Trabajo de Fin de Grado

Aplicación Segura de Videovigilancia para Android

Secure Video Surveillance Application for Android

Jonay Suárez Armas

La Laguna, 1 de junio de 2016

Dña. **Pino Teresa Caballero Gil**, con N.I.F. 45.534.310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Cándido Caballero Gil** con N.I.F. 42.201.070-A Profesor Contratado Laboral Interno adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"Aplicación Segura de Videovigilancia para Android."

ha sido realizada bajo su dirección por D. **Jonay Suárez Armas**, con N.I.F. 78.639.262-P.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de junio de 2016.

Agradecimientos

En primer lugar, a la tutora Pino Caballero Gil y el cotutor Cándido Caballero Gil por su excelente trato, por su ayuda y por todas las oportunidades brindadas.

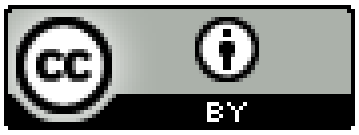
En segundo lugar al grupo CryptULL de la Universidad de La Laguna por la ayuda prestada en todo momento.

En tercer lugar a mi familia. En especial a mis padres por darme la oportunidad de estudiar una carrera y por animarme a seguir en todo momento.

También a mi pareja, por casi obligarme a matricularme en la carrera y por todo el apoyo en los buenos y malos momentos de esta etapa.

Y por supuesto a mis amigos; los de antes y los que he conocido en la carrera y con los que además de formarme he compartido buenos momentos.

Licencia



Resumen

En este Trabajo de Fin de Grado se ha desarrollado una aplicación segura de videovigilancia para Android, a la cual se le ha puesto el nombre "RoboCAM".

La propuesta se basa en un smartphone Android para captar y enviar imágenes, colocado encima de un robot Lego MindStorms EV3 con el cual está conectado mediante Bluetooth. En el momento en que se detecte movimiento delante de la cámara, la aplicación empieza a mandar las imágenes a un servidor de streaming en el que se pueden ver a través de una aplicación web cliente. Además, a través de la aplicación web, se puede mover el robot para intentar obtener más información de la zona vigilada.

Además de enviar las imágenes al servidor de streaming, en el momento en el que se detecta movimiento delante de la cámara, la aplicación móvil se encarga de enviar notificaciones a través de mensajes de texto, indicando la fecha y hora de la detección, a los destinatarios que se hayan escogido con anterioridad.

Como aplicación segura, provee los mecanismos de seguridad necesarios; tanto en la aplicación principal desarrollada en Android como en la aplicación web cliente. El cifrado de la información se realiza mediante el AES de 256 bits en modo CBC, para el acuerdo de clave secreta se utiliza el algoritmo Diffie Hellman de Curva Elíptica, y para la verificación de la autenticidad del emisor de los mensajes y la integridad de los datos se usa el algoritmo de Firma Digital de Curva Elíptica.

Palabras clave: Android, Videovigilancia, Streaming, Detección de movimiento, Control de robot, Seguridad.

Abstract

In this End-Term Project, a secure application of video surveillance for Android, which has been given the name of "RoboCAM", has been developed.

The proposal is based on an Android smartphone to capture and send images, placed on top of a Lego Mindstorms EV3 robot with which it is connected via Bluetooth. As soon as motion is detected in front of the camera, the application begins to send images to the streaming server so that they can be seen through the client web application. In addition, through the web application, the Robot can be moved to get more information on the surveillance area.

In addition to sending images to the streaming server at the time that movement is detected in front of the camera, the mobile application is able to send notifications through text messages, indicating the date and time of detection, to the receivers have been chosen previously.

As a secure application, it provides all the necessary security mechanisms; both in the main application developed in Android and in the client web application. The encryption of data is performed by 256-bit AES in CBC mode, for the secret key agreement algorithm Diffie Hellman Elliptic Curve is used, and for verifying the authenticity of the sender of the messages and the integrity of the data the algorithm Elliptic Curve Digital Signature is used.

Keywords: Android, Video Surveillance, Streaming, Motion Detection, Robot Control, Security.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción al trabajo

1.1 Motivación

El robo en viviendas y establecimientos es algo que está a la orden del día. España es el tercer país de la Unión Europea que sufre más robos en las tiendas, y en casas se registran más de 310 robos al día. Por ello son cada vez más los establecimientos y viviendas que instalan alarmas. Dichas alarmas son capaces de contactar con la policía en caso de que se activen.

Además de alarmas, los establecimientos comerciales suelen montar equipos de videovigilancia. Estos permiten ver las imágenes en tiempo real desde un monitor, y además graban las imágenes en servidores para poder verlas en cualquier momento si fuera necesario. Además, con el avance tecnológico que han sufrido las redes, es posible ver las imágenes a través de Internet estando en cualquier otro lugar siempre que se disponga de una conexión a Internet.

Debido a todo esto, surge en este proyecto la idea de un sistema de videovigilancia que mejore a los actuales en prestaciones y en coste. Por ello se decide el uso de smartphones Android como dispositivos captadores de imágenes, aprovechando sus propias cámaras. Además, de esta manera se tendrían dispositivos inteligentes capaces de realizar otras tareas aparte de la captura de imágenes. También, si se dispone de teléfonos móviles Android que ya no se estén usando, se podrían aprovechar y de ese modo se ahorrarían costes.

Para diferenciar este sistema de videovigilancia del resto de los que hay en el mercado, y así darle más valor, se piensa en cámaras moviéndose por las diferentes estancias, de modo que se pueda intentar obtener información de todo el entorno. Para poder mover las cámaras, que en este caso serían teléfonos móviles inteligentes, es posible montarlos sobre un robot de modo que el teléfono se conectaría con el robot sobre el que esté colocado, y desde donde se estén viendo las imágenes se podría mover el robot en tiempo real.

1.2 Estado del arte

Hoy en día, en cuanto a seguridad en el hogar o en el negocio, podemos encontrar bastante variedad en sistemas de seguridad. Podemos encontrar, desde simples alarmas hasta sistemas de videovigilancia con cámaras IP.

La opción más sencilla, y probablemente la más utilizada, es instalar alarmas. En el mercado se pueden encontrar desde alarmas muy sencillas que se basan en detectores de movimiento y lo que hacen es activarse en cuanto uno de los sensores detecta movimiento, o también se pueden encontrar sistemas de alarma más complejos y con más funcionalidades. Existen sistemas de alarma con diferentes tipos de sensores, como pueden ser los ya nombrados sensores de movimiento, detectores de humo o detectores magnéticos para las puertas. Además de sonar al activarse, estas alarmas son capaces de enviar notificaciones a través de llamadas, mensajes de texto (SMS), correos electrónicos o incluso en una aplicación nativa para Android o iOS.

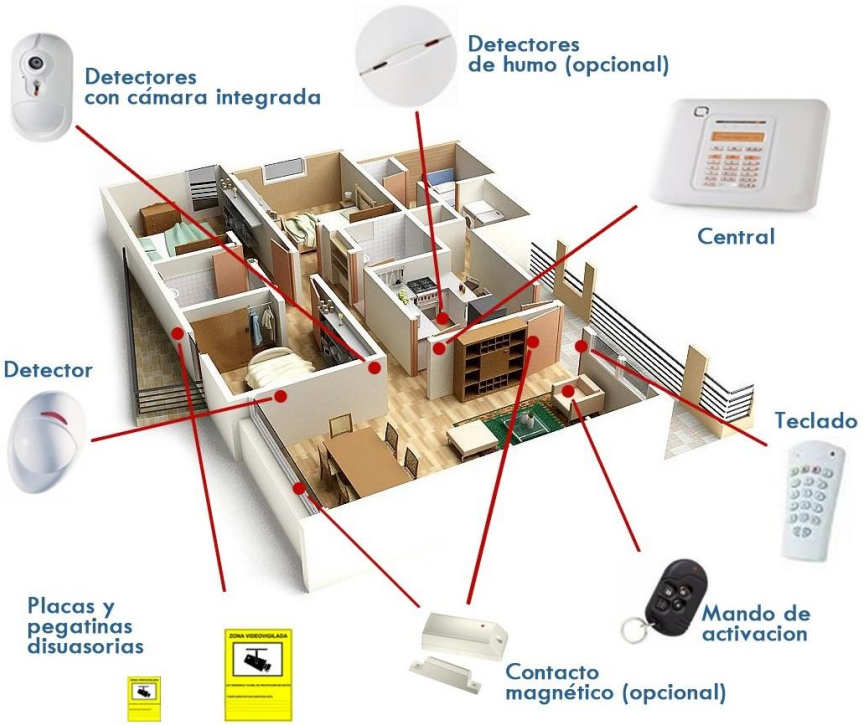


Figura 1.2.1. Sistema de alarma moderno

Otra de las opciones, que se suele escoger sobre todo en locales comerciales, es montar un sistema de cámaras en el cual, a través de un monitor, se pueden visualizar las imágenes que están captando las diferentes cámaras, en tiempo real. Estos sistemas se suelen usar en las tiendas para que la persona encargada pueda vigilar que ningún cliente pueda salir con un producto que no ha pagado.

Algunos de estos sistemas también son capaces de guardar las imágenes en algún soporte físico para visualizarlas en otro momento si fuera necesario.

Figura 1.2.2. Pantalla de sistema de videovigilancia

La última tendencia en sistemas de videovigilancia es usar cámaras IP, que se conectan por medio de la red a un servidor de vídeo para enviarle las imágenes. Se conectan a la red por medio de una conexión Ethernet, y muchas tienen la ventaja de contar con la tecnología PoE (Power over Ethernet), la cual permite obtener la alimentación eléctrica a través de la propia red sin necesidad de ser conectadas a la red eléctrica. Hay que destacar que para utilizar PoE, la electrónica de red debe estar preparada para ello.

Figura 1.2.3. Cámara IP PoE y Switch PoE

También hay que decir, que gracias a los avances de los últimos años en las redes de comunicación, los sistemas de videovigilancia modernos permiten ver las imágenes en tiempo real a través de Internet desde cualquier otra ubicación.

Por otra parte, también existen aplicaciones para dispositivos móviles capaces de realizar tareas de videovigilancia. A continuación vemos algunas aplicaciones interesantes para smartphones Android.

- Motion Detector Pro. Se trata de una aplicación de detección de movimiento. Cuando detecta movimiento, envía una notificación por correo electrónico o por mensaje de texto. Además, envía las imágenes del momento de la detección de movimiento. También permite ajustar el rango de sensibilidad, de modo que dependiendo del entorno se pueda ajustar para obviar algunos pequeños movimiento que sean algo normal.

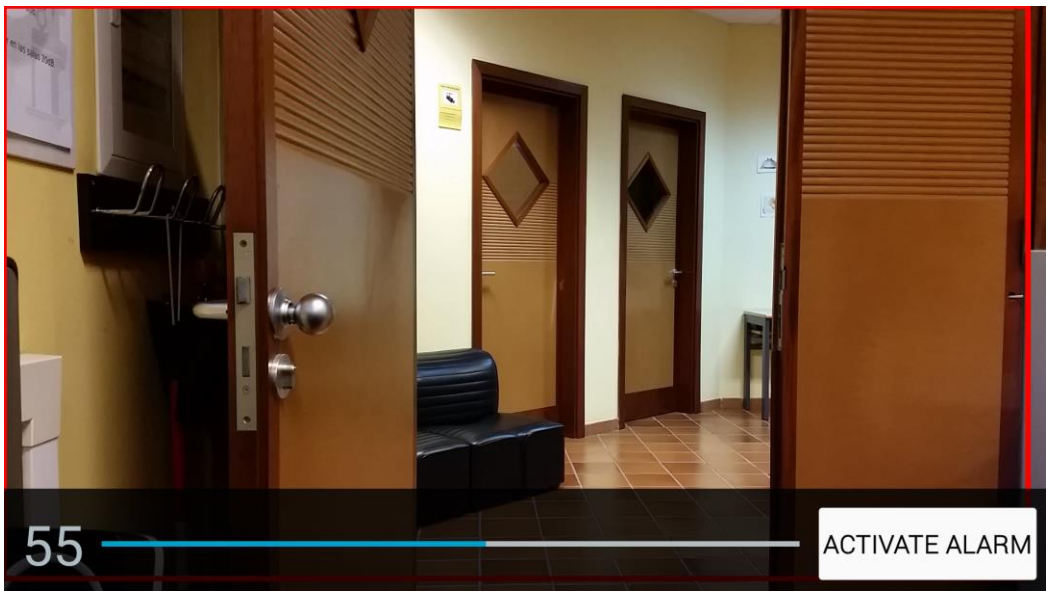


Figura 1.2.4. Aplicación Motion Detector Pro

- Cámara de vigilancia wifi. Esta es una aplicación muy interesante ya que ofrece bastantes posibilidades de uso. Se trata de una aplicación que se puede usar en dos modos: modo cámara y modo reproductor. El modo cámara, como su nombre indica es el que permite el envío de vídeo, y además puede detectar movimiento. El modo reproductor permite visualizar las imágenes que están enviando otros dispositivos a través de esta aplicación. Para comunicar el dispositivo reproductor (o en su defecto, la aplicación web disponible para el navegador Mozilla Firefox) con los dispositivos en modo cámara es necesario una cuenta de Google, que es la que hace la asociación entre los dispositivos.

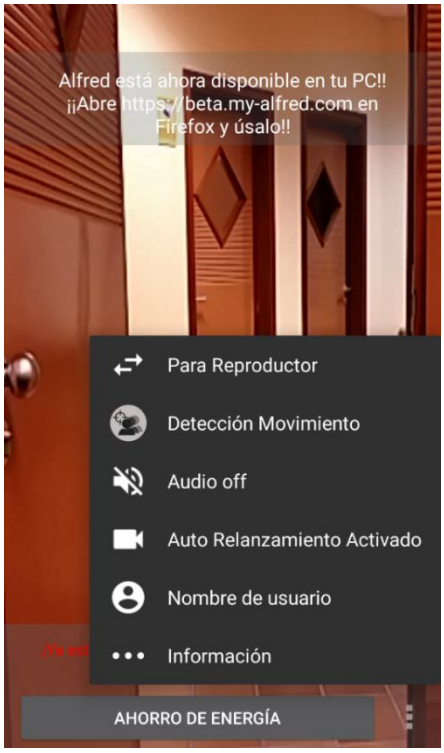


Figura 1.2.5. Aplicación cámara de vigilancia wifi

Además de aplicaciones para Android de videovigilancia, en este proyecto también interesan las aplicaciones para el control de robots, y más concretamente para el control del robot Lego MindStorms EV3. La compañía Lego dispone de una aplicación propia que es capaz de controlar el robot a través de Bluetooth, y además existen otras más sencillas de utilizar como es EV3 Simple Remote.

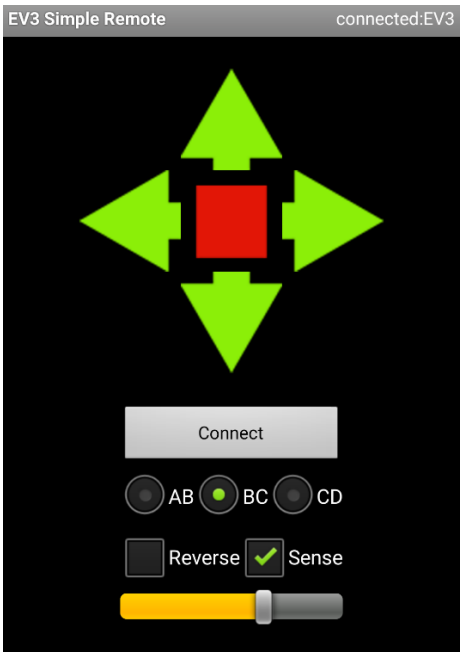


Figura 1.2.6. Aplicación EV3 Simple Remote

1.3 Objetivos

El objetivo principal del Trabajo de Fin de Grado es implementar una aplicación de videovigilancia para smartphones Android, que además de captar las imágenes para verlas desde otro lugar, es capaz de mover las cámaras para obtener información de todas las zonas que se están vigilando. Para ello, el teléfono que esté ejecutando la aplicación tendrá que ir colocado sobre un robot. En este caso, se ha elegido un robot Lego MindStorms EV3.

