseñar habrá una caja de texto donde poder introducir la URL del *manifest* de un vídeo alojado en un servidor web de internet, por lo tanto, se deberá conectar también a la variable que guardaremos.

El template es un nodo de la categoría de "Dashboard" de Node-RED, en el cual podemos introducir contenido HTML y Angular/Angular-Material que puede utilizarse para crear un elemento de interfaz de usuario.

En este caso va a servir para programar el marco en el cual se va a reproducir el vídeo. Para esto tendremos que tener en cuenta que será necesario llamar al reproductor, en este caso Shaka Player, además de tener contemplados los posibles errores [10].

```
<!DOCTYPE html>
<html>
    <head>
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/shaka-player/2.4.5/shaka-
player.compiled.js"></script> //se utilizan las librerías de Shaka compiladas
    </head>
    <body>
    <div class="video">
      <video id="video"
           width="720"
           controls autoplay>
      </video>
    </div>

    <script>
       function initApp() {         //definimos la función initApp
       shaka.polyfill.installAll(); //Instalamos componentes para la
                                    //compatibilidad con el buscador
       if (shaka.Player.isBrowserSupported()) { //Se comprueba que sea compatible
                 initPlayer();                  //y se inicia el reproductor
               } else {                         //Si no lo es, saltará un error
                 console.error('Browser not supported!');
               }
             }
       function initPlayer() {       //definimos la función initPlayer
       var video = document.getElementById( 'video' );
       var player = new shaka.Player( video );
       window.player = player;       //unimos el reproductor a la ventana de
                                     //visualización
       player.addEventListener('error', onErrorEvent); //Contemplamos posibles
                                                       //errores
       function onErrorEvent(event) {
         onError(event.detail); //Extraemos el objeto shaka.util.Error
             }
       function onError(error) {
         console.error('Codigo de error: ', error.code, ' en ', error);
             }
    </script>
```

```
<script>
  (function(scope) { //Con scope creamos una función que actúa cuando
                     //cambia la variable URL para volver a cargar el nuevo vídeo
     var timer = setInterval(function() {
     if (!window.shaka) return; //Comprobamos el funcionamiento de la ventana
     clearInterval(timer);      //se restablece el valor de timer
     initApp();                 //se llama a initApp
     scope.$watch('msg', function(msg) {
          if (msg) {
               var url = msg.payload;
               window.player.load(url).then(function() {
               console.log('El vídeo se ha cargado');
               }).catch(onError);
          }
          });
     }, 100);
   })(scope);
    </script>
    </body>
</html>
```

Con el código anterior se completa el diseño del Dashboard. Finalmente en Node-RED queda un diseño así:

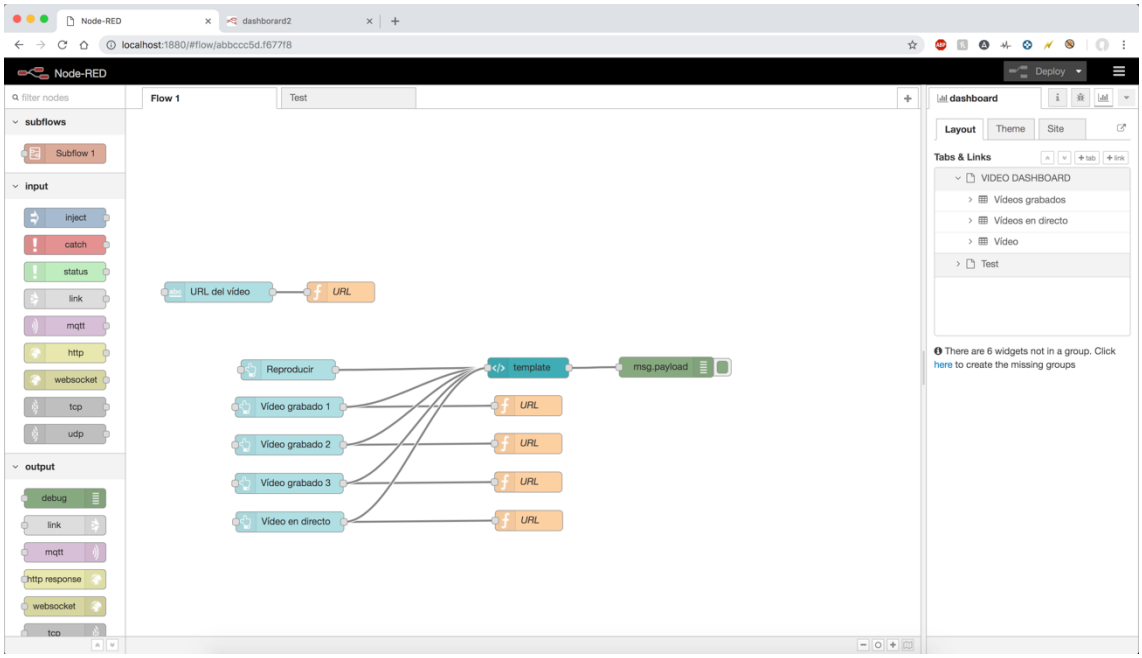

Fig. 21: Diagrama de nodos en Node-RED

Este conjunto de nodos finalmente consigue que se visualice este Dashboard:

