UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Soluciones para el control de enjambres de multicópteros

Trabajo Fin de Master

Máster Universitario en Ingeniería de Computadores y Redes

Autor: Pablo Reyes Pausa

Tutor: Carlos Tavares Calafate

Resumen

El presente proyecto nace de la necesidad de crear un algoritmo para el control de enjambres de multicópteros para múltiples tareas. Para ello, dicho algoritmo debe ser compatible con el protocolo de comunicación *MAVLink* por ser el más extendido entre las controladoras de vuelo abiertas. Las comunicaciones a nivel de transporte entre las aeronaves se realizan mediante el protocolo *UDP*, asegurando así una latencia mínima y altas tasas de transferencia. El nivel de acceso al medio (MAC) y la capa física (PHY) se controlan mediante el estándar *IEEE 802.11a* en modo *ad-hoc* bajo la banda de los *5,8GHz* para asegurar el ancho de banda necesario, y mitigar las interferencias entre el algoritmo y los sistemas de control externos de las aeronaves. El algoritmo se ejecuta desde una *Raspberry Pi* siguiendo una máquina de estados que, a su vez, se comunica mediante una conexión en serie con la controladora de vuelo. Ésta es la encargada final de comunicar las órdenes en formato *MAVLink* a una controladora de vuelo *PIXHAWK*. Para el desarrollo y la verificación del algoritmo se utiliza el simulador de vuelo *Ardusim*, que, mediante la replicación de múltiples instancias del simulador de aeronaves *SITL*, permite la simulación de hasta 256 aeronaves en un mismo sistema.

Palabras clave: Multicóptero, MAVLink, Enjambre, UDP, PIXHAWK.

Abstract

The present project is born as a requirement to create an algorithm for the control of an UAV swarm applicable to multiple tasks. For this reason, this algorithm must be compatible with the *MAVLink* communications protocol, as it is the most widespread among open-source flight controllers. Communications at the transport layer between aircrafts are made using the *UDP* protocol, thus ensuring a minimum latency and high transfer rates. The media access control (MAC) and physical layer (PHY) are controlled by the *IEEE 802.11a* standard in ad-hoc mode under the *5.8GHz* band to ensure the necessary bandwidth and mitigate the interference between the algorithm and the external control systems of the aircraft. The algorithm is executed from a *Raspberry Pi* following a state machine which communicates through a serial connection with the flight controller. This is the final responsible for communicating orders in *MAVLink* format to a *PIXHAWK* flight controller. *Ardusim* is used for the development and verification of the algorithm, which through the replication of multiple instances of the *SITL* aircraft simulator, allows the simulation of up to 256 aircrafts in the same system.

Keywords: Multicopter, MAVLink, Swarm, UDP, PIXHAWK.

Índice

Índice de Ilustraciones

Índice de Tablas

1. Introducción

Actualmente los drones o UAV empiezan a ser un elemento habitual entre la población. Los podemos observar realizando tareas de vigilancia, ayudando a la Policía en distintas tareas, realizando trabajos de alto riesgo como la inspección de redes eléctricas, e incluso como transportistas aéreos.

Esta popularidad se debe en parte a su bajo coste y las grandes mejoras que aportan a la sociedad. Un ejemplo podría ser la toma de fotografía aérea, que no muchos años atrás solo era posible con aviones y/o helicópteros, haciendo que el coste económico fuese muy elevado. Ahora, con uno de estos dispositivos, se pueden hacer tareas similares de manera barata, sencilla y rápida.

Pero antes de seguir con sus ventajas, procede entender mejor qué son estas aeronaves. El concepto de aeronave sin piloto o, en términos actuales, vehículos aéreos no tripulados o UAVs (por sus siglas en inglés, «**Unmanned Aerial Vehicle**»), ha venido siendo interpretado por la comunidad internacional como un concepto comprensivo de las aeronaves que vuelan sin un piloto a bordo, y que pueden, o bien ser controladas plenamente por el piloto remoto, aeronaves pilotadas por control remoto, o bien estar programadas y ser aeronaves autónomas en terminología de la Organización Internacional de Aviación Civil (OACI) [1]. Los drones más utilizados en la actualidad suelen estar formados por cuatro, seis u ocho motores controlados mediante un control remoto y asistidos por una controladora de vuelo, que se encarga de realizar tareas de estabilización y asistencia en vuelo. Esto es posible gracias a la utilización de antenas GPS, Acelerómetros, Giroscopios, Altímetros y Barómetros encapsulados en un pequeño computador de placa simple (SBC) [2].

Estas características han llevado a los UAV a ser una gran herramienta para todo tipo de trabajos, pero hasta la fecha todos ellos han sido utilizados unitariamente. Existe una nueva posibilidad, todavía por explorar, que es el uso de estas aeronaves en conjunto, para la creación de enjambres, lo que amplía sus capacidades y utilidades a nuevos horizontes.

La suma de un conjunto de estas aeronaves permite la realización de tareas que anteriormente eran casi imposibles de realizar o tenían un coste muy elevado. Técnicamente, un "enjambre" es un grupo de *UAVs* impulsados por inteligencia artificial. Los drones en enjambre se comunican entre ellos mientras están en vuelo, y pueden responder a las condiciones cambiantes de forma autónoma. Una buena analogía sería una bandada de estorninos reaccionando a una amenaza repentina como pude ser un depredador. Toda la bandada maniobra como un solo organismo [3].

Existen multitud de nuevas posibilidades, como la búsqueda de personas desaparecidas en áreas extensas, grabaciones aéreas de amplios terrenos sin la necesidad de realizar incursiones terrestres, aplicaciones agrícolas con grandes enjambres fumigadores o fertilizadores que respeten los cultivos, etc.

El principal problema que podría surgir para la utilización de esta nueva tecnología podría ser la mala praxis de los usuarios, pero cada país tiene sus regulaciones y, a través del cumplimiento de las mismas, se podría realizar un uso seguro y legal de estas aeronaves.

Actualmente la normativa española solo contempla dos tipos de vuelo:

- **EVLOS**: Un vuelo dentro del alcance visual aumentado (EVLOS, por sus siglas en inglés Extended Visual Line of Sight), es una operación en la que el contacto visual directo con la aeronave se satisface utilizando medios alternativos, en particular, observadores en contacto permanente por radio con el piloto.

- **BVLOS**: Las operaciones de más allá del alcance visual del piloto (BVLOS) con una aeronave de más de 2kg de MTOM, requieren que estas dispongan de sistemas aprobados o autorizados por la Agencia Estatal de Seguridad Aérea, que permitan a su piloto detectar y evitar a otros usuarios del espacio aéreo. En caso contrario, estas operaciones fuera del alcance visual del piloto (BVLOS) solo podrán realizarse en espacio aéreo temporalmente segregado (TSA).

Como se ha podido observar, en ningún momento se mencionan los enjambres, y esto es debido a su inexistencia de uso general, pero en un futuro, su utilización exigirá la revisión de esta normativa, al igual que sucederá en el resto de países que utilicen este tipo de tecnologías.

1.1 Motivación

La principal motivación que impulsó la realización del presente proyecto fue la creación de un algoritmo para el control de enjambres de drones basados en controladoras de código abierto, que posibiliten su programación y permitan que un gran número de usuarios puedan beneficiarse sin la necesidad de adquirir hardware propietario.

Mediante la utilización de esta tecnología, se ofrece un algoritmo de código abierto disponible para su modificación y/o ampliación, basado en un lenguaje de programación altamente utilizado como lo es *JAVA* [4].

Con todo ello se busca que esta tecnología sea de uso público y que cualquier persona con una idea o proyecto pueda beneficiarse de todas las ventajas que ofrece un sistema autónomo de control de enjambres.

Otras motivaciones que originaron el desarrollo del presente proyecto fueron, por una parte, que habían pocas alternativas existentes en el mercado en el momento de la realización del mismo, y, por otra parte, que las disponibles no eran de acceso libre ni gratuito.

En la actualidad solo una empresa tecnológica está realizando avances en este sector, en concreto, Intel. Más adelante se presentará su solución y se analizará su comportamiento, pero cabe destacar que su solución no es pública, y solo ellos pueden disfrutar de sus beneficios.

Por otro lado, el gobierno chino también ha realizado diversas pruebas con enjambres, aunque destinados a usos militares, y sin ofrecer ninguna información al respecto al resto de investigadores. En las siguientes secciones se profundizará más en ambas soluciones.

1.2 Objetivos

El principal objetivo de este proyecto es la creación y diseño de un algoritmo autónomo capaz de controlar enjambres de drones. Para ello las aeronaves deberán de comunicarse utilizando el protocolo *UDP* y sincronizarse continuamente mediante el intercambio de mensajes.

Para realizar las funciones de enjambre, las aeronaves deben ser completamente autónomas y no necesitar control por parte de ningún ser humano. Es decir, la persona que desee utilizar el algoritmo solo debe enviar la ruta que ha de seguir el enjambre, y éste deberá adaptarse a las necesidades y a la ruta introducida.

Otro objetivo es la imposibilidad de que dicho enjambre quede bloqueado por la desconexión o por el siniestro de alguno de los drones presentes en el enjambre. La misión establecida deberá cumplirse siempre y cuando el Maestro del enjambre lo indique. En secciones futuras se especificará quien es el Maestro, y qué significa su utilización.

Para que dicho algoritmo sea de uso público se opta por la utilización de *MAVLink* como protocolo de comunicaciones interno para las controladoras de vuelo. Esta elección no es casual, y se debe a que dicho protocolo es el más extendido entre controladoras de vuelo de uso masivo. Además, es el único protocolo libre para el control de *UAVs* disponible en el mercado que es capaz de soportar algoritmos como el que se desarrolla en este proyecto.

Gracias a la combinación de software basado en *JAVA* y del protocolo *MAVLink* mediante comunicaciones basadas en *UDP* se desarrolla una solución flexible y aplicable en la actualidad.

1.3 Estructura del documento

El presente documento se divide en nueve bloques con distintas aportaciones. El primer bloque consta de una introducción a la problemática presente, y a las necesidades surgidas tras la popularización de los drones.

Seguidamente se repasa la situación actual del mercado, repasando las distintas soluciones para esta misma problemática, que han surgido mediante el estudio de distintas entidades tanto públicas como privadas.

El tercer bloque instruye al lector con las tecnologías utilizadas por los drones y sus características más reseñables, para así obtener una base técnica y entender los términos más comúnmente utilizados en este ámbito. Además, se presenta el protocolo MAVLink y sus distintos mensajes.

El cuarto bloque muestra el funcionamiento del simulador SITL y sus características internas, las cuales serán a posteriori explotadas por el simulador Ardusim.

El quinto bloque anuncia la solución aportada por esta investigación, y repasa el hardware utilizado para poder poner en uso la solución ideada.

El sexto bloque anuncia el algoritmo desarrollado para el control de enjambres de drones mediante el uso del protocolo MAVLink, y la máquina de estados desarrollada para la sincronización y cálculo de las rutas de vuelo. Además, se incidirá en las distintas fases del vuelo y en las soluciones aportadas para su correcto funcionamiento.

En el séptimo bloque se evalúan las prestaciones obtenidas por el algoritmo de vuelo. Se analizan los resultados tanto desde un punto de vista práctico, mediante mapas de calor, como desde un punto de vista funcional. Para ello se analizará el comportamiento del algoritmo en las distintas fases de vuelo, como lo es el despegue, la ruta de vuelo y el aterrizaje.

Para finalizar, tras el análisis de los puntos anteriores, se procede a la obtención de unas conclusiones finales, y se concluye con una serie de trabajos futuros posibles basándose en la solución aportada.

2. Estado del arte

En la actualidad los drones son una herramienta más a tener en cuenta en todos los ámbitos, ya sea como objeto de investigación o para su uso intensivo en tareas específicas. En sus inicios los drones o UAVs eran objetos de uso exclusivamente militar pero, con el paso de los años, y con sus precios más aptos para todos los públicos, se ha popularizado la utilización de los mismos.

La aplicabilidad de estos dispositivos es muy amplia, ya que los podemos observar realizando mapeado 3D [5], tareas de control fiscal, trabajos topográficos [6], publicidad, búsqueda de personas [7], y una larga lista más de posibilidades.

Este proyecto se centra en el uso de drones para la creación de enjambres. En la actualidad solo se conocen dos entidades que hayan conseguido realizar enjambres de drones capaces de volar de forma autónoma sin la necesidad de pilotos para su control.

Uno de los casos de uso de enjambres, es el Gobierno de China, que en el año 2016 realizó unas pruebas utilizando estas aeronaves. En este caso no es posible verificar si fue realmente un enjambre autónomo o no porque la hermeticidad del proyecto no aportó ninguna información que pudiese esclarecer su funcionamiento. También cabe destacar que no utilizaban multicópteros, sino que utilizaban aviones planeadores, lo que cambia en su totalidad las ventajas que obtenemos al utilizar multicópteros, o como se conocen comúnmente, drones [8].

El segundo caso de uso de enjambres lo realizó la multinacional Intel. En este caso sí que se han podido obtener detalles de su estrategia, ya que los drones están basados en controladoras *PIXHAWK*, que son las mismas que se han utilizado en este proyecto. Su solución ha sido posible visualizarla en los juegos olímpicos de invierno en *Pieonchang,* donde han sido capaces de controlar grandes cantidades de aeronaves. Detalles más concretos sobre el protocolo no se han obtenido ni se han publicado, así que no se puede extraer más información acerca de cómo se sincroniza el protocolo ni si los propios drones son los que se encargan de sincronizarse, o se trata de un supercomputador [9]. Como pasaba con la solución anterior, es privada y no se espera que Intel la libere para el uso civil.

Existen otras aproximaciones que intentan ofrecer herramientas para crear enjambres de drones, pero no aportan una solución específica. En el 2013 la *University of Technology, Kluyverweg* en Holanda, mediante el uso de drones de la compañía *Parrot* ofrecían la posibilidad de comunicar varios drones a través

de una estación de tierra, que sería la encargada de recibir todas las comunicaciones y actuar en consecuencia. Como se ha mencionado anteriormente, dejaban la ventana abierta a realizar enjambres con su solución, pero sin aportar un recurso específico. Este proyecto tiene como contrapartida la obligada utilización de drones de la compañía Parrot, y la necesidad de una estación de tierra (GCS) que se encargue de sincronizar las aeronaves. Esta solución tendría también problemas si las formaciones tienen que recorrer grandes distancias, por la dependencia de un punto terrestre [10].

Por último, un proyecto muy interesante es el creado por la *University of Pennsylvania*. En este caso sí que se dispone de toda la información acerca del algoritmo y de su implementación hardware. En esta solución utilizan micro drones equipados con módulos *Zigbee* para comunicarse. Los movimientos son precisos, y son capaces de realizar patrones de movilidad complejos de forma segura y rápida, aunque no cubre las necesidades de un enjambre autónomo a campo abierto, ya que la sincronización de las aeronaves viene dada por una red de sensores instalada en la sala de pruebas. Este motivo hace que dicha solución no sea posible explotarla en los ámbitos donde realmente estas aeronaves están realizando trabajos en la actualidad [11].

	CL	MVL	WSN	CC	GCS
CHINA GOV.	✗	?	✗	?	?
INTEL	✗	✓	✗	?	?
PARROT AR	✓	✗	✗	✓	✓
U. PENNSYLVANIA	✓	✗	✓	✓	✗
NECESIDAD	✓	✓	✗	✓	✓

Tabla 1 – Comparativa de protocolos para gestión de Swarms.

* MVL – Mavlink, * CL – Código libre, *WSN – Wireless Sensor Network, CC – Collective Comunications, * GCS – Ground Control Station

Estas nuevas tecnologías pueden ofrecer grandes mejoras y beneficios para los usuarios, pero también es conveniente que se trabaje de forma segura, ya que existe la posibilidad de amenazas externas [12].

Tras el análisis realizado se dedujo que existía una necesidad no resuelta en este ámbito; por ello, el presente proyecto aporta una solución actual de acceso masivo y fácil modificación a un sector generalmente hermético y de difícil acceso.

3. Tecnologías para multicópteros

3.1 Tipos de drones

Un dron es un vehículo aéreo no tripulado (VANT) o UAV (del inglés *unmanned aerial vehicle*). Se caracteriza principalmente por ser una aeronave que vuela sin tripulación, capaz de mantener de manera autónoma un nivel de vuelo controlado y sostenido, propulsada por un motor de explosión, eléctrico o de reacción.

Dentro de los UAV existen distintas posibilidades de construcción. Este algoritmo está especialmente diseñado para los multicópteros cuyo movimiento se controla acelerando o ralentizando múltiples unidades de propulsión / motor. Estos UAVs están constituidos por un número igual o mayor a 3 motores [13].