Otro de los objetivos de este Trabajo de Fin de Grado está basado en la detección de movimiento ya que la aplicación tiene que estar constantemente captando imágenes a través de la cámara del dispositivo, para en caso de detectar movimiento empezar a enviar el vídeo mediante streaming.

1.4 Fases del desarrollo

Las fases de desarrollo del proyecto han sido:

- Primera fase. Primeros pasos de la aplicación: En esta primera fase de desarrollo se ha trabajado en la interfaz gráfica de la aplicación, para luego, en las siguientes fases a implementar las funcionalidades de la misma.
- Segunda fase. Streaming: En esta fase se ha trabajado en la implementación del sistema de streaming de la aplicación y en la aplicación web cliente encargada de recibir y mostrar las imágenes desde el servidor de streaming.
- Tercera fase. Detección de movimiento: En esta fase se ha hecho la implementación de la detección de movimiento de la aplicación.
- Cuarta fase. Control de robot: En esta fase se ha trabajado en la conexión con el robot, así como en la comunicación entre la aplicación web cliente y la aplicación principal Android para el envío de comandos al robot.
- Quinta fase. Seguridad: En esta última fase de desarrollo se ha trabajado en la inclusión de los diferentes métodos de seguridad de la aplicación.

1.5 Estructura de la memoria

A continuación se describe brevemente de qué trata cada capítulo de esta memoria.

- Capítulo 1. Introducción: En este primer capítulo introducimos los antecedentes del Trabajo de Fin de Grado así como los motivos que llevan a su desarrollo y los objetivos a cumplir.
- Capítulo 2. Introducción a la herramienta: Incluye la explicación del funcionamiento de la aplicación.
- Capítulo 3. Tecnologías Inalámbricas: proporciona detalles de las tecnologías inalámbricas que intervienen en este Trabajo de Fin de Grado.
- Capítulo 4. Sistema de Detección de Movimiento: En este capítulo se define el sistema de detección de movimiento que utiliza la aplicación.
- Capítulo 5. Sistema de Streaming: En este capítulo se comenta cómo funciona el sistema de streaming que utiliza la aplicación.
- Capítulo 6. Sistema de Control de robot: En este capítulo se trata la conexión con el robot y el funcionamiento de las comunicaciones entre la aplicación web cliente y la aplicación principal, y entre la aplicación principal y el robot.
- Capítulo 7. Combinación de Sistemas: Describe los esquemas del sistema completo (todos los sistemas ensamblados) y funcionamiento general.
- Capítulo 8. Implementación: Muestra los detalles más importantes de la implementación así como los problemas que surgieron y sus correspondientes soluciones.
- Capítulo 9. Presupuesto: En este capítulo se relacionan los costes que supone llevar a cabo este proyecto.
- Capítulo 10. Conclusiones y trabajos futuros: En este capítulo se presentan las conclusiones finales y los posibles trabajos futuros que pueden surgir.
- Capítulo 11. Conclusions and future works: Este capítulo es la traducción al inglés del capítulo anterior.
- Apéndice A. Código destacable: En este apéndice se muestran los fragmentos del código más importante de la aplicación.
- Apéndice B. Conference paper: En este apéndice se presenta un short paper enviado a un congreso internacional, relacionado con este Trabajo de Fin de Grado.

Capítulo 2. Introducción a la herramienta

2.1 Definición

La aplicación segura de videovigilancia para Android RoboCAM diseñada e implementada en este trabajo se compone de dos aplicaciones:

- Aplicación principal Android: Es la aplicación principal, la que va instalada en el dispositivo móvil. Realiza las funciones de envío de vídeo mediante streaming, detección de movimiento y control del robot Lego MindStorms EV3.
- Aplicación web cliente: Aplicación web alojada en el servidor. En ella se pueden ver las imágenes captadas por el dispositivo móvil en tiempo real y se pueden enviar comandos al robot para moverlo.

2.2 Aplicación principal Android

La aplicación principal, desarrollada para dispositivos Android, se compone de tres funcionalidades:

- Envío de audio y vídeo mediante streaming
- Detección de movimiento
- Control de robot

En la pantalla principal de la aplicación tenemos el menú principal, el cual se compone de cuatro botones:

- Ver Vídeo
- Configurar Servidor

- Configurar Robot
- Configurar Notificaciones

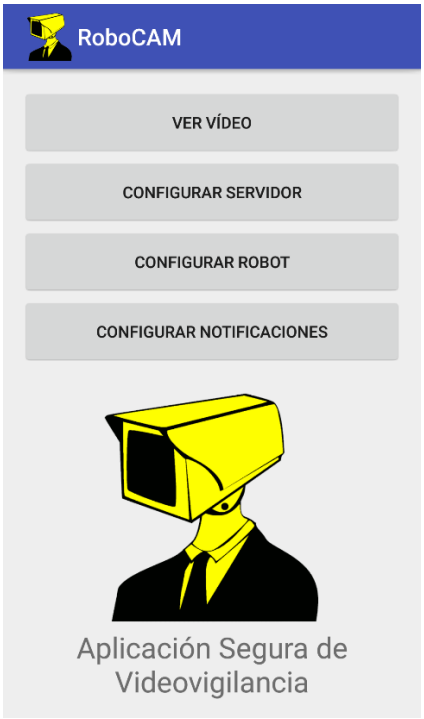


Figura 2.2.1. Menú principal de la aplicación Android

Con el botón "Ver Vídeo" accedemos a la pantalla de la cámara. Nada más entrar, aparece un diálogo que explica que para poder utilizar el robot es necesario activar el Bluetooth, y da la opción de activarlo o no. Si se activa en ese momento, al cerrar la pantalla de la cámara, automáticamente se desactiva para que el usuario no se olvide de desactivarlo luego.

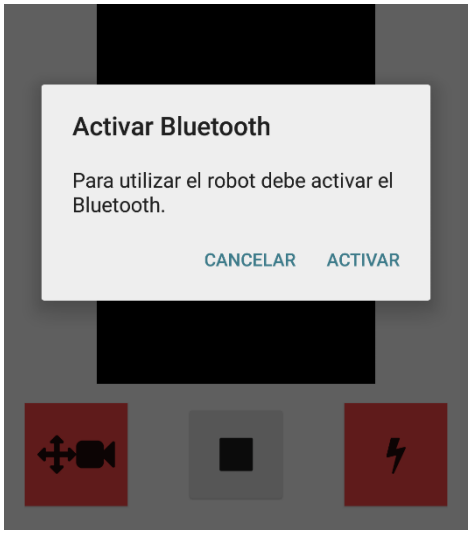


Figura 2.2.2. Dialogo de activar robot de la aplicación Android

En esta última pantalla es donde la cámara al pulsar el botón correspondiente empieza a hacer la detección de movimiento (el botón se pone en color amarillo), y cuando se detecta algo diferente delante de la cámara, la aplicación empieza a enviar el vídeo mediante streaming al servidor (botón en color verde) y además, envía un mensaje de texto a los destinatarios que hayamos elegido.

Mientras estamos en esta pantalla, se activa la conexión con el robot Lego MindStorms EV3 para poder moverlo de forma remota.

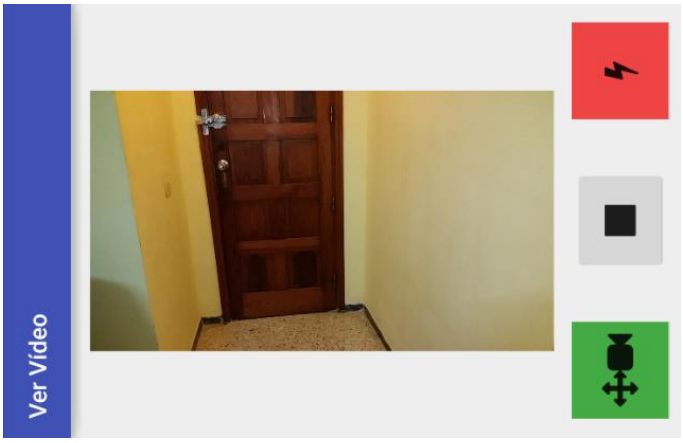


Figura 2.2.3. Pantalla de la cámara de la aplicación Android

Con el botón "Configurar Servidor" podemos configurar las direcciones del servidor web y del servidor de streaming para que la aplicación sepa a donde enviar las imágenes.

Figura 2.2.4. Pantalla de configuración de servidor de la aplicación móvil

Con el botón “Configurar Robot” podemos elegir el dispositivo robot al que se conectará la aplicación. Se trata de una lista de dispositivos Bluetooth encontrados por la aplicación.

Figura 2.2.5. Pantalla de elección del robot de la aplicación Android

Y con el último botón, “Configurar Notificaciones”, podemos añadir los destinatarios a los que queremos que se envíe un mensaje de texto cuando la aplicación detecta movimiento delante de la cámara. Se pueden añadir los destinatarios directamente desde los contactos que están guardados en el teléfono o introduciendo el número de teléfono. Además, se pueden deshabilitar los destinatarios para no tener que eliminarlos o cambiarles el nombre.

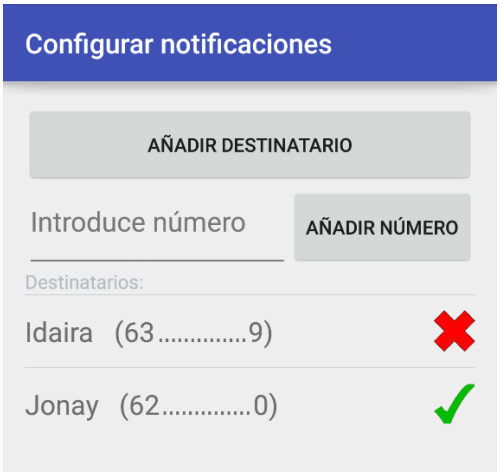


Figura 2.2.6. Pantalla de notificaciones de la aplicación Android

Para cambiar el nombre de un destinatario, eliminarlo de la lista de notificaciones, y habilitarlo o deshabilitarlo hay que dejar pulsado el contacto hasta que aparezca el menú secundario y pulsar en la opción deseada.

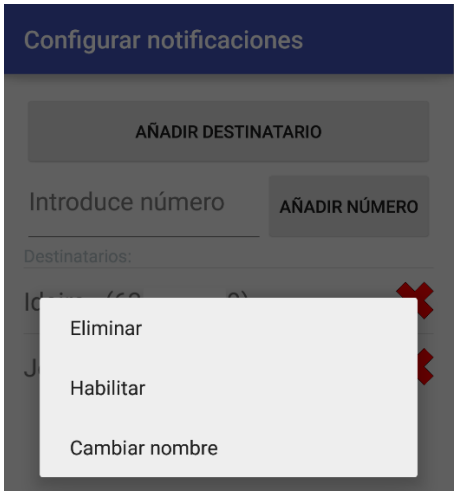


Figura 2.2.7. Menú secundario de los destinatarios de la aplicación móvil

Al pulsar cada una de estas opciones, aparece un diálogo para confirmar o cancelar la operación elegida.

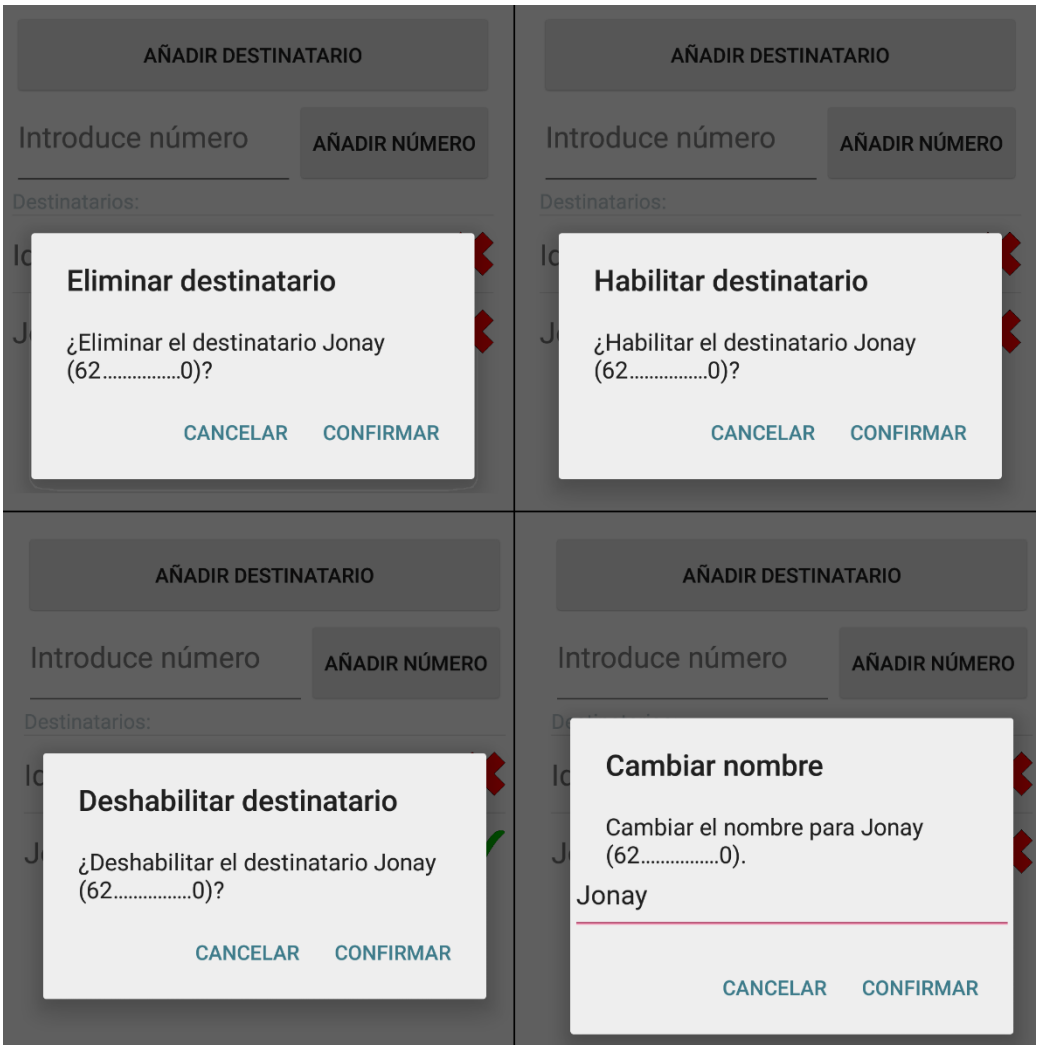


Figura 2.2.8. Diálogos de las notificaciones de la aplicación Android

2.3 Aplicación web cliente

La aplicación cliente, como principal ventaja tiene que, al ser una aplicación web, no es necesario instalarla en el dispositivo en el que se va a utilizar. Simplemente sería necesario disponer de un navegador web para poder utilizarla.

