UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Sistema de Vigilancia Adaptativo basado en la Coordinación de UAVs en Entornos Afectados por Catástrofes

Juan Manuel Pérez Ramos

Septiembre, 2016

Sistema de Vigilancia Adaptativo basado en la Coordinación de UAVs en Entornos Afectados por Catástrofes

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

Sistema de Vigilancia Adaptativo basado en la Coordinación de UAVs en Entornos Afectados por Catástrofes

Autor: Juan Manuel Pérez Ramos

Director: Dr. David Vallejo Fernández

Septiembre, 2016

Juan Manuel Pérez Ramos

Ciudad Real – España

E-mail: JuanManuel.Perez@alu.uclm.es
Teléfono: 654 036 422

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

En este proyecto se analizan los vehículos aéreos no tripulados (más comúnmente conocidos como drones) como parte fundamental para la ayuda, a los equipos de emergencia, en labores relacionadas con situaciones de catástrofe, bien sea natural o artificial. Las situaciones de desastre pueden ser el escenario ideal para trabajar con drones, debido a las ventajas que proporcionan en este tipo de entornos.

Por esta razón, se diseña y desarrolla un sistema adaptativo que permita la coordinación de distintos UAVs en entornos que han sido afectados por una catástrofe. El sistema propuesto es capaz de monitorizar diferentes puntos de localización, según una prioridad determinada por los equipos de rescate, desplegar los drones que los equipos de emergencia consideren oportunos, acudir a los puntos de interés de forma coordinada y recolectar imágenes sobre el terreno que posteriormente serán analizadas de manera automática. Para llevar a cabo las pruebas del sistema se ha contado con el vehículo aéreo *3DR IRIS+*, que puede ser programado gracias al entorno de programación desarrollado por 3D Robotics, *DroneKit-Python*. Gracias a estos dos elementos el sistema es capaz de hacer volar el dron a cualquier punto de interés, pero para la captura de imágenes es necesario el uso de la librería *OpenCV*. Por último, las imágenes que han sido capturadas, en las diferentes zonas que los equipos de rescate han considerado oportunas, son analizadas por *Google Cloud Vision* API, obteniendo información complementaria del estado del territorio castigado por la catástrofe.

Mediante el uso de este sistema se espera conseguir una mejor productividad y eficiencia en las labores de rescate. Los drones pueden realizar un análisis aéreo del entorno en mucho menos tiempo que el personal de rescate terrestre. Además, la visión aérea ayuda a tomar decisiones más adecuadas y evita la exposición al peligro por parte del personal de emergencia.

Abstract

In this project, UAVs (more commonly known as drones) are analyzed as a key element to assist emergency teams in either natural or artificial disaster-related situations. The aforementioned situations might involve the ideal environment to work with drones because of the advantages they provide in such scenarios.

For that reason, an adaptive system that allows the coordination of different UAVs in disaster affected environments is designed and developed. The proposed system is able to monitor different location points depending on a priority previously specified by the rescue teams, to deploy drones that the emergency teams have deemed fit, to go to the points of interest in a coordinated manner, and to collect images on the ground which will then be automatically analyzed. To carry out the system testing, the aerial vehicle *3DR IRIS+* has been used, which can be programmed through the programming tool, developed by 3D Robotics, *DroneKit-Python*. Thanks to these two elements the system is able to fly the drone to any point of interest, while the use of the *OpenCV* Library is necessary when it comes to video capturing. Finally, the images that have been gathered in the different areas, considered by the rescue teams, are analyzed through the *Google Cloud Vision* API, retrieving additional information about the state of the disaster-damaged territory.

Thanks to the proposed system, improvements in terms of productivity and efficiency are expected to be achieved within the context of rescue efforts. Drones can perform an aerial analysis of the environment in much less time than rescue personnel on the ground. In addition, the aerial view is helpful to make better decisions and avoids the emergency personnel's exposure to danger.

Agradecimientos

Gracias a mis padres, Juan y Esther, por su apoyo incondicional y por enseñarme, a través de su trabajo, a seguir luchando para salir adelante en los momentos más difíciles. A mi hermana, Belén, por su paciencia durante tantos años, por ser mi red de seguridad, a pesar de la distancia, y por transmitirme siempre su alegría.

Gracias a mis abuelos, Andrés y Josefina, por demostrarme día tras día que la felicidad es una fuente de energía inagotable.

Por mostrarme que los tíos y primos pueden convertirse en segundos padres y hermanos, le doy las gracias a Jose, Maribel, Jose Luis, David, Sandra e Isabel.

No puedo olvidarme de mis amigos. Gracias a cada uno de ellos por compartir conmigo el camino y por conseguir hacerlo mucho más ameno.

Por último, pero no menos importante, agradecer a David Vallejo su confianza y su inestimable ayuda a lo largo de este proyecto.

Juanma

El éxito es aprender a ir de fracaso en fracaso sin desesperarse.

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

GNU GNU is Not Unix

TFG Trabajo Fin de Grado

UAV Unmanned Aerial Vehicle

RPV Remotely Piloted Vehicle

UAS Unmanned Aircraft System

GPS Global Positioning System

DFCS Digital Flight Control System

RPA Remotely Piloted Aircraft

RPAS Remotely Piloted Aircraft System

RPS Remote Pilot Station

UA Unmanned Aircraft

FPV First Person View

VLOS Visual Line of Sight

GCS Ground Control Station

RTL Return To Launch

API Application Programming Interface

HMM Hidden Markov Models

IAD Inteligencia Artificial Distribuida

IA Inteligencia Artificial

SMA Sistemas Multiagente

ByR Búsqueda y Rescate

ATLANTE Avión Táctico de Largo Alcance No Tripulado Español

RAF Robot Aéreo Fotogramétrico

VTOL Vertical Take-Off and Landing

LTS Long Term Support

SITL Software in the Loop

SDK Software Development Kit

XP eXtreme Programming

GSM Global System for Mobile communications

HTTP Hypertext Transfer Protocol

ICE Internet Comunication Engine

LNA Ley de Navegación Aérea

AIR Artificial Intelligence and Representation

Capítulo 1

Introducción

Cada vez es más habitual que se produzcan **cataclismos atmosféricos** [des15] que puedan provocar la pérdida de vidas, como incendios, terremotos, inundaciones o tsunamis. Además, a estas catástrofes naturales se le añaden los **factores de riesgo provocados por los humanos** durante el ejercicio de actividades industriales [des15], como puede ser un accidente nuclear.

Los momentos inmediatamente posteriores a la ocurrencia de estas catástrofes son extremadamente críticos para minimizar el daño causado. Normalmente, existe un protocolo de actuación, bien definido, basado en la coordinación de personal especializado y la gestión de recursos, para responder de manera eficiente cuando tiene lugar una catástrofe. Por ejemplo, en el caso de un terremoto resulta esencial la intervención inmediata de un equipo de rescate para maximizar el número de personas rescatadas con vida. No obstante, **la actuación humana resulta muy complicada en determinadas circunstancias** debido a las propias consecuencias de la catástrofe y al riesgo real para el personal responsable de gestionar la crisis.

Este tipo de entornos son ideales para **emplear vehículos aéreos no tripulados (UAVs)**, más comúnmente llamados drones, ya que en este contexto la tecnología puede contribuir a mejorar dicha gestión gracias al soporte de drones autónomos, que sirvan como primera línea de actuación para recabar información, en escenarios poco accesibles, de una forma rápida y lo más precisa posible. Si además esta intervención se realiza de manera coordinada, las probabilidades de minimizar los daños de la catástrofe sufrida crecen considerablemente.

Los drones, generalmente, exhiben una **capacidad de respuesta en tiempo record** y continuamente están aportando nuevas oportunidades para el apoyo, que mediante el uso de otros recursos sería inviable o poco efectivo. En cuanto al empleo de cámaras, los drones son capaces de trabajar con termografía, visión nocturna, multiespectrales, réflex, etc.

En la actualidad, existen sistemas de drones capaces de transportar salvavidas desde la playa hasta varios kilómetros mar adentro [Gon16], y que confirman que es plausible una **asistencia más rápida** que una moto de agua.

El ayuntamiento de Madrid se ha convertido en el primer consistorio en proporcionar un UAV cuyo uso se extiende al Cuerpo de Bomberos, Samur y Policía Municipal. Se enmarca en el proyecto piloto SOS Drone [Pla13] y es capaz de **actuar en condiciones adversas** como altas temperaturas, humos, contaminantes y tóxicos. El dron fue utilizado por primera vez en septiembre de 2013 en la Puerta de Alcalá durante la elección de la sede de los Juegos Olímpicos de 2020.

En este TFG se propone la creación de un **sistema que permita un despliegue coordinado** de varios UAVs para conseguir información útil sobre el entorno damnificado por el desastre. A través del equipamiento de cámaras, en drones, se pueden llevar a cabo misiones de búsqueda y rescate, maximizando las posibilidades de éxito de la misión iniciada y reduciendo ostensiblemente el coste operacional.

1.1 Contexto

El contexto del TFG está enfocado a la **utilización y coordinación de UAVs en situaciones de emergencias**, por ejemplo desastres naturales o provocados por el ser humano. Los científicos han confirmado que estos sucesos han **aumentado de manera preocupante** en las últimas décadas [Gut09].

Para constatar este hecho, en la Figura 1.1 se observa la evolución en la cantidad de desastres naturales identificados desde el año 1900 hasta el año 2015. En esta gráfica se advierte un incremento mayúsculo de este tipo de catástrofes en los últimos 30 años.

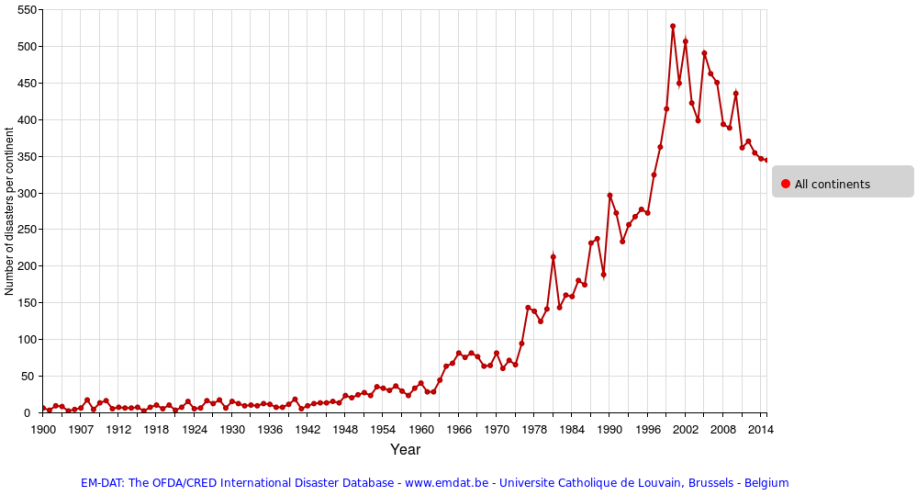


Figura 1.1: Número de desastres naturales registrados en el periodo 1900-2015 [1]

[1] http://www.emdat.be/disaster_trends/index.html

Con el auge existente en torno a los drones, en la sociedad actual, **se abre todo un mundo de posibilidades para la creación de sistemas** que sean útiles para la humanidad. Según Sophic Capital (ver Figura 1.2), los ingresos totales de las compañías que trabajan con UAVs pueden ascender hasta más de 12.000 millones de dólares en el año 2025.

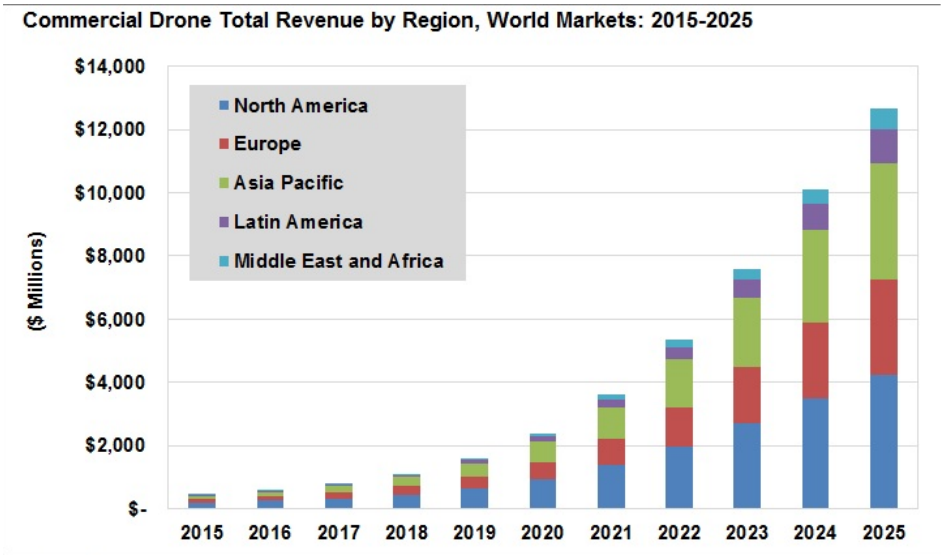


Figura 1.2: Previsión de ingresos por regiones en el periodo 2015-2025 [PV15]

Aprovechando sus ventajas, los drones constituyen una **valiosa ayuda para los cuerpos de seguridad** del Estado, los ciudadanos, y en conclusión implican un gran avance para todos, algo que se demuestra hoy en día, dado que, evitan muchas pérdidas humanas.

1.2 Motivación

Estimar el tiempo de actuación en labores de socorro es una tarea compleja, pero los drones pueden **desplegarse con rapidez** y acortar sustancialmente el tiempo de respuesta y el riesgo de lesiones de los individuos desaparecidos y de los equipos de rescate. En el caso de rescates acuáticos, según la empresa Vodafone, responsable de un proyecto piloto que utiliza drones para ayudar a rescatar a los bañistas en apuros, «un socorrista tarda el triple de tiempo que un dron necesita para llegar a un bañista en peligro» [Cob15].

Los UAVs son un **aliado perfecto en misiones de búsqueda y rescate** puesto que sus limitaciones en condiciones meteorológicas adversas son menores. Aparte de esto, el uso de sensores especializados contribuyen a equipar los UAV de ventajas importantes en la realización de este tipo de operaciones. Sirva como precedente «la utilización de helicópteros no tripulados franceses Elipse HE 300 durante la actual crisis de la central de Fukushima» [Fer12].

Este tipo de dispositivos, utilizados de manera simultánea con la intervención humana de los equipos de rescate, resultan herramientas vitales a la hora de detectar, localizar y rescatar

a víctimas de catástrofes naturales como los terremotos o incidentes como el hundimiento de minas. El auxilio de personas en esta clase de condiciones puede resultar peligroso para los miembros del servicio de respuesta rápida ante emergencias. Sin embargo, los dispositivos no tripulados pueden impedir males mayores y colaborar de manera eficaz con los humanos en condiciones complicadas.

1.3 Propuesta de sistema de vigilancia adaptativo basado en UAVs

Con todo lo anterior, en este TFG se pretende **mejorar la productividad de los equipos de rescate**, en situaciones de catástrofe, permitiendo rebajar los tiempos de reconocimiento del terreno con la ayuda de una flota de UAVs que se coordinen y comuniquen.

La propuesta de este proyecto gira en torno a un **sistema adaptativo basado en la coordinación de drones** para obtener información útil sobre el entorno afectado por la catástrofe. El esquema de referencia sería la identificación de puntos de monitorización, en base a un sistema de prioridades, con el objetivo de obtener información teniendo en cuenta la importancia de cada uno de los puntos del entorno monitorizado.

Con el uso de este sistema se pretende reducir costes en cuanto a:

- El tiempo de reconocimiento del territorio afectado por el desastre.
- El tiempo de rescate de individuos.
- La exposición al peligro por parte del personal de emergencia.

Se plantea el uso de **DroneKit y su API en Python para la programación del vehículo** aéreo *3DR IRIS+*. Por medio de esta API, los desarrolladores pueden crear aplicaciones que se ejecutan en un ordenador y que se comunican con el controlador de vuelo *ArduPilot*. *DroneKit-Python* proporciona acceso mediante programación a información de telemetría, del estado y de los parámetros del vehículo, y permite tanto la gestión de misiones como el control directo sobre el movimiento y las operaciones del vehículo.

Por otro lado, la **herramienta de simulación SITL** permite hacer uso de los drones que se consideren oportunos sin necesidad de poseer la infraestructura hardware. En resumen, SITL permite ejecutar *ArduPilot* directamente en el ordenador, sin ningún hardware especial. De este modo, el papel de SITL, en este proyecto, será el de desplegar pruebas de manera previa a los vuelos con el *3DR IRIS+* y el de llevar a cabo la coordinación de varios drones, al disponer solo de un UAV real.

Por último, a fin de obtener información en cada uno de los puntos de monitorización **se empleará la librería *OpenCV*** para la visión por computador. Gracias a *OpenCV* es posible la captura de imágenes en primera persona que seguidamente serán **analizadas mediante *Google Cloud Vision* API**, consiguiendo así la información sobre el entorno afectado por la catástrofe.

1.4 Estructura del documento

Capítulo 1: Introducción

Es el presenta capítulo. Se introduce el Trabajo Fin de Grado.

Capítulo 2: Objetivos

Se exponen los objetivos que se aspiran satisfacer una vez concluido el proyecto.

Capítulo 3: Estado de la cuestión

Se hace un repaso acerca de los antecedentes de la aviación no tripulada, se describe el estado actual de la inteligencia artificial distribuida, se habla sobre los principales sistemas de catástrofes con drones que se han creado y de las herramientas más importantes para el desarrollo de estos sistemas.

Capítulo 4: Método de trabajo

Se especifica la metodología, así como los medios hardware y software que han sido utilizados durante el proyecto.

Capítulo 5: Arquitectura

Se detalla la arquitectura propuesta para el sistema, puntualizando en sus diferentes capas y funciones.

Capítulo 6: Evolución, resultados y costes

Muestra y explica la evolución del proyecto, los resultados que se han conseguido y el coste total en euros.

Capítulo 7: Conclusiones y trabajo futuro

Se enuncian las conclusiones alcanzadas tras haber finalizado el sistema y se proponen posibles actividades a realizar en un futuro.

Capítulo 2

Objetivos

En este capítulo se enumeran los objetivos a cumplir en el proyecto. Se parte de un objetivo general, que será descompuesto en varios subobjetivos más específicos.

2.1 Objetivo general

El objetivo general del proyecto es el diseño y desarrollo de un **sistema adaptativo** que sea capaz de desplegar y coordinar varios UAVs que permitan obtener información relevante, simultáneamente analizada, sobre un entorno que ha sido afectado por algún tipo de catástrofe. Mediante el cumplimiento de este objetivo se espera:

- **Reducir la exposición de agentes humanos** del peligro derivado de la propia catástrofe.
- **Disminuir el tiempo** empleado en recolectar información acerca del territorio afectado por la catástrofe para ayudar al equipo de rescate.
- **Minimizar el daño** resultante global.

El **atributo adaptativo** responde a la capacidad del sistema de solicitar dispositivos de monitorización bajo demanda cuando así sea necesario. De este modo, se construirá una arquitectura adaptativa que posibilite el funcionamiento del sistema independientemente del número de vehículos con el que se cuente.

2.2 Objetivos específicos

Para lograr el objetivo general, es necesario especificar los objetivos específicos que lo componen y que se describen a continuación:

- **Especificar puntos de monitorización**: el primero de los objetivos específicos, cubre la necesidad de identificar los puntos de localización a los que los drones deben acudir. Estas localizaciones serán especificadas en función a un sistema de prioridades, que tendrá como objetivo reunir información, en base a la importancia, de cada uno de los puntos del entorno que han sido especificados. Por ejemplo, un equipo de emergencias considerará más importante acudir a un lugar donde exista una mayor concentración

de personas. Es decir, el sistema debe estar preparado para recibir información de una serie de puntos de interés y acudir a ellos en el orden determinado por su prioridad.

- **Escalabilidad**: el sistema debe estar listo para responder a la demanda de dispositivos de monitorización cuando sea necesario. Será capaz de desplegar los drones que son solicitados por los equipos de emergencia, siempre y cuando se disponga en la flota de ese número de vehículos. Es evidente, que varios drones son capaces de cubrir más terreno de una manera más eficiente y en un tiempo menor, por lo que el sistema debe estar diseñado para soportar el despliegue de varios drones.
- **Plantear la coordinación**: de nada sirve el empleo de varios drones, si estos no realizan acciones que sean coordinadas. Si todos los drones vuelan hacia la misma localización o no se les asignan nuevos puntos de interés, después de haber acudido a uno, el sistema sería totalmente ineficaz. En otras palabras, los UAVs deben volar a puntos de monitorización que no se hayan asignado a otros. Además, estos puntos deben ser asignados a vehículos en el momento que hayan terminado el vuelo que tenía como destino otro punto de monitorización.
- **Proporcionar capacidad de análisis forense**: el sistema debe disponer de las herramientas necesarias para permitir la grabación y el visionado, en directo, del entorno que ha sido dañado por alguna clase de desastre. Una visión aérea de la zona afectada por la catástrofe, puede ofrecer grandes ventajas a los sistemas de emergencia, como por ejemplo, la identificación de supervivientes. Además, gracias a la obtención de imágenes se puede llevar a cabo una toma de decisiones más correcta, ya que se dispone de información trascendental acerca del estado del terreno.
- **Analizar imágenes**: es importante implementar algún tipo de análisis que permita realizar un estudio de las imágenes que se obtienen en directo. Este análisis debe ayudar a los equipos de emergencia a reconocer ciertas características del entorno, como puede ser la identificación de agua que bloquee los accesos a la zona afectada. En una situación de desastre es fundamental contar con toda la información posible del terreno, y este tipo de análisis con respecto a las imágenes grabadas puede brindar la oportunidad de obtener el máximo conocimiento posible sobre área perjudicada.

Capítulo 3

Estado de la cuestión

En este capítulo se lleva a cabo una prospección de los sistemas y artículos que se han efectuado hasta la fecha, de modo que se contextualice el TFG en el ámbito de la utilización de UAVs, en diferentes entornos de la ingeniería civil en general y, más concretamente, en situaciones de emergencia. Asimismo se analizarán las tecnologías de mayor relevancia para el desarrollo e implementación del sistema adaptativo.

3.1 Vehículo aéreo no tripulado

La aviación no tripulada comprende una amplia gama de aeronaves. El origen de estas aeronaves no tripuladas reside en la creación de los torpedos aéreos, que después evolucionaron a través de las bombas guiadas, los blancos aéreos (o drones), los señuelos, los modelos deportivos de radiocontrol, las aeronaves de investigación, las aeronaves de reconocimiento y las aeronaves de combate.

Un UAV es «un vehículo aéreo motorizado **i)** que no lleva a bordo a un operador humano, **ii)** que utiliza las fuerzas aerodinámicas para proporcionar la elevación del vehículo, **iii)** que puede volar autónomamente o ser tripulado por control remoto, **iv)** que puede ser sustituido o recuperado, y **v)** que puede transportar una carga letal o no» [UAV05]. Dada esta definición quedan excluidos:

- Los planeadores, debido a que no usan una planta propulsora.
- Los globos y dirigibles, ya que no se elevan mediante fuerzas aerodinámicas sino mediante fuerzas de flotabilidad.
- Los misiles balísticos, misiles de crucero y proyectiles de artillería.

La palabra UAV se comienza a hacer frecuente en los años 90 para representar a las aeronaves robóticas y así, reemplazar el término vehículo aéreo pilotado remotamente o RPV. UAV y RPV no son más que dos nombres entre aproximadamente la docena que han recibido las aeronaves robóticas no tripuladas a lo largo de la historia (ver Figura 3.1).

«En el año 2011 la Organización de Aviación Civil Internacional, organismo especializado de las Naciones Unidas para la aviación civil y del cual España forma parte al haber suscrito el Convenio de Chicago de 1944, publicó su Circular 328 en la cual por vez primera

reconoce a las aeronaves no tripuladas como aeronaves, con todo lo que ello trae consigo, y de entre todas las posibles tipologías escoge a las que se pilotan de manera remota para ser consideradas como aptas para la aviación civil» [Cue15]. Así es como se acuñan los términos que a continuación se detallan, y que en la actualidad tienen una validez internacional y prácticamente única en todos los ámbitos. Estos términos son:

- **Aeronave pilotada remotamente** o **RPA**: aeronave en la que el piloto al mando no está a bordo.
- **Sistema de aeronave pilotada remotamente** o **RPAS**: conjunto de elementos configurables formado por un RPA, una estación de pilotaje remoto asociada o RPS, un sistema de enlace de mando y cualquier otro elemento requerido en cualquier punto durante la operación del vuelo.

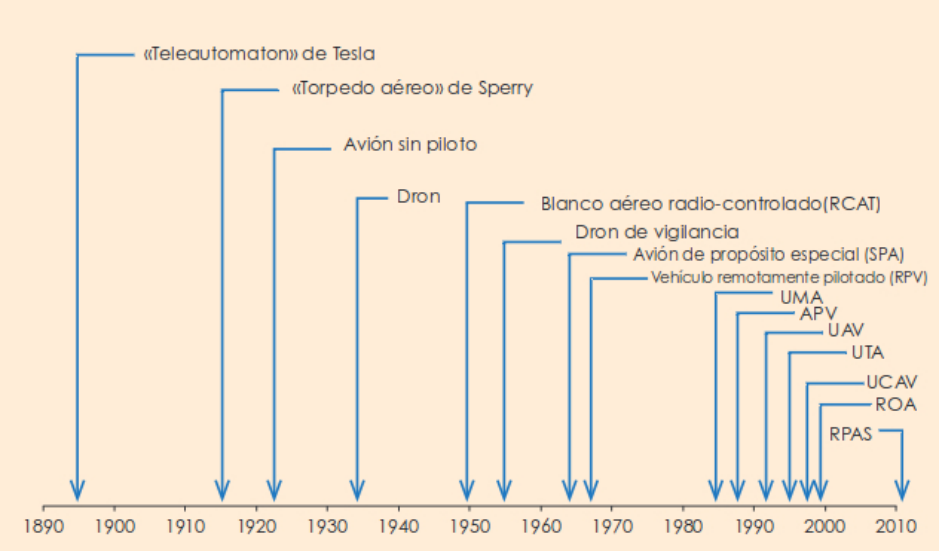


Figura 3.1: Cronología de los nombres aplicados a las aeronaves robóticas [1]

3.1.1 Nacimiento y desarrollo de aeronaves pilotadas por control remoto

Los europeos fueron pioneros en concebir los principios de la aeronáutica y, al tratar de emplearlos en aeronaves, volaron modelos no tripulados que pueden ser los primeros vehículos aéreos no tripulados de la historia. Precursores de la aviación en distintos países siguieron una progresión que va de los planeadores a los aviones propulsados no tripulados, y de los vuelos no tripulados a los tripulados (ver Cuadro 3.1). Su limitación tecnológica, no disponer de un motor con suficiente relación potencia-peso, impidió que sus diseños pudieran mantenerse en el aire.

[1] Los acrónimos no especificados se corresponden con: UMA = Unmanned Aircraft, APV = Automatically Piloted Vehicle, UTA = Unmanned Tactical Aircraft, UCAV = Unmanned Combat Air Vehicle, ROA = Remotely Operated Aircraft.

País	Planeador no tripulado	Planeador tripulado	Avión no tripulado	Avión tripulado
Inglaterra	Cayley, 1809	Cayley, 1849	Cody, 1907	Cody, 1908
Francia		Ferber, 1901	Du Temple, 1857	Santos-Dumont, 1906
Alemania		Lilienthal, 1891		
Japón		Leprieur/Aibara, 1909	Ninomiya, 1891	Nagahara, 1911
Rusia				Rossinsky, 1910
Estados Unidos		Chanute, 1896	Langley, 1896	Hnos. Wright, 1903

Cuadro 3.1: Primeros vuelos conocidos en diversos países [Cue15]

Durante la **Primera Guerra Mundial**, la aviación no tripulada se veía entorpecida por la falta de progreso tecnológico. Los obstáculos se encontraban en los problemas de estabilización automática, control remoto y navegación autónoma. Fue Elmer Sperry el primero en resolver estos inconvenientes en una aeronave no tripulada. Elmer Sperry creó un giroestabilizador para un avión en **1909**, que tenía un rendimiento mediocre y era demasiado pesado. Ayudado por Glenn Hammond Curtiss optimizó su invento, que ahora era más pequeño y permitía controlar el avión en los tres ejes.

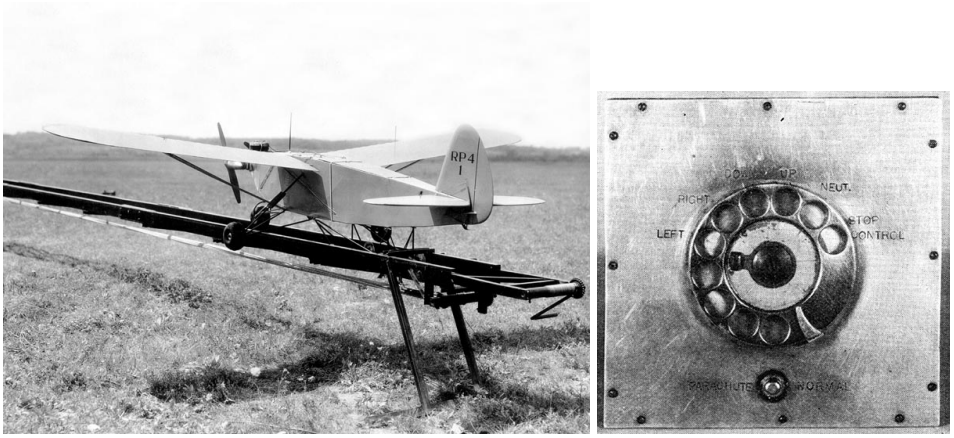
En **1916** se realiza la primera demostración del mecanismo de Sperry para guiar un avión convencional, el Hewitt-Sperry Automatic Airplane. Para efectuar la prueba, el aviador debía levantar el vuelo antes de activar el piloto automático. Después el avión volaba una ruta previamente programada y picaba. El piloto recuperaba el control de la aeronave en dicho momento y retornaba al aeródromo.