Dependiendo del tipo de uso que se pretenda dar a la aeronave, será conveniente elegir un tipo de UAV u otro. La forma más común de clasificar estas es según la cantidad de motores que constituye la aeronave, partiendo de los 3 motores hasta llegar a los 16, siendo estas las configuraciones más comunes.

La elección de la configuración no es un asunto irrelevante ya que, dependiendo de la elección hecha, se pueden obtener distintas características:

- **Menor cantidad de motores:** aeronaves más rápidas y menos pesadas, en general con un menor consumo pero más inestables; si el número de motores es igual o menor a cuatro, el fallo de un motor es catastrófico para la aeronave.
- **Mayor cantidad de motores:** aeronaves más lentas y pesadas, con un consumo ligeramente superior a medida que se aumenta la cantidad de motores. Permiten cargas mayores de peso y son más estables. El fallo de un motor es asumible y la aeronave puede seguir en vuelo.

Como se observa en la *Ilustración 1,* incluso para un mismo número de motores existen distintas configuraciones; por ello hay que estudiar cual será el uso que se dará al UAV antes de su compra o construcción [14].

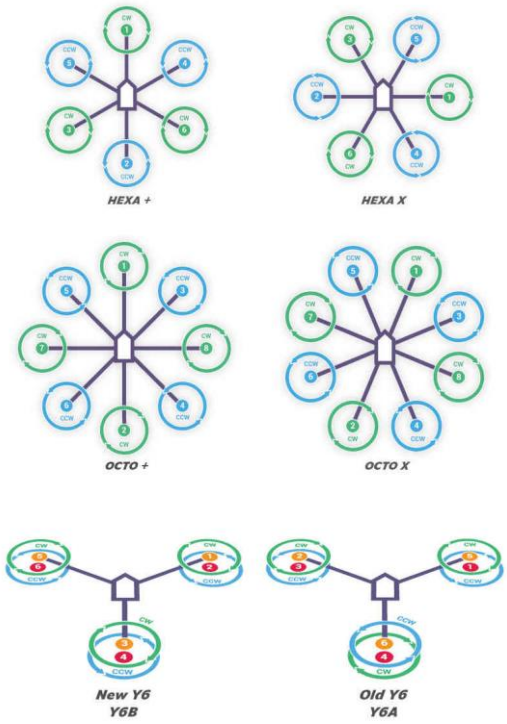

Ilustración 1 - Esquemas de Hexacópteros.

Para controlar este tipo de naves es necesaria la intervención de distintos motores actuando de manera sincronizada, para poder ofrecer los distintos tipos de movimientos que pueden realizar. Además, para una correcta estabilización, y para ofrecer un control sencillo, es necesario que incorporen un elemento clave en este tipo de aeronaves: la controladora de vuelo.

Existen distintos tipos de controladoras, desde las privativas, como las construidas por la compañía DJI, a las de código abierto, como lo son PIXHAWK, CC3D o APM.

En cuanto a las controladoras de código abierto, todas comparten una base común: el protocolo de comunicaciones *MAVLink*. Este protocolo nos permite una comunicación directa con la aeronave que es explotada por el protocolo de gestión del enjambre. A continuación se expone de forma más detallada el funcionamiento de dicho protocolo.

3.2 MAVLink

MAVLink es un protocolo de comunicación diseñado para el intercambio de información entre UAVs y para el control de los mismos. Concretamente, MAVLink es una biblioteca de clasificación de mensajes del tipo *header-only*, caracterizada por ser muy ligera, consiguiendo así que estas pequeñas aeronaves lo soporten.

Los mensajes dentro del protocolo se definen como archivos XML, a partir del cual será exportado a los distintos lenguajes de programación que soporta. Cada archivo XML define el conjunto de mensajes admitidos por un sistema en particular, también denominado "dialecto" [15].

Existen diversas versiones del protocolo; las versiones más utilizadas son, la 0.9 y la 1.0, en un futuro está previsto que se lance la versión 1.1 que será compatible con su predecesora, pero hasta la fecha todas las aeronaves utilizadas incorporaban la versión 1.0 [16].

Dentro del protocolo existen dos tipos distintos de mensajes, los pertenecientes a la versión 1 y la 2, pero la versión 2 todavía no ha sido implementada en las controladoras de vuelo. A continuación describiremos los campos presentes en la versión 1 de los mensajes.

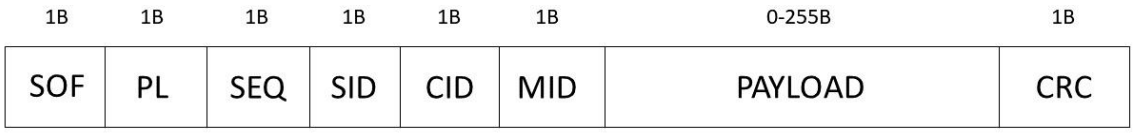

Ilustración 2 - Mensaje MAVLink V1.

- **Payload**: es el único campo variable en el mensaje; esto es así para poder crear distintos tipos de mensaje según el contenido del mismo.

- **SOF** *(Start of frame)*: Este campo se utiliza para indicar el comienzo del mensaje. Sin él, el mensaje seria descartado ya que la controladora no lo reconocería como mensaje propio.

- **PL** *(Payload length)*: Indica la longitud que tiene el *Payload* en bytes.

- **SEQ** *(Sequence of Paquet)*: Número de secuencia del paquete desde el inicio de la trasmisión entre la controladora y la estación de tierra o dispositivo controlador.

- **SID** (*Sender identificator*): Identificador del sistema emisor.

- **CID** (*Component Identificator*): Identificador del componente emisor del sistema.

- **CRC** *(Cyclic Redundancy Check)*: MAVLink implementa dos comprobaciones de integridad: la primera comprobación es la integridad del paquete durante la transmisión, utilizando *CRC-16-CCITT*. Sin embargo, esto solo garantiza que los datos no se hayan modificado en el enlace; no garantiza la coherencia con la definición de datos. La segunda verificación de integridad está en la descripción de datos, para garantizar así que dos mensajes con el mismo ID contengan la misma información. Para lograr esto, se ejecuta la comprobación mediante *CRC-16-CCITT,* y el valor resultante se utiliza para inicializar el paquete CRC [17].

Otra característica remarcable del protocolo es la posibilidad de crear mensajes propios, eso sí, siguiendo los campos descritos anteriormente y usando valores no reservados por otros mensajes predeterminados. A continuación se pude observar la descripción parcial de un mensaje especificado en XML.

```
<message id="24" name="GPS_RAW_INT">
        <description>The global position, as returned by the Global Positioning
        System (GPS). This is NOT the global position estimate of the
        system, but rather a RAW sensor  value.
        </description>
        <field type="uint64_t" name="time_usec">Timestamp (microseconds since
        UNIX epoch or microseconds since system boot)
        </field>
.
.
.
        <field type="uint16_t" name="cog">Course over ground (NOT heading, but
                direction of movement) in degrees * 100, 0.0..359.99
                degrees. If unknown, set to: UINT16_MAX
        </field>
        <field type="uint8_t" name="satellites_visible">Number of satellites
                visible. If unknown, set to 255
        </field>
</message>
```

4. Emulación de Multicópteros

En este apartado vamos a analizar la herramienta SITL, la cual permite emulador con gran detalle un único UAV, así como el simulador ArduSIm, el cual hace uso de varias instancias SITL para gestionar en simultáneo varios UAVs, así como las comunicaciones entre ellos con el fin de lograr una coordinación en vuelo en tiempo real.

4.1 SITL

El simulador SITL (*Software In The Loop*) permite ejecutar Planeadores, Multicópteros o Rovers sin la necesidad de utilizar ningún hardware. Es una compilación del código *Autopilot* que utiliza un compilador C++, que tras la compilación proporciona un ejecutable nativo que permite, mediante software, emular el comportamiento de un hardware real.

Esta herramienta permite la realización de pruebas sin la necesidad de poner en riesgo el material disponible en el laboratorio, ahorrando así tiempo y dinero. Este ahorro es debido a que dichas aeronaves suelen quedar inservibles tras un accidente ya que durante el periodo de desarrollo pueden aparecer fallos y sería inviable la realización de pruebas mediante hardware real.

SITL permite ejecutar UAVs aprovechando que *Ardupilot* está basado en MAVLink, y este, a su vez, es compatible con una gran cantidad de lenguajes de programación, por ello es posible migrar las aeronaves *Ardupilot* a ordenadores personales mediante la compilación en el lenguaje correspondiente.

Cuando se ejecuta SITL los datos de los sensores provienen de un modelo de dinámica de vuelo integrado en el simulador, ofreciendo así una simulación realista de la física terrestre [18].

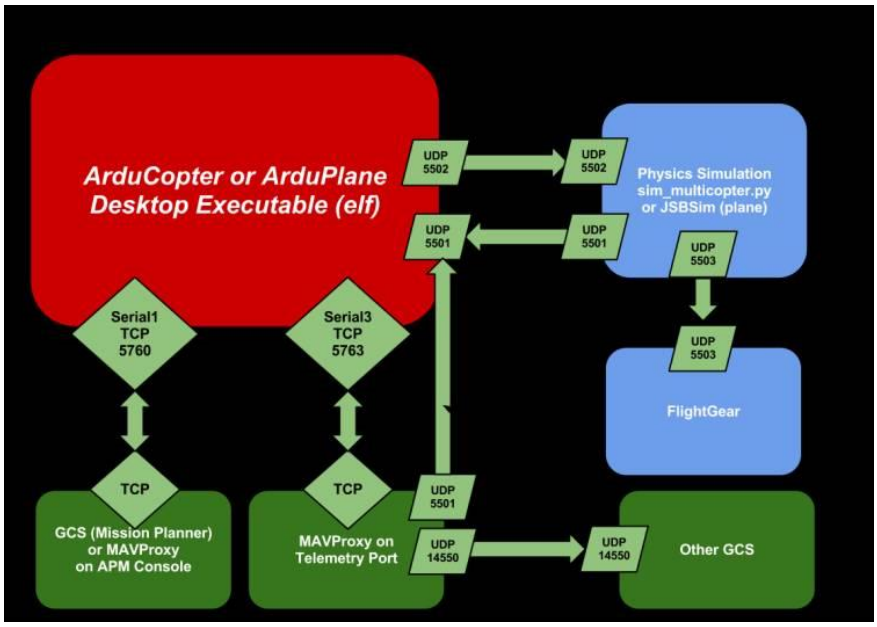

Ilustración 3 - Arquitectura SITL.

4.2 Ardusim

Ardusim es un simulador de vuelo para multicópteros desarrollado en la Universidad Politécnica de Valencia (UPV), por el Grupo de Redes de Computadores (GRC[1]). Está basado en SITL, el cual puede simular con gran precisión distintos tipos de UAVs y, mediante una conexión TCP, establecer comunicaciones con ellos. Ardusim crea una capa lógica superior a través de la cual se puede interactuar con SITL, pero su principal aportación frente a SITL es la posibilidad de simular grandes cantidades de UAVs en una sola computadora.

Ardusim es capaz de simular hasta 256 UAVs bajo una computadora equipada con un *INTEL Core i7*. La cantidad de drones simulados no es casual, y es debida a que SITL utiliza un identificador de 8-bits para identificar la aeronave, y justamente por este motivo el número máximo que se ha conseguido simular es de 256. Esta posibilidad, junto con otras expuestas a continuación, son el motivo por el cual se opta por esta plataforma. Además, ofrece una ***API*** sencilla y un entorno para el desarrollo de protocolos. Otras de sus características añadidas reseñables respecto a SITL son:

- Ardusim es una plataforma abierta que ofrece un entorno de simulación en tiempo real para UAVs.
- Utiliza un modelo de comunicaciones realista utilizando el protocolo IEEE 802.11a.

[1] http://www.grc.upv.es/

- La potencia de escalado de Ardusim ha sido estudiada y analizada de forma exhaustiva para ofrecer los mejores resultados.

Dependiendo del escenario en el que se encuentre Ardusim, se puede realizar una ejecución en UAVs simulados o directamente aplicarlo a UAVs reales.

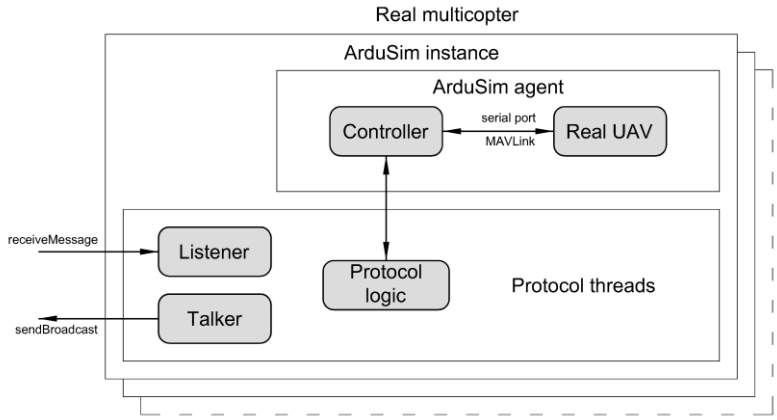

Ilustración 4 – Soporte para UAVs reales mediante Ardusim.

Como se pude observar en la *Ilustración 4,* la arquitectura seguida por Ardusim en drones reales es sencilla. Existen dos hilos principales para **enviar** y **recibir** órdenes de la aeronave y, mediante la lógica del protocolo implementado y el controlador propio de Ardusim, se realiza la conexión con el dron real.

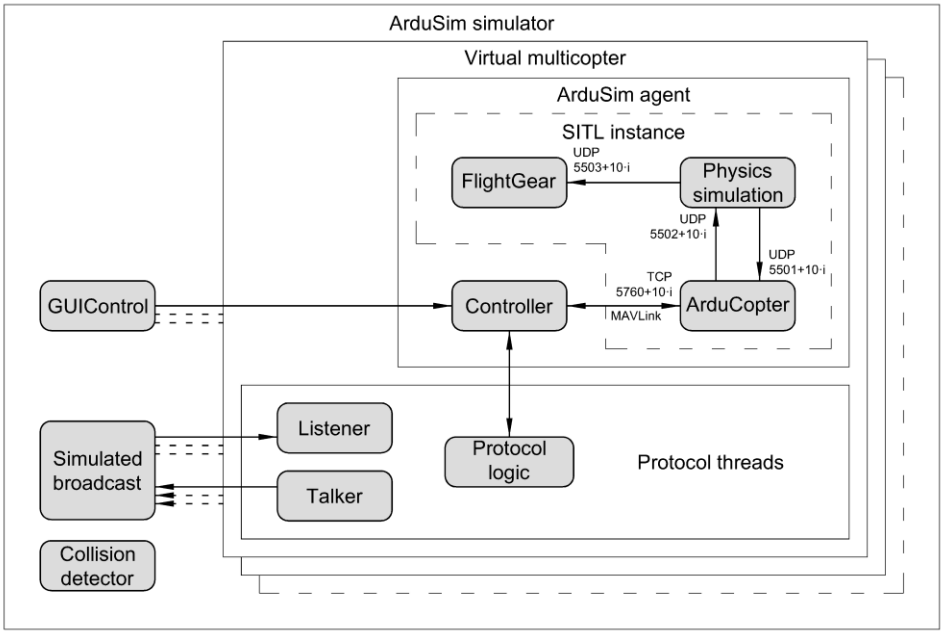

Ilustración 5 – Ardusim: Dron Simulado.

La ejecución basada en simulación es muy similar a la real, pero, para poder simular el comportamiento y la física de nuestro entorno, se instancia SITL para aprovechar su motor físico. En la *Ilustración 5* también podemos observar que los protocolos desarrollados sobre Ardusim no detectan estas diferencias, ya que se comunican directamente con los hilos lectores o publicadores.

Gracias a estas características, la solución propuesta en este trabajo encaja con la herramienta y ha hecho posible su desarrollo

5. Solución propuesta

Tras el análisis de los puntos anteriores la situación actual de los drones ofrece la posibilidad de desarrollar aplicaciones novedosas. En concreto, en el entorno de los enjambres de drones, aparece una oportunidad de aprovechar estas aeronaves para realizar tareas de difícil cumplimiento sin ellos.

El algoritmo desarrollado está creado para ser utilizado bajo controladoras basadas en MAVLink, y es una solución con poco coste computacional para que estas aeronaves puedan aplicarlo de forma autónoma y sin la necesidad de ningún hardware exterior. En concreto, el software desarrollado desde Ardusim utiliza el lenguaje JAVA y, mediante la ejecución del algoritmo generado sobre una Raspberry Pi integrada en el dron, permiten su funcionamiento.

El algoritmo sigue durante todas sus etapas el cumplimiento de una máquina de estados finita desarrollada para la ocasión. El cumplimiento de ésta hace posible que cada dron trabaje sobre su propio hardware, y se sincronice con los demás utilizando el protocolo IEEE 802.11a sobre la frecuencia de los 5,8GHz. A nivel de trasporte, UDP es el encargado de la trasferencia de información ya que, mediante la utilización de este protocolo, podemos obtener mejores latencias y tasas de transferencias, que para este caso, es la mejor opción.