Está desarrollada en HTML, CSS, JavaScript y NodeJS; y hace uso del framework web AngularJS para las vistas. Además, utiliza una base de datos MongoDB para guardar la información de las cámaras que se conectan a la aplicación.

Figura 2.3.1. Página principal de la aplicación web cliente

Se trata de una aplicación sencilla, que tiene solamente dos funcionalidades:

- Visualizar las imágenes desde el servidor de streaming
- Enviar comandos al robot para moverlo

Debajo del vídeo recibido, tenemos unos botones para enviar los comandos de movimiento al robot. Los movimientos que puede realizar el robot son: adelante, atrás, izquierda, derecha, rotación hacia la izquierda y rotación hacia la derecha. Además se puede ajustar la velocidad y la duración del movimiento enviado.

Figura 2.3.2. Página de las cámaras en emisión en la aplicación web

También es posible ver las cámaras registradas en el sistema. Se muestran en una tabla que tiene los siguientes elementos:

- Indicador de emisión en directo si se diera el caso
- Dirección MAC del dispositivo
- Dirección IP del dispositivo
- Dirección del streaming
- Botón para eliminar la cámara registrada

Figura 2.3.3. Página de las cámaras registradas en la aplicación web

Capítulo 3. Tecnologías

3.1 Tecnologías inalámbricas

En este Trabajo de Fin de Grado se hace uso de varias tecnologías inalámbricas.

Se utilizan las tecnologías Wi-Fi o 4G (LTE), para el envío de vídeo mediante streaming y para la conexión con la aplicación web cliente, dependiendo de si disponemos de una conexión Wi-Fi o si en cambio, tenemos que utilizar la red móvil.

Además, se utiliza la tecnología Bluetooth para la conexión del dispositivo móvil con el robot Lego MindStorms EV3.

3.1.1 Wi-Fi

Wi-Fi es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos que disponen de Wi-Fi pueden conectarse a Internet a través de un punto de acceso de red inalámbrica.

Tiene la ventaja de que al tratarse de una red inalámbrica, es posible conectarse a ella desde diferentes puntos dentro de un espacio amplio, sin la necesidad de utilizar cables. Por otro lado, tiene la desventaja de que debido a las interferencias, obstáculos y distancia con el punto de acceso inalámbrico, tendrá menor velocidad que las redes cableadas.

En cuando a la seguridad en las redes Wi-Fi, básicamente podemos distinguir tres tipos de cifrado: WEP, WPA y WPA2.

WEP fue el primer estándar de seguridad para redes Wi-Fi. Se basa en el cifrado de clave secreta en flujo RC4. Este tipo de cifrado no está recomendado ya que presenta grandes vulnerabilidades. Aunque la clave utilizada sea grande y compleja, cualquier *cracker* puede conseguirla sin demasiado esfuerzo.

WPA surgió para corregir las limitaciones de WEP. El cifrado sigue siendo RC4, aunque incluye distribución dinámica y automática de claves, nuevas técnicas de integridad y autentificación, y TKIP (Temporal Key Integrity Protocol) para la generación dinámica de claves de sesión.

WPA2 es la opción más segura en redes Wi-Fi. La variante WPA2-PSK es la opción más segura ya que garantiza el uso del cifrado en bloque Rijndael (AES).

3.1.2 4G (LTE)

4G o LTE (Long Term Evolution) hace referencia a redes móviles. Este tipo de red móvil mejora considerablemente su velocidad frente a su antecesora 3G. Las redes 3G alcanzan una velocidad de 7,2 Mbps frente a los 80 Mbps que alcanzan las redes 4G.

A pesar de que se dice que la red de telefonía móvil 4G surgió para mejorar la velocidad de su antecesora 3G, verdaderamente surgió porque se consiguió vulnerar la seguridad del cifrado en bloque Kasumi utilizado por las redes 3G.

4G se basa en el cifrado en flujo SNOW3G, basado en LFSRs (Linear Feedback Shift Register) o en español, RDRLs (Registro de Desplazamiento con Realimentación Lineal).

3.1.3 Bluetooth

Bluetooth es una especificación para Redes Inalámbricas de Área Personal (WPAN) que permite la transmisión de voz y datos entre diferentes dispositivos.

Bluetooth permite una distancia máxima entre dispositivos de 100 metros en un espacio abierto. Si hay obstáculos de por medio, la distancia máxima se va reduciendo.

Además, la especificación 4.0 o BLE (Bluetooth Low Energy), como su nombre indica permite su uso en modo de bajo consumo. También se pueden utilizar dispositivos Bluetooth Low Energy en modo baliza, aunque para ello es necesario que el dispositivo cuente con la versión de Bluetooth 4.1.

Bluetooth usa el cifrado en flujo E0. La clave de cifrado varía entre 8 y 128 bits, aunque se recomienda el uso de claves de 65 bits ya que con claves más

largas no se consigue mejorar la seguridad. Por otro lado, como se trata de una tecnología de uso en distancias cortas, se limita bastante la posibilidad de ataques.

La versión 4.0 o BLE basa su seguridad en el cifrado en bloque Rijndael (AES).

3.2 Lenguajes de programación

En esta sección se hace un breve recorrido por los diferentes lenguajes de programación que se utilizan en este Trabajo de Fin de Grado.

3.2.1 Android

Android no es un lenguaje de programación como tal, sino que es un sistema operativo con base Linux. Este sistema operativo fue diseñado principalmente para dispositivos móviles con pantalla táctil como tablets y smartphones, aunque en la actualidad también se encuentra disponible en smartwatchs, televisores y otros dispositivos.

Como lenguajes de programación para Android se utiliza Java y el lenguaje de marcas XML. Aun así, la programación en Android tiene sus particularidades respecto a Java.

Para el desarrollo de aplicaciones para este sistema operativo se usa el entorno de desarrollo Android Studio.

3.2.2 HTML

HTML (HyperText Markup Language) no es un lenguaje de programación como tal, sino que es un lenguaje de marcado para el desarrollo de documentos y páginas web.

Este lenguaje de marcado se basa en el uso de etiquetas, rodeadas de corchetes angulares (<etiqueta>), para definir los diferentes elementos del lenguaje.

La última versión es HTML5 y cuenta con bastantes características nuevas e interesantes que son de gran utilidad.

3.2.3 CSS

CSS (Cascading Style Sheets) es un lenguaje de estilos que se utiliza para definir la presentación y los estilos de un documento, escrito en HTML, como puede ser una página web.

Actualmente, este lenguaje se encuentra en la versión CSS3. En dicha versión aparecen novedades interesantes como son las transiciones y las animaciones. Gracias a ello, se puede sustituir la tecnología Flash con el uso de HTML5 junto a CSS3.

3.2.4 JavaScript

JavaScript es un lenguaje de programación interpretado que procede del estándar ECMAScript. Es un lenguaje que se utiliza en el lado del cliente (front-end), y permite crear acciones en las páginas web.

Los encargados de interpretar el código escrito en JavaScript son los navegadores web.

Actualmente, ECMAScript se encuentra en la versión 6, aunque no todos los navegadores han implementado todas las funcionas de esta última versión.

3.2.5 Node.js

Node.js es un entorno de ejecución JavaScript para la capa del servidor basado en el lenguaje de programación ECMAScript. Se puede decir que se trata de JavaScript desde el lado del servidor (back-end).

Node.js aprovecha el motor V8 de Google de forma que obtiene un entorno de ejecución del lado del servidor que compila y ejecuta JavaScript a grandes velocidades. El aumento de velocidad es importante ya que compila JavaScript en código máquina en lugar de interpretarlo.

Además, Node.js es de código abierto y se ejecuta en todos los sistemas operativos de escritorio.

3.3 Seguridad

Al tratarse de una aplicación segura, los datos que se envían entre las dos aplicaciones (aplicación principal Android y aplicación web cliente) están cifrados.

Para el cifrado de los datos se hace uso del algoritmo AES de 256 bits en modo CBC (Cipher Block Chaining) y para establecer la clave secreta compartida se utiliza el algoritmo Diffie Hellman de Curva Elíptica. Además, para la verificación se usa el algoritmo de Firma Digital de Curva Elíptica.

Seguidamente vemos de forma más detallada estos algoritmos criptográficos.

3.3.1 AES 256 CBC

El AES (Advanced Encryption Standard) es un algoritmo de cifrado de clave secreta que surgió para sustituir al DES (Data Encryption Standard). Este algoritmo de cifrado se eligió mediante un proceso de selección abierto en el que los participantes presentaban sus propuestas de forma pública.

Trabaja con una longitud de bloque de 128 bits y permite claves de longitud de 128, 192 o 256 bits. En cuanto al número de iteraciones, trabaja con 10, 12 o 14 iteraciones dependiendo de la longitud de clave escogida. Realiza las operaciones a nivel de byte, con palabras de 4 bytes.

Se compone de una fase inicial, una fase de rondas que se repite un número determinado de veces y una fase final. La fase de rondas se compone de cuatro operaciones realizadas en el siguiente orden:

- SubBytes: sustitución de cada byte por su recíproco.
- ShiftRows: desplazamiento de cada fila de bytes.
- MixColumns: producto de cada columna por una matriz.
- AddRoundKey: XOR de la subclave y la información del estado actual.

En la fase inicial se realiza una operación AddRoundKey, y en la fase final se realizan las operaciones SubBytes, ShiftRows y AddRoundKey en el orden nombrado.

Para el descifrado se hacen las operaciones inversas, en el orden inverso al llevado a cabo en la operación de cifrado.

Con el modo CBC (Cipher Block Chaining), a cada bloque de texto se le aplica la operación XOR con el bloque cifrado anterior antes de ser cifrado. Por ello, cada bloque de texto cifrado depende de todo lo cifrado hasta el momento.

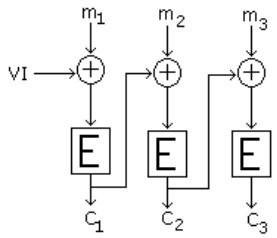


Figura 3.3.1.1. Modo de cifrado CBC

3.3.2 Diffie Hellman de Curva Elíptica

Cuando se utiliza un cifrado de clave secreta, como en este caso el AES, es necesario establecer la clave secreta compartida entre el emisor y el receptor del mensaje. Para ello, utilizamos en este trabajo el algoritmo Diffie Hellman de Curva Elíptica (ECDH).

La criptografía de curva elíptica es una variante de la criptografía de clave pública basada en las matemáticas de curva elíptica. Este tipo de criptografía permite el uso de claves más cortas y además es más rápida que la tradicional.

El algoritmo de Diffie Hellman es un mecanismo criptográfico de establecimiento de claves entre partes que no han tenido un contacto previo, en el que se utiliza un canal inseguro y de manera anónima. Gracias a este algoritmo se acuerda una clave secreta compartida, la cual no puede ser descubierta por un atacante aunque obtenga los mensajes intercambiados. La clave secreta acordada posteriormente será utilizada para cifrar y descifrar.

Para realizar el acuerdo de clave, cada una de las partes tiene una clave secreta y una clave pública. Suponiendo que son dos las partes que se quieren comunicar, se envían entre ellas las claves públicas por un canal inseguro y con la clave privada propia de cada uno y la clave pública del otro se calcula la clave secreta compartida. En ambos lados se genera la misma clave compartida sin conocer la clave privada de la otra parte, ya que de otro modo el algoritmo no tendría sentido.

En ECDH se suelen transformar las claves simétricas generadas usando alguna función hash, ya que el resultado obtenido de dicho tipo de función es fácil de calcular pero difícil de invertir.

3.3.3 Firma Digital de Curva Elíptica

La criptografía de clave pública permite el desarrollo de firmas digitales. La finalidad de las firmas digitales es la misma que la de la firma manuscrita, pero con la ventaja de que no es sencillo falsificarla al contrario que una firma hecha a mano.

La firma digital permite al receptor de un mensaje verificar la autenticidad del origen de la información, así como verificar que el mensaje no ha sido alterado; es decir, la firma digital ofrece autenticación e integridad de los datos, así como no repudio en el origen.

La criptografía de clave pública tiene el inconveniente de la velocidad, y además, la lentitud crece a medida que aumenta el tamaño del mensaje a cifrar. Para evitar esto, la firma digital hace uso de funciones hash. Una función hash es una operación que se realiza sobre un conjunto de datos y se obtiene, como resultado, otro conjunto de datos de un tamaño menor equivalente al conjunto original.

En cuanto a las claves pública y privada, en la firma digital se usan al contrario que a la hora de cifrar. En la operación de cifrado se usa la clave pública para cifrar y la privada para descifrar; en cambio, para firmar se utiliza la clave privada y para verificar la firma se utiliza la clave pública.

En el momento en el que se envía un mensaje firmado, también se envía la clave pública de la entidad firmante. Cuando el receptor recibe el mensaje, utiliza la clave pública que también le han enviado para verificar la autenticidad del emisor así como la integridad de los datos recibidos.

El algoritmo escogido para la firma es el ECDSA (Elliptic Curve Digital Signature Algorithm). Se trata de una modificación del DSA en el que se realizan operaciones sobre puntos de curvas elípticas en vez de las exponenciaciones que se usan en DSA. La ventaja de usar la versión de curva elíptica es que con tamaños menores de claves se puede brindar la misma seguridad que con tamaños mayores en la versión tradicional.

ECDSA se compone de tres fases. Las dos primeras son la generación de las claves y el proceso de firma, que las lleva a cabo el emisor, y la tercera es el proceso de verificación, que la lleva a cabo el receptor.

Capítulo 4.
Sistema de detección de movimiento

4.1 Definición

La detección de movimiento se utiliza en cámaras de videovigilancia para detectar movimiento producido delante de las cámaras. Esta es un característica muy útil en las cámaras de videovigilancia, pero su efectividad puede verse comprometida por falsas alarmas, ya que se si el algoritmo utilizado es demasiado sencillo puede ser engañado incluso por alguna pequeña señal de ruido.

Las falsas alarmas en la detección de movimiento pueden provocar que los sistemas hagan un uso ineficiente de los recursos, ya que podrían realizar acciones, como el almacenamiento de vídeo, en momentos en los que verdaderamente no ha habido un movimiento real.

La idea más sencilla para detectar movimiento sería comparar dos frames píxel a píxel, pero sería casi imposible que las dos imágenes sean exactamente iguales. Por tanto, esta primera idea es descartada, ya que seguramente siempre se estaría detectando movimiento aunque no lo hubiera.

Otra idea que sí que se utiliza normalmente en la detección de movimiento es la denominada substracción de vídeo. Consiste en que primero, al empezar a capturar las imágenes, se toma como referencia el primer fotograma, que es la imagen que se llama fondo. Luego, se compara cada frame con el fondo haciendo una resta entre las dos imágenes, de forma que se genera una nueva imagen en la que se puede observar dónde ha habido cambios. Para decidir si verdaderamente ha habido movimiento se define un umbral, el cual se compara con el tamaño de la zona en movimiento. Si la zona supera el umbral, se activa el aviso de movimiento; si no, queda como una falsa alarma.

