UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Solución para la gestión dinámica de vuelo de enjambres de drones híbridos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Arenillas Pozas, Julian

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

Director/a Experimental: WUBBEN, JAMIE

CURSO ACADÉMICO: 2021/2022

Resumen

En los últimos años los Vehículos Aéreos No Tripulados (VANTs) han demostrado su utilidad para optimizar y/o solucionar problemas en el ámbito aéreo de forma remota, ya sea para identificación de objetos o personas, operaciones en un perímetro determinado, rescates en catástrofes medioambientales, u otras aplicaciones. Dependiendo de la complejidad de la misión a realizar, hay que tener en cuenta la variedad y el número de VANTs a utilizar. Si se decide usar varios VANTs en vez de uno solo, estos deben de comunicarse entre sí para que el enjambre que formen sea consistente en todo momento; además, el comportamiento del mismo durante el vuelo cambiará según la variedad y el modelo de los VANTs usados en el enjambre. En este trabajo se propone desarrollar un protocolo de vuelo que se encargue de coordinar un enjambre de variedad multirrotor que permitirá seguir en tiempo real a uno de variedad ala fija, adaptando varios protocolos existentes en el simulador ArduSim, que permitirá así la inclusión de una formación híbrida. Mediante pruebas de vuelo realizadas en este mismo simulador, se ha medido el rendimiento de los VANTs en cuanto a nivel de respuesta en tiempo real, consistencia en la formación, y retardo respecto al líder.

Palabras clave: VANTs, Heterogeneidad, Enjambre, Multirrotor, Ala Fija, ArduSim

Resum

En els últims anys, els Vehicles Aeris no Tripulats (VANTs) han demostrat la seua utilitat per a optimitzar i/o solucionar problemes en l'àmbit aeri de manera remota, ja vinga a ser identificació d'objectes o persones, operacions en un perímetre determinat, rescats en catàstrofes mediambientals, i moltes més. Depenent de la complexitat de la missió a realitzar, cal tindre en compte la varietat i el número de VANTs a utilitzar. Si es decideix usar diversos VANTs en comptes d'un només, aquests han de comunicar-se entre si perquè l'eixam estiga d'acord en tot moment, i a més, el comportament canviarà respecte a la varietat i el model. En aquest treball es proposa realitzar un protocol de vol que s'encarregue de coordinar un eixam de varietat multirotor que continuaran a temps real a un de varietat ala fixa, adaptant diversos protocols existents en el simulador ArduSim, que permetran la inclusió d'una formació híbrida. Mitjançant proves de vol realitzades en aquest mateix simulador, s'ha mesurat el rendiment dels VANTs quant a nivell de resposta en temps real, consistència en la formació, i retard respecte al líder.

Paraules clau: VANTs, Heterogeneïtat, Eixam, Multirotor, Ala fixa, ArduSim

Abstract

In recent years, Unmanned Aerial Vehicles (UAVs) have demonstrated their usefulness in optimizing and/or solving airborne problems remotely, whether it is object or person identification, operations in a specific perimeter, rescues in environmental catastrophes, and many more. Depending on the complexity of the mission to be performed, the variety and number of UAVs to be used must be taken into account. If it is decided to use several UAVs instead of a single one, these must communicate with each other so that the swarm is in agreement at all times; also, the behavior will change with respect to the variety and the model. In this work we propose to develop a flight protocol that is in charge of coordinating a multirotor variety swarm that will follow one with fixed-wing variety in real time, adapting several existing protocols in the ArduSim simulator, which will allow the inclusion of a hybrid formation. Through flight tests performed in this same simulator, the performance of the UAVs has been measured in terms of real-time response level, formation consistency, and delay with respect to the leader.

Key words: UAVs, Heterogeneity, Swarm, Multirotor, Fixed-Wing, ArduSim

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Desde la Primera Guerra Mundial, los VANTs, o comúnmente llamados drones, han sido investigados para realizar funciones militares, desarrollando un recurso más barato, eficiente, y evitando el uso de un tripulante humano. A medida que pasa el tiempo, los ingenieros se van dando cuenta que pueden encaminar el uso de estos drones para tareas mas comunes dentro del ámbito aéreo, e incluso para tareas cotidianas.

Los primeros drones creados eran de variedad ala fija, con el comportamiento de una avioneta en miniatura. Ahora, los drones mas populares y modernos son de variedad multirrotor que, como bien indica su nombre, utilizan varios rotores para propulsarse. Dependiendo de el tipo de misión o tarea que se desea cumplir, uno debe pensar qué tipo de dron puede ser más adecuado para cumplirla, ya que cada uno tiene sus ventajas y desventajas respecto al otro.

Hoy día, gracias a los avances tecnológicos de la programación, podemos configurar estos drones para que realicen una tarea determinada con la ayuda de simuladores, evitando el coste que genere su despliegue y reparación en caso de avería.

Gracias al simulador ArduSim [1], perteneciente al Grupo de Redes y Computadores de la Universidad Politécnica de Valencia, podemos emular pruebas virtuales y configurar distintos protocolos de vuelo que se acomoden al objetivo del programador. Mediante una adaptación, combinación y mejora de dos protocolos ya existentes dentro de ArduSim, el objetivo principal de este TFG es crear un protocolo aéreo nuevo donde se incorpore un dron de ala fija a un enjambre de drones multirrotores, permitiendo la gestión de enjambres híbridos.

1.1 Motivación

El campo de ingeniería informática basada en drones va ligada en cierto aspecto a la ingeniería aeroespacial y, aunque en los estudios generales estas difieren sustancialmente, ambas áreas están íntimamente relacionados a la hora de desarrollar sistemas complejos.

Actualmente este campo está cada vez más en auge, y muchas empresas están adaptando cadenas de trabajo para incluir el uso de drones. También son más

frecuentemente utilizados para estudios ambientales y análisis rápido de situaciones en caso de catástrofes.

Una de mis motivaciones principales a la hora de realizar este trabajo es esta curiosidad en el campo aeroespacial. Mediante charlas y consejos de familiares que trabajan en este ámbito, he ido ganando atracción e interés a lo largo de mis estudios.

Por otra parte, mi deseo de trabajar con el *Grupo de Redes y Computadores* y su software ha sido también de importante influencia, ya que deseo realizar en el siguiente curso el *Máster Universitario en Ingeniería de Computadores y Redes*, en la *Universidad Politécnica de Valencia*.

1.2 Objetivos

El principal propósito de este proyecto es dotar al simulador ArduSim de un soporte de funcionalidades distintas al habitual, añadiendo el uso de aviones de ala fija por primera vez. Esto permitirá dar lugar a un abanico de nuevos protocolos, ideas y combinaciones mediante el uso de enjambres híbridos. Para lograr esto, se mencionarán los siguientes objetivos para alcanzar el cumplimiento del proyecto:

- Estudio y comprensión del código interno de ArduSim, y de como se ejecutan los distintos protocolos dentro del simulador.
- Creación de un nuevo protocolo, donde se utilizarán los protocolos Mision y FollowMe [2] como referencia para el comportamiento del líder y del enjambre, respectivamente.
- Implementación de las librerías ArduPlane, pertenecientes al software ArduPilot, para la simulación del dron de ala fija.
- Ejecución de distintos experimentos para la observación y análisis del proyecto.

1.3 Estructura de la memoria

Para cubrir todo el desarrollo y experiencias relacionadas con este proyecto, se ha repartido el contenido del proyecto en seis capítulos:

- En el primer capitulo se ofrece una visión general del proyecto, la motivación para su realización, los objetivos que se pretenden cumplir, y cómo esta estructurada cada parte de la memoria.
- En el segundo capítulo se procede a describir y analizar otros proyectos que tienen relación con este mismo.
- En el tercer capítulo se explicará el uso del simulador ArduSim, que es la plataforma donde se realizará todo el proyecto.

- El capítulo cuatro es donde se detalla la solución propuesta, incluyendo los problemas que se han detectado a lo largo del proyecto, y las soluciones a los mismos.
- Mediante experimentos y pruebas, el capítulo cinco tratará sobre el análisis y validación del nuevo protocolo, realizando pruebas de vuelo en del simulador.
- Por último, el capítulo seis recapitulará todo lo abarcado, describiendo las conclusiones al finalizar el proyecto, y posibles trabajos futuros.

CAPÍTULO 2
Estado del arte

En las últimas décadas se han desarrollado muchos tipos de drones, partiendo de diferentes líneas de investigación. Dependiendo del comportamiento del dron, su estudio y desarrollo puede variar bastante. Con el uso híbrido de varios tipos de drones que trabajan entre sí, el alcance de estas líneas se reduce ligeramente al estudio de modelos de ala fija, multirrotor, y su respectiva cohesión cuando están ambos en vuelo. En esta memoria nos centraremos en exponer diferentes series de proyectos, artículos y trabajos realizados por personas expertas que trabajan en este ámbito.

En [3] se ha propuesto MUSCOP, un protocolo para el vuelo coordinado de enjambres en base a misiones. Este protocolo permite al enjambre obtener un alto grado de comunicación entre los drones que lo integran. También ofrece buenos resultados en un ambiente con pérdidas de mensajes, permitiendo la comunicación entre los drones con mínimos retrasos. Utilizando el modelo de maestro-esclavo, el dron maestro realiza una misión planificada que se ve copiada a los esclavos, y en cada punto de la misión estos se sincronizan por el maestro. Esta sincronización viene dada a través de dos hilos (*Talker Thread* y *Listener Thread*), donde el maestro utiliza el primero para enviar mensajes de coordinación a los esclavos, y estos le responden con mensajes de reconocimiento. El maestro luego realizará la operación necesaria del protocolo para resincronizar los esclavos a la misión.

El protocolo FollowMe [2] permite que un enjambre vuele de manera coordinada cuando el dron líder está siendo controlado manualmente por un piloto. Este protocolo utiliza la misma tecnología de hilos mencionada en el protocolo MUSCOP anterior, pero esta vez no existe una misión previamente planificada que los drones deben seguir. El protocolo permite al maestro difundir su posición con un intervalo determinado a todos los esclavos, y estos, al recibir el mensaje de posición, cambian su rumbo para seguirlo. Con esto se logra una coordinación en tiempo real entre el enjambre sin llegar a definir un destino específico.

En la propuesta de He et al. [4] se describe un algoritmo para la planificación automática de la trayectoria de una misión de un dron en ambientes complejos. Esto se realiza a través del algoritmo HIPSO-MSOS, que logra este objetivo combinando dos algoritmos, la optimización mejorada de enjambre de partículas (*IPSO*) con la búsqueda modificada de organismos simbióticos (*MSOS*). Este algoritmo calcula la trayectoria a realizar, que luego se verá suavizada mediante

una curva cúbica *B-spline* para el vuelo real del dron. Los resultados finales son simulados en un entorno virtual 3D, donde se alcanzan mejores resultados que otros algoritmos con finalidad similar.