Finalmente, tras dos años de intentos fallidos, el torpedo aéreo de Sperry (ver Figura 3.2) hizo un vuelo de 900 metros, el **6 de marzo de 1918**, siendo el primer viaje exitoso de un avión no tripulado en la historia. Su método de guiado para llegar al objetivo era primitivo pero ingenioso. Una vez que se conocía el viento y la distancia al objetivo, se calculaban las revoluciones necesarias para acertar en el blanco. Una vez alcanzadas las revoluciones precisas, se separaban las alas del fuselaje, dejando caer el torpedo sobre el objetivo.

Los primeros sistemas fueron creados como armamento de largo alcance en artefactos tales como el torpedo aéreo de Sperry, mencionado anteriormente, y el blanco aéreo británico Aerial Target. El Aerial Target era un monoplano, sin piloto, controlado por radio que sirvió para verificar la viabilidad de utilizar señales de radio como sistema de guiado.

En el transcurso de la **Segunda Guerra Mundial**, Gran Bretaña abandonó la elaboración de misiles de crucero y se introdujo en el sector de los blancos aéreos con control completo por radio, a pesar de que el alcance era muy limitado. En paralelo en Estados Unidos se fabricó el RP-4 (ver Figura 3.3) de Radioplane Company. A través de estos aviones, se fue perfeccionando la tecnología y el uso del control remoto por radio.

Figura 3.2: Torpedo aéreo de Sperry



(a) RP-4 listo para volar (b) Controlador de vuelo

Figura 3.3: Sistema RP-4

En la **Posguerra** de la **Segunda Guerra Mundial**, la compañía Northrop produjo una serie de blancos aéreos no tripulados, llamados Falconer (ver Figura 3.4), que llevaban integrados sistemas de radiocontrol más modernos. Otra aplicación relevante durante la **Posguerra** fue la de señuelos antirradar (ver Figura 3.4). Estos eran soltados desde bombarderos, donde eran controlados por radio con la ayuda de imágenes de vídeo, con la finalidad de confundir a los sistemas radar enemigos.

(a) Northrop Falconer

(b) Señuelo antirradar Northrop Crossbow

Figura 3.4: Aeronaves no tripuladas de la compañía Northrop

Con la aparición de aviones militares con sistemas de propulsión a reacción, a lo largo de **1960** se fabricaron blancos aéreos más rápidos y con mayor alcance (ver Figura 3.5). De esta manera, eran más difíciles de detectar y de derribar que los aviones de reconocimiento tripulados y, adicionalmente, no provocarían incidentes diplomáticos asociados con la captura de un piloto. Posteriormente estos UAVs fueron equipados con cámaras para misiones de reconocimiento, revelando las fotos en la base cuando el UAV regresaba. Algunos incluso incorporaron buscadores de radiación y cabezas de combate antirradar.

Figura 3.5: Ryan Firebee

Los **años 70** iban a ser testigos de la inclusión de UAS en misiones de reconocimiento y vigilancia tanto de corto alcance, como de largo alcance y elevada altitud. Fueron sistemas más complejos tanto en los requisitos de la misión como en la seguridad en las comunicaciones.

En **1980** se creó un sistema en el que el UAV se lanzaba desde una rampa (ver Figura 3.6) y se rescataba con un paracaídas y un airbag. Cuando se realizaban análisis diurnos el UAV estaba equipado con una cámara convencional más una cámara de infrarrojos y para los análisis nocturnos únicamente se equipaba la cámara infrarroja. El guiado se llevaba a cabo mediante la preprogramación de un autopiloto basándose en giróscopos verticales y direccionales, información de altímetros y anemómetros barométricos. Además, transportaba un transmisor de vídeo que podía enviar imágenes, en tiempo real, a la estación de control en tierra, estando a 70 km de la base.

Figura 3.6: Lanzamiento del Canadair CL-89

En los **años 90** con la mayor disponibilidad del GPS y de las comunicaciones vía satélite se consigue que los UAS puedan operar por encima de la señal de radio y que puedan abandonar el uso de sistemas de navegación imprecisos basados en giróscopos y datos de aire. Es así como, junto con los sistemas digitales de control de vuelo o DFCS, el alcance y la precisión de la navegación mejoraron apreciablemente. Como resultado se desarrollaron sistemas de medio y largo alcance.

En la **década de los 2000** se produce un incremento en el uso militar de los sistemas no tripulados. Por otro lado, las posibles aplicaciones civiles no han fructificado debido a la dificultad a la hora de asegurar la separación entre las aeronaves tripuladas y no tripuladas. Otro paso adelante en esta década fue la instalación de armamento en UAVs para una respuesta inmediata ante el posible descubrimiento de fuerzas enemigas. Otras líneas de avance, en

este periodo de tiempo, son las que tratan de aumentar la automatización, reduciendo así la carga de trabajo y los errores de las tripulaciones en tierra.

Más allá del 2010, el mercado militar de los RPAS continúa con una tendencia positiva desde el final de la Guerra Fría y se espera que se acelere en las primeras décadas del siglo XXI. La tendencia comercial en el mercado de la robótica está también en constante crecimiento (ver Figura 3.7). Como consecuencia de esta corriente, la tecnología está apoyando estas tendencias con microprocesadores cada vez más baratos y competentes que incentiven su utilización.

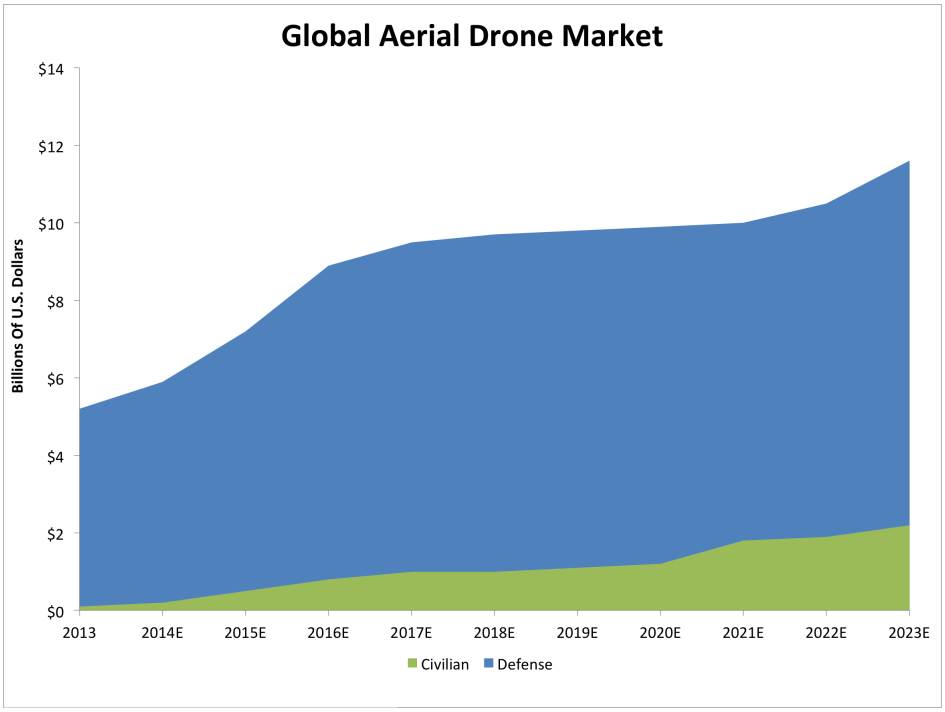


Figura 3.7: Estimación del mercado de drones en el periodo 2013-2023 [Bal14]

3.1.2 Tipos de aeronaves pilotadas por control remoto

En conformidad con la Organización de Aviación Civil Internacional, «El hecho de que la aeronave sea tripulada o no tripulada no afecta su condición de aeronave. Cada categoría de aeronave tendrá posiblemente versiones no tripuladas en el futuro. Este punto es fundamental para todos los aspectos futuros relativos a las UA y proporciona la base para tratar la aeronavegabilidad, el otorgamiento de licencias al personal, las normas de separación, etc.» [dACI11]. En otras palabras, las aeronaves no tripuladas son, en primer lugar, aeronaves, por lo que están sujetas a las mismas normas y restricciones que las aeronaves tripuladas.

Existen múltiples formas de clasificar las aeronaves, pero lo habitual es utilizar una clasificación atendiendo al modo en la que las aeronaves consiguen su sustentación en la atmósfera. A continuación se propone una clasificación en la que se presenta los principales tipos de aeronaves:

- **Aerostato**: Más ligeros que el aire.
  - Globo aerostático.
  - Dirigible.
- **Aerodino**: Más pesados que el aire.
  - **Ala fija**.
    - Avión.
    - Planeador.
    - Ala delta.
    - Parapente.
    - Paramotor.
  - **Ala rotatoria**.
    - Helicóptero.
    - Multirrotor.
    - Autogiro.

Por otra parte, aparecen aeronaves de nuevas clases, como los híbridos, que ejecutan parte del vuelo haciendo uso del método de ala rotatoria, generalmente en el despegue y aterrizaje, realizando un salto al uso de ala fija para dirigirse de forma rápida y eficiente a su destino.

El uso de vehículos aéreos no tripulados ha alcanzado un grado de madurez significativo en el terreno militar. Tanto es así, que en el ejército norteamericano constituyen cerca de un tercio de la flota de aeronaves, desempeñando en exclusiva las misiones de inteligencia, vigilancia y reconocimiento que deben cumplir las fuerzas armadas. «De acuerdo con la consultora de defensa Teal Group Corporation [2], el inventario en 2013 de sistemas aéreos no tripulados contaba con más de 10.000 UAS (ver Figura 3.8) [...], de lo que se deduce que en el ámbito militar la preponderancia de los sistemas de ala fija (ver Figura 3.9) es casi absoluta (97 %)» [Oñ15].

En el ámbito civil, mucho menos avanzado que el militar, la situación es la contraria. De acuerdo con la información suministrada por la Agencia Estatal de Seguridad Aérea [3] (AESA) referida a las autorizaciones otorgadas hasta julio de 2016 [4], el número de operadores habilitados en España es de 1.521. Estos operadores han registrado 2.931 aeronaves, de las cuales 88 son de ala fija y el resto de ala rotatoria. Por lo que los sistemas basados en aeronaves de ala rotatoria superan ampliamente a los de otros tipos.

[2] http://www.tealgroup.com

[3] http://www.seguridadaerea.gob.es/

[4] http://www.seguridadaerea.gob.es/media/4305572/listado_operadores.pdf

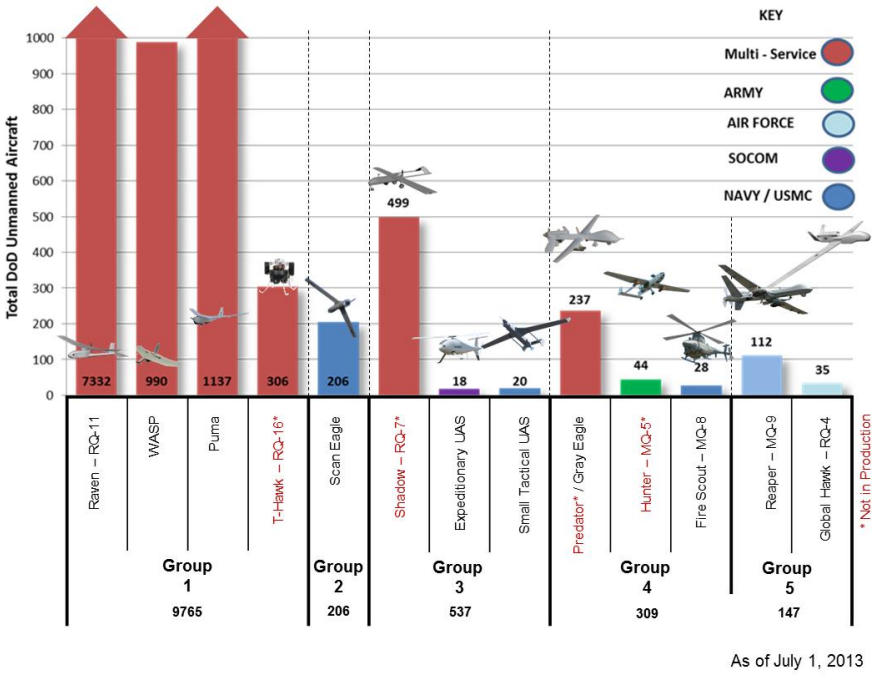


Figura 3.8: UAVs del Departamento de Defensa de EE.UU. en el 2013 [CHC14]

Figura 3.9: Sistema de ala fija

Esta inclinación, hacia las aeronaves multirrotor (ver Figura 3.10) en el entorno civil, se debe a que son altamente apropiadas para la principal actividad desempeñada en estos momentos, la toma de imágenes y vídeos, que según las estimaciones de la Asociación Española de RPAS [5] constituyen posiblemente más del 90 % de las tareas ejecutadas por drones [Oñ14], al igual que ocurre en otros países europeos.

Figura 3.10: Sistema multirrotor

Las principales ventajas de los multirrotores son las siguientes:

- **Despegue y aterrizaje vertical**: que minimiza las necesidades del espacio requerido en tierra para su puesta en marcha .
- La posibilidad de **volar a punto fijo** o **a muy baja velocidad**: lo cual resulta idóneo para software en la que la observación sea un requisito vital.
- Mayor **maniobrabilidad** y **precisión** de vuelo: mientras que los sistemas de ala fija siguen rutas curvilíneas, los multirrotores son capaces de volar siguiendo cualquier trayectoria deseada en las tres dimensiones.
- Su diseño les permite acarrear **cargas más voluminosas**, en relación con su propio tamaño, que los aviones.

Por el contrario, los multirrotores tienen una serie de desventajas frente a los sistemas de ala fija:

- Son **menos eficientes**: lo que provoca que aunque tenga el mismo tamaño, la autonomía sea menor.

[5] `http://www.aerpas.es/`

- Vuelan a **menor velocidad**: lo que combinado con su menor autonomía significa que pueden cubrir un área mucho menor.
- Tienen una **huella sonora sensiblemente mayor**, por lo que resultan poco indicados para operaciones de vigilancia.

A la vista de lo expuesto anteriormente se justifica que los multirrotores dominen hoy en día el mercado, «si bien es previsible que en el futuro, a medida que se desarrollen aplicaciones de ejecución más complejas, cubriendo mayores distancias y desarrolladas a mayor altura sobre el terreno, al igual que ocurre en el caso militar, los sistemas de ala fija aumentarán su peso» [Oñ15].

3.1.3 Modos de operación en el guiado de una aeronave no tripulada

Sólo existen cuatro modos de operación en cuanto a la manera de guiar una aeronave de forma remota, con un grado de automatización creciente:

- **Modo manual**: el piloto remoto actúa sobre las superficies de control y la potencia de los motores, a través de una emisora de radiocontrol.
- **Modo asistido**: similar al modo manual, pero el piloto remoto no interviene directamente sobre las superficies de control o los motores, sino que indica sus intenciones (girar a la derecha, subir, etc.), desde el puesto de radiocontrol, y el autopiloto las transforma en actuaciones que logren ese propósito.
- **Modo automático**: el piloto remoto traza un plan de vuelo, es decir, un número de puntos de paso o «waypoints» antes del inicio del vuelo. La aeronave dispone de un piloto automático que sigue el plan previsto, realizando automáticamente las acciones requeridas en cada momento. Aun así el piloto mantiene el control en todo momento, pudiendo modificar los puntos de paso durante el vuelo, ejecutar maniobras predeterminadas (como por ejemplo la «vuelta a casa») o incluso tomar el control directamente.
- **Modo autónomo**: es semejante al modo automático, en cuanto que se establece un plan de vuelo con anterioridad, pero una vez iniciado el vuelo se continúa con el plan de forma completamente autónoma, sin precisar la intervención del piloto incluso en caso de producirse situaciones de emergencia.

Es obvio que el modo manual y el modo asistido requieren que la aeronave se encuentre a la vista del piloto o, que al menos, transmita información suficiente para que el piloto posea el conocimiento necesario de la situación y del escenario como para poder tomar las decisiones adecuadas. Por ello, el uso de estos modos suele estar restringido a las circunstancias de vuelo en línea de vista visual o VLOS.

Se destaca el hecho de que el modo manual se utiliza normalmente en las aeronaves de ala fija. Las de ala rotatoria, sobre todo los multirrotores, suelen utilizar el modo asistido, por la

dificultad que implica el coordinar todas las acciones para mantener la aeronave estabilizada y ejecutar las maniobras deseadas. En ambos casos es necesaria una considerable destreza por parte del piloto para controlar la aeronave desde tierra.

Por esta razón existe una tendencia a utilizar de forma exclusiva RPAS en modo automático, o por lo menos en una forma de modo asistido en la que el piloto recibe una imagen tomada por una cámara dirigida hacia adelante, denominada visión en primera persona o FPV, lo que le permite actuar como si estuviera embarcado en ella. Este modo resulta también muy indicado en vuelos en línea de vista para misiones rutinarias. La principal ventaja que proporciona el modo automático es la posibilidad de utilizar pilotos de menor capacitación y, por lo tanto, de reducir el coste de operación.

3.1.4 Visión por computador en UAVs

«El objetivo de la **visión por computador** es modelar y automatizar el proceso de reconocimiento visual, un término que se interpreta en líneas generales como "percibir distinciones entre objetos con diferencias importantes entre ellos". La visión por computador utiliza métodos estadísticos para separar los datos, haciendo uso de modelos que han sido construidos con la ayuda de la geometría, la física y la teoría del aprendizaje. De este modo la visión por computador se basa en un sólido conocimiento de las cámaras y del proceso físico para la formación de imágenes» [FP12].

Los sistemas de visión artificial están formados por:

- Una **fuente de luz**.
- Un **sensor de imagen** que capture las imágenes que después se procesan.
- Una **tarjeta de adquisición de imágenes**, como interfaz entre sensor y el computador.
- Una serie de **algoritmos de análisis** en los que se procesan, segmentan y se aplican las transformaciones necesarias para obtener los resultados deseados.
- Un **procesador** que analice las imágenes recibidas ejecutando los algoritmos diseñados.
- Un **sistema de respuesta** a tiempo real que ejecute los comandos.

El ser humano percibe la luz gracias a los ojos y la información captada se traslada al cerebro por el nervio óptico. El cerebro procesa la imagen, analiza la escena y actúa en consecuencia. Los sistemas de visión por computador buscan imitar este comportamiento, para lo cual, se definen cuatro fases (ver Figura 3.11), estas fases son:

- **Captura**: se adquieren las imágenes.
- **Procesamiento**: se aplican filtros o transformaciones que eliminan partes no deseadas, facilitando las próximas etapas.

- **Segmentación**: se abstraen los elementos que interesan del resto de la imagen para su comprensión.
- **Reconocimiento**: se distinguen los objetos segmentados haciéndose uso de una serie de características que previamente se han establecido.

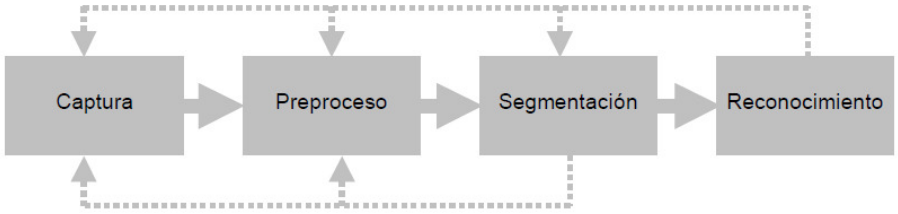


Figura 3.11: Fases de un sistema de visión por computador [FP12]

En el campo de los sistemas aéreos no tripulados, la visión por computador es una parte importante para la navegación y el control, y en ella se basan numerosas aplicaciones. Los UAVs aparecen como una alternativa barata y adecuada en muchos campos de aplicación, dadas sus capacidades de vuelo y a la posibilidad de integrar sistemas de visión que permiten mejorar determinadas tareas como el pilotaje, el guiado autónomo del vehículo o la obtención de imágenes.

En la actualidad, se han desarrollado aplicaciones con vehículos aéreos no tripulados, que hacen uso de la visión por computador, en distintos ámbitos de aplicación como: la gestión del tráfico [Pur08], la detección de incendios [KVT04] o la evitación de obstáculos [GSC+06].

3.1.5 Aplicación de UAVs en el sector civil

Los drones pueden ser calificados como robots no antropomorfos con una enorme autonomía de vuelo y con un abanico de posibilidades de aplicación.

«Gracias al uso de procesadores electrónicos, de software especializado y del Sistema de Posicionamiento Global (GPS), las capacidades de control automático en base a la información procedente de los sensores instalados en las aeronaves, y la rápida reacción correctiva como consecuencia del procesamiento local de la información del estado de vuelo, hacen posible que la retroalimentación aporte un inmenso potencial, minimizando para estos artefactos las posibles desviaciones entre el comportamiento real y el esperado» [Ló15].

Actualmente, haciendo caso omiso a los usos bélicos, estos equipos brindan **amplias oportunidades** de aplicación **al sector de la ingeniería civil** (ver Cuadro 3.2): investigación atmosférica, filmación de películas y fotografía deportiva, gestión de riegos y desastres naturales, inspecciones de infraestructuras, levantamientos topográficos, agricultura de precisión, control de caza, localización de bancos de pesca, mantenimiento de parque eó-

licos e infraestructuras energéticas (ver Figura 3.12), control medioambiental, exploración geológico-minera, etc.

Figura 3.12: Revisión de líneas eléctricas con drones en Ciudad Real [Chi16]

En los países más avanzados se está fomentando el uso de estos aparatos en diferentes proyectos de investigación [UA14], que abarcan desde el ensayo de material aeronáutico en condiciones peligrosas, pasando por el reparto de cargas, hasta el desarrollo de tecnologías, como las pilas de combustible de hidrógeno, que triplicarían la duración de los vuelos.

Fecha	Actividad	Lugar	Usuario	Operador
Enero 2014	Cartografía	Suiza	Pix4D	ATyges
Enero 2014	Vigilancia de fronteras	California	Homeland Security Department	
Enero 2014	Formación	Madrid		CB Group Universidad Politécnica
Enero 2014	Protección Civil	Extremadura	Junta de Extremadura	ATyges
Enero 2014	Experimentación y ensayos	Villacarrillo	Centro ATLAS	Boeing
Diciembre 2013	Seguridad ciudadana	Madrid	Ayuntamiento de Madrid	Policía Local
Diciembre 2013	Entrega de paquetería	EE.UU.	Amazon	Amazon Prime Air
Noviembre 2013	Apoyo a ciclo de vida	Francia	Ejército de Tierra	ADS
Noviembre 2013	Revisión de infraestructura	Viaducto de Roquemaure	SNCF Francia	Diades, Red Bird, Azur Drones
Noviembre 2013	Pruebas de vuelo e infraestructura	Villacarrillo	Centro ATLAS	FADA - CATEC
Octubre 2013	Inspección de cultivos	Jerez de la Frontera	Proyecto Fieldcopter	
Octubre 2013	Prevención de incendios		Junta de Andalucía	ELIMCO
Octubre 2013	Asistencia sanitaria	Alemania	Definetz	
Septiembre 2013	Publicidad	Madrid	Candidatura Madrid 2020	
Septiembre 2013	Lucha contra incendios	California	Guardia Nacional	Guardia Nacional
Septiembre 2013	Investigación medioambiental	Antártida	NASA	Universidad de Colorado

Cuadro 3.2: Principales actividades civiles desde septiembre de 2013 [CHC14]

3.2 Inteligencia Artificial Distribuida

La evolución de los procesadores multinúcleo y de las redes informáticas combinadas con que muchas actividades, incluyendo la resolución de problemas de las personas, involucran a grupos de gente condujo al desarrollo de un nuevo campo de investigación: la **Inteligencia Artificial Distribuida** (IAD). Fusiona la Inteligencia Artificial (IA) con la Computación Distribuida y es definida como «un subcampo de la IA que se centra en los comportamientos inteligentes colectivos que son producto de la cooperación de diversas entidades» [AG93].

La IAD aparece en la década de los 80 para estudiar los sistemas inteligentes, que están formados por un conjunto de varios agentes, e intenta resolver problemas donde una conducta colectiva es más eficiente que una conducta individual. Según Iglesias [Igl98], a pesar de ser una disciplina joven, podemos distinguir tres periodos cronológicos bien diferenciados:

- **IAD clásica**: centrada en el estudio de la conducta colectiva, en oposición a la IA, que estudia la conducta individual.
- **IAD autónoma**: focalizada en el estudio de los agentes individuales situados en un mundo social.
- **IAD comercial**: centrada en la aplicación de la IAD clásica y autónoma, desarrollando agentes con características muy diferenciadas que están siendo explotados de forma comercial.

El campo de investigación de la IAD desencadenó en campos más específicos: la Inteligencia Artificial Paralela, la Resolución Distribuida de Problemas y los **Sistemas Multiagente** (SMA). El nacimiento de estos SMA implicó la irrupción de un nuevo paradigma en el desarrollo del software que no solo afectó en las fases de conceptualización, diseño e implementación, sino también en la aplicabilidad de las soluciones propuestas.

3.2.1 Sistemas Multiagente

Cada agente «es un sistema computacional que está situado en algún entorno, y que es capaz de actuar autónomamente en dicho entorno con el fin de cumplir sus objetivos» [WJ95]. En esta definición debe remarcarse que:

- No se refiere a agente inteligente, solo a agente.
- No habla sobre el tipo de entorno al que se refiere.
- No se define el término autonomía.

Sobre la autonomía, aunque no sea un concepto simple a la hora de definirlo, se puede decir que se refiere a que «los agentes son capaces de decidir por sí mismos que tienen que hacer con el fin de satisfacer sus objetivos de diseño» [Woo01].

No se acostumbra a pensar en un termostato o un proceso Unix como agentes, y desde luego no como **agentes inteligentes**. Para considerar a estos agentes como inteligentes deben cumplir una serie de capacidades propuestas por Wooldridge y Jennings en 1995 [WJ95]:

- **Reactividad**: Los agentes inteligentes son capaces de percibir su entorno y responder de manera oportuna a los cambios que se producen en él con el fin de satisfacer sus objetivos de diseño.
- **Proactividad**: Los agentes inteligentes son capaces de mostrar un comportamiento dirigido por sus objetivos tomando la iniciativa con el fin de satisfacerlos.
- **Habilidad social**: Los agentes inteligentes son capaces de interactuar con otros agentes (e incluso con humanos) con el fin de satisfacer sus objetivos de diseño.

Sin embargo, muchas aplicaciones requieren la interacción entre diversos individuos para poder alcanzar un determinado objetivo. Estos sistemas, compuestos de varios agentes, son denominados **Sistemas Multiagente**.

Un SMA puede ser definido formalmente como «una organización o sociedad de agentes, donde un conjunto de agentes interactúan y cooperan entre sí para alcanzar algún objetivo colectivo» [FGM04]. Los SMA son un subcampo de la informática relativamente nuevo, estudiado desde 1980 y con un amplio reconocimiento desde aproximadamente mediados de la década de los 90. Desde entonces, el interés internacional ha crecido enormemente. Este rápido crecimiento ha sido estimulado, en parte, por la creencia de que los agentes son un paradigma de software adecuado a través del cual se pueden explotar las posibilidades presentadas por los sistemas distribuidos masivos.

Cuando los problemas que se quieren solventar incorporan peculiaridades como: gran tamaño, alto grado de incertidumbre, dinamismo, o singularidad en la distribución de sus elementos, son extraordinarios candidatos a la aplicación de agentes para obtener ventajas en el proceso de resolución.

De acuerdo con Sycara [Syc98], un SMA está caracterizado por lo siguiente:

- Cada agente tiene **información incompleta** o **no tiene** la **suficiente capacidad** para resolver el problema global. Es decir, tiene un punto de vista limitado.
- **Necesitan** de formas de **coordinarse** .
- Los **datos** están **descentralizados**.
- Los **cálculos** son **asíncronos**.

La teoría de agentes y SMA propicia dar un mayor realismo a la gestión de ciertos problemas, sumando a los modelos características que normalmente no se tienen en cuenta por la dificultad que incrementan al proceso de resolución del mismo. Dicha gestión se logra mediante la **coordinación** de los distintos subsistemas que lo componen e integrando los objetivos particulares de cada subsistema en un objetivo común.

Si cada subsistema tiene una capacidad de decisión local, el problema de la gestión se puede abordar definiendo políticas de cooperación, coordinación y negociación entre agentes. En general, un SMA cooperante presentará las siguientes características:

- Está **formado por un conjunto de agentes**, cada uno de los cuales mantiene sus propias habilidades.
- El sistema multiagente **tiene una misión común**, la cual puede **descomponerse** en diferentes **tareas independientes**, de forma que se puedan ejecutar en paralelo.
- Cada agente del sistema tiene un **conocimiento limitado**.
- Cada agente del sistema tiene **cierta especialización para realizar determinadas tareas**, en función de lo que conoce, la capacidad del proceso y la habilidad requerida.

3.2.2 Coordinación y esquemas de planificación

La comunicación adquiere relevancia, en el comportamiento de los agentes, en el momento de coordinar sus acciones, consiguiendo sistemas más consistentes que logren alcanzar los objetivos individuales de dichos agentes y el objetivo global de la sociedad a la que pertenecen de la mejor manera posible.

«La **coordinación** es una propiedad de los SMA que posibilita el desarrollo de una acción en un entorno compartido. El grado de coordinación determinará hasta qué punto el sistema es capaz de evitar acciones innecesarias o ajenas para minimizar el gasto de recursos, evitar situaciones de bloqueo y mantener unas determinadas condiciones de seguridad» [Wei99]. Ejemplos convencionales de coordinación son proveer la información pertinente a otros agentes, garantizar que las acciones de los agentes están sincronizadas y evitar la solución redundante de problemas.