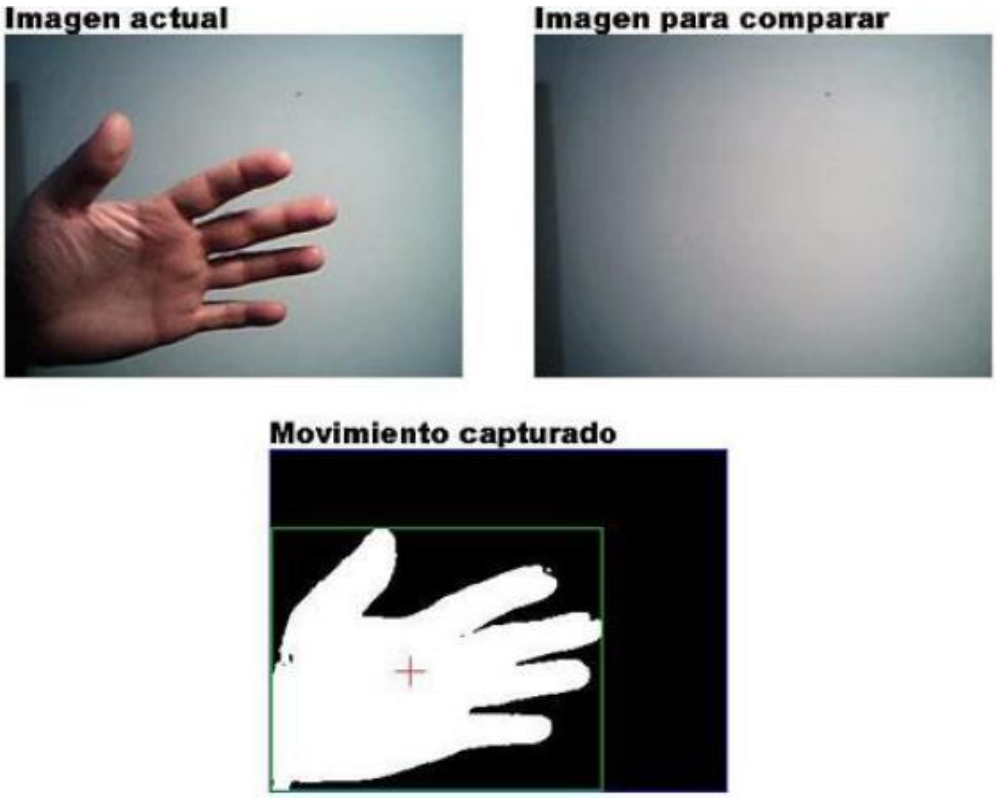


Figura 4.1.1. Detección de movimiento mediante substracción de vídeo

Los algoritmos de detección de movimiento más avanzados tienen en cuenta posibles cambios en las imágenes. De este modo, ante cambios de iluminación o pequeños movimientos, estos algoritmos gracias a unos umbrales de sensibilidad que se definen, son capaces de obviarlos y solamente detectar el movimiento real.

4.2 Funcionamiento

Para la detección de movimiento usamos una librería externa (Android Motion Detector) y la propia cámara del dispositivo móvil.

Además, entra en juego el sistema de notificaciones de la aplicación, que es el encargado de enviar notificaciones a través de mensajes de texto (SMS).

La detección de movimiento funciona de forma que, desde el momento en que se activa la cámara del dispositivo móvil, constantemente se hace la comparación del último frame del vídeo con los anteriores.

En el código referente a la detección de movimiento, existe un condicional que evalúa una expresión booleana de forma que cuando el condicional da como

resultado un valor verdadero, esto quiere decir que se ha detectado movimiento delante de la cámara.

Cuando esto ocurre, automáticamente, el smartphone se encarga de enviar un aviso mediante mensaje de texto a los destinatarios configurados con anterioridad. En dicho mensaje se comunica que ha habido movimiento y además se informa de la fecha y hora del acontecimiento. Además, el dispositivo conecta con el servidor de streaming Wowza y empieza a enviar el audio y vídeo en directo.

4.3 Librería Android Motion Detector

Para la detección de movimiento realizada en la aplicación principal, se utiliza la librería "Android Motion Detector". Se trata de una librería fácil de utilizar, que se basa en incluir tres líneas de código.

En la primera línea se obtienen los fotogramas de la cámara y se convierten al formato RGB. El resultado de la conversión se guarda en un array de enteros.

```
int[] rgb = ImageProcessing.decodeYUV420SPtoRGB(data, width, height);
```

Figura 4.3.1. Primera línea de la detección de movimiento

En la segunda línea se declara un objeto del tipo IMotionDetection, que es el encargado de hacer la detección de movimiento.

```
IMotionDetection detector = new RgbMotionDetection();
```

Figura 4.3.2. Segunda línea de la detección de movimiento

En la tercera línea se hace uso del método detect del objeto declarado anteriormente. A este método se le pasa por parámetros el array creado en la primera línea y devuelve verdadero si ha habido movimiento y falso si no.

```
boolean detected = detector.detect(rgb, width, height)
```

Figura 4.3.3. Tercera línea de la detección de movimiento

4.4 Obtención de la cámara del dispositivo

Para la detección de movimiento, lo primero que hay que hacer es obtener la cámara del smartphone y añadirle una Callback del tipo PreviewCallbackWithBuffer. Dentro de dicha callback, existe un método onPreviewFrame que obtiene constantemente los fotogramas de la previsualización de la cámara. Es en ese método donde se hace la detección de movimiento.

```
public void startCamara() throws IOException{
        cammov = Camera.open();
        cammov.addCallbackBuffer(new byte[3110400]);
        cammov.setPreviewCallbackWithBuffer(new Camera.PreviewCallback() {
            @Override
            public void onPreviewFrame(byte[] data, Camera camera) {
                cammov.addCallbackBuffer(data);
                int[] rgb = ImageProcessing.decodeYUV420SPtoRGB(data, 1152, 648);
                IMotionDetection detector = new RgbMotionDetection();
                boolean detected = detector.detect(rgb, 1152, 648);
```

Figura 4.4.1. Código de la detección de movimiento

Dentro del propio método onPreviewFrame, existe un condicional para la variable booleana **detected**, de modo que en el caso de que sea verdadera significa que ha habido movimiento y por tanto, realiza las acciones que tiene dentro.

```
if (detected) {
    cammov.release();
    notif.enviarSmsMovimiento();
    serv.iniciar(surfaceView, getApplicationContext());
    startStreaming();
}
```

Figura 4.4.2. Código de las acciones al detectar movimiento

Lo primero que hace la aplicación es liberar la cámara. Después envía una notificación mediante mensaje de texto informando del movimiento detectado. Seguidamente, prepara los elementos necesarios para el envío de vídeo mediante streaming (cámara, previsualización, conexión con el servidor de streaming, etc.), y finalmente empieza a mandar las imágenes al servidor.

Capítulo 5.
Sistema de streaming

5.1 Funcionamiento

Para el envío de vídeo mediante streaming hacemos uso de la librería "Libstreaming" junto a la cámara del teléfono móvil.

Como servidor de streaming usamos Wowza Streaming Engine, el cual dispone de una suscripción de prueba gratuita de seis meses.

El envío de vídeo hacia el servidor de streaming Wowza se hace mediante el protocolo seguro RTSP (Real Time Streaming Protocol), mientras que la recepción desde el servidor hacia la aplicación web cliente se hace a través del protocolo RTMP (Real Time Messaging Protocol).

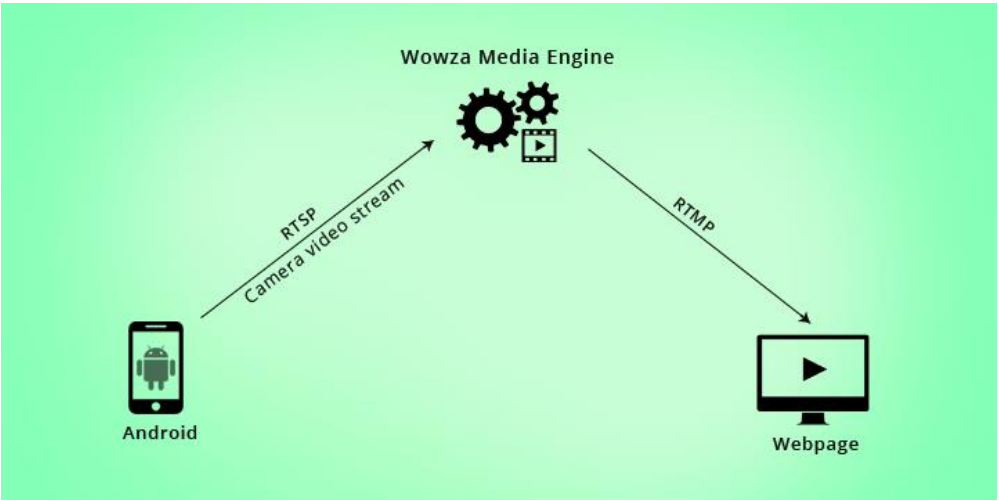


Figura 5.1.1. Protocolos de la comunicación vía streaming

Para enviar vídeo hacia el servidor, es necesario tener una aplicación autorizada en el servidor, así como un usuario y una contraseña. Si no tenemos estos datos, no podremos comunicarnos con el servidor Wowza.

Para, desde la aplicación Android, enviar el audio y el vídeo al servidor de streaming, la librería "Libstreaming" dispone de las funciones necesarias. Básicamente, necesitamos tener declarados dos objetos de la librería:

- Session: este objeto es el encargado de comunicarse con la cámara del dispositivo móvil y de las diferentes configuraciones en relación al audio y vídeo, como son el tamaño de la imagen y la calidad del audio.
- RtspClient: este otro objeto es el encargado de la comunicación con el servidor de streaming. En él se encuentran las credenciales que se necesitan para comunicarse con el servidor.

Para, desde la aplicación web cliente, recibir el vídeo desde el servidor, simplemente es necesario disponer de la dirección del servidor y del nombre de la transmisión. En nuestro caso, utilizamos la dirección MAC del dispositivo móvil como nombre de la transmisión. Así, en caso de tener varias cámaras, podemos distinguir fácilmente que transmisión queremos visualizar.

5.2 Protocolo RTSP

RTSP (Real Time Streaming Protocol) es un protocolo que se usa para establecer sesiones multimedia entre el cliente y el servidor. Verdaderamente, la transmisión de vídeo mediante streaming no es una tarea de este protocolo, si no que actúa como control remoto a través de la red para servidores multimedia.

RTSP utiliza el protocolo TCP (Transmission Control Protocol) para mantener una conexión punto a punto entre el cliente y el servidor y, además para enviar mensajes entre ellos. Para la transmisión de datos utiliza el protocolo UDP (User Datagram Protocol).

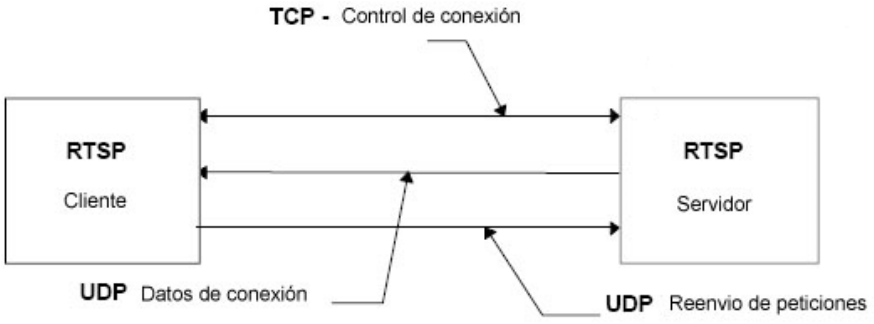


Figura 5.2.1. Protocolo RTSP

Las peticiones RTSP están basadas en peticiones HTTP y normalmente es el cliente el que se las envía al servidor. A continuación vemos las diferentes peticiones.

- DESCRIBE: Es el mecanismo que utiliza el servidor para comunicarse con el cliente. Se encarga de inicializar la sesión para el envío de vídeo. El flujo de comunicación va desde el cliente al servidor.
- OPTIONS: Esta petición es utilizada en el inicio para establecer la sesión con los parámetros necesarios. Puede ser enviada en cualquier momento. El flujo de comunicación va de cliente a servidor o de servidor a cliente.
- SETUP: Especifica los parámetros de transporte como el protocolo, el puerto para recibir los datos y el puerto para los metadatos. El flujo de comunicación va desde el cliente al servidor.
- PLAY: Esta petición indica al servidor que empiece a enviar los datos, utilizando, la configuración hecha con SETUP. El flujo de comunicación va desde el cliente al servidor.
- PAUSE: Esta petición indica al servidor que detenga temporalmente uno o todos los flujos de datos. El flujo de comunicación va desde el cliente al servidor.
- TEARDOWN: Esta petición detiene la entrega de datos para la dirección indicada y libera los recursos asociados. El flujo de comunicación va desde el cliente al servidor.
- REDIRECT: Con esta petición se informa al cliente de que ha cambiado la dirección del servidor y debe enviar los datos a la dirección nueva. El flujo de datos va desde el servidor hacia el cliente.
- GET_PARAMETER: Esta petición sirve para obtener algún parámetro de la sesión en curso. El flujo de datos puede ser de cliente a servidor y de servidor a cliente.
- SET_PARAMETER: Esta petición sirve para establecer el valor de algún parámetro de la sesión en curso. El flujo de datos puede ser de cliente a servidor y de servidor a cliente.

RTSP es un protocolo seguro; como protocolo de transporte usa TLS (Transport Layer Security) y además proporciona mecanismos de autenticación e integridad de los mensajes y de cifrado de los datos.

5.3 Protocolo RTMP

RTMP (Real Time Messaging Protocol) es un protocolo propietario que en un principio fue desarrollado por Macromedia y posteriormente, fue adquirido por Adobe. Es un protocolo que se utiliza para los servicios de streaming entre un servidor de audio y vídeo y un reproductor de vídeo que soporte Flash.

RTMP está disponible como una especificación libre para crear productos que permitan la entrega de audio, vídeo y datos en los formatos libres AMF, SWF, FLV y F4V.

Los paquetes RTMP pueden ser intercambiados a través de dos protocolos HTTP:

- En RTMP Tunneled (RTMPT), los datos RTMP son encapsulados e intercambiados a través del protocolo HTTP. Los mensajes son dirigidos a través del puerto 80.
- En RTMP Secure (RTMPS) los datos RTMP son encapsulados e intercambiados a través del protocolo HTTPS. Los mensajes son dirigidos a través del puerto 443 en esta versión. Se trata de un protocolo seguro que utiliza el protocolo de transporte TLS/SSL.

Los mensajes encapsulados en RTMPT y RTMPS son más grandes que los mensajes sin túnel RTMP debido a las cabeceras HTTP, HTTPS, RTMPT y RTMPS, pero facilitan la utilización de RTMP en escenarios en los que el uso de RTMP directamente no es posible, como puede ser cuando el cliente está detrás de un firewall que bloquea el tráfico RTMP.

Capítulo 6. Sistema de control de robot

6.1 Funcionamiento

Para controlar el robot Lego MindStorms EV3 hacemos uso de una librería externa (Lego MindStorms EV3 Android API) en la aplicación Android, y en la aplicación web hacemos uso de la librería net de Node.js.

Para la comunicación entre las dos aplicaciones, en la aplicación móvil se utiliza un socket servidor, y en la aplicación web cliente se utiliza un socket cliente.

La conexión entre el smartphone y el robot se hace mediante Bluetooth.

El control de robot entra en funcionamiento desde el momento en el que en la aplicación móvil accedemos a la pantalla de la cámara. Justo en ese momento, se crea un socket servidor que permite recibir las órdenes de robot enviadas a través de la aplicación web cliente.

A su vez, en la aplicación cliente, si se pulsa un botón de movimiento del robot se crea un socket cliente que se conecta a través de la red al socket servidor de la aplicación Android, y se le envía un mensaje diferente dependiendo del botón de movimiento pulsado.