El intercambio de mensajes es bidireccional entre las aeronaves, y se consigue aplicando la técnica de comunicación **Maestro-Esclavo**. Durante todas las etapas del algoritmo los drones Esclavos se comunican con el dron Maestro para acordar así unos parámetros de vuelo como: la ruta de vuelo, la velocidad, la altura de despegue, el orden en el que despegan, y una serie de parámetros adicionales.

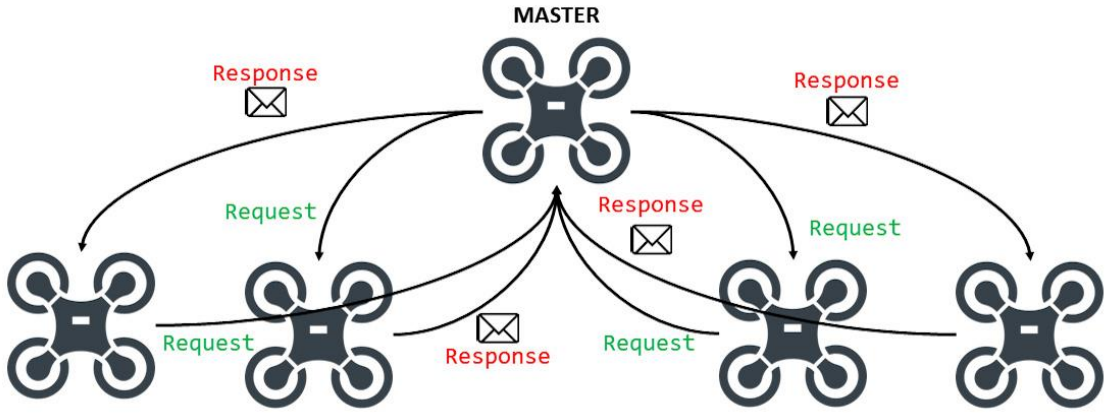

Ilustración 6 - Diagrama de la interacción entre UAVs.

Para la sincronización de las aeronaves en función de los eventos ocurridos, se han diseñado una serie de mensajes predeterminados que ayudan al algoritmo a mantener una sincronía y una coherencia en la información, necesaria para su correcto funcionamiento. Además, para asegurarse una fiabilidad alta, cada uno de estos mensajes debe proporcionar un ACK propio, para así informar al dron Esclavo que ninguna de sus órdenes es obviada o se pierde a través del medio de comunicación.

Las etapas que todo vuelo tiene son: cálculo del algoritmo en función del estado actual de los multicópteros, despegue, ruta de vuelo y aterrizaje. Para que dichas fases se realicen correctamente se han desarrollado diversos métodos tales como:

- **Takeoff**: utilizado para despegar de forma sincronizada todas las aeronaves del enjambre, evitando la posible colisión entre ellas. Para realizar un despegue seguro se realiza en dos fases. Primero se asciende a una altura intermedia calculada en función de la altura final y, dependiendo del cálculo realizado por el Maestro, se va ordenando progresivamente el despegue de las aeronaves.

- **WpReached**: comprueba si una aeronave ha pasado por cada uno de los puntos de su ruta. Para ello, mediante una fórmula matemática, se establece un perímetro alrededor de los *Waypoints,* a partir del cual el algoritmo confirma si dicho *Waypoint* ha sido alcanzado o no.

- **Land**: encargado de aterrizar los drones de forma segura. Basado en WpReached, detecta si se trata del último *Waypoint* de la misión sincronizada. De ser así, se espera a que todas las aeronaves estén en la posición final, y a continuación se procede con el aterrizaje en paralelo.

Gracias a estas características, el algoritmo es capaz de seguir una formación de vuelo estable y uniforme. En la sección dedicada a la evaluación del algoritmo se comprobará dicha afirmación.

5.2 Hardware empleado

Durante el desarrollo del proyecto se ha buscado la máxima compatibilidad con las aeronaves más comunes del mercado; para ello se eligió MAVLink como base de toda interacción con las aeronaves. Las controladoras integradas (PIXHAWK) no son lo suficientemente potentes para procesar algoritmos externos, y por ello los drones tienen que ser equipados con una Raspberry pi. A continuación se enumeran algunas de las características más importantes de estos dispositivos.

PIXHAWK es una controladora de vuelo de hardware libre, el objetivo de la cual es proveer un hardware dedicado al control de UAVs, para el entretenimiento, la comunidad académica, e incluso la industria. Se caracteriza por su bajo costo, la utilización de MAVLink para las comunicaciones, y una gran conectividad.

Características PIXHAWK

Procesador	• 32bit STM32F427 Cortex M4 core con FPU
Sensores	• Barómetro • Giroscopio • Acelerómetro
Interfaces	• UART • CAN • PPM • RSSI • I2C • SPI • Micro USB
Alimentación	• Puertos Main Out • Micro USB • Pin Power
Dimensiones	• Anchura 50 mm • Altura 15.5 mm • Longitud 81.5 mm (3.2”)

Tabla 2 - Características de la controladora PIXHAWK.

Ilustración 7 - PIXHAWK

La **Raspberry Pi** es un ordenador de placa reducida o en inglés: **Single Board Computer o SBC**. Los sistemas compatibles con este computador están basados en Linux. Ofrece alta conectividad tanto inalámbrica como física, e incluye pines GPIO. Su tamaño y costes son reducidos, y han sido objeto de uso en muchas investigaciones.

Características de la Raspberry Pi

CPU	4X ARM Cortex-A53, 1,2GHz
GPU	Broadcom VideoCore IV
RAM	1GB DDR2
RED	Ethernet / WiFI 2,4GHz
Conexiones	• GPIO • HDMI • Jack audio • Video analógico • USB • DSI • CSI
Almacenamiento	microSD

Tabla 3 - Caracteristicas de la Raspberry Pi.

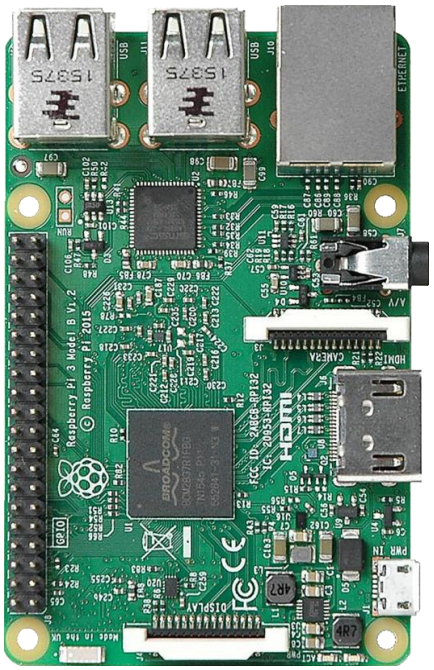

Ilustración 8 - Raspberry Pi

Una vez se dispone del hardware mencionado, se debe realizar una conexión hardware entre los dos micro computadores. Para ello se aprovechan los pines GPIO de la Raspberry Pi, y la segunda entrada para telemetría de la PIXHAWK. Si la conexión es exitosa, la controladora será capaz de recibir y enviar órdenes desde/hacia la Raspberry Pi [19].

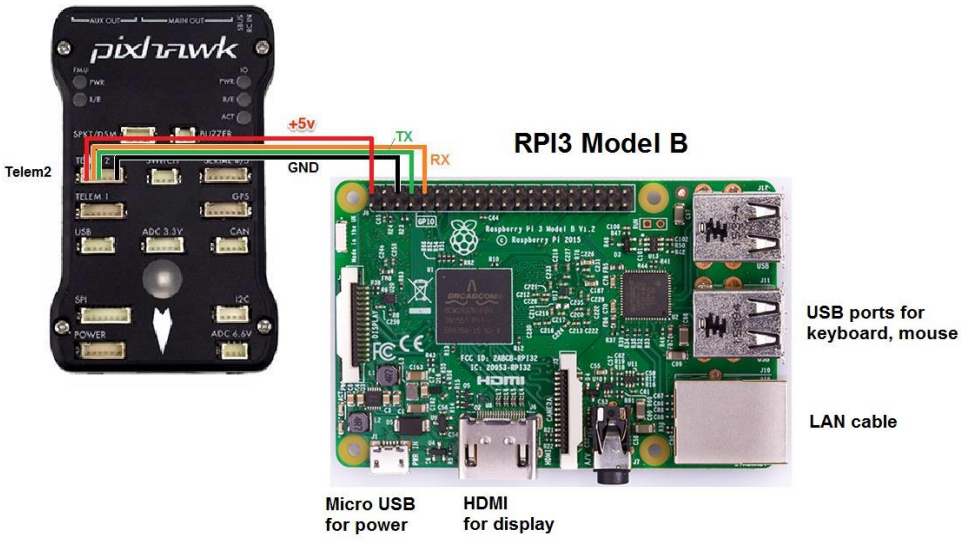

Ilustración 9 - Conexión en serie

Con la integración de la Raspberry Pi se hace posible la creación física de la red ad-hoc, que permite a los drones realizar el intercambio de información entre ellos, y poder así aplicar el algoritmo desarrollado.

A continuación se procede a explicar cuáles son los componentes que forman los multicópteros, y cuál es la función de cada uno de ellos, para así esclarecer conceptos que se usarán en siguientes secciones.

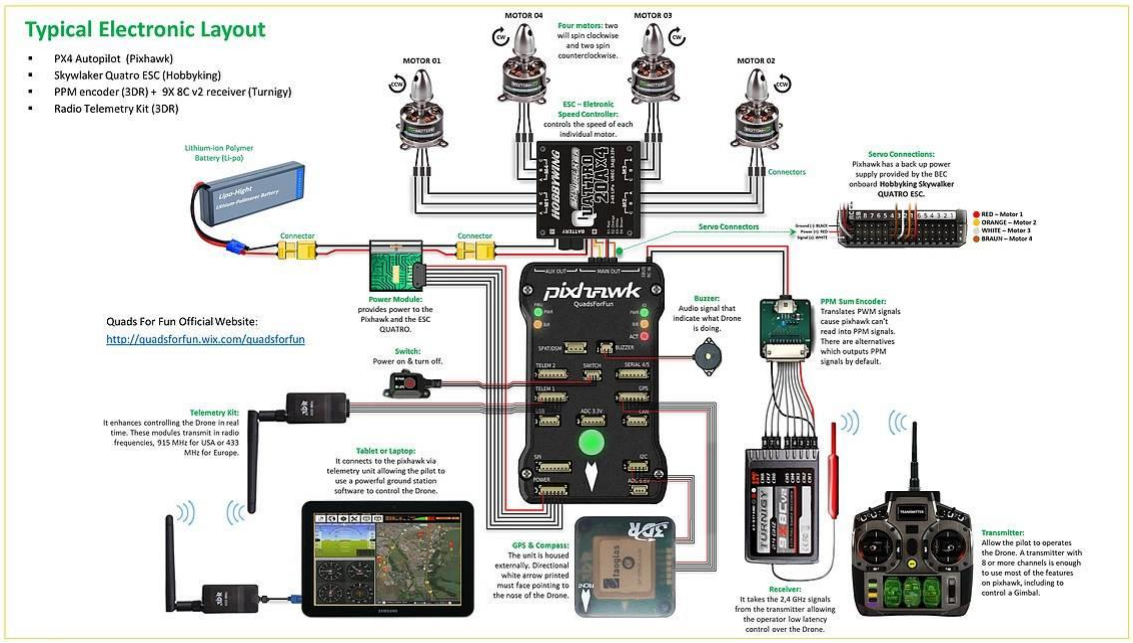

Ilustración 10 - Diagrama de un Multicóptero.

Como se observa en la Ilustración 10, un multicóptero está formado por:

- **Motores:** Encargados de realizar la fuerza motriz para su movimiento. La combinación de fuerzas entre ellos ofrece los distintos tipos de movimientos que pueden realizar.
- **Variadores o ESC:** (Electronic Speed Controller): Sirven para que los motores actúen de forma inteligente y funcionen según los estímulos proporcionados por la controladora, para así ajustar su velocidad de rotación en función de las ordenes emitidas por la controladora de vuelo.
- **Controladora de vuelo:** En este caso se opta por la PIXHAWK al ser basada en MAVLink, y ofrecer una gran potencia de cómputo.
- **Receptor RC:** Para el control desde mandos radio control, ejerce de punto de enlace entre tierra-aire.
- **GPS:** Necesario para cualquier tipo de vuelo que implique movimientos autónomos o asistidos, por ello, esencial para el protocolo desarrollado.
- **Telemetría:** Se utiliza para la obtención de datos de vuelo en tiempo real.

6. Sistema Maestro Esclavo

En las redes informáticas, Maestro / Esclavo o primario / secundario es un modelo de comunicación en el que un dispositivo o proceso (conocido como el **Maestro**) controla uno o más dispositivos o procesos (conocidos como **Esclavos**). Una vez que se establece la relación Maestro / Esclavo, la dirección de control es siempre del Maestro al Esclavo (s) aunque en el caso del algoritmo desarrollado es bidireccional.

El algoritmo desarrollado es un sistema Maestro Esclavo bidireccional en el cual los Esclavos envían información al Maestro, tanto de posición como del estado en el que se encuentran. El Maestro recibe dicha información y la procesa para así dar las órdenes pertinentes a sus Esclavos. Se podría decir que el Maestro es el jefe del enjambre, y se encarga de que cada uno de los Esclavos esté en la misma etapa que los otros, y que haya una sincronía en el movimiento de todos. De esta manera se asegura que el enjambre se mueve de forma idéntica, y que se minimizan las fluctuaciones o desvíos.

Durante la etapa del despegue el Maestro se encarga de obtener los datos de **geoposicionamiento** de los Esclavos. Una vez dispone de estos datos es el Maestro el que realiza el cálculo del algoritmo de despegue.

Durante la etapa de vuelo, el Maestro se encarga de sincronizar a todos sus Esclavos en cada ***waypoint*** de la misión. Los *waypoints* son coordenadas para ubicar puntos de referencia tridimensionales utilizados en la navegación basada en GPS.

Por último, en el aterrizaje, el Maestro se asegura de que los drones Esclavos están en la etapa de la máquina de estados pertinente y, una vez todos están en su posición final, obtienen la orden de aterrizaje del Maestro.

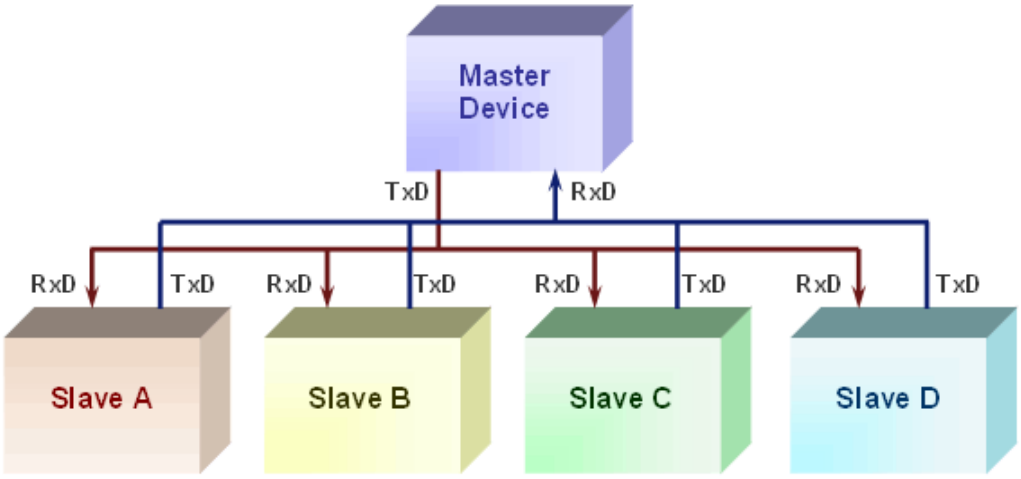

Ilustración 11 - Interacción Maestro Esclavo [20].

7. Algoritmo de enjambre

En este apartado se describirá el algoritmo de enjambre diseñado para cumplir con los requisitos de este TFM. Para ello se detallará en primer lugar la máquina de estados propuesta, seguido de una descripción de los formatos de mensaje utilizados, y los algoritmos para despegue y coordinación en vuelo, validación de *Waypoints* y aterrizaje.

