Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Comunicaciones inalámbricas seguras para UAV (RPA) con dispositivos Android

Secure wireless communications for UAV (RPA) with Android devices

Jaime Madico Cañete

La Laguna, 4 de septiembre de 2018

Dña. MARÍA CANDELARIA HERNÁNDEZ GOYA , con N.I.F. 45.441.714-Q profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

D. JOSÉ IVÁN SANTOS GONZÁLEZ, con N.I.F. 78.637.989-T Investigador FPI 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:

Comunicaciones inalámbricas seguras para UAV (RPA) con dispositivos Android.

ha sido realizada bajo su dirección por **D. Jaime Madico Cañete**, con N.I.F. 42.238.205-Q.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de septiembre de 2018.

Agradecimientos

A mi tutora Dña. María Candelaria Hernández Goya
y a mi cotutor D. José Iván Santos González,
que desde el inicio del proyecto han demostrado real interés y voluntad en guiarme y aconsejarme para la realización satisfactoria del mismo.
Además quería agradecer el apoyo y colaboración de
D. Ricardo José Aguasca Colomo (SIANI, ULPGC).

Licencia

Resumen

Actualmente el uso de drones y robots está cada vez más extendido como tecnología de apoyo para situaciones de emergencias. Por una parte, los drones civiles o RPAS (Remotelly Piloted Aircraft Systems) han ganado protagonismo en los últimos años en gran parte debido a los avances técnicos que los han hecho cada vez más fiables, seguros, económicos y versátiles. Así el uso de RPAS es ya conocido para vigilancia y control medioambiental, para inspección de instalaciones, y en particular para detectar personas y animales a gran distancia, lo que resulta fundamental en situaciones de emergencias. Por otra parte, desde hace tiempo los robots constituyen una herramienta muy útil para ayudar en situaciones de emergencia o desastres naturales en donde las condiciones pueden ser peligrosas para equipos humanos de rescate.

Este TFG tiene por objetivo integrar la tecnología de dispositivos móviles 4G en la estructura de un RPAS tipo multicóptero de corto alcance, dotándolo de un software que le permita realizar misiones en situaciones de emergencias de diversa índole. Para ello se pretende desarrollar una aplicación móvil que permita la comunicación entre un terminal sujeto al drone con otro situado en el Puesto de Mando Avanzado, de manera que se puedan monitorizar situaciones de riesgo, y obtener información valiosa, desde el aire.

Palabras clave: dron, situación, emergencia, RPA, rescate, smatphone, 4G...

Summary

Nowadays, the use of drones and robots is considerably widespread as support technology for emergency situations. On the one hand, civilian drones or RPAS (Remotelly Piloted Aircraft Systems) have gained prominence in recent years due to the technical advances that have made them more reliable, safer, cheaper and versatile. The use of RPAS is already a fact in environmental monitoring and control, inspection of facilities, and in particular to detect people and animals at a great distance, which is essential in emergency situations. On the other hand, robots have been a very useful tool to help in emergency situations or natural disasters where conditions can be dangerous for human rescue teams.

This TFG aims to integrate 4G wireless technology into the structure of a short range multicopter RPAS, providing software that allows to perform missions in various kinds of emergency situations. For this, it is intended to develop a mobile application that allows communication between a terminal associated to the drone with another located in the Advanced Command Post, so that risk situations can be monitored, and valuable information from the air may be obtained.

Keywords: drone, robot, situation, emergency, RPA, human, rescue, CASUS, mobile, 4G ...

Índice General

Índice de figuras

Índice de tablas

Capítulo 1.
Introducción.

1.1 Motivación

Las situaciones de emergencia tienen un gran impacto tanto material como personal, llegando en muchas ocasiones a provocar la pérdida de vidas humanas y altos gastos económicos.
Como primer objetivo, ante cualquier catástrofe, debemos plantear si ésta podría evitarse o, al menos predecirse para poder reducir, en la medida de lo posible, el impacto en la población y daños materiales. En este sentido existen muchos estudios y proyectos que buscan aprovechar los avances tecnológicos y las TIC como apoyo preventivo ante estas situaciones.

Sin embargo, en casos en los que estas catástrofes generan situaciones de peligro para la vida humana se debe proceder a actuar movilizando a los cuerpos de rescate correspondientes a la zona de peligro. Para la correcta labor de estos grupos es imprescindible contar con una cantidad de información suficiente y efectiva.

Este Trabajo de Fin de Grado nace con el objetivo de proporcionar información útil de forma rápida, precisa y concisa a los equipos de rescate y de protección, permitiendo así que los agentes puedan actuar de una manera ágil en situaciones de emergencia que sucedan en localizaciones en las que el medio oponga mayor dificultad para la actuación humana.

Además se ha intentado aprovechar el avance de la tecnología móvil actual, que permite tener sistemas ligeros y de tamaño reducido a pesar de constar con una elevada velocidad de cómputo y ser capaces de mantener comunicaciones inalámbricas globales y rápidas, gracias a las últimas redes de telefonía móvil. Lejos de ser dispositivos sencillos constan de cámaras de calidad suficiente como también de numerosos sensores como pueden ser magnetómetros, acelerómetros y localizadores GPS, que los convierten en dispositivos ideales como infraestructura de soporte polivalente donde además premia un aprovechamiento óptimo del espacio.

Por otro lado, cada vez está más extendido el uso de drones y robots, como tecnología de apoyo para situaciones de emergencias. Por una parte, los drones civiles o RPAS (Remotelly Piloted Aircraft Systems) han ganado protagonismo en los últimos años en gran parte debido a los avances técnicos que los han hecho cada vez más fiables, seguros, económicos y versátiles. Así el uso de RPAS es ya conocido para vigilancia y control medioambiental, para inspección de instalaciones, y en particular para detectar personas y animales a gran distancia, lo que resulta fundamental en situaciones de emergencias.

Este Trabajo de Fin de Grado está enmarcado en una de las líneas de trabajo del Proyecto de Investigación CASUS - Cooperación Móvil Segura Aplicada a Situaciones de Emergencia e ICT liderado por el grupo de investigación CryptULL.

1.2 Estructura de la memoria

Este documento se encuentra estructurado en siete capítulos, compuestos por diferentes apartados como podemos ver en el índice. Comienza realizando una introducción a la justificación del proyecto. Se trata de definir el problema que se pretende resolver y el marco de trabajo en el que está comprendido el proyecto. Además se incluye un breve estado del arte en este ámbito y se establece una primera conceptualización de cuál es el objetivo principal del proyecto.

Las cuestiones respecto a las características del proyecto, qué objetivos en concreto se quieren abordar y la organización en fases tanto inicial como final, se tratan de manera más detallada en segundo capítulo. En el tercero se definen las tecnologías utilizadas, además de sus características, funcionalidades y los motivos de selección. Se continúa el capítulo cuarto detallando las características de la aplicación móvil conseguidas, detalles sobre su elaboración y la funcionalidad final conseguida. En el quinto se estima un presupuesto, personal, hardware, software y de manera global del desarrollo del proyecto. Por último los capítulos 6 y 7 exponen las conclusiones tanto en español como en inglés respectivamente. También se

incluyen futuras implementaciones interesantes a desarrollar. Tras esto se incluye la bibliografía con enlaces y recursos de referencia que han servido de apoyo durante el proceso de desarrollo.

1.3 Estado del arte

Entre los distintos objetivos a lograr para el desarrollo de la aplicación móvil destaca la emisión de streaming de vídeo cifrado, desde un dispositivo android acoplado al drone, aunque este tipo de funcionalidad se encuentra implementada en la actualidad, existen determinados factores adversos que en nuestro caso pueden ser muy determinantes. Debido al medio y al estado actual de la tecnología es inevitable que exista una pequeña latencia, sobre todo si los fotogramas son cifrados antes de emitirse, de esta manera hay que tener un especial cuidado en este sentido. Existen herramientas en la actualidad que nos permiten transmitir flujo de vídeo cifrados, algunos ejemplos podrían ser:

- El cifrado **HTTP Live Streaming (HLS)** que permite enviar vídeo codificado a través de HTTP para su reproducción en dispositivos móviles y de escritorio, apoyándonos para ello en el estándar AES (“Advanced Encryption Standard”), sin que se produzca ninguna diferencia detectable a la hora de reproducir el vídeo.

  El protocolo HLS divide el vídeo en varios archivos descargables menores y luego transmite secuencialmente estos segmentos a través de HTTP. De esta manera protege el contenido añadiendo funciones de AES a nuestra solución estándar de HLS. Un ejemplo sería la herramienta Video Cloud HLS, de la marca Apple Apple Inc, que cifra cada uno de los pequeños segmentos del vídeo y entrega de forma segura los archivos que gestionan la selección de variantes de representación.

- **WebRTC (Web Real-Time Communication)** se trata de una API que está siendo elaborada por la World Wide Web Consortium (W3C), con el objetivo de lograr la comunicación en tiempo real basada en navegador, punto a punto y sin ningún complemento adicional.

Además de esto a través de los componentes de la API sería posible acceder a la cámara y el micrófono del dispositivo e incluso establecer llamadas de audio-vídeo.

Para que WebRTC transfiera datos en tiempo real y de forma segura, los datos primero se cifran utilizando el método **DTLS (Datagram Transport Layer Security)**, de manera que sería posible evitar con esto el espionaje y la manipulación de información.

Además WebRTC también se encarga de encriptar los datos de vídeo y audio apoyándose en el protocolo **SRTP (Secure Real-Time Protocol)**, de esta manera podríamos asegurarnos de que el audio y vídeo no sea capturado y observado por terceros.

- El protocolo de transmisión en tiempo real **(RTSP, Real Time Streaming Protocol)** establece y controla uno o muchos flujos sincronizados de datos, ya sean de audio o de video. De esta manera se trata de un protocolo no orientado a conexión sino que el servidor mantiene una sesión asociada a un identificador, normalmente usando TCP para datos de control del reproductor y UDP para los datos de audio y vídeo. De forma intencionada, el protocolo es similar en sintaxis y operación a HTTP para que los mecanismos de expansión añadidos a HTTP se puedan añadir también RTSP.

  RTSP reutiliza mecanismos de seguridad web ya sea a los protocolos de transporte **(TLS)** o dentro del mismo protocolo. Todas las formas de autentificación HTTP ya sea básica o basada en resumen son directamente aplicables.

  Se trata además del protocolo implementado en el proyecto, ya que, a pesar de otorgar más latencia que otros protocolos, es de fácil implementación y no tiene un tratamiento tan complejo. Además existen varias librerías, como la usada para este caso (**libstreaming)**, que implementan este protocolo permitiendo incluir esta funcionalidad en aplicaciones android.

Respecto al uso de drones en situaciones de emergencia podemos encontrar cada vez más casos en los que su utilización supone un avance significativo en cuestiones de seguridad.
Por ejemplo, para el desplazamiento de tecnologías de salvamento compactas (AED, medicamentos, ayudas para la Resucitación Cardiopulmonar, etc.) en situaciones en las que sea necesario actuar con gran velocidad y/o acceder a lugares elevados de difícil acceso. De esta manera **Ambulance Drone** fue diseñado para el auxilio en este tipo de situaciones.

En el caso de este proyecto, se parte de un contexto donde el dron no tendrá la función del desplazamiento de materiales, pero tendrá otra de vital importancia, ya que debe permitir realizar un reconocimiento de la zona de emergencia e identificar posibles víctimas y/o afectados, además de otras posiciones de interés de forma precisa. Para determinar las posiciones será indispensable acceder a la telemetría del dispositivo en aire, aunque para esto el UAV únicamente funcionará como soporte físico, ya que todos los sistemas para la monitorización vienen otorgados por el terminal móvil.

También en referencia al uso de conexiones de red 4G en drones existen implementaciones que buscan poder hacer volar al UAV de manera que sea capaz de alejarse más que su capacidad por defecto. Sin embargo, para este proyecto la comunicación 4G será de un dispositivo android a otro, de manera que "no habrá interacción" con el cuadracóptero. Algunos ejemplos de otras aplicaciones actuales, que interactúan (ya sea en menor o mayor medida) con drones podrían ser ***UAV Forecast*** o ***Airmap***. La primera se centra sobretodo en cuestiones referidas al clima, con el objeto de conocer y decidir si es seguro volar. La segunda en cambio es una aplicación referente como herramienta de facto para vuelos comerciales en el negocio de pilotaje de drones.