En consecuencia, la cooperación (ver Figura 3.13) surge como la intercomunicación entre agentes que no se enfrentan entre ellos, es decir, que comparten un propósito, mientras que la negociación (ver Figura 3.13) se corresponde con la coordinación entre agentes competitivos o interesados en su beneficio. Normalmente, la cooperación entre agentes tiene éxito si mantienen un modelo interno del resto de agentes y un modelo para llevar a cabo futuras interacciones.

Existen tres aspectos principales para estudiar de una manera formal la comunicación entre agentes [Wei99]:

- La **sintaxis**: cómo se estructuran los símbolos de la comunicación,
- La **semántica**: qué significan los símbolos.
- La **pragmática**: cómo se interpretan los símbolos.

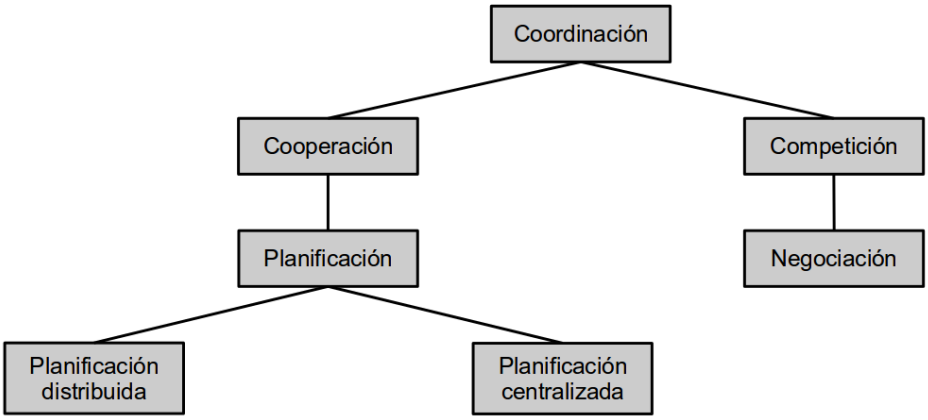


Figura 3.13: Clasificación de las principales vías de coordinación entre agentes [Val09]

El significado se considera una combinación de la semántica y la pragmática. Los agentes se comunican con el fin de comprender y ser comprendidos, por lo que es importante tener en cuenta las diferentes dimensiones de significado que están asociadas con la comunicación.

Las acciones de varios agentes deben coordinarse porque normalmente existe una dependencia entre ellas, debido a la necesidad de cumplir unas restricciones globales y a que ningún agente tiene la capacidad, los recursos o la información necesaria para alcanzar dichas metas globales.

Con el objetivo de obtener sistemas coordinados, gran parte de la investigación en el ámbito de la IAD ha sido el desarrollo de técnicas para dividir el control y los datos. Esta acción implica que los agentes tengan la suficiente autonomía para tomar sus propias decisiones y generar las acciones oportunas para alcanzar unos determinados objetivos. Sin embargo, la principal desventaja de este enfoque es que el conocimiento global del sistema está disperso, de manera que cada agente mantiene una visión parcial e imprecisa del problema global.

Una cuestión obvia en la resolución de problemas multiagente es la planificación de las actividades de un grupo de agentes. La **planificación**, se puede definir como «un mecanismo de coordinación en el que los agentes construyen secuencias de acciones o planes, los cuales especifican todas sus acciones futuras e interacciones con respecto a un objetivo particular. Cuando estos planes se ejecutan correctamente y en orden, llevan al sistema a un estado u objetivo deseado» [PLA01].

Con esta aproximación los agentes saben de antemano qué acciones llevarán a cabo ellos mismos y los restantes agentes, y qué interacciones se producirán. Esta técnica de coordinación tiene un horizonte temporal corto, debido a que los planes requieren una especificación de comportamiento completa, y pueden producirse en el entorno eventos inesperados que eviten la ejecución con éxito de los planes, siendo necesaria una replanificación.

A la hora de realizar la planificación multiagente hay que tener en cuenta un conjunto de criterios:

- El **coste computacional**.
- Los **costos de comunicación**: número de mensajes, volumen de datos y latencia necesaria.
- La **flexibilidad**: el grado de libertad que cada agente da a los otros, la hora, los recursos y la elección de la acción.
- La **robustez**: la capacidad de tener éxito si se produce un cambio en el entorno.
- El **plan de calidad**.
- La **escalabilidad** en cuanto a: número de agentes, tamaño de entrada/salida del problema y las interacciones.

«La planificación multiagente debe tomar en consideración el hecho de que las actividades de los agentes pueden interferir entre sí, por lo tanto, sus actividades deben ser coordinadas» [Woo01]. Generalizando existen dos opciones para la planificación multiagente:

- **Planificación centralizada** para planes distribuidos: un sistema de planificación centralizado desarrolla un plan para un grupo de agentes, en el que se define la división y ordenación del trabajo. Este agente «maestro» distribuye el plan entre los «esclavos», que luego ejecutan su parte del plan.
- **Planificación distribuida**: un grupo de agentes cooperan para formar un plan centralizado. Por lo general, existirán agentes que son «especialistas» en diferentes aspectos del plan, y contribuirán a una parte de él. Sin embargo, los agentes que forman el plan no serán los que lo ejecuten; su función es meramente para generar el plan.

Por lo general, la planificación centralizada es más rápida, más simple y suele dar mejores resultados que la planificación descentralizada, debido a que el «maestro» puede tener una visión de conjunto, y puede dictar las relaciones de coordinación según se requiera.

3.3 Sistemas de ayuda y emergencia en catástrofes

La alta frecuencia de desastres, tanto naturales como artificiales, que puedan causar la pérdida de seres humanos, es un problema en el que los drones pueden contribuir a una notable **mejoría en los tiempos de respuesta y actuación**.

Un grupo multidisciplinar de voluntarios con **equipos de alta tecnología** puede colaborar en la búsqueda de personas o realizar un análisis más veloz en situaciones de emergencia, en las cuales pueden estar en peligro cientos de vidas. Es evidente que una visión global aérea de un cataclismo proporciona una **toma de decisiones más correcta**.

Si algo ha cambiado en estos últimos meses, es la percepción pública acerca de los drones. Es usual que quien observa el vuelo de un dron se cuestione si está prohibido hacerlo. Da

la impresión que se está estigmatizando este instrumento cuyas capacidades son tan beneficiosas que deberían despejar la incertidumbre sobre la legalidad de su manejo (véase Anexo A).

El despliegue de esta tecnología implica una **renovación radical en las tareas de ayuda y rescate** de vidas humanas, desde el posicionamiento óptimo, la logística y aporte de medios, hasta el rescate de vidas en sí mismo.

Los drones, en contraste con una UVI móvil, no pueden socorrer a un paciente en movimiento durante su transporte con la participación de varios doctores. Pero sí que es posible el transporte urgente y existen proyectos para el envío de un dron que pueda portar un desfibrilador [6] (ver Figura 3.14).

Figura 3.14: Ambulance Drone

«Los drones de rescate han entrado en acción en zonas de contaminación, tras un accidente nuclear, en terremotos o inundaciones. Son pequeños y pueden buscar personas y objetos en grandes áreas, incluso de noche. Consiguen imágenes accediendo a edificios a punto de caer, buscando supervivientes y obteniendo datos e información vital. Son capaces de realizar vuelos entre el humo de los incendios e identifican térmicamente siluetas de personas para saber dónde están atrapados o si se han desmayado y están inconscientes. En atmósferas explosivas y tras accidentes por explosión de gas, inspeccionan y toman muestras del aire, pasan entre los escombros localizando a las personas, se mueven rápido y pueden recoger muestras de agua contaminada o sustancias químicas tras acceder a sitios inverosímiles. Y todo ello en apenas unos minutos, visualmente, y en tiempo real, enviándolo al instante a los centros de control, mejorando así los resultados y disminuyendo los costes» [Dí15].

[6] `http://www.io.tudelft.nl/onderzoek/delft-design-labs/applied-labs/ambulance-drone/`

En conclusión, es factible imaginar un futuro muy cercano con aplicaciones en las que los ciudadanos tengan la seguridad de poder solicitar y disponer de un dron ante cualquier situación de necesidad, urgencia y riesgo.

3.3.1 Proyecto ICARUS

Tras los terremotos en L'Aquila, Haití y Japón, la Comisión Europea corroboró que existe una gigantesca discrepancia entre la tecnología, que se desarrolla en el laboratorio, y el empleo de dicha tecnología, en el terreno para las operaciones de búsqueda y rescate (BYR) y la gestión de crisis.

De este modo, la Dirección General de Empresa e Industria de la Comisión Europea decidió financiar **ICARUS** [7], un proyecto de investigación, con un presupuesto total de 17,5 millones de euros, cuyo objetivo es desarrollar herramientas robóticas que puedan ayudar a los equipos de intervención en situaciones de crisis y de este modo, reducir el riesgo y el impacto de la crisis sobre los ciudadanos.

La inclusión de dispositivos no tripulados, en misiones de BYR, aporta una **herramienta de utilidad para salvar vidas humanas** y para **acelerar el proceso de BYR**. ICARUS se concentra en el desarrollo de las tecnologías de búsqueda y rescate, con UAVs, para la detección, localización y el rescate de seres humanos.

Existe una abundante literatura sobre los esfuerzos de investigación hacia el desarrollo de herramientas de BYR. Sin embargo, esas investigaciones están en contraste con la realidad práctica en el campo, en el que las herramientas de búsqueda y salvamento no tripuladas tienen muchas dificultades para encontrar a usuarios finales. El proyecto ICARUS se ocupa de estos problemas, con el objetivo de cerrar la brecha entre la comunidad científica y los usuarios finales.

El uso de dispositivos no tripulados de BYR, embebidos en una arquitectura apropiada e integrados en las infraestructuras existentes, ayudarán al personal de crisis, proporcionando **información detallada** y fácil de entender sobre la situación. El sistema propuesto debe informar al equipo sobre los peligros reales presentes en el terreno, y por lo tanto **aumentará su rendimiento** en la resolución de la situación.

3.3.2 Sistema Pelícano

El **sistema Pelícano** de Indra [8] (ver Figura 3.15), está constituido por cuatro helicópteros no tripulados y una estación de control, que proporciona autonomía suficiente para operar las 24 horas del día durante ciclos de tiempo prolongados. Su diseño está pensado para satisfacer los requisitos y necesidades de las fuerzas armadas y de seguridad.

[7] http://www.fp7-icarus.eu/

[8] http://www.indracompany.com/es/

Figura 3.15: UAV Pelícano de Indra

El sistema de UAVs Pelícano [PEL] está preparado para proporcionar apoyo, tanto en misiones de inteligencia como en gestión de emergencias, tales como **desastres naturales o medioambientales**, que impliquen el seguimiento, vigilancia y reconocimiento de amplias superficies, eliminando así pérdidas humanas.

Indra ha cimentado el sistema sobre un helicóptero de tamaño medio de la compañía sueca CybAero [9] e incorporará los sistemas tecnológicos más avanzados para adaptarlo a las necesidades operativas militares y civiles.

Entre los sensores con los que está equipada la aeronave destacan los sistemas electroópticos de visión diurna e infrarroja, capaces de **tomar imágenes de muy alta resolución a gran altura**. También esta acondicionado para poder portar un radar ligero, así como sistemas de inteligencia electrónica y sensores de detección de amenazas químicas, bacteriológicas, radioactivas y nucleares.

3.3.3 Proyecto ATLANTE

El Centro para el Desarrollo Tecnológico Industrial (CDTI) actuó como organismo gestor de programas del sector aeronáutico, y lanzó el programa **ATLANTE** [10], liderado por Cassidian, tras constatar el interés de la Industria española por los UAS y con el fin de potenciar el desarrollo de la tecnología asociada a este tipo de sistemas mediante un proyecto realizado íntegramente en España.

El ATLANTE (ver Figura 3.16) es un sistema diseñado para responder a las exigencias de las fuerzas terrestres militares. Permite realizar misiones de inteligencia, vigilancia y reconocimiento, de día y noche, en condiciones climatológicas adversas. También puede llevar a cabo labores de identificación de blancos, corrección de tiro y evaluación de daños.

[9] http://www.cybaero.se

[10] http://www.indracompany.com/es/indra/avion-tactico-largo-alcance-tripulado-espanol-atlante

Figura 3.16: UAV ATLANTE

Entre las aplicaciones no militares se encuentran la lucha anti-terrorista y contra la piratería, el control de inmigración ilegal y el tráfico de drogas, la supervisión de infraestructuras críticas y, cómo no, la **lucha contra incendios y otros desastres naturales**.

El sistema se compone de cuatro o más vehículos aéreos no tripulados, una estación de control terrestre, una unidad de transporte y **terminales de vídeo remoto**. Los equipos de misión de la plataforma aérea recogen la información de los sensores y los transmiten a tierra a través de los enlaces de datos de alta velocidad para su posterior aprovechamiento.

3.3.4 Proyecto Multielfo

Unmanned Solutions, también conocida como **USol**, es una de las empresas españolas con una oferta más variada en el sector de los drones.

Desde el 2004, USol ha trabajado, y sigue haciéndolo, en la Serie K, unos drones de ala fija en varias configuraciones, con pesos máximos al despegue de hasta 150 kg y autonomía superior a las 18 horas. El mundo de los multirrotores lo tienen también cubierto con el **proyecto Multielfo** [11], con drones de cuatro y seis rotores de varios pesos y configuraciones.

Los drones del proyecto Multielfo, entre los que se encuentran el Elfo Mini, el Elfo IV (ver Figura 3.17), el Hexaelfo y el Metaelfo, tienen capacidades comunes:

- **Admiten diferentes modelos de cámaras** y permiten **adaptar cualquier gimbal**.
- **Despegue y aterrizaje manual o autónomo**, con **vuelta a casa** y **recorrido por puntos GPS**.

[11] `http://www.multielfo.es/index.html`

Figura 3.17: Multirrotor Elfo IV

Las aplicaciones del proyecto MultiElfo son muy diversas, incluyendo:

- **Filmaciones** y fotografía aérea.
- Cartografía y fotogrametría.
- Monitorización de infraestructuras industriales.
- Soporte en acciones de **búsqueda y rescate**.
- Agricultura de precisión.
- Vigilancia perimetral, durante el día y la noche.
- Soporte en emergencias y **desastres naturales**.

Se pueden encontrar grabaciones aéreas [12] en las que el proyecto Multielfo participa en un **simulacro de emergencia** celebrado en Daimiel el 15 de Marzo de 2015.

3.3.5 Robot Aéreo Fotogramétrico

El **RAF** [13] (ver Figura 3.18) es una nueva tecnología para monitoreo y levantamientos fotogramétricos detallados. El Robot Aéreo Fotogramétrico es un novedoso hexacóptero inteligente basado en tecnología de multirotores coordinados, ideal para aplicaciones de monitoreo, cartografía, **inspección** y vigilancia.

[12] https://www.youtube.com/watch?v=vZ2-hFHZs9s
[13] http://www.cartodata.com/technologies/raf-4/

Figura 3.18: Hexacóptero RAF

Comparado con otros drones, el RAF presenta las siguientes ventajas:

- Estabilidad de cámara.
- Resistente a impacto.
- Sensores de proximidad.
- Piloto automático.
- Despegue y aterrizaje automático.
- Trayectoria de vuelo programable.

La tecnología innovadora del RAF permite generar fotografías aéreas en un tiempo récord, simplificando así la consecución de diversos objetivos como por ejemplo, la seguridad y vigilancia, **la inspección y evaluación de desastres** o la monitorización del desgaste de infraestructuras.

3.3.6 Proyecto Hawk

El grupo de investigación BISITE [14], de la Universidad de Salamanca, mediante el **proyecto HAWK** [15] pretende diseñar, crear y controlar un VTOL UAV (ver Figura 3.19), o vehículo aéreo no tripulado de aterrizaje y despegue vertical, de tipo multirrotor.

BISITE considera que de momento hay un espacio vacío en el mercado de los UAVs multirotores desde un punto de vista tecnológico. El proyecto HAWK intenta llenar ese hueco a través del uso de nuevas tecnologías.

El control de los UAVs es un área con muchas posibilidades y BISITE está actualmente trabajando en un proyecto para las fuerzas de seguridad del estado en colaboración con un consorcio de compañías. Se ha desarrollado un software de control y monitorización que se ejecuta desde una estación base y que permite el control manual y automático desde el puesto de control en tierra.

[14] http://bisite.usal.es/es
[15] http://bisite.usal.es/es/apps/hawk-multicopter

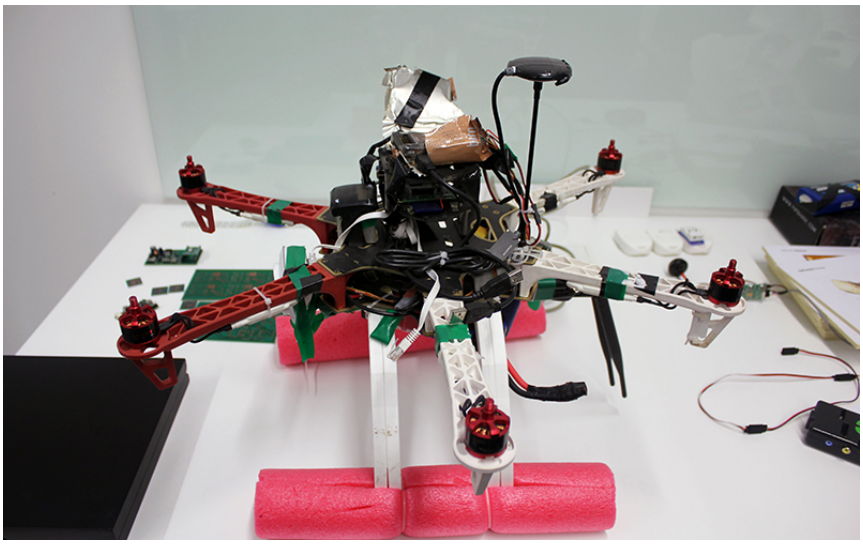


Figura 3.19: UAV para el proyecto Hawk

Algunos posibles usos potenciales son:

- Estudios de **imágenes aéreas**.
- Control de líneas de alto voltaje.
- Reconocimiento de aerogeneradores.
- Análisis de estructuras arquitectónicas.
- **Inspección de zonas dañadas por desastres**.
- Control de plagas.
- Filmación aérea profesional.
- Supervisión de la vida salvaje.
- Estudios científicos y técnicos.
- Usos de la policía y cuerpo militar.

3.4 Hardware, herramientas y entornos de desarrollo

En esta sección se estudia tanto el hardware como las principales herramientas y entornos involucrados en el desarrollo de sistemas con drones.

3.4.1 UAVs más utilizados en el ámbito del desarrollo

3.4.1.1. DJI Matrice 100

El **DJI Matrice 100** (ver Figura 3.20) es una plataforma de vuelo estable, flexible y de gran alcance. Su plataforma abierta y su diseño altamente personalizable lo hace adecuado para una amplia gama de aplicaciones en las áreas de investigación, los negocios y el ocio.

Figura 3.20: DJI Matrice 100

El DJI Matrice 100 es un dron expansible (ver Cuadro 3.3) que hace que sea fácil montar componentes adicionales para conseguir una mayor funcionalidad. Cuenta con compartimiento de batería adicional para una segunda batería de vuelo inteligente puede ser instalado para ofrecer un tiempo de vuelo extendido de hasta 40 minutos.

Especificaciones	
Piloto automático	N1
Firmware	Zenmuse X3 v.1.3.1.00
GPS	DJIMAT-27
Telemetría	5.725 5.825 GHz
Motores	DJI 3510 (2 unidades en sentido horario y 2 unidades en sentido antihorario)
Tipo de chasis	V
Hélices	DJI 1345s (2 unidades en sentido horario y 2 unidades en sentido antihorario)
Batería	Batería de polímero de litio 2S 6,0 Ah
Peso con batería	2355 g
Distancia de motor a motor	650 mm

Cuadro 3.3: Especificaciones técnicas del DJI Matrice 100 [DJI16]

A los desarrolladores se les concede un control total sobre su plataforma de vuelo, dándoles la libertad de personalizar una solución integral aérea utilizando el **DJI SDK**. Esta SDK permite usar el sistema dedicado para comunicarse con el controlador de vuelo DJI a través de una conexión serie directa. Posibilita la monitorización y el control del comportamiento de vuelo de la aeronave con las funciones que implemente la API, mientras se utilizan los modos de navegación inteligentes integrados para crear trayectorias de vuelo y maniobras autónomas.

3.4.1.2. Parrot AR Drone 2

El **Parrot AR Drone 2** (ver Figura 3.21) es un vehículo aéreo no tripulado radiocontrolado de uso recreativo civil.

Figura 3.21: Parrot AR Drone 2

Funciona propulsado por cuatro motores eléctricos en configuración cuadricóptero y es similar en su estructura básica y aerodinámica a otros modelos radiocontrolados (ver Cuadro 3.4), pero se diferencia de todos ellos en que cuenta con un microprocesador y una serie de sensores, entre los cuales se incluyen dos cámaras que le permiten captar lo que ocurre a su alrededor, más un conector Wi-Fi integrado que le permite vincularse a dispositivos móviles personales que cuenten con los sistemas operativos iOS, Android o Linux.

Especificaciones	
Piloto automático	PX4IOAR
Firmware	ARDrone2 firmware v2.4.8
GPS	uBlox GPS
Motores	4 motores de rotor interno sin escobillas. 14,5 W y 28,5 cuando queda suspendido en el aíre.
Batería	Batería de polímero de litio 3S 1,0 Ah
Peso con batería	420 g
Alcance radio	50 m

Cuadro 3.4: Especificaciones técnicas del Parrot AR Drone 2 [16]

[16] `http://ardrone-2.es/especificaciones-ar-drone-2/`

El **Software Development Kit** del AR.Drone permite a los desarrolladores realizar aplicaciones y proporciona la API necesaria para compilar los programas. Según la versión, tiene soporte para crear aplicaciones en iOS, Linux, Windows y Android.

3.4.1.3. 3DR IRIS+

«El **3DR IRIS+** (ver Figura 3.22) es un vehículo aéreo autónomo todo en uno, producido por 3D Robotics [17], que se ejecuta sobre el sistema de piloto automático Pixhawk, lo último en electrónica avanzada para el piloto automático desde el proyecto del PX4. Con la ayuda de puntos GPS preprogramados es capaz de efectuar operaciones de nivel profesional, como el mapeo, la toma de imágenes con scripts, la investigación científica, y otras aplicaciones donde se requieren planes de vuelo repetitivos» [Def14]. Es concebido como una plataforma de captación de imágenes aéreas potenciado por hardware, software y firmware de código abierto.

Figura 3.22: 3DR IRIS+

Este UAV está capacitado para formar parte de sistemas, programándolo mediante la **API DroneKit** (véase Sección 3.4.4), que se ajusten a ciertos propósitos como: una aplicación agrícola que monitorice el terreno, una aplicación de búsqueda y rescate o una aplicación de cartografía. El 3DR IRIS+ puede formar parte de sistemas desarrollados para aplicaciones móviles, aplicaciones basadas en la web o aplicaciones para computadores.

El 3DR IRIS+ cuenta con una serie de componentes y especificaciones (ver Cuadro 3.5) como: GPS integrado, que permite establecer una ruta de antemano o volver a casa en caso de pérdida, o un kit de telemetría que permite obtener información de vuelo como la distancia,

[17] `https://www.3dr.com`

altitud, velocidad, nivel de batería e intensidad de la señal GPS. Como en cualquier dron, se puede personalizar, y es posible la inclusión de una cámara GoPro HERO si se quiere mejorar la calidad de la imagen.

Especificaciones	
Piloto automático	Pixhawk v2.4.5
Firmware	ArduCopter 3.2
GPS	3DR uBlox GPS con brújula (módulo LEA-6H)
Telemetría	3DR Radio Telemetry v2 (433 mHz)
Motores	MN2213 950 kV (2 unidades en sentido horario y 2 unidades en sentido antihorario)
Tipo de chasis	V
Hélices	9,5 x 4,5 T-Motor multirrotor (2 unidades en sentido horario y 2 unidades en sentido antihorario)
Batería	Batería de polímero de litio 3S 5,1 Ah 8C
Voltaje de batería baja	10,5 V
Máximo voltaje	12,6 V
Peso de la batería	320 g
Peso con batería	1282 g
Altura	100 mm
Distancia de motor a motor	550 mm
Capacidad de carga	400 g
Alcance radio	Hasta 1 km

Cuadro 3.5: Especificaciones técnicas del 3DR IRIS+ [3D 15]

3.4.2 ArduPilot

ArduPilot [18] es una plataforma de código abierto para pilotos automáticos, de bajo coste y fácil de usar, creada en 2007 por Chris Anderson y Jordi Muñoz, miembros de la comunidad DIY Drones [19]. Se apoya en la plataforma de prototipado electrónico Arduino [20] y es capaz de controlar multirrotores, aeronaves de ala fija, helicópteros tradicionales y vehículos de exploración terrestre. ArduPilot es una plataforma galardonada en 2012 y 2014 en las competiciones UAV Challenge [21].

La primera versión de ArduPilot (ver Figura 3.23), lanzada en 2009, se basó en una termopila, cuya clave era la determinación de la ubicación del horizonte con respecto al avión por la medición de la diferencia de temperatura entre el cielo y el suelo. Más tarde, el sistema se mejoró reemplazando las termopilas con una unidad de medición inercial y utilizando una combinación de acelerómetros, giróscopos y magnetómetros.

[18] http://www.ardupilot.com
[19] http://diydrones.com
[20] https://www.arduino.cc/
[21] https://www.uavchallenge.org

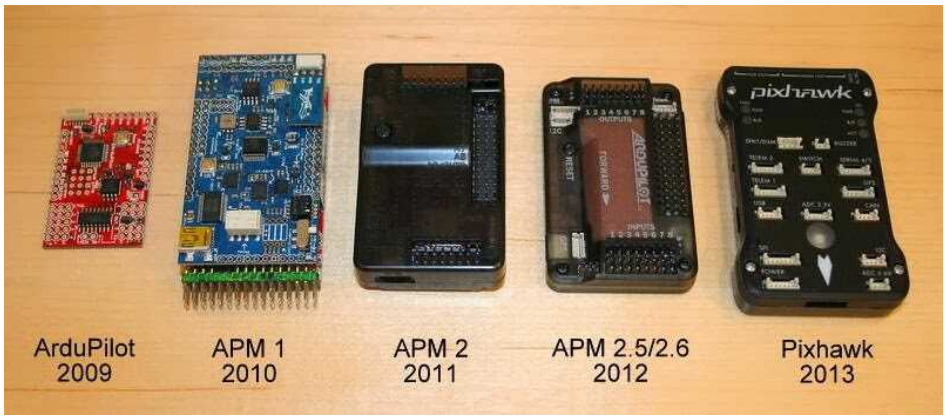


Figura 3.23: Historia de ArduPilot [22]

En el presente, el proyecto ArduPilot se sustenta sobre cuatro pilares fundamentales:

- **Hardware**: Los sistemas embebidos y sensores periféricos que actúan como el cerebro del vehículo, los ojos, las orejas, etc. Entre ellos se encuentra la línea APM y **Pixhawk**/PX4 de pilotos automáticos.
- **Firmware**: El código que implementa el conjunto de habilidades que se ejecuta en el hardware, configurado para el tipo de vehículo que se elija. El firmware elegido puede ser **ArduCopter** (véase Sección 3.4.3), ArduPlane o ArduRover.
- **Software**: La interfaz con el hardware. Por ejemplo, la aplicación de estación de tierra **APM Planner** (véase Sección 4.2.2.5)
- **Comunidad**: Un área para el discurso abierto entre los desarrolladores y sus clientes.

El código de vuelo principal de ArduPilot está escrito en C++, mientras que las herramientas de apoyo se escriben en diversos lenguajes, siendo Python el más común. Actualmente el código principal del vehículo está escrito en archivos «.pde», que provienen del sistema de compilación Arduino. Los archivos PDE son preprocesados en un archivo .cpp como parte de la compilación.

El enfoque de ArduPilot respecto a su software libre es que el bajo coste y la disponibilidad permitan el uso en pequeños aviones pilotados a distancia.

3.4.2.1. Pixhawk

Pixhawk [23] es un **piloto automático de alto rendimiento** adecuado para vehículos aereos de ala fija, multirrotores y cualquier otra plataforma robótica que se pueda mover. Está dirigido a las necesidades en el ámbito de la investigación, de la industria y del ocio. Es el piloto automático estándar de la industria, diseñado en colaboración con 3D Robotics.

[22] http://www.ardupilot.org/copter/docs/common-history-of-ardupilot.html
[23] https://www.pixhawk.org

El piloto automático Pixhawk (ver Figura 3.24) cuenta con un procesador avanzado, tecnología de sensores y un sistema operativo en tiempo real, que ofrece un alto rendimiento, flexibilidad y fiabilidad para controlar cualquier vehículo autónomo.

Figura 3.24: Piloto automático Pixhawk

Los beneficios del sistema Pixhawk incluyen multi-threading, un entorno de programación Unix, funciones de piloto automático completamente nuevas y una capa de control personalizada que asegure la sincronización en los procesos integrados. Estas capacidades avanzadas garantizan que no existan limitaciones en el vehículo autónomo.

3.4.3 ArduCopter

«**ArduCopter** [24] es el firmware desarrollado por la comunidad del Proyecto ArduPilot que permite el vuelo, y la monitorización, de aeronaves que basan su funcionamiento, generalmente, en motores eléctricos con hélices que ejerzan la fuerza actuadora. Esta característica reúne vehículos aéreos como los quadrotors, hexarotors, trirotors, octarotors o helicópteros» [Rom15]. Ofrece tanto una experiencia mejorada de vuelo por control remoto como la ejecución de misiones totalmente autónomas.

ArduCopter facilita el uso de UAVs, de tipo multirrotor o helicoptero, pudiendo ser ejecutado sobre un piloto automático de ArduPilot. Junto con el uso de un módulo GPS, se convierte

[24] http://www.arducopter.co.uk

en una solución completa que lo diferencia de multirrotores tradicionales que sólo soportan el control remoto. El firmware ArduCopter proporciona un vuelo totalmente autónomo basado en «waypoints», a través de una estación de control terrestre o GCS, que permite la planificación de misiones sirviéndose para ello del empleo de telemetría en tiempo real.