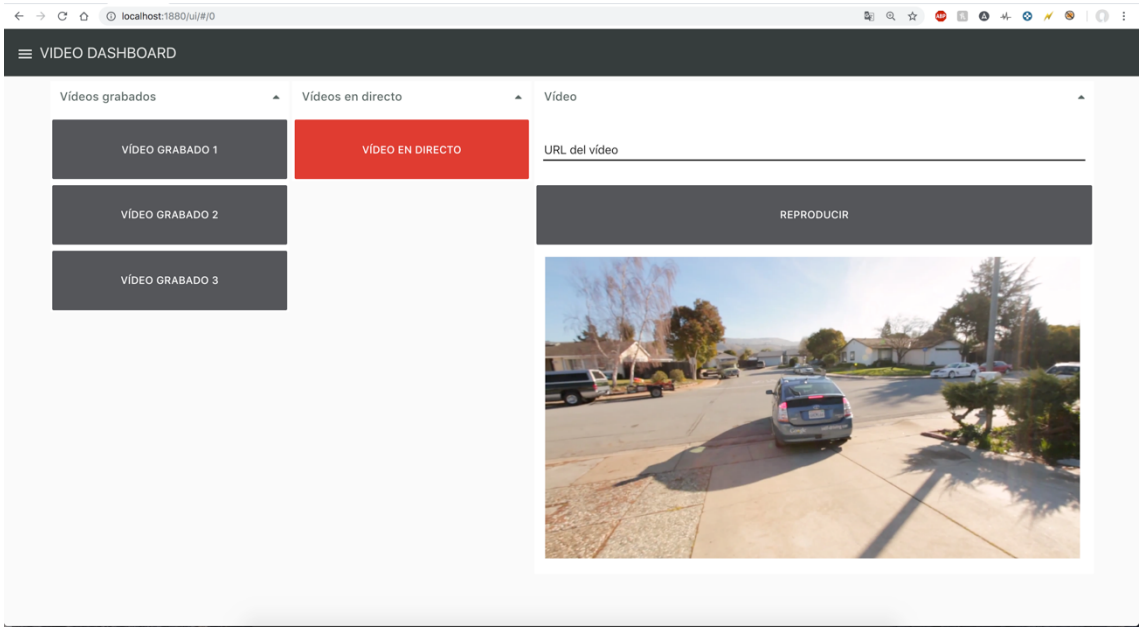

Fig. 22: Dashboard

Tal y como se ha explicado las distintas maneras de reproducir un vídeo son:

- Mediante botones que envían una URL (predefinida en las características del nodo) del lugar donde está alojado el vídeo.
- Mediante botones que envían la ruta donde se aloja el vídeo que está emitiendo en directo desde el propio dispositivo.
- Mediante la introducción de una URL cualquiera en la cual se redirija al *manifest* de un vídeo alojado en un servidor web de internet.

Capítulo 5. Conclusiones y propuesta de trabajo futuro

A raíz de la realización del presente proyecto podemos concluir que es un proyecto que abarca tecnología presente hoy en día en la transmisión de vídeo. Con la creciente demanda del conjunto de usuarios de internet por el contenido en directo se crea la necesidad de disponer de las técnicas más avanzadas y que proporcionen al cliente un mayor grado de satisfacción, consiguiéndose con la adaptación de los reproductores a la red del cliente con el fin de no provocar una insatisfacción al consumir contenido multimedia.

Se abarca desde el primer momento en el que la cámara comienza a captar vídeo hasta el momento en el que el usuario final tendría que consumir dicho contenido, por lo que se adquiere una visión global muy completa sobre el diseño de entornos de emisión de contenido multimedia en tiempo real.

Fig. 21: Diagrama de flujos del proyecto

Se han aplicado conocimientos adquiridos durante el Grado provenientes de asignaturas como Sistemas Multimedia, en la cual se incluyen contenidos sobre la técnica DASH y otros sistemas de transmisión de vídeo, Aplicaciones Telemáticas, donde el contenido de programación en Java (similar a JavaScript) fue necesario para la programación en Android, Servicios Telemáticos para la Gestión de la Información, a nivel de programación web y Diseño de Servicios Telemáticos donde se abarcó el diseño de servidores web.

Como propuesta de trabajo futuro este proyecto podría escalarse conectando más de una cámara a la Raspberry Pi, o incluso conectando varias cámaras en diferentes Raspberry Pi, todas ellas aportando los fragmentos a un servidor web en común desde el cual obtener los ficheros del *manifest* para poder reproducirlos.

Capítulo 6. Bibliografía

[1] Köhn, Rüdiger. "Online-Kriminalität: Konzerne verbünden sich gegen Hacker"

http://www.faz.net/aktuell/wirtschaft/diginomics/grosse-internationale-allianz-gegen-cyber-attacken-15451953-p2.html?printPagedArticle=true#pageIndex_1. [Online]

[2] Gartner, "Leading the IoT, Garner Insights on How to Lead in a Connected World" https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf. [Online]

[3] Node-RED, "Home Web Page" https://nodered.org/ [Online]

[4] Wikipedia, "Dynamic Adaptive Streaming over HTTP" https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP [Online]

[5] Shaka Player GitHub, "DASH features" https://github.com/google/shaka-player [Online]

[6] encoding.com , "MPEG-DASH, An overview" https://www.encoding.com/mpeg-dash/ [Online]

[7] Coursera, "A developer's guide to the Internet of Things" https://www.coursera.org/learn/developer-iot [Online]

[8] Raspberry Pi Web, "Download Raspbian" https://www.raspberrypi.org/downloads/raspbian/ [Online]

[9] FFmpeg web, "About FFmpeg" https://www.ffmpeg.org/about.html [Online]

[10] Jungle Disk, "Live Streaming MPEG Dash with Raspberry Pi 3" https://www.jungledisk.com/blog/2017/07/03/live-streaming-mpeg-dash-with-raspberry-pi-3/ [Online]