Para el desarrollo de la aplicaciones móviles Android, el entorno de desarrollo estándar fue inicialmente **Eclipse**, que consiste en una plataforma de software compuesto por diferentes herramientas de programación de código abierto multiplataforma. Esta es usada habitualmente para el desarrollo de ***IDEs*** (Entornos de Desarrollo

integrados), como el IDE de Java llamado Java Development Toolkit (**JDT**) y el compilador (ECJ) que se entrega como parte de Eclipse. Entre los lenguajes soportados más utilizados para el desarrollo en esta plataforma se encuentran: ***Java***, ***ANSI C***, y ***C++***.

Sin embargo, desde 2014 el estándar y referente comenzó a ser el entorno de desarrollo **Android Studio**, herramienta principal para el desarrollo de la aplicación móvil que funciona como piedra angular de este proyecto. Este entorno facilita de forma cómoda el modelado de la APP separando la lógica e interfaz de la aplicación, de esta manera se ha hecho uso de Java como lenguaje de programación y del lenguaje de etiquetas XML para las cuestiones gráficas de la APP.
Este entorno de desarrollo fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, reemplazando a Eclipse como IDE estándar para el desarrollo de aplicaciones móviles para Android. En 2014 se publicó la primera versión estable y desde entonces ha recibido numerosas actualizaciones y mejoras, la última versión es la 3.1, lanzada en agosto de 2018. A diferencia de eclipse, ha sido diseñado específicamente para el desarrollo de Android, en concreto con soporte en los lenguajes de programación ***Java***, ***C++***, ***Kotlin, Javascript, HTML5, CSS y XML***.

Otras de las características más relevantes de Android Studio son:

- Capacidad de renderización en tiempo real para el desarrollo de interfaces.
- Posibilidad de uso de la Consola de Desarrollador. Ésta ofrece: consejos de optimización, ayuda para la traducción y estadísticas de uso entre otros.
- Hace uso de construcción basada en Gradle (sistema de automatización de construcción de código abierto).
- Un potente editor de diseño enriquecido, podemos observar un ejemplo de la interfaz de éste en la *Figura 1: Interfaz de Android Studio 3.0*.
- Disponibilidad de plantillas para la creación de diseños comunes de Android.
- Un dispositivo virtual de Android de gran ayuda y potencia, ya que emula un terminal móvil que permite ejecutar y probar aplicaciones.

Podemos ver un ejemplo de la interfaz del mismo en la *Figura 2: Ejemplo de dispositivo Android Virtual.*

- Funciones de firma de aplicaciones e integración con ProGuard.
- Soporte para la programación de aplicaciones Android Wear.
- Herramientas Lint que permiten detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, etc.
- Soporte integrado para Google Cloud Platform, que permite la integración con Google Cloud Messaging y App Engine.

Las características técnicas y requisitos aparecen detallados en el apartado *2.2.2 Android Studio*.

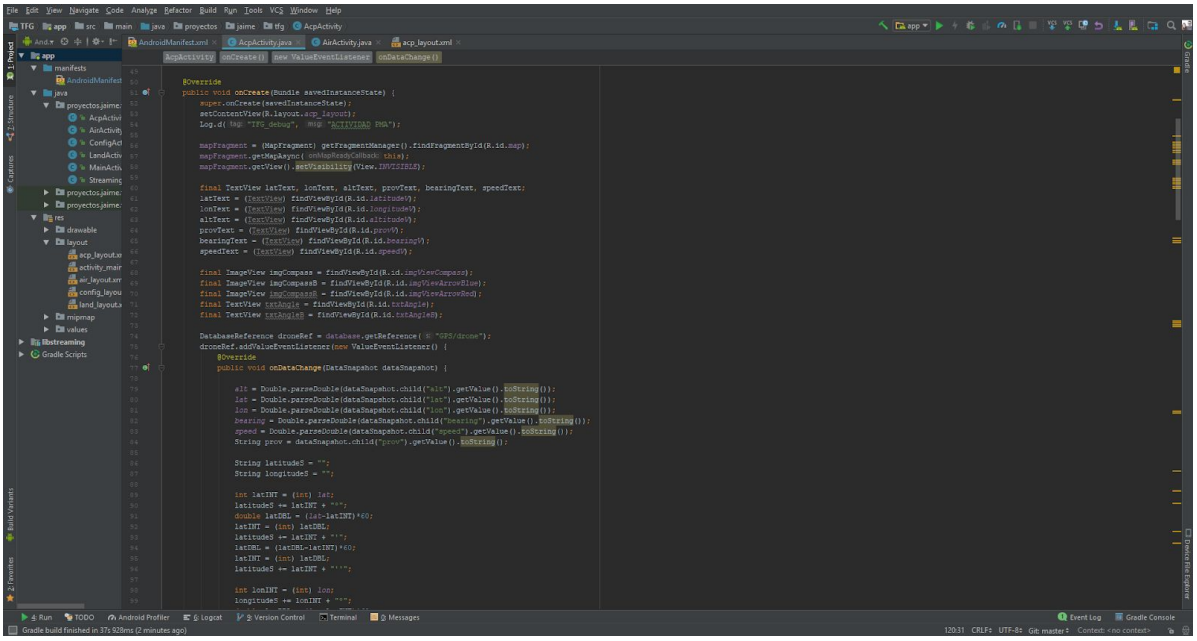

Figura 1: Interfaz de Android Studio 3.0

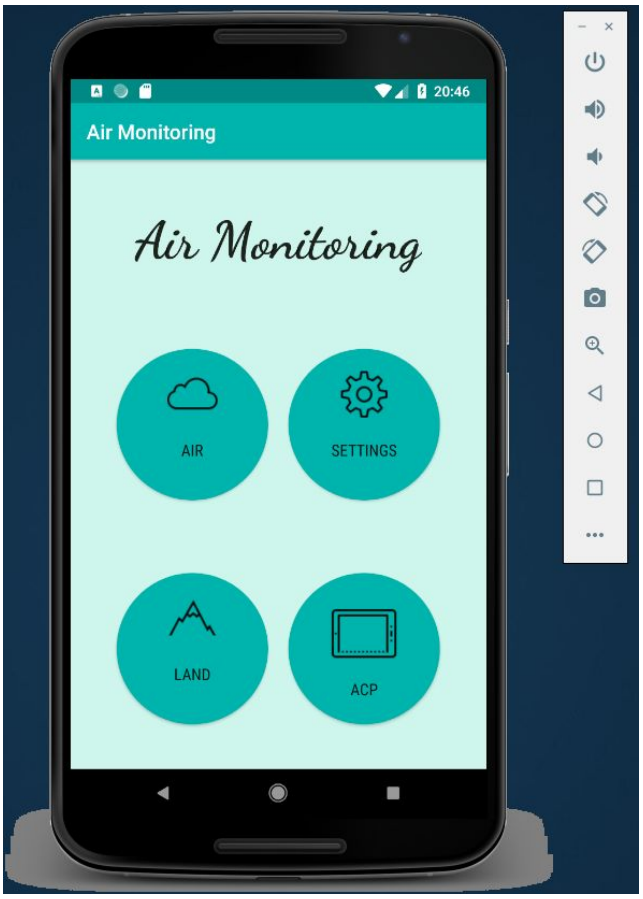

Figura 2: Ejemplo de dispositivo Android Virtual

1.4 Conceptualización de la aplicación

Como se ha comentado anteriormente, el objetivo principal del proyecto es el desarrollo una aplicación móvil sobre el sistema operativo android que en principio estará dotada 2 perfiles de funcionamiento principales:

- **Modo emisor**. En este perfil, un dispositivo android acoplado al **drone** capta a través de la cámara del smartphone flujo de vídeo. Este flujo se transmite en tiempo real (asumiendo una pequeña latencia) a otro dispositivo. Tanto este flujo de vídeo como otra información extraída de los sensores del smartphone sería protegida mediante cifrado y transmitida a otro dispositivo android a través de la red 4G. Algunos ejemplos de esta información extraída de sensores serían: localización GPS, inclinación, altitud, y más información que puede ser de interés para los grupos correspondientes como receptores.

En la *Figura 2: Ejemplo de dispositivo Android Virtual* podemos observar el botón que conduce a este modo de ejecución dentro de la aplicación, situado arriba a la izquierda.

- **Modo receptor**. Este perfil estaría en ejecución en un segundo dispositivo android distinto al primero, el cual se encontraría situado en el **PMA (Puesto de Mando Avanzado)**, autoridad encargada de analizar y tratar los datos obtenidos del smartphone que se encuentre en vuelo y de forma consecuente indicar órdenes y actuaciones a las distintas unidades de tierra.
  En este rol se recibirá el streaming de vídeo e información de los sensores. En la pantalla del dispositivo se mostrará el flujo de imagen cenital que se está capturando desde el emisor junto con la información extraída de los sensores. Mediante superposición y jugando con diferentes menús podremos interactuar con la aplicación para poder acceder a toda la información de interés, como por ejemplo intercambiar de vista de vídeo a Google Maps donde además se mostrará información de interés de las distintas localizaciones pertinentes.

  En la *Figura 2: Ejemplo de dispositivo Android Virtual* podemos observar el botón que conduce a este modo de ejecución dentro de la aplicación, situado abajo a la izquierda.

  Además, en la *Figura 3: Esquema de modos principales de funcionamiento* se muestra un gráfico con una representación de ambos modos principales (emisor y receptor), que identificarán en la aplicación como *"Air"* y *"ACP"*.

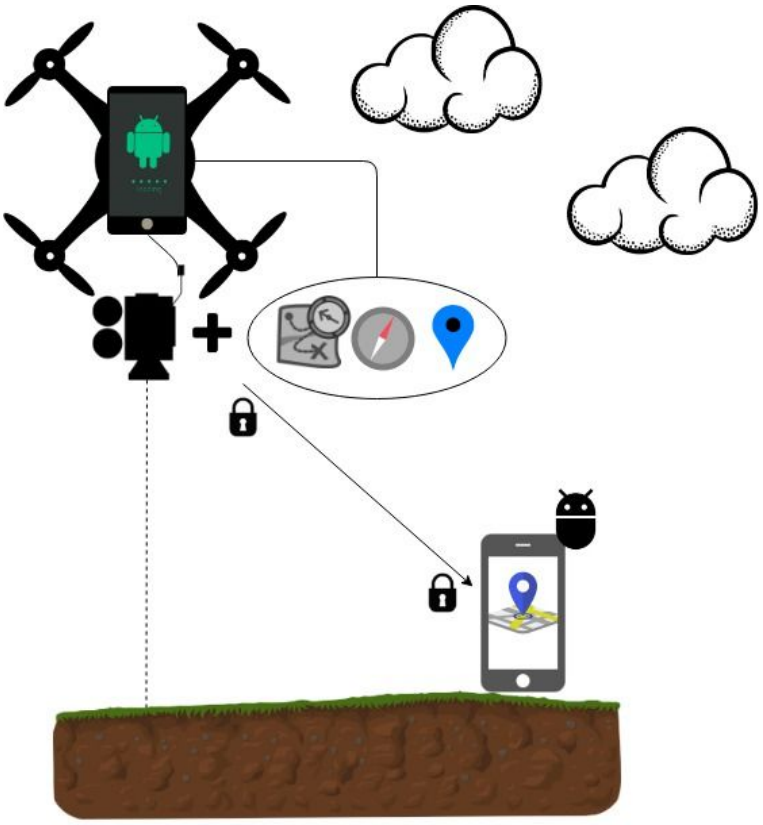

Figura 3: Esquema de modos principales de funcionamiento

Además existe otro perfil para la aplicación se comportaría de manera similar al de emisión:

- **Modo emisión terrestre.** Este perfil actuaría como una versión acotada del modo emisor, pero realizará menos cálculos. Únicamente transmitirán al PMA su ubicación GPS y flujo de vídeo, estos se tratarán por el dispositivo funcionando como receptor de la manera correspondiente. Además el perfil de tierra podrá recibir notificaciones y alarmas generadas por el PMA.