Las principales características que ofrece ArduCopter son las siguientes:

- **No requiere conocimientos de programación**: para usar ArduCopter solo se debe descargar una utilidad de escritorio, como APM Planner (véase Sección 4.2.2.5), que cargue el firmware con un solo clic.
- Uso de **puntos de ruta ilimitados**.
- Permite **mantener el dron en una localización** determinada valiéndose del GPS y del sensor de altitud.
- **Vuelta a casa** o **RTL**: ArduCopter establece una posición como punto de despegue, de tal modo, que accionado una palanca el dron regresa automáticamente hasta allí.
- **Despegue** y **aterrizaje automático.**
- **Control de la cámara** totalmente programable: incluyendo la capacidad para mantener la cámara apuntando a un objeto en el suelo.
- **Multiplataforma**: compatible con Windows, Mac y Linux.
- **Compatibilidad con los estándares** de robótica líderes en la industria como el protocolo de comunicación MAVLink [25].

3.4.4 DroneKit-Python

DroneKit-Python [26] es una API desarrollada por 3D Robotics, de código abierto y orientada a la comunidad, que permite crear aplicaciones, para vehículos aéreos no tripulados, que se ejecutan en un ordenador. Dichas aplicaciones se comunican con el controlador de vuelo ArduPilot, utilizando un enlace de baja latencia, y pueden mejorar significativamente el piloto automático, añadiendo una mayor inteligencia al comportamiento del vehículo o realizando tareas que son computacionalmente intensivas o sensibles al tiempo (por ejemplo, la visión por ordenador o la planificación de trayectorias).

Se comunica con los vehículos por medio del protocolo **MAVLink**. Proporciona acceso, mediante programación, a la información de telemetría, al estado y parámetros de información del vehículo, y permite gestionar las misiones y tener un control directo sobre el vehículo. Se ejecuta en Linux, Mac OS X o Windows.

[25] http://www.qgroundcontrol.org/mavlink/start
[26] http://python.dronekit.io/

Dronekit-Python proporciona clases y métodos para realizar múltiples funciones, entre otras:

- **Conexión** a uno o varios **vehículos**.
- **Obtener y establecer información** sobre el estado del vehículo.
- Recibir **notificaciones** asíncronas de los **cambios de estado**.
- **Guiar** un **UAV** hasta una posición especificada.
- Crear y **gestionar misiones** gracias a «waypoints».
- **Reemplazar** la configuración de los **canales de radiocontrol**.

«Cuando DroneKit no proporciona un comando para una maniobra deseada, se requiere el uso de mensajes personalizados MAVLink. Cualquier maniobra aérea debe ser capaz de llevarse a cabo a través de una de estas dos interfaces» [LTT].

DroneKit-Python ofrece una guía que incluye una visión general de cómo utilizar la API, así como un listado de buenas prácticas (véase Anexo B).

3.4.5 OpenCV

OpenCV [27] es una biblioteca de funciones de programación, para la visión por computador en tiempo real, iniciado en Intel en 1999 por Gary Bradsky. Se encuentra bajo licencia BSD, por lo tanto, es libre tanto en el uso académico como en el comercial. Posee interfaces en C++, C, Python y Java, y es compatible con Windows, Linux, Android, iOS y Mac OS. Cuenta con más de 2500 algoritmos optimizados y más de 9 millones de descargas.

«La librería OpenCV está dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus muchas áreas de aplicación destacarían: interacción hombre-máquina, segmentación y reconocimiento de objetos, reconocimiento de gestos, seguimiento del movimiento, estructura del movimiento y robots móviles» [AGA].

El conjunto de funciones suministradas por la librería OpenCV se agrupan en los siguientes bloques:

- Estructuras y **operaciones básicas**: matrices, grafos, árboles, etc.
- Procesamiento y **análisis de imágenes**: filtros, momentos, histogramas, etc.
- **Análisis estructural**: geometría, procesamiento del contorno, etc.
- **Análisis del movimiento** y **seguimiento de objetos**.
- **Reconocimiento de objetos**: objetos propios, modelos HMM, etc.
- **Calibración de la cámara**: morphing, geometría epipolar, estimación de la pose, etc.
- Interfaces gráficos de usuarios y **adquisición de vídeo**.

[27] `http://www.opencv.org`

3.4.6 Google Cloud Vision API

Google Cloud Vision API [28] permite a los desarrolladores entender el contenido de una imagen, mediante la encapsulación de potentes modelos de aprendizaje automático. Esta REST API clasifica rápidamente las imágenes en miles de categorías (ver Figura 3.25), detecta objetos individuales y caras, y encuentra y lee palabras impresas.

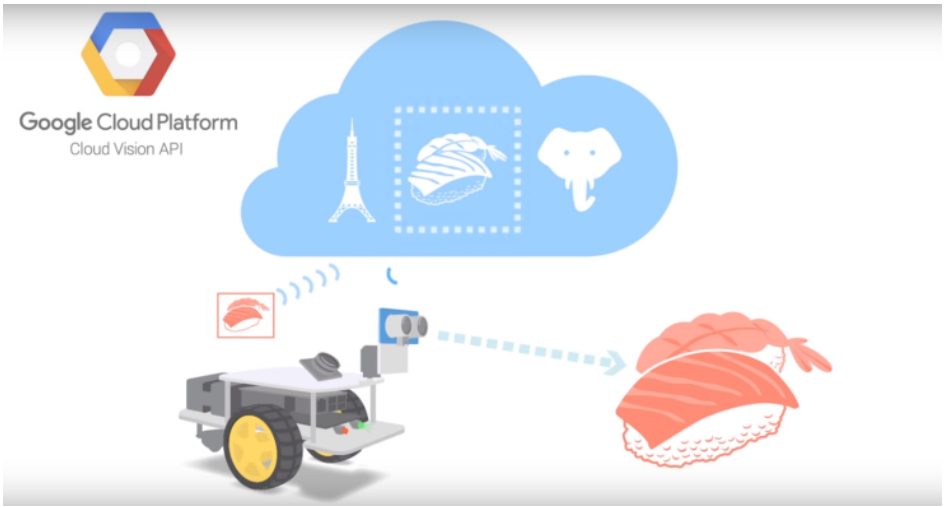


Figura 3.25: Funcionamiento de Google Cloud Vision API

Es capaz de detectar conjuntos de objetos en las imágenes, como flores, automóviles, animales u otros de los muchos elementos que se encuentran comúnmente dentro de las fotografías. También, puede analizar atributos faciales emocionales de las personas en sus imágenes, como la alegría, la tristeza y la ira. Se encuentra en constante progreso a medida que se introducen nuevos conceptos y se mejora la precisión.

Vision API ofrece las siguientes funcionalidades:

- **Detección de etiquetas** (ver Figura 3.26): detecta amplios conjuntos de categorías.
- **Detección de rostros**: detección de múltiples caras, junto con los atributos faciales claves asociados.
- **Detección de contenido explícito**: detecta contenido explícito como contenido para adultos o de contenido violento.
- **Atributos de imagen**: detecta los atributos generales de la imagen, como el color dominante.
- **Detección de logotipos** populares.
- **Detección de interés**: detección de estructuras naturales y artificiales populares.

[28] `https://cloud.google.com/vision/`

- **Reconocimiento óptico de caracteres**: detecta y extrae el texto, con soporte para una amplia gama de idiomas, junto con el soporte para la identificación automática del lenguaje.
- **REST API integrada**: acceso a través de la REST API para solicitar uno o más tipos de detección por imagen.

El uso de cada una de estas características conlleva gastos.

Figura 3.26: Detección de etiquetas

Capítulo 4

Método de trabajo

Esta sección está centrada en detallar y justificar la metodología usada durante la elaboración del proyecto, de tal forma que se exponga al lector cómo ha sido llevado a cabo el desarrollo a nivel organizativo. Conjuntamente, se especifican los medios hardware y software que han sido utilizados a lo largo de todo el TFG.

4.1 Metodología

Para la elección de la metodología de trabajo se debe escoger una que se ajuste a las condiciones del proyecto, es decir, que satisfaga la dinámica de trabajo que se desea seguir. Por lo cual es primordial que la metodología elegida divida el **desarrollo en hitos** que son potencialmente entregables y que no contienen errores. Igualmente, se debe tener en consideración el contexto donde se desarrolla el sistema. En este proyecto, al tiempo que se realiza el desarrollo del sistema es plausible que aparezcan complicaciones, por lo que es preciso que la metodología sea flexible, en otras palabras, que sea posible corregir las tareas con respecto a los inconvenientes que aparecerán durante el progreso del proyecto, conservando como línea de referencia los objetivos precisados (Referenciar Capitulo de objetivos).

En consecuencia, se excluyen las metodologías de desarrollo tradicionales, basadas en una planificación inicial que permanece intacta de principio a fin. Se decide escoger una **metodología de desarrollo ágil** cuyo principal objetivo es desarrollar proyectos de forma **iterativa e incremental**, en la que puedan definirse objetivos en cada iteración. La preferencia sobre una metodología ágil posibilita la entrega, en un periodo de tiempo reducido, de un proyecto funcional y que el usuario pueda evaluar. Entre las metodologías de desarrollo ágil existe un amplio abanico donde elegir: Programación Extrema, Desarrollo Adaptativo de Software, Scrum, etcétera. De ellas, **Programación Extrema** es la que mejor se ajusta a las exigencias del proyecto.

4.1.1 Manifiesto ágil

Para comprender mejor la filosofía tras las metodologías ágiles, seguidamente se cita el **Manifiesto Ágil** [1] y sus cuatro valores:

[1] `http://www.agilemanifesto.org/iso/es/`

«Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- **Individuos e interacciones** sobre **procesos y herramientas**.
- **Software funcionando** sobre **documentación extensiva**.
- **Colaboración con el cliente** sobre **negociación contractual**.
- **Respuesta ante el cambio** sobre **seguir un plan**.

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda». De estos cuatro valores, se desarrollaron doce principios para el Manifiesto Ágil [2], que son:

- La mayor prioridad es **satisfacer al cliente** mediante la entrega temprana y continua de software con valor.
- **Aceptar** que los **requisitos cambien**, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Se **entrega** software funcional **frecuentemente**, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los **responsables** de negocio, en este caso David Vallejo como director del proyecto, y los **desarrolladores trabajan juntos** de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a **individuos motivados**. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de **comunicar información** al equipo de desarrollo y entre sus miembros es la conversación **cara a cara**.
- El **software funcionando** es la **medida principal de progreso**.
- Los procesos Ágiles promueven el **desarrollo sostenible**. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La **atención continua** a la excelencia técnica y al buen diseño **mejora la Agilidad**.
- La **simplicidad**, o el arte de maximizar la cantidad de trabajo no realizado, **es esencial**.
- Las mejores arquitecturas, requisitos y diseños emergen de **equipos auto-organizados**.
- A intervalos regulares **el equipo reflexiona sobre cómo ser más efectivo** para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

[2] `http://agilemanifesto.org/iso/es/principles.html`

4.1.2 Programación Extrema

La **Programación Extrema** (XP) «requiere que se complete algún tipo de producto potencialmente liberable al final de cada iteración. Estas iteraciones están diseñadas para ser cortas y de duración fija. Este enfoque en entregar código funcional cada poco tiempo significa que el desarrollador no tiene tiempo para teorías. No persiguen dibujar el modelo UML perfecto en una herramienta CASE, escribir el documento de requisitos perfecto o escribir código que se adapte a todos los cambios futuros imaginables. En vez de eso, el desarrollador se enfoca en que las cosas se hagan. Se acepta que puede equivocarse por el camino, pero también es consciente de que la mejor manera de encontrar dichos errores es dejar de pensar en el software a un nivel teórico de análisis y diseño y sumergirse en él, ensuciarse las manos y comenzar a construir el producto» [Kni07].

Por consiguiente, se puede definir XP como una metodología de desarrollo ágil que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en las que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

El **ciclo de vida ideal de XP** según [LS06] consiste en seis fases:

- **Exploración**: los clientes, en este caso David Vallejo como director del proyecto, plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el desarrollador se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.
- **Planificación de la entrega**: el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.
- **Iteraciones**: incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción.
- **Producción**: requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
- **Mantenimiento**: el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente.

- **Muerte del proyecto**: cuando el cliente no tiene más historias para ser incluidas en el sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura.

La utilización de XP en el proyecto tiene un conjunto de ventajas asociadas:

- El proceso de **integración es continuo**, por lo que el esfuerzo final para la integración es nulo.
- Se consiguen productos con mayor **rapidez**.
- El desarrollo iterativo hace que los proyectos sean **adaptables** y abiertos a la incorporación de cambios.

4.2 Herramientas

4.2.1 Hardware

Este TFG tiene un elevado componente hardware, siendo los siguiente medios (ver Figura 4.1) los destinados a su elaboración:

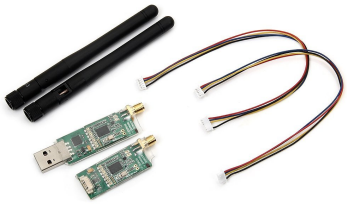
- **Computador Packard Bell EN-TS44HR**.
- **Quadcopter 3DR IRIS+** (véase Sección 3.4.1.3): 3DR IRIS+ hace uso de:
  - **Piloto automático Pixhawk** (véase Sección 3.4.2.1).
  - **Tarot T-2D Gimbal** [3]: asegura tener imágenes estables y claras desde el vehículo.
  - **Kit de telemetría Flug51 FL1515** a 433 MHz: proporciona una manera fácil de configurar una conexión entre el Pixhawk y una estación terrestre.
  - **Sistema radiocontrol FlySky FS-TH9X** [4]: sistema extremadamente versátil que permite controlar el 3DR IRIS+ y puede ser usado por principiantes y expertos.
- Cámara **GoPro HERO 3+ Silver Edition** [5]: permite la visualización de imágenes, así como la grabación de las mismas.
- **Kit FPV** [6]: incluye todo lo necesario para empezar a emitir en directo imágenes a distancia y en primera persona de la GoPro acoplada al 3DR IRIS+. Para su utilización es necesario:
  - **LogiLink USB 2.0** [7]: su función es digitalizar la señal de vídeo.

[3] https://store.3dr.com/products/tarot-t-2d-brushless-gimbal-kit
[4] http://www.flysky-cn.com/products_detail/&productId=42.html
[5] https://es.gopro.com/support/hero3plus-silver-support
[6] http://www.robotshop.com/media/files/pdf/user-manual-vidkit0007.pdf
[7] http://www.logilink.eu/showproduct/VG0001A.htm?seticlanguage=en

(a) Maleta para transportar el 3DR IRIS+

(b) Tarot T-2D Gimbal con GoPro HERO 3+

(c) Kit FPV



(d) Kit de telemetría FL1515

(e) 3DR IRIS+, FS-TH9X y batería LiPo

(f) LogiLink USB 2.0

Figura 4.1: Ejemplo de medios hardware empleados durante el TFG

4.2.2 Software

Para el desarrollo del proyecto se ha hecho uso de los siguientes medios software:

4.2.2.1. Sistema operativo GNU/Linux Ubuntu 14.04 LTS

Ubuntu [8] fue concebido en 2004 por Mark Shuttleworth, un exitoso emprendedor sudafricano, y su compañía Canonical. Es una distribución GNU/Linux que ofrece un sistema operativo que está disponible de forma libre y cuenta con apoyo de la comunidad de usuarios y con soporte profesional.

La comunidad Ubuntu se basa en las ideas consagradas en el **Manifiesto Ubuntu** [9]: **i)** el software deberá estar siempre disponible **sin costo** alguno, **ii)** dicho software podrá ser utilizado en la lengua materna del usuario y a pesar de cualquier discapacidad y **iii)** los usuarios siempre tendrán la **libertad de adaptar** y modificar el software de acuerdo a sus necesidades particulares. Esta libertad es la que hace a Ubuntu radicalmente diferente del software propietario tradicional.

Ubuntu es apropiado tanto para ordenadores de escritorio como para servidores. Incluye más de 1.000 paquetes entre los cuales se incluyen el kernel de Linux y Gnome. También se incluyen las aplicaciones que se esperan en cualquier ordenador de escritorio, como procesador de texto, hoja de cálculo y navegador para Internet. Por lo tanto, su uso es apto en el ámbito doméstico, profesional o educativo.

4.2.2.2. LaTeX

Para la confección del presente documento se ha utilizado **LaTeX** [10]. LaTeXes un sistema de preparación de documentos para la composición tipográfica de alta calidad. Con frecuencia se utiliza para documentos técnicos o científicos, pero se puede utilizar para casi cualquier forma de publicación.

El término LaTeXhace referencia solo al **lenguaje** en que estos documentos son escritos, pero no al editor para escribirlos. Para generar un documento en LaTeX, un archivo .tex debe ser creado empleando un editor de textos y posteriormente será compilado para producir, en este caso, un **archivo PDF**. Cualquier editor puede ser utilizado para crear un documento LaTeX, pero existen también algunos editores específicos para trabajar con LaTeX.

La idea de LaTeXes que el autor se **concentre en el contenido de lo que escribe**, en lugar de la presentación visual. Al preparar un documento LaTeX, el autor especifica la estructura lógica usando conceptos familiares como: capítulo, sección, tabla, figura, etc; dejando al sistema LaTeXpreocuparse de la presentación visual de esas estructuras.

[8] http://www.ubuntu.com

[9] `http://www.ubuntu.com/about/about-ubuntu/our-philosophy`

[10] `https://www.latex-project.org/`

4.2.2.3. GNU Emacs

Para la creación de documentos en LaTeXy para implementar el código fuente, **GNU Emacs** ha sido el editor de texto empleado. GNU Emacs es un editor de textos avanzado que es auto-documentado, extensible y personalizable. Su núcleo es un intérprete de Emacs Lisp, un dialecto del lenguaje de programación Lisp con extensiones para la edición de texto.

Emacs se considera un editor de textos avanzado porque puede hacer mucho más que la simple inserción y eliminación de texto. Puede controlar subprocesos, programas de sangría automática, mostrar varios archivos a la vez, y mucho más.

Auto-documentado significa que en cualquier momento se pueden utilizar comandos especiales, conocidos como comandos de ayuda, para averiguar cuáles son sus opciones, o para averiguar lo que hace cualquier orden. **Personalizable** significa que se puede alterar fácilmente el comportamiento de los comandos de Emacs. Y **extensible** significa que puede ir más allá de la personalización sencilla y crear nuevos comandos.

4.2.2.4. Git

Para el control de versiones se ha optado por el software **Git** [11]. El proyecto se ha alojado en un repositorio online de **Github** [12], que es un servicio gestionado por el sistema de control de versiones Git.

El control de versiones es un sistema que **registra los cambios** de un archivo, o un conjunto de archivos, a largo del tiempo, para poder recordar versiones específicas más adelante. Permite revertir los archivos de nuevo a un estado anterior, comparar los cambios durante el transcurso del proyecto o ver quién modificó algo que podría ser la causa de un problema.

Git es un **sistema distribuido de control de código fuente** diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Ayuda a los programadores a ser más eficientes en el trabajo, ya que ha universalizado las herramientas de control de versiones que no estaban tan popularizadas y tan al alcance de los desarrolladores.

4.2.2.5. APM Planner 2.0

APM Planner es una aplicación de estación de tierra con todas las funciones necesarias para el proyecto de piloto automático de código abierto Ardupilot.

APM Planner es una **estación de control de tierra** para aviones, helicópteros y vehículos. Es compatible con Windows, Linux y Mac OSX. Puede ser usado como una utilidad de configuración o como un complemento de control dinámico para el vehículo autónomo.

[11] `https://git-scm.com/`
[12] `https://www.github.com`

Entre las tareas que APM Planner puede realizar están las siguientes:

- **Cargar el firmware** en el piloto automático.
- **Instalar, configurar y ajustar** el vehículo para mejorar el rendimiento.
- **Introducción de «waypoints»** mediante point-and-click, usando Google Maps,Bing u Open Street Maps.
- **Selección de comandos** de misión mediante desplegables.
- **Descarga y análisis** de los archivos de **log** de misión.
- Con el hardware de telemetría adecuado puede:
  - **Supervisar el estado** del vehículo mientras está funcionando.
  - **Grabar registros** de telemetría.
  - Ver y **analizar estos registros** de telemetría.
  - Hacer funcionar su vehículo en **FPV**.

4.2.2.6. Software in the Loop

La simulación permite realizar pruebas de seguridad de código y ajustes experimentales, además de servir para practicar el uso de la estación de tierra desde el escritorio. **SITL** consigue simular helicópteros, aviones o vehículos terrestres sin la necesidad de ningún hardware. Su ejecución es posible en Windows, Linux y Mac OSX.

SITL es una reproducción del código del piloto automático utilizando un compilador C++, resultando esto en un ejecutable que posibilita **probar el comportamiento del código sin hardware**. El simulador SITL permite ejecutar ArduPilot en su computador directamente, sin ningún hardware especial. Esto es porque aprovecha el hecho de que ArduPilot es un piloto automático portátil que puede funcionar en una variedad muy amplia de plataformas, siendo el ordenador otra plataforma en la que ArduPilot puede ser ejecutada.

Una gran ventaja de ArduPilot en SITL es que le da acceso a toda la gama de herramientas de desarrollo disponibles, como depuradores interactivos, analizadores estáticos y herramientas para el análisis dinámico. Esto hace que el desarrollo y prueba de nuevas funciones en ArduPilot sea mucho más sencillo.

4.2.3 Lenguajes

4.2.3.1. Python 2.7.6

El lenguaje principal para el desarrollo de este proyecto es **Python** [13], ya que se dispone de un SDK en este lenguaje, como es DroneKit-Python (véase Sección 3.4.4), que facilitará la implementación considerablemente.

[13] `https://www.python.org/`

Python es un lenguaje de propósito general, de alto nivel, interpretado y que admite la aplicación de diferentes paradigmas de programación, como por ejemplo, la programación procedural, la programación imperativa o la **programación orientada a objetos**. Es ideal para prototipado rápido, pero también permite el desarrollo de grandes aplicaciones. Su filosofía de diseño enfatiza la legibilidad del código y su sintaxis permite al programador expresar los conceptos en menos líneas que en otros lenguajes. Se trata de un lenguaje potente, flexible y con una sintaxis clara y concisa.

Python puede ser utilizado en diversas plataformas y sistemas operativos, entre los que podemos destacar los más populares, cómo Windows, Linux y Mac OSX. Además, dispone de un excelente conjunto de **librerías para extender su funcionalidad**, incluyendo librerías científicas que permiten realizar numerosas tareas de tratamiento de datos, visualización, cálculo numérico y simbólico y otras aplicaciones específicas.

Capítulo 5

Arquitectura

Este capítulo analiza en detalle, mediante un enfoque top-down, la arquitectura diseñada para dar soporte al sistema propuesto en este proyecto. Se discutirá la funcionalidad de cada módulo perteneciente a la arquitectura, se explicará cómo están conectados y cómo fluye la información entre ellos hasta obtener el resultado final.

5.1 Visión general de la arquitectura

La arquitectura diseñada (ver Figura 5.1) es una arquitectura basada en módulos que contribuye a simplificar significativamente el desarrollo del sistema.

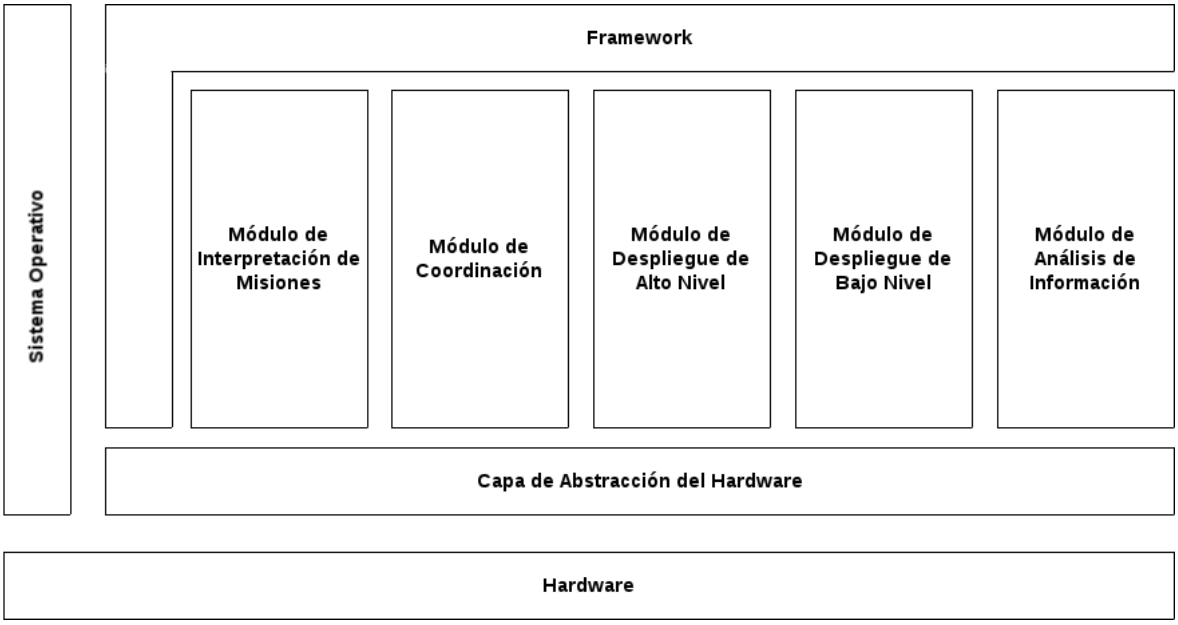


Figura 5.1: Arquitectura del sistema

La arquitectura modular se refiere al diseño de sistemas compuestos por elementos separados que pueden conectarse preservando relaciones proporcionales. La utilidad de la arquitectura modular se basa en la posibilidad de reemplazar o agregar cualquier componente sin afectar al resto del sistema.

Al aplicar la arquitectura modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación.

Un módulo es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida. No debe confundirse el término módulo con términos como función o procedimiento, propios del lenguaje que lo soporte.

En líneas generales, el sistema (ver Figura 5.2) debe ser capaz de extraer «waypoints» de un *Pool de Misiones*, clasificarlos conforme a la prioridad fijada y conseguir que varios drones recorran esos puntos, de forma coordinada, hasta vaciar el *Pool de Misiones*. Así, el sistema reunirá una serie de imágenes, capturadas por los UAVs, que deben ser analizadas.

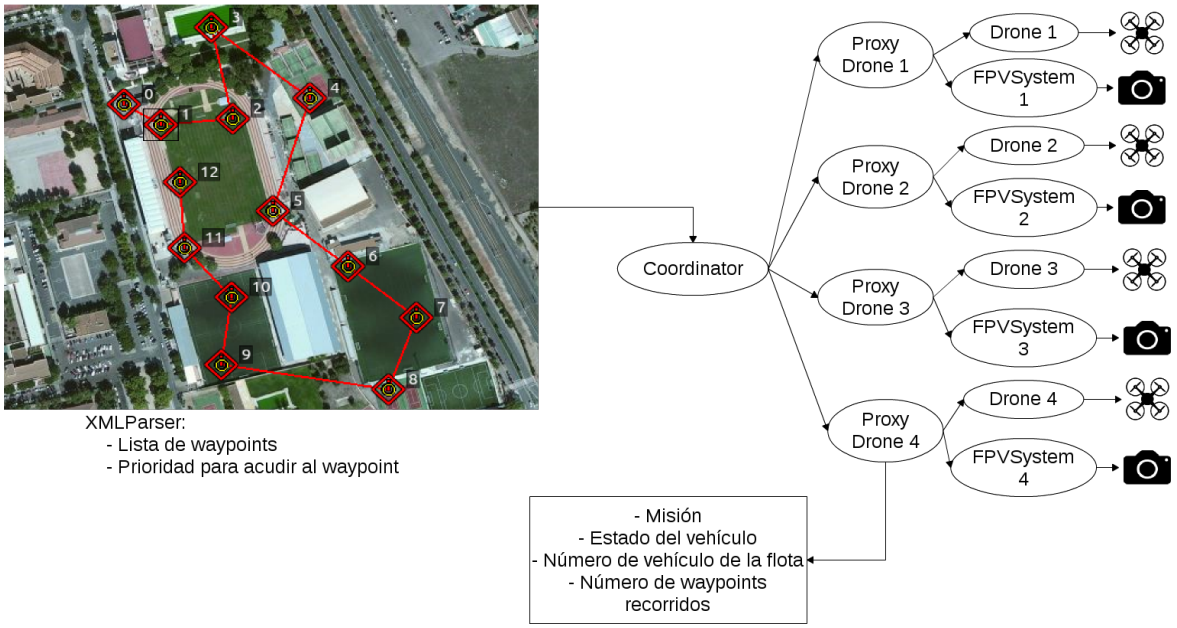


Figura 5.2: Descripción del sistema

En el sistema la información fluye a través de clases (ver Figura 5.3), transformándose hasta alcanzar el resultado final. Esto es, los vehículos aéreos se desplazan de manera coordinada a diversas localizaciones y a su vez retransmiten imágenes, que son analizadas para contribuir a mejorar los tiempos de respuesta en situaciones de emergencia.