Por otra parte, [5] cita el uso de otro algoritmo que se centra en la recopilación de información de un enjambre de drones, ofreciendo datos en tiempo real de diferentes fuentes, como por ejemplo imágenes de vídeo. Este algoritmo es aplicable en aplicaciones tales como agricultura de precisión, respuesta ante catástrofes, servicios para ciudades inteligentes, protección civil, etc.

En [6] se propone una ley de control para una formación de dos vehículos aéreos que describen una curvatura en la trayectoria. Mediante la ecuación diferencial de geometría de curvas Frenet-Serret, se calcula la posición y dirección del dron para realizar la curva de forma unitaria a la formación. También se comenta la investigación futura necesaria para aplicar y escalar esta ley a n drones.

Con el protocolo *UAV Search Mission Protocol* (USMP) [7], se coordina la comunicación entre todos los drones de un enjambre dentro de una área geográfica; el objetivo es el de mejorar varios aspectos de una misión de búsqueda. Entre las características que este protocolo mejora cabe destacar la distancia recorrida por los drones, la disminución de cambios de sentido, y la eficacia de la misión.

En [8] se fomenta el uso de micro-drones de cuatro rotores mediante la creación de un prototipo de 75 gramos. A través de varios experimentos, se confirma las ventajas de agilidad y adaptabilidad en entornos restringidos que tiene este micro-dron, respecto a uno normal. Se explica también el software y hardware implementados, y se ofrecen resultados experimentales para una formación con 20 micro-drones del mismo prototipo.

Los drones de hoy en día utilizan muchos recursos para pilotarlos, especialmente los humanos. En [9] se crea una solución de independencia con una aplicación de recogida de datos que luego inyecta en una simulación. El *Dynamic Data-Driven Application System* (DDDAS) recoge datos de un enjambre y los respalda en una simulación en curso, que, así mismo, influye sobre los datos reales a recoger, permitiendo la variación en la cantidad y el tipo de datos que se quiere analizar a medida que la simulación sigue su rumbo.

Aquí [10] se vuelve a comentar un nuevo algoritmo derivado de la optimización de enjambres de partículas (PSO), como se ha hecho en uno de los anteriores artículos con el IPSO, pero en este caso se centra en el cálculo de las posiciones en una formación, en lugar del calculo de la trayectoria. Este algoritmo, llamado nPSO, encuentra una formación óptima en un entorno urbano con grandes obstáculos, y requiere de menos de 1000 iteraciones del algoritmo para obtener la formación óptima del enjambre en cada situación.

Con este último artículo [11], se vuelve a remarcar una derivación del algoritmo PSO, utilizando operadores mutantes de Cauchy (CM) para mejorarlo. La finalidad de esta variación es el control de la formación en un enjambre híbrido. Con varias simulaciones en terrenos tridimensionales, se obtienen resultados que mejoran la tasa de convergencia del dron de ala fija en la formación, y optimizan la solución del enjambre, demostrando la efectividad de la formación híbrida.

Como se ha planteado en estos artículos, se han realizado una gran variedad de trabajos con el objetivo principal de solucionar diferentes situaciones, tanto en

el ámbito de la comunicación entre drones, como en la convivencia de diferentes tipos de drones en un mismo enjambre, permitiendo el uso de misiones de vuelo con funcionalidad híbrida. La finalidad del trabajo a expresar en esta memoria viene influenciada por estas diferentes soluciones, cultivando la creación de un protocolo que permita una comunicación y cohesión entre un dron de ala fija y varios multirrotores en una misma misión.

CAPÍTULO 3
Simulador ArduSim

ArduSim [1] es un simulador de vuelo en tiempo real que permite el desarrollo y coordinación de protocolos de vuelo basados en drones de modelo multirrotor. Según las prestaciones, la cantidad de drones a simular varía, permitiendo simulaciones a gran escala con un sistema con buenos requisitos. Su software es de código abierto [12], e incluye instrucciones para su instalación, uso, desarrollo de protocolos nuevos y lanzamiento en dispositivos reales.

Para el desarrollo de un nuevo protocolo de vuelo, hay que tener en cuenta que este protocolo no puede estar directamente enlazado a otros protocolos, es decir, cada protocolo debe ser independiente. El funcionamiento interno de ArduSim tampoco debe depender del nuevo protocolo; de esta forma se asegura de no alterar el código de ArduSim, permitiendo adaptabilidad a versiones nuevas.

Uno de los objetivos de este proyecto es desarrollar un nuevo protocolo y, para esto, adaptaremos un par de protocolos ya existentes en el simulador:

- **Mision**: De los primeros protocolos del simulador. El objetivo de este protocolo es hacer que un enjambre de drones realicen una misión anteriormente planeada. Estas misiones están estructuradas por puntos, donde cada dron deberá seguirlos en el orden establecido hasta finalizar dicha misión.

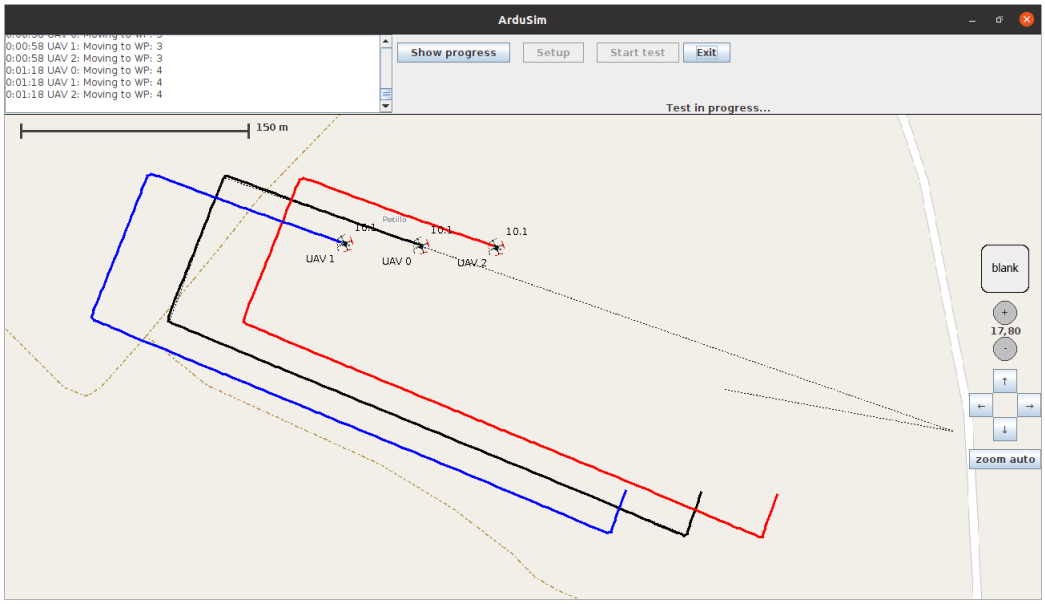

Figura 3.1: Protocolo Mission en ejecución.

Una vez los drones son configurados y preparados para el despegue por el simulador, cada uno inicia un hilo individual para despegar el dron hasta alcanzar la altitud deseada. Una vez estén todos en el aire, se pone en marcha el protocolo, ejecutando un bucle que volverá a crear un hilo individual a cada dron por cada punto en la misión, indicando la dirección hacia donde deben moverse. Una vez el dron llegue al primer punto, el hilo anterior finalizará, y se vuelve a crear otro con las coordenadas del siguiente punto, repitiendo este proceso hasta llegar al último punto de la misión. Una vez dada por terminada la misión, estos drones procederán a aterrizar en el suelo, acabando el protocolo.

La configuración de este protocolo es muy sencilla. Como se puede observar en la imagen, cargando la misión a realizar, la formación que desarrollarán los drones en el vuelo, y la distancia mínima que deben respetar entre ellos, este protocolo puede ejecutarse sin ningún problema.

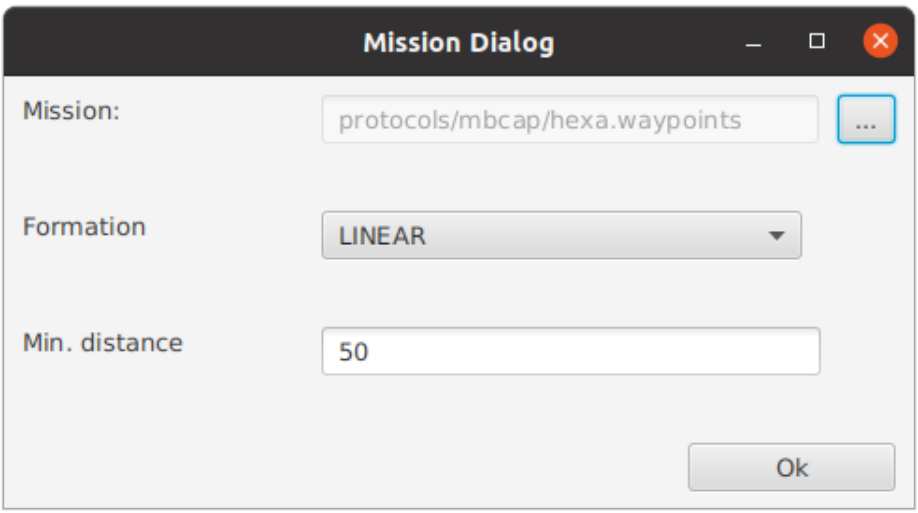

Figura 3.2: Configuración en el simulador del protocolo Mission.

En nuestro propio protocolo a desarrollar, adaptaremos la función de los hilos para desplazarse a través de una misión, pero solo la aplicaremos para el dron líder. El resto de drones del enjambre deberán seguirlo partiendo de los principios del protocolo FollowMe, que explicaremos a continuación.

- **FollowMe**: Este protocolo fuerza a un enjambre a seguir, en tiempo real, a un dron con la función de líder, donde este es pilotado manualmente por un usuario real.

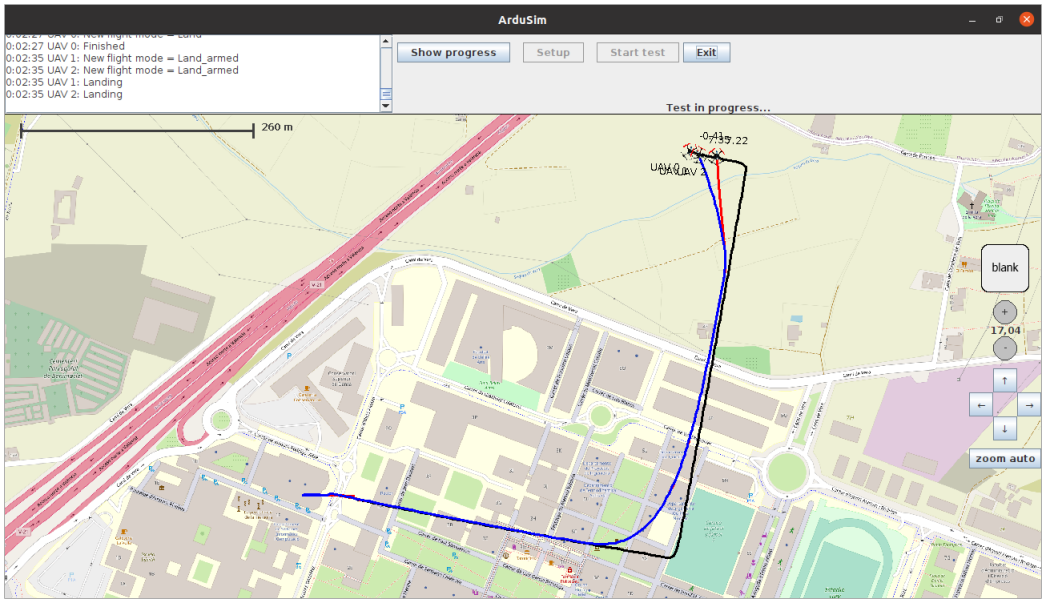

Figura 3.3: Protocolo FollowMe en ejecución

Como bien se ha comentado en el capitulo anterior, el funcionamiento de este protocolo se basa en el uso de dos hilos importantes, el *Listener Thread* y el *Talker Thread*. Una vez el simulador realiza la configuración de los drones, se inicializa un *Listener Thread* por cada dron, incluso el maestro. Este hilo se encargará de monitorizar en qué estado se encuentra cada dron, y realizar las funciones necesarias en cada caso.