  En la *Figura 2: Ejemplo de dispositivo Android Virtual* podemos observar el botón que conduce a este modo de ejecución dentro de la aplicación, situado abajo a la derecha.

1.5 Objetivos

Como se ha comentado, el objetivo del proyecto es el desarrollo completo de un sistema que pueda dar servicio a las autoridades en situaciones de emergencias. Para ello el núcleo del sistema, la aplicación móvil, se ejecutará sobre el S.O. android comunicando distintos dispositivos: un dispositivo en aire anclado a un dron, un dispositivo terrestre situado en el Puesto de Mando Avanzado (PMA) y varios dispositivos para el personal localizado en terreno. Sin embargo, para simplificar las pruebas únicamente se ha hecho uso de un dispositivo por rol. Para lograr esto, se definieron las siguientes actividades como requisitos para el eficiente funcionamiento del sistema:

1. Recolección de datos:
    a. Análisis e implementación de streaming de video descentralizado entre dispositivos móviles.
    b. Obtener datos de sensores del dispositivo móvil.
    c. Obtener datos del magnetómetro del dispositivo móvil.
    d. Obtener datos del GPS del dispositivo móvil.

2. Funciones de transmisión:
    a. Transmisión de imágenes y datos (telemetría) obtenidos por el dispositivo móvil.
    b. Análisis de métodos de cifrado de vídeo e implementación.
    c. Compartir información de posicionamiento.

3. Funciones de recepción:
    a. Diseño de la interfaz de recepción de datos.
    b. Superposición de capas de Google Maps.
    c. Gestión de alarmas y notificaciones.

Capítulo 2.
Requisitos y características del proyecto.

2.1 Elementos del proyecto

Para la puesta en marcha de todo el sistema se precisa de diferentes dispositivos y herramientas.

- **Drone** cuadracóptero.
- **Dispositivo móvil anclado al drone**, encargado de la monitorización aérea haciendo uso de la aplicación móvil.
- **Tableta móvil** encargada de funcionar como centro de control en el **Puesto de Mando Avanzado**.
- **Dispositivo móvil** para emitir vídeo desde la **unidad de tierra**.
- **Servidor** que aloje el servicio **RTSP**.
- **Computadora** para el desarrollo de la aplicación a través del entorno Android Studio.

Además se hace uso de los siguientes servicios:

- Entorno de desarrollo Android Studio para el desarrollo de la aplicación móvil.
- Servicio Wowza Streaming Engine para la transmisión de vídeo mediante el protocolo RTSP.
- Servicio de comunicación de aplicaciones a través de internet, Firebase.
- Servicio de control de versiones en nube, Github.

2.2 Requisitos Hardware

En este apartado se detallan las características técnicas de los diferentes dispositivos.

2.2.1 Servidor RTSP

Se precisa de un equipo que aloje el servicio de streaming de vídeo, Wowza Streaming Engine. Para el desarrollo se ha hecho uso del equipo personal del alumno. Las características de la computadora son las siguientes.

Procesador	Intel(R) Core(TM) i5-4430 CPU @ 3.00GHz (4 CPUs), ~3.0GHz
Memoria RAM	16GB
Disco duro	1 TB HDD + 225 GB SSD
Tarjeta Gráfica	NVIDIA GeForce GTX 1070
Sistema Operativo	Windows 10 Pro 64-bit

Tabla 1: Características PC

En la página oficial del proveedor podemos encontrar los requisitos para el correcto funcionamiento del servicio.

CPU	Procesador de 4 núcleos ~ 3.00 GHz o superior
Memoria RAM	4GB
Disco duro	2 o más en RAID 0
Ancho de banda	1Gbps Ethernet

Tabla 2: Requisitos mínimos recomendados para WSE

CPU	Doble procesador de 4 núcleos ~ 3.00 GHz o superior
Memoria RAM	16 - 32GB
Disco duro	2 o más en RAID 0
Ancho de banda	10Gbps Ethernet

Tabla 3: Requisitos recomendados para rendimiento máximo

2.2.2 Android Studio

Para el desarrollo de la aplicación se hizo uso del IDE Android Studio, para alojar tal software se hizo uso del mismo PC que funciona como servidor RTSP. Las características del mismo se encuentran visibles en la *Tabla 2: Características PC*.

Sin embargo, los requisitos del sistema mínimos (última versión: 3.X) para el correcto funcionamiento del entorno, según la web oficial, serían los siguientes

	Windows	**OS X/macOS**	**Linux**
OS version	Windows 10/8/7 (32- o 64-bit)	Mac OS X 10.10 (Yosemite) o superior, hasta 10.13 (macOS High Sierra)	GNOME o KDE desktop
RAM	3 GB RAM mínimo, 8 GB RAM recomendado más 1GB adicional para el emulador de Android		
Espacio en disco	2 GB de espacio en disco para Android Studio, 4GB recomendados (500MB para la IDE y al menos 1.5 GB para Android SDK, imágenes de sistema de emulador y cachés)		
Java version	Java Development Kit (JDK) 8		
Resolución de pantalla	1280x800 mínimo, 1440x900 recomendado		

Tabla 4: Requisitos mínimos recomendados para Android Studio

Respecto a las características técnicas tiene soporte para las plataformas más extendidas: Windows 2003, Vista, 7, 8, y 10, tanto plataformas de 32 como de 64 bits, GNU/Linux, Linux con GNOME o KDE y 2 GB de memoria RAM mínimo y macOS, desde 10.8.5 en adelante.

2.2.3 Dispositivos móviles

En este apartado se concretan los recursos hardware de todos los dispositivos móviles utilizados:

- Samsung Galaxy S7 Edge.
- Sony Xperia Z3.
- BQ Aquarius X Pro.
- Samsung Galaxy Tab A.

Las características de estos terminales se encuentran especificadas en el anexo B.

2.2.4 Requisitos de la APP

Como recursos hardware mínimos para la correcta ejecución de la aplicación en la versión actual, se consideran las siguientes:

Procesador	IDual-core, 1200 MHz o superior
Memoria RAM	500 MB o superior
Almacenamiento	10 MB de espacio de almacenamiento
Conectividad	Red 3G (recomendable 4G)
Sensores	Acelerómetro, Magnetómetro y localizador GPS
Cámaras	Frontal: prescindible. Posterior: imprescindible.
Versión de Android	Android 4.4 o superior

Tabla 5: Requisitos Hardware APP

2.2 Requisitos Software

Para el correcto funcionamiento de las aplicaciones y herramientas se precisa de los siguientes requisitos software:

- Wowza Streaming Engine para proveer del servicio de vídeo cifrado RTSP. Para alojar este servicio se debe ejecutar sobre el sistema operativo Windows® (XP, Vista, 7, 8, 10; Server 2003, 2008, 2012, 2016), Linux (todas las distribuciones) o Mac® OS X 10.8 o posterior. Para este proyecto se ha utilizado Windows 10 Pro 64-bit. Además es necesario constar de Java Runtime Environment (JRE) o Java Development Kit (JDK) 1.8.0 o superior.

- Además para ejecutar la aplicación desarrollada se ha utilizado la versión 8.0.0 API 26 de Android (Oreo), sin embargo sería compatible con la versión 4.4 KitKat API 19 o superior.

- Para el desarrollo de la aplicación se ha usado el software Android Studio, en concreto su versión 3.0.1. Usando el lenguaje de programación Java y el lenguaje de etiquetas XML para implementar la APP. Además se ha hecho uso del control de versiones Git, con su versión en nube Github manteniendo el código en un repositorio; y de la herramienta Firebase para permitir el traspaso de información entre diferentes terminales que ejecuten la aplicación.

Capítulo 3.
Tecnologías.

El desarrollo y funcionamiento de la aplicación se ha apoyado en tecnologías, herramientas, hardware y software de diverso tipo, a continuación se definen las características de cada uno de ellos.

3.1 Terminal móvil: BQ Aquarius X Pro

Como se ha comentado, el objetivo principal del proyecto es el desarrollo de una aplicación móvil sobre el sistema operativo Android. De esta manera hemos seleccionado como smartphone principal, para realizar las pruebas el modelo Aquarius X Pro de la marca BQ. La selección de este smartphone viene dada por las características que se indican a continuación.

Como características fundamentales consta de los 3 sensores que se usan para la adquisición de la información: **magnetómetro**, **acelerómetro** y **GPS**. Los dos primeros son necesarios para conocer la orientación del dispositivo en los 3 ejes y que la *"brújula"* de la aplicación funcione de manera correcta, y el último nos permite conocer la localización exacta del dispositivo. Además, combinándolo con los otros sensores podemos calcular trayectorias y direcciones a puntos de interés. Además, el proyecto utiliza la comunicación inalámbrica a través de la red inalámbrica móvil 4G, a la que este dispositivo tiene conectividad.

Otras de las características que sin ser fundamentales son recomendables conseguir sería:

- Cámara posterior del terminal (partimos del supuesto de que todos los smartphones incorporan una) de calidad.
- Peso del dispositivo ligero para ser anclado al dron.
- Versión de android considerablemente reciente, para tener mayor compatibilidad con las API y librerías.
- USB tipo C.

Para más detalle se adjunta la ficha de características del terminal en la *Figura 4: Especificaciones BQ Aquarius X Pro*.

Además del terminal proporcionado para el desarrollo del proyecto se ha hecho uso de otros terminales móviles para realizar diferentes pruebas y simulaciones, las características de todos ellos quedan mencionadas en el capítulo 2.2.3

3.2 Dron cuadracóptero: DJI Phantom 4

El contexto del proyecto es el introducido por el proyecto CASUS, por ello se trata de un supuesto en que el terminal anteriormente comentado se encuentre anclado a un dron tipo cuadracóptero, que nos permita el soporte aéreo para obtener información del terreno de manera cómoda y rápida, sobretodo cuando este se encuentra en estado de difícil inspección o acceso a los agentes de tierra.

De esta manera, el dron habilitado para el anclaje del terminal BQ por parte del equipo para el proyecto CASUS al que este TFG pertenece, sería el dron cuadracóptero DJI PHANTOM 4. Algunas de las especificaciones del mismo son las visibles en la *Figura 8: Especificaciones DRON PHANTOM 4*, aunque podemos encontrar más información en la página oficial del fabricante:

3.3 Entorno de desarrollo: Android Studio

Android Studio (figura 9) es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones sobre la plataforma Android. Como comentamos existen otros entornos pero el seleccionado para el proyecto fue este debido a la potencia y comodidad con respecto a sus homólogos. Además se trata el IDE Oficial proporcionado por Android.

Las características de éste se encuentran mejor detalladas en los capítulos *1.3 Estado del arte* y *2.2.2 Android Studio*.

Figura 4: Logo Android Studio

3.4 Motor de Streaming: Wowza Streaming Engine

Como se mencionó en apartado 1.2, existen diferentes protocolos y herramientas para la transmisión de vídeo cifrado. Para este proyecto se ha utilizado la herramienta Wowza Streaming Engine (figura 10), que nos va a permitir la transmisión de flujo de vídeo cifrado gracias al protocolo RTSP.

Algunas de las características que han motivado el uso de esta herramienta son:

- Facilidad de instalación y uso, además de posible escalabilidad.
- Configuración y adaptación no complejas.
- Flujo de vídeo protegido mediante cifrado.
- Transmisión de vídeo bajo demanda a cualquier dispositivo.
- Consta con la opción de instalación en nube o en servidores externos.
- Control de interfaz de usuario (UI) y control de API de manera cómoda gracias Wowza Stream Engine Manager.
- Planes de soporte técnico y servicios de consultoría.

Para la configuración del servidor se adquirió una licencia gratuita por tiempo limitada facilitada para estudiantes universitarios. La instalación se llevó a cabo en un PC cuyas características se detallan en la *Tabla 2: Características PC*, la versión de la herramienta utilizada es la **4.7.6** sobre el sistema operativo 10 Windows 10 Pro 64-bit.

Figura 10: Logo WOWZA STREAM ENGINE

Para prestar el funcionamiento de la herramienta se hace uso de 2 servicios (figura 11):

- Wowza Streaming Engine 4.7.6, que permite la transmisión y tratamiento de vídeo y audio contra el servidor.
- Wowza Streaming Engine Manager 4.7.6, que aloja el servicio web para la configuración del servidor a través de su interfaz.