Según el comando recibido desde la aplicación cliente, el teléfono, a través de Bluetooth y por medio de la librería Lego MindStorms EV3 Android API envía la orden adecuada al robot, dependiendo del movimiento, la velocidad y el tiempo que se ha indicado en la web.

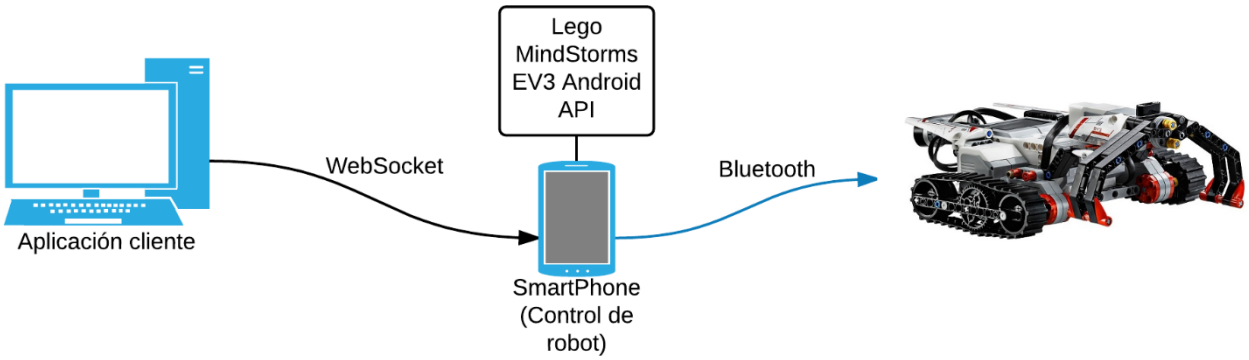


Figura 6.1.1. Esquema del control de robot

6.2 Sockets de comunicación

Para poder enviar comandos de movimiento al robot, desde la aplicación web cliente, es necesario que esta se comunique con la aplicación principal Android. Para establecer esta comunicación se hace uso de sockets. En la aplicación móvil se ha implementado un socket servidor que está en constante escucha. En la aplicación web se hace uso de sockets en modo cliente, los cuales se conectan con el socket servidor.

6.2.1 Socket servidor de la aplicación Android

En cuanto en la aplicación móvil se accede a la pantalla de la cámara desde la que se hace detección de movimiento y se envía el vídeo mediante streaming, se crea un socket en modo servidor que está escuchando en el puerto 1234.

```
public RobotSocket(Context ctx){
    globales = (GlobalClass) ctx.getApplicationContext();
    puerto = 1234;
    socketAbierto = true;
    btAdapter = BluetoothAdapter.getDefaultAdapter();
    crearSocket();
}

private void crearSocket(){
    threadSocket = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                serverSocket = new ServerSocket(puerto);
```

Figura 6.2.1.1. Código de la apertura del socket servidor

A través de este socket se reciben las órdenes de robot enviadas desde la aplicación web cliente. Estas órdenes se envían como cadena de texto, y en ellas se pueden distinguir: la dirección, la velocidad y el tiempo que tiene que durar el movimiento.

En cuanto se recibe una orden, se comprueba que efectivamente es un comando para el robot. Para ello, se verifica que la cadena de texto recibida empieza de la forma "legoev3". Seguidamente, se hace la conexión Bluetooth con el robot y se le envía el comando adecuado. Finalmente, se cierra la conexión con el robot y el socket servidor vuelve a la escucha de una nueva orden.

```
String[] comando = msg.split("-");
switch (comando[0]) {
    case "legoev3arriba":
        if (btAdapter != null) {
            if (btAdapter.isEnabled()) {
                globales.getRobot().conectar2(globales.getRobotElegido());
                robot.moverAdelante2(Integer.parseInt(comando[1]), Integer.parseInt(comando[2]));
                globales.getRobot().desconectar();
```

Figura 6.2.1.2. Código de la conexión con el robot

6.2.2 Socket cliente de la aplicación web

Para el socket cliente se utiliza la librería net de Node.js.

En la aplicación web, cada vez que pulsa un botón de movimiento del robot, se crea un socket de comunicación que conecta con la aplicación Android a través del puerto 1234 a la dirección IP del smartphone. Una vez está conectado, envía un comando en el que se indica la dirección de movimiento, la velocidad y el tiempo que debe estar realizando el movimiento el robot. Una vez enviado el comando se cierra la conexión ya que no necesitan estar constantemente conectados.

```
var net = require('net');
var client = net.connect(1234, request.body.ip);
client.on('error', function(e){
    console.log(e);
});
client.write('legoev3rotarizquierda-' + request.body.vel + '-' + request.body.tiempo + '\n');
client.end();
```

Figura 6.2.2.1. Código de la conexión del socket cliente

6.3 Librería Lego MindStorms EV3 Android API

La librería Lego MindStorms EV3 Android API permite a los dispositivos con sistema operativo Android interactuar con robots Lego MindStorms EV3 mediante Bluetooth.

Para establecer la conexión con el robot, hay que declarar un objeto del tipo Brick, al que se le pasa como parámetro un objeto del tipo BluetoothCommunication. Ambos tipos de objetos pertenecen a la librería.

```
public void conectar2(String nombre_robot){
    ev3 = new Brick(new BluetoothCommunication(nombre_robot));
    try {
        ev3.connect();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 6.3.1. Código de la conexión Bluetooth con el robot

Para enviar comandos al robot, hacemos uso del método directCommand del objeto de tipo Brick declarado anteriormente. A su vez, usamos el método timeMotorSpeed para enviar órdenes de movimiento al robot. Como parámetros, le pasamos:

- La velocidad de movimiento (entre 0 y 100)
- El tiempo que ha de estar realizando el movimiento (en milisegundos)
- El puerto de salida del robot al que se enviará la señal de movimiento

```
public void moverAdelante2(int velocidad, int tiempo){
    System.out.println("Robot hacia adelante");
    try {
        ev3.directCommand.timeMotorSpeed(velocidad, (tiempo * 1000), true, OutputPort.B);
        ev3.directCommand.timeMotorSpeed(velocidad, (tiempo * 1000), true, OutputPort.C);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 6.3.2. Código de orden de movimiento hacia el robot

Para cerrar la conexión Bluetooth con el robot, simplemente hay que ejecutar el método disconnect de la clase Brick.

Capítulo 7. RoboCAM

7.1 Definición

La aplicación principal Android combina los sistemas definidos en los capítulos anteriores: sistema de detección de movimiento, sistema de streaming y sistema de control de robot.

Esta combinación de los tres sistemas se hace mediante las diferentes clases de la aplicación. La detección de movimiento está definida en la clase "CameraActivity", el streaming está definido en la clase "Servidor", y el control de robot se encuentra en la clase "Robot" y en la clase "RobotSocket".

Para poder utilizar todas las funciones de RoboCAM, disponemos de una aplicación web cliente que, como ya hemos visto, está en comunicación con la aplicación móvil.

7.2 Funcionamiento

El sistema de detección de movimiento es el primero que entra en juego. En el momento en que se empiezan a captar imágenes a través de la cámara del dispositivo móvil, se empieza a hacer detección de movimiento, y hasta que este no detecta movimiento no entran en funcionamiento los otros dos sistemas.

Cuando se ha detectado movimiento a través de la aplicación, es cuando entran en juego el sistema de streaming y el sistema de control de robot. Justo en ese momento, se empieza a enviar vídeo mediante streaming, y además la aplicación abre el socket de comunicación para que cuando a través de él se reciba una orden, se haga la conexión con el robot para poder enviarle el comando de movimiento enviado desde la aplicación web. Una vez que el robot ha hecho el movimiento indicado, este se desconecta de la aplicación hasta que

vuelva a recibir otra orden. De esa manera, no está siempre conectado con la aplicación móvil, con la ventaja de que se ahorra batería en los dos dispositivos.

7.3 Comunicaciones

Los elementos que intervienen en las comunicaciones son los siguientes:

- Smartphone Android: en él se ejecuta la aplicación principal.
- Robot Lego MindStorms EV3: es el encargado de mover el Smartphone.
- Servidor: se compone del servidor de streaming Wowza y del servidor web.
- Ordenador: ejecuta la aplicación web cliente a través de un navegador.
- Router Wi-Fi: es elemento de interconexión entre los diferentes dispositivos para hacer las pruebas.

En el siguiente esquema podemos ver las comunicaciones que se producen entre la aplicación Android, la aplicación web cliente, el robot Lego MindStorms EV3 y el servidor.

Las flechas negras indican el flujo de comunicación del audio y vídeo mediante streaming, y las flechas rojas indican el flujo de comunicación del control de robot.

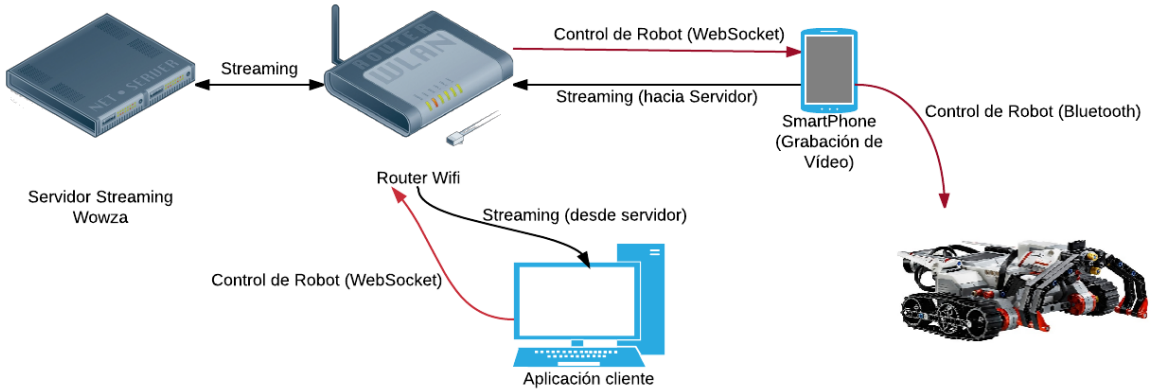


Figura 7.3.1. Esquema general de comunicaciones

7.3.1 Streaming

La aplicación Android envía el vídeo mediante streaming al servidor Wowza a través del protocolo seguro RTSP. Además, cuando se inicia el streaming, se envía una respuesta al servidor web, para que, en la aplicación cliente se muestre la transmisión de vídeo.

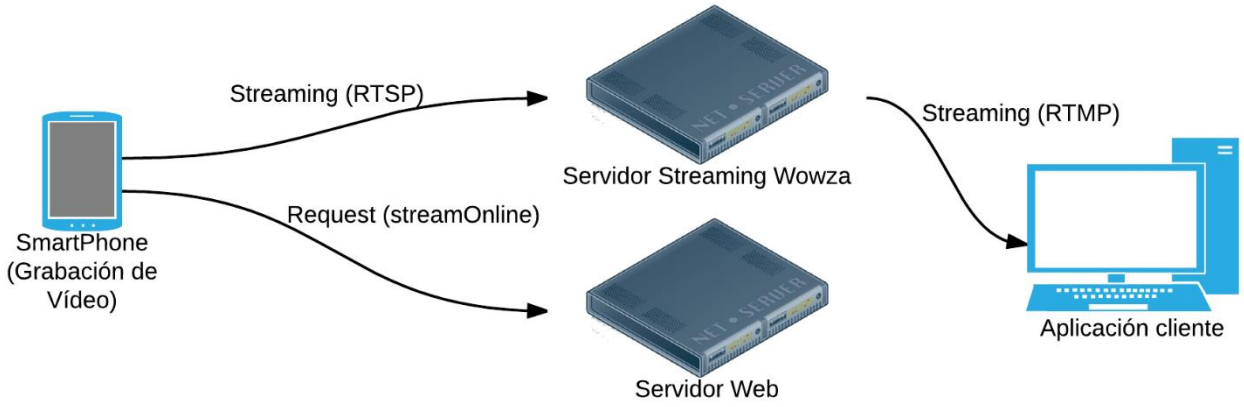


Figura 7.3.1.1. Esquema de comunicación streaming al empezar a emitir

La aplicación web recibe el vídeo mediante streaming desde el servidor Wowza a través del protocolo RTMP.

Cuando se termina de emitir el streaming, se envía una respuesta al servidor web para que la aplicación web deje de mostrar la ventana de la transmisión de vídeo.

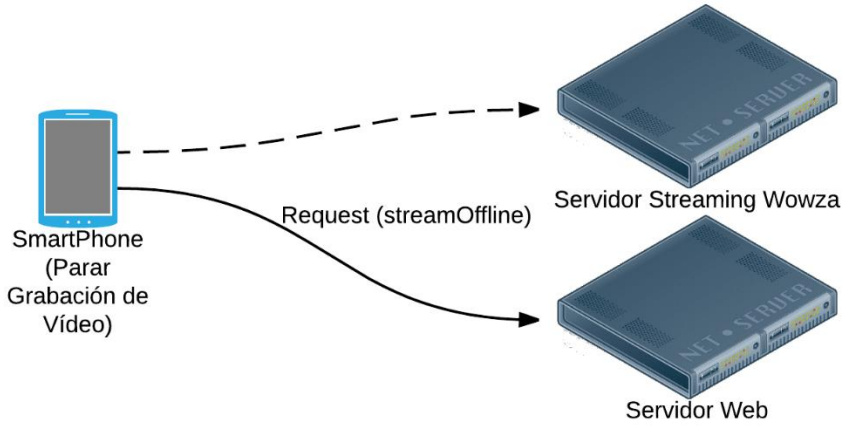


Figura 7.3.1.2. Esquema de comunicación streaming al parar la emisión

7.3.2 Control de robot

Para el control de robot, en el momento que se pulsa un botón en la aplicación web cliente, esta crea un socket que conecta con la aplicación Android gracias a la dirección IP de la aplicación principal y se envía un mensaje que es recibido por el smartphone. A su vez, la aplicación, dependiendo del mensaje recibido, mediante Bluetooth envía el comando correspondiente al robot para que se mueva en la dirección deseada.

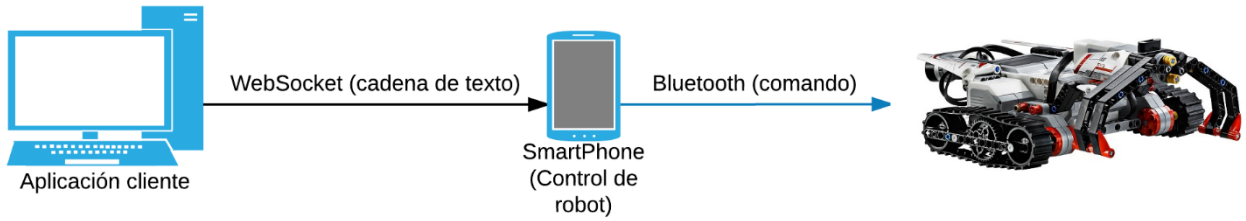


Figura 7.3.2.1. Esquema de comunicación del control de robot

7.4 Base de datos

Para guardar las configuraciones que hace el usuario en la aplicación Android RoboCAM, se utiliza una base de datos SQLite. Al abrir la aplicación se comprueba si existe la base de datos y si no es así se crea.

```
public void ini_bdd(){
    SQLiteDatabase bdd = openOrCreateDatabase("basededatos", MODE_PRIVATE, null);
    app_data = new BaseDeDatos(bdd);
    app_data.crearTablas();
}
```

Figura 7.4.1. Código de apertura o creación de la base de datos

A su vez pasa lo mismo con las tablas, que si no existen se crean.

```
public void crearTablas(){
    db.execSQL("CREATE TABLE IF NOT EXISTS destinatarios(nombre VARCHAR, numero VARCHAR, activo BOOLEAN);");
    db.execSQL("CREATE TABLE IF NOT EXISTS servidor(url VARCHAR, user VARCHAR, pass VARCHAR, web VARCHAR);");
    db.execSQL("CREATE TABLE IF NOT EXISTS robot(nombre VARCHAR);");
}
```

Figura 7.4.2. Código de creación de las tablas de la base de datos

Las tablas que se crean son las que se muestran en la siguiente tabla.