Una vez se ejecute el experimento con el protocolo, el hilo realizará el despegue del enjambre en formación y, cuando todos los drones lleguen a la altura ideal para comenzar el seguimiento, se ejecutará el *Talker Thread*, el cual estará enlazado con el dron maestro. Este hilo se encargará de enviar constantemente un mensaje broadcast de posición, que reconocerán los drones esclavos para moverse hacia la dirección que han recibido. La comunicación y movimiento constante por parte de todo el enjambre simula un efecto de seguimiento de los esclavos al líder, mientras este último es pilotado manualmente por un piloto.

Una vez este desee aterrizar el dron maestro, el *Talker Thread* enviará un mensaje de aterrizaje con la posición donde se va a efectuar; los esclavos recibirán dicho mensaje, y entonces procederán también ellos a aterrizar en el mismo lugar.

La configuración del protocolo FollowMe es más detallada que la de Mission, permitiendo configurar como están distribuidos los drones en el suelo, qué for-

mación va a tener el enjambre durante el vuelo, la distancia mínima entre elementos del enjambre en estas formaciones y en su aterrizaje, y el intervalo de tiempo utilizado para refrescar la posición del líder, la cual se envía a los esclavos por difusión.

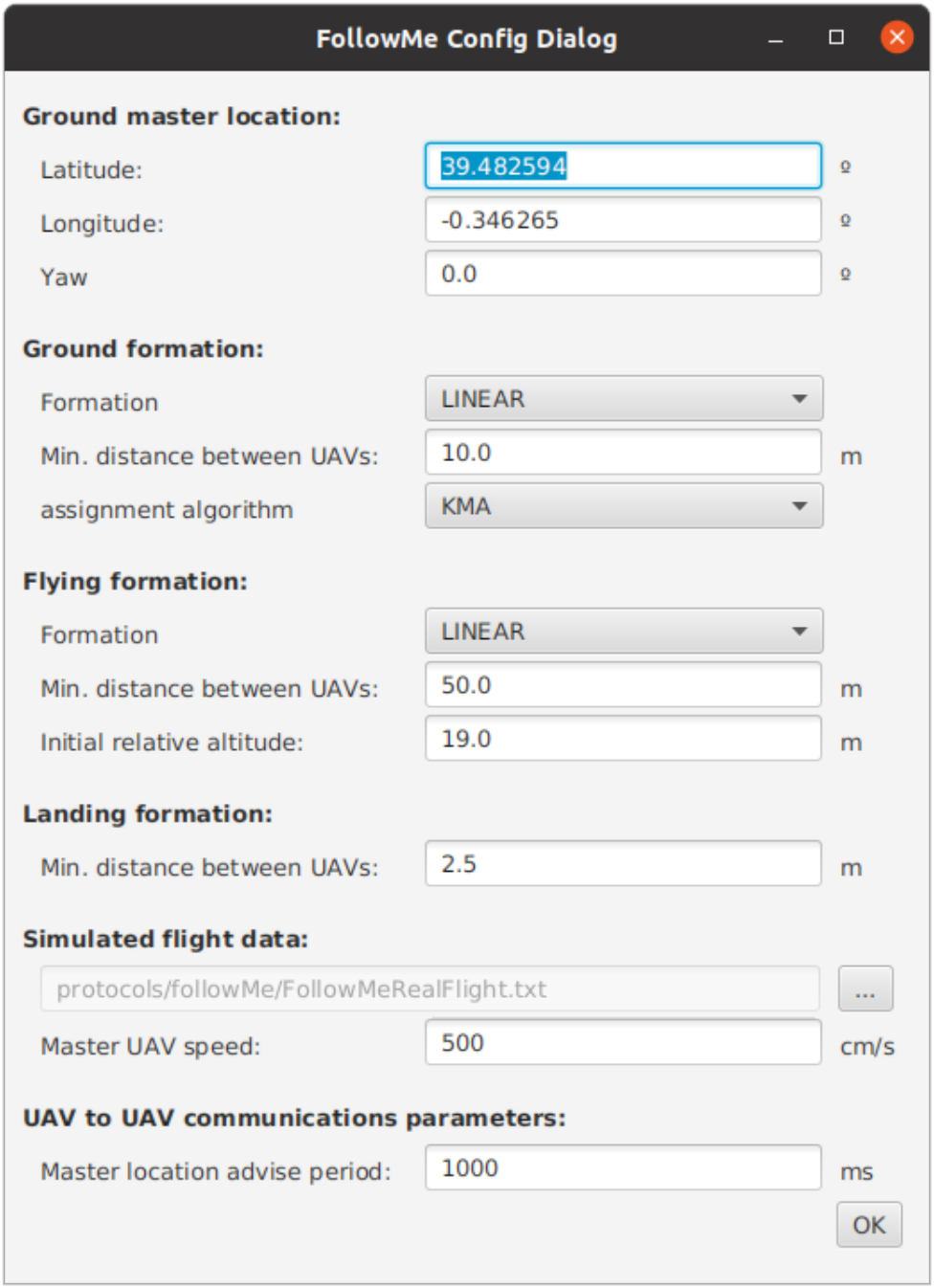

Figura 3.4: Configuración en el simulador del protocolo FollowMe.

El uso de varios hilos para la comunicación va a servirnos para el desarrollo de nuestro protocolo, utilizando el mismo principio de envío broadcast de posición para que el resto de drones del enjambre sigan al líder.

Actualmente, en el *Grupo de Redes y Computadores*, existe una mejor versión del protocolo FollowMe [13] donde se implementa un calculo de predicción de la

posición del líder una vez este ha enviado el broadcast con su posición al resto del enjambre. Esta predicción reduce el retraso del resto de drones respecto al líder, ya que en la versión original esta posición viene condicionada por el intervalo de su envío.

En nuestro protocolo también hemos decidido implementar esta nueva versión, mejorando ligeramente el desfase de la formación del enjambre respecto al líder.

CAPÍTULO 4
Diseño y desarrollo del problema

A continuación, dedicaré este capítulo a detallar el análisis, diseño e implementación de este protocolo que se pretende crear en este proyecto. También se hablará de los requisitos necesarios para abordar el problema, las librerías necesarias para el avión de ala fija y las diferentes limitaciones que han surgido a lo largo del desarrollo del proyecto.

4.1 Análisis del problema

El propósito general de este proyecto es la creación de un protocolo nuevo dentro del simulador ArduSim, que permitirá la cohesión entre un dron de ala fija y un enjambre de drones multirrotores que lo seguirán a tiempo real con una formación previamente elegida.

A partir de este propósito, podemos subdividir el plan de desarrollo en tres partes: el estudio de los diferentes programas utilizados en el proyecto, la implementación de las librerías ArduPlane para el avión de ala fija, y la creación del protocolo dentro del simulador ArduSim para su ejecución.

- En primer lugar, el lenguaje de programación que se ha utilizado para el proyecto es Java [14] (Java SE 17) junto al entorno de desarrollo integrado de JetBrains, IntelliJ [15]. Esto se debe a la necesidad de trabajar con el simulador ArduSim, desarrollado con el lenguaje Java dentro de este IDE mencionado. También se ha trabajado con el sistema operativo Ubuntu [16] (Ubuntu 20.04.4 LTS), ya que es requisito recomendado para el rendimiento de ArduSim. Por otra parte, diferentes programas como MAVLink [17], Wireshark [18] y MissionPlanner [19] han sido utilizados para el estudio de la comunicación de mensajes del dron de ala fija, donde este último también se ha utilizado para simular el comportamiento del dron.
- En segundo lugar, para poder emular el comportamiento de un dron de tipo ala fija, es necesaria la siguiente librería llamada ArduPlane. Esta librería proviene del firmware ArduPilot [20], que desarrolla diferentes librerías por

cada tipo de dron, ofreciendo versiones actualizadas periódicamente con un amplio foro donde los usuarios pueden comentar sus problemas y ayudarse entre ellos. En el apartado 4.3 se explicará con más detalle las necesidades de ArduPlane y sus diferentes configuraciones.

- Por último, siguiendo los pasos de como crear un protocolo en el github de ArduSim [12], hemos podido observar como se debe estructur ar un protocolo, las diferentes clases y funciones necesarias para su ejecución y como dividirlas para lograr una independencia entre otros protocolos. También se explica la funcionalidad de los dos protocolos que adaptaremos dentro del nuestro, Mission y FollowMe, que nos permitirá un mejor entendimiento a la hora de replicarlos en el código interno, ya que no podemos referirlos porque esto crearía una dependencia del protocolo con estos dos, violando una de las premisas de la estructuración de ArduSim.

Logrando el estudio, implementación y desarrollo de los diferentes programas, librerías y protocolos de vuelo, este proyecto puede desarrollarse y alcanzar los objetivos propuestos. En el siguiente apartado, explicaremos los distintos requisitos necesarios para poder trabajar con esta solución planteada.

4.2 Requisitos

Como todo este proyecto se implementa en el uso de un simulador, los únicos requisitos necesarios para poder llevarlo a cabo son técnicos, relacionados con los componentes de un ordenador para que soporte el uso del simulador.

Estos requisitos están especificados en el github de ArduSim, concretamente en requisitos mínimos y recomendados:

Requisitos mínimos:

- Intel core i5 (versión de 4 núcleos)
- 6 GB de RAM
- Sistema Operativo Windows, Linux o MacOS
- Java SE 11 de 32 bits
- Cygwin y ImDisk Virtual Disk Driver (solo para Windows)

Requisitos recomendados:

- Intel core i7 (4 núcleos con Hyper-Threading)
- 16 GB de RAM
- Linux (Ubuntu 16.04 o 18.04)
- Java SE 11 de 64 bits
- Cygwin y ImDisk Virtual Disk Driver (solo para Windows)

Concretamente para nuestro caso, se ha utilizado un portátil HP con Intel core i5 de séptima generación de cuatro núcleos, 8 GB de RAM, sistema operativo Linux (Ubuntu 20.04.4 LTS de 64 bits) y Java SE 17.