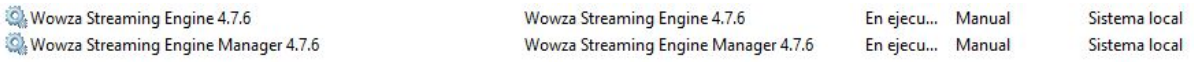

Figura 11: Servicios de Wowza Streaming Engine en Windows

Para acceder a la interfaz de configuración del servidor disponemos del siguiente enlace local:
http://localhost:8088/enginemanager/login.htm#home/_defaultVHost_
desde el que es posible acceder mediante un navegador web haciendo uso del protocolo HTTP.
Para ello se definen unos credenciales de administración durante la instalación. Ejemplos de las interfaces de acceso y del monitor del "*Manager*" se pueden observar en las Figuras 12 y 13.

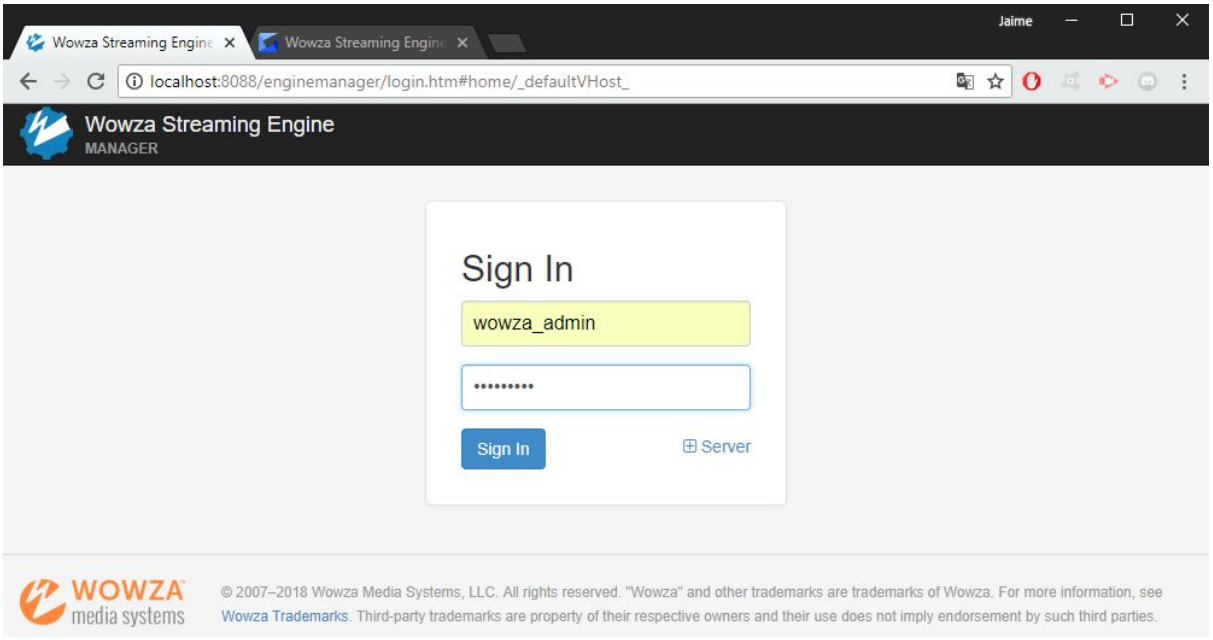

Figura 12: Pantalla de acceso a Wowza Streaming Engine Manager

Desde el interior de Wowza Streaming Engine Manager es posible observar y modificar los parámetros de configuración del servidor, además de monitorizar el uso de recursos y obtener estadísticas. Para la configuración del canal de transmisión vídeo se optó por configurar uno para la transmisión del vídeo de los dispositivos móviles, bajo el nombre de aplicación ***casus***.

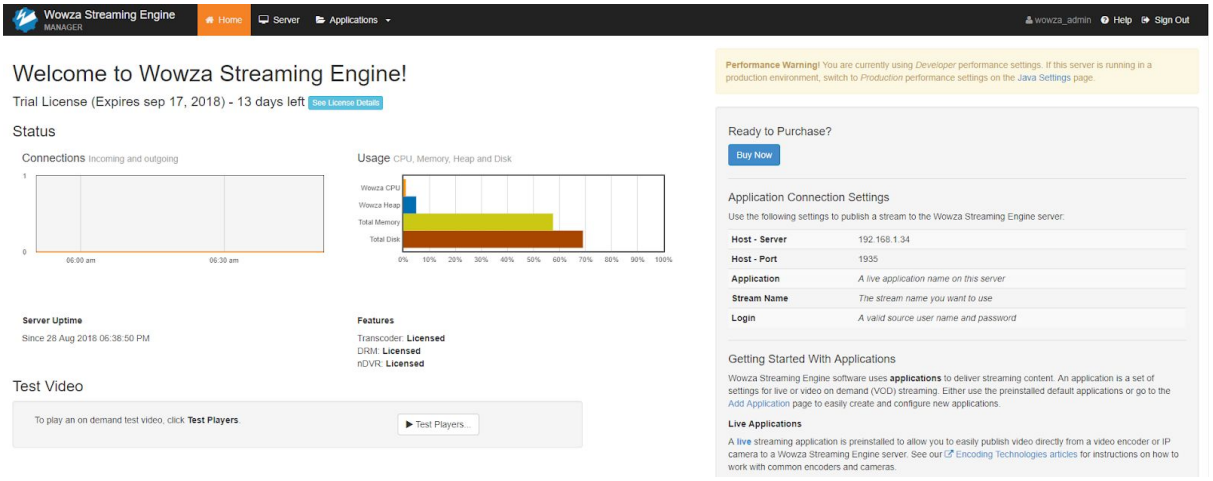

Figura 13: Interfaz de Wowza Streaming Engine Manager

La herramienta proporciona múltiples medios para aportar seguridad a las conexiones, tanto para la transmisión de vídeo como para la reproducción del mismo. Por ejemplo, para este proyecto se definieron unas credenciales ***publisher*** que son necesarias para poder realizar el envío de vídeo en la aplicación *casus*.

3.5 Firebase

Firebase supone el medio principal de comunicación a través de internet para la aplicación. Consiste en una plataforma que permite el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2011.

Esta plataforma comenzó proporcionando a los desarrolladores una API que permitía la integración de sistemas de chat en sus páginas web. Sin embargo este chat estaba siendo utilizado en gran escala por los desarrolladores para pasar paquetes de información de sus aplicaciones, como el estado de las partidas en el caso de los juegos. De manera que se decidió separar estas funcionalidades, el sistema de chat y el sistema de arquitectura en tiempo real consiguiendo una API con un gran potencial que acabó dando como resultado la fundación de Firebase (Abril de 2012). Tras esto, en 2014, Firebase fue comprado por Google y desde su adquisición, ha crecido y ha expandido sus servicios para convertirse en una plataforma unificada que otorga gran funcionalidad y posibilidades para el desarrollo de aplicaciones móviles. Además, actualmente se integra con otros servicios de Google para poder ofrecer productos a mayor escala para los desarrolladores, como por ejemplo, en octubre 2017 Firebase lanzó Cloud Firestore, una base de datos NoSQL en la nube.

En este proyecto, Firebase es de vital importancia, porque supone la plataforma de comunicación inalámbrica a través de la red 4G. En este sentido, todo la información obtenida de los sensores, posición GPS, datos del magnetómetro y acelerómetro, son transferidos de diferentes dispositivos android que se encuentren ejecutando la aplicación, gracias a

las herramientas que nos otorga Firebase Console. Entre ellas se encuentra: sistemas de almacenamiento, sistemas de autenticación, estadísticas y analíticas, sistemas de alojamiento, etc. Podemos ver la interfaz de la consola en la *Figura 13:Firebase console*, junto con algunas de las funciones principales.

En concreto en este proyecto se ha dado uso de la herramienta de base de datos en tiempo real (figura 14), que de manera muy rápida almacena datos en formato PHP (campo-valor de texto) y ser actualizados en otros dispositivos, detectando el cambio de forma automática.

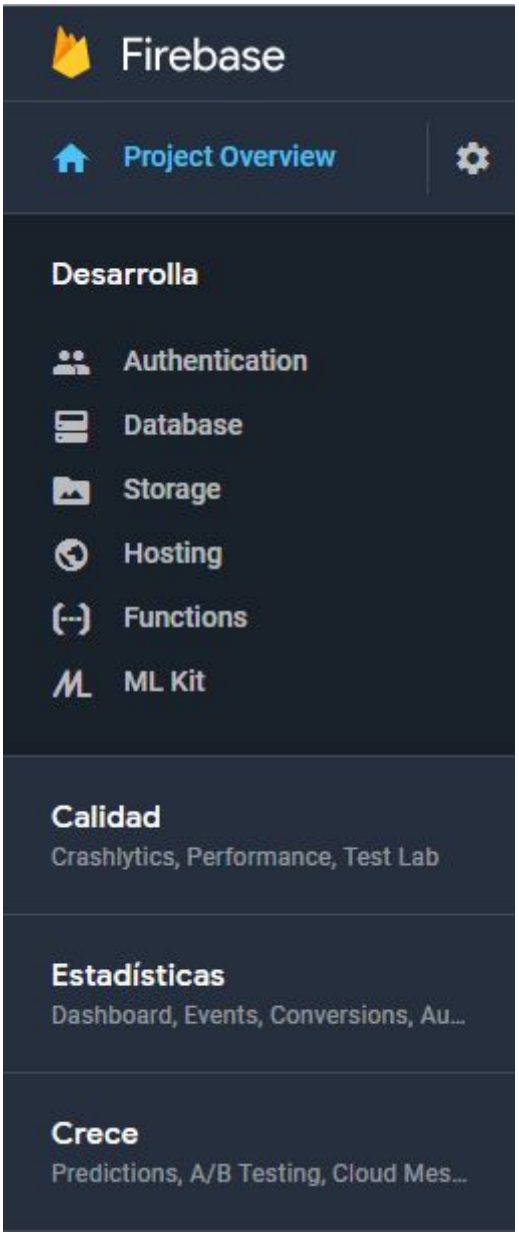

Figura 13:Firebase console

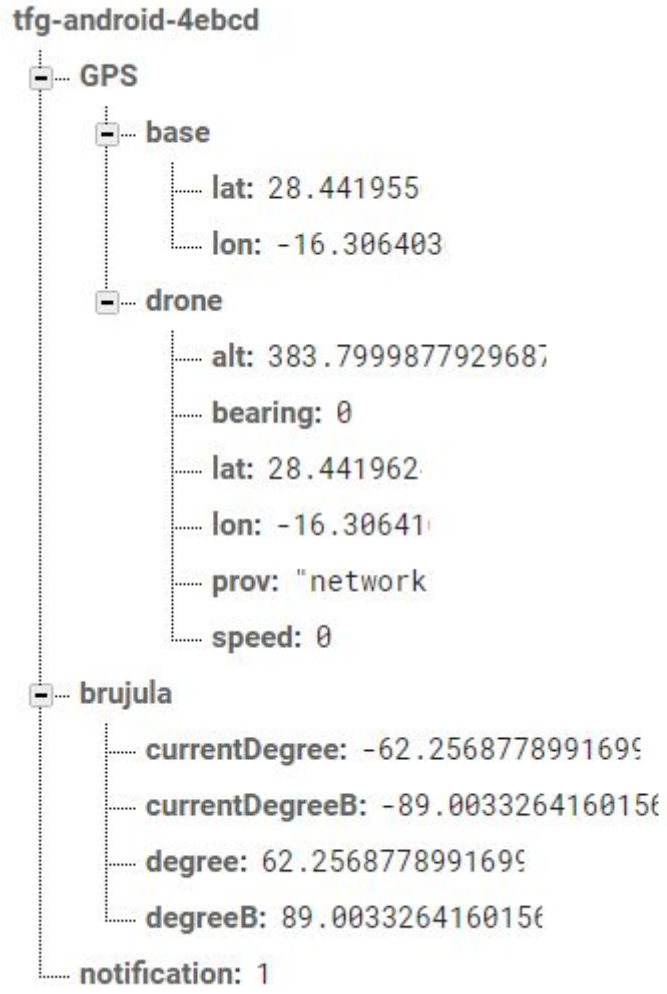

Figura 14: Base de Datos en tiempo real de Firebase de este proyecto

3.6 Github

GitHub es la plataforma de desarrollo colaborativo más extendida, esta permite alojar proyectos utilizando el sistema de control de versiones Git. En concreto, su uso principal es el de la creación de código fuente de programas de computadora. El software que opera GitHub fue escrito en Ruby on Rails, y el código de los proyectos alojados en en ella se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

En este proyecto ha supuesto una vía de intercambio que ha permitido hacer el código accesible para la revisión y consulta de los tutores del mismo. Además, aunque se trate de un desarrollo individual me ha permitido desarrollar el código mediante un control de versiones y conseguir accesibilidad total del código permitiendo su desarrollo en diferentes dispositivos.