7.1 Máquina de estados

Las máquinas de estado son un método para modelar sistemas cuyas salidas (outputs) dependen de la historia completa de sus entradas (inputs). Dicho de otra forma, el valor de las entradas condiciona el de las salidas. En comparación con los sistemas puramente funcionales, en el que la salida está claramente determinada por la entrada, las máquinas de estado tienen un rendimiento que está determinado por su historia. También se conocen como máquina de estados finitos (***FSM por finite state machine***) si el conjunto de estados de la máquina es finito; este es el único tipo de máquinas de estado que se pueden implementar en los computadores actuales. Las máquinas de estado sirven para modelar una amplia variedad de sistemas, por ejemplo:

- Diseño en la interacción con la interfaz de usuario.
- Desarrollo de IA para juegos por computador
- Maquinaria industrial
- El estado de una nave espacial, incluidas las válvulas que están abiertas y cerradas, los niveles de combustible y oxígeno, etc.

Como se ha podido comprobar, las máquinas de estados sirven para multitud de situaciones distintas. En este caso se aplicará para el control de un enjambre de drones, en concreto para el control de las etapas de vuelo en la que se encuentran cada uno de los drones, y para la sincronía de los mismos.

En todas las etapas las aeronaves tienen dos puertos de comunicación, en concreto un hilo de lectura y otro de escritura. Esto es necesario para que los multicópteros puedan comunicarse y recibir información en un mismo instante de tiempo, para así poder recibir y enviar información desde y hacia el Maestro.

A continuación se expone la máquina de estados desarrollada, se procede a definir cada uno de los estados, y a la definición de funcionalidad de los mismos.

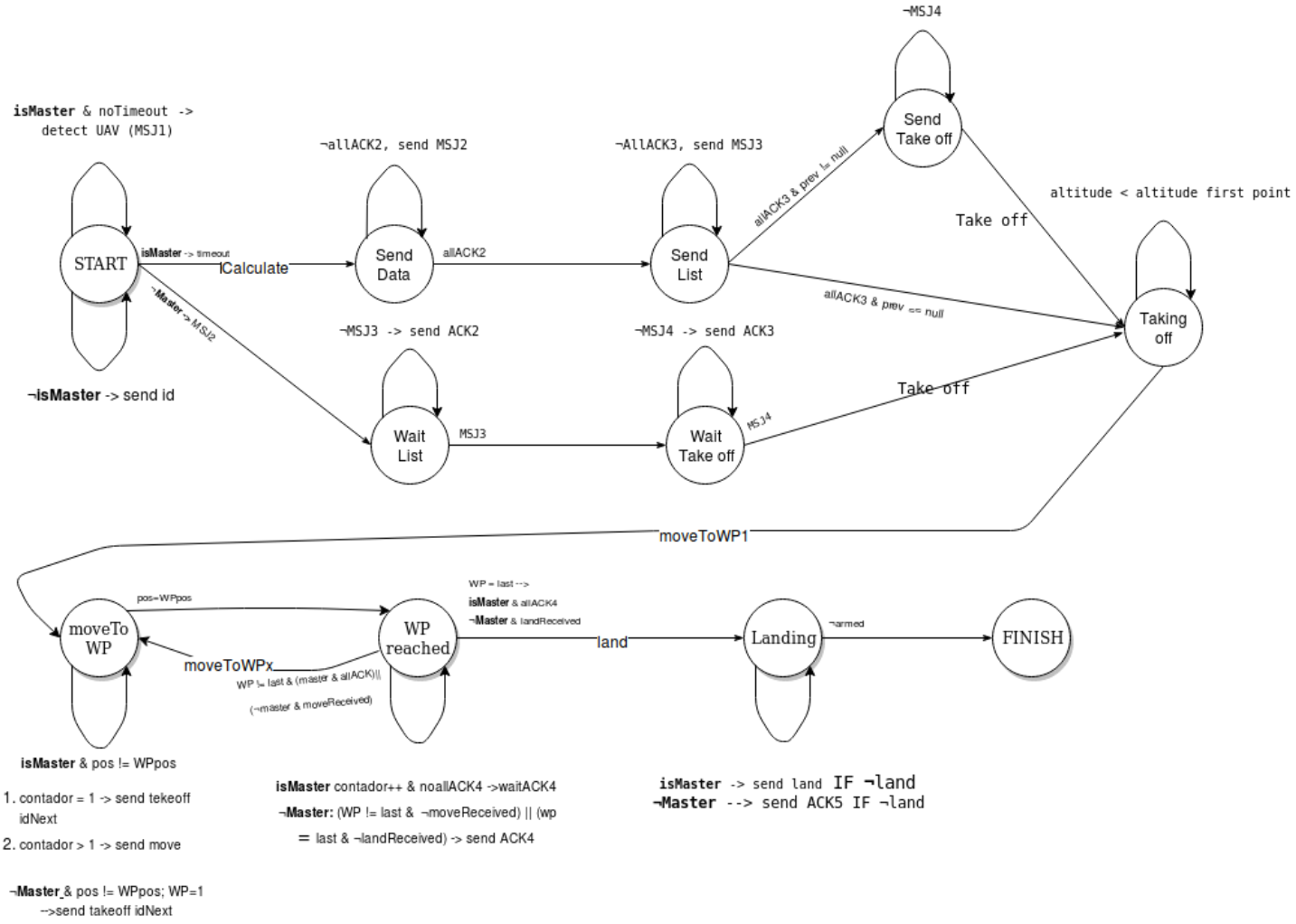

Ilustración 12 - Máquina de estados.

Para el análisis de la máquina de estados se describirán las etapas que lo forman, pero primero, desde el punto de vista del ***Maestro*** y después, desde el punto de vista de sus ***Esclavos***. Durante todo el desarrollo se ha tenido en cuenta si se trata de un dron simulado o real, ya que dependiendo de si estamos en un medio u otro, el algoritmo varia ligeramente.

Un ejemplo para entender esto son los identificadores de las aeronaves. En simulación son números enteros almacenados en un ***array*** (*1, 2, 3…n*), y en dron real se trata de la dirección **MAC** de las controladoras WiFi. Con ello se asegura que son únicas, y permite la identificación las aeronaves de forma inequívoca.

Estados multicóptero Maestro:

- **START:** En este estado el multicóptero se encarga de recibir el ***Mensaje 1***, por lo tanto, de sus dos hilos activos (*Listener* y *Talker*) solo está en uso el *Listener*. Este mensaje proviene de todos sus Esclavos, y se informa al dron Maestro cual es la posición desde la cual se pretende despegar, y el identificador de la aeronave (MAC o ID entero). El dron Maestro almacena en un ***HashMap*** cada una de las aeronaves que se dan de alta en el vuelo, y por último se incluye a sí mismo.

- **SEND DATA:** Durante esta etapa el hilo *Talker* se encarga de enviar el ***Mensaje 2***. Este mensaje contiene la información relativa a la **altura del primer punto de ascenso**. Esta altura se utilizaba durante la etapa de despegue para realizar un ascenso particionado y ordenado. Mientras tanto, el Hilo *Listener* está a la espera del *ACK* de este mismo mensaje.

- **SEND LIST:** En esta etapa el dron Maestro se encarga de enviar el ***Mensaje 3***. Este mensaje se envía con **los datos de vuelo personalizados** para cada aeronave. Para entender este cálculo, en futuras secciones se explica su funcionamiento en más detalle. En resumen, envía una ruta de vuelo precalculada, específica para cada aeronave, creando así el enjambre de drones. Mientras tanto, el Hilo *Listener* está a la escucha de los *ACK* enviados por los Esclavos, confirmando así la recepción de sus datos.

- **SEND TAKE OFF / TAKING OFF:** En esta etapa se incluyen los dos estados porque, dependiendo de las condiciones iniciales, se hará una etapa u otra.
    - SEND TAKE OFF: Esta etapa solo es alcanzable si el dron Maestro no es el primero en despegar. Como se verá en la sección dedicada al algoritmo de despegue, los drones despegan según su proximidad al primer *Waypoint*. Es por ello que, si el Maestro no es el primero, queda en esta fase aguardando que alguno de los Esclavos le ordene su despegue. Cuando esto ocurra pasará a la etapa TAKE OFF.
    - TAKING OFF: Si el dron Maestro es el primero de la lista de despegue pasará a esta etapa desde SEND LIST, y realizará la ascensión hacia el primer punto del despegue (Altura intermedia del primer *Waypoint*). Si no es el primero, viene desde el estado SEND TAKE OFF.

- **MOVE TO WAYPOINT:** Durante esta etapa el dron Maestro tiene distintas funciones. La primera de ellas es dar la orden para dirigirse al primer punto geográfico indicado (punto de salida a la altura final), la segunda función se realiza mientras el dron está en vuelo, en la cual debe estar escuchando por el Hilo *Listener* el ACK correspondiente a esta orden.

- **WP REACHED:** En esta etapa el Maestro espera la llegada *(ACK 9)* de todas las aeronaves al punto indicado en la etapa MOVE TO WAYPOINT. Si todavía quedan *Waypoints* en la misión, se vuelve a la etapa MOVE TO WAYPOINT para seguir con la misión. Si ya no quedan puntos de misión, se procede a acceder a la etapa LANDING.

- **LANDING:** Esta etapa se alcanza cuando todos los puntos de la misión han sido alcanzados. Una vez el dron Maestro se encuentra en esta etapa, escucha mediante su hilo Listener los paquetes *ACK 10* enviados por los Esclavos. Una vez detectados todos los drones, se procede al aterrizaje autónomo de forma simultánea.

- **FINISH:** Los drones ya han realizado el aterrizaje y se encuentran en el suelo desarmados. Se obtienen los resultados del experimento.

Estados de los multicópteros Esclavos:

- **START:** Los Esclavos envían al dron Maestro información mediante su Hilo *Talker*; en concreto la envían en el **Mensaje 1**. En este mensaje se incluye información de geoposicionamiento y de *Heading* de la aeronave. Durante esta etapa el hilo *Listener* estará escuchando el **Mensaje 2**, enviado por el dron Maestro, que contiene la altura del punto intermedio de despegue. Una vez enviada y recibida la información se pasa a la siguiente etapa.

- **WAIT LIST:** En esta fase los drones Esclavos envían el *ACK* correspondiente a la información recibida durante la etapa START, y quedan a la escucha del **Mensaje 3**, que contiene información personalizada para cada aeronave, la cual engloba la ruta de vuelo y un campo con el orden de despegue.

- **WAIT TAKE OFF:** Durante esta etapa, los drones Esclavos envían el ACK correspondiente a la etapa WAIT LIST (ACK 3), y quedan a la espera del **Mensaje 4** que es el encargado de indicar el despegue de los Esclavos. En ciertos cálculos de despegue es posible que este mensaje no lo reciba

un Esclavo si se trata del primero en despegar. De ser así, solo enviaría el ACK y despegaría.

- **TAKING OFF:** A este etapa acceden los drones Esclavos de dos formas distintas:
    - Directamente: Cuando el dron Esclavo es el primero en la lista de despegue.
    - Indirectamente: Cuando el dron Esclavo espera su turno para despegar. En el instante que el dron predecesor llegue a su altura intermedia de despegue, envía la orden para que este Esclavo realice el despegue a la altura intermedia.

- **MOVE TO WAYPOINT:** Durante esta etapa los drones Esclavos escuchan la orden de movimiento hacia un *Waypoint*. Una vez se recibe esta orden pueden realizar el viaje hasta su siguiente Waypoint. Si se trata del primer Waypoint, este viaje se realiza desde la altura intermedia hasta la altura y posición del inicio de su ruta, y si no es el primero avanzan hacia el siguiente punto de la ruta.

- **WP REACHED:** Una vez se ha llegado al punto de la ruta de vuelo indicado en la etapa anterior, se alcanza esta etapa. Los drones Esclavos envían un mensaje de *ACK* confirmando que han alcanzado este punto. De esta manera, en función de que punto sea: intermedio o final, pasar otra vez a la etapa MOVE TO WAYPOINT o aterrizar mediante la etapa LANDING.

- **LANDING:** Esta etapa se alcanza cuando los Esclavos llegan a su último *Waypoint*. Una vez alcanzado envían un mensaje de confirmación *ACK* avisando al dron Maestro de que se encuentran listos para realizar el aterrizaje. Mientras tanto esperan mediante su Hilo *Listener* la orden de descenso emitida por el Maestro, y una vez recibida aterrizan.

- **FINISH:** El dron ya ha realizado el aterrizaje y se encuentra en el suelo desarmado. Se obtienen los resultados del experimento.

7.2 Tipos de mensajes intercambiados

Durante la ejecución del algoritmo, se están enviando mensajes con información y mensajes del tipo ACK continuamente. Para que el algoritmo funcione de forma correcta la creación de estos paquetes y el envío deben de ser rápidos y gestionados de forma eficiente. Por ello se ha optado por el uso de la librería *Kryo*. Esta librería tiene multitud de usos, pero en este proyecto se utiliza para serialización por su rapidez en la creación y el envío de información. Esta información, a su vez, será enviada por los sockets UDP creados en cada uno de los multicópteros.

Como veremos en el ejemplo de código *(para el envío de datos)*, tanto la recepción como el envío se realiza de forma estructurada y clara:

```
//Se crea un paquete Krio con el tamaño del buffer de recepción
output = new Output(buffer);
//Se vacia el paquete
output.clear();

//Se escriben los datos que se desean enviar
output.writeShort(10);
output.writeLong(Param.id[numUAV]);
output.flush();

//Se introduce en el paquete UDP el Kryo
packet.setData(buffer, 0, output.position());

//Se envía el paquete
packet.send();
```

El tamaño del *payload* de los paquetes se puede calcular en función de las cabeceras que lo constituyen, por ello los paquetes utilizados en este proyecto se configuran de la siguiente forma:

Ilustración 13 - Paquete Swarm.

Como se ha podido observar en el estudio de la máquina de estados, existen distintos tipos de paquetes. A continuación se expondrán los que se han creado para la ejecución del algoritmo y sus propiedades:

- **Mensaje 1:** El primer mensaje se utiliza para la fase START y contiene los siguientes datos:

    - *Type:* Indica el tipo de mensaje, en este caso tipo 1.
    - *Id:* Para que cada dron solo reciba su información, el id es el encargado de hacerlo único, y que solo el destinatario reciba el paquete.
    - *Location*: Contiene la posición inicial del dron, Latitud y Longitud.
    - *Heading:* Almacena la orientación de la aeronave en tierra.

Ilustración 14 – Formato del Mensaje 1.

- **Mensaje 2:** El segundo mensaje se envía desde el Maestro, y contiene la altura del punto intermedio de despegue. Este punto se calcula en función de la altura final de la ruta de vuelo. Los datos contenidos en el paquete son los siguientes:

    - *Type*: Indica el tipo de mensaje, en este caso tipo 2.
    - Id: identifica el destinatario final.
    - *Altitude*: Altitud del punto intermedio del despegue.

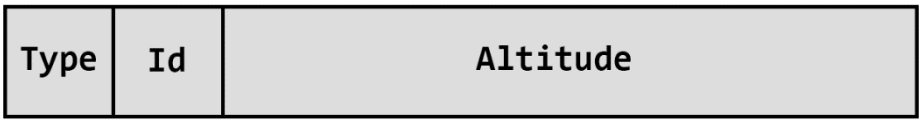

Ilustración 15 – Formato del Mensaje 2.

- **Mensaje 3**: Este paquete contiene la información relativa a la ruta de vuelo personalizada de cada aeronave y también contiene la información respecto al orden del despegue. A continuación se expondrán los datos que lo forman:

    - *Type*: Indica el tipo de mensaje, en este caso tipo 3.
    - *Id*: Identifica el destinatario final.
    - *IdP*: Contiene el identificador del dron que debe despegar, previamente al receptor de dicho paquete (Si existe un dron previo).

- *IdN*: Indica el dron que debe realizar el siguiente despegue. Este dato es necesario para que así se pueda dar la orden desde el dron que está despegando en ese mismo instante.
- *NºPt*: Este campo es necesario para que el receptor del mensaje pueda saber qué cantidad de *Waypoints* contiene el campo siguiente.
- *Data Points*: Contiene todos los puntos de la misión del dron receptor. Estos puntos son calculados por el dron Maestro, y personalizados para cada una de las aeronaves.
- Este paquete esta formado por los siguientes tipos de datos:

  Short + Long + Long + Long + Int + *n* Double

  Ocupando un total de 240 *bits*. Cada paquete puede tener un payload de máximo 1472 Bytes, es por ello que realizando el siguiente cálculo:

  $$1472 * 8 - 240 = 11536\ bits\ totales\ para\ payload$$

  Cada Waypoint está formado por coordenadas *X*, *Y*, *Z* por tanto cada uno ocupa 192 bits. Si se divide el número de *bits* disponibles en el payload entre el tamaño de un Waypoint:

  $$11536/192 = 60$$

  Podemos concluir que el número máximo de *Waypoints* que puede tener una misión es de **60 WP**

Type	Id	IdP	IdN	NºPt	Data Points

Ilustración 16 – Formato del Mensaje 3.