Se llega a alcanzar los requisitos mínimos, pero al no tener un procesador y memoria RAM recomendados, se nota un ligero error de rendimiento a la hora de ejecutar las pruebas de vuelo. Esto no afecta en los resultados, solo a la hora de trabajar con el programa, cosa que no es un impedimento a la hora de desarrollar el proyecto.

4.3 Librería ArduPlane

Para poder emular el uso de un avión de ala fija dentro del simulador de vuelo, existe una librería firmware llamada ArduPlane [21]. En la página oficial de ArduPlane se tratan diferentes temas para su uso, desde instalación en un dron real, la configuración de parámetros del avión y su inclusión en un simulador. ArduPlane es de código libre, permitiendo su uso para aplicaciones, estudios y/o investigación.

ArduPlane ofrece distintos tipos de controladores para cada tipo de avión de ala fija, ya sea aviones propulsados con ruedas incorporadas para el despegue en una pista, aviones que despegan lanzándolos con la mano, aviones con rotores incorporados, y muchos más. En nuestro proyecto nos interesa el primer tipo mencionado.

Incorporados con modos de vuelo especiales para su uso, *Automatic Takeoff* es uno de los modos más importantes, permitiendo al avión su despegue automático con un par de configuraciones previas. Insertando los grados de inclinación y la altura deseada a este modo, el avión pone al máximo su acelerador y empieza ascender hasta llegar a la altura establecida.

Dependiendo de su modo de despegue, también hay que cambiar diferentes parámetros del avión. En nuestro caso nos interesa tocar parámetros relacionados con la sección *Runaway Takeoff*, más conocido como *Conventional Takeoff and Landing*(CTOL). Los parámetros de esta sección ajustan el comportamiento de los componentes *tail dragger* (dirección de la rueda trasera) y *trycicle undercarriage* (dirección de la rueda delantera):

- **TKOFF_TDRAG_ELEV** sirve para mantener la rueda trasera con fuerza en la etapa principal del despegue, dándole suficiente agarre a la rueda para dirigir el avión en la pista.
- **TKOFF_TDRAG_SPD1** aplicará la fuerza del parámetro anterior hasta que se alcanza la velocidad establecida en este parámetro. Con esto se logra iniciar una inclinación necesaria para el despegue con una velocidad ganada previamente.
- **TKOFF_THR_SLEW** permite que el acelerador de vuelo aumente a una velocidad apropiada para el despegue del avión.

- **TKOFF_ROTATE_SPD** controla cuando el piloto automático deberá levantar el morro del avión para levantarse del suelo. Cuanto más alto sea este valor, el avión realizara un despegue más lento y recorrerá más la pista.
- **TECS_PITCH_MAX** asigna el ángulo máximo que puede tener el avión mientras realiza el takeoff.
- Por último, el parámetro **GROUND_STEER_ALT** viene relacionado con el ajuste automático de la dirección del avión en tierra, que hay que realizar antes del despegue para asegurar que el controlador de dirección pueda dirigir la aeronave de forma fiable. Este parámetro indica a que altitud se deberá empezar a utilizar este controlador.

El resto de modos de vuelo que nos interesa son el modo *Guided* y el *Automatic Landing*, siendo el primero el modo base que vamos a utilizar mientras indicamos al avión que realice la misión de vuelo, y el último para finalizar el protocolo con un aterrizaje.

4.4 Limitaciones de simulador

Después de varios meses de inserción de la librería ArduPlane anteriormente mencionada, no se ha podido lograr su funcionamiento dentro del simulador ArduSim.

Esto se debe principalmente al determinismo del simulador a la hora de trabajar con diferentes tipos de drones. Cuando se desarrolló ArduSim, no se contempló el uso de diferentes drones salvo los multirrotores, utilizando solo la librería ArduCopter, con similar uso de la librería ArduPlane pero para drones multirrotores, perteneciente al firmware ArduPilot. Este hecho se conocía antes del desarrollo del proyecto y invertimos alrededor de tres meses para permitir la incorporación de la librería nueva dentro del simulador, pero no obtuvimos resultado.

Modificando el código del denso simulador, nos dimos cuenta que los mensajes del *Ground Control Station*(GCS) implementado dentro enviaba peticiones y respuestas con diferente estructura y contenido a las que necesitaba el avión de ala fija. No obstante, su modificación para el uso de la aeronave no solo destruía la propia ejecución del simulador, sino que también inhabilitaba el uso de drones multirrotores, cosa que no era de buena idea implementar si queríamos tener un enjambre híbrido.

El protocolo se ha desarrollado sin el uso de esta librería pero teniendo en cuenta la futura inserción de esta, adaptando lo máximo posible el protocolo a el comportamiento de un avión.

Actualmente mi cotutor Jamie Wubben está trabajando con el *Grupo de Redes y Computadores* para lograr un rediseño del simulador que permita la inclusión de librerías distintas en este mismo y mejorar aspectos generales del simulador, donde se podrá lograr como trabajo futuro esta inserción dentro del protocolo.

4.5 Desarrollo de la solución

En este último subapartado del capítulo se explicará como se ha desarrollado el nuevo protocolo al que llamaremos *PlaneFollower*, siguiendo los pasos para crear un protocolo dentro del simulador ArduSim y detallando cada aspecto de este.

Cada protocolo esta organizado dentro del simulador en tres carpetas: *gui*, *logic* y *pojo*.

- En *gui* se almacenaran las clases que pertenecen a la visualización gráfica de la configuración del protocolo dentro del simulador y inicializaran la configuración del protocolo que el usuario ha establecido.
- Dentro de *logic* residirán las clases que lograrán todos los cálculos y ejecución del protocolo, es decir, toda la lógica detrás del protocolo.
- *Pojo* sirve para contener clases con funcionalidades varias que sirven para apoyar a las clases de *logic*.

La estructuración de clases de nuestro protocolo es la siguiente:

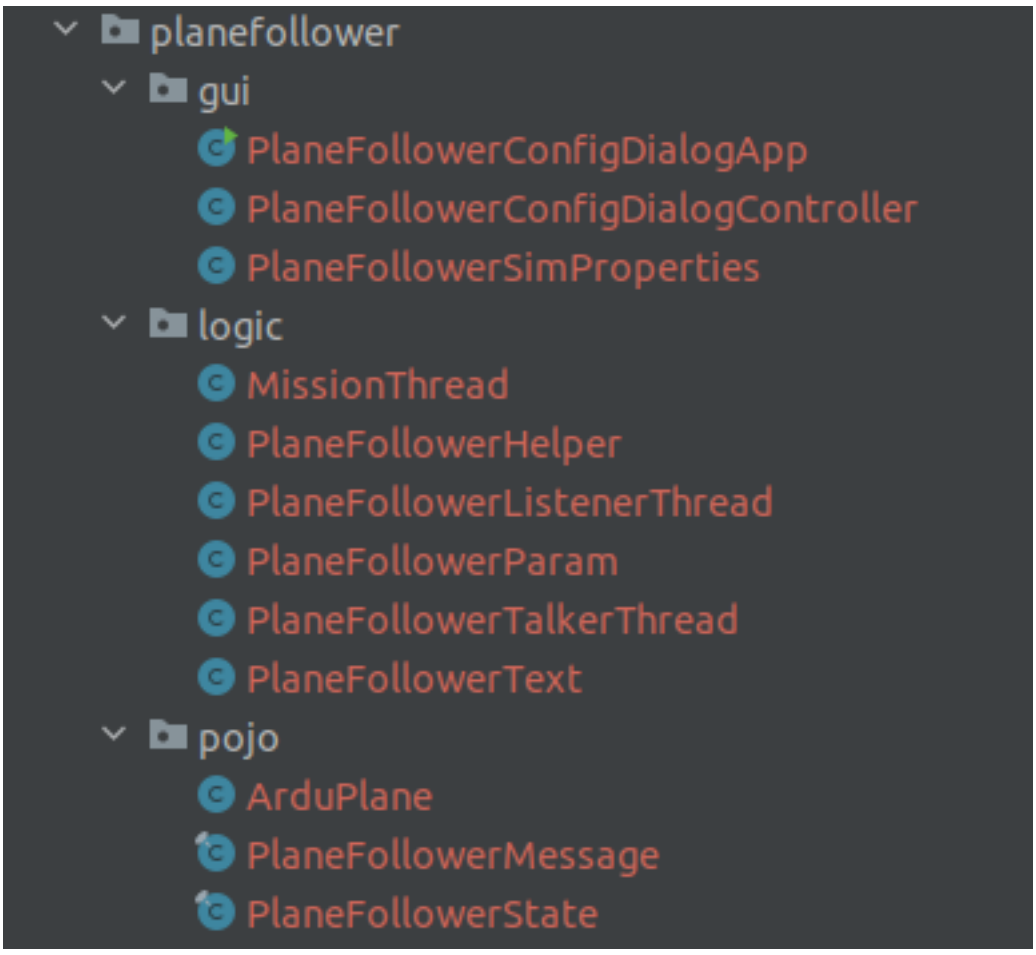

Figura 4.1: Distribución de las clases de PlaneFollower dentro de ArduSim

Empezando por la carpeta *gui*, las siguientes 3 clases muestran la siguiente ventana de configuración del protocolo:

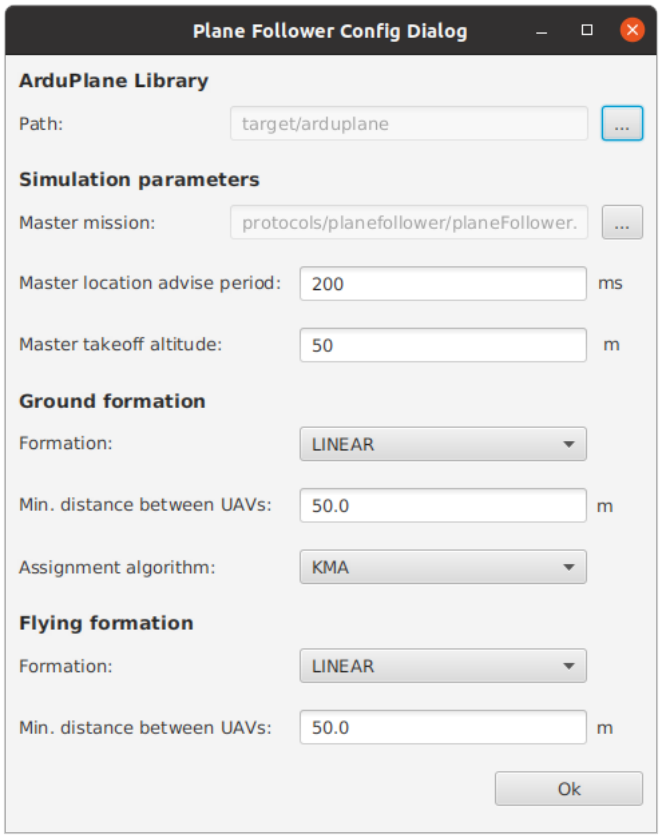

Figura 4.2: Configuración del protocolo PlaneFollower

- **PlaneFollowerConfigDialogApp:** Inicializa la escena dentro del simulador, mostrando la ventana de configuración del protocolo cargando previamente las dos clases siguientes.
- **PlaneFollowerConfigDialogController:** Carga todos los elementos creados en un archivo .fxml y enlaza los eventos adjuntos a cada uno para que la escena de la configuración actúe correctamente como una ventana.
- **PlaneFollowerSimProperties:** Almacena todos los valores que se han elegido en la configuración y los asigna para su posterior uso en la ejecución del simulador.