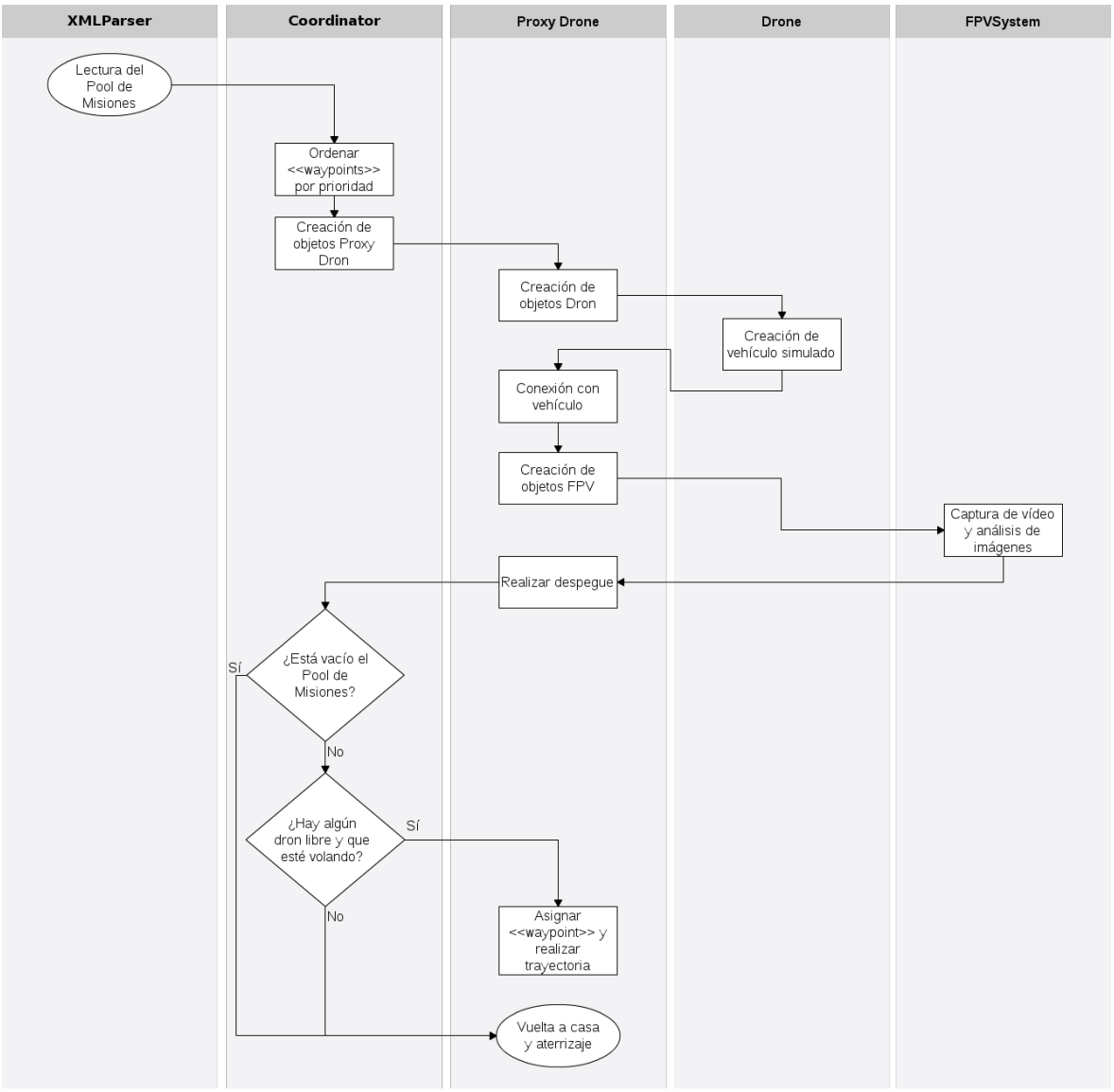


Figura 5.3: Diagrama de flujo del sistema con **un solo dron**

Para ello una iteracción con un solo dron realiza las siguientes tareas:

- En la clase *XMLParser* se leen los archivos XML, desembocando esto en la **creación de un *Pool de Misiones*** que contenga todos los puntos que el UAV debe monitorizar.
- El *Coordinator* se encarga de **ordenar el *Pool de Misiones*** atendiendo a una prioridad extraída de los archivos XML. Además, debe **crear** tantos **objetos del tipo *Proxy Drone*** como se hayan especificado, en este caso uno.
- En la clase *Drone* se obtiene una **simulación del vehículo aéreo**, a través de SITL.
- Una vez que disponemos de la simulación del dron, se debe realizar la **conexión con el dron y con el *FPVSystem***. De esta manera, mientras el UAV realiza la misión correspondiente, retransmite imágenes en directo que son analizadas por una API externa, *Google Cloud Vision* API.

- Por último, el *Coordinator* debe **asignar «waypoints»** hasta vaciar el *Pool de Misiones* a drones que se encuentren en estado ocioso.

La descripción general de la arquitectura (ver Figura 5.1) se elaborará a continuación atendiendo al framework y a los módulos que lo componen.

5.1.1 Framework

Un **framework** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

En este caso, es el proyecto propuesto, el cual consiste en un sistema de vigilancia adaptativo basado en la coordinación de UAVs. Proporciona un conjunto de **servicios, funciones y herramientas** como puede ser el uso de una estación de control de tierra, la lectura de «waypoints», la ordenación de estos por prioridad o el vuelo de un vehículo aéreo.

5.1.2 Módulos desarrollados

Los **módulos** que se han desarrollado son las unidades básicas que el framework es capaz de manejar y de las que depende la solución que se implemente con él. El sistema está compuesto por los siguientes módulos:

- **Módulo de Interpretación de Misiones**: en él se accede a los archivos XML que contienen las referencias de puntos de localización.
- **Módulo de Coordinación**: se ordenan esas localizaciones por la prioridad de acudir a cada una de ellas y se coordina la actividad de los drones para mejorar la productividad del sistema. Asigna misiones a vehículos que no estén ejecutando algún vuelo hacia algún «waypoint». Además, ordena el aterrizaje cuando en el *Módulo de Interpretación de Misiones* no queden misiones por asignar.
- **Módulo de Despliegue de Alto Nivel**: es el módulo que se conecta y comunica con el vehículo, de modo que, actúa como mediador entre el *Módulo de Coordinación* y el *Módulo de Despliegue de Bajo Nivel*. Permite efectuar tareas relacionadas con el dron, así como administrar la conexión con el *Módulo de Análisis de Información*. Por ejemplo, realizar el despegue de un dron.
- **Módulo de Despliegue de Bajo Nivel**: se encarga de crear una simulación de un dron, en una localización específica, para poder realizar pruebas, misiones o cualquier tipo de operación que se podría llevar a cabo con un dron real.
- **Módulo de Análisis de Información**: captura, graba y almacena imágenes, en FPV, del vuelo del vehículo. Las imágenes son analizadas por una API externa de Google, Google Cloud Vision API, que detecta conjuntos de categorías y devuelve una serie de resultados después de realizar una petición HTTP.

Cada módulo representa una o más clases (ver Figura 5.4) en las que se define el comportamiento mediante atributos y funciones. Las clases son instanciadas en objetos, que se corresponden tanto con objetos reales como con objetos internos del sistema, en los que se leen estas definiciones.

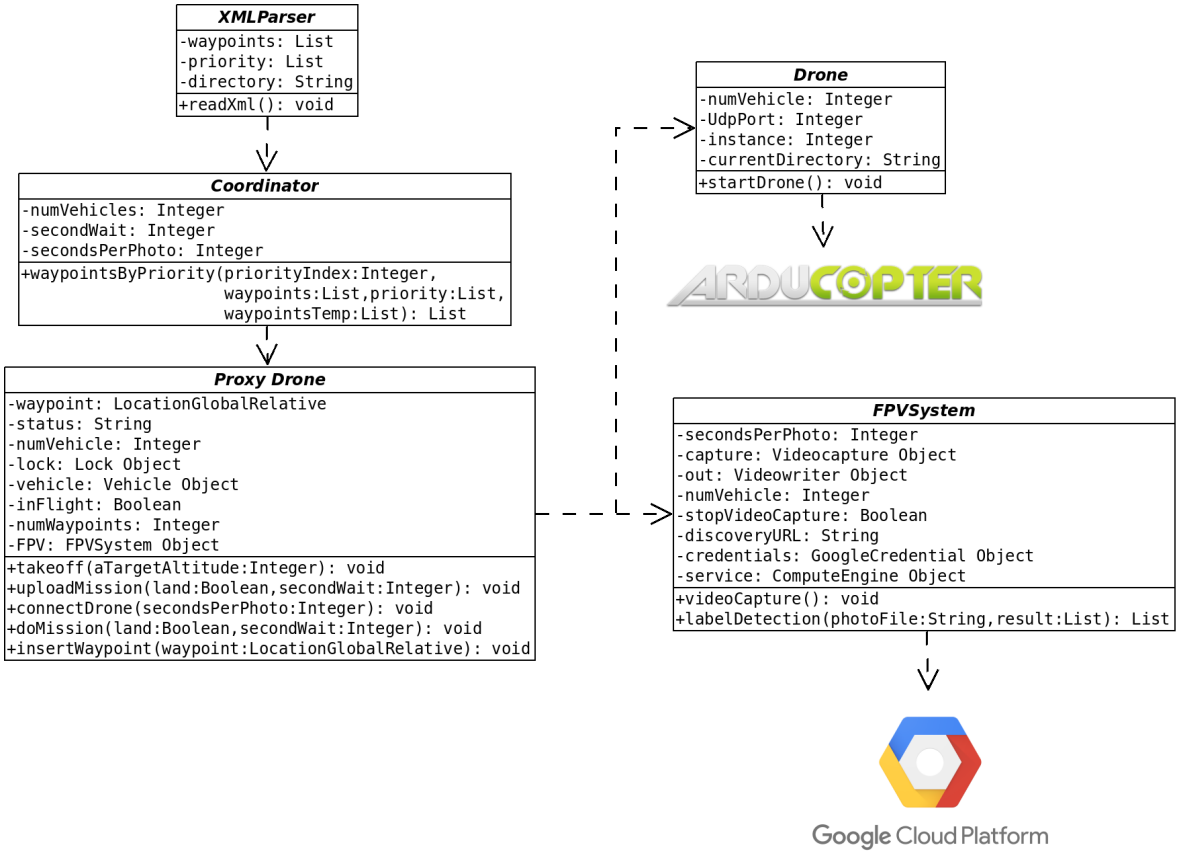


Figura 5.4: Diagrama de clases del sistema

El diseño de las clases del sistema se fundamenta en la **separación de responsabilidades**. La segregación de funciones es un método para separar las responsabilidades de las diversas actividades que intervienen en el sistema, de esta forma, la arquitectura está preparada, por ejemplo, para integrar fácilmente nuevos módulos de análisis a través de otros sensores. También, se debe subrayar que la división entre la clase *Proxy Drone* y las clases *Drone* y *FPVSystem* permite separar las competencias software y hardware, es decir, el *Proxy Drone* se encargará de comunicarse y ordenar tareas a estos dispositivos hardware para así poder cumplir con los objetivos del sistema.

Seguidamente, se expone en detalle cada uno de los módulos desarrollados junto a las clases que forman parte de él.

5.2 Módulo de Interpretación de Misiones

El cometido de este módulo es **leer los archivos XML**. En esta lectura se recoge la información acerca de los puntos de interés. Esto se realiza con el objetivo de **crear una estructura de datos**, denominada *Pool de Misiones*, que contenga todo el conocimiento necesario sobre los «waypoints», formados por una ubicación y una prioridad, a los que los drones deben volar durante la misión.

5.2.1 Clase XMLParser

XML es un **lenguaje de marcas**, desarrollado por el World Wide Web Consortium, utilizado para almacenar datos en forma legible. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

XML se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la **compatibilidad entre sistemas para compartir la información** de una manera segura, fiable y fácil. Entre las ventajas que ofrece el uso de XML se encuentran las siguientes:

- **Separa** radicalmente **la información** o el contenido de su presentación o formato.
- Simplifica compartir e intercambiar datos
- La información **se almacena en texto plano**: software y hardware independiente.
- **Estructura jerárquica**.
- Simplifica los cambios de plataforma
- **Fácilmente procesable** tanto por humanos como por software.

Por esto, se decide hacer uso de archivos XML en los que se especifique la información de los puntos de interés (ver Listado 5.1) que formarán parte de la misión a realizar: **longitud**, **latitud**, **altura** a la que el dron debe volar en dicha posición y la **prioridad** de que el UAV vuele hasta esa localización.

La clase *XMLParser* proporciona una forma de acceder a los datos presentes en los documentos XML. La lectura de estos archivos, que tiene lugar en esta clase, ocasiona la **creación de una estructura de tipo lista**, que denominaremos ***Pool de Misiones***, que contiene una serie de puntos de interés.

```
<?xml version="1.0" ?>
<mission>
        <waypoint>
                <lat>-3.9162183</lat>
                <long>38.9878348</long>
                <alt>30</alt>
                <priority>1</priority>
        </waypoint>

        <waypoint>
                <lat>-3.9181817</lat>
                <long>38.9907202</long>
                <alt>25</alt>
                <priority>2</priority>
        </waypoint>

        <waypoint>
                <lat>-3.9180422</lat>
                <long>38.9877598</long>
                <alt>20</alt>
                <priority>3</priority>
        </waypoint>
</mission>
```

Listado 5.1: Ejemplo de archivo XML que contiene la información de los «waypoints»

5.3 Módulo de Coordinación

Su misión es coordinar la actividad de los drones, durante las misiones, para optimizar la eficiencia del sistema. El cometido de este módulo se divide en tres tareas:

- **Ordenar *Pool de Misiones* por prioridad**: con la prioridad que se ha extraído de los archivos XML, se realiza una clasificación del Pool de Misiones, que contiene todos los «waypoints». A causa de esto, las misiones quedarán diseñadas de tal forma que los drones asistan primero a los puntos de interés cuya prioridad sea superior.
- **Asignar «waypoints»**: las localizaciones son **adjudicadas**, hasta vaciar el *Pool de Misiones*, a drones que se encuentren ***en vuelo*** y que se hallen **en un *estado ocioso***, es decir, a drones que ya hayan realizado el despegue y, además, hayan terminado de volar hacia un punto de interés en el que han recogido las imágenes especificadas.
- **Ordenar el aterrizaje**: cuando el *Pool de Misiones* ha sido vaciado, ordenar la vuelta a casa y el aterrizaje de cada uno de los drones que se estaban empleando en el proceso.

5.3.1 Clase Coordinator

El *Coordinator* es el responsable de llevar a cabo la **coordinación**, **entre los UAVs** que existan disponibles, con el objetivo de poder abarcar el mayor terreno posible en un tiempo que es considerablemente menor. Para llevar a cabo este propósito es necesario contar antes con un *Pool de Misiones* que se encuentre ordenado por la prioridad de los «waypoints».

Una vez que los archivos XML han sido leídos y los «waypoints» han sido almacenados en el *Pool de Misiones*, el *Coordinator* debe **ordenar** esas localizaciones **por medio del campo prioridad**. La prioridad puede tomar tres valores diferentes: 1, 2 y 3. Siendo 1 la prioridad más elevada y 3 la de menor relevancia. De modo que, en el *Pool de Misiones* deben aparecer primero los cuya prioridad sea de 1 y en último lugar los que tengan una prioridad de 3, consiguiendo que los drones lleguen antes a los puntos de interés más importantes.

Con el *Pool de Misiones* ordenado, el *Coordinator* comenzará a asignar «waypoints» a drones que ya hayan realizado el despegue y no se encuentren volando hacia algún otro punto de interés. Cuando el *Pool de Misiones* se encuentre vacío, el *Coordinator* dará la orden de aterrizaje a los drones existentes.

Es crucial que varios drones puedan **efectuar a la vez el vuelo y la captura de vídeo**, por lo que es necesario la **introducción de hilos** en el *Coordinator*, que permitan ejecutar, de manera paralela, estas funciones.

La técnica de multi-hilos permite **desacoplar tareas que no tienen dependencia secuencial**. El trabajo con threads se lleva a cabo mediante el **módulo *threading*** que se apoya en el módulo *thread* para proporcionar una API de más alto nivel, más completa y orientada a objetos.

El desafío principal de las aplicaciones multi-hilo es la coordinación entre los hilos que comparten datos u otros recursos. Con ese fin, el módulo *threading* provee una serie de primitivas de sincronización que incluyen ***locks***, eventos, variables de condición y semáforos.

Los *locks*, también llamados mutex, cierres de exclusión mutua, cierres o candados, son objetos con **dos estados posibles: adquirido o libre**. Cuando un *thread* adquiere el candado, los demás *thread* que lleguen a ese punto posteriormente y pidan adquirirlo se bloquearán hasta que el *thread* que lo ha adquirido libere el candado, momento en el cual podrá entrar otro *thread*.

Para iniciar el sistema, se debe realizar una llamada a la clase ***Coordinator***, la cual cuenta con tres argumentos que son parametrizables al realizar su invocación por consola. Estos atributos parametrizables son los siguientes:

- **numVehicles**: se refiere al número de vehículos que participarán en las misiones y capturarán imágenes en directo del vuelo. Si se omite, el valor por defecto es 1 vehículo.
- **secondWait**: afecta al número de segundos que el vehículo permanecerá parado sobre un «waypoint» para captar imágenes relevantes de esa localización durante ese tiempo. Si se omite, el valor por defecto es de 10 segundos.
- **secondsPerPhoto**: referido al número de segundos que deben transcurrir para tomar una fotografía. Por ejemplo, si el valor es de 5 segundos, el sistema realizará una fotografía cada 5 segundos. Si se omite, el valor por defecto es de 10 segundos.

Un ejemplo de cómo realizar la llamada a *Coordinator* puede ser:

```
$ python coordinator.py --vehicles 2 --secondWait 15 --secondsPerPhoto 7
```

Listado 5.2: Ejemplo de llamada a *Coordinator*

Para conseguir que el sistema funcione, es necesaria la instalación de varias dependencias. Se pueden consultar en el Anexo C.

5.4 Módulo de Despliegue de Alto Nivel

Es el módulo responsable a la hora de realizar las operaciones que conciernen al vehículo, como el despegue o la subida de misiones, y de gestionar la conexión con los dispositivos finales, como el **dron** o el **sistema FPV**. Actúa de intermediario entre el *Módulo de Coordinación* y los módulos de *Despliegue de Bajo Nivel* y de *Análisis de Información*.

5.4.1 Clase Proxy Drone

El ***Proxy Drone*** es el encargado de ejecutar las tareas que tienen que ver con el UAV y de administrar la conexión con los dispositivos finales. Se puede considerar un **intermediario** entre el *Coordinator* y los instrumentos que se encuentran en el extremo del sistema, *Drone* y *FPVSystem*, que son usados para realizar el vuelo y captar imágenes.

Mediante el *Proxy Drone* se resuelve el problema de **guiar el vehículo** a través de una serie de puntos de localización. Gracias a él, el sistema cuenta con una clase que posibilita la **comunicación con el dron** y el **visionado en primera persona**, permitiendo así realizar cualquier tipo de operación como el despegue, la asignación de puntos de interés, la realización de vuelos a dichos puntos o la captación de imágenes que puedan ayudar al personal de emergencia en las labores de identificación de supervivientes o inspección del terreno.

Tan pronto como se haya realizado la lectura de los archivos XML y se cuente con una lista de «waypoints» ordenados por prioridad, se procede a crear objetos de tipo ***Proxy Drone***. Para ello, el *Coordinator* hace uso de su atributo *numVehicles*, creando tantos objetos *Proxy Drone* como número de vehículos se haya especificado.

Entre las competencias de esta clase se encuentran las que se exponen a continuación:

- **Conexión con el dron** y con el **Sistema FPV**: En el caso de ejecutar una simulación del sistema, se instancia un objeto de tipo *Drone*, que creará un vehículo simulado mediante SITL (véase Sección 4.2.2.6). A continuación, se realiza la conexión con el vehículo aéreo, ya sea real o simulado, y por último se crea un objeto *FPVSystem* para iniciar la grabación y el análisis de imágenes.

- **Despegue** del dron: se comprueba que el vehículo cumpla ciertos requisitos antes de armar los motores, como que disponga de una buena señal de GPS o que el nivel de la batería sea superior al 50 %, se configura el modo de pilotaje del dron y se procede al despegue a una altura especificada por parámetro. Cuando el despegue ha terminado, el dron se considera *en vuelo*.
- **Inserción de «waypoints»** y **escritura de misiones**: los «waypoints», que han sido extraídos del archivo XML en el *Coordinator*, son adjudicados a drones que se encuentren en *estado ocioso* y *en vuelo*. La localización de estos puntos debe ser agregada a comandos de misión que puedan ser interpretados por el vehículo. Entre estos comandos se encuentran las órdenes necesarias para: acudir a una localización específica, mantener el vehículo en un punto concreto o realizar la vuelta a casa y el aterrizaje.

El *Proxy Drone* es inicializado pasándole por parámetro el número de vehículo con el que se corresponde dicho objeto. Gracias a esto, el usuario puede obtener **información del estado del dron**, sabiendo si esa información está asociada al *Dron 1*, al *Dron 2*, etc.

5.5 Módulo de Despliegue de Bajo Nivel

Se ocupa de generar una simulación de un UAV, en una ubicación determinada, de modo que se puedan realizar pruebas, misiones o cualquier tipo de operación sin tener que disponer de un vehículo real.

5.5.1 Clase Drone

La clase ***Drone*** ofrece la posibilidad de **simular vehículos aéreos** (ver Figura 5.5), en el caso de este proyecto un quadcopter, haciendo uso de la aplicación SITL.

El uso de SITL (véase Sección 4.2.2.6) permite crear vehículos simulados que pueden usarse como un vehículo real: se puede conectar al vehículo usando una estación de control de tierra, despegar, cambiar los modos de vuelo, realizar misiones guiadas o automáticas, y aterrizar. La diferencia principal es que además de ser capaz de configurar el vehículo, los parámetros del simulador permiten configurar el entorno físico (por ejemplo, la velocidad y dirección del viento) y también simular el fallo de distintos componentes. Esto significa que SITL es el entorno perfecto para probar correcciones de errores, modos de fallo y aplicaciones desarrolladas mediante DroneKit-Python (véase Sección 3.4.4).

El cometido que tiene esta clase es, dependiendo del número de vehículo que lo instancie, acceder a una carpeta determinada y **ejecutar el comando** (ver Listado 5.3) **para generar la simulación** del dron, en un puerto UDP concreto, al que posteriormente el sistema se conectará para comunicarse con él.

Figura 5.5: Simulación de dron mediante SITL

```
$ sim_vehicle.sh -I %d -L prueba --map --out 127.0.0.1:%d --aircraft test
```

Listado 5.3: Comando para generar una simulación de un vehículo mediante SITL

El comando está formado por unos parámetros que representan lo siguiente:

- **-I**: el número de instancia para permitir múltiples copias del simulador funcionando a la vez.
- **-L**: ubicación inicial, que es configurada en un archivo de tipo texto denominado *locations.txt*.
- −−**map**: posibilita disponer de una ventana con un mapa, para mostrar el progreso del dron al realizar las misiones.
- −−**out**: dirección en la que se podrá realizar la conexión con el dron. Está formada por una dirección IP y un puerto UDP.

5.6 Módulo de Análisis de Información

Capta, registra y guarda imágenes de vídeo del vuelo del vehículo. Realiza capturas de pantalla, del vídeo en FPV, que son analizadas por Google Cloud Vision API, que es capaz de detectar grupos de categorías y retornar una cadena de resultados.

5.6.1 Clase FPVSystem

Los objetos de la clase ***FPVSystem*** facilitan la captura de imágenes en FPV y el análisis de estas.

Las tareas que se desarrollan en esta clase son fundamentalmente tres:

- **Grabación de vídeo**: el sistema graba y almacena un vídeo, haciendo uso de *OpenCV* (véase Sección 3.4.5), que facilitará a los equipos de emergencia la visión del terreno, para así, poder tomar unas decisiones mejores y más correctas.
- Realización de **captura de pantalla**: cada un cierto número de segundos, especificado al realizar la llamada al *Coordinator*, se llevará a cabo una captura de pantalla que será almacenada en el computador y que será utilizada para realizar el análisis.
- **Análisis de imágenes**: haciendo uso de Google Cloud Vision API (véase Sección 3.4.6) se analizarán las capturas de pantalla realizadas anteriormente. Google Cloud Vision API, mediante la detección de etiquetas, detecta amplios conjuntos de categorías y arroja resultados de los elementos que se encuentran dentro de las fotografías. Estos resultados serán imprimidos y mostrados en el vídeo que se está grabando.

Para llevar a cabo el análisis de imágenes es necesario realizar una **petición HTTP a Google Cloud Vision API** en la que se especifique el tipo de análisis que se quiere realizar, en este supuesto *Detección de etiquetas*, y el número máximo de resultados a obtener.

Un ejemplo de esta petición puede ser la especificada a continuación:

```
with open(photoFile, 'rb') as image:
        image_content = base64.b64encode(image.read())
    service_request = self.service.images().annotate(body = {
        'requests': [{
                'image': {
                        'content': image_content.decode('UTF-8')
                                },
                        'features': [{
                                'type': 'LABEL_DETECTION',
                                'maxResults': 5
                        }]
        }]
    }]


    response = service_request.execute()
        responseLength = len(response['responses'][0])
```

Listado 5.4: Ejemplo de petición a Google Cloud Vision API para la detección de etiquetas

Haciendo uso de dicha función, se obtiene el análisis de una imagen que arroja como resultado cinco etiquetas.

Figura 5.6: Ejemplo de captura y análisis de imagen

Capítulo 6

Evolución, resultados y costes

En este capítulo se hará un repaso completo a la evolución del proyecto, destacando cada uno de los problemas que han aparecido durante el periodo de desarrollo, así como, la solución que se ha propuesto. De igual manera, se discutirán los resultados que se han obtenido al plantear dos casos de estudio para comprobar el funcionamiento del sistema. Por último, se realizará un desglose de los costes que ha tenido la elaboración del TFG.

6.1 Evolución

En Julio de 2015, el artífice de este proyecto comenzó a concentrarse, desde el grupo de investigación AIR, en el **desarrollo de un sistema de coordinación de UAVs en situaciones de catástrofe**.

Cabe destacar la **complejidad de trabajar, desarrollar y depender de varios componentes hardware**. Como se muestra seguidamente, en varias fases del proyecto ocurre algún tipo de inconveniente, por lo que valerse de una metodología ágil como la **Programación Extrema** (véase Sección 4.1.2) es crucial. Este tipo de metodología no sigue un plan cerrado, sino que se adapta a los cambios que surjan a lo largo del desarrollo. Por ejemplo, en ocasiones es necesario adquirir repuestos nuevos y, por lo tanto, los tiempos de desarrollo de una iteración pueden incrementarse.

A continuación, se disgregan en detalle cada una de las fases o periodos de desarrollo del proyecto (ver Figura 6.1).

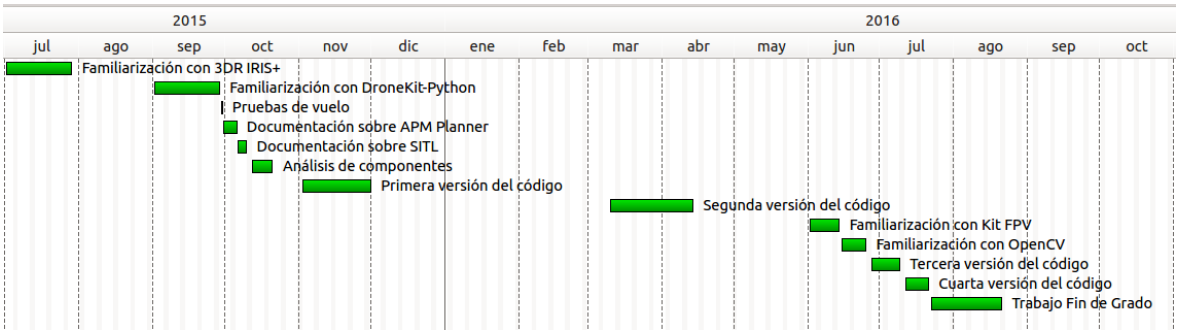


Figura 6.1: Diagrama de distribución de tiempo por tareas en el TFG

6.1.1 Fase 0: Julio 2015 - Octubre 2015

En esta primera etapa, el autor debió **familiarizarse** con el componente fundamental de este proyecto, el dron 3DR IRIS+ y su entorno de desarrollo DroneKit-Python. Se realizaron **pruebas de simulación**, con código de ejemplo proporcionado por 3D Robotics. También, se hicieron **pruebas de vuelo** automáticas, en el Instituto de Tecnologías y Sistemas de Información, con el 3DR IRIS+ y la estación de control de tierra APM Planner.

Se profundizó en el estudio de APM Planner y SITL, llevando a cabo documentos que sirvieran para dar los primeros pasos en estas plataformas.

Como última parte de esta primera etapa, se realizó un **análisis de los componentes que se necesitan** y que debían comprarse para llevar a cabo el desarrollo del proyecto. Entre otros se acordó la compra de una cámara GoPro HERO y un kit FPV.

Entre los **problemas producidos** durante esta etapa se encuentra:

- **Sistema radiocontrol FlySky FS-TH9X defectuoso**: la palanca correspondiente al *pitch* y al *roll*, con la que se le indica al dron la dirección a la que debe moverse, llegó dañada en el envío. Esto provocaba que el dron ejecutara siempre vuelos hacia la izquierda.
  La **solución propuesta** fue reenviar el sistema radiocontrol a 3D Robotics para ser reemplazado. El envío se realizó en Septiembre y no fue recibido de nuevo hasta Noviembre, lo que supuso dos meses sin poder trabajar con el 3DR IRIS+.
- **Frecuencia errónea en la telemetría**: el 3DR IRIS+ cuenta con dos modelos disponibles: uno para el uso en Estados Unidos y otro para el uso en el resto del mundo. La diferencia, entre uno y otro, es la frecuencia a la que funciona la telemetría, ya que en Estados Unidos la banda que utiliza es 915 mHz y en el resto del mundo 433 mHz. En este caso, el dron fue adquirido por error con la banda de frecuencia de Estados Unidos, que en España está ocupada por el sistema de telefonía móvil GSM. Esto provocaba una serie de interferencias, en el sistema de telemetría, que desembocaban en una lentitud considerable a la hora de cargar los parámetros del 3DR IRIS+ en la estación de control de tierra. Además, no era posible llevar a cabo la conexión con el dron real mediante esta telemetría.
  La **solución** a este problema fue comprar una nueva telemetría que funcionara a 433 mHz.