Tabla	Descripción	Campos
destinatarios	Almacena los destinatarios a los que se tiene que notificar en caso de que se detecte movimiento a través de la cámara del dispositivo móvil.	Nombre del destinatario, número del destinatario, Destinatario habilitado o deshabilitado.
servidor	Almacena una única fila en la que se encuentran los datos que utiliza la aplicación Android para conectar con el servidor de streaming y con el servidor web.	URL del servidor de streaming, nombre de usuario, contraseña, URL del servidor web.
robot	Almacena una única fila en la que se encuentra en nombre del robot elegido para controlar desde la aplicación móvil.	Nombre del robot.

Tabla 7.4.1. Tablas de la base de datos de la aplicación móvil

Para la tabla destinatarios, se han implementado varios métodos que permiten añadir, eliminar y editar destinatarios en la tabla.

```
public void addDestinatario(String numero){
    db.execSQL("INSERT INTO destinatarios VALUES('" + "Sin nombre" + "', '" + numero +"', 1);");
}

public void delDestinatario(String numero){
    db.execSQL("DELETE FROM destinatarios WHERE numero = " + numero + ";");
}

public void updateDestinatarioNombre(String numero, String nombre_nuevo){
    db.execSQL("UPDATE destinatarios SET nombre = '" + nombre_nuevo + "' WHERE numero = " + numero + ";");
}

public void updateDestinatarioEstado(String numero, int estado){
    db.execSQL("UPDATE destinatarios SET activo = " + Integer.toString(estado) + " WHERE numero = " + numero + ";");
}
```

Figura 7.4.3. Métodos para interactuar con la tabla destinatarios

Para la tabla servidor, se ha creado un método que permite insertar los datos del servidor en caso de que no haya datos aún, o actualizarlos si ya se han insertado previamente.

```
public void updateServidor(String url, String user, String pass, String web){
    db.execSQL("DELETE FROM servidor");
    db.execSQL("INSERT INTO servidor VALUES('" + url + "', '" + user + "', '" + pass + "', '" + web + "');");
}
```

Figura 7.4.4. Método para actualizar la tabla servidor

Para la tabla robot, se ha desarrollado un método que permite insertar el nombre del robot elegido por el usuario o actualizar el nombre del robot en caso de que ya haya sido insertado.

```
public void updateRobot(String nombre){
    db.execSQL("DELETE FROM robot");
    db.execSQL("INSERT INTO robot VALUES('" + nombre + "')");
}
```

Figura 7.4.5. Método para actualizar la tabla robot

La finalidad de esta base de datos es guardar los datos de configuración del usuario, para que cuanto este cierre la aplicación queden guardados. De esta forma, al volver a abrirla en otro momento, se cargan sus preferencias desde la propia base de datos y no tiene que volver a introducirlos.

Para cargar la configuración del usuario al iniciar la aplicación se han implementado varios métodos. A continuación vemos los más importantes.

```
public void cargarDesdeBDD(String[] datos){
    stream_url = datos[0];
    publisher_user = datos[1];
    publisher_pass = datos[2];
    web_url = datos[3];
}
```

Figura 7.4.6. Método para cargar los datos del servidor

```
public String[] getServData(){
    String[] datos = new String[4];
    Cursor result = db.rawQuery("SELECT * FROM servidor;", null);
    while(result.moveToNext()){
        datos[0] = result.getString(0);
        datos[1] = result.getString(1);
        datos[2] = result.getString(2);
        datos[3] = result.getString(3);
    }
    return datos;
}
```

Figura 7.4.7. Método para obtener los destinatarios desde la base de datos

```
public void cargarDesdeBDD(ArrayList<Contacto> lista){
    for(int i = 0; i < lista.size(); i++){
        destinatarios.add(lista.get(i));
    }
}
```

Figura 7.4.8. Método para cargar los destinatarios

Capítulo 8. Implementación

8.1 Estructura de directorios

Seguidamente veremos los ficheros más importantes de las aplicaciones.

8.1.1 Aplicación Android

A continuación se presentan los diferentes ficheros que forman parte de la aplicación.

En la tabla siguiente, podemos ver las diferentes clases que se han creado para la aplicación Android.

Clase	Descripción
AdaptadorListaContactos	Donde se define el comportamiento de la lista de contactos para la pantalla de configuración de las notificaciones.
BaseDeDatos	Contiene la base de datos SQLite en donde se guarda la configuración de la aplicación.
CameraActivity	Donde se define la detección de movimiento y donde se da la ejecución del streaming.
Contacto	Donde se definen los campos que tendrán los objetos de tipo Contacto.
GlobalClass	Clase global donde están contenidos los objetos globales a toda la aplicación.
MainActivity	Clase principal de la aplicación.
MainActivityFragment	Donde se define el comportamiento de los botones del menú principal.
Notificaciones	Donde se definen las diferentes funcionalidades de la pantalla de configuración de las notificaciones.
NotificacionesActivity	Donde se define el comportamiento de los botones y otros elementos de la pantalla de configuración de las notificaciones.
Request	Donde están definidas las comunicaciones con el servidor web.
Robot	Donde se define la conexión y los movimientos del robot.
RobotActivity	Donde se define la búsqueda de dispositivos Bluetooth para elegir el robot.
RobotSocket	Donde está definido el socket de comunicación para el control de robot.
Servidor	Donde se define el envío de vídeo mediante streaming.
ServidorActivity	Donde se hace la configuración de las conexiones con los servidores web y de streaming.

Tabla 8.1.1.1. Clases de la aplicación Android

En la siguiente tabla, podemos ver las diferentes layouts que se han creado para la aplicación Android.

Layout	Descripción
activity_camera	Donde se define la pantalla de la cámara.
Activity_main	Plantilla de la actividad principal de la aplicación.
Activity_notificaciones	Donde se define la pantalla de configuración de las notificaciones.
Activity_robot	Donde se define la pantalla de elección del robot.
Activity_servidor	Donde se define la pantalla de configuración de los servidores.
Contacto_layout	Donde se define el formato en el que se muestran los contactos en la pantalla de configuración de las notificaciones.
Content_main	Contiene el menú principal de la aplicación.
Fragment_main	Donde se definen los botones del menú principal de la aplicación.

Tabla 8.1.1.2. Layouts de la aplicación Android

Otros ficheros importantes que se usan en esta aplicación son los siguientes.

Fichero	Descripción
colors.xml	Donde se definen los diferentes colores que se usan en la aplicación.
dimens.xml	Donde se definen los diferentes tamaños que se usan en la aplicación.
strings.xml	Donde se definen las cadenas de texto que se utilizan en la aplicación

Tabla 8.1.1.3. Otros ficheros de la aplicación Android

8.1.2 Aplicación web cliente

La siguiente tabla, incluye un breve resumen de los ficheros y directorios más importantes de la aplicación web cliente.

Fichero	Descripción
app.js	Fichero principal de la aplicación web.
package.json	En este fichero se define el nombre y versión de la aplicación así como las dependencias a las librerías que se van a utilizar.
routes/routes.js	Las rutas a las que acceder, según las peticiones que se manden, se encuentran en el fichero routes.js.
app/models/camara.js	En este fichero se define la tabla cámara de la base de datos.
app/controllers/camaras.js	Fichero en el que se encuentran las funcionalidades de la sección de cámaras de la aplicación así como del control de robot.
app/views/_head.ejs	Se trata de una vista parcial. Contiene el head (cabecera) de la web con las librerías necesarias.
app/views/_navbar.ejs	Se trata de una vista parcial. Contiene el menú principal de la aplicación web.
app/views/_footer.ejs	Se trata de una vista parcial. Contiene el footer (pie) de la web.
app/views/home.ejs	En este fichero está definida la página principal de la web.

app/views/camaras.ejs	En esta vista se define la forma en que se muestran las cámaras cuando están emitiendo vídeo mediante streaming así como el control de robot para cada una de ellas.
app/views/list_camara.ejs	Este fichero es la vista en la que se muestran las cámaras que han sido añadidas, a la base de datos de la aplicación, al conectarse por primera vez.
public/css/videovigilancia.css	Esta fichero CSS contiene los estilos personalizados de la web. En él se define el tamaño de la zona de vídeo así como los diferentes botones para el control de robot.
public/img/favicon	Directorio en el que se encuentra los favicon de la web para diferentes tipos de dispositivos y pantallas.

Tabla 8.1.2.1. Ficheros importantes de la aplicación web

8.2 Problemas y soluciones

En este punto se presentan los problemas surgidos durante el desarrollo de este Trabajo de Fin de Grado así como las diferentes soluciones que se han adoptado para resolverlos.

8.2.1 Obtención de la cámara por dos APIs

Uno de los principales problemas en el desarrollo de la aplicación Android surgió al querer que dos librerías accedieran a la vez a la cámara del dispositivo móvil.

En un primer momento se pretendía que la aplicación, a la vez que transmitía el vídeo mediante streaming, hiciera detección de movimiento de modo que, aunque no se hubiera detectado movimiento aún, se pudieran ver las imágenes.

Finalmente, la solución que se puso a este problema fue hacer detección de movimiento en un principio, y justo cuando se detectara movimiento empezar a enviar el vídeo mediante streaming. De este modo, la librería de detección de movimiento estaría usando la cámara en un primer momento, y luego la dejaría libre para que pudiera usarla la librería de streaming.

Por otra parte, se decidió aplicar esta solución para proteger la privacidad de los usuarios, ya que es mejor que solamente se envíen las imágenes si de verdad es necesario. Es decir, que solo se envíen las imágenes mediante streaming si el dispositivo móvil ha detectado movimiento con su cámara.

8.2.2 Conexión con el robot

Al empezar a trabajar con el robot Lego MindStorms EV3, empezaron los verdaderos problemas con la aplicación Android.

El primero fue conseguir establecer la comunicación Bluetooth con el robot. Para ello, se disponía de una API que parecía permitir la conexión con el robot así como enviarle las órdenes necesarias. En principio era bastante fácil usar dicha librería, pero no era capaz de conectar con el robot. Tras buscar información, se vio que cada dispositivo Bluetooth dispone de unos identificadores únicos universales (UUID, Universally Unique Identifier), y hay que hacer la conexión utilizando alguno de los identificadores del dispositivo. La solución fue cambiar el identificador que estaba por uno de los del robot en la API.

El segundo problema que surgió fue que el robot, pasados aproximadamente treinta segundos, dejaba de responder a las órdenes que se le enviaban. La conexión estaba hecha de forma que la aplicación Android siempre estaba conectada con robot mediante Bluetooth, y se desconectaba cuando se cerraba la aplicación en el smartphone. La solución a este problema fue, cada vez que se le quisiera enviar una orden al robot, establecer la conexión, enviar la orden y cerrar la conexión. Por otra parte, de este modo también se gana en ahorro de energía tanto en el teléfono móvil como en el robot.

Cuando parecía que todo estaba funcionando correctamente, pudimos observar que si mandábamos al robot un comando muy seguido del otro, la aplicación Android fallaba y se cerraba. Esto se debía a que no daba tiempo a cerrarse la conexión anterior y abrirse la nueva, por lo que en ese momento, el flujo de salida Bluetooth era nulo. La solución que se adoptó para solucionar este problema fue poner un condicional en el flujo de salida de la API, de modo que solo mande mensajes al robot mediante Bluetooth si el flujo de salida no es nulo. De este modo, cuando se manden mensajes muy seguidos de forma que no se puedan enviar, se obviarán y así la aplicación no fallará.

8.2.3 Sockets de comunicación

Una de las partes fundamentales de este Trabajo de Fin de Grado es la comunicación entre la aplicación web cliente y la aplicación Android.

Principalmente, será la aplicación web cliente la encargada de enviar mensajes a la aplicación móvil.

La finalidad de esta comunicación es enviar una serie de comandos desee la web hacia el smartphone de modo que cuando este último los recibe, dependiendo del comando recibido efectúa una acción u otra. Estas acciones que va a ejecutar son simplemente la comunicación con el robot así como los movimientos que tiene que hacer.

En un principio, se pensó en utilizar la API WebSocket de HTML5, ya que desde un primer momento se vio que era bastante fácil de utilizar, aunque finalmente no fue así.

Respecto a una aplicación de prueba que se utilizó al principio para hacer las pruebas, con la API WebSocket era necesario iniciar el proceso de handshake entre el socket servidor (aplicación Android) y el socket cliente (aplicación web) para establecer conexión. Finalmente, la aplicación móvil era capaz de enviar la cabecera necesaria para establecer conexión, pero otro problema hizo que se descartara esta API; al enviar los mensajes desde la aplicación cliente, el teléfono los recibía en una codificación diferente que fue imposible de leer.

Buscando posibles soluciones, se encontró la librería Net de Node.js. Una de las principales ventajas de usar esta librería es que pasaríamos de usar WebSocket desde el front-end a usar Net desde el back-end, por lo que se puede incluso cifrar los mensajes que se envían para el control de robot. Si estuvieran en el front-end, aunque se cifraran, cualquier usuario podría ver los mensajes en el código.

Una vez implementadas las funciones para enviar los comandos de robot desde la aplicación web cliente surgieron algunos problemas más. Uno de ellos fue que al enviar un comando con la aplicación móvil apagada o al desconectar la aplicación móvil, la aplicación cliente daba un error de conexión. Para solucionar esto, bastó con capturar los errores del socket de comunicación para que la aplicación cliente no se cerrara es estos casos.

Otro de los problemas que surgió fue respecto a la aplicación Android. Hubo que adaptar el Socket servidor de la aplicación para que funcionara de modo que los clientes (en este caso la aplicación web) se conecten, manden el mensaje y se desconecten, ya que al cerrar la conexión en el cliente, fallaba la aplicación móvil.

Capítulo 9. Presupuesto

9.1 Personal

Para este Trabajo de Fin de Grado se estiman 300 horas de trabajo para una persona. Si la hora de trabajo se valora en 15€, tenemos un total de 4500€ en el coste del personal.

Estas 300 horas se dividen en pequeñas tareas que hay que realizar para la consecución del Trabajo de Fin de Grado. A continuación las vemos en más detalle.