Pasamos ahora a la carpeta *logic* donde reside toda la programación detrás del protocolo PlaneFollower, explicando cada clase interna y mostrando código importante de cada una:

- **MissionThread:** Esta clase se encarga de crear los hilos que realizará el dron líder para seguir la misión previamente establecida en la configuración.

```
@Override
public void run(){
    PlaneFollowerTalkerThread.protocolStarted = true;

    MissionHelper missionHelper = API.getCopter(numUAV).
        getMissionHelper();
    List<Waypoint>[] missions = missionHelper.getMissionsLoaded();
    List<Waypoint> mission = missions[0];
    Formation f = UAVParam.airFormation.get();
    for(int wp_index = 2; wp_index<mission.size(); wp_index++) {
        Waypoint wp = mission.get(wp_index);
        API.getGUI(numUAV).logUAV("Moving to WP: " + (wp_index - 1));

        try {
            Location2DUTM locInSwarm = f.get3DUTMLocation(new
                Location3DUTM(wp.getUTM(), 0), numUAV);

            Location3D loc = new Location3D(locInSwarm.getGeo(),
                altitude);
            Thread t = API.getCopter(numUAV).moveTo(loc, new
                MoveToListener() {
                @Override
                public void onCompleteActionPerformed() {
                }

                @Override
                public void onFailure() {
                }
            });
            t.start();
            t.join();
        } catch (LocationNotReadyException | InterruptedException e) {
            e.printStackTrace();
        }
    }
    currentState.set(LANDING);
}
```

Figura 4.3: Función *run()* en MissionThread

Ejecutando la función *run()* de esta clase, se recoge la misión de la configuración y se crea un hilo por cada waypoint, que hará que el dron se mueva a la posición del punto en el mapa. Una vez logre llegar a este punto, el hilo se descartará y se creara otro para el siguiente, así hasta desplazarse por la misión entera y poniendo al dron en modo de aterrizaje para que el resto del enjambre lo sepa.

- **PlaneFollowerHelper:** Esta es la clase que todo protocolo debe tener dentro del simulador, siendo esta una clase que extiende de ProtocolHelper. Aqui se establecen que se debe realizar en cada paso de la ejecución del protocolo, siendo uno de estos pasos importantes la creación de hilos que permitiran a los drones comunicarse entre sí.

```
@Override
public void startThreads(){
    int numUAVs = API.getArduSim().getNumUAVs();
    master = new PlaneFollowerListenerThread(0);
    master.start();
    for(int i = 1; i < numUAVs; i++){
        new PlaneFollowerListenerThread(i).start();
    }
    API.getGUI(0).log(PlaneFollowerText.ENABLING);
}
```

Figura 4.4: Función *startThreads()* en PlaneFollowerHelper

- **PlaneFollowerListenerThread:** Esta es una de las clases importantes a la hora del funcionamiento del protocolo, junto a **PlaneFollowerTalkerThread**. Permite el recibimiento de mensajes y su contestación por parte del líder y del enjambre. Su principal hilo de ejecución es el siguiente:

```
@Override
public void run(){
    while(!arduSim.isSetupInProgress()){ arduSim.sleep(
        CompareTakeOffSimProperties.timeout);}
    Swarm swarm = setup();
    setupDone = true;
    while(!arduSim.isExperimentInProgress()){arduSim.sleep(
        CompareTakeOffSimProperties.timeout);}
    takeoff(swarm);
    takeoffDone = true;
    waitUntilSwarmTakeOff();
    follow();
    land();
    finish();
}
```

Figura 4.5: Función *run()* en PlaneFollowerListenerThread

Mientras el simulador aun esté en fase de preparación de los drones, el hilo mantendrá en espera. Una vez se empiece el experimento del protocolo, el enjambre realizará un despegue simultáneo en formación, donde la función takeoff(swarm) se encargará de esto. Después, se verificará con los estados de los drones que todos han realizado el despegue, y cuando esto ocurra, se procederá a realizar la función de follow():

```
private void follow(){
    currentState.set(FOLLOWING);
    gui.logUAV(PlaneFollowerText.FOLLOWING);
    gui.updateProtocolState(PlaneFollowerText.FOLLOWING);

    currentTime = System.currentTimeMillis();
    prevTime = currentTime;
    if(isMaster){
        beginMissionAndTalkerThread();
    } else{
        gui.logVerboseUAV(PlaneFollowerText.SLAVE_WAIT_ORDER_LISTENER);
        while(currentState.get() == FOLLOWING){
            processMessages();
        }
    }
}
```

Figura 4.6: Función *follow()* en PlaneFollowerListenerThread

La función follow() pone a todos los drones en el estado FOLLOWING y inicializará dos hilos adicionales para el líder, el **MissionThread** previamente explicado y el **PlaneFollowerTalkerThread** que se explicará posteriormente, permitiendo el envío de la posición del líder mientras el resto del enjambre procesa estas posiciones con la función processMessages().

La función processMessages() es una función simple que se encarga de redirigir la operación dependiendo de si el tipo de mensaje recibido por el líder es de posición o de aterrizaje. En case de recibir un mensaje de posición se pasará a realizar la siguiente función:

```
1  private void processIamHereMessage(){
2
3      // Master
4      Location2DUTM masterLocation = new Location2DUTM(input.readDouble()
           , input.readDouble());
5      double relAltitude = input.readDouble();
6      double masterSpeed = input.readDouble();
7      double masterTrack = input.readDouble();
8
9      // Prediction
10     double predictedT = 0.2;
11     double predictedX = masterLocation.x + masterSpeed * Math.sin(
           masterTrack) * predictedT;
12     double predictedY = masterLocation.y + masterSpeed * Math.cos(
           masterTrack) * predictedT;
13     Location2DUTM predictedPos = new Location2DUTM(predictedX,
           predictedY);
14     Location3DUTM predictedPos3D = new Location3DUTM (predictedPos,
           relAltitude);
15
16     try{
17         Location2DUTM targetLocationUTM = UAVParam.airFormation.get().
               get3DUTMLocation(predictedPos3D, numUAV);
18         Location3DGeo targetLocation = new Location3DUTM(
               targetLocationUTM, relAltitude).getGeo3D();
19         copter.moveTo(targetLocation);
20
21     }catch(LocationNotReadyException e){
22         gui.log(e.getMessage());
23         e.printStackTrace();
24         // Fatal error. It lands
25         currentState.set(LANDING);
26     }
27 }
```

Figura 4.7: Función *processIamHereMessages()* en PlaneFollowerListenerThread

Con processIamHereMessage(), el enjambre leerá las diferentes coordenadas, velocidad y la dirección en la que se mueve por la misión, permitiendo un cálculo de predicción de posición del líder que logrará una mejora en el retraso del seguimiento por parte del enjambre.

- **PlaneFollowerParam:** Aquí residen diferentes parámetros con valores establecidos por la configuración anterior del protocolo, donde diversas clases las usan para evitar código duplicado.
- **PlaneFollowerTalkerThread:** Este hilo tiene una estructura similar a la del **PlaneFollowerListenerThread**, pero con el cambio de que en vez de recibir mensajes es este el que los envía. Vinculado al líder, este enviará mensajes de posición mientras se esta realizando el desplazamiento a través de la posición. También se realiza un calculo tangente con su posición anterior para obtener la dirección en la que se mueve.

```
private void sendPosition(){
    Location2DUTM here = copter.getLocationUTM();

    cicleTime = System.currentTimeMillis();

    // Track
    prevLocation = here;
    here = copter.getLocationUTM();

    if(here.x != prevLocation.x && here.y != prevLocation.y){
        track = Math.PI / 2 - Math.atan2(here.y - prevLocation.y, here.
           x - prevLocation.x);
        if(track < 0){
            track = track + 2*Math.PI;
        }
        else{
            if(track > 2*Math.PI){
                track = track - 2*Math.PI;
            }
        }
    }

    z = copter.getAltitudeRelative();
    speed = copter.getPlannedSpeed();

    output.reset();
    output.writeShort(PlaneFollowerMessage.I_AM_HERE);
    output.writeDouble(here.x);
    output.writeDouble(here.y);
    output.writeDouble(z);
    output.writeDouble(speed);
    output.writeDouble(track);
    output.flush();

    byte[] message = Arrays.copyOf(outBuffer, output.position());
    link.sendBroadcastMessage(message);
}
```

Figura 4.8: Función *sendPosition()* en PlaneFollowerTalkerThread

Una vez realizado este cálculo, procederá a enviar su posición junto a su velocidad y dirección a través de un mensaje broadcast, que recogerán el enjambre para su posterior recolección y seguimiento al líder. Este envío de posición se realiza periódicamente mientras sigue su rumbo con la misión, con un intervalo modificable de valor principal de 200 milisegundos.

- **PlaneFollowerText:** Por último en la carpeta logic, esta clase contiene una lista de strings que indican diferentes estados y errores que se muestran en la interfaz gráfica del simulador, permitiendo al usuario tener un seguimiento del protocolo a tiempo real.

Con la última carpeta, pojo, mostramos todas las clases que toman acción en la ejecución de este protocolo:

- **ArduPlane:** Esta clase se encarga de cargar la librería previamente adjuntada en la configuración.

- **PlaneFollowerMessage:** Aquí residen los distintos tipos de mensajes que puede enviar el líder, siendo estos de posición o de aterrizaje. Esta clase solo sirve para adjuntar una identificación al tipo de mensaje, y que esta sea percibida por el enjambre.
- **PlaneFollowerState:** Similar a la clase anterior,en esta clase se definen los diferentes estados que puede tener un dron, permitiendo un orden en la ejecución de la lógica.