6.1.2 Fase 1: Noviembre 2015 - Febrero 2016

Meses más tarde, en Noviembre de 2015 se comenzó a desarrollar la **primera versión del código**. En esta primera versión, se diseñó el sistema, mediante reuniones con David Vallejo, y se implementó el esqueleto de código que ha sido utilizado y mejorado a lo largo del proyecto.

La primera versión del código contaba con el *Coordinador*, el *Proxy Dron* y el *Dron*. En él, los «waypoints» eran ordenados por prioridad, divididos en listas, que dependían del número de drones que hubiera disponibles, y asignados al vehículo que le correspondiera. Es decir, no se realizaba ningún tipo de coordinación, simplemente todos los puntos de interés eran repartidos, respetando la prioridad, entre él número de vehículos existentes.

Durante el mes de Diciembre y el mes de Enero, se procedió a **pausar el proyecto** debido a los exámenes finales del primer cuatrimestre.

Entre los **problemas producidos** durante esta etapa se encuentra:

- **Hilos accediendo a recursos compartidos**: al hacer uso de hilos y ejecutar misiones con más de un dron, los vehículos se creaban en el mismo puerto UDP. Este inconveniente ocasionaba que todos los vehículos volaran hacia los mismos puntos de interés, haciendo inservible el empleo de más de un dron en el sistema.
  La **solución** a este problema fue utilizar *locks*, como método de sincronización, en el momento en el que se produce la conexión con el dron.

6.1.3 Fase 2: Marzo 2016 - Mayo 2016

En la segunda versión, se modificó la manera en que las misiones son asignadas, puesto que se introdujo el concepto de **coordinación entre drones**. Se decidió que los puntos de interés se adjudicaran a vehículos que son detectados en un estado ocioso o desocupado. De esta manera, se consiguió mejorar la productividad y el rendimiento del sistema, debido a que anteriormente un vehículo que terminaba de recorrer los «waypoints» que se le habían asignado, no colaboraba con el resto de drones en la misión.

Además se **cambió el orden en el que se ejecutaban las operaciones de asignación de puntos de interés y de conexión** con el vehículo. Antes, los puntos de interés eran asignados a vehículos aun no conectados, lo que ocasionaba que si la conexión a un vehículo, al que se le había designado un punto de interés de alta prioridad, tardaba en realizarse, este punto de interés podría ser pasado por alto por otro vehículo cuya conexión haya sido más rápida. De forma que, se estaría incumpliendo el modelo de prioridades del sistema.

Durante el mes de Mayo se volvió a **detener el desarrollo del proyecto** a causa de los exámenes finales del segundo cuatrimestre.

Entre los **problemas producidos** durante esta etapa se encuentra:

- **Rotura de las patas del dron**: al efectuar el aterrizaje, tras uno de los vuelos de prueba, se parte una de las patas del dron. Esto imposibilita el uso del dron hasta tener piezas de repuesto, dado que el vehículo no dispone de una plataforma segura para el aterrizaje.
  La **solución** a este problema fue comprar un pack de patas de repuesto para el 3DR IRIS+, pero en vista de que los pedidos de 3D Robotics tardan en llegar más de un

mes, se decidió hacer uso de la impresora 3D, que existe en la Escuela Superior de Informática, para fabricar cuatro patas nuevas que permitieran volar el dron mientras los repuestos eran enviados.

6.1.4 Fase 3: Junio 2016 - Septiembre 2016

Considerando que la coordinación de UAVs en situaciones de emergencia ha sido resuelta, con las versiones desarrolladas previamente, en esta etapa se procedió a incorporar los procedimientos para poder **capturar imágenes** que posteriormente sean **analizadas**.

Lo primero fue **familiarizarse con el kit FPV y la biblioteca OpenCV**. Ejemplos de código, proporcionados al instalar OpenCV, fueron ejecutados y se grabaron vídeos de prueba hasta comprender que funciones eran necesarias para añadir esta funcionalidad al código del proyecto.

Se realizó una tercera versión, en la que entra en juego el *Sistema FPV*, en la que se grababan las imágenes que procedían de la cámara equipada en el vehículo. Adicionalmente, se estipuló parametrizar el periodo de tiempo, en segundos, cada el que se toma una captura de pantalla de la grabación.

Por último, para realizar la cuarta y última versión del sistema fue necesario entender el funcionamiento de Google Cloud Vision API. En la última versión del sistema, se integró el módulo de análisis de imágenes, en el que las capturas de pantalla son examinadas y se muestran por pantalla los resultados obtenidos.

Se decide entregar el TFG en Septiembre, por lo que, el mes de Julio y Agosto fue dedicado a documentar el proyecto. Con la entrega del TFG el proyecto se considera cerrado en Septiembre de 2016.

Entre los **problemas producidos** durante esta etapa se encuentra:

- **Placa de Tarot T-2D Gimbal estropeada**: mientras se llevaban a cabo pruebas de vuelo, en las que se recogían imágenes, la placa de la gimbal resultó dañada. Se trató de un problema grave, puesto que, no es posible conseguir una grabación de vídeo estable sin ella.
  La **solución** a este problema fue comprar una nueva placa, junto con unos nuevos motores estabilizadores.
- **Baterías para el sistema FPV deterioradas**: para que el sistema FPV funcione, es necesario contar con dos baterias de 1300 mAh, una para el transmisor y otra para el receptor. Dos de las tres baterías de las que se disponía, sufrieron desperfectos y dejaron de cargar. Por lo tanto, el sistema FPV quedó inoperativo.
  La **solución** a este problema fue comprar nuevas baterías de 1300 mAh.

6.2 Casos de estudio

Se plantean dos casos de estudio para probar el funcionamiento del sistema.

El primero de ellos servirá para **poner a prueba el método de coordinación entre UAVs**, para ello, se propone una situación de emergencia en la que varios drones deben cubrir, de forma conjunta, el escenario que ha sido afectado por un ataque terrorista. Al disponer de un solo vehículo aéreo, el caso se lleva a cabo mediante la ejecución de SITL.

El segundo caso de estudio se realizará para comprobar la **operatividad completa del sistema**. Esto se debe a que, este caso de estudio se efectuará con el único dron disponible, el 3DR IRIS+. Se diseñará una misión en un entorno en la que el UAV pueda capturar y analizar imágenes de relevancia, verificando así que el sistema es capaz de cumplir con los objetivos fijados.

Como se observa, el planteamiento propuesto consiste en **primero simular** la funcionalidad del sistema y **después ejecutar una prueba real** con el vehículo aéreo. Este enfoque, de simular antes para posteriormente realizar una prueba real, se ha seguido a lo largo de todo el desarrollo del proyecto. Gracias a esto se consigue:

- **Experimentar** con código implementado **sin poner en peligro la infraestructura** hardware del proyecto.
- **Probar el código** del sistema sin hacer uso del 3DR IRIS+, lo que permite desarrollar y comprobar el funcionamiento del código **en cualquier lugar**, además de evitar posibles daños en el vehículo.
- **Coordinar** misiones, con tantos vehículos como se desee, sin necesidad de poseer una flota de drones.

En conclusión, la mayor ventaja, de tener la posibilidad de ejecutar el sistema mediante simulaciones de UAVs, es tener la seguridad y la certeza, a la hora de realizar pruebas con el vehículo real, de que el código se ha desarrollado correctamente y su ejecución no debería causar problemas.

Por el contrario, **la simulación no permite la captura y el análisis de imágenes**, por lo que, para probar esta funcionalidad no queda más remedio que usar el 3DR IRIS+.

6.2.1 Caso de estudio 1: coordinación de UAVs en estado de emergencia

Para probar y **demostrar la funcionalidad del sistema**, se ha diseñado una misión en la que varios UAVs puedan coordinarse y, de este modo, ofrecer una solución al problema de monitorizar diferentes puntos de un territorio.

Como solo se dispone de un dispositivo de vuelo real, se ha hecho uso de la **simulación de tres vehículos aéreos**, a través de SITL, que llevarán a cabo labores de coordinación para inspeccionar de una forma más rápida el lugar.

Recordar que, al tratarse de vehículos simulados, el ***Módulo de Análisis de Imágenes*** **no estará activo**.

6.2.1.1. Planteamiento

El problema que se plantea es el siguiente: la **monitorización de distintos puntos** de interés dentro de una **instalación deportiva que ha sufrido un ataque terrorista**. La localización elegida, por cercanía y por familiaridad con el entorno, es el Polideportivo Juan Carlos I (ver Figura 6.2). Sus instalaciones cuentan con:

- 3 campos de fútbol, con un aforo total de **4.500 personas**.
- 3 pistas polideportivas descubiertas.
- 1 Pabellón cubierto, con capacidad para **500 personas**.
- 7 pistas de pádel.
- 2 pistas de tenis.
- 2 pistas de frontón.
- 1 piscina cubierta, con aforo para **250 personas**.
- 1 piscina descubierta.

Figura 6.2: Polideportivo Juan Carlos I

En definitiva, se trata de una ubicación en la que, en un momento dado, puede existir una **gran confluencia de gente**. Por lo que, tras un posible ataque terrorista es adecuada la utilización de drones, para recolectar información del entorno que sea de utilidad para los equipos de emergencia.

6.2.1.2. Diseño de la solución

Para cubrir el mayor terreno posible en el menor tiempo, se siguen estas pautas:

- Se fijan tres puntos, en los extremos de la instalación deportiva, con una prioridad de 1. Con esto lo que se pretende, es que los drones acudan **primero a las ubicaciones más lejanas** para que después vayan cerrando, de forma coordinada, el radio de actuación. En caso contrario, un dron podría tener que recorrer una gran distancia, haciendo el sistema menos eficiente.
- Se fijan tres puntos, cercanos a las ubicaciones anteriores, con una prioridad de 2. De este modo, se consigue que el **radio de actuación se estreche**, posibilitando que, posteriormente, al resto de puntos pueda volar cualquier vehículo sin que esté demasiado lejos.
- El resto de puntos se fijan con una prioridad de 3. Los vehículos vuelan a una distancia cercana entre ellos, por lo tanto, el vuelo a **cualquier ubicación no supondrá una pérdida de tiempo elevada**.

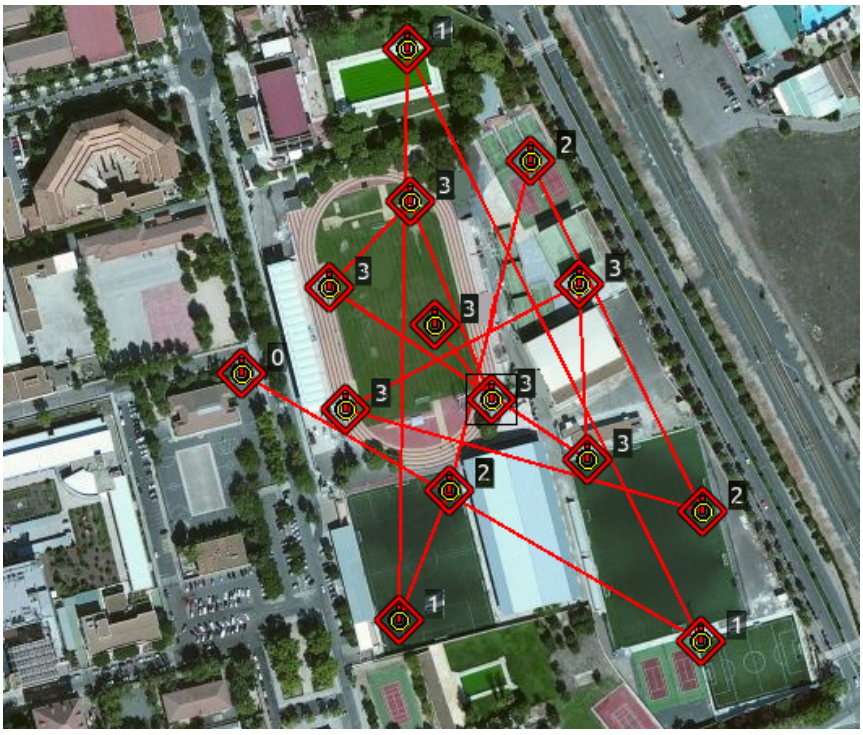


Figura 6.3: Diseño de la misión en el Polideportivo Juan Carlos I

6.2.1.3. Ejecución de la solución

Para comenzar el vuelo de los vehículos, que van a ejecutar la misión diseñada, se debe **llamar por línea de comandos a *Coordinator.py*** con el parámetro de vehículos a 3 (ver Listado 6.1).

```
$ python coordinator.py --vehicles 3
```

Listado 6.1: Inicio del sistema para simulación de tres vehículos

Al realizar está llamada, el sistema desplegará tres ventanas (ver Figura 6.4), que se corresponden con los vehículos simulados, mientras que en la terminal de Ubuntu se ofrece información del sistema y de los drones como, por ejemplo:

- Creación de objetos del sistema.
- Realización del despegue.
- Llegada a puntos de interés.
- Realización del aterrizaje.
- Puntos de interés visitados por cada UAV.

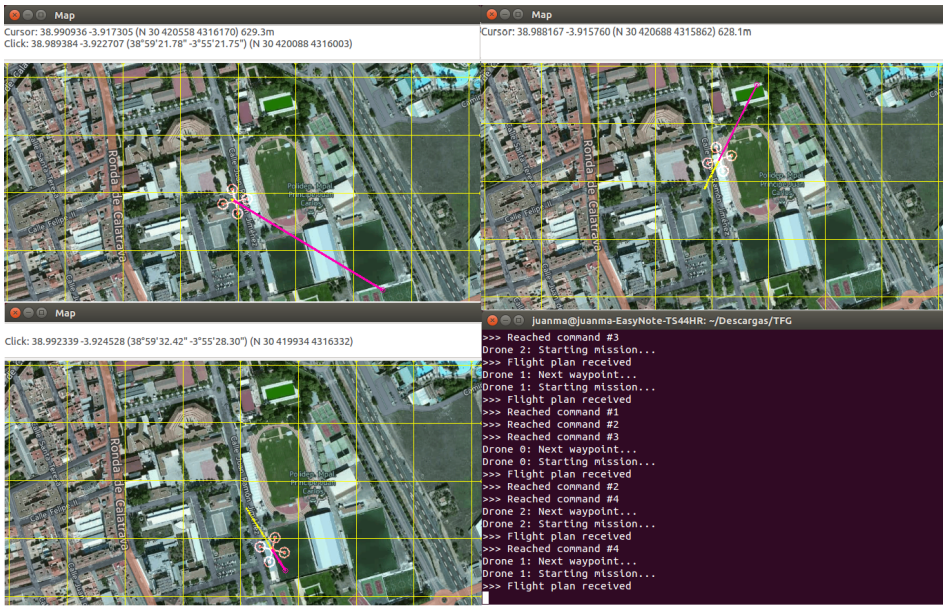


Figura 6.4: Despliegue del sistema con tres drones

El sistema seguirá la solución que se ha diseñado, de tal forma que el **dron 1 volará a cuatro puntos de interés** (ver Figura 6.5).

El **dron 2 también visitará cuatro puntos de interés** antes de volver a la zona de despegue (ver Figura 6.6).

Por último, el **dron 3 acudirá a cinco puntos de interés** (ver Figura 7.7).

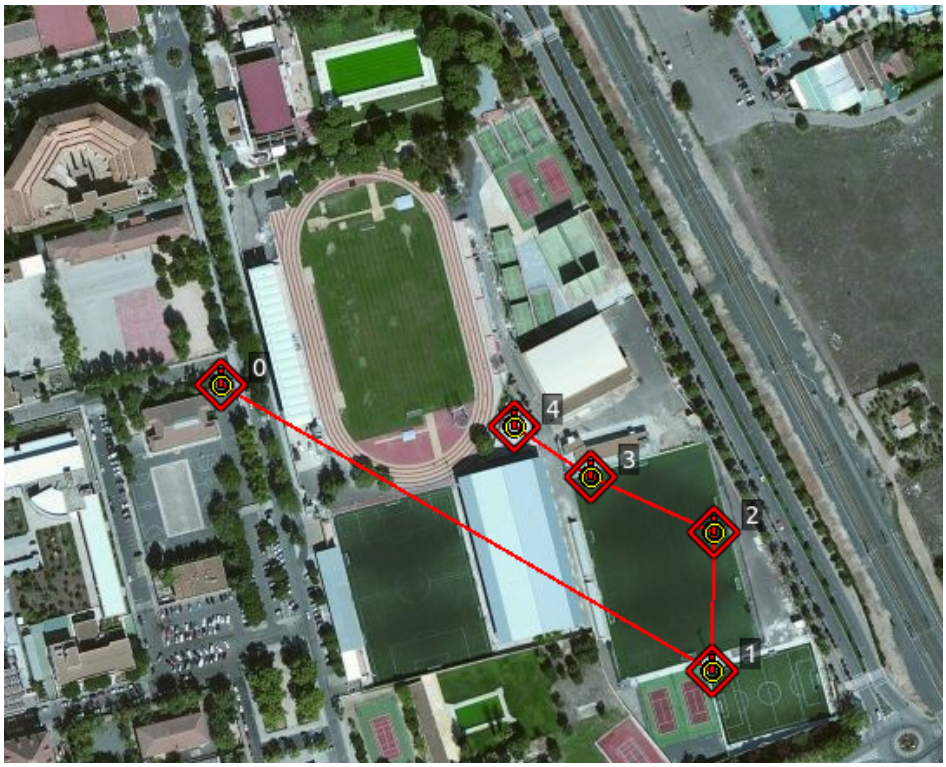


Figura 6.5: Misión realizada por el dron 1

Figura 6.6: Misión realizada por el dron 2

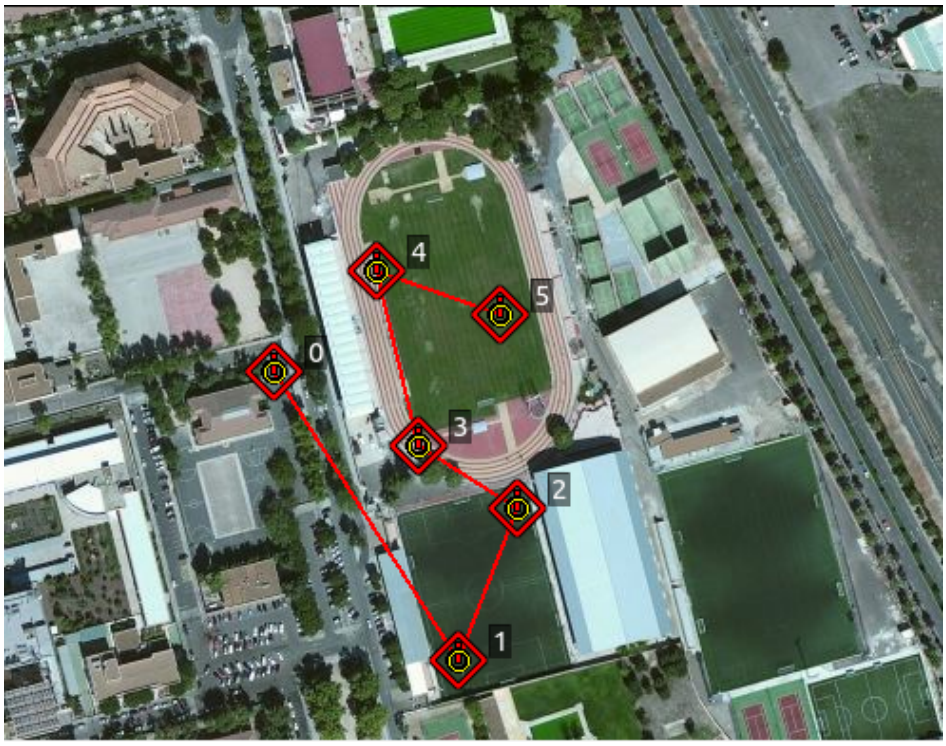


Figura 6.7: Misión realizada por el dron 3

6.2.1.4. Conclusión del caso de estudio

Después de la ejecución de la solución haciendo uso del sistema desarrollado, se obtienen una serie de conclusiones:

- Se confirma que la coordinación de varios UAVs es una funcionalidad fundamental en situaciones de catástrofe. Utilizando tres drones, el tiempo para recorrer todos los puntos de la misión ha sido de 4 minutos y 30 segundos. Esto quiere decir, que en menos de 5 minutos los equipos de emergencia podrían haber dispuesto de información vital acerca del escenario. En cambio, haciendo uso de un solo vehículo, el tiempo para recorrer los puntos de interés por prioridad asciende hasta algo más de 13 minutos. Así pues, gracias al despliegue de varios drones se consigue **reducir el tiempo de actuación en un tercio**.
- Es **esencial un buen diseño a la hora de crear la misión**. Una misión en la que un dron acuda a un punto de prioridad, que se encuentra lejos de su ubicación, existiendo uno más cercano, de la misma prioridad, convierte el sistema en menos eficiente. Por ello, es primordial fijar los puntos de interés en un buen orden de ejecución. Por ejemplo, en el supuesto del caso de estudio, cerrando poco a poco el radio de actuación.

6.2.2 Caso de estudio 2: ejecutando el sistema sobre el 3DR IRIS+

Para verificar y **confirmar que el sistema cumple con los objetivos propuestos**, se creará una misión con varios «waypoints» en la que, usando el vehículo aéreo 3DR IRIS+, el sistema haga volar el dron hasta cada uno de ellos y consiga capturar y analizar las imágenes tomadas desde el UAV.

En este caso de estudio se demostrará el resultado real, al ejecutar una misión con el 3DR IRIS+, del sistema desarrollado en este TFG.

6.2.2.1. Planteamiento

Al hacer uso del 3DR IRIS+, cuyo peso no excede los 25 kg, se deben seguir una serie de **normas establecidas en la Normativa Española de Seguridad Aérea** (ver Anexo A):

- Operar en zonas **fuera de aglomeraciones**, de edificios y personas, en ciudades, pueblos o lugares habitados.
- Operar a una **altura** sobre el terreno **no mayor de 120 metros**.
- Volar **dentro del alcance visual** del piloto, a una distancia de éste no mayor de 500 metros

Siguiendo esta reglas, la misión se proyecta sobre una parcela desocupada, de personas e infraestructuras, a las afueras de Ciudad Real (ver Figura 6.8).

Figura 6.8: Extensión donde ejecutar el caso de estudio

La ubicación elegida cumple con la Normativa Española de Seguridad Aérea, pero debido a este reglamento tan estricto, es **difícil** poder **capturar imágenes de cierta relevancia** que sean analizadas por *Google Cloud Vision* API.

6.2.2.2. Diseño de la solución

Puesto que en este caso de estudio no es necesario coordinar varios vehículos, siendo lo realmente importante que el **3DR IRIS+ vuele a los «waypoints» mientras recoge y analiza imágenes**, simplemente se fijarán tres puntos de interés en la zona antes mencionada, cada uno de una prioridad distinta, para que el 3DR IRIS+ acuda a cada uno de ellos y capture imágenes del entorno.

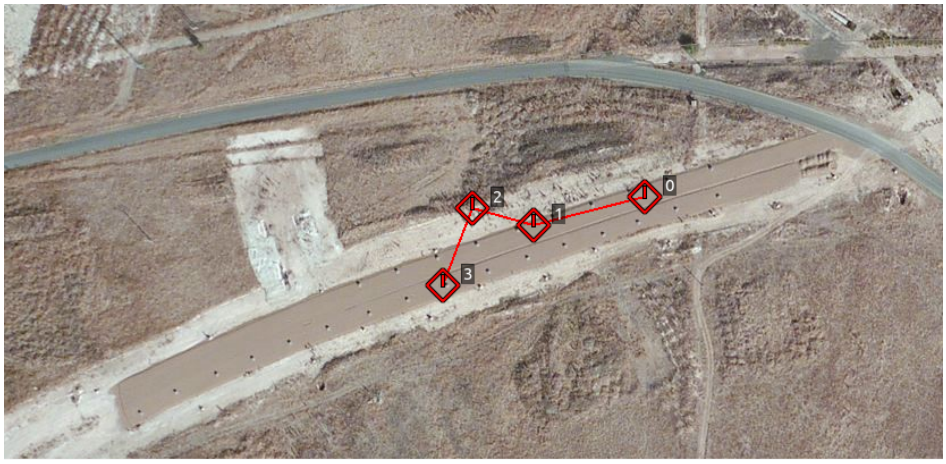


Figura 6.9: Diseño de la misión del caso de estudio 2

6.2.2.3. Ejecución de la solución

Antes de iniciar la llamada al sistema, se deben **verificar ciertos aspectos** como: haber conectado la telemetría, para así, ser capaces de comunicarse con el dron o asegurarse de que el mando radiocontrol esté operativo, debido a que, en caso de que el UAV tenga algún fallo se pueda ordenar inmediatamente la vuelta al punto de despegue, evitando así daños en el aparato.

Para iniciar el sistema, que debe llevar a cabo la misión diseñada, se debe **llamar por línea de comandos a *Coordinator.py***, en este caso con el parámetro de vehículos a 1, el parámetro de espera en el punto de interés a 20 y el parámetro de segundos entre fotos a 5 (ver Listado 6.2). De este modo, se conseguirá que el sistema despliegue un solo vehículo que realizará y analizará una foto cada 5 segundos, esperando en los «waypoints» especificados un tiempo de 20 segundos.

```
$ python coordinator.py --vehicles 1 --secondWait 20 --secondsPerPhoto 5
```

Listado 6.2: Inicio del sistema para caso de estudio 2

Cuando se realiza está llamada, el sistema desplegará tres ventanas (ver Figura 6.10): la primera se corresponde con el 3DR IRIS+, la segunda con el sistema FPV que muestra y analiza las imágenes y la tercera es la terminal de Ubuntu que ofrece información del sistema y del UAV.

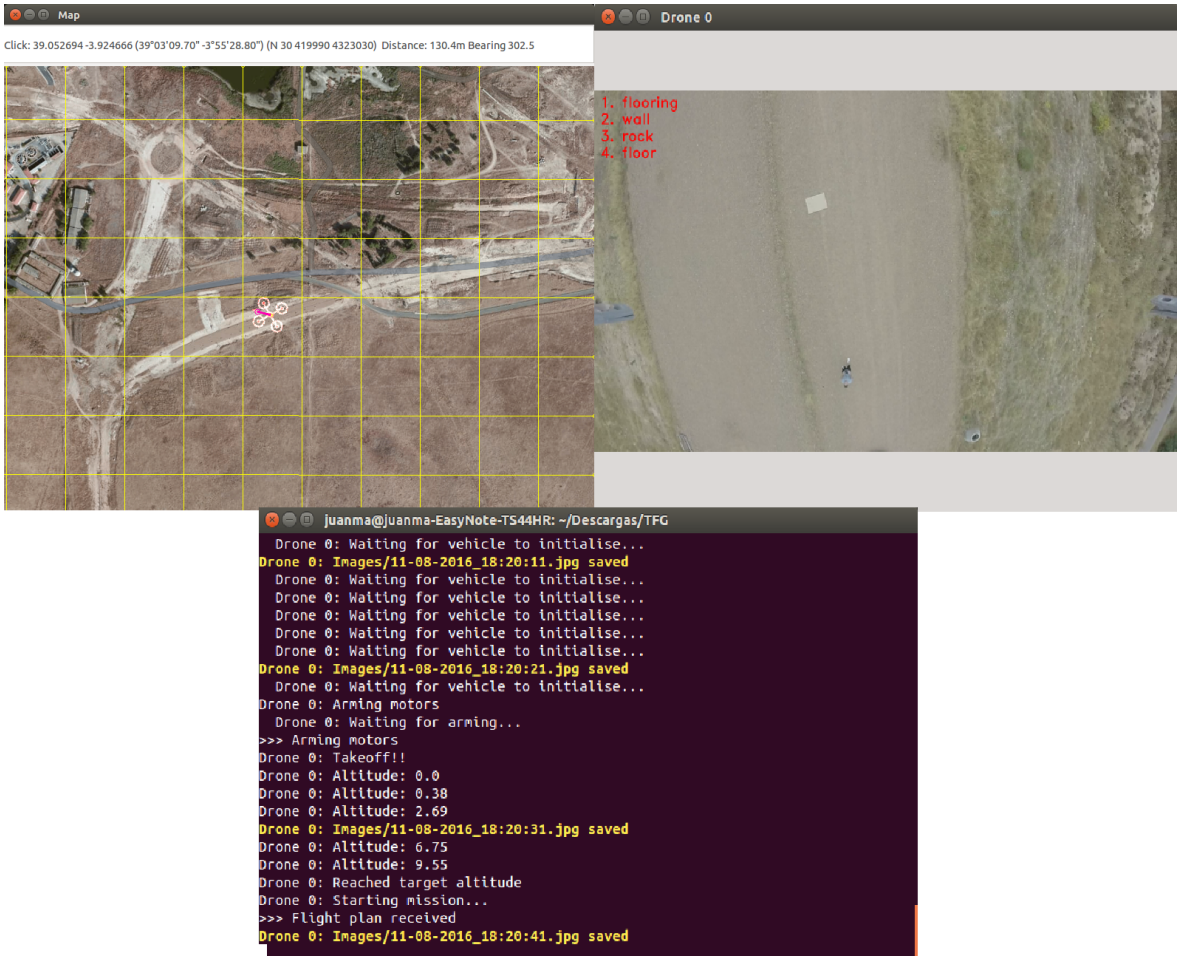


Figura 6.10: Despliegue del sistema con tres drones

El sistema desarrollado hará despegar el 3DR IRIS+ para que vuele hasta los puntos de interés (ver Figura 6.11), que se han determinado en el diseño de la misión, de tal forma que a lo largo del vuelo grabará y analizará las imágenes que está capturando (ver Figura 6.12).

El resto de imágenes de la ejecución del caso de estudio se encuentran en el Anexo D. Debido a las limitaciones que impone la Normativa Española de Seguridad Aérea, como tener que operar en zonas deshabitadas, es complejo poder obtener imágenes de cierta relevancia. Es por ello que, en el Anexo D también se muestran imágenes extraídas de internet, de manera que, el *Módulo de Análisis de Imágenes* pueda ser probado con imágenes que contienen mayor información.