Hito	Tarea	Horas
Toma de información	Prueba de aplicaciones Android de vigilancia y detección de movimiento	10
	Búsqueda de información del desarrollo en Android	15
Web Informativa	Desarrollo de la web informativa del proyecto	20
	Añadir información a la web del proyecto	4
Interfaz Gráfica	Desarrollo del menú principal de la aplicación Android	1
	Desarrollo de la pantalla de la cámara	10
	Desarrollo de la pantalla de configuración del servidor	5
	Desarrollo de la pantalla de configuración del robot	5
	Desarrollo de la pantalla de configuración de las notificaciones	10
Streaming	Pruebas con la cámara del dispositivo Android	5
	Búsqueda de información sobre streaming	15
	Implantación de la librería "libstreaming"	10
Control de robot	Prueba de aplicaciones Android para el control del robot Lego MindStorms EV3	10
	Búsqueda de información sobre Bluetooth en Android y pruebas en la aplicación	5
	Búsqueda de una librería para controlar el robot Lego MindStorms EV3	10
	Implantación de la librería "Lego MindStorms EV3 Android API"	15
	Corrección de errores de la librería "Lego MindStorms EV3 Android API" para funcionar correctamente con el robot	10
	Desarrollo del socket servidor de comunicación para el control de robot	20
Detección de movimiento	Búsqueda de algoritmos de detección de movimiento	10
	Búsqueda de librerías de detección de movimiento para Android	20
	Implantación de la librería "Android Motion Detector"	4

	Acoplar la detección de movimiento junto con el envío de streaming	10
Aplicación web	Diseño de la aplicación web	6
	Desarrollo del socket cliente para el control de robot	30
Seguridad	Implementación de los mecanismos de seguridad en la aplicación Android	20
	Implementación de los mecanismos de seguridad en la aplicación web	20

Tabla 9.1.1. Horas empleadas en el desarrollo de las tareas

9.2 Componentes

Aunque no ha habido que comprar nada, ya que el alumno disponía de algunos componentes y la Universidad de La Laguna de otros, a continuación podemos ver el coste de los componentes que se han utilizado para el desarrollo del Trabajo de Fin de Grado.

Artículo	Unidades	Precio unidad	Precio
Samsung Galaxy S5	1	400€	400€
Samsung Galaxy Ace 2	1	109€	109€
Lego MindStorms EV3	1	350€	350€
Suscripción de prueba Wowza	1	0€	0€
TOTAL			859€

Tabla 9.2.1. Presupuesto de los componentes

9.3 Coste total

Sumando el coste del personal necesario para el desarrollo del proyecto y de los componentes, tenemos los datos siguientes.

Artículo	Unidades	Precio unidad	Precio
Horas de trabajo	300 horas	15€	4.500€
Samsung Galaxy S5	1	400€	400€
Samsung Galaxy Ace 2	1	109€	109€
Lego MindStorms EV3	1	350€	350€
Suscripción de prueba Wowza	1	0€	0€
TOTAL			5.359€

Tabla 9.3.1. Presupuesto total

Como vemos en la tabla anterior, el precio total del proyecto que se calcula sumando el coste del personal y de los componentes, asciende a 5.359€.

Capítulo 10. Conclusiones y trabajos futuros

A modo de conclusión, con este Trabajo de Fin de Grado se ha diseñado e implementado un sistema de videovigilancia diferente a los típicos que podemos encontrar en una tienda. Se trata de un sistema de videovigilancia inteligente, al cual es posible añadirle nuevas funcionalidades.

En cualquier lugar, a través de Internet podremos ver las imágenes que captan las cámaras de los dispositivos móviles a tiempo real. También podremos mover las cámaras gracias a los robots que las soportan.

Además, aprovechando viejos smartphones Android que ya no se estén utilizando, ahorraremos bastante en los costes.

A este Trabajo de Fin de Grado se le podrían añadir más funcionalidades en el futuro, como pueden ser:

- Enviar el vídeo en horizontal o en vertical, dependiendo de la orientación que disponga el dispositivo móvil en ese momento.
- Enviar desde el dispositivo móvil, en un correo electrónico, un fragmento del vídeo una vez se detecte movimiento.
- Almacenamiento de los vídeos, en el servidor, durante un tiempo para poder visualizarlos en otro momento.
- Mejoras en la interfaz de la aplicación Android.
- Implementar el sistema de registro de usuarios en la aplicación web cliente.

Capítulo 11. Conclusions and future works

To conclude we have a different system from the typical video surveillance that it can be found in a store. We will have an intelligent video surveillance system, which we will be able to develop new features in the future.

Anywhere, via the Internet, we can view images captured by the cameras of mobile devices in real time. We may also move the cameras thanks to robots that support them.

In addition, using old Android smartphones that are no longer being used, we'll save on costs.

To this Final Degree Project we could add more features in the future, such as:

- Sending video horizontally or vertically, depending on the orientation available in the mobile device at the time.
- Sending from the mobile device, in an email, a fragment of the video once motion is detected.
- Storing videos on the server for a while to watch them later.
- Improvements in Android application inferface.
- Implement user registration on the web client application.

Bibliografía

[1] El País. España es el tercer país de la UE que sufre más robos en las tiendas. http://politica.elpais.com/politica/2015/11/04/actualidad/1446665088_986375.html

[2] La Nueva España. En España se registran más de 310 robos en casas al día. http://www.lne.es/sociedad/2016/02/10/espana-registran-310-robos-viviendas/1881035.html

[3] Eurostats Statistics Explained. Crime statistics. http://ec.europa.eu/eurostat/statistics-explained/index.php/Crime_statistics#Robbery

[4] Lego. Lego MindStorms EV3. http://www.lego.com/es-es/mindstorms/about-ev3

[5] Wikipedia. Wifi. https://es.wikipedia.org/wiki/Wifi

[6] Wikipedia. Telefonía móvil 4G. https://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_4G

[7] Wikipedia. Bluetooth. https://es.wikipedia.org/wiki/Bluetooth

[8] Androidhive. Android Streaming Live Camera Video to Web Page. http://www.androidhive.info/2014/06/android-streaming-live-camera-video-to-web-page/

[9] Wowza. Wowza Streaming Engine. https://www.wowza.com/products/streaming-engine

[10] Libstreaming. https://github.com/fyhertz/libstreaming

[11] Android Motion Detector. https://github.com/jkwiecien/android-motion-detector

[12] Lego MindStorms EV3 Android API. https://github.com/cqjjjzr/LEGO-MINDSTORMS-EV3-Android-API

[13] Lego MindStorms EV3 Android API Sample. https://github.com/cqjjjzr/LEGO-MINDSTORMS-EV3-Android-API-Sample

[14] WebTCP: Making TCP connections from Browser. http://artemyankov.com/tcp-client-for-browsers/

[15] Node.js Net API. https://nodejs.org/api/net.html

[16] Wikipedia. HTML. https://es.wikipedia.org/wiki/HTML

[17] Wikipedia. Hoja de estilos en cascada. https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

[18] Wikipedia. JavaScript. https://es.wikipedia.org/wiki/JavaScript

[19] Wikipedia. Node.js. https://es.wikipedia.org/wiki/Node.js

[20] Biopus. Algoritmos de captura óptica de movimiento por substracción de video. http://www.biopus.com.ar/txt/textos/Emiliano_Causa___Algoritmos_de_Captura_de_Movimiento.pdf

[21] Wikipedia. Diffie-Hellman. https://es.wikipedia.org/wiki/Diffie-Hellman

[22] Wikipedia. Criptografía de curva elíptica. https://es.wikipedia.org/wiki/Criptograf%C3%ADa_de_curva_el%C3%ADptica

[23] FNMC. Criptografía de Clave Asimétrica. Firma digital. http://www.cert.fnmt.es/content/pages_std/html/tutoriales/tuto7.htm

[24] Cifrado AES y RSA. https://www.boxcryptor.com/es/cifrado

[25] Elliptic Curve Cryptography: ECDH and ECDSA. http://andrea.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/

Apéndice A.
Código destacable

Entre todo el código que se ha desarrollado en la aplicación Android podemos destacar los siguientes fragmentos.

- Código de la detección de movimiento

```
if(camRelease && !emitiendo && !serv.getClient().isStreaming()){
    cammov = Camera.open();
    cammov.setDisplayOrientation(90);

    Camera.Parameters param = cammov.getParameters();
    param.setPreviewFrameRate(30);
    param.setPreviewFpsRange(15000, 30000);
    cammov.setParameters(param);
    try {
        cammov.setPreviewDisplay(surfaceView.getHolder());
    } catch (IOException e) {
        e.printStackTrace();
    }
    cammov.addCallbackBuffer(new byte[3110400]);
    contadorFrame = 0;
    cammov.setPreviewCallbackWithBuffer(new Camera.PreviewCallback() {
        @Override
        public void onPreviewFrame(byte[] data, Camera camera) {
            contadorFrame++;

            cammov.addCallbackBuffer(data);
            int[] rgb = ImageProcessing.decodeYUV420SPtoRGB(data, 1152, 648);
            IMotionDetection detector = new RgbMotionDetection();
            boolean detected = detector.detect(rgb, 1152, 648);

            if (detected && contadorFrame > 1) {
                cammov.release();
                notif.enviarSmsMovimiento();
                serv.iniciar(surfaceView, getApplicationContext());
                startStreaming();
                emitiendo = true;
                camRelease = true;
            }
            contadorFrame = 2;
        }
    });
    cammov.startPreview();
    camRelease = false;
}
```

Figura A.1. Código de la detección de movimiento de la aplicación Android

- Código del envío de streaming

```
public void iniciar(SurfaceView sView, Context context){
    surfaceView = sView;
    session = SessionBuilder.getInstance()
            .setCallback(this)
            .setSurfaceView(surfaceView)
            .setPreviewOrientation(90)
            .setContext(context)
            .setAudioEncoder(SessionBuilder.AUDIO_AAC)
            .setAudioQuality(new AudioQuality(8000, 16000))
            .setVideoEncoder(SessionBuilder.VIDEO_H264)
            .setVideoQuality(new VideoQuality(1280, 720, 120, 500000))
            .build();
    surfaceView.getHolder().addCallback(this);

    client = new RtspClient();
    client.setSession(session);
    client.setCallback(this);
    surfaceView.setAspectRatioMode(SurfaceView.ASPECT_RATIO_PREVIEW);

    Pattern uri = Pattern.compile("rtsp://(.+):(\\d+)/(.+)");
    Matcher m = uri.matcher(stream_url);
    m.find();
    ip = m.group(1);
    port = m.group(2);
    path = m.group(3);

    client.setCredentials(publisher_user, publisher_pass);
    client.setServerAddress(ip, Integer.parseInt(port));
    client.setStreamPath("/" + path + "/" + macAddress);
}

public void iniciarStreaming(){
    if(!client.isStreaming()){
        session.startPreview();
        client.startStream();
        Request.newUser(macAddress, requestQueue, ip + ":" + port + "/" + path + "/", web_url, ip_actual);
    }else{
        session.stopPreview();
        client.stopStream();
    }
}

@Override
public void onSessionStarted() {
    Request.streamOnline(macAddress, requestQueue, web_url, ip_actual, ip + ":" + port + "/" + path + "/");
}

@Override
public void onSessionStopped() {
    Request.streamOffline(macAddress, requestQueue, web_url);
}
```

Figura A.2. Código del streaming de la aplicación Android

- Código del control de robot

```
public class Robot {
    private Brick ev3;

    public Robot(){}

    public void conectar2(String nombre_robot){
        ev3 = new Brick(new BluetoothCommunication(nombre_robot));
        try {
            ev3.connect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void desconectar(){
        try {
            ev3.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void moverAdelante2(int velocidad, int tiempo){
        try {
            ev3.directCommand.timeMotorSpeed(velocidad, (tiempo * 1000), true, OutputPort.B);
            ev3.directCommand.timeMotorSpeed(velocidad, (tiempo * 1000), true, OutputPort.C);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void moverAtras2(int velocidad, int tiempo){
        try {
            ev3.directCommand.timeMotorSpeed(-(velocidad), (tiempo * 1000), true, OutputPort.B);
            ev3.directCommand.timeMotorSpeed(-(velocidad), (tiempo * 1000), true, OutputPort.C);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void moverIzquierda2(int velocidad, int tiempo){
        try {
            ev3.directCommand.timeMotorSpeed(velocidad, (tiempo * 1000), true, OutputPort.C);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Figura A.3. Código del control de robot de la aplicación Android

- Código del socket servidor para el control de robot

```
private void crearSocket(){
    threadSocket = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                serverSocket = new ServerSocket(puerto);
            } catch (IOException e) {
                e.printStackTrace();
            }

            while(socketAbierto){
                if(serverSocket == null) break;
                try {
                    clientSocket = serverSocket.accept();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                try {
                    BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

                    int caracter = -1;
                    String msg = "";
                    while ((caracter = in.read()) != 10) {
                        msg += (char) caracter;
                    }

                    String[] comando = msg.split("-");
                    switch (comando[0]) {
                        case "legoev3arriba":
                            if (btAdapter != null) {
                                if (btAdapter.isEnabled()) {
                                    globales.getRobot().conectar2(globales.getRobotElegido());
                                    robot.moverAdelante2(Integer.parseInt(comando[1]), Integer.parseInt(comando[2]));
                                    globales.getRobot().desconectar();
                                }
                            }
                            break;
```

Figura A.4. Código del socket de la aplicación Android

También podemos destacar algunos fragmentos de código relacionados con la aplicación web cliente.

- Código de las peticiones al añadir cámaras, al empezar y al terminar la emisión de vídeo mediante streaming

```
/* Cámara nueva */
exports.new = function (request, response) {
    if ( Utilities.isEmpty(request.body.name)) return response.send(error_400);
    if ( Utilities.isEmpty(request.body.server)) return response.send(error_400);
    if ( Utilities.isEmpty(request.body.ipcamara)) return response.send(error_400);
    Camara.find({name: request.body.name}).exec(function (err, camaras) {
        if (err) return response.send(error);
        if (!Utilities.isEmpty(camaras)) return response.send(error);
        var server = "rtmp://" + request.body.server + request.body.name;
        var camaranueva = new Camara({ server: server, name: request.body.name, ip: request.body.ipcamara});
        camaranueva.save();
        response.send(ok);
    });
};

/* Cámara online */
exports.putonline = function (request, response) {
    if (Utilities.isEmpty(request.params.name)) return response.send(error_400);
    Camara.find({name: request.params.name}).exec(function (err, camara) {
        if (err) response.send(error_400);
        if (Utilities.isEmpty(camara)) return response.send(error_400);
        camara[0].online = true;
        camara[0].ip = request.body.ipcamara; // Actualizar la ip de la cámara.
        camara[0].server = "rtmp://" + request.body.server + request.body.name; // Actualizar ip del servidor.
        camara[0].save();
        response.send(ok);
    });
};

/* Cámara offline  */
exports.putoffline = function (request, response) {
    if (Utilities.isEmpty(request.params.name)) return response.send(error_400);
    Camara.find({name: request.params.name}).exec(function (err, camara) {
        if (err) response.send(error_400);
        if (Utilities.isEmpty(camara)) return response.send(error_400);
        camara[0].online = false;
        camara[0].save();
        response.send(ok);
    });
};
```

Figura A.5. Código de las peticiones de la aplicación web cliente

- Código de la recepción de streaming

```
<div class="row">
    <div class="col m3 s3">
        <div class="collection" ng-repeat="camara in camaras">
            <a ng-click="scrollTo(camara._id)" href="" class="collection-item">{{camara.name}}</a>
        </div>
    </div>
    <div class="col m9 s9">
        <div class="container">
            <div class="row">
                <div ng-repeat="camara in camaras" class="repeat-item animated">
                    <link href="lib/videojs/video-js.css" rel="stylesheet">
                    <link href="css/customvideo.css" rel="stylesheet">
                    <!-- If you'd like to support IE8 -->
                    <script src="lib/videojs/videojs-ie8.min.js"></script>
                    <script src="lib/videojs/video.js"></script>
                    <div class="card v-camara-todo" ng-attr-id="{{'camara-'+camara._id}}">
                        <div class="card waves-block waves-light z-depth-3 hoverable">
                            <div class="card-content v-camara-video">
                                <div class="row">
                                    <video  ng-attr-id="{{'video-'+camara._id}}" class="video-js vjs-default-skin responsive-video"
                                           autoplay="true" preload="none" width="440" height="248" data-setup='{}'>
                                        <source ng-src="{{camara.server}}" type='rtmp/mp4'>
                                        <p class="vjs-no-js"> To view this video please enable JavaScript </p>
                                    </video>
                                </div>
                            </div>
```

Figura A.6. Código del streaming de la aplicación web cliente

- Código del control de robot