- **Mensaje 4**: El cuarto mensaje se utiliza para dar la orden de despegue, y contiene los siguientes campos:

  - *Type*: Contiene el tipo del mensaje, en este caso tipo 4.
  - *Id*: Identifica el dron al cual va dirigido el mensaje para que, una vez se reciba con su identificador, proceda con el despegue.

Ilustración 17 – Formato del Mensaje 4.

- **Mensaje 5:** Los mensajes del tipo 5 se utilizan para dar la orden de movimiento hacia el siguiente *Waypoint*. Estos mensajes son enviados por el Maestro, y contienen los siguientes campos:

    - *Type*: Contiene el tipo del mensaje, en este caso tipo 5.
    - *ID*: Identifica el dron al que va dirigido la orden de movimiento.

Ilustración 18 – Formato del Mensaje 5.

- **Mensaje 6:** Este mensaje indica el momento en el que se debe aterrizar. Se envía desde el dron Maestro hacia todos sus esclavos, y los campos que lo forman son los siguientes:
    - *Type*: Contiene el tipo del mensaje, en este caso tipo 6.
    - *ID*: Identifica el dron al que va dirigido la orden de aterrizaje**.**

Ilustración 19 – Formato del Mensaje 6.

- **ACKs:** Los ACKs son utilizados por el algoritmo para confirmar la recepción de los distintos mensajes anunciados anteriormente. El dron Maestro los utiliza para confirmar la recepción de la información que él mismo envía, y le permite así avanzar en su máquina de estados.
    - *Type*: En este caso este valor cambia en función de para cual mensaje sea el ACK; existen del tipo 8, 9, 10, 11,12 y 13, uno por cada tipo de mensaje distinto.
    - *Id*: Identifica la aeronave a la cual va dirigido el paquete.

Ilustración 20 – Formato de los ACKs.

7.3 Algoritmo de despegue

Uno de los momentos más críticos durante el vuelo de un enjambre de drones es el despegue. Cuando despegan las aeronaves suelen estar situadas en un radio pequeño, estando unas muy cerca de otras. Para poder realizar un despegue seguro y ordenado se debe de implementar un algoritmo que cumpla con estas dos premisas.

El algoritmo desarrollado funciona de la siguiente forma: El dron Maestro, durante su etapa START, está recibiendo información de todas las aeronaves que quieren formar parte del algoritmo de *Swarm.* La información que recibe el Maestro es la ubicación actual del esclavo y su identificador único. En este instante el dron Maestro dispone de tres datos necesarios para calcular el despegue:

1. Ubicación de los Esclavos.
2. Identificador de los Esclavos.
3. Rutas calculadas por el Maestro para la cantidad exacta que forma el enjambre.

Como se ha podido observar en los datos anteriores, el dron Maestro precalcula las rutas de los drones Esclavos conociendo solo la cantidad de drones que formarán el Enjambre. Posteriormente el algoritmo de despegue es el encargado de asignar cada una de esas rutas al dron Esclavo que más cerca esté de cada una de ellas, para así poder evitar cualquier tipo de cruce aéreo entre ellos. Como medida extra de seguridad, el despegue se realiza en dos fases, primero se despega hacia el punto intermedio y, una vez alcanzado dicho punto, se da la orden para que el siguiente dron despegue. Finalmente se dirige hacia su primer Waypoint:

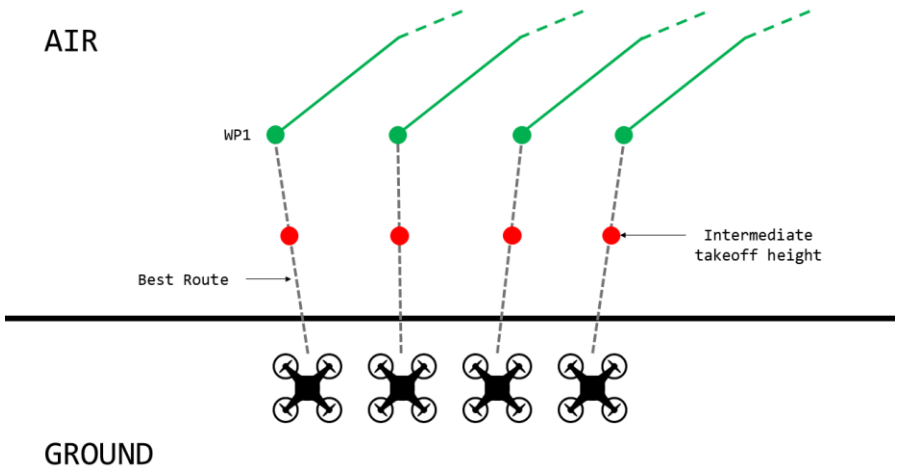

Ilustración 21 - Algoritmo de despegue propuesto.

El algoritmo desarrollado sigue los siguientes puntos durante su ejecución:

1. Se obtienen las posiciones iniciales de todos los multicópteros que forman parte del enjambre.

2. Se permuta el dron seleccionado en cada una de las rutas precalculadas por el maestro, y se busca cual es la ruta más cercana a él.

3. Se realiza el punto '2' para cada una de las aeronaves buscando así la mejor posición para cada una.

4. La mejor posición viene determinada por el siguiente cálculo: ajuste de mínimos cuadrado entre drones. Éste cálculo permite mediante el conocimiento de la geoposición de cada una de las aeronaves exagerar la distancia que separa a cada una de ellas en el suelo. Esto significa que, si en la disposición terrestre las aeronaves están relativamente cerca se pueda determinar claramente cuál es la mejor aeronave para cada una de las misiones precalculadas por el Maestro. Se realiza el siguiente cálculo iterativamente para cada *'distanciaDronSeleccionado' :*

$$acumulación = acumulación + distanciaDronSeleccionado^2$$

5. Una vez obtenida la distancia amplificada entre los drones en el suelo, se procede al cálculo de la mejor posición de cada aeronave en las posiciones aéreas que más cercana esté de cada dron.

6. Determinadas las posiciones más cercanas a cada una de las rutas, se selecciona cual será el dron que mejor se adapte a la ruta central, y se procede a su asignación.

7. Por último, se asigna a cada uno de los Esclavos la ruta precalculada por el Maestro que más cercana se encuentra de su posición inicial.

7.4 Algoritmo de vuelo

Durante el vuelo de las aeronaves se precisa de algún sistema que sincronice el movimiento de las mismas. Cuando se vuela formando un enjambre adquiere una importante relevancia la sincronización entre todos los actores del enjambre. Para que esto sea posible debe existir algún tipo de comunicación durante todo el trayecto de vuelo.

En el algoritmo desarrollado, se ha optado por sincronizar todas las aeronaves en cada uno de los Waypoints que forman la ruta de vuelo. Esta solución pretende que, para cada uno de los puntos de ruta, el Maestro tenga que ser el encargado de verificar que todas las aeronaves han alcanzado el mismo punto. Una vez estén todas en el mismo punto, el dron maestro procede a enviar la orden para que se continúe con la misión, y para que las aeronaves se dirijan hacia el siguiente Waypoint.

El principal componente del algoritmo de vuelo es la ruta que tiene cada aeronave. Esta ruta es la calculada por el Maestro, y cada una de ellas es exactamente igual a la otra pero con una variación de distancia entre drones. La *Ilustración 22* permite esclarecer este concepto:

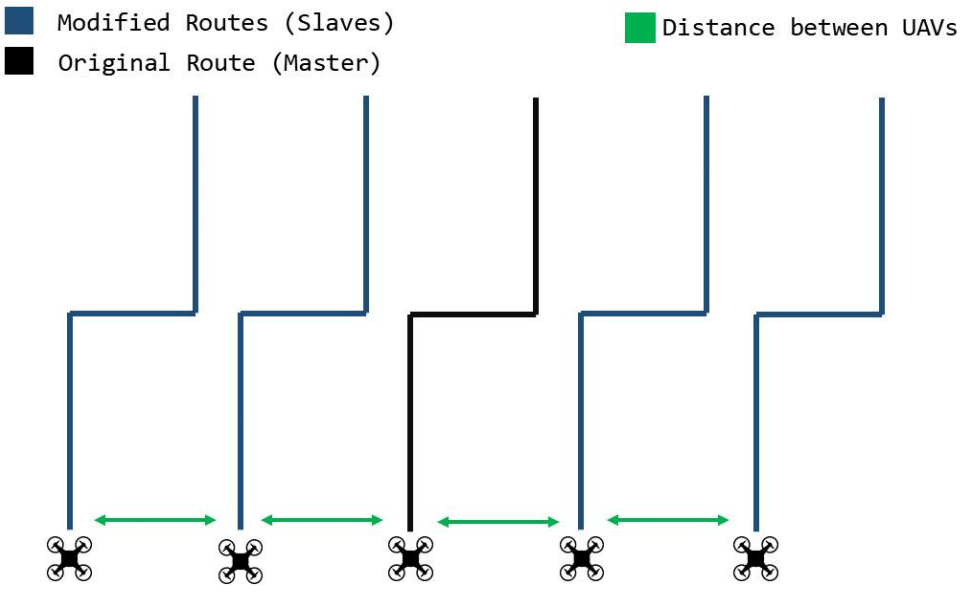

Ilustración 22 - Algoritmo de vuelo

Para hacer posible esta replicación de ruta, se ha desarrollado la siguiente fórmula matemática que replica en el eje X e Y la posición del Maestro:

$$X_2 = X_1 * \cos\alpha + Y_1 * \sin\alpha + X_0$$

$$Y_2 = Y_1 * \cos\alpha - X_1 * \sin\alpha + Y_0$$

El cálculo replica cada uno de los Waypoints en función de la ruta original y la cantidad de drones que forman el enjambre. La ruta original es la que tendrá almacenada el dron Maestro en su memoria. Durante la configuración del vuelo, esta fórmula permite modificar la separación existente entre las aeronaves en la ruta, permitiendo así adaptarse a las distintas necesidades de los usuarios.

A continuación, se ilustran dos configuraciones realizadas utilizando el simulador. En ellas se observa la configuración correspondiente y el algoritmo ejecutado para una misma ruta con distinta separación entre aeronaves:

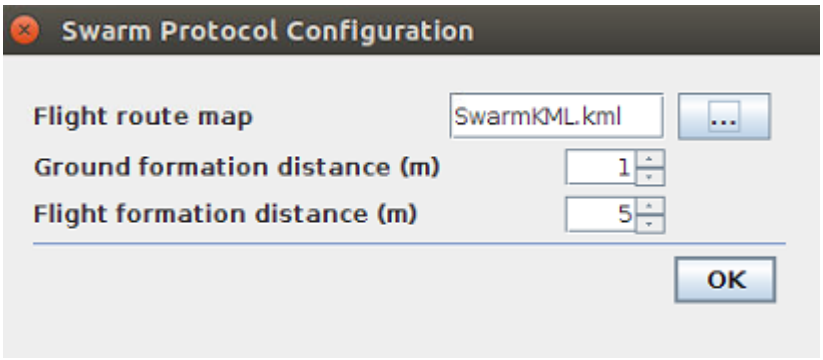

Ilustración 23 - Configuración Swarm 5m.

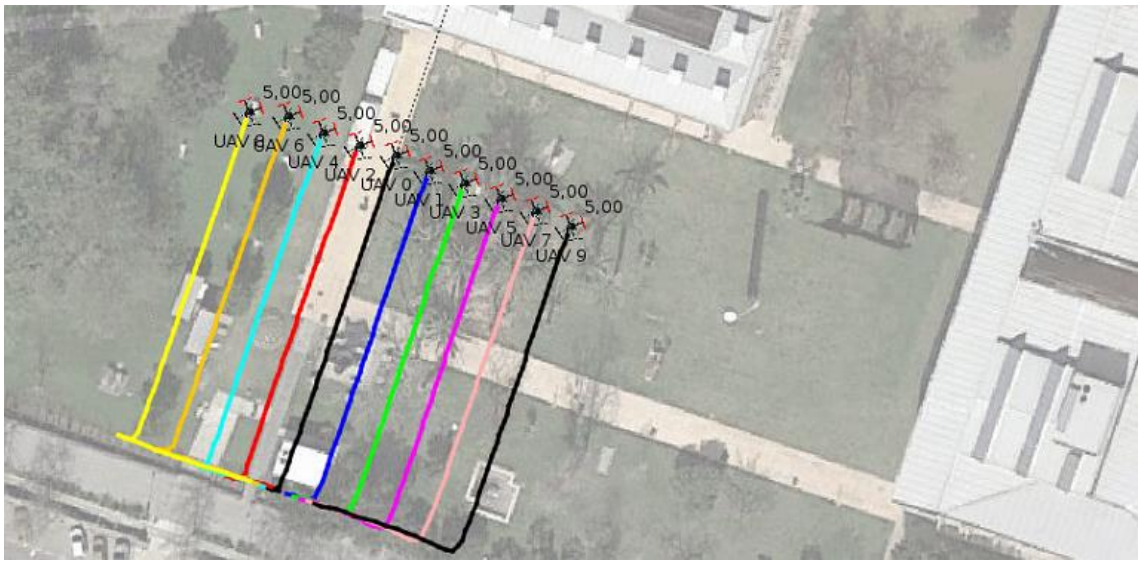

Ilustración 24 - Vuelo Swarm 5m

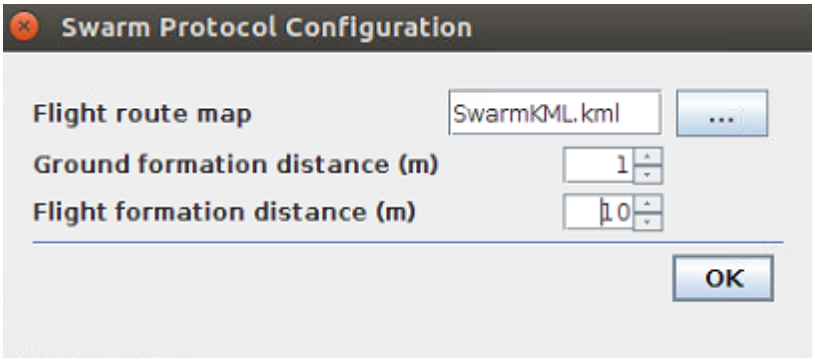

Ilustración 25 - Configuración Swarm 5m

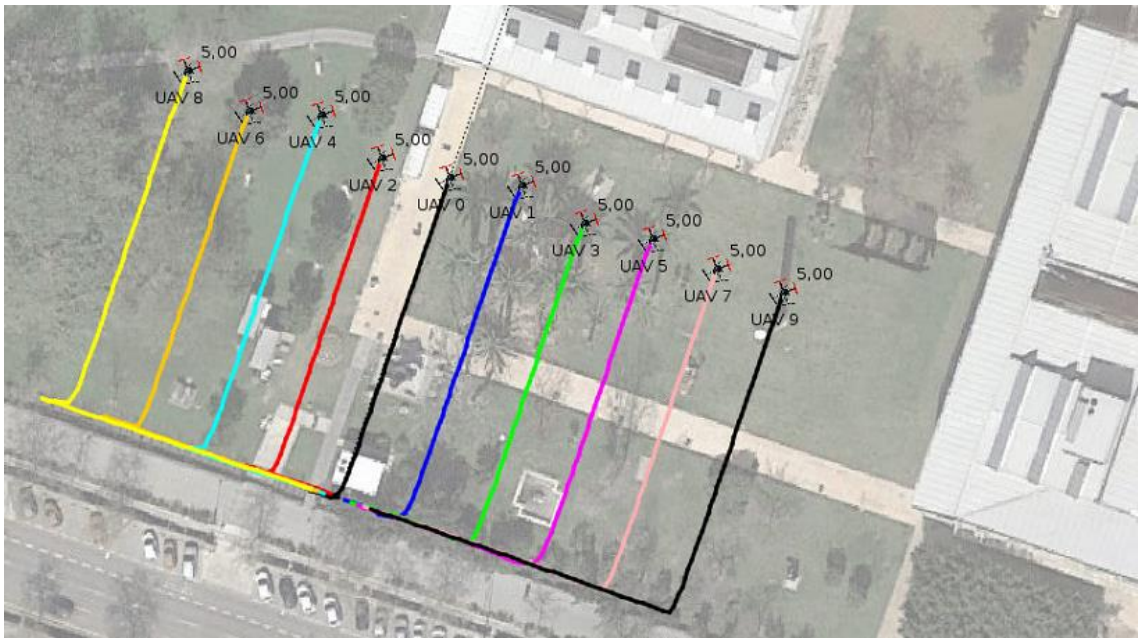

Ilustración 26 - Vuelo Swarm 5m

Como se puede observar en las ilustraciones anteriores, el algoritmo de *Swarm* ha sido diseñado para poder configurar las siguientes opciones:

- **Flight route map:** Donde se introduce la ruta de vuelo sobre la cual se formará el enjambre, y es la ruta que se asigna al Maestro por defecto.
- **Ground formation distance:** Permite configurar la distancia en el suelo entre las aeronaves seleccionadas para formar el *Swarm*.
- **Flight formation distance:** Es el parámetro utilizado para establecer la distancia entre drones cuando se encuentran en vuelo.