Con el conjunto de todas estas clases programadas dentro del simulador ArduSim, podemos ejecutar el protocolo, permitiendo diferentes experimentos y comprobando el comportamiento de los drones:

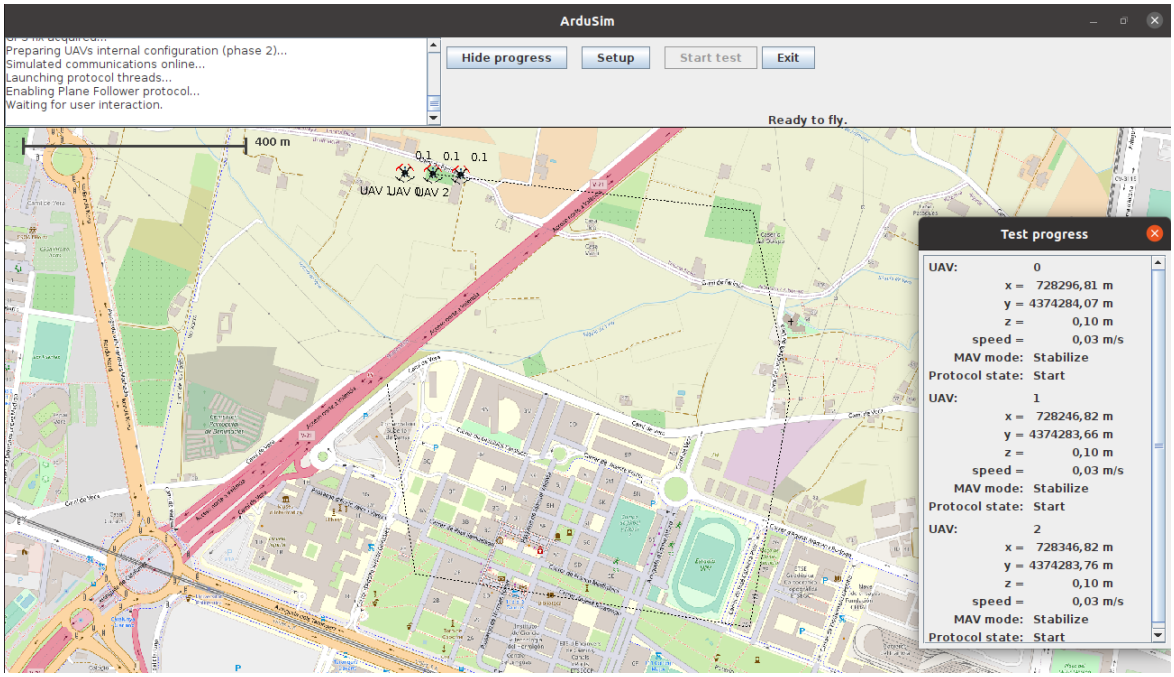

Figura 4.9: PlaneFollower en ejecución - Principio de la misión

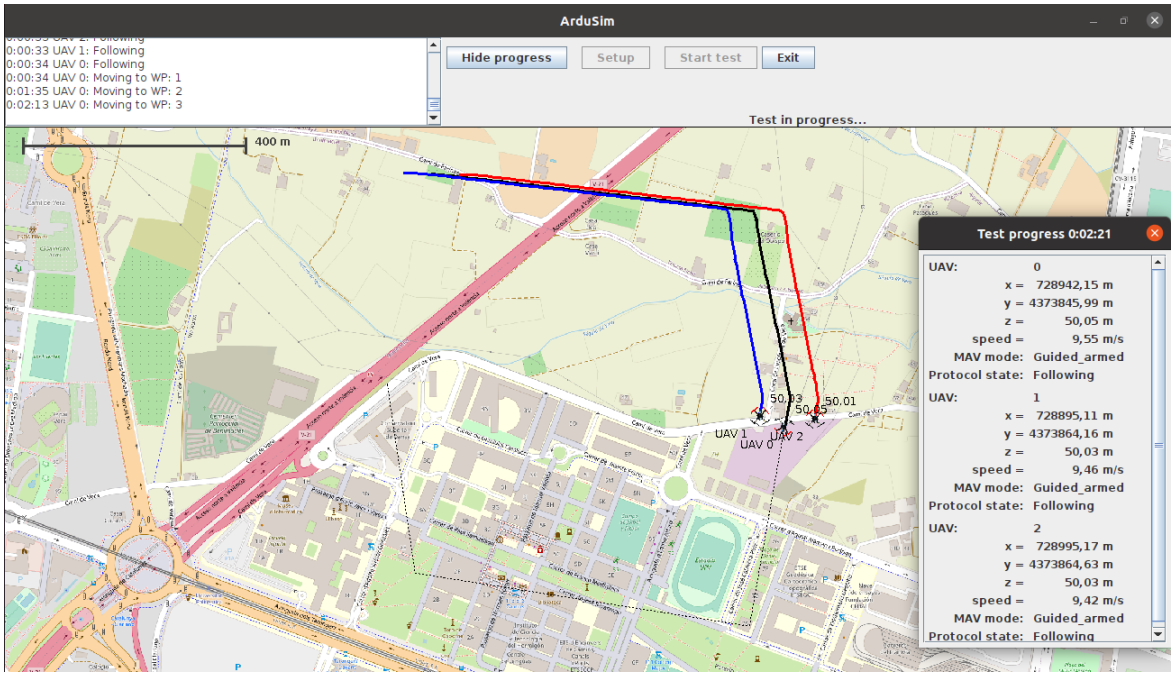

Figura 4.10: PlaneFollower en ejecución - Mitad de la misión

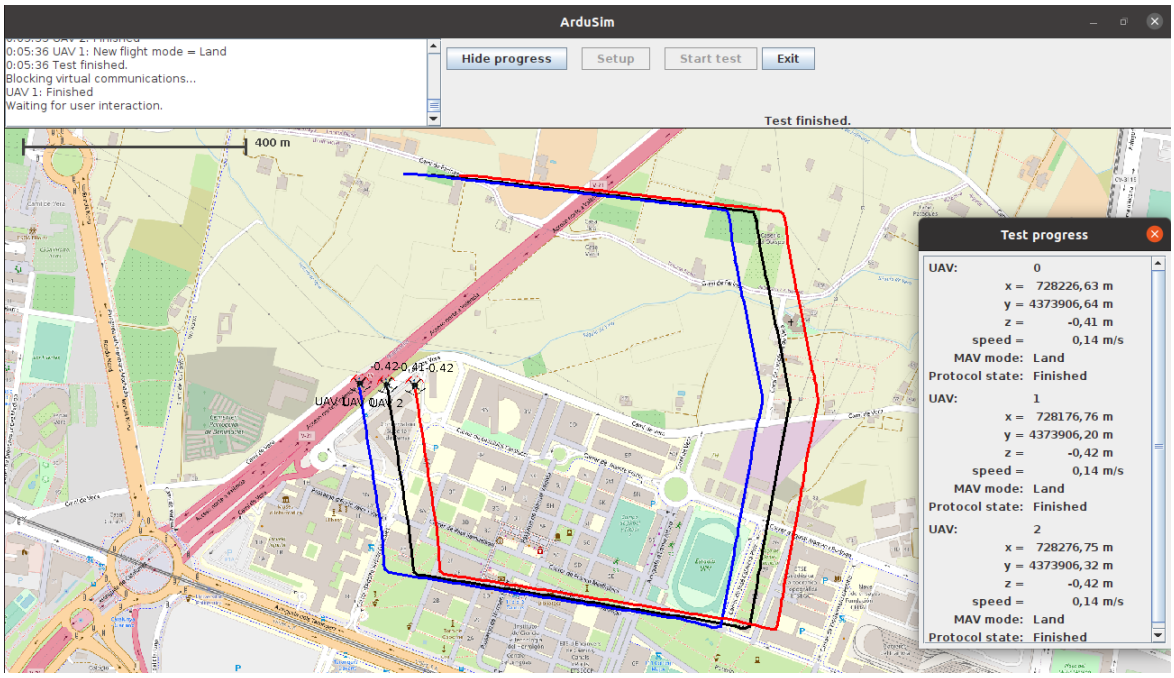

Figura 4.11: PlaneFollower en ejecución - Final de la misión

Como bien se muestran en las figuras, el protocolo se divide en tres partes diferentes: despegue, seguimiento y aterrizaje. De esta forma podemos realizar una resumen general del comportamiento después de detallar la funcionalidad de cada clase.

Mientras se realiza la configuración previa de cada dron, se crea un hilo por cada dron presente en el experimento. Este hilo permitirá al enjambre recibir mensajes del líder, y también permitirá al líder crear el hilo que envíe estos mensajes cuando se llegue al punto ideal del protocolo donde se necesiten.

Cuando se empieza la ejecución en el simulador, todo el enjambre empieza a despegar con la formación establecida en la configuración. Una vez todos los drones han alcanzado la altitud deseada, el líder procederá a realizar la misión a través de un hilo.

Mientras este realiza la misión, el líder irá enviando periódicamente su posición 3D junto a su velocidad y dirección mediante un envío broadcast. El enjambre, por otra parte, estará escuchando estos mensajes y se desplazará a la posición que se le ha enviado, simulando un seguimiento hacia el líder.

Una vez el líder haya llegado hasta el punto máximo de la misión, enviará un último mensaje broadcast para indicar que va a proceder su aterrizaje. El enjambre escuchará la petición y procederá a aterrizar en la misma posición y con la misma formación.

Por último, cuando todos los drones cambien su estado a la de aterrizaje, el protocolo pasará a verificar que efectivamente todos los drones han aterrizado, y cuando esto pasé, el protocolo terminará con éxito.

CAPÍTULO 5
Experimentos y pruebas finales

A continuación vamos a recopilar todos los datos realizados a través de experimentos sobre el simulador utilizando este nuevo protocolo anteriormente descrito. Con una serie de modificaciones en los experimentos, estos datos se han obtenido mediante diferentes pruebas de vuelo calculando el desfase en la distancia del enjambre sobre el líder, y variando la velocidad del líder y de la formación de todos los drones.

Como objetivo general, todos los drones del enjambre deberían de tener constantemente una separación de 50 metros con el líder, pero esto solo ocurre en un caso ideal, demostrando con la experimentación qué casos específicos favorece o desfavorece este desfase dentro del protocolo.

Se han realizado tres experimentos con diferentes velocidades del líder para las formaciones lineal, circular y matricial, con un total de 9 experimentos. Estas pruebas se realizan con 5 drones en el enjambre, excepto en la formación lineal, en la cual se usan 3. Esto se debe a que la composición de más de 3 drones en esta formación aumenta la distancia para los drones adicionales al estar más alejados del líder, lo cual afectaría los resultados de las pruebas al experimentar esos drones más pérdida en el canal.

En las gráficas resultantes se ha omitido el cálculo del desfase en la etapa inicial de despegue, empezando a recaudar datos a partir del inicio de la misión. Sin embargo, se ha decidido incluir los datos del aterrizaje para verificar que se cumple el desfase ideal al menos en esta fase.

Todas estas gráficas que mostraremos a continuación son resultado de recolectar, cada 5 segundos, la distancia de separación entre cada dron en el enjambre mientras se realiza la misma misión previamente mostrada en la explicación del protocolo del capitulo anterior.