```
<div class="v-robot">
    <form action="/robot" method="post">
        <button name="movimiento" value="rleft" id="rleft" class="btn-robot"><i class="fa fa-undo fa-2x" aria-hidden="true"></i></button>
        <button name="movimiento" value="up" id="up" class="btn-robot"><i class="fa fa-arrow-up fa-2x"></i></button>
        <button name="movimiento" value="rright" id="rright" class="btn-robot"><i class="fa fa-repeat fa-2x" aria-hidden="true"></i></button>
        <div class="vacio"></div>
        <button name="movimiento" value="left" id="left" class="btn-robot"><i class="fa fa-arrow-left fa-2x"></i></button>
        <button name="movimiento" value="down" id="down" class="btn-robot"><i class="fa fa-arrow-down fa-2x"></i></button>
        <button name="movimiento" value="right" id="right" class="btn-robot"><i class="fa fa-arrow-right fa-2x"></i></button>
        <div class="vacio"></div>
        <label for="vel-{{'video-'+camara._id}}">Velocidad</label>
        <input name="vel" id="vel-{{'video-'+camara._id}}" class="v-vel" type="range" min="10" max="100" value="40" step="10">
        <label for="tiempo-{{'video-'+camara._id}}">Tiempo</label>
        <input name="tiempo" id="tiempo-{{'video-'+camara._id}}" class="v-tiempo" type="range" min="1" max="10" value="3" step="1">
        <input type="hidden" name="ip" value="{{camara.ip}}">
    </form>
</div>
```

Figura A.7. Código del control de robot de la aplicación web cliente

- Código del socket para el envío de comandos al robot

```
exports.enviarComando = function (request, response) {
    switch(request.body.movimiento){
        case 'rleft':
            var net = require('net');
            var client = net.connect(1234, request.body.ip);
            client.on('error', function(e){
                console.log(e);
            });
            client.write('legoev3rotarizquierda-' + request.body.vel + '-' + request.body.tiempo + '\n');
            client.end();
            break;
        case 'up':
            var net = require('net');
            var client = net.connect(1234, request.body.ip);
            client.on('error', function(e){
                console.log(e);
            });
            client.write('legoev3arriba-' + request.body.vel + '-' + request.body.tiempo + '\n');
            client.end();
            break;
        case 'rright':
            var net = require('net');
            var client = net.connect(1234, request.body.ip);
            client.on('error', function(e){
                console.log(e);
            });
            client.write('legoev3rotarderecha-' + request.body.vel + '-' + request.body.tiempo + '\n');
            client.end();
            break;
        case 'left':
            var net = require('net');
            var client = net.connect(1234, request.body.ip);
            client.on('error', function(e){
                console.log(e);
            });
            client.write('legoev3izquierda-' + request.body.vel + '-' + request.body.tiempo + '\n');
            client.end();
            break;
        case 'down':
            var net = require('net');
            var client = net.connect(1234, request.body.ip);
            client.on('error', function(e){
                console.log(e);
            });
            client.write('legoev3abajo-' + request.body.vel + '-' + request.body.tiempo + '\n');
            client.end();
            break;
```

Figura A.8. Código del socket del robot de la aplicación web cliente

Apéndice B.
Conference paper

Este Trabajo de Fin de Grado ha sido enviado al congreso UCAmI (10^{th} International Conference on Ubiquitous Computing & Ambient Intelligence) en forma de short paper. A continuación se presenta dicho documento.

Video Surveillance Application for Android Smartphones

J. Suárez Armas[1], P. Caballero Gil[2], C. Caballero Gil[3]

[1] Departament of Computer Engineering, University of La Laguna
La Laguna. Tenerife. Spain
alu0100600674@ull.edu.es

[2] Departament of Computer Engineering, University of La Laguna
La Laguna. Tenerife. Spain
pcaballe@ull.es

[3] Departament of Computer Engineering, University of La Laguna
La Laguna. Tenerife. Spain
ccabgil@ull.es

Abstract. This paper describes the development of a secure video surveillance application for Android devices. The proposal is based on a smartphone placed on a Lego Mindstorms EV3 robot and connected via Bluetooth to it. The smartphone runs in background a motion detection application with its camera. If motion is detected, it begins to capture images that are sent to a streaming server, in order to see them live from a web application. Remotely via the web application and the smartphone connected via Bluetooth it is possible to move the robot in order to find out more information about what is happening in the monitored location.

1 Introduction

Theft of homes and businesses is a problem that requires urgent solution. Domestic burglary increased by 14% between 2007 and 2012 in Europe [1]. Among the EU countries, the highest increases in the number of domestic burglary cases occurred in Greece, Spain, Italy, Romania and Croatia.

Furthermore, Spain is the third EU country that suffers more shoplifting, and more than 310 home burglaries a day are recorded [2]. Therefore, more and more establishments and homes are installing alarms that automatically contact the police in case of intrusion. In addition, many of these alarms consist of video surveillance systems that allow view images in real time from a centralized monitor and/or remotely from another location via the Internet.

This work aims to upgrade to a higher-level current video surveillance systems, by adding the ability to move the video surveillance cameras through the guarded stay. In order to take advantage of current smartphones, the proposal does not propose the use

of conventional video surveillance cameras, but Android smartphones, because they are much more than simple cameras.

The next section describes in more detail the main ideas of the proposal, and its main functions, which are video streaming, motion detection and robot control. Then, Section 3 sketches the used architecture and technologies and Section 4 mentions the communication between the two applications. The security mechanisms used are explained in Section 5. Finally, a brief conclusion closes the paper.

2 Description of the proposal

The most basic function of the proposal is to send a video captured in real time to a streaming server in order to display images on another device through a web application.

The proposal also takes advantage of using a smartphone to detect motion so that if motion is detected, it sends a notification and a video recorded in real time via streaming.

In addition, the smartphone is connected via Bluetooth with a robot Lego Mindstorms EV3, so that the user can move the smartphone through the room as he/she wishes. To this end, at the web application, while the images are captured in real time, it is possible to control the robot.

Given the high degree of sensitivity of sent information, the security of the connection is protected with encryption and authentication methods.

The three main functions of the proposal are:

1. Sending video and audio streaming: The main function of this proposal is to captured audio and image by a mobile device and send them to a streaming server, so that with the client application it is possible to display this video in real time and perform other functions. See Fig. 1 and Fig. 2.

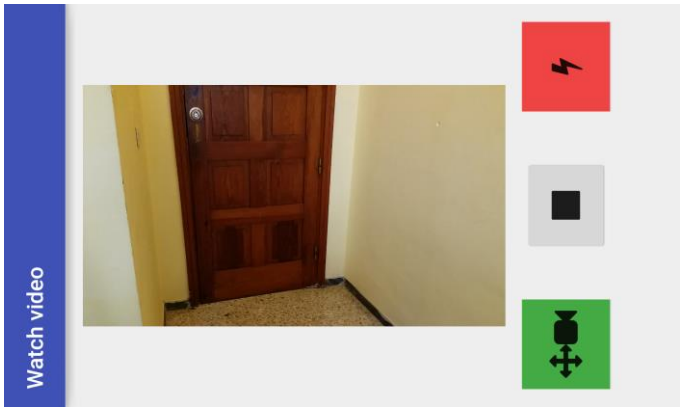


Figure 1. Captured audio and image by a mobile device

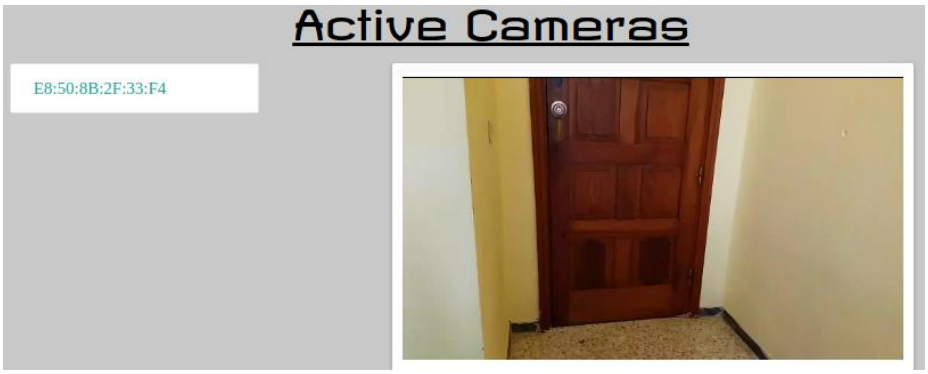


Figure 2. Video received at a streaming server

2. Motion detection: The application, while is capturing images around, is constantly checking if there is movement in front of the camera. If so, send a text message to recipients that have been added to the notification list. See Fig. 3.

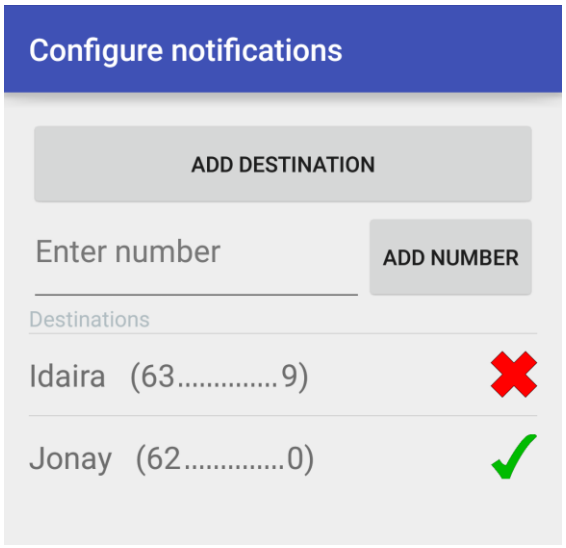


Figure 3. Notification list

3. Robot control: Through the client application, while the images captured by the main application are being displayed, it is possible to move the robot in order to analyze the different areas of the monitored site. See Fig. 4.

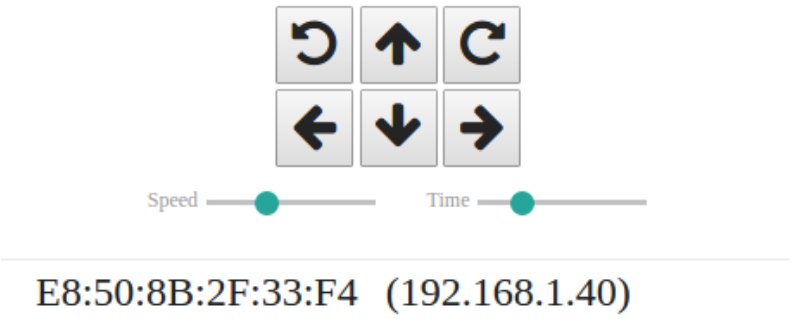


Figure 4. Robot control

3 Architecture

The complete system comprises the elements shown in Fig. 5.

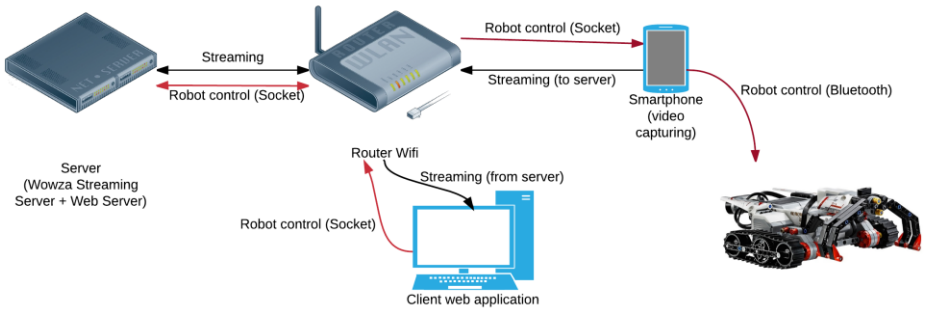


Figure 5. Architecture

- Wowza streaming server: It is the streaming server that stores and sends the images.
- WLAN router: It is in charge of making the interconnection between the smartphone, the streaming server (cloud), and the computer running the client application.
- Client application: It receives the images from the streaming server, and allows sending commands to the smartphone so that it can send them to the robot via Bluetooth.
- Smartphone: It is one of the two essential elements of the proposal because it runs the main application, which sends the video recording to the streaming server, and allows the robot to move according to commands received from the client application.

- Lego Mindstorms EV3 robot: It is the other essential element of the proposal because it carries the smartphone with the main application connected via Bluetooth, so that it moves according to the signals it receives from the smartphone.

The main technologies used by the system are:

- Main application: The main application, which runs the smartphone, is developed on Android. It connects to the network via WiFi or 4G (LTE) and also connects to the Lego Mindstorms robot via Bluetooth.
- Client application: The client application is a web application, allowing run on any computer without the need to pre-install it. To run it requires only a browser. For the development of this client application it has been made use of HTML5, CSS, JavaScript and NodeJS.

4 Communication between the two applications

Communication between the two applications, for robot control, is established through sockets. See Fig. 6.

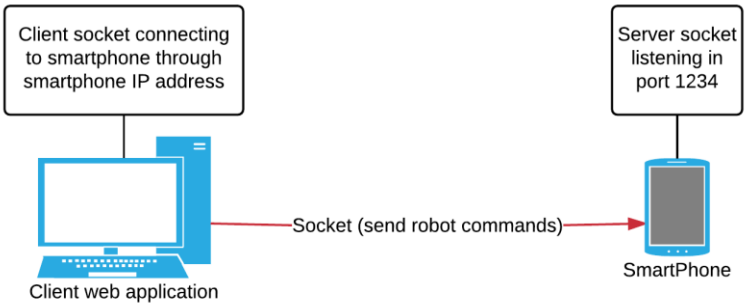


Figure 6. Communication between mobile and web application

- The principal Android application has a server socket that is constantly listening on port 1234 to receive robot commands. Depending of the received command, the robot moves in a direction with a certain speed during a determined time.
- The client web application connect to the mobile application through a client socket thanks to the IP address of the smartphone. Once connected, sends the corresponding command to move the robot and close the connection.

The communication for robot control is protected with security mechanisms explained in next section.

5 Security

The security mechanisms used to protect information are:

- 256-bit AES encryption in CBC (Cipher Block Chaining) mode to encrypt information sent between the Android application and client web application.
- Diffie Hellman Elliptic Curve for secret key agreement.
- Digital Signature Elliptic Curve for verification of the authenticity and integrity of information.

6 Conclusions

The proposal here described, thanks simply to an Android smartphone and a Lego Mindstorms EV3 robot connected via Bluetooth, allows enjoying a cheap system superior to the traditional video surveillance systems, thanks to the innovative ability of remotely browsing the monitored stay. Therefore, it is expected that soon this proposal would have received in the world of security establishments and homes. A beta version of this application is currently being developed.

7 Acknowledgments

Research supported by projects RTC-2014-1648-8 and TEC2014-54110-R.

References

[1] Eurostats Statistics Explained. Crime statistics. http://ec.europa.eu/eurostat/statistics-explained/index.php/Crime_statistics#Robbery

[2] La Nueva España. En España se registran más de 310 robos en casas al día. http://www.lne.es/sociedad/2016/02/10/espana-registran-310-robos-viviendas/1881035.html

[3] Lego. Lego MindStorms EV3. http://www.lego.com/es-es/mindstorms/about-ev3

[4] Wowza. Wowza Streaming Engine. https://www.wowza.com/products/streaming-engine