[1]Para una mejor comprensión del vuelo del vehículo, se muestran capturas de pantalla de Mission Planner

Figura 6.11: 3DR IRIS+ volando a «waypoint» [1]

Figura 6.12: Imagen capturada y analizada por el sistema

6.2.2.4. Conclusión del caso de estudio

Tras haber ejecutado en el sistema el segundo caso de estudio, las conclusiones obtenidas son:

- La **captura de imágenes aéreas debe ser un pilar fundamental en situaciones de catástrofe**. La posibilidad de contar con un UAV que pueda inspeccionar de manera rápida, y coordinada, una porción del terreno afectado por un desastre, proporciona grandes ventajas a los equipos de emergencia. Gracias a la información que proporcionan estas imágenes, los equipos de emergencia pueden localizar a supervivientes y elaborar un plan de rescate mucho más apropiado y en un tiempo menor.
- El **análisis de imágenes puede llegar a ser una funcionalidad vital** del sistema. Actualmente, el *Módulo de Análisis de Imágenes* está enfocado a proporcionar infor-

mación adicional, de tal forma que, sirva como ayuda al personal de rescate. Esto se debe a que, *Google Cloud Vision* API fue lanzado en versión beta en febrero de 2016, de modo que, los resultados que proyecta tienen un margen de mejora grande. Quizás en un futuro muy lejano, la detección de supervivientes se pueda realizar de manera automática mediante el uso de esta API.

6.3 Costes

Es complicado dar un valor exacto al coste del proyecto, a causa de que los ciclos de trabajo no han sido constantes durante todo el desarrollo del sistema. Varias circunstancias han contribuido a que se detenga o se aminore, en varias ocasiones, la actividad del proyecto. Algunos de estas circunstancias son:

- Fases en las que se debía esperar la llegada de algún repuesto para poder hacer uso del sistema.
- Periodos de exámenes.
- Fechas en las que se debían entregar prácticas y trabajos de distintas asignaturas.
- Compaginación con contrato deportivo.

Por esta razón, se hará una estimación en la que se tase el coste del proyecto atendiendo a las siguientes pautas:

- Sueldo de un programador en Python: el salario se encuentra establecido en **20 euros por hora**.
- El proyecto ha supuesto 13 meses de desarrollo, de los cuales se atribuyen **6 meses como productivos** al descontar los periodos de exámenes, las vacaciones y la espera hasta la llegada de repuestos.
- Se establece un horario laboral de **37,5 horas semanales**.
- El coste de los dispositivos hardware (ver Cuadro 6.1) necesarios para elaborar el TFG: la suma de la compra de todos los componentes asciende a **1.845,21 euros**.

Así pues, **el coste del proyecto se estima en 19.845,21 euros**.

Componente	Cantidad	Precio/Unidad	Precio Total
3DR IRIS+	1	660,00 €	660,00 €
Maleta para 3DR IRIS+	1	250,57 €	250,57 €
GoPro HERO 3+ Silver Edition	1	295,00 €	295,00 €
Tarot Gimbal TL2D01	1	187,92 €	187,92 €
3DR FPV Video/OSD System Kit	1	169,13 €	169,13 €
LogiLink USB 2.0	1	16,12 €	16,12 €
Tarot Gimbal GoPro Video Cable	1	10,00 €	10,00 €
Telemetría Flug51 FL1515 433 MHz	1	37,70 €	37,70 €
Patas altas para IRIS+	1	22,36 €	22,36 €
Batería LiPo 1300 mAh 3S	2	10,95 €	21,90 €
Cargador profesional iMAX B6 Mini	1	39,40 €	39,40 €
Adaptador de corriente para iMAX B6 Mini	1	21,10 €	21,10 €
Motor de repuesto TL68A06 para Tarot Gimbal	1	10,94 €	10,94 €
Motor de repuesto TL68A09 para Tarot Gimbal	1	9,59 €	9,59 €
Placa de repuesto ZYX22 para Tarot Gimbal	1	38,48 €	38,48 €
Motores y hélices de repuesto MN2213+T9545 para IRIS+	1	55,00 €	55,00 €
		TOTAL	**1.845,21 €**

Cuadro 6.1: Desglose de costes de los dispositivos hardware

Capítulo 7

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones que se pueden extraer después de todo el trabajo que se ha realizado en el proyecto. Se justificarán las competencias abordadas en el TFG. De igual modo, se propondrá un conjunto de tareas que pueden servir como posibles líneas de trabajo futuro.

7.1 Conclusiones

La **consecuencia principal** de este proyecto es el desarrollo de un sistema capaz de monitorizar ciertos puntos de interés, mediante la coordinación de varios UAVs, en vuelo, que retransmiten imágenes en directo de la zona afectada por una catástrofe, de modo que, los servicios de emergencia reciban una ayuda adicional que permita mejorar los tiempos y la eficiencia de las misiones en esta clase de situaciones. A continuación, las conclusiones serán alineadas con la consecución de los objetivos planteados en el Capítulo 2.

7.1.1 Objetivo general cumplido

Para conseguir este fin, se ha hecho uso del vehículo aéreo **3DR IRIS+**. Tras finalizar el proyecto, su compra se puede considerar un **gran acierto**, debido a que, su guiado automático es intuitivo y sencillo en su configuración. Además, implementa medidas de seguridad que son fundamentales para pilotos que, como en el supuesto del TFG, no disponen de experiencia. Permitiendo así, que mediante el accionamiento de una palanca, el dron vuelva al punto en el que realizó el despegue o realice automáticamente un aterrizaje.

La utilización de **DroneKit-Python**, como plataforma de desarrollo de software, es otra de las **ventajas** y de los motivos por los que comprar el 3DR IRIS+ ha sido todo un acierto. DroneKit-Python cuenta con un amplio abanico de ejemplos y funciones para realizar, de una manera simple, cualquier tipo de operación con el dron. Tanto es así, que permite llevar a cabo procedimientos como la conexión con el vehículo, el despegue, el aterrizaje, la elaboración de misiones, etc.

Al final del proyecto se consigue desarrollar un **sistema adaptativo** capaz de desplegar y coordinar UAVs que recogen información en un entorno afectado por algún tipo de catástrofe.

El sistema logra:

- **Reducir la exposición de agentes humanos**: con la ayuda de drones se posibilita inspeccionar la zona sin poner en peligro a ningún integrante del equipo de emergencia. Además, la visión aérea proporcionada por el vehículo contribuye a la realización de mejores planes de rescate, reduciendo así la exposición al peligro.
- **Disminuir el tiempo** empleado en recoger información del lugar afectado por la catástrofe: en los casos de estudio llevados a cabo en el proyecto, se comprueba como la utilización de varios UAVs puede reducir el tiempo, para la obtención de imágenes, hasta en un tercio del tiempo que se gasta con un solo dron.
- **Minimizar el daño** resultante global: con todo lo anterior, se posibilita una mejor y más rapida toma de decisiones, de tal manera que, el daño global es mucho menor utilizando el sistema.

Cabe destacar la **dificultad** que entraña al proyecto el empleo de varios **dispositivos hardware**. Al desarrollo del código se le une la complejidad, y el esfuerzo en tiempo, de estudiar y comprender el funcionamiento de los componentes. Más aún, si entre estos dispositivos se encuentra un dron, dado que, en las diferentes pruebas de vuelo que se deben realizar es más que probable que el aparato sufra alguna caída o golpe que pueda causar un determinado deterioro en alguno de sus componentes. Estos imprevistos, relacionados con la avería del hardware, provocan que el desarrollo se alargue, a consecuencia de que, se debe esperar hasta el envío de repuestos nuevos.

7.1.2 Objetivos específicos cumplidos

- Se establece un modo de **precisar puntos de monitorización** mediante archivos XML en los que se indica la ubicación del punto de interés, la altura que debe alcanzar el UAV en ese punto y la prioridad de volar hasta él.
- El sistema es totalmente **escalable** en cuanto al número de vehículos a utilizar. En el caso de hacer uso de un solo dron el coordinador se abstrae de esta situación y trabaja del mismo modo que si dispusiera de *N* drones.
- Se **plantea la coordinación**, con el fin de que, ningún vehículo se encuentre en estado ocioso mientras el *Pool de Misiones* no esté vacío.
- El sistema **proporciona capacidad de análisis forense** por medio de la utilización de un sistema FPV, compuesto entre otros por una cámara GoPro HERO 3+.
- Se implementa el **análisis de imágenes** a través de Google Cloud Vision API. Es interesante contar con una plataforma que posibilita el análisis de las imágenes que la cámara, incorporada en el dron, captura. Cualquier clase de información complementaria, aunque parezca insignificante, puede ser una gran contribución a los equipos de emergencia.

7.2 Trabajos futuros

Con el fin de proseguir el desarrollo del proyecto, partiendo de la base de este TFG, se proponen varias líneas de trabajo:

- **Mejora del algoritmo de coordinación utilizando un enfoque descentralizado**: un sistema descentralizado, en oposición a uno centralizado, es una conjunto de ordenadores autónomos que se comunican uno con el otro para realizar un servicio común. Típicamente, los sistemas descentralizados se ubican en diversas ubicaciones geográfica y organizativas.

  Si se desarrolla correctamente, un sistema descentralizado permite una mayor flexibilidad en el crecimiento de las capacidades y las funciones. Además, un sistema descentralizado puede ser más fácil de mantener.

  Descentralizar la manera en la que se realiza la coordinación añadiría robustez al sistema, en cuanto a que, dispondría de capacidad para no depender de una única máquina. Para ello, puede ser adecuado el **uso de un servidor en ZeroC ICE** como plataforma de comunicación.

  ZeroC ICE es un «middleware orientado a objetos, es decir, proporciona herramientas, APIs y soporte de bibliotecas para construir aplicaciones cliente-servidor orientadas a objetos. Una aplicación ICE se puede usar en entornos heterogéneos: los clientes y los servidores pueden escribirse en diferentes lenguajes de programación, pueden ejecutarse en distintos sistemas operativos y en distintas arquitecturas, y pueden comunicarse empleando diferentes tecnologías de red. Además, el código fuente de estas aplicaciones puede portarse de manera independiente al entorno de desarrollo» [Val06].

- **Implementar detección de movimiento**: una funcionalidad más que interesante sería que el sistema fuera capaz de detectar cualquier tipo de movimiento existente en las imágenes.

  El movimiento es un concepto relativo al plano de la imagen y a los objetos existentes en ella, existiendo factores que implican que detectar el movimiento sea complicado como: i) los cambios extrínsecos e intrínsecos de la cámara, ii) los cambios de iluminación o iii) la superposición de múltiples movimientos semitransparentes como la niebla o el polvo.

  La identificación de supervivientes en situaciones de emergencia resultaría mucho más sencilla. Sin embargo, este tipo de implementación es costosa, a causa de que, el dron permanece en constante movimiento y por ello no se posee una imagen fija del entorno.

  La mayoría de los algoritmos de detección automática y de seguimiento están diseñados para configuraciones con cámara fija. En los últimos años ha habido intentos

para desarrollar sistemas de videovigilancia que sean capaces de operar con cámaras en movimiento dando como resultado una serie de algoritmos que, con restricciones, son capaces de resolver el problema. Estos se dividen en dos grandes grupos: métodos que diferencian imágenes consecutivas alineadas y métodos que detectan movimiento directamente en la imagen.

Con todo esto, se propone **entrenar el sistema**, como por ejemplo en [Ven13], **mediante modelos de aprendizaje** para así poder detectar el movimiento en situaciones de emergencia.

- **Reconstrucción de mapas**: un uso actual que tienen los drones es el de realizar mapas topográficos mediante la realización de fotografías a cierta altura. La aparición, en los últimos años, de software topográfico ha contribuido a generar modelos 3D, mediante técnicas de fotogrametría, a partir de fotografías tomadas de un determinado territorio.

  La idea de poder sobrevolar una extensión determinada para sacar unas fotografías y posteriormente poder recomponer el modelo para obtener la geometría y los elementos de dicha extensión hace que esta técnica sea muy potente. Los UAV facilitan la fotogrametría empleando instrumental de bajo coste, como cámaras digitales compactas o réflex, para capturar las imágenes.

  Según [KD] los verdaderos beneficios de esta tecnología no son simplemente duplicar los resultados convencionales. El tiempo necesario para hacer estas mediciones es sustancialmente diferente. La medición de tierra puede requerir días de tiempo completo en el campo, mientras que la medición aérea requiere horas.

  Por ese motivo, se plantea **acoplar al *Modulo de Análisis de Imágenes* la funcionalidad topográfica**, que permita reconstruir el mapa de la zona afectada por la catástrofe, haciendo uso del software, actualmente en desarrollo, incluido en el TFG [Fru].

- **Aumentar la flota de vehículos**: sería importante ampliar el número de drones del que se dispone, de manera que, se pueda plantear la coordinación de los UAVs en entornos reales y no solo simulados. La utilización de una flota de UAVs ofrece una serie de ventajas en misiones de rescate en comparación a la utilización de un único UAV. Una flota de drones que colaboran entre sí permite aumentar el área o reducir el tiempo requerido de una misión de vigilancia y reconocimiento.

ANEXOS

Anexo A

Normativa Española de Seguridad Aérea

La normativa temporal española para regular la utilización civil de las RPAs se contiene en los artículos 50 y 51 de la Ley 18/2014, de 15 de octubre, de aprobación de medidas urgentes para el crecimiento, la competitividad y la eficiencia, publicada en el boletín oficial del estado nº 252, del 17 de octubre de 2014. Esta normativa ha sido extraída íntegramente del documento «*Los Drones y sus aplicaciones a la ingeniería civil*» [Ram15].

A.1 Modificaciones a la Ley de Navegación Aérea

En el artículo 51 se introducen modificaciones a la Ley 48/1960, de 21 de julio, sobre navegación Aérea (LNA), para adaptarla a las aeronaves pilotadas por control remoto. En particular:

- Se modifica la definición de aeronave del artículo 11 de la LNA para establecer sin lugar a dudas que las aeronaves pilotadas por control remoto o drones son aeronaves.
- Se modifica el artículo 150 de la LNA para especificar que las aeronaves civiles pilotadas por control remoto, cualesquiera que sean las finalidades a las que se destinen excepto las que sean utilizadas exclusivamente con fines recreativos o deportivos, quedarán sujetas asimismo a lo establecido en la LNA y en sus normas de desarrollo, y que no están obligadas a utilizar infraestructuras aeroportuarias autorizadas.
- Se modifica el artículo 151 de la LNA para permitir que las actividades de trabajos técnicos y científicos realizadas con aeronaves pilotadas por control remoto puedan iniciarse con una comunicación previa a la Agencia Estatal de Seguridad Aérea (AESA) conforme al artículo 71 bis de la Ley 30/1992 de 26 de noviembre de régimen jurídico de las Administraciones Públicas y del Procedimiento Administrativo Común, sin necesidad de autorización explícita.

A.2 Normas para la operación de aeronaves civiles pilotadas por control remoto

El artículo 50 contiene las disposiciones que regulan la utilización civil de las aeronaves pilotadas por control remoto. En su primer apartado se deja sentada la responsabilidad del operador sobre la aeronave y su operación, y se hace referencia a la obligación de éste de

cumplir con todo el resto de normas que sean aplicables, en particular las que se refieren a la utilización del espectro radioeléctrico, y, en su caso, la protección de datos y la toma de imágenes aéreas, de las que se habla en el apartado 5 posterior.

En el apartado 2 del artículo 50 se establece la exención para las aeronaves civiles pilotadas por control remoto cuya masa máxima al despegue no exceda de 25 kg de los requisitos establecidos con carácter general para todas las aeronaves en los artículos 29 y 36 de la LNA de inscribirse en el registro de Matrícula de Aeronaves y disponer de certificado de aeronavegabilidad, conforme a lo ya previsto en el artículo 151 de esta misma Ley.

Por otro lado, en el mismo apartado se establece la obligación para todas las aeronaves pilotadas por control remoto de llevar fijada a su estructura una placa de identificación en la que figure, de forma legible a simple vista e indeleble, la identificación de la aeronave, mediante la designación específica y, en su caso, número de serie, así como el nombre de la empresa operadora y sus datos de contacto.

Las actividades aéreas con aeronaves pilotadas por control remoto que se contemplan en este artículo 50 son:

- Trabajos técnicos o científicos, ya sea por cuenta ajena o por cuenta propia; los requisitos para su realización y los escenarios operacionales en que se pueden realizar están contemplados en el apartado 3 del artículo 50.
- Vuelos especiales, cuyos requisitos se contemplan en el apartado 4 del artículo 50.

En todos los casos, trabajos aéreos y vuelos especiales, se establece la limitación de que los vuelos habrán de realizarse de día, en condiciones meteorológicas visuales y en espacio aéreo no controlado.

A.3 Normas para las operaciones de trabajos aéreos

Las operaciones de trabajos técnicos o científicos están sujetas además a las siguientes condiciones y limitaciones:

- Las aeronaves pilotadas por control remoto cuya masa máxima al despegue no exceda de 25 kg sólo podrán operar en zonas fuera de aglomeraciones de edificios y personas en ciudades, pueblos o lugares habitados, y a una altura sobre el terreno no mayor de 120 metros. Estas aeronaves habrán de volar dentro del alcance visual del piloto, a una distancia de éste no mayor de 500 metros. Se exceptúan aquéllas cuya masa máxima al despegue sea inferior a 2 kg, que podrán operar más allá del alcance visual del piloto, siempre que se mantengan dentro del alcance de la emisión por radio de la estación de control y que cuenten con medios para poder conocer la posición de la aeronave.
- El resto de las aeronaves, podrán operar con las condiciones establecidas en su certificado de aeronavegabilidad emitido por la Agencia Estatal de Seguridad Aérea.

Los operadores de trabajos técnicos o científicos con aeronaves pilotadas por control remoto habrán de cumplir los siguientes 10 requisitos:

- Disponer de la documentación relativa a la caracterización de las aeronaves que vaya a utilizar.
- Haber elaborado un «Manual de Operaciones del operador» que establezca los procedimientos de la operación. Este documento no debe confundirse con el «Manual de Vuelo» o documento equivalente de la aeronave, que explica su funcionamiento y da instrucciones para su manejo, incluyendo las situaciones anormales y de emergencia, sino que debe contener los criterios y procedimientos que va a utilizar el operador para realizar de manera segura los diferentes tipos de operaciones que lleve a cabo.
- Haber realizado un estudio aeronáutico de seguridad de la operación u operaciones, en el que se constate que la misma puede realizarse con seguridad, que puede ser específico para un área geográfica o tipo de operación determinado, o genérico de manera que abarque un abanico amplio de tipos de operación y/o áreas geográficas.
- Haber realizado con resultado satisfactorio los vuelos de prueba necesarios para demostrar que la operación pretendida puede realizarse con seguridad.
- Haber establecido un programa de mantenimiento de la aeronave, de acuerdo a las recomendaciones del fabricante.
- Que la aeronave esté pilotada por pilotos que cumplan los requisitos establecidos en el apartado 5 del mismo artículo 50.
- Que el operador cuente con una póliza de seguro u otra garantía financiera que cubra la responsabilidad civil frente a terceros por daños que puedan surgir durante y por causa de la ejecución del vuelo, conforme a la normativa aplicable.
- Haber adoptado las medidas adecuadas para proteger a la aeronave de actos de interferencia ilícita durante las operaciones y establecido los procedimientos necesarios para evitar el acceso de personal no autorizado a la estación de control y al lugar de almacenamiento de la aeronave.
- Haber adoptado las medidas adicionales necesarias para garantizar la seguridad de la operación y para la protección de las personas y bienes subyacentes.
- No volar en ningún caso a menos de 8 km de cualquier aeropuerto o aeródromo, o si se trata de una operación más allá del alcance visual del piloto de una aeronave de menos de 2 kg y el aeropuerto cuenta con procedimientos de vuelo instrumental, a menos de 15 km de su punto de referencia.

A.4 Normas para vuelos especiales

Los vuelos especiales que se contemplan son:

- Vuelos de prueba de producción o de mantenimiento, realizados por los fabricantes u organizaciones dedicadas al mantenimiento.
- Vuelos de demostración no abiertos al público, dirigidos a grupos cerrados de asistentes a un evento o clientes potenciales de un fabricante u operador.
- Vuelos para programas de investigación, en que se intenta demostrar la viabilidad de realizar determinadas operaciones con aeronaves de este tipo.
- Vuelos de desarrollo, para poner a punto técnicas y procedimientos para poner en producción una actividad con este tipo de aeronaves.
- Vuelos de I+D realizados por fabricantes de aeronaves pilotadas por control remoto para el desarrollo de nuevos productos.
- Los necesarios para demostrar que las actividades de trabajos aéreos solicitadas conforme al apartado 3 pueden realizarse con seguridad.

Además de la limitación general mencionada anteriormente, estos vuelos habrán de realizarse en zonas fuera de aglomeraciones de edificios en ciudades, pueblos o lugares habitados o de reuniones de personas al aire libre, y dentro del alcance visual del piloto, o, en caso contrario, en una zona del espacio aéreo segregada al efecto.

Los requisitos para su realización son los mismos que para las operaciones de trabajos aéreos, enumerados en la sección anterior, con excepción de los puntos 2, 4 y 5. Por el contrario, han de cumplir con un requisito adicional, que consiste en establecer una zona de seguridad en relación con la zona de realización del vuelo.

A.5 Situaciones de riesgo, catástrofe o calamidad pública

Los operadores de trabajos aéreos habilitados conforme a la normativa para realizar esas actividades podrán realizar, bajo su responsabilidad, vuelos que no se ajusten a las condiciones y limitaciones mencionadas anteriormente para las operaciones de trabajos aéreos y para los vuelos especiales, en situaciones de grave riesgo, catástrofe o calamidad pública, así como para la protección y socorro de personas y bienes en los casos en que dichas situaciones se produzcan, cuando les sea requerido por las autoridades responsables de la gestión de dichas situaciones.

A.6 Requisitos aplicables a los pilotos de las aeronaves civiles pilotadas por control remoto

Se contienen en el apartado 5 del artículo 50, y consisten en:

- Acreditar que poseen los conocimientos teóricos correspondientes a cualquier licencia de piloto, lo que puede hacerse de una de estas tres maneras:
  - Siendo titular de una licencia, o habiéndolo sido dentro de los últimos 5 años.
  - Demostrando de forma fehaciente que disponen de los conocimientos teóricos necesarios para la obtención.
  - Para aeronaves de hasta 25 kg de masa máxima al despegue, disponer de un certificado básico para el pilotaje de aeronaves pilotadas por control remoto, expedido por una Organización de Formación Aprobada por AESA conforme al Reglamento de Personal de Vuelo de la Comisión Europea.
- Quienes no sean titulares de una licencia de piloto deberán acreditar además:
  - Tener 18 años de edad cumplidos.
  - Ser titulares de un certificado médico correspondiente al menos al requerido para una licencia de piloto de aviación ligera, para aeronaves de hasta 25 kg, y de Clase 2, para las de masa superior, conforme al citado Reglamento de Personal de Vuelo de la Comisión Europea.
- Además, todos los pilotos deberán disponer de un documento que acredite que disponen de los conocimientos adecuados de la aeronave que van a pilotar y sus sistemas, así como de su pilotaje.

Anexo B

DroneKit-Python: guía y buenas prácticas

Esta guía ofrece una visión general de cómo utilizar DroneKit-Python mediante un conjunto de buenas prácticas recomendadas. DroneKit-Python se comunica con los pilotos automáticos de los vehículos mediante el protocolo MAVLink, que define cómo se envían los comandos y la configuración de telemetría entre los vehículos, estaciones terrestres y sistemas en otros una red MAVLink.

B.1 Conexión

En la mayoría de los casos se debe usar la forma normal para conectar a un vehículo, es decir, se debe establecer el parámetro *wait_ready* = *True* para asegurar que el vehículo cuente con todos los atributos una vez que la conexión se ha realizado.

```
from dronekit import connect

# Connect to the Vehicle (in this case a UDP endpoint)
vehicle = connect('REPLACE_connection_string_for_your_vehicle', wait_ready=True)
```

Listado B.1: Conexión con el vehículo mediante DroneKit-Python

La llamada a la conexión a veces puede fallar con una excepción. Información adicional acerca de la excepción puede ser obtenida mediante la ejecución de la llamada a la conexión dentro de un bloque try-catch (ver Listado B.2).

Si una conexión se realiza correctamente desde una estación terrestre, pero no desde DroneKit-Python puede ser que su velocidad de transmisión sea incorrecta para su hardware. Esta tasa también se puede configurar en el método *connect()*.

```
import dronekit
import socket
import exceptions

try:
    dronekit.connect('REPLACE_connection_string_for_your_vehicle', heartbeat_timeout=15)

# Bad TCP connection
except socket.error:
    print 'No server exists!'

# Bad TTY connection
except exceptions.OSError as e:
    print 'No serial exists!'

# API Error
except dronekit.APIException:
    print 'Timeout!'

# Other error
except:
    print 'Some other error!'
```

Listado B.2: Conexión haciendo uso de try-catch

B.2 Secuencia de inicio

Generalmente, se debe utilizar la secuencia de lanzamiento estándar que consiste en:

- Consultar el parámetro *Vehicle.is_armable* hasta que el vehículo está listo para ser armado.
- Ajustar el *Vehicle.mode* como *GUIDED*.
- Establecer *Vehicle.armed* a *True* y consultar el mismo atributo hasta que el vehículo se encuentre armado.
- Llamar a *Vehicle.simple_takeoff* con la altitud deseada.
- Consultar la altitud y solo permitir que el código continúe sólo cuando se alcanza dicha altura.

Este planteamiento garantiza que los comandos se envían sólo al vehículo cuando es capaz de actuar sobre ellos.

```
# Connect to the Vehicle (in this case a simulator running the same computer)
vehicle = connect('tcp:127.0.0.1:5760', wait_ready=True)

def arm_and_takeoff(aTargetAltitude):
    """
    Arms vehicle and fly to aTargetAltitude.
    """

    print "Basic pre-arm checks"
    # Don't try to arm until autopilot is ready
    while not vehicle.is_armable:
        print " Waiting for vehicle to initialise..."
        time.sleep(1)

    print "Arming motors"
    # Copter should arm in GUIDED mode
    vehicle.mode    = VehicleMode("GUIDED")
    vehicle.armed   = True

    # Confirm vehicle armed before attempting to take off
    while not vehicle.armed:
        print " Waiting for arming..."
        time.sleep(1)

    print "Taking off!"
    vehicle.simple_takeoff(aTargetAltitude) # Take off to target altitude

    # Wait until the vehicle reaches a safe height before processing the goto (otherwise the command
    #  after Vehicle.simple_takeoff will execute immediately).
    while True:
        print " Altitude: ", vehicle.location.global_relative_frame.alt
        #Break and return from function just below target altitude.
        if vehicle.location.global_relative_frame.alt>=aTargetAltitude*0.95:
            print "Reached target altitude"
            break
        time.sleep(1)

arm_and_takeoff(20)
```

Listado B.3: Secuencia de inicio

B.3 Misiones y «waypoints»

DroneKit-Python puede crear y modificar misiones autónomas. Si bien es posible construir aplicaciones con DroneKit-Python de forma dinámica, mediante la construcción de misiones «on the fly», 3D Robotics recomienda utilizar el modo guiado para aplicaciones con ArduCopter.

El modo *AUTO* se utiliza para la ejecución de misiones con «waypoints» predefinidos.

DroneKit-Python proporciona funciones básicas para descargar y borrar los comandos actuales de la misión del vehículo, agregar y cargar nuevos comandos de misión y contar el número de «waypoints» visitados. Es una buena idea apoyarse en estas primitivas básicas para crear la funcionalidad de planificación de la misión de alto nivel.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.1:14550)
vehicle = connect('127.0.0.1:14550', wait_ready=True)

# Get the set of commands from the vehicle
cmds = vehicle.commands
cmds.download()
cmds.wait_ready()

# Clear the current mission (command is sent when we call upload())
cmds.clear()

# Create and add commands
cmd1=Command( 0, 0, 0, mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT, mavutil.mavlink.
    MAV_CMD_NAV_TAKEOFF, 0, 0, 0, 0, 0, 0, 0, 10)
cmd2=Command( 0, 0, 0, mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT, mavutil.mavlink.
    MAV_CMD_NAV_WAYPOINT, 0, 0, 0, 0, 0, 0, 10, 10, 10)
cmds.add(cmd1)
cmds.add(cmd2)
cmds.upload() # Send commands
```

Listado B.4: Creación de misiones

Para iniciar una misión, solo se debe cambiar el modo a *AUTO*.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.1:14550)
vehicle = connect('127.0.0.1:14550', wait_ready=True)

# Set the vehicle into auto mode
vehicle.mode = VehicleMode("AUTO")
```

Listado B.5: Iniciar misión

B.4 Pausar el script cuando no se necesita

Pausar la secuencia de comandos puede reducir la carga del computador.

Por ejemplo, a bajas velocidades es posible que sólo tenga que comprobar si se ha alcanzado un objetivo cada pocos segundos. Usando *time.sleep(2)* entre comprobaciones será más eficiente que comprobarlo más a menudo.

B.5 Finalizar el script

Los scripts deben llamar a *Vehicle.close()* antes de salir para asegurarse de que todos los mensajes se han vaciado antes de que el script finalice.

```
# About to exit script
vehicle.close()
```

Listado B.6: Cerrar la conexión con el vehículo

Anexo C

Instalación de dependencias

Debido al uso de múltiples librerías y herramientas que han servido de ayuda en el desarrollo del proyecto, resulta adecuado explicar cuál ha sido el proceso de instalación. A continuación, se listan todos las dependencias que han sido necesarias.

C.1 SITL

Es probablemente la mayor dependencia del sistema, ya que, permite simular el dron sin disponer de ningún hardware, pudiendo así, poner a prueba el comportamiento del código sin necesidad de hardware. Gracias a él, ha sido posible plantear la coordinación con hasta 3 vehículos.