Formación lineal

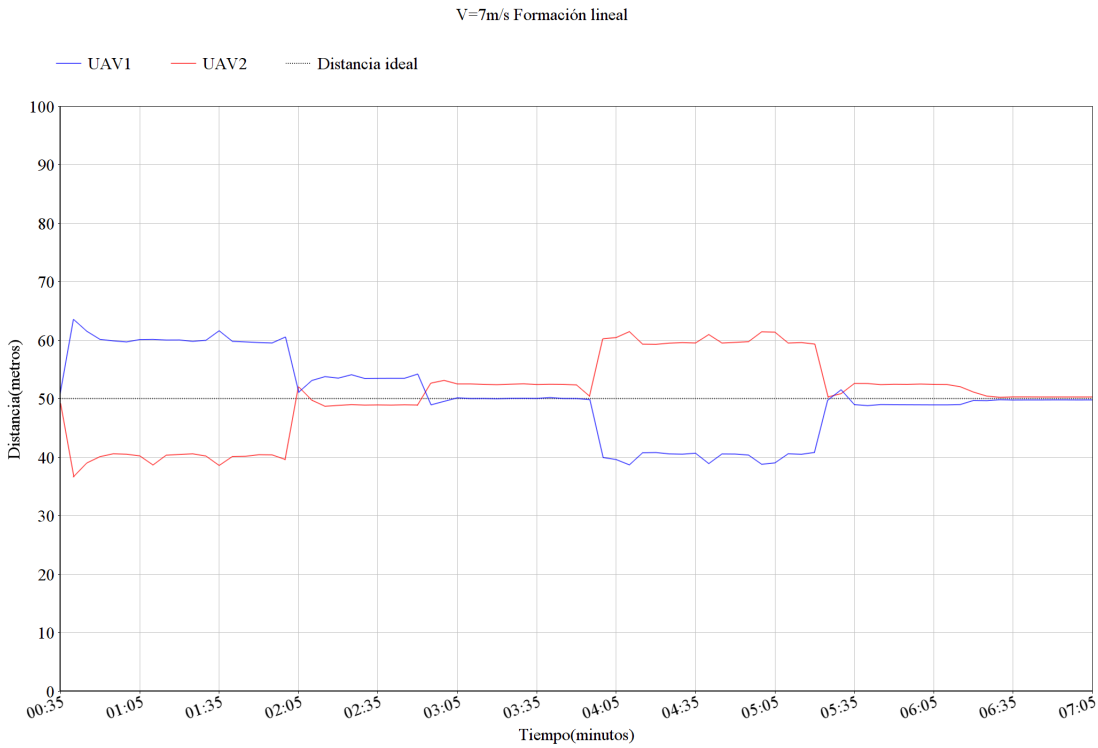

Figura 5.1: Resultados del experimento 1.

Tabla 5.1: Información del experimento 1.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 1	Lineal	7 m/s	7:08

Con este primer experimento, podemos observar que el desfase de la distancia ideal no sobrepasa los 10 metros cuando el líder se mueve en el eje horizontal, mostrando mejores resultados cuando todo el enjambre tiene un rumbo vertical, y cuando se realiza el aterrizaje una vez acabada la misión. Sin embargo, esta velocidad de 7 m/s no es muy óptima debido al tiempo del recorrido, siendo este de 7:08 minutos

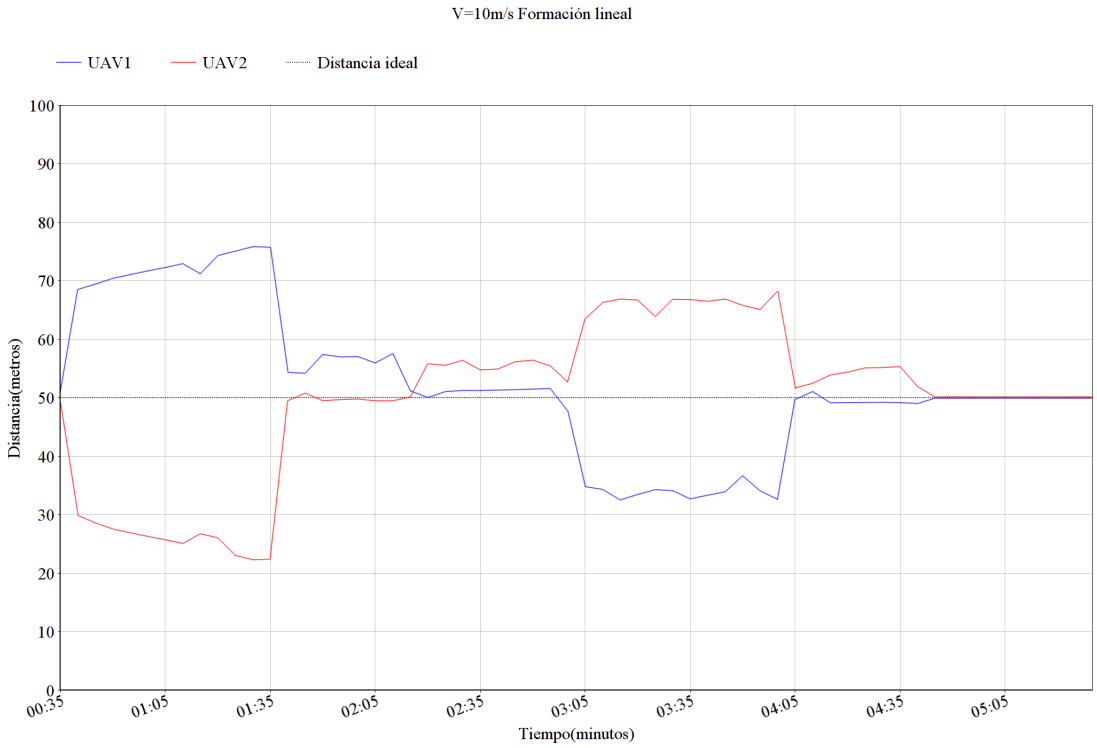

Figura 5.2: Resultados del experimento 2.

Tabla 5.2: Información del experimento 2.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 2	Lineal	10 m/s	5:36

Con la velocidad del líder de 10 m/s podemos ver que el desfase sobrepasa los 20 metros, con un pico de casi 30 en la primera etapa de la misión. Esto roza la distancia de seguridad de los drones para evitar colisiones, pero aun así es viable. También se reduce el tiempo de recorrido de la misión en 1 minuto y medio, haciendo más favorable esta combinación.

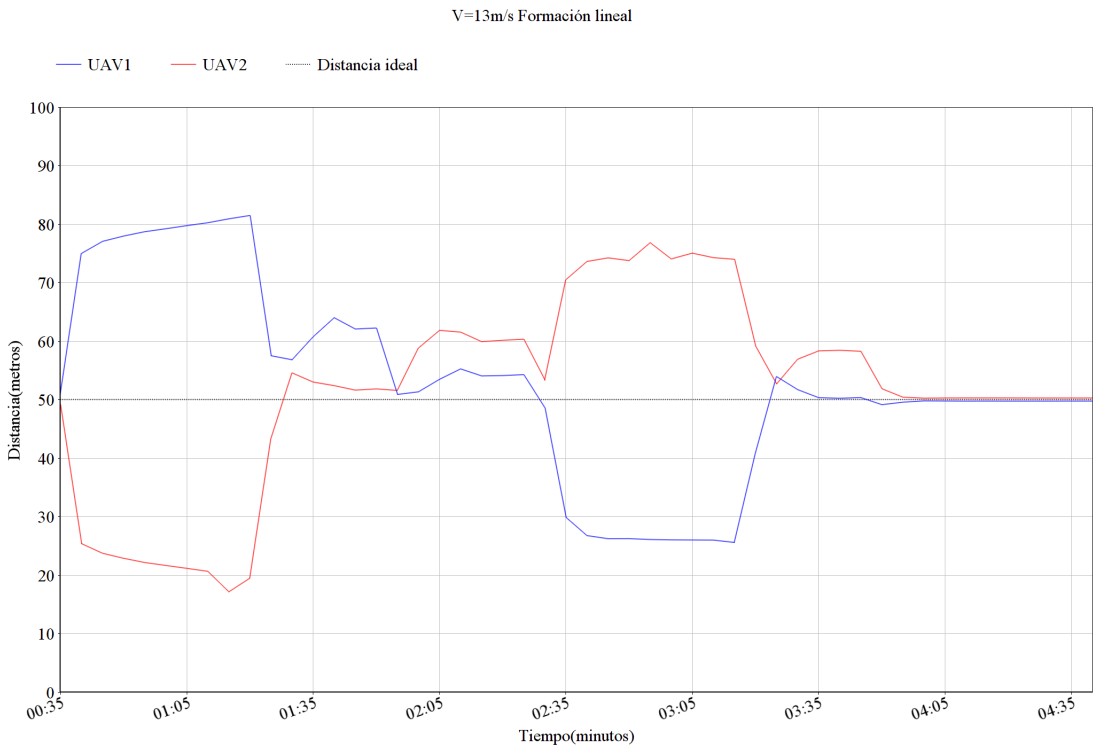

Figura 5.3: Resultados del experimento 3.

Tabla 5.3: Información del experimento 3.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 3	Lineal	13 m/s	4:44

Aumentando un poco más la velocidad, vemos que se sobrepasa los 30 metros de desfase para la primera parte de la misión. Esto es bastante peligroso, ya que el segundo dron mantiene una distancia muy peligrosa con el líder mientras que el primero se queda atrás. Para el poco tiempo que se reduce respecto al experimento anterior y el aumento del desfase, no es muy favorable sobrepasar los 10 m/s. Por otra parte, si se desea aumentar la velocidad por necesidad extrema, hay que tener en cuenta el riesgo de colisión con el dron líder.

Formación circular

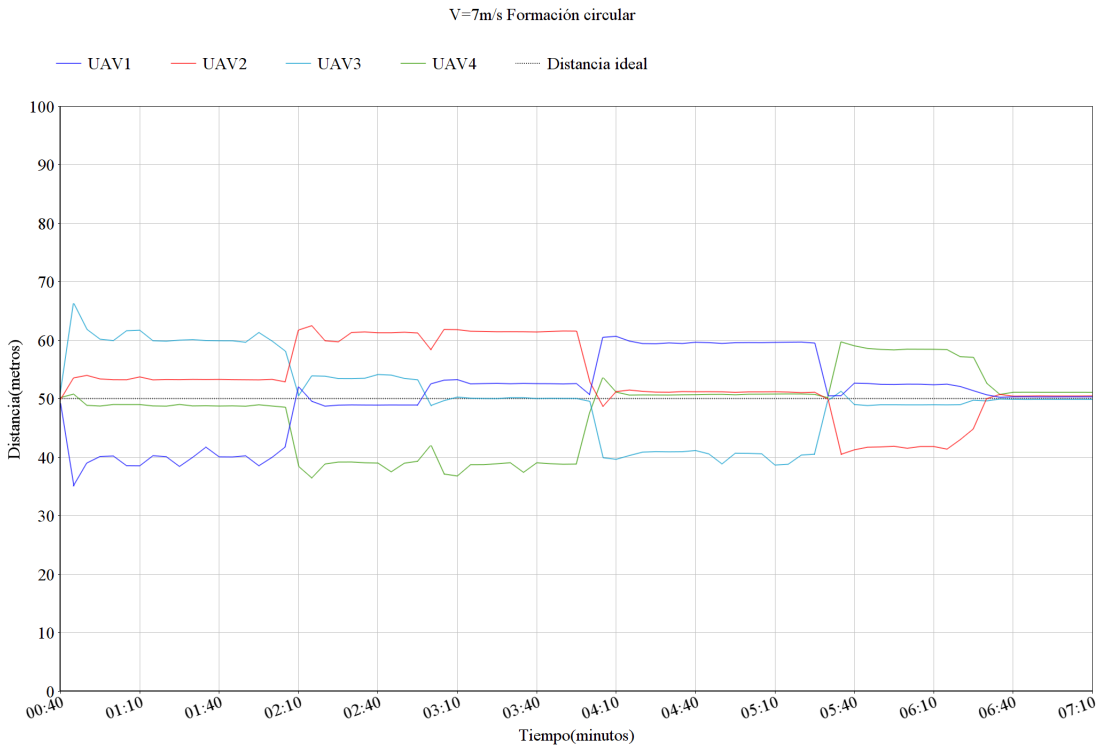

Figura 5.4: Resultados del experimento 4.