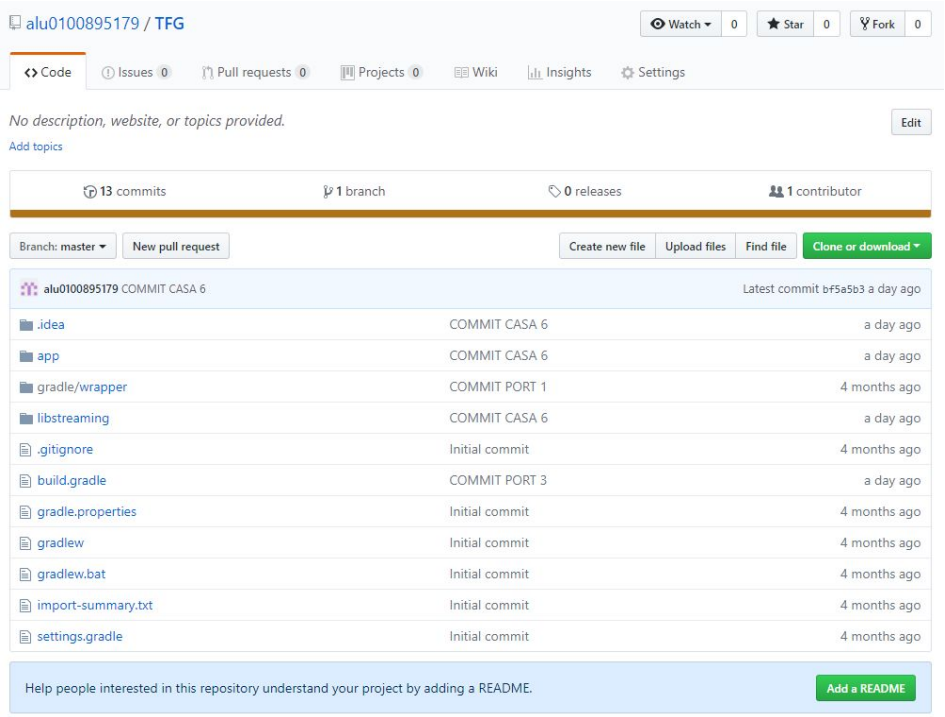

Figura 15: Repositoria de Github que aloja el proyecto

Tras la finalización del proyecto, todo el código y estructura del mismo se encuentran accesibles para su libre consulta y uso en el siguiente repositorio

(figura 15): https://github.com/alu0100895179/TFG

3.7 Librerías, APIs y otros recursos

Para el desarrollo de la aplicación se han incluido algunas librerías fundamentales como soporte al código desarrollado.

- La principal de todas ellas sería la librería ***libstreaming***, incorporada al proyecto para lograr la transmisión de vídeo a través del protocolo RTSP. El código fuente se encuentra accesible en el siguiente repositorio: https://github.com/fyhertz/libstreaming.
  De esta manera se desarrolló la clase StreamingClass (figura 17), que hace uso de los métodos y definiciones de esta librería para lograr el envío de streaming de vídeo.

- La API de Google Maps, ***Maps SDK for Android***, para poder interactuar con los mapas en la APP. De esta manera fue necesario incluir las librerías correspondientes. Para ello se definió una clase ***MapsActivity*** y se incluyeron las librerías que se muestran en la *Figura 16: Librerías de Google Maps incluidas*.

- Además se incluyó ***Firebase*** en el proyecto como mecanismo seguro de comunicación entre distintos dispositivos que se encuentren ejecutando la aplicación. Para ello se hizo uso de las herramientas que ofrece Android Studio para incorporar las funcionalidades de Firebase al proyecto, como es ***Firebase Assistant*** (figura 18). Para interactuar con la base de datos en tiempo real es necesario incluir las librerías y utilizar los métodos que proporciona.

- Elementos gráficos. Se incluyeron numerosos elementos gráficos, sobretodo iconos a través de la siguiente web: https://icons8.com/icon/new-icons/all . Estos se encuentran en la ruta **app > res > drawable** dentro de la estructura del proyecto.

```
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.UiSettings;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
```

Figura 16: Librerías de Google Maps incluidas

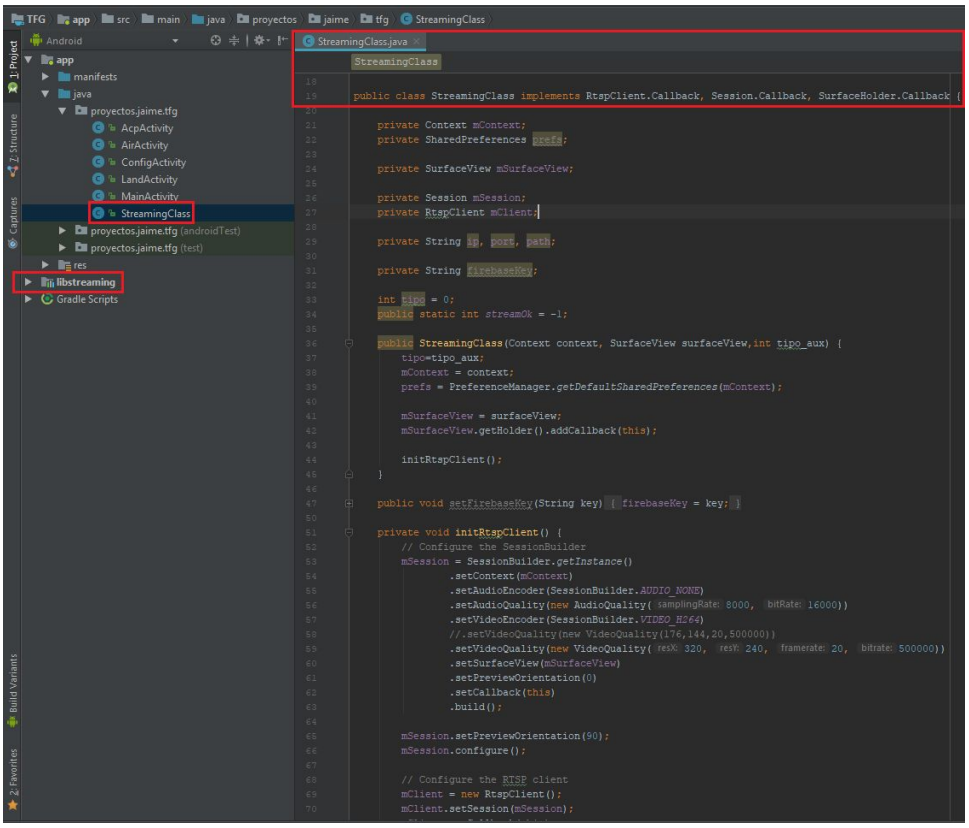

Figura 17: Implementación de clase StreamingClass

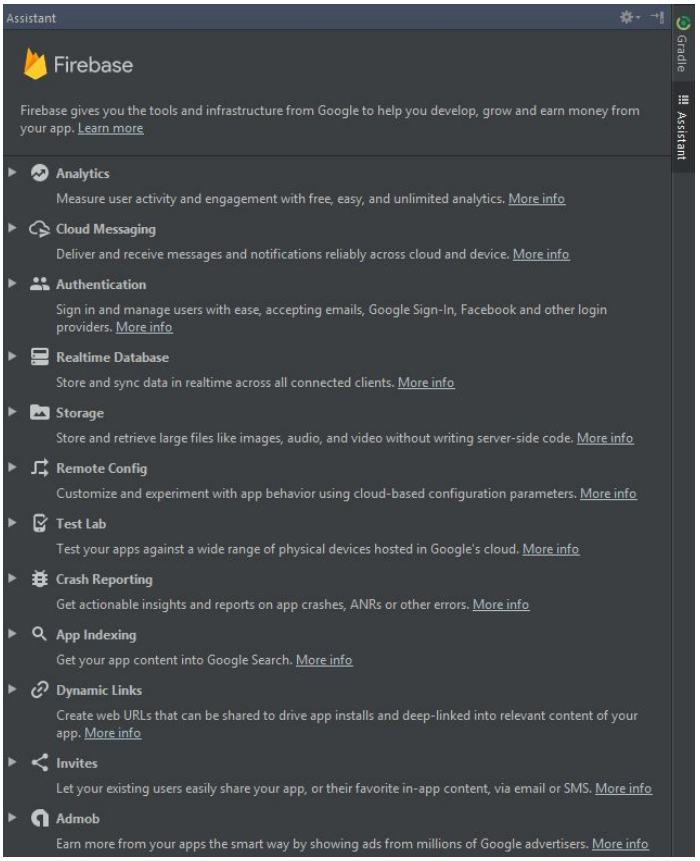

Figura 18: Firebase Assistant

Capítulo 4.
Aplicación móvil.

4.1 Características

Las características de la aplicación móvil en el momento de desarrollo de esta memoria, son las que figuran en la *Tabla 7: Características de APP*, además los requisitos técnicos para la ejecución de la aplicación ya se mencionaron en el capítulo *2.2.4 Requisitos de la APP*.
Sin embargo, es posible que versiones posteriores a la actual estas características y requisitos varíen. Por ejemplo al incluir funcionalidades como las que se encuentran en el capítulo 6, por lo que estos datos podrían variar en versiones posteriores.

Sistema Operativo	Android (versión 4.4 o superior)
SDK Mínimo necesario	19
SDK Mínimo recomendado	24
Nombre de la APP	Air Monitoring
Espacio de almacenamiento	4 MB apróx.
Uso medio de memoria	16 MB apróx.
Uso de datos móviles	Sí.
Uso de notificaciones	Sí
Permisos necesarios a conceder por el usuario	Cámara (perfiles aire y tierra) y ubicación (perfil aire).
Versión	1.0

Tabla 7: Características de APP

4.2 Funcionalidades

En el apartado 2.3 se menciona una secuencia de consecución de objetivos para el desarrollo del proyecto, al cumplir todos ellos hemos logrado que la aplicación cuente con diferentes funciones que pueden ser de utilidad en situaciones de emergencia tal como se planteaba en un origen.

Para ello las funciones de cada perfil han sido programadas a través de 6 *Activities* y una clase (figura 18), todas ellas definidas en los ficheros: **AcpActivity.java, AirActivity.java, ConfigActivity.java, LandActivity.java, MainActivity.java, MapsActivity.java, StreamingClass.java**.

Todos ellos localizados en la ruta de la aplicación: **TFG\app\src\main\java\proyectos\jaime\tfg**

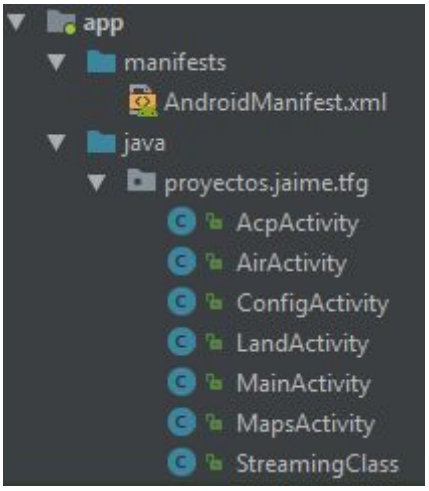

Figura 18: Actividades y clases

4.2.1 Menú de selección de modo

Al comenzar la ejecución de la aplicación entra en funcionamiento la actividad: ***MainActivity***, esta funciona como un menú donde se debe seleccionar el modo de ejecución correspondiente, la interfaz de este queda comentada en el capítulo 4.3.
A través de este menú se podrán seleccionar 4 modos diferentes: Aire, Configuración, Tierra y PMA (identificados en inglés).