Para poder realizar un control exhaustivo del vuelo en enjambre, el simulador ofrece en todo momento el estado de la maquina interna de cada aeronave. Esta información se actualiza en tiempo real, y se muestra en la esquina derecha de la pantalla del simulador. En las dos ilustraciones siguientes se puede ver el estado de las aeronaves en dos instantes de tiempo distinto durante su vuelo.

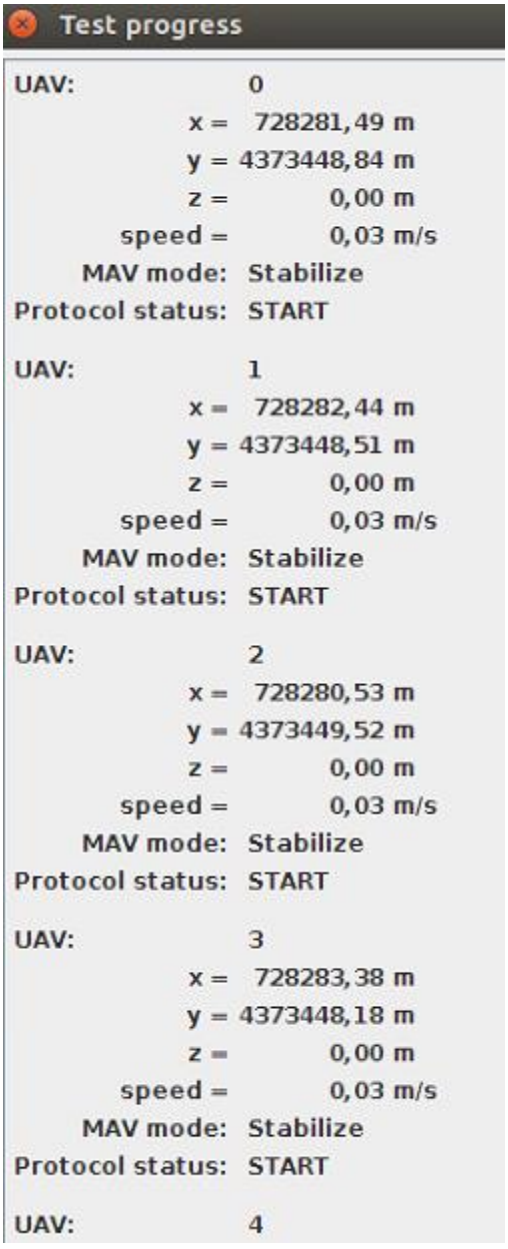

Ilustración 27 - Estado Simulador 1

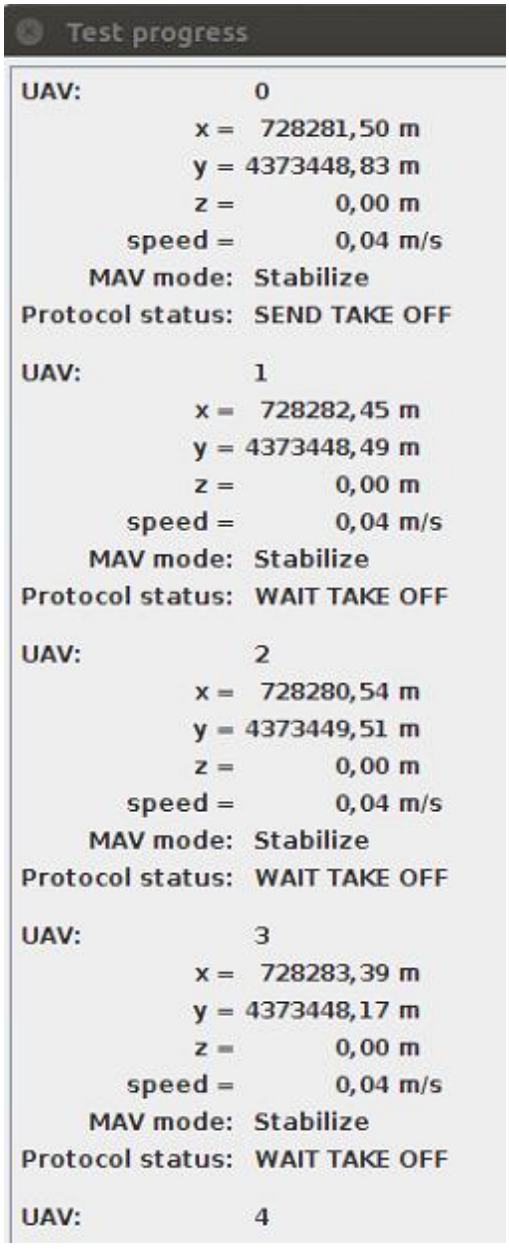

Ilustración 28 - Estado Simulador 2

7.5 Algoritmo de Waypoint alcanzado

Las rutas de vuelo almacenadas en las aeronaves están formadas por Waypoints. Estos puntos de vuelo se encargan de señalizar puntos por los cuales debe pasar la aeronave, para así cumplir con su misión. Si no se desarrolla ninguna solución para controlar estos eventos, las aeronaves detendrían su vuelo en cada uno de los puntos, reduciendo su velocidad para aproximarse al Waypoint de una forma segura y lenta. Posteriormente se dirigirían a su siguiente punto en la misión.

En un vuelo en enjambre, no se desea este comportamiento ya que el vuelo seria lento y con interrupciones. Para evitar esto, se ha desarrollado un método que se encarga de detectar a qué distancia se considera que el Waypoint ha sido alcanzado. Esta medida es posible gracias a la inclusión de un parámetro llamado *threshold*.

Modificando el **threshold** podemos definir a partir de qué distancia respecto al Waypoint se puede considerar que éste ya ha sido alcanzado, ofreciendo, durante ese intervalo de tiempo (hasta llegar realmente al Waypoint), la posibilidad de realizar envíos de datos entre las aeronaves, y crear un paso por los Waypoints más suave y rápido.

```
UAVParam.newLocation[numUAV][0] = (float)geo.latitude;
UAVParam.newLocation[numUAV][1] = (float)geo.longitude;
UAVParam.newLocation[numUAV][2] = relAltitude;
UAVParam.MAVStatus.set(numUAV, UAVParam.MAV_STATUS_MOVE_UAV);
while (UAVParam.MAVStatus.get(numUAV) != UAVParam.MAV_STATUS_OK
        && UAVParam.MAVStatus.get(numUAV) != UAVParam.MAV_STATUS_MOVE_UAV_ERROR) {
    GUIHelper.waiting(UAVParam.COMMAND_WAIT);
}
if (UAVParam.MAVStatus.get(numUAV) == UAVParam.MAV_STATUS_MOVE_UAV_ERROR) {
    SimTools.println(SimParam.prefix[numUAV] + Text.MOVING_ERROR_1);
    return false;
} else {
    UTMCoordinates utm = GUIHelper.geoToUTM(geo.latitude, geo.longitude);
    Point2D.Double destination = new Point2D.Double(utm.Easting, utm.Northing);
    // Once the command is issued, we have to wait until the UAV approaches to destination
    while (UAVParam.uavCurrentData[numUAV].getUTMLocation().distance(destination) > destThreshold
            || Math.abs(relAltitude -
            UAVParam.uavCurrentData[numUAV].getZRelative()) > altThreshold) {
        GUIHelper.waiting(UAVParam.STABILIZATION_WAIT_TIME);
    }
    return true;
}
```

El algoritmo utiliza la ubicación real del dron, y comprueba en tiempo real cual es la distancia entre esta posición y la del futuro Waypoint. Si esa distancia es menor a la establecida por *destThreshold* continua su movimiento, y de lo contrario detectará que ya está en zona de WP_REACHED y procederá a ejecutar las instrucciones necesarias para continuar con el vuelo.

7.6 Aterrizaje

Para realizar el aterrizaje se barajaron distintas opciones. Dependiendo del uso que se le fuese a dar al algoritmo, unas podrían ser mejores que otras, por ejemplo, para aterrizar en espacios reducidos sería conveniente que todos los drones se acercaran a la distancia mínima segura entre ellos y aterrizasen. Esta solución es más pesada en cuanto a tiempo de aterrizaje que la empleada en el algoritmo, y sería conveniente realizarla si fuesen a utilizarse grandes cantidades de aeronaves. La solución final por la que se optó fue la más sencilla y rápida de ejecutar, pero a su vez la que más terreno necesita, ya que el aterrizaje se realiza conservando la distancia existente entre las aeronaves en su último Waypoint de la misión.

Para que el enjambre aterrice, los Esclavos deberán notificar al Maestro mediante un mensaje, indicando que ya se encuentran en su punto final de la misión. Una vez en dicho punto, los Esclavos y el Maestro comprueban la información recibida y, si es correcta, el Maestro manda un mensaje informando de que todos deben pasar a su etapa *LANDING*.

Una vez recibida la orden, las aeronaves proceden a su aterrizaje autónomo, el cual finalizará al llegar a tierra. Automáticamente la controladora de vuelo cambiara su estado a *LANDING DISARMED*. Esto significa que las aeronaves se encuentran en tierra con todos los motores apagados, y que ha sido satisfactorio su aterrizaje.

Por último, si nos encontramos realizando los vuelos mediante simulación, aparecerá una ventana informándonos de datos relevantes del vuelo y, si se realizó con aeronaves reales, se almacena en memoria un log con la misma información.

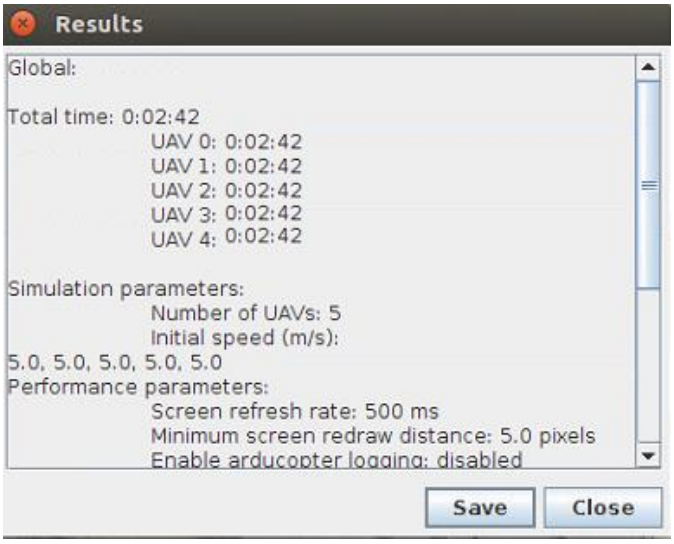

Ilustración 29 - Resultados del vuelo.

En la *Ilustración 29* se muestran los datos tras finalizar el vuelo. Los datos ofrecidos tras la ejecución son los siguientes:

- **Total time:** En este campo se muestra el tiempo de vuelo de cada una de las aeronaves que forman el enjambre.

- **Simulation parameters:** Este apartado muestra diversos datos en cuanto a la configuración del vuelo. Los elementos más importantes son *Number of UAVs* e *Initial Speeds*. Los valores de *Initial Speeds* se establecen en el menú principal del simulador *Ardusim*. Estos campos están establecidos por el desarrollador del mismo.

Tras la finalización del vuelo, el estado final del simulador es el mostrado en la siguiente ilustración:

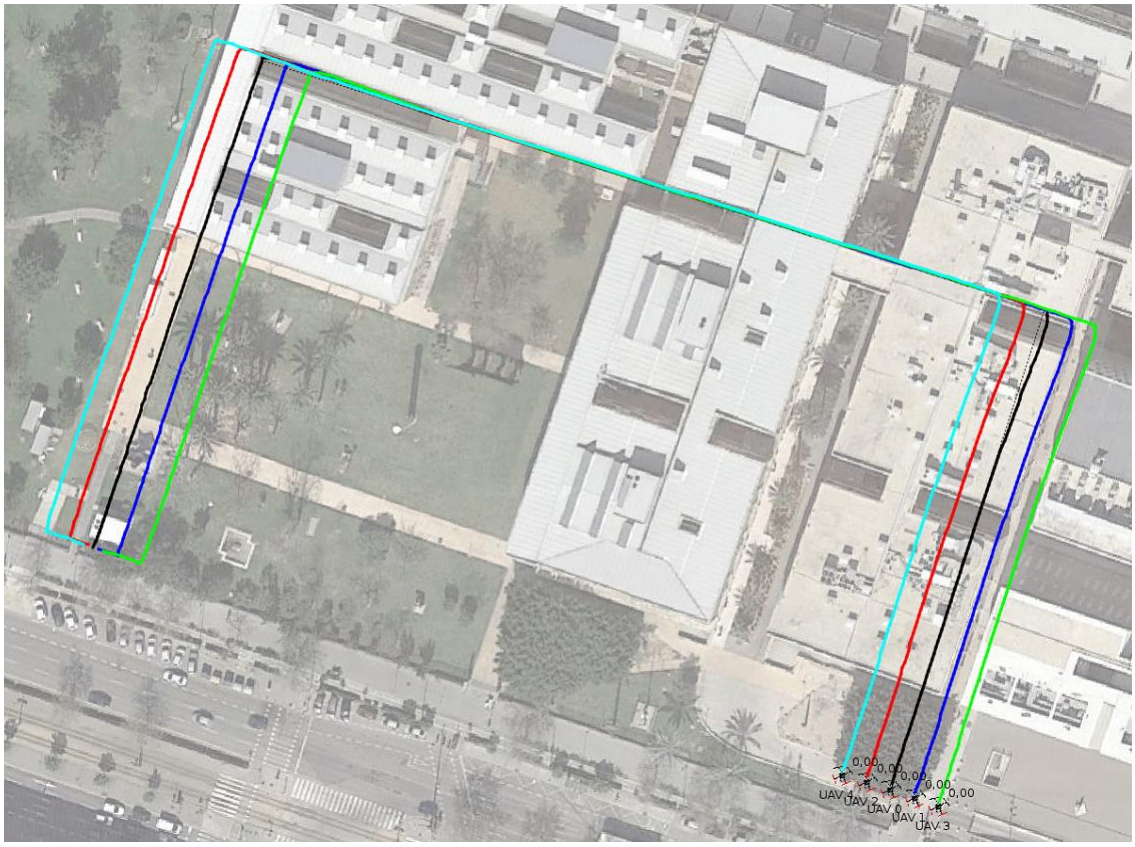

Ilustración 30 - Vuelo completado

8. Validación

Para realizar la validación del algoritmo desarrollado, se evalúan las distintas fases que forman el vuelo. Como estación de pruebas se utiliza un PC con las siguientes características, sobre el cual se ejecutará Ardusim:

PC COMPONENTES	
CPU	Intel Core i7 7700K
RAM	32GB DDR4
TARJETA GRAFICA	NVIDIA GT 710 2GB
S.O	Ubuntu 16.04

Tabla 4 - Componentes PC usados para simulaciones.

8.1 Evaluación de prestaciones

La aplicabilidad del algoritmo es muy alta, ya que se podría aplicar a tantos usos como ideas se tengan, pero para escenificar uno de sus posibles usos se concreta una prueba, detallada a continuación, que permite validar su uso.

Para comprobar uno de sus beneficios frente al uso unitario de aeronaves se realiza una prueba simulada en la cual se pretende comparar la cantidad de terreno que puede analizar un UAV frente a la cantidad de terreno que pueden analizar tres UAVs volando como un enjambre. En ambas pruebas los drones recorren la misma distancia (*500 metros*).