Tabla 5.4: Información del experimento 4.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 4	Circular	7 m/s	7:16

Pasamos ahora a la formación circular, aumentando el número de drones a 5 (incluyendo al líder). Se puede contemplar como, dependiendo de la posición en la formación y el movimiento direccional en la misión del líder, el desfase en la distancia aumenta o se mantiene, pero ninguno muestra inconveniencias. El tiempo de la misión no se ve afectado tampoco por la formación, mostrando un valor similar al primer experimento.

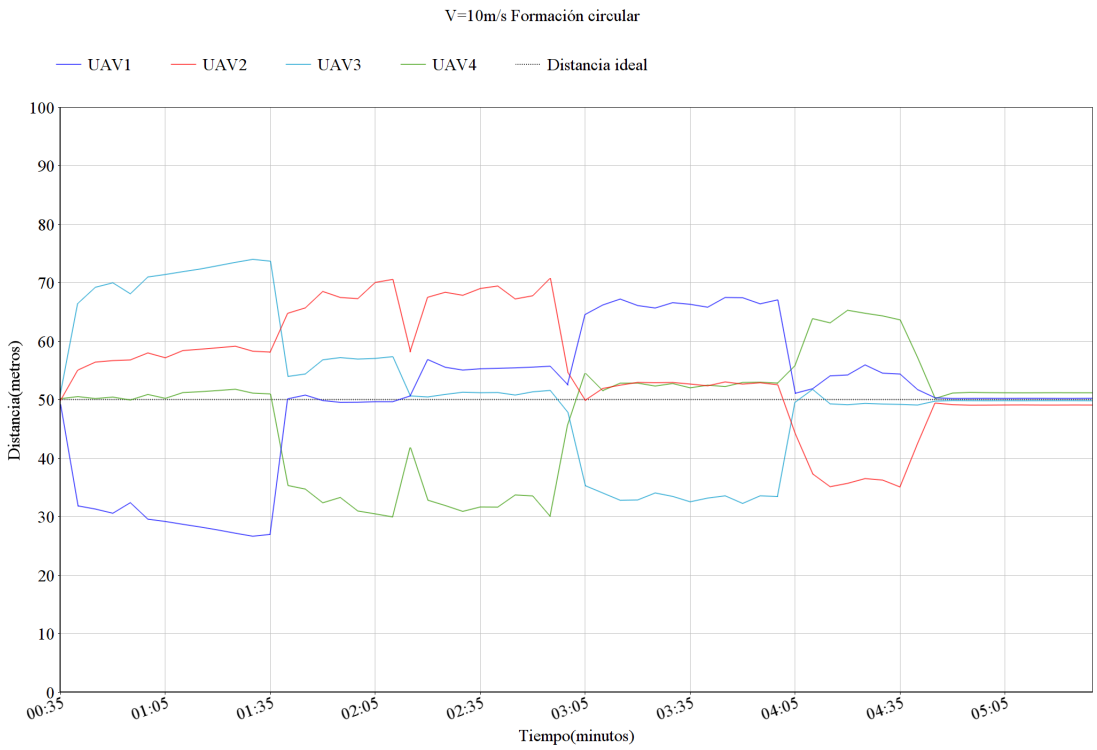

Figura 5.5: Resultados del experimento 5.

Tabla 5.5: Información del experimento 5.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 5	Circular	10 m/s	5:40

Aquí, los drones 1 y 3 muestran picos de desfase altos en la primera etapa de la misión, mientras los drones 2 y 4 se mantiene estables. Sin embargo, estos primeros drones muestran un desfase mas reducido que el resto en toda la misión restante. Con esto afirmamos que, dependiendo de la posición en la formación, el dron percibirá una mayor o menor distancia del líder según esté en la línea de dirección del movimiento de este, o no.

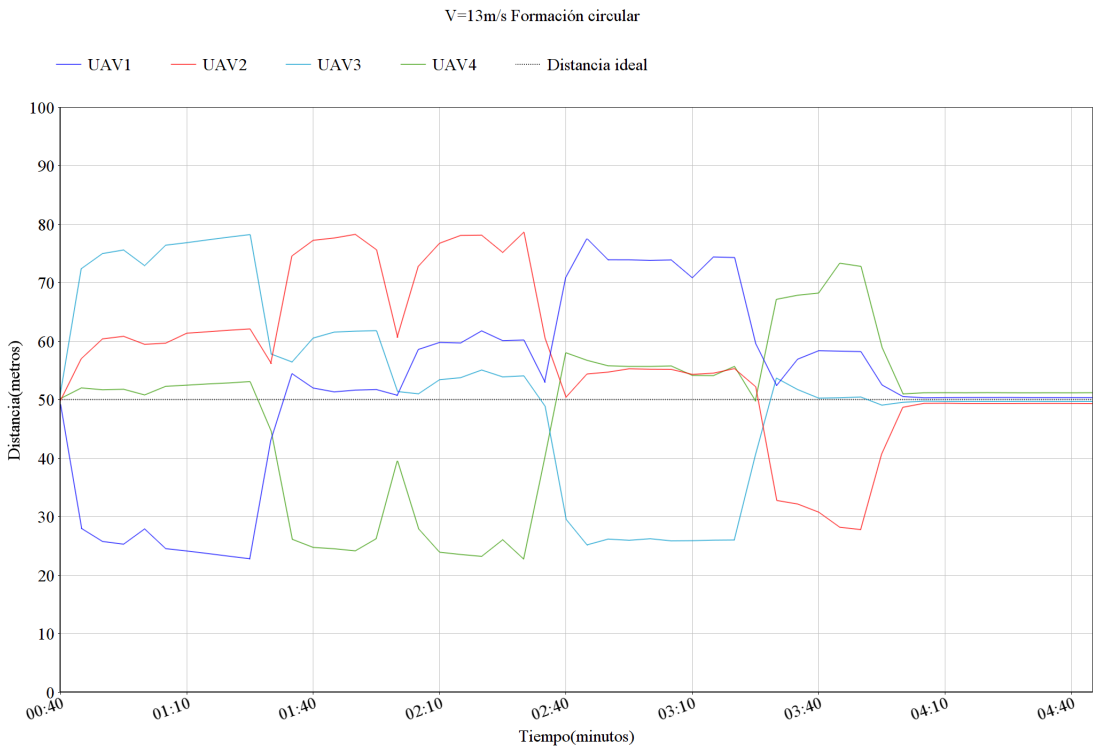

Figura 5.6: Resultados del experimento 6.

Tabla 5.6: Información del experimento 6.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 6	Circular	13 m/s	4:50

En el sexto experimento, aumentamos más la velocidad con la formación circular. Podemos observar que los drones no aumentan tanto el desfase como en la formación lineal, pero aun así alcanzan una diferencia alta respecto al líder. En este caso no habría tanto riesgo de colisión con el líder, mejorando el tiempo de la misión en 50 segundos respecto a la prueba anterior.

Formación matricial

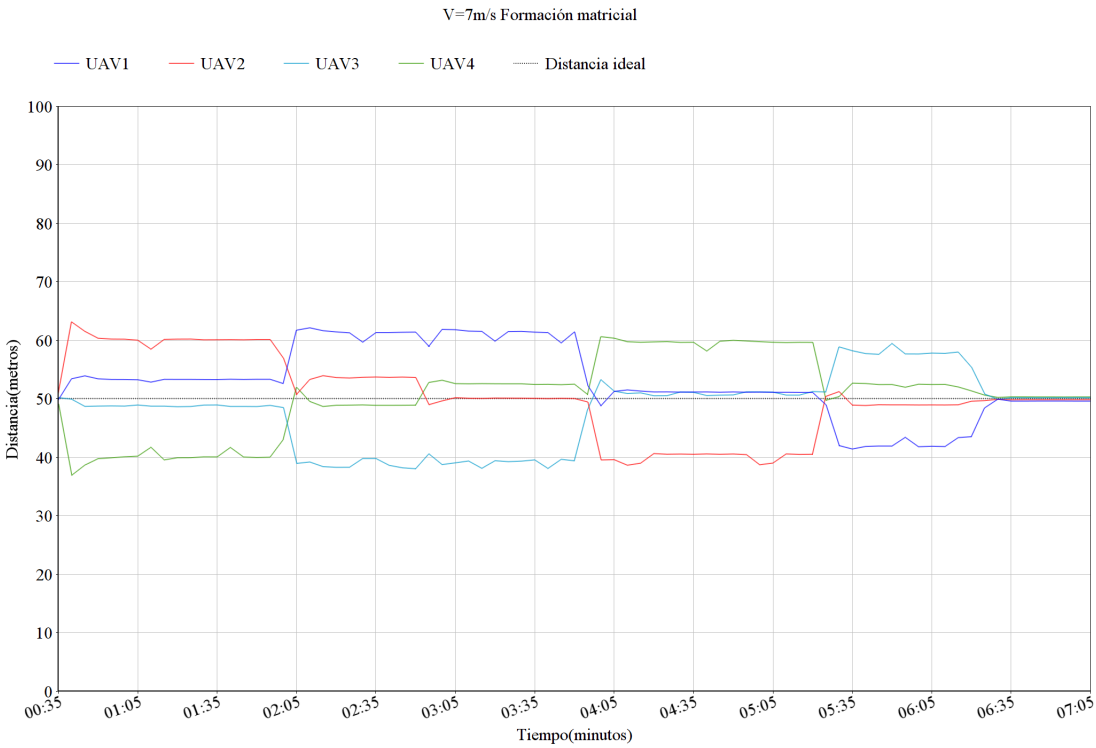

Figura 5.7: Resultados del experimento 7.

Tabla 5.7: Información del experimento 7.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 7	Matricial	7 m/s	7:14

Pasando ahora a la formación matricial, a simple vista no se contempla ninguna diferencia respecto al primer experimento de la formación circular, pero si observamos el comportamiento de los drones, esta vez son los drones 2 y 4 los que, en la primera etapa de la misión, tienen un mayor desfase, con una disminución en el resto. Como antes hemos comentado, la formación no afecta el tiempo realizado de la misión, como bien confirmamos con el valor de este, siendo similar a los experimentos previos con la misma velocidad.