Tras seleccionar el modo correspondiente se verifica que se tengan los permisos correspondientes, en caso de que no estén concedidos se solicitarán gráficamente al usuario (figura 19) y se procederá a crear un nuevo *Intent* para llamar a la siguiente activity, como se observa en la figura 20.

Figura 19: Solicitud de permisos

```
Intent i = new Intent( packageContext: this, AirActivity.class);
startActivity(i);
```

Figura 20: Comienzo de nueva activity

4.2.2 Modo aire

Si en el menú inicial se conceden los permisos necesarios (GPS y Cámara) tras presionar el botón correspondiente se crea la actividad ***AirActivity***.

Este modo tiene la función de enviar todos los datos de telemetría y vídeo desde el dron. De manera que tras haber cargado la interfaz lo primero que realiza la activity es la declaración un objeto de la clase ***StreamingClass*** (figura 21), tipo 0 ya que se trata de un streaming de modo aire, para comenzar la transmisión RTSP haciendo uso de la librería ***libstreaming*** como se comentó en el apartado 3.7

```
SurfaceView mSurfaceView;
mSurfaceView = findViewById(R.id.surface_air);
StreamingClass st = new StreamingClass( context: this, mSurfaceView,  tipo_auxc: 0);
st.toggleStreaming();
```

Figura 21: Declaración de objeto StreamingClass para aire

Con esto se establece la conexión con el servidor RTSP y comienza la transmisión de vídeo capturado por la cámara al motor Wowza.

Seguidamente de esto se declaran los sensores acelerómetro y magnetómetro, además del proveedor GPS, (figura 22) para actualizar los datos obtenidos de los sensores de forma continua y automática.

```
Log.d( tag: "TFG_debug",  msg: "Sensores brujula");
mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
magnetometer = mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
mGravity = null;
mGeomagnetic = null;

Log.d( tag: "TFG_debug",  msg: "Obtener proveedores de GPS.");
locationManager = (LocationManager) this.getSystemService(LOCATION_SERVICE);
String provider = get_best_provider();
if(provider!=null) {
    locationManager.requestLocationUpdates(get_best_provider(),  minTime: 500,  minDistance: 0,  listener: this);
}
```

Figura 22: Declaración de sensores

El dispositivo en este perfil actualiza la localización GPS en tiempo real gracias a la ejecución de la función:

```
public void onLocationChanged(Location location);
```

Cuando la posición cambia, ésta entra en ejecución almacenando en la variable ***location*** y accedemos a la información contenida en ella de la siguiente forma:

```
lat = location.getLatitude();
lon = location.getLongitude();
alt = location.getAltitude();
bearing = location.getBearing();
speed = location.getSpeed();
String prov = location.getProvider();
```

Extraemos altitud, latitud y longitud, velocidad de desplazamiento y *"bearing"* (rumbo, ángulo con el norte respecto al punto donde nos encontramos). Sin embargo, esta información puede aparecer incompleta si el proveedor no es el GPS sino que la triangulación se realiza a través de la red GSM. Esto suele suceder cuando el terminal no se encuentra en exteriores, impidiendo que el GPS funcione de manera correcta. Cuando la aplicación detecte su primera ubicación toma ésta como las coordenadas *"base"*.

Toda esta información tras haberla almacenado se envía a la base de datos de Firebase para que pueda ser tratada por el perfil PMA ejecutado en otro dispositivo.

```
DatabaseReference latRef = database.getReference("GPS/drone/lat");
DatabaseReference lonRef = database.getReference("GPS/drone/lon");
DatabaseReference altRef = database.getReference("GPS/drone/alt");
DatabaseReference bearingRef = database.getReference("GPS/drone/bearing");
DatabaseReference speedRef = database.getReference("GPS/drone/speed");
DatabaseReference provRef = database.getReference("GPS/drone/prov");

latRef.setValue(lat);
lonRef.setValue(lon);
altRef.setValue(alt);
bearingRef.setValue(bearing);
speedRef.setValue(speed);
provRef.setValue(prov);
```

Magnetómetro y acelerómetro producirán un evento cada vez detecten un cambio en sus valores, y este es capturado por la función

```
public void onSensorChanged(SensorEvent event);
```

En esta se realizan los cálculos correspondientes en grados y se envían a Firebase para poder tratarlos en el PMA logrando generar una brújula remota en tiempo real.
Conoceríamos la orientación respecto al norte, respecto a la base y el rumbo que está tomando el dron.

4.2.3 Modo Tierra

Este perfil está programado como una versión reducida del perfil de aire, en la que únicamente se accede a la cámara.
Se declara el objeto ***StreamingClass*** de igual modo que en el perfil de aire, pero con tipo auxiliar 1 ya que es el canal de tierra (figura 23).

```
SurfaceView mSurfaceView;
mSurfaceView = findViewById(R.id.surface_land);
StreamingClass st = new StreamingClass( context: this, mSurfaceView, tipo_aux: 1);
st.toggleStreaming();
```

Figura 23: Declaración de objeto StreamingClass para tierra

Además de esto se realizan comprobaciones de si se debe notificar las coordenadas del drone, en caso de que el PMA haya solicitado la notificación se actualiza la posición accediendo a Firebase (figura 24) y se crea la notificación (figuras 25 y 29).

```
droneRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        lat = Double.parseDouble(dataSnapshot.child("lat").getValue().toString());
        lon = Double.parseDouble(dataSnapshot.child("lon").getValue().toString());
    }
```

Figura 24: Acceder a la posición del drone en Firebase

```
createNotificationChannel();
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID)
        .setSmallIcon(R.drawable.droneicon)
        .setContentTitle("DRONE Coordinates")
        .setContentText("Latitude: " + lat + " | Longitude: " + lon)
        .setStyle(new NotificationCompat.BigTextStyle()
                .bigText("Latitude: " + lat + " | Longitude: " + lon))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(getApplicationContext());
notificationManager.notify(idNoti, mBuilder.build());
```

Figura 25: Construcción de notificación con coordenadas

4.2.4 Modo PMA

Si en el menú inicial se presiona el botón de este modo (ACP ya que se trata de las siglas en inglés) se crea la actividad ***AcpActivity***. Este rol no es necesario que conceda ningún permiso en concreto, ya que únicamente accede a la información a través de internet.

Comienzan declarándose las variables que permiten mostrar el streaming de vídeo y el mapa de google maps, además de múltiples variables ***ImageView*** y ***TextView*** ya que en este perfil, a diferencia del resto, abundan los elementos gráficos.

```
MapFragment mapFragment;
GoogleMap mMap;
VideoView mVideoView;
```

A través de las variables TextView se mostrará la información obtenida del GPS accediendo a Firebase. Los datos se actualizan de forma automática detectando los cambios en la base de datos y se formatean antes de mostrarlos en la interfaz, en la figura 26 se muestra como se actualiza y tratan los datos para mostrar las coordenadas del drone.

Los datos de la brújula se actualizan de forma automática de igual manera que se hace para el GPS, tras esto se generan los movimientos sobre la ImageView de la brújula, a través de la función:

```
RotateAnimation ra = new RotateAnimation(
        currentDegree,
        degree,
        Animation.RELATIVE_TO_SELF, 0.5f,
        Animation.RELATIVE_TO_SELF,
        0.5f);
imgCompass.startAnimation(ra);
```

Para los 4 botones que se muestran existe una función para cada uno:

```
public void pulsado_boton_drone (View vista);
public void pulsado_boton_man (View vista);
public void pulsado_boton_maps (View vista);
public void pulsado_boton_notification (View vista);
```

De manera que se muestra el elemento gráfico correspondiente y se oculta el resto (alternando entre Streaming del drone, de tierra y el mapa). El 4º botón únicamente indica que se debe generar la notificación.

```
DatabaseReference droneRef = database.getReference( s: "GPS/drone");
droneRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

        alt = Double.parseDouble(dataSnapshot.child("alt").getValue().toString());
        lat = Double.parseDouble(dataSnapshot.child("lat").getValue().toString());
        lon = Double.parseDouble(dataSnapshot.child("lon").getValue().toString());
        bearing = Double.parseDouble(dataSnapshot.child("bearing").getValue().toString());
        speed = Double.parseDouble(dataSnapshot.child("speed").getValue().toString());
        String prov = dataSnapshot.child("prov").getValue().toString();

        String latitudeS = "";
        String longitudeS = "";

        int latINT = (int) lat;
        latitudeS += latINT + "°";
        double latDBL = (lat-latINT)*60;
        latINT = (int) latDBL;
        latitudeS += latINT + "'";
        latDBL = (latDBL-latINT)*60;
        latINT = (int) latDBL;
        latitudeS += latINT + "''";

        int lonINT = (int) lon;
        longitudeS += lonINT + "°";
        double lonDBL = (lon-lonINT)*60;
        lonINT = (int) lonDBL;
        longitudeS += lonINT + "'";
        lonDBL = (lonDBL-lonINT)*60;
        lonINT = (int) lonDBL;
        longitudeS += lonINT + "''";

        latText.setText(latitudeS);
        lonText.setText(longitudeS);
        altText.setText(dec2.format(alt) + "m");
        bearingText.setText(""+ dec2.format(bearing) + " º");
        speedText.setText("" + dec2.format(speed) + " m/s");
```

Figura 26: Actualizar coordenadas drone - PMA

4.3 Diseño

El diseño de la aplicación en su versión actual viene determinado por diferentes pantallas, todas ellas diseñadas mediantes layouts en formato XML.

- Menú inicial - vertical (figura 27), que permite interactuar al usuario seleccionando entre los 3 perfiles de funcionamiento cuál desea utilizar o dando opción a configurar la dirección del servidor RTSP. Para ello se muestran 4 botones circulares que representan los 3 modos además del botón de configuración. Viene definido en el fichero ***activity_main.xml***

- La pantalla en modo aire y en modo tierra - horizontal, permanecerá en negro ya que el usuario no debe interactuar con el dispositivo y así permitimos ahorrar batería del terminal. Únicamente se mostrará el estado de los sensores y la cámara (rojo activo, verde inactivo). Estas interfaces vienen definidas en los ficheros ***air_layout.xml*** y ***land_layout.xml***

- La pantalla PMA - horizontal (figura 28), mostrará una interfaz donde se podrá mostrar el vídeo obtenido del dispositivo de aire, tierra o el mapa, alternando con los botones correspondientes. Superpuesto sobre esta se encontrará en la esquina superior izquierda la información de localización del mismo en tiempo real. En la esquina inferior derecha podremos ver una pequeña brújula que nos permitirá conocer la orientación hacia el norte, base y rumbo gracias a diferentes flechas de colores. En la parte inferior izquierda encontraremos botones circulares para alternar la visualización mencionada o generar notificaciones al personal de tierra, todas ellas cuentan con un icono identificativo. Estas interfaz viene se encuentra definida en el fichero ***acp_layout.xml***.

Figura 27: Pantalla inicial

Figura 28: Interfaz de PMA

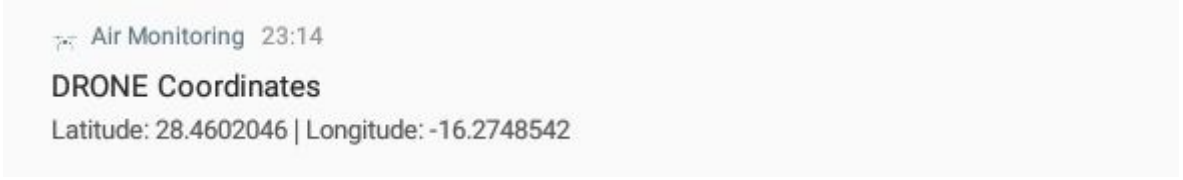

Figura 29: Diseño de las notificaciones

Capítulo 5.
Presupuesto.

En este capítulo se pretende establecer un presupuesto global de desarrollo del proyecto en un entorno real. De manera que se realizará una división en: Costes de Personal, Hardware, Software y Global.

5.1 Personal

En este apartado se exponen los costes de los recursos humanos necesarios para el desarrollo del proyecto. Se ha tomado el valor de 15€ la hora para la jornada de un ingeniero informático.

Tarea	Horas	Coste (€)
Estudio de mercado de mecanismos de transmisión de streaming de vídeo cifrado.	15	225
Estudio del protocolo de Streaming RTSP.	10	150
Estudio y selección de dispositivos móviles.	5	75
Estudio y selección de RPAs.	5	75
Estudio y selección de Hardware para el servidor RTSP.	5	75
Estudio y selección del resto de tecnologías y herramientas necesarias.	10	150
Montaje y configuración motor de streaming de vídeo. Red de área local.	20	300
Finalizar implementación de servicio RTSP. Servicio hacia a internet.	20	300
Programación de la aplicación: perfil de aire.	50	750

Programación de la aplicación: perfil de tierra.	10	150
Programación de la aplicación: perfil de PMA.	45	675
Programación de la aplicación: parámetros de configuración.	30	450
Diseño gráfico de toda la aplicación.	60	900
Interacción con Google Maps y notificaciones.	30	450
Estudiar y aplicar seguridad a todas las comunicaciones.	50	750
Pruebas, configuración global de todo el sistema, depuración de los últimos errores	30	450
Redacción de la memoria y documentación.	50	750
Total	**445**	**6.775**

Tabla 5: Costes Personales.

5.2 Hardware

En este apartado se exponen el coste de la adquisición del hardware y sistemas para el desarrollo del proyecto.

Tarea	Coste (€)
Servidor para alojar servicio RTSP.	1000
Drone: Phantom 4.	1500
PC para el desarrollo de la aplicación.	800
BQ Aquarius X Pro (anclado a drone)	260
Terminales móviles para unidades de tierra ej. BQ Aquarius X Pro (x 10)	2600
Total	**6.216**

Tabla 6: Costes Hardware.

5.3 Software

En este apartado se exponen el coste de la adquisición del software y herramientas de licencia necesarias para el desarrollo del proyecto. Suponemos coste 0 para S.O. ya que existe la posibilidad de utilizar en todos los sistemas sistemas operativos libres como podría ser Linux.

Tarea	Coste (€)
Licencia desarrollador Android	22
Licencia Anual Wowza Streaming Engine	56
Licencia mensual de servicio Wowza Stream CLOUD$_{x12}$	2.040
Total	**2.118**

Tabla 7: Costes Software.

5.4 Global

Por último se adjunta el coste total del proyecto, observando así el presupuesto global.

Tarea	Coste (€)
Costes: Personal.	6.775
Costes: Hardware.	6.216
Costes: Software	2.118
Total	**15.109**

Tabla 8: Presupuesto Global.

Capítulo 6.
Conclusiones y trabajos futuros.

Este trabajo ha conseguido desarrollar un sistema que puede ser de una utilidad real en situaciones de emergencia, donde el control de la información es de vital importancia. De esta manera podemos obtener información en tiempo real que en muchas ocasiones va a acelerar las respuestas a situaciones críticas donde cada segundo cuenta, tanto desde el PMA como al tener posibilidad de dar órdenes a las unidades de tierra. De manera que se logra lo que se planteaba en un comienzo. Para que el código de la APP sea accesible queda disponible en el siguiente repositorio de Github: https://github.com/alu0100895179/TFG

Se pretende dejar este trabajo disponible para desarrollar nuevas líneas de desarrollo como podrían ser:

- Calcular las rotaciones en los tres ejes perpendiculares entre sí, cuyo punto de intersección está situado sobre su centro de gravedad del DRON. Es decir representar el cabeceo, alabeo y guiñada del RPA en vuelo.
- Calcular información de óptica conociendo los parámetros del objetivo de la cámara del smartphone. Esto nos podría servir como información útil para el cálculo de parámetros de interés, como podría ser la altura con respecto al suelo, distancia a la que se encuentran víctimas o generar una escala.
- Pintar una retícula a modo de cuadrícula sobre la pantalla con información de una escala para el cálculo de distancias. Tal vez además podría intentarse hacer esta responsiva a la altura de vuelo del dron.
- Geolocalizar al personal de tierra, de igual modo que se hizo con el drone. De esta manera también podríamos pintar esta posición en Google Maps.
- A pesar de no contar con muchos elementos de texto, añadir más lenguajes quedando como una aplicación multiidioma.

Capítulo 7.
Summary and Conclusions.

This work has managed to develop a system that can be of real utility in emergency situations, where the control of information is of vital importance. In this way we can obtain information in real time that in many occasions will accelerate the responses to critical situations where every second counts, both from the PMA and having the possibility of giving orders to the land units. The APP code is accessible and available at the following Github repository: https://github.com/alu0100895179/TFG

It is intended to leave this work available to develop new lines of development such as:

- Calculate the rotations in the three axes, whose point of intersection is located on the center of gravity of the DRON. That is, to represent the pitch, roll and yaw of the RPA in flight.
- Calculate optical information knowing the parameters of the lens of the smartphone camera. This could serve as useful information for the calculation of parameters of interest, such as height with respect to the ground, distance at which victims are found, etc.
- Paint a grid on the screen with information of a scale for the calculation of distances. Maybe you could also try to make this responsive to the height of the drone flight.
- Despite not having many text elements, add more languages remaining as a multi-language application.
- Geolocalize land staff, just as it was done with the drone. This way we could also paint this position in Google Maps.

Anexo A.
Ciclo de desarrollo

Como tarea de Ingeniería del Software, disciplina de desarrollo en la que este proyecto se engloba. Se registró de manera concreta el flujo de trabajo como reflejo de la consecución de objetivos durante el desarrollo del proyecto:

Objetivo	Tarea	Herramientas y Dispositivos
1	Configurar motor para streaming de vídeo cifrado, en concreto Wowza Streaming Engine.	PC personal para pruebas en área local.
2	Diseño de 1ª interfaz de menú de la aplicación para la selección de perfiles o modos de funcionamiento.	Android Studio y aplicación en Dispositivo Virtual.
3	Comenzar a trabajar en el perfil de modo aire.	Android Studio y smartphone personal.
4	Incluir librería *libstreaming* en el proyecto para la transmisión de vídeo.	Android Studio.
5	Comenzar a transmitir vídeo desde el smartphone al motor en la red de área local.	Android Studio y smartphone personal.
6	Comprobar recepción de vídeo desde reproductor multimedia de PC local, haciendo uso para ello del protocolo RTSP.	VLC media player, smartphone personal, PC personal.
7	Comenzar a trabajar con los sensores de manera local, perfil de PMA.	Android Studio y smartphone personal.
8	Trabajar en la interfaz de recepción y muestreo de los datos, perfil PMA.	Android Studio y aplicación en Dispositivo Virtual.
9	Obtener localización GPS de manera local	Android Studio y smartphone personal.
10	Obtener información del magnetómetro y	Android Studio y

	acelerómetro de manera local.	smartphone personal.
11	Interpretar, formatear y mostrar información extraída del GPS.	Android Studio y smartphone personal.
12	Interpretar, formatear y mostrar información extraída del acelerómetro y magnetómetro.	Android Studio y smartphone personal.
13	Comenzar a utilizar el smartphone seleccionado para el proyecto para las pruebas y desarrollo: BQ Aquarius X Pro.	Android Studio y smartphone seleccionado para proyecto (BQ).
14	Transmisión de vídeo entre 2 terminales móviles a través de la red local.	Smartphone seleccionado para proyecto (BQ) y smartphone personal.
15	Montaje de servidor RTSP público por parte del equipo de CyrptULL.	Servidores proporcionados por ULL.
16	Transmisión de vídeo entre 2 terminales móviles a través de la red móvil 4G.	Smartphone seleccionado para proyecto (BQ) y smartphone personal.
17	Adición de Firebase al proyecto para la transmisión datos entre los dispositivos.	Android Studio y Firebase.
18	Envío de datos extraídos del GPS de manera automática a Firebase.	Smartphone seleccionado para proyecto (BQ) y Firebase.
19	Envío de datos extraídos del acelerómetro y magnetómetro de manera automática a Firebase.	Smartphone seleccionado para proyecto (BQ), smartphone personal y Firebase.
20	Recepción, interpretación y mostrado de datos extraídos del GPS de manera automática a través de Firebase.	Smartphone seleccionado para proyecto (BQ), smartphone

		personal y Firebase.
21	Recepción, interpretación y mostrado de datos extraídos del acelerómetro y magnetómetro de manera automática a través de Firebase.	Smartphone seleccionado para proyecto (BQ), smartphone personal y Firebase.
22	Aplicar mayores capas de protección a la transmisión de datos a través de Firebase.	Android Studio.
23	Actualizar interfaz de menú de selección de perfiles, añadir modo de tierra y parámetros de configuración.	Android Studio y aplicación en Dispositivo Virtual.
24	Incorporar la funcionalidad correspondiente ya implementada en el perfil de aire al perfil de tierra. Pruebas y configuraciones necesarias.	Smartphone seleccionado para proyecto (BQ), smartphone personal y Firebase.
25	Funcionalidad alternar/superposición de capas al añadir librería de Google Maps al proyecto.	Android Studio y smartphone personal.
26	Google Maps funcionando correctamente con los datos obtenidos de manera remota y representandolos de manera correspondiente. Alternar capas de manera satisfactoria.	Smartphone seleccionado para proyecto (BQ), smartphone personal y Firebase.
27	Añadir funcionalidad: alarmas y notificaciones. Generadas por PMA a unidades de tierra.	Smartphone seleccionado para proyecto (BQ), smartphone personal y Firebase.
28	Depuración, control y manejo de errores. Pruebas finales de validación.	Todas las anteriormente mencionadas.
29	Finalización de la memoria esbozada simultáneamente al desarrollo de la aplicación.	Google Docs.

Tabla x: Secuencia final de objetivos

Anexo B.
Especificaciones de terminales móviles.

NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2017, April
	Status	Available. Released 2017, June
BODY	Dimensions	146.5 x 72.7 x 7.8 mm (5.77 x 2.86 x 0.31 in)
	Weight	158 g (5.57 oz)
	SIM	Hybrid Dual SIM (Nano-SIM, dual stand-by)
DISPLAY	Type	LTPS IPS LCD capacitive touchscreen, 16M colors
	Size	5.2 inches, 74.5 cm^2 (~70.0% screen-to-body ratio)
	Resolution	1080 x 1920 pixels, 16:9 ratio (~424 ppi density)
	Multitouch	Yes, up to 10 fingers
	Protection	Dinorex glass
PLATFORM	OS	Android 7.1.1 (Nougat)
	Chipset	Qualcomm MSM8953-Pro Snapdragon 626
	CPU	Octa-core 2.2 GHz Cortex-A53
	GPU	Adreno 506
MEMORY	Card slot	microSD, up to 256 GB (uses SIM 2 slot)
	Internal	64/128 GB, 4 GB RAM
CAMERA	Primary	12 MP (f/1.8, 1/2.5", 1.4µm), PDAF, dual-LED dual-tone flash
	Features	Geo-tagging, touch focus, face/smile detection, Auto HDR, panorama
	Video	2160p@30fps, 720p@120fps
	Secondary	8 MP, f/2.0, 1080p@60fps, LED flash
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
		- 24-bit/192kHz audio - Active noise cancellation with dedicated mic
COMMS	WLAN	Wi-Fi 802.11 a/b/g/n/ac, dual-band, WiFi Direct, hotspot
	Bluetooth	4.2, A2DP, LE, aptX
	GPS	Yes, with A-GPS, GLONASS, GALILEO
	NFC	Yes
	Radio	FM radio
	USB	2.0, Type-C 1.0 reversible connector, USB On-The-Go
FEATURES	Sensors	Fingerprint (rear-mounted), accelerometer, gyro, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Email, IM
	Browser	HTML5
		- Fast battery charging (Quick Charge 3.0) - MP3/AAC/WAV player - MP4/H.264 player - Document viewer - Photo/video editor
BATTERY		Non-removable Li-Ion 3100 mAh battery
MISC	Colors	Midnight Black, Glaze White
	Price	About 370 EUR

Figura x: Especificaciones BQ Aquarius X Pro

NETWORK	Technology	GSM / HSPA / LTE EXPAND ▾
LAUNCH	Announced	2016, February
	Status	Available. Released 2016, March
BODY	Dimensions	150.9 x 72.6 x 7.7 mm (5.94 x 2.86 x 0.30 in)
	Weight	157 g (5.54 oz)
	Build	Front/back glass (Gorilla Glass 4), aluminum frame
	SIM	Single SIM (Nano-SIM) - G935F, G935W8 Hybrid Dual SIM (Nano-SIM, dual stand-by) - G935FD
		- Samsung Pay (Visa, MasterCard certified) - IP68 dust/water proof (up to 1.5m for 30 mins)
DISPLAY	Type	Super AMOLED capacitive touchscreen, 16M colors
	Size	5.5 inches, 83.4 cm^2 (~76.1% screen-to-body ratio)
	Resolution	1440 x 2560 pixels, 16:9 ratio (~534 ppi density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass 4
		- Always-on display - TouchWiz UI - Curved edge screen
PLATFORM	OS	Android 6.0 (Marshmallow), upgradable to Android 8.0 (Oreo)
	Chipset	Qualcomm MSM8996 Snapdragon 820 - G9350 Exynos 8890 Octa - G935FD, G935F, G935W8
	CPU	Quad-core (2x2.15 GHz Kryo & 2x1.6 GHz Kryo) - G9350 Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53) - G935FD, G935F, G935W8
	GPU	Adreno 530 - G9350 Mali-T880 MP12 - G935FD, G935F, G935W8
MEMORY	Card slot	microSD, up to 512 GB (uses SIM 2 slot) - dual SIM model only
	Internal	32/64/128 GB, 4 GB RAM
CAMERA	Primary	12 MP (f/1.7, 26mm, 1/2.5", 1.4µm, dual pixel PDAF), OIS, LED flash, check quality
	Features	Geo-tagging, simultaneous 4K video and 9MP image recording, touch focus, face/smile detection, Auto HDR, panorama
	Video	2160p@30fps, 1080p@60fps, 720p@240fps, HDR, dual-video rec., check quality
	Secondary	5 MP (f/1.7, 22mm, 1/4.1", 1.34µm), dual video call, Auto HDR
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
		- 24-bit/192kHz audio - Active noise cancellation with dedicated mic
COMMS	WLAN	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, hotspot
	Bluetooth	4.2, A2DP, LE, aptX
	GPS	Yes, with A-GPS, GLONASS, BDS
	NFC	Yes
	Radio	No
	USB	microUSB 2.0, USB Host
FEATURES	Sensors	Fingerprint (front-mounted), accelerometer, gyro, proximity, compass, barometer, heart rate, SpO2
	Messaging	SMS(threaded view), MMS, Email, Push Email, IM
	Browser	HTML5
		- Fast battery charging (Quick Charge 2.0) - Qi/PMA wireless charging (market dependent) - ANT+ support - S-Voice natural language commands and dictation - OneDrive (115 GB cloud storage) - MP4/DivX/XviD/WMV/H.264 player - MP3/WAV/WMA/eAAC+/FLAC player - Photo/video editor - Document editor
BATTERY		Non-removable Li-Ion 3600 mAh battery
	Talk time	Up to 27 h (3G)
	Music play	Up to 74 h
MISC	Colors	Black, White, Gold, Silver, Pink Gold, Black Pearl, Coral Blue
	SAR	1.17 W/kg (head) 1.59 W/kg (body)
	SAR EU	0.26 W/kg (head) 0.51 W/kg (body)
	Price	About 450 EUR
TESTS	Performance	Basemark OS II: 2107 / Basemark OS II 2.0: 2050 Basemark X: 28480
	Display	Contrast ratio: Infinite (nominal), 4.439 (sunlight)
	Camera	Photo / Video
	Loudspeaker	Voice 70dB / Noise 69dB / Ring 71dB
	Audio quality	Noise -92.5dB / Crosstalk -92.2dB
	Battery life	Endurance rating 92h

Figura x: Especificaciones Samsung Galaxy S7 Edge

NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2014, September
	Status	Available. Released 2014, September
BODY	Dimensions	146 x 72 x 7.3 mm (5.75 x 2.83 x 0.29 in)
	Weight	152 g (5.36 oz)
	Build	Front/back glass & aluminum frame
	SIM	Nano-SIM
		- IP68 dust/water proof (up to 1.5m for 30 mins)
DISPLAY	Type	IPS LCD capacitive touchscreen, 16M colors
	Size	5.2 inches, 74.5 cm^2 (~70.9% screen-to-body ratio)
	Resolution	1080 x 1920 pixels, 16:9 ratio (~424 ppi density)
	Multitouch	Yes, up to 10 fingers
	Protection	Scratch-resistant glass, oleophobic coating
		- Triluminos display - X-Reality Engine
PLATFORM	OS	Android 4.4.4 (KitKat), upgradable to 6.0 (Marshmallow)
	Chipset	Qualcomm MSM8974AC Snapdragon 801
	CPU	Quad-core 2.5 GHz Krait 400
	GPU	Adreno 330
MEMORY	Card slot	microSD, up to 256 GB (dedicated slot)
	Internal	16/32 GB, 3 GB RAM
MAIN CAMERA	Single	20.7 MP, f/2.0, 25mm, 1/2.3", 1.12µm, AF
	Features	LED flash, HDR, panorama
	Video	2160p@30fps, 1080p@60fps, 720p@120fps, HDR
SELFIE CAMERA	Single	2.2 MP, f/2.8
	Video	1080p@30fps
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes, with stereo speakers
	3.5mm jack	Yes
		- Active noise cancellation with dedicated mic
COMMS	WLAN	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, DLNA, hotspot
	Bluetooth	4.0, A2DP, aptX
	GPS	Yes, with A-GPS, GLONASS, BDS
	NFC	Yes
	Radio	FM radio, RDS
	USB	microUSB 2.0 (MHL 3 TV-out), USB Host; magnetic connector
FEATURES	Sensors	Accelerometer, gyro, proximity, compass, barometer
	Messaging	SMS(threaded view), MMS, Email, IM, Push Email
	Browser	HTML5
		- Fast battery charging (Quick Charge 2.0) - only Japanese versions - ANT+ support - Xvid/MP4/H.264 player - MP3/eAAC+/WAV/Flac player - Document viewer - Photo viewer/editor - Voice memo/dial
BATTERY		Non-removable Li-Ion 3100 mAh battery
	Stand-by	Up to 890 h (2G) / Up to 740 h (3G)
	Talk time	Up to 14 h (2G) / Up to 16 h (3G)
	Music play	Up to 130 h
MISC	Colors	Black, White, Copper, Silver Green, Purple
	SAR	0.80 W/kg (head) 1.44 W/kg (body)
	SAR EU	0.62 W/kg (head) 0.65 W/kg (body)
	Price	About 180 EUR
TESTS	Performance	Basemark OS II: 1109 / Basemark X: 12637
	Display	Contrast ratio: 1333:1 (nominal), 2.618 (sunlight)
	Camera	Photo / Video
	Loudspeaker	Voice 69dB / Noise 66dB / Ring 67dB
	Audio quality	Noise -86.4dB / Crosstalk -86.6dB
	Battery life	Endurance rating 85h

Figura x: Especificaciones Sony Xperia Z3

NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2015, March
	Status	Available. Released 2015, May
BODY	Dimensions	242.5 x 166.8 x 7.5 mm (9.55 x 6.57 x 0.30 in)
	Weight	450 g (Wi-Fi), 453 g (LTE) (1.00 lb)
	SIM	Micro-SIM
DISPLAY	Type	TFT capacitive touchscreen, 16M colors
	Size	9.7 inches, 291.4 cm^2 (~72.0% screen-to-body ratio)
	Resolution	768 x 1024 pixels, 4:3 ratio (~132 ppi density)
	Multitouch	Yes
PLATFORM	OS	Android 5.0 (Lollipop), upgradable to 6.0.1 (Marshmallow)
	CPU	Quad-core 1.2 GHz
MEMORY	Card slot	microSD, up to 256 GB (dedicated slot)
	Internal	16 GB, 1.5 GB RAM (Wi-Fi) 16/32 GB, 2 GB RAM (LTE)
MAIN CAMERA	Single	5 MP, AF
	Video	720p@30fps
SELFIE CAMERA	Single	2 MP
	Video	
SOUND	Alert types	Yes
	Loudspeaker	Yes, dual speakers
	3.5mm jack	Yes
COMMS	WLAN	Wi-Fi 802.11 a/b/g/n, dual-band, WiFi Direct, hotspot
	Bluetooth	4.1, A2DP
	GPS	Yes, with A-GPS, GLONASS
	Radio	No
	USB	microUSB 2.0
FEATURES	Sensors	Accelerometer
	Messaging	SMS(threaded view), MMS, Email, Push Email, IM
	Browser	HTML5
		- ANT+ support - MP4/H.264 player - MP3/WAV/eAAC+/Flac player - Photo/video editor - Document viewer
BATTERY		Non-removable Li-Ion 6000 mAh battery
	Talk time	Up to 14 h 30 min (multimedia) (2G) / Up to 40 h (3G)
	Music play	Up to 160 h
MISC	Colors	Smoky Titanium, Smoky Blue, White
	SAR EU	0.96 W/kg (body)
	Price	About 250 EUR

Figura x: Especificaciones Samsung Galaxy Tab A

Anexo C.
Especificaciones de drone PHANTOM IV.

AERONAVE

Peso (batería y hélices incluidas)	1 388 g
Tamaño diagonal (sin hélices)	350 mm
Velocidad de ascenso max.	Modo-S: 6 m/s (19.7 ft/s) Modo-P: 5 m/s (16.4 ft/s)
Velocidad de descenso max.	Modo-S: 4 m/s (13.1 ft/s) modo-P: 3 m/s (9.8 ft/s)
Velocidad max.	72 km/h (45 mph) (modo-S) 58 km/h (36 mph) (modo-A) 50 km/h (31 mph) (modo-P)
Ángulo de inclinación max.	42° (Modo-S) 35° (Modo-A) 25° (Modo-P)
Velocidad angular max.	250°/s (Modo-S) 150°/s (Modo-A)
Altura max. de servicio sobre el nivel del mar	6 000 m (19 685 pies)
Resistencia al viento max.	10 m/s
Tiempo de vuelo max.	30 minutos aprox.
Rango de temperatura de funcionamiento	De 0 a 40 °C (de 32 a 104 °F)
Sistemas de posicionamiento por satélite	GPS / GLONASS
Rango de precisión de vuelo estacionario	Vertical: ±0,1 m (con posicionamiento visual) ±0,5 m (con posicionamiento por GPS) Horizontal: ±0,3 m (con posicionamiento visual) ±1,5 m (con posicionamiento por GPS)

ESTABILIZADOR

Estabilización	3-ejes (cabeceo, alabeo, guiñada)
Intervalo controlable	Inclinación: -90° a +30°
Velocidad angular max. controlable	Cabeceo: 90°/s
Precisión del control angular	±0.02°

Figura X: Especificaciones DRON PHANTOM 4

Bibliografía

https://www.androidauthority.com/best-drone-apps-761228/
https://es.wikipedia.org/wiki/Android_Studio
http://cryptull.webs.ull.es/CASUS/
https://www.wowza.com/es/products/streaming-engine
https://github.com/fyhertz/libstreaming
https://developers.google.com/maps/documentation/android-sdk/intro
https://firebase.google.com/
https://support.brightcove.com/es/protecting-videos-hls-encryption
https://es.wikipedia.org/wiki/WebRTC
https://es.wikipedia.org/wiki/Protocolo_de_transmisi%C3%B3n_en_tiempo_real
https://www.tudelft.nl/en/ide/research/research-labs/applied-labs/ambulance-dron/
https://www.3cx.es/blog/webrtc-segura/
https://www.phonearena.com/