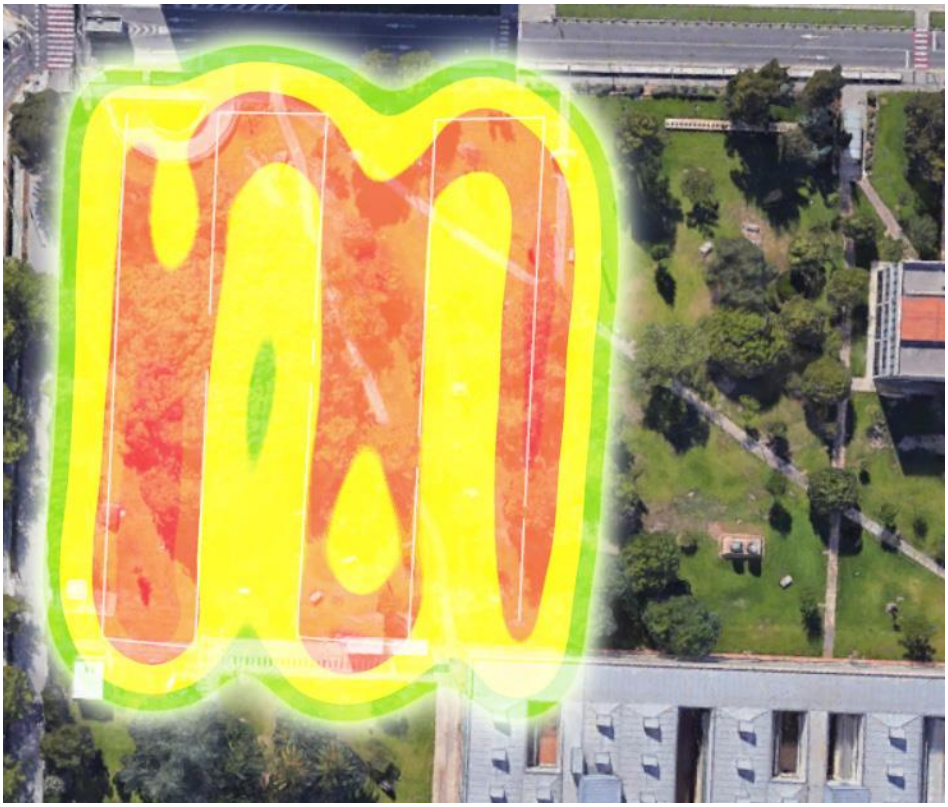

Ilustración 31 - Mapa de calor 1 dron

Ilustración 32 - Mapa de calor 3 drones

En base a los resultados podemos apreciar su beneficio frente al uso de una aeronave. Con tres aeronaves podemos analizar una cantidad de terreno más amplia en el mismo tiempo. Esta posibilidad amplía el uso de estas aeronaves para realizar escaneos 3D de terrenos, agricultura o imagen. Con una sola aeronave no se podían analizar grandes parcelas. Esto se debe esencialmente a que este tipo de aeronaves tiene una duración de vuelo baja, y la solución para recorrer grandes extensiones de terreno debía de ser la detención de la prueba y el cambio de la batería.

Con la solución aportada, los drones unitariamente recorren la misma distancia, pero al trabajar en cooperación pueden abarcan cantidades mayores de terreno facilitando su aplicabilidad. Los mapas de calor muestran las zonas que abarcarían las cámaras integradas en ellos, ajustando el color según la cantidad de imágenes tomadas por zona. El color rojo indica cual es la zona más filmada y la verde la que menos, pero todas ellas se exploran.

8.2 Evaluación del despegue

Una de las etapas más críticas durante la ejecución del algoritmo es el despegue. Esto se debe a que las aeronaves parten de su ubicación terrestre (normalmente muy próximas unas de otras), y deben despegar hacia sus posiciones iniciales en el espacio aéreo. Durante esta etapa existe una posibilidad más alta de que las aeronaves choquen, y por ello se realizó el despegue en dos fases, y con un orden secuencial.

La implantación de estas medidas de seguridad para evitar problemas durante el despegue causa un impacto temporal en la ruta de vuelo, ya que se está añadiendo una capa de complejidad al despegue de las aeronaves.

Para verificar el impacto introducido por estas medidas, se realizaron diversas simulaciones en distintos entornos de vuelo y los resultados obtenidos pueden verse reflejados en los siguientes datos.

En las primeras de las pruebas se analizó el **impacto temporal del despegue,** y se comparó el mismo despegue a distancias distintas entre las aeronaves.

Numero de UAVs	***t.(s) 5m de distancia***	***t.(s) 20m de distancia***
2	28,053	28,654
3	38,603	38,658
4	48,002	50,253
5	57,803	58,855
6	68,203	70,257
7	78,803	79,707
8	87,799	97,866
9	98,201	117,962
10	108,203	120,361
11	116,204	125,862
12	125,002	140,213
13	137,001	152,517
14	144,202	167,069
15	155,803	172,99

Tabla 5 - Tiempos de despegue.

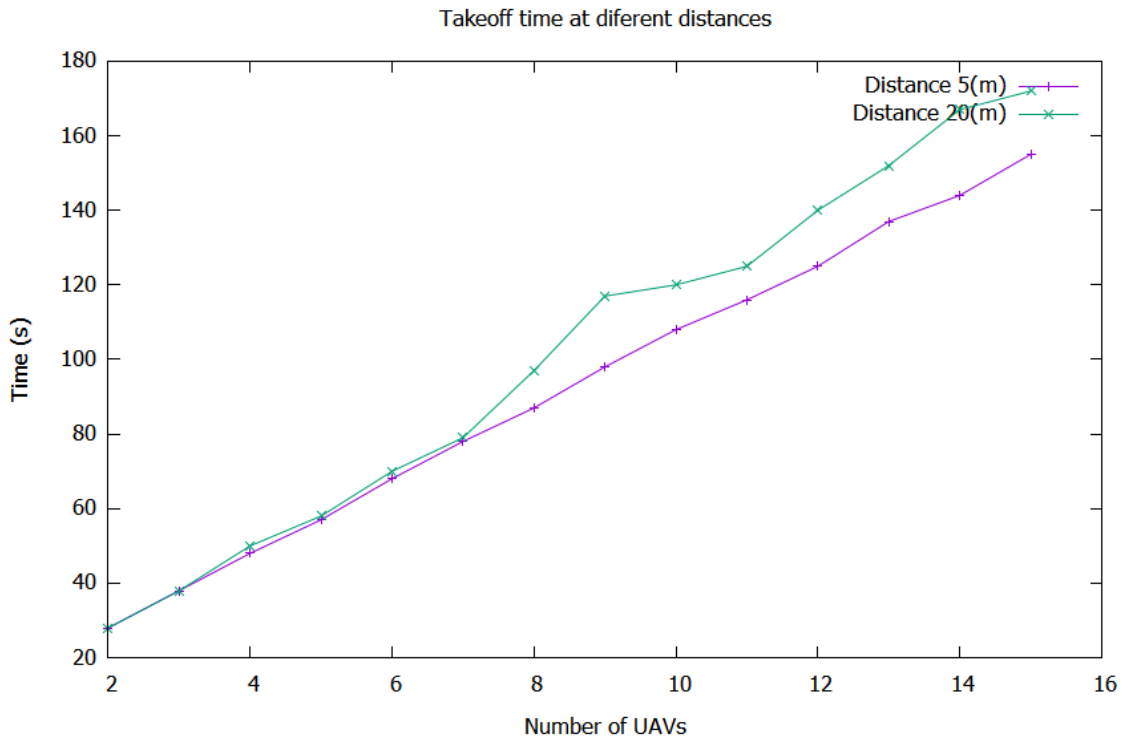

Tabla 6 - Grafica comparativa tiempo de Takeoff.

Como se puede observar en la gráfica, la distancia entre drones no afecta al tiempo de despegue hasta que se supera la cantidad de 6 drones. Esto se debe a la optimización del algoritmo.

El despegue de los drones se realiza secuencialmente, pero deja como último dron en despegar el central. Las demás aeronaves despegan de la posición más cercana del centro a la más lejana, y es por ello que el tiempo de despegue no difiere mucho cuando tenemos pocas aeronaves. Cuando se aumenta la cantidad, el tiempo empieza a subir cuanto mayor es la distancia. Esto se debe a que el tiempo que tarda la penúltima aeronave en llegar a su posición final es mayor que el tiempo de despegue del dron central (que es el último en despegar) a su altura intermedia de despegue. Se puede concluir que el tiempo de despegue viene marcado por el tiempo de traslado del penúltimo dron a su posición final de despegue cuando el tiempo del penúltimo dron a su posición inicial es mayor que el tiempo de despegue del dron central.

Antes de poder realizar el despegue, el dron central debe realizar el cálculo de la mejor combinación para realizar el ascenso. Este cálculo se realiza para obtener cual es la posición idónea para cada uno de los drones en el aire. Con esto se consigue eliminar por completo la posibilidad de que las aeronaves choquen durante el despegue. Este cálculo elimina por completo la posibilidad de choque, pero implica un alto coste computacional. A continuación se pude observar gráficamente el coste temporal del cálculo.

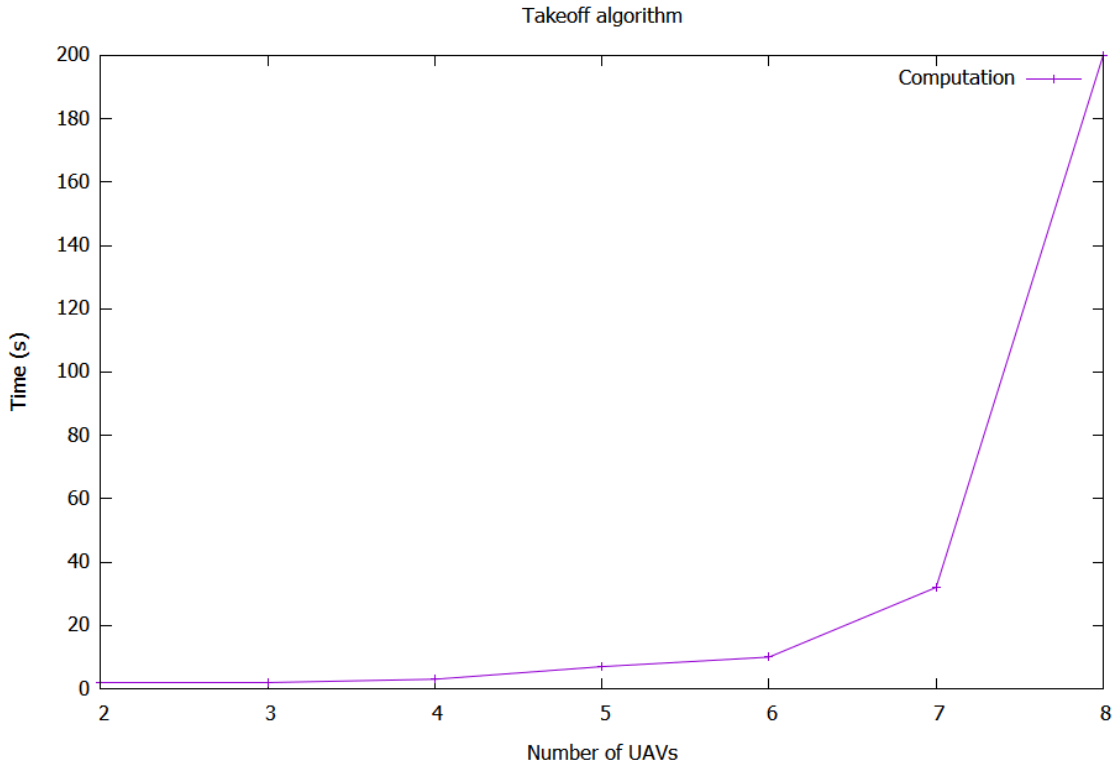

Ilustración 33 - Cálculo del despegue óptimo.

Tras el análisis de los resultados obtenidos, se puede concluir que el algoritmo es muy seguro, pero implica unos costes computacionales/temporales altos, y por ello la solución aportada no sería la más conveniente para cantidades ingentes de drones. Para cantidades más contenidas la solución es idónea.

Para realizar un análisis realista del coste temporal del despegue se debe contabilizar el coste temporal tanto del cálculo como de la aplicación del despegue. Es por ello que se realizó el siguiente grafico comparativo.

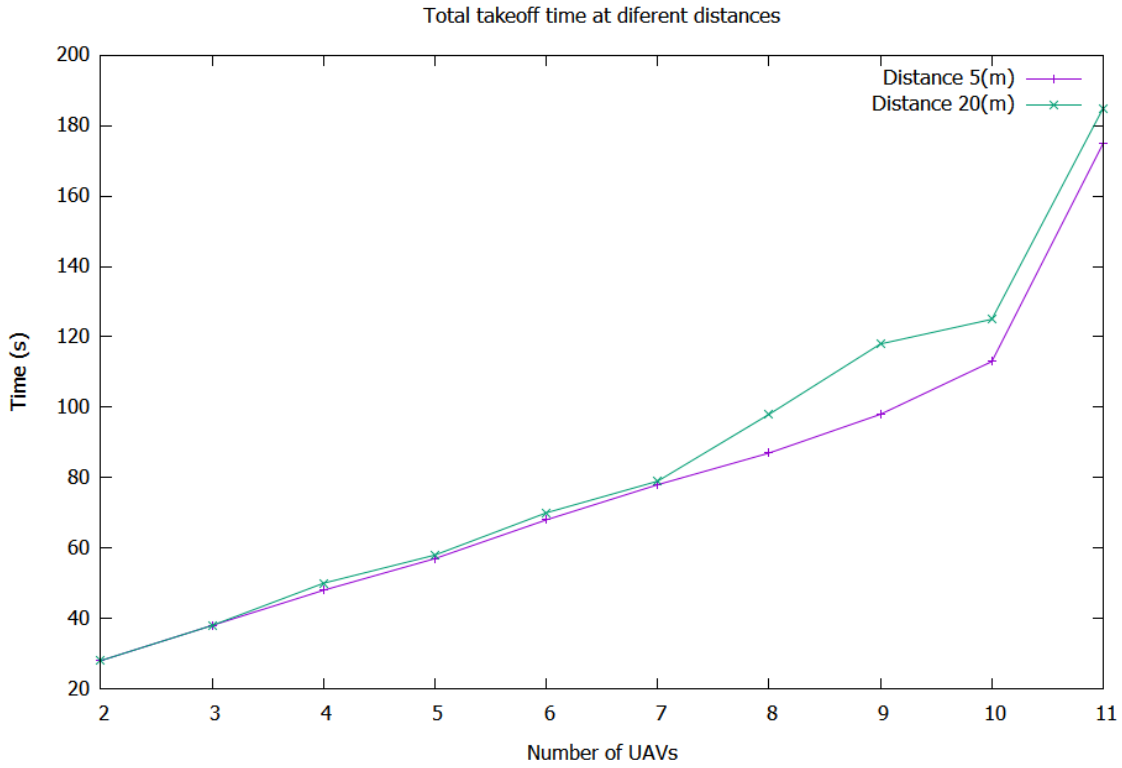

Ilustración 34 - Tiempo total de despegue

Como conclusión, se observa que, con el aumento del número de drones se aumenta el tiempo de despegue, pero especialmente a causa del cálculo. Si se aumenta la distancia de ascenso el tiempo total varía ligeramente a causa del coste superior del cálculo.

Se podría optar por una solución menos segura que tuviese un coste computacional menor pero, en este caso, se requería una alta seguridad en el despegue de las aeronaves. Este requisito, junto con el uso que se pretende dar a este algoritmo (cantidades entorno a los 15 UAVs), fueron los que hicieron que se decidiese por la opción más costosa computacionalmente.

8.3 Evaluación de rutas de vuelo

Durante la fase de vuelo, el algoritmo debe de ser lo suficientemente rápido como para no influir en el tiempo de la ruta. Para ello el algoritmo debe anticiparse a los *Waypoints* y gestionarlos de forma rápida y segura. Detalles del funcionamiento del sistema de aproximación a *Waypoints* están descritos en el apartado *7.5*. Para validar el algoritmo desarrollado, se realizaron diversos tipos de pruebas.

La primera prueba fue verificar si el aumento del número de *Waypoints* en las rutas de vuelo penalizaba el tiempo total de vuelo. Podría darse el caso si la validación de Waypoint alcanzado no estuviese bien definida. Los resultados obtenidos fueron los siguientes.

NUMERO DE UAVS	2 GIROS DISTANCIA * 1	4 GIROS DISTANCIA * 2	8 GIROS DISTANCIA * 4	18 GIROS DISTANCIA * 8
2	47,897	72,919	118,897	220,458
3	49,587	73,009	119,421	224,589
4	48,083	73,587	119,324	221,458
5	49,241	72,936	120,032	221,475
6	48,436	74,523	119,675	222,047
7	49,047	73,222	119,654	224,273
8	49,03	74,06	120,358	225,475
9	48,121	73,517	119,785	221,987
10	49,326	74,968	119,978	225,893
11	48,546	74,319	120,745	225,05
12	50,411	74,024	120,458	225,851
13	50,192	74,555	120,136	224,587
14	50,858	75,241	119,229	223,188
15	50,957	74,983	121,687	222,638

Tabla 7 - Ruta de vuelo lenta.