Lo primero que se debe hacer para poder utilizar SITL es descargar una copia del repositorio *git* de ArduPilot:

```
$ git clone git://github.com/ArduPilot/ardupilot.git
```

Listado C.1: Copia de repositorio de ardupilot

Una vez que se dispone de una copia de ArduPilot, es necesario modificar el archivo nombrado como *sim_vehicle.sh* que se puede encontrar en la siguiente ubicación *ardupilot/-Tools/autotest/sim_vehicle.sh*. Debemos modificar las líneas correspondientes a:

```
if [ $USER == "vagrant" ]; then
options="$options --out 10.0.2.2:14550"
fi
options="$options --out 127.0.0.1:14550 --out 127.0.0.1:14551"
```

Listado C.2: Archivo *sim_vehicle.sh* original

por las siguientes:

```
if [ $USER == "vagrant" ]; then
options="$options"
fi
options="$options"
```

Listado C.3: Archivo *sim_vehicle.sh*

Una vez realizado esto, se puede realizar tantas copias de la carpeta *ardupilot*, como se desee, a más copias más simulaciones se podrán crear. En el caso del proyecto se realizan tres copias, de modo que el directorio de trabajo quede de la siguiente forma:

```
TFG/
   ardupilot/
   ardupilot2/
   ardupilot3/
   Images/
   Missions/
   Videos/
   Waypoints/
   convertMission.py
   coordinator.py
   drone.py
   FPVSystem.py
   proxyDrone.py
```

Listado C.4: Estructura del directorio de trabajo tras la descarga de *ardupilot*

ArduPilot proporciona una serie de dependencias para trabajar con SITL. Se deben ejecutar los siguientes comandos:

```
$ sudo apt-get install python-matplotlib python-serial python-wxgtk2.8 python-lxml
$ sudo apt-get install python-scipy python-opencv ccache gawk git python-pip python-pexpect
$ sudo pip install pymavlink MAVProxy
```

Listado C.5: Dependencias de SITL

Antes de hacer uso del simulador se deben añadir las siguientes lineas al final de su *.bashrc*:

```
export PATH=$PATH:/path_to_dir/ardupilot/Tools/autotest
export PATH=/usr/lib/ccache:$PATH
```

Listado C.6: Líneas a incorporar en *.bashrc* para hacer uso de SITL

Para iniciar el simulador primero se debe ejecutar *sim_vehicle.sh*, pero antes de debe cambiar al directorio del vehículo que se quiera hacer uso. Por ejemplo, para vehículos de ala rotatoria el directorio es *ardupilot/ArduCopter*. La primera vez que se ejecute se debe usar la opción -w para limpiar la memoria virtual EEPROM y cargar los parámetros por defecto.

```
ardupilot/ArduCopter$ sim_vehicle.sh -w
```

Listado C.7: Primer inicio de SITL

Se puede iniciar el simulador con el vehículo en un lugar determinado llamando a *sim_vehicle.sh* con el parámetro *-L* y una ubicación especificada en el archivo *ardupilot/Tools/autotest/locations.txt*.

```
ardupilot/ArduCopter$ sim_vehicle.sh -j4 -L Ballarat --console --map
```

Listado C.8: LLamada a *sim_vehicle.sh* para iniciar simulación en localización determinada

C.2 DroneKit-Python

DroneKit-Python se instala desde *pip* en todas las plataformas. En el caso de Linux se tendrá que instalar previamente *python-dev*:

```
$ sudo apt-get install python-dev
```

Listado C.9: Dependencias de DroneKit-Python

A continuación, mediante *pip* se instala la libreria DroneKit.

```
$ sudo pip install dronekit
```

Listado C.10: Instalación de DroneKit-Python

C.3 Google Cloud Vision API

Es necesario completar unos cuantos pasos de configuración antes de poder utilizar esta biblioteca:

- Creación de una cuenta de Google.
- Crear un proyecto en la consola de Google API.
- Instalar la biblioteca.

Se debe usar *pip* para gestionar la instalación de la biblioteca:

```
$ sudo pip install google-api-python-client
```

Listado C.11: Instalación de Google API

Para poder hacer uso de *Google Cloud Vision* API es necesario poseer unas credenciales de Google que se utilizan para realizar la autenticación y, poder así, hacer uso del servicio. Una vez que se disponga de las credenciales, la estructura del directorio de trabajo será la siguiente:

```
TFG/
   GoogleCredentials/
   ardupilot/
   ardupilot2/
   ardupilot3/
   Images/
   Missions/
   Videos/
   Waypoints/
   convertMission.py
   coordinator.py
   drone.py
   FPVSystem.py
   proxyDrone.py
```

Listado C.12: Estructura del directorio de trabajo tras descargar las credenciales de Google

La carpeta *GoogleCredentials* contiene un archivo .json para la autenticación. Antes de hacer uso del servicio de detección de etiquetas se deben añadir las siguientes lineas al final de su *.bashrc*:

```
export GOOGLE_APPLICATION_CREDENTIALS="/path/to/keyfile.json"
```

Listado C.13: Líneas a incorporar en *.bashrc* para hacer uso de Google Cloud Vision API

Anexo D

Imágenes analizadas con Google Cloud Vision API

En este anexo se reflejan las imágenes del caso de estudio 2, tanto las que tienen que ver con la estación de control terrestre como las pertenecientes al *Módulo de Análisis de Imágenes*. También, se presentan un conjunto de imágenes, con sus resultados, al realizar el análisis con vídeos extraídos de internet.

D.1 Imágenes del caso de estudio 2

D.1.1 Estación de control terrestre

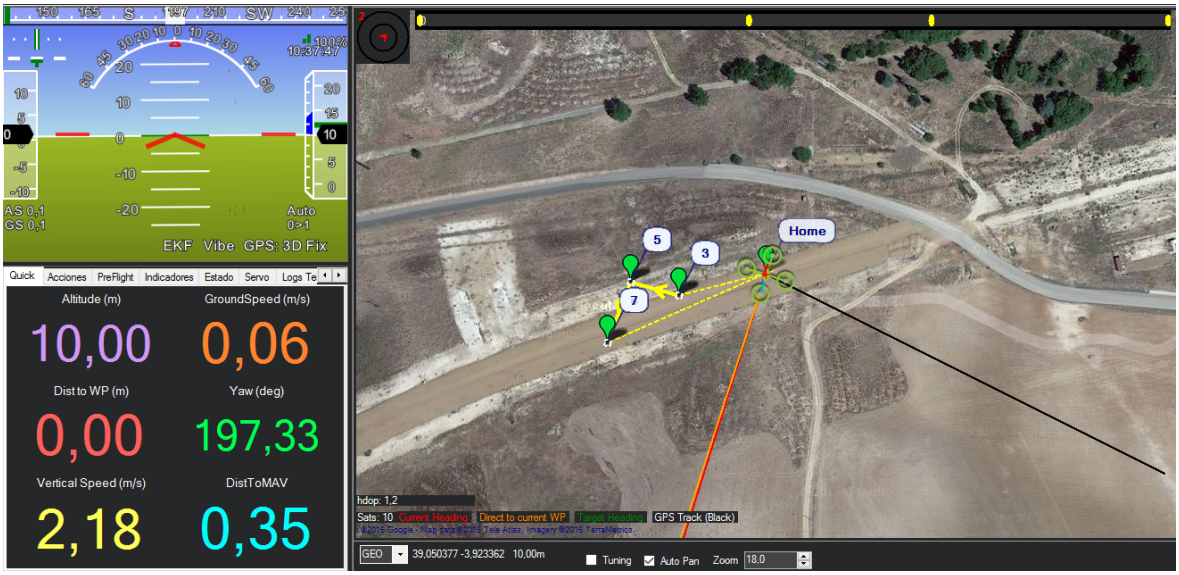


Figura D.1: Despegue del 3DR IRIS+

Figura D.2: 3DR IRIS+ volando a «waypoint» 1

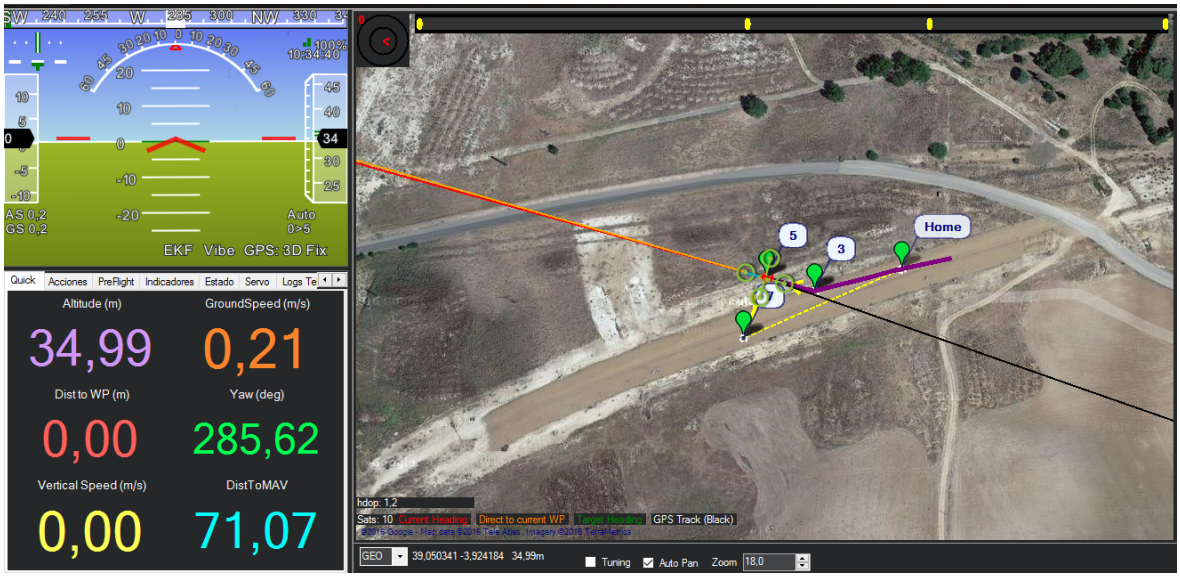


Figura D.3: 3DR IRIS+ volando a «waypoint» 2

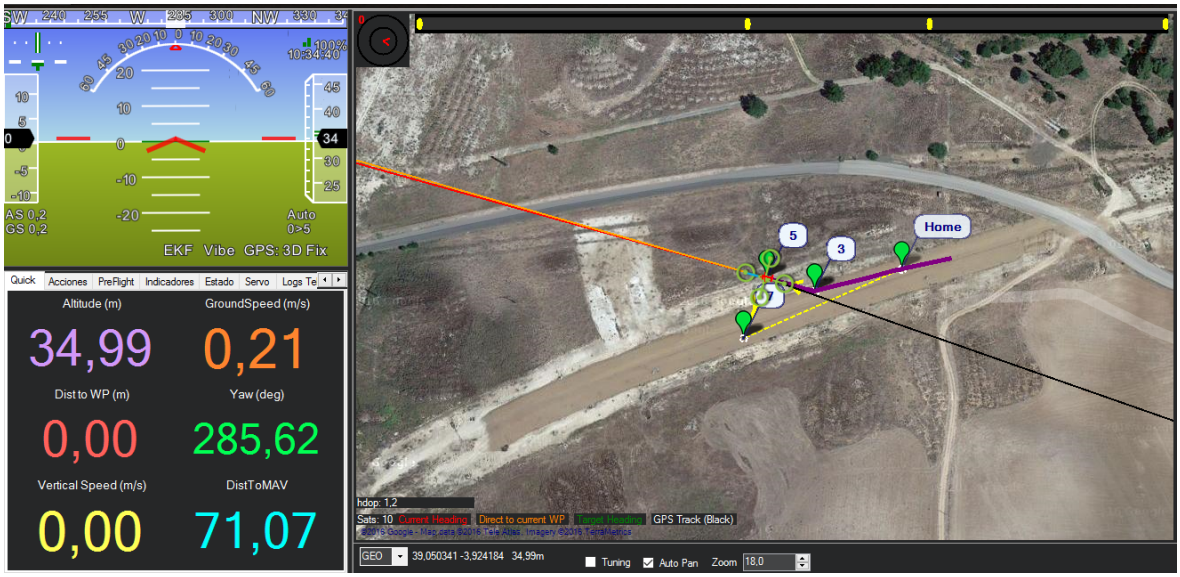


Figura D.4: 3DR IRIS+ volando a «waypoint» 3

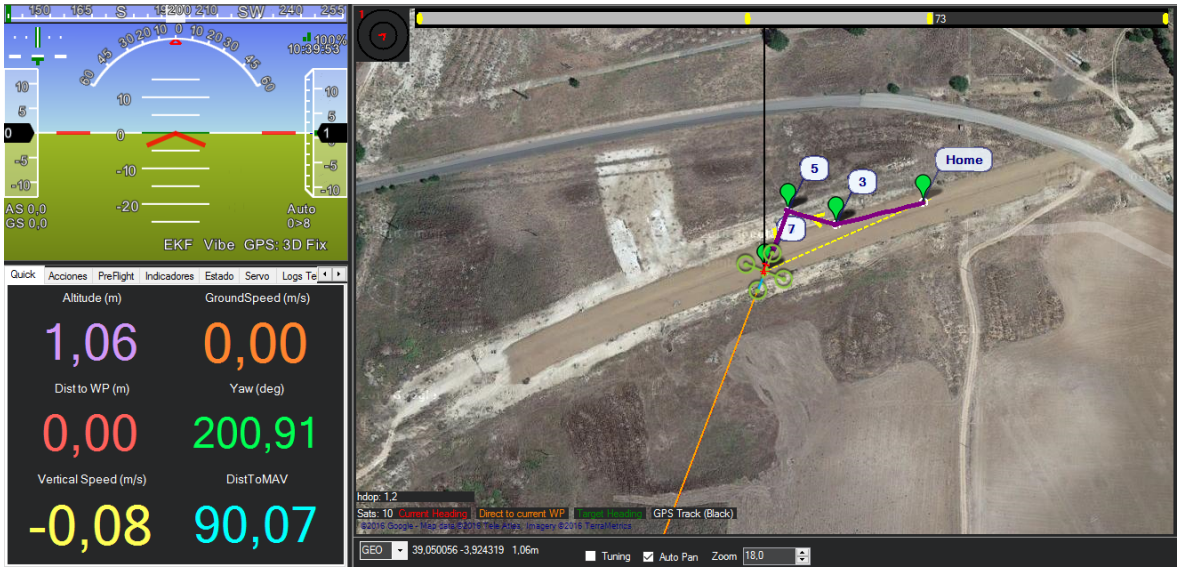


Figura D.5: 3DR IRIS+ realizando el aterrizaje

D.1.2 Imágenes analizadas

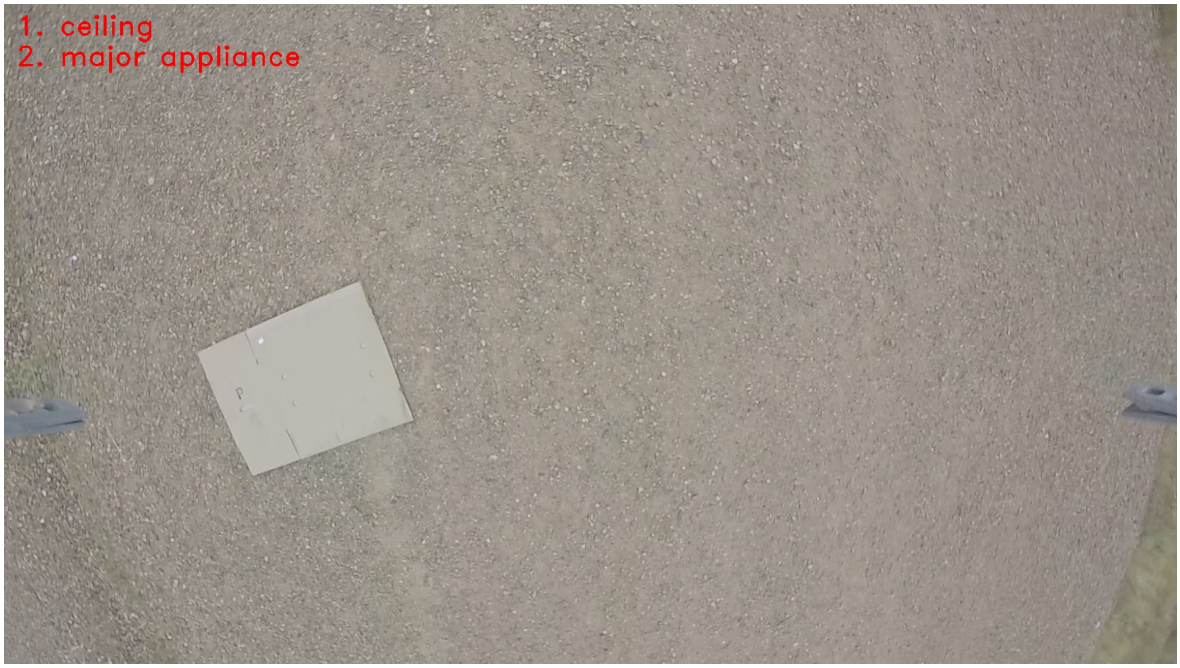


Figura D.6: Análisis de imagen durante despegue

Figura D.7: Análisis de imagen al llegar al «waypoint» 1

Figura D.8: Análisis de imagen al llegar al «waypoint» 2

Figura D.9: Análisis de imagen al llegar al «waypoint» 3

D.2 Otras imágenes

Figura D.10: Análisis de imagen tomada en una metrópoli

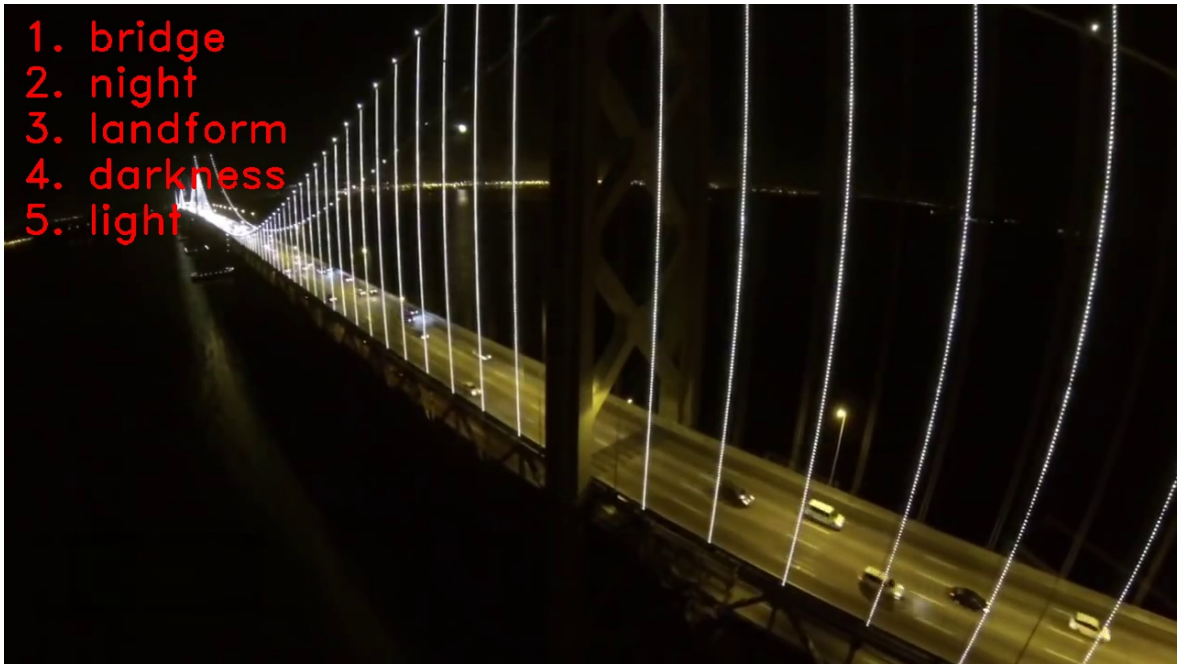


Figura D.11: Análisis de imagen tomada en un puente

Figura D.12: Análisis de imagen de un barco incendiado

Figura D.13: Análisis de imagen tomada en el campo

Referencias

[3D 15] 3D Robotics. *IRIS+: Operation manual*, 2015.

[AG93] N. Avouris y L. Gasser, editors. *Distributed artificial intelligence: Theory and praxis*. Kluwer Academic Publishers, 1993. ISBN: 0-7923-1585-5.

[AGA] V. Arevalo, J. González, y G. Ambrosio. La librería de visión artificial OpenCV. Aplicación a la docencia e investigación. `http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf` (visitado el 09-07-2016). Departamento De Ingeniería de Sistemas y Automática, Universidad de Málaga.

[Bal14] M. Ballve. Commercial drones: Assessing the potential for a new drone-powered economy. `http://www.drones.uk.com/home/commercial-drones-assessing-the-potential-for-a-new-drone-powered-economy` (visitado el 14-07-2016), Octubre 2014. Business Insider.

[CHC14] C. Calvo, F. Herranz, y P. Calvo. De los UAV a los RPAS. Technical report, IDS, Febrero 2014.

[Chi16] J. Chinchilla. Un trabajo de altura. `http://www.lanzadigital.com/news/show/sociedad/un-trabajo-de-altura/100263` (visitado el 02-08-2016), Junio 2016. Diario Lanza.

[Cob15] A. Cobo. El dron socorrista de Isla. `http://www.eldiariomontanes.es/castro-oriental/201507/16/dron-socorrista-playa-arnuero-20150716152205.html` (visitado el 18-07-2016), Julio 2015. El Diario Montañés.

[Cue15] C. Cuerno. Capítulo 1: Origen y desarrollo de los sistemas de aeronaves pilotadas por control remoto. En *Los drones y sus aplicaciones a la ingeniería civil*, páginas 15–32. 2015. E.T.S.I. Aeronáuticos, Universidad Politécnica de Madrid, Depósito Legal: M. 4519-2015.

[dACI11] Organización de Aviación Civil Internacional, editor. *Sistemas de aeronaves no tripuladas (*UAS*)*. Organización de Aviación Civil Internacional, 2011. Circular OACI 328-AN/190. ISBN: 78-92-9231-809-3.

[Def14] S. Deffree. 8 open-source holiday gifts. http://www.edn.com/electronics-blogs/the-workbench/4438049/6/8-open-source-holiday-gifts (visitado el 11-07-2016), Diciembre 2014. EDN Network.

[des15] Desastres naturales y tecnológicos. En Alimentación y Medio Ambiente Ministerio de Agricultura, editor, *Perfil ambiental de España 2014*, páginas 226–239. 2015. NIPO: 280-15-162-3.

[DJI16] DJI. *DJI Matrice 100: User Manual*, Marzo 2016.

[Dí15] J. Díaz. Capítulo 17: Aplicaciones de rescate. En *Los drones y sus aplicaciones a la ingeniería civil*, páginas 199–209. 2015. Drones Rescue Spain. Depósito Legal: M. 4519-2015.

[Fer12] F. Fernández. *Los sistemas no tripulados*, capítulo Generalidades, misión y capacidades. Documentos de seguridad y defensa. Ministerio de Defensa, Subdirección General de Publicaciones, Marzo 2012. ISBN: 978-84-9781-733-2.

[FGM04] J. Ferber, O. Gutknecht, y F. Michel. From agents to organizations: An organizational view of multi-agent systems. En *Agent-Oriented Software Engineering IV*, páginas 214–230. Springer Berlin Heidelberg, 2004. ISBN: 978-3-540-20826-6.

[FP12] D. Forsyth y J. Ponce. *Computer vision: A modern approach*. Pearson, edición 2, 2012. ISBN: 978-0136085928.

[Fru] D. Frutos. Sistema automático para la obtención de fotografías de monumentos mediante drones. Universidad de Castilla-La Mancha.

[Gon16] M. González. Los drones sustituirán a los socorristas en las playas de Torrevieja. http://www.elmundo.es/comunidad-valenciana/2016/02/10/56bb7fb7ca4741b0188b45bc.html (visitado el 18-07-2016), Febrero 2016. El Mundo.

[GSC+06] S. Griffiths, J. Saunders, A. Curtis, T. McLain, y R. Beard. *Obstacle and terrain avoidance for miniature aerial vehicles*, páginas 213–244. Springer Netherlands, Septiembre 2006. ISBN: 978-1-4020-6113-4.

[Gut09] A. Guterres. Cambio climático, desastres naturales y desplazamiento humano: la perspectiva del ACNUR. http://www.acnur.org/t3/fileadmin/Documentos/BDL/2009/6936.pdf?view=1 (visitado el 23-07-2016), Agosto 2009. Alto Comisionado de Naciones Unidas para los Refugiados.

[Igl98] C. Iglesias. *Definición de una metodología para el desarrollo de sistemas multiagente*. PhD thesis, Universidad Politécnica de Madrid, Enero 1998.

[KD] B. Kerr y J-F. Dionne. Levantamiento topográfico aéreo hecho por un UAV ¿Puede sustituir levantamientos topográficos terrestres GPS? http://www.uavsensefly.cl/wp-content/uploads/2014/04/Levantamiento-topogr%C3%A1fico-a%C3%A9reo-hecho-por-un-UAV.pdf (visitado el 05-08-2016).

[Kni07] H. Kniberg. *Scrum y* XP *desde las trincheras: Cómo hacemos Scrum*. Lulu.com, 2007. ISBN: 978-1-4303-2264-1.

[KVT04] M. Kontitsis, K. Valavanis, y N. Tsourveloudis. A UAV vision system for airborne surveillance. En *ICRA*, páginas 77–83. IEEE, Abril 2004.

[LS06] P. Letelier y E. Sánchez. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, 5(26), Junio 2006.

[LTT] A. Lockman, G. Tighe, y H. Tucker. De-Mining with UAVs. https://www.wpi.edu/Pubs/E-project/Available/E-project-031216-112944/unrestricted/de-mining-with-uavs.pdf (visitado el 08-07-2016). Robotics Engineering Program, Worcester Polytechnic Institute.

[Ló15] C. López. Prólogo. En *Los drones y sus aplicaciones a la ingeniería civil*, páginas 13–14. 2015. Consejería de Economía y Hacienda, Comunidad de Madrid. Depósito Legal: M. 4519-2015.

[Oñ14] M. Oñate. Tipología de los RPAS. http://www.aerpas.es/wp-content/uploads/2015/08/20140630-AERPAS-Tipolog%C3%ADa-de-RPAS.pdf (visitado el 05-07-2016), Junio 2014. Asociación Española de RPAS.

[Oñ15] M. Oñate. Capítulo 3: Tipología de aeronaves pilotadas por control remoto. En *Los drones y sus aplicaciones a la ingeniería civil*, páginas 49–57. 2015. USOL - Unmanned Solutions. Depósito Legal: M. 4519-2015.

[PEL] Pelícano: Seguridad y defensa nacional en cinco continentes. http://www.indracompany.com/sites/default/files/PELICANO_Esp_0.pdf (visitado el 28-07-2016). Indra.

[PLA01] Modelos de coordinación para inteligencia artificial distribuida. Technical report, Universidad de Málaga, Junio 2001. Proyecto DAMMAD, Diseño y Aplicación de Modelos Multiagente para Ayuda a la Decisión.

[Pla13] N. Platón. Un dron para emergencias. https://www.maxcolchon.com/images/maxcolchon-venecia3-larazon-dic-13.pdf (visitado el 17-07-2016), Diciembre 2013. La Razón.

[Pur08] A. Puri. *Statistical profile generation for traffic monitoring using real-time* UAV *based video data*. PhD thesis, University of South Florida, 2008. Department of Computer Science and Engineering.

[PV15] S. Peasgood y M. Valentin. Drones: A rising market. http://sophiccapital.com/wp-content/uploads/2015/09/Download-Sophic-Capitals-Aerial-Drone-Report-Here.pdf (visitado el 05-07-2016), Septiembre 2015. Sophic Capital.

[Ram15] J. Ramírez. Capítulo 2: Aspectos Reglamentarios. En *Los drones y sus aplicaciones a la ingeniería civil*, páginas 33–47. 2015. Consejería de Economía y Hacienda, Comunidad de Madrid. Depósito Legal: M. 4519-2015.

[Rom15] A. Romero. Revisión y modificación del firmware de libre acceso arduCopter para su uso en el proyecto AirWhale. http://bibing.us.es/proyectos/abreproy/90411/fichero/Memoria+Alejandro+Romero+Gal%C3%A1n.pdf+ (visitado el 07-07-2016), 2015. Departamento de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingeniería Universidad de Sevilla.

[Syc98] K. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[UA14] La UA, pionera en el empleo de drones como herramienta para la docencia e investigación en el campo de la ingeniería civil. http://www.lasprovincias.es/v/20140405/alicante/universidad-alicante-pionera-usar-20140405.html (visitado el 18-07-2016), Abril 2014. Las Provincias.

[UAV05] Dictionary of military and associated terms. http://www.bits.de/NRANEU/others/jp-doctrine/jp1_02(05).pdf (visitado el 02-07-2016), Agosto 2005. United States Department of Defense.

[Val06] D. Vallejo. Documentación de ZeroC ICE. http://www.esi.uclm.es/www/dvallejo/docs/ICE/docICE.pdf (visitado el 05-08-2016), Diciembre 2006. Universidad de Castilla-La Mancha.

[Val09] D. Vallejo. *Arquitectura multi-agente basada en servicios para un sistema de vigilancia cognitivo*. PhD thesis, Universidad de Castilla-La Mancha, Noviembre 2009.

[Ven13] J. Ventayol. Detección de personas desde aviones no tripulados UAV. https://ddd.uab.cat/pub/trerecpro/2013/hdl_2072_233490/PFC_JordiVentayolMarimon.pdf (visitado el 04-08-2016), Junio 2013. Universitat Autònoma de Barcelona.

[Wei99] G. Weiss, editor. *Multiagent systems: A modern approach to distributed modern approach to artificial intelligence*. The MIT Press, 1999. ISBN: 0-262-23203-0.

[WJ95] M. Wooldridge y N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[Woo01] M. Woolridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Ltd., 2001. ISBN: 047149691X.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20160819) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`