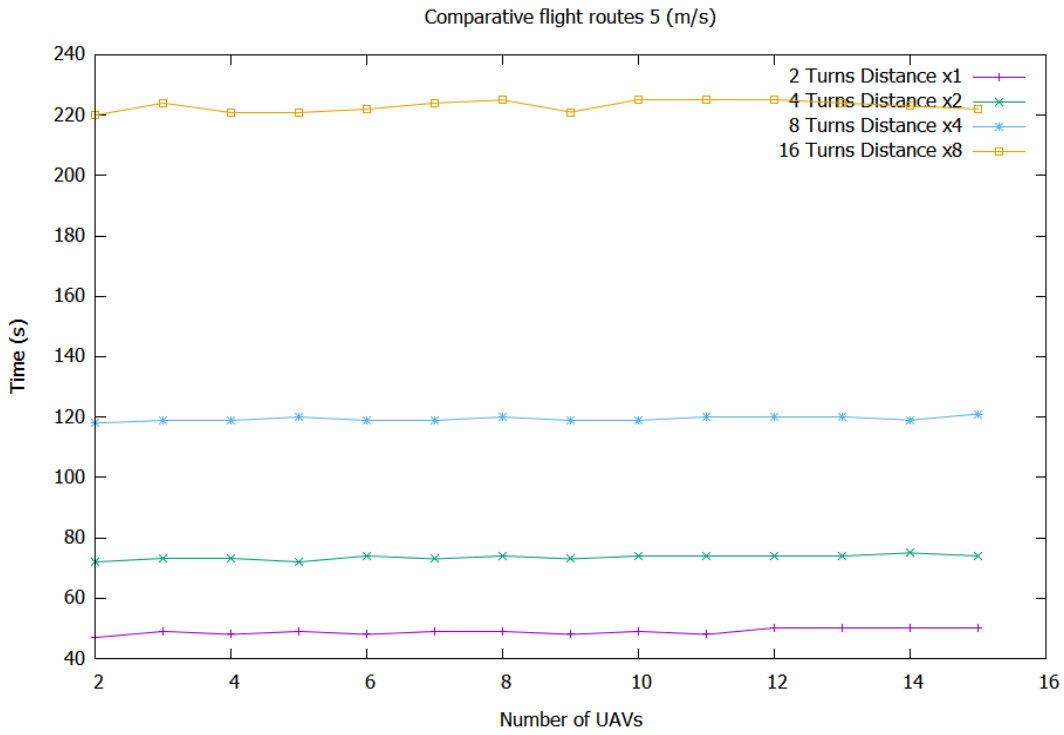

Ilustración 35 - Ruta de vuelo lenta.

Tras analizar la gráfica resultante se puede afirmar que no existe penalización por el incremento de *Waypoints* en el algoritmo de vuelo, ya que el tiempo también se mantiene uniforme aumentando la cantidad de aeronaves.

Las diferencias de tiempos vienen dadas por el aumento de la distancia en las rutas. Cada vez que se añaden dos giros a la ruta, la distancia se duplica respecto de la anterior.

Por último, para verificar no solo si el número de *Waypoints* influye, se aumenta la velocidad de las aeronaves a velocidades extremas, y se analiza si la rápida aproximación a los *Waypoints* desestabiliza el tiempo de vuelo. Con ello se pretende comprobar si el algoritmo de aproximación al Waypoint realiza su función con la suficiente antelación como para realizar la ruta sin cambios de dirección erróneos, o si aparecen imperfecciones en la ruta.

NUMERO DE UAVS	2 GIROS DIST * 1	4 GIROS DIST * 2	8 GIROS DIST * 4	18 GIROS DIST * 8
2	25,709	70,025	113,46	210,021
3	25,021	71,439	113,852	210,067
4	24,479	71,112	114,624	214,269
5	24,856	71,115	114,109	214,11
6	25,472	70,984	114,789	214,054
7	23,417	70,446	114,633	213,989
8	25,752	70,754	115,037	214,782
9	23,892	70,025	115,879	214,587
10	25,534	71,421	115,002	215,984
11	24,074	70,699	115,098	215,047
12	25,098	72,132	114,887	214,997
13	25,874	70,742	115,007	214,983
14	25,545	71,487	115,048	215,018
15	25,427	70,764	115,603	215,576

Tabla 8 - Ruta vuelo rápida

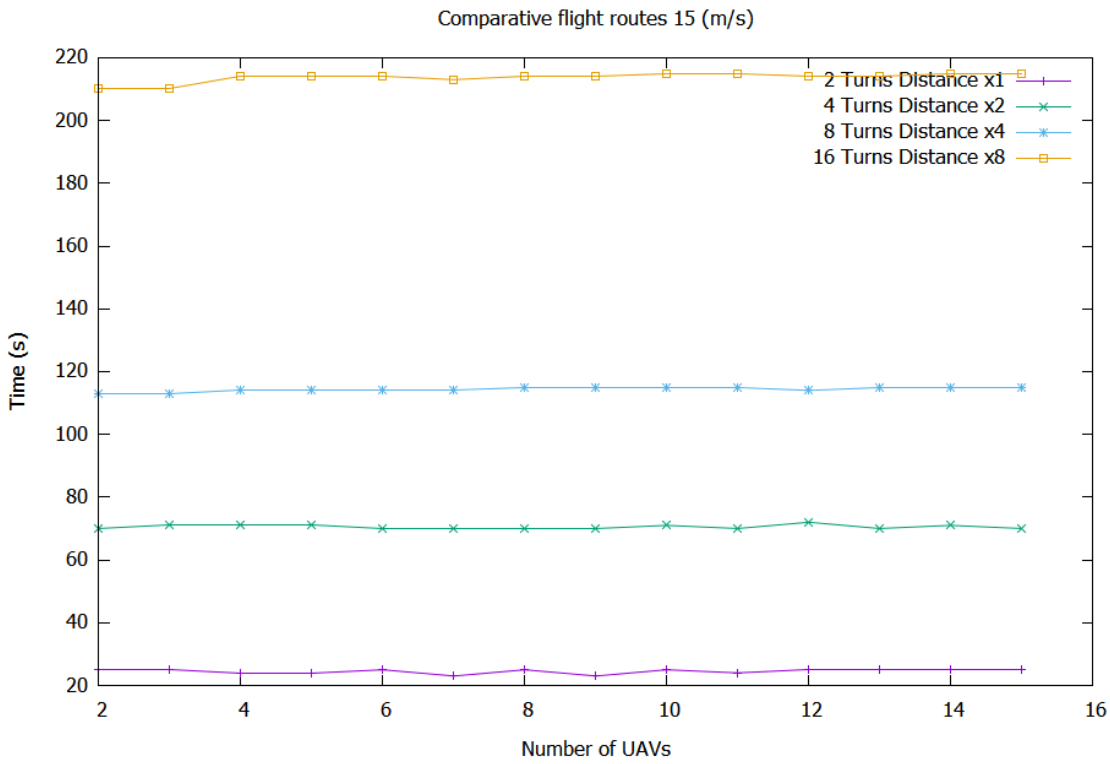

Ilustración 36 - Ruta de vuelo rápida

Como se puede observar en los resultados, el aumento de la velocidad no influye negativamente en el tiempo de vuelo, sino que se comporta con el resultado deseado, que es, a más velocidad, menor tiempo de vuelo sin importar el número de *Waypoints* que formen la ruta.

8.4 Evaluación del aterrizaje

Para realizar el aterrizaje se barajaron distintas opciones. Como se detalló en apartados anteriores, se optó por un aterrizaje simultáneo manteniendo la formación establecida en vuelo. Con ello se reduce el tiempo de aterrizaje, pero se necesita más espacio para realizarlo. Modificar el algoritmo de aterrizaje para variar su funcionalidad e implementar nuevas soluciones es posible, ya que se programó de manera que se pudiesen introducir variantes y que el algoritmo de enjambre continuase funcionando.

Para verificar que el aterrizaje no se ve perjudicado temporalmente, se realizan pruebas con distintas cantidades de drones.

Numero de UAVs	***t.(s) Aterrizaje***
2	12,851
3	13,654
4	13,454
5	12,657
6	13,65
7	13,266
8	13,466
9	12,857
10	13,941
11	13,092
12	13,659
13	12,954
14	13,702
15	13,645

Tabla 9 – Aterrizaje.

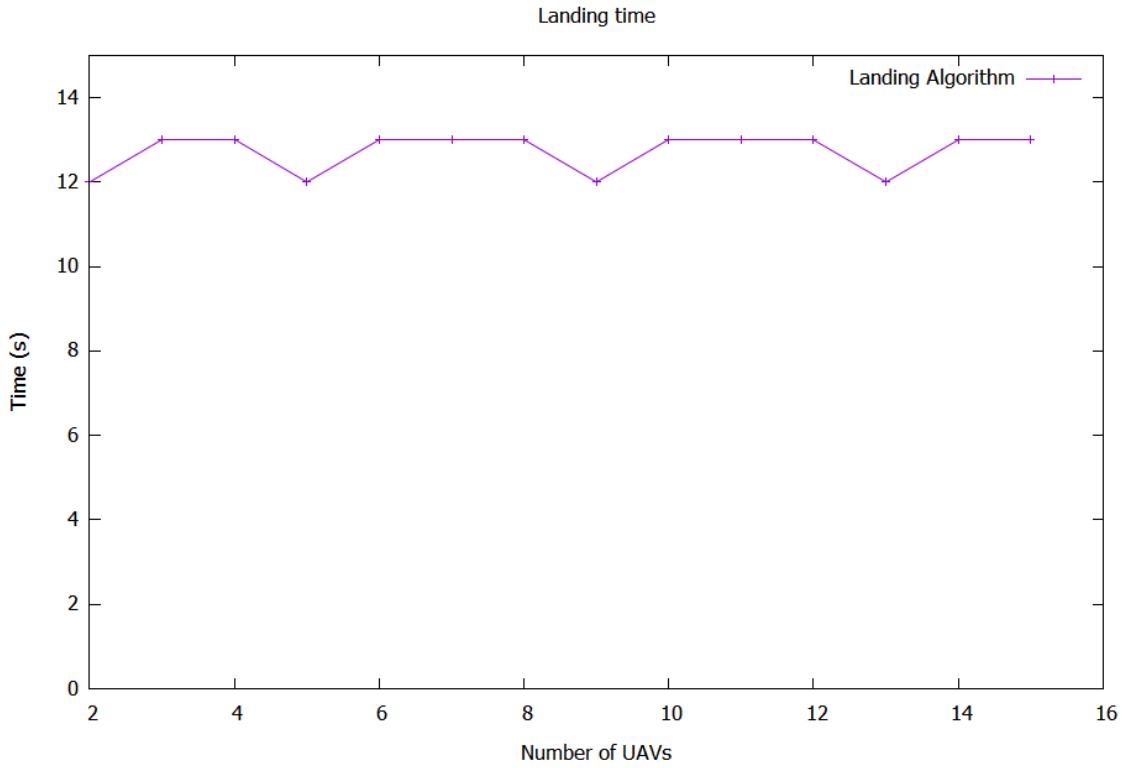

Ilustración 37 - Aterrizaje

Los resultados obtenidos fueron los esperados, ya que los tiempos de aterrizaje no se vieron penalizados por la cantidad de aeronaves que forman el enjambre. En la gráfica anterior se observa como todas las aeronaves aterrizan en un período cercano a los 13 segundos.

9. Conclusiones

Todo proyecto conlleva la consecución de unos objetivos. Consideramos que los objetivos del presente proyecto se han alcanzado de manera plena y satisfactoria.

El primer objetivo se basaba en obtener una solución para el control de enjambres basados en el protocolo de comunicaciones más utilizado en este tipo de aeronaves. Se pudo desarrollar satisfactoriamente mediante la interfaz MAVLINK, ofreciendo así una solución aplicable a la gran mayoría de drones existentes en el mercado.

Por otra parte dicha solución debía ser óptima para su uso en escenarios reales, y no aportar un sobrecoste temporal ni en componentes hardware. El coste temporal ha sido reducido como se ha podido observar en la evaluación, y el coste en cuanto a hardware ha sido ínfimo, ya que bastaría con añadir una Raspberry Pi a los drones que vayan a participar en el enjambre.

Un punto importante en el desarrollo del algoritmo fue realizar un enjambre que necesitase la mínima intervención humana para su gestión. Este objetivo ha sido alcanzado gracias a que la utilización del algoritmo solo precisa una ruta y una distancia de vuelo entre aeronaves para funcionar, sin ningún tipo de interacción más.

Las posibilidades de uso del enjambre son muchas, desde el uso para escaneo de incendios hasta la búsqueda de personas. La posibilidad de sincronizar un número de aeronaves para que realicen un vuelo sincronizado por la zona deseada aumenta todavía más la aplicabilidad de estas aeronaves.

Por último, el algoritmo está íntegramente desarrollado en JAVA, lo que facilita su portabilidad a otros computadores embebidos de reducido tamaño. Además, la máquina de estados desarrollada está ideada para evitar que las aeronaves queden en un estado inconsistente, añadiendo así seguridad a su uso.

10. Trabajos futuros

Alcanzados los objetivos propuestos, y tras las conclusiones obtenidas, han surgido distintas posibilidades de mejora o de ampliación sobre el algoritmo que se listarán a continuación:

- Ampliar la funcionalidad de vuelo para ofrecer distintas formaciones de vuelo.

- Reducir el consumo de CPU en el cálculo del despegue, reduciendo así el tiempo de despegue.

- Posibilitar el uso del enjambre con giros radiales, ya que, con la configuración actual, los giros se realizan siguiendo el mismo desplazamiento pero manteniendo la distancia establecida entre ellos.

- Analizar la nueva Raspberry Pi 3+ y su módulo WiFi 5GHz para evitar así el uso de antenas externas y facilitar su aplicación en drones que se encuentren ya en uso.

- Añadir la posibilidad de sustituir al dron Maestro en pleno vuelo; esta mejora ofrecería un nivel extra de seguridad, y posibilitaría la continuidad del enjambre incluso si el maestro deja de existir en él.

- Combinar el algoritmo desarrollado con aplicaciones ya creadas sobre computadoras Raspberry Pi. Esto incluye los proyectos desarrollados para el control de polución sobre ciudades, el escaneo de infraestructuras en búsqueda de defectos, etc.

Bibliografía

[1] Gobierno de España, «www.boe.es,» 15 Diciembre 2017. [En línea]. Available: https://www.boe.es/boe/dias/2017/12/29/pdfs/BOE-A-2017-15721.pdf.

[2] D. Systems, «COM-Based SBCs: The Superior Architecture».

[3] T. Wright, «Airspacemag,» 12 Enero 2018. [En línea]. Available: https://www.airspacemag.com/daily-planet/when-drone-swarm-not-swarm-180967820/.

[4] «Tiobe,» 28 Mayo 2018. [En línea]. Available: https://www.tiobe.com/tiobe-index/.

[5] J. K. b. F. Neitzel a, «MOBILE 3D MAPPING WITH A LOW-COST UAV SYSTEM,» 2011.

[6] L. F. N. M. S. D. S. F. Remondino, «UAV PHOTOGRAMMETRY FOR MAPPING AND 3D MODELING,» 2011.

[7] P. S. C. a. J. J. N. A. N. Chaves, «Adaptive Search Control Applied to Search and Rescue,» *IEEE LATIN AMERICA TRANSACTIONS,* p. Octubre, 2014.

[8] D. Hambling, 2016. [En línea]. Available: https://www.popularmechanics.com/military/research/a24494/chinese-drones-swarms/.

[9] Intel, «Intel Corporation,» 2018. [En línea]. Available: https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html.

[10] D. H. F. v. T. C. D. W. E. v. d. H. a. G. d. C. Bart Remes, «Paparazzi: how to make a swarm of Parrot AR Drones fly autonomously based on GPS.,» *International Micro Air Vehicle Conference and Flight Competition,* 2013.

[11] D. M. V. K. Alex Kushleyev, «Towards A Swarm of Agile Micro Quadrotors».

[12] CNN, 2011. [En línea]. Available: https://edition.cnn.com/2011/10/10/us/military-drones-virus/index.html.

[13] Ardupilot, 2016. [En línea]. Available: http://ardupilot.org/copter/docs/what-is-a-multicopter-and-how-does-it-work.html.

[14] Ardupilot, «Ardupilot.org,» 2016. [En línea]. Available: http://ardupilot.org/copter/docs/frame-type-configuration.html.

[15] «MAVLink,» 2017. [En línea]. Available: https://mavlink.io/en/.

[16] qgroundcontrol, «qgroundcontrol.org,» [En línea]. Available: http://qgroundcontrol.org/mavlink/release_11.

[17] Mavlink, «Mavlink.io,» 2018. [En línea]. Available: https://mavlink.io/en/protocol/overview.html.

[18] Ardupilot.org, «SITL,» 2018. [En línea]. Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[19] «Raspberry Pi to PIXHAWK,» [En línea]. Available: http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html.

[20] M. Industries. [En línea]. Available: http://www.mosaic-industries.com/embedded-systems/gui-user-interface/qvga-lcd-touchscreen/instrument-control/spi-rs232-serial-rs485-protocol-uart-usart.

[21] V. Salinas, «LevanteEMV,» 2 Enero 2017. [En línea]. Available: http://www.levante-emv.com/comunitat-valenciana/2017/01/02/hacienda-rastrea-drones-obras-declarar/1510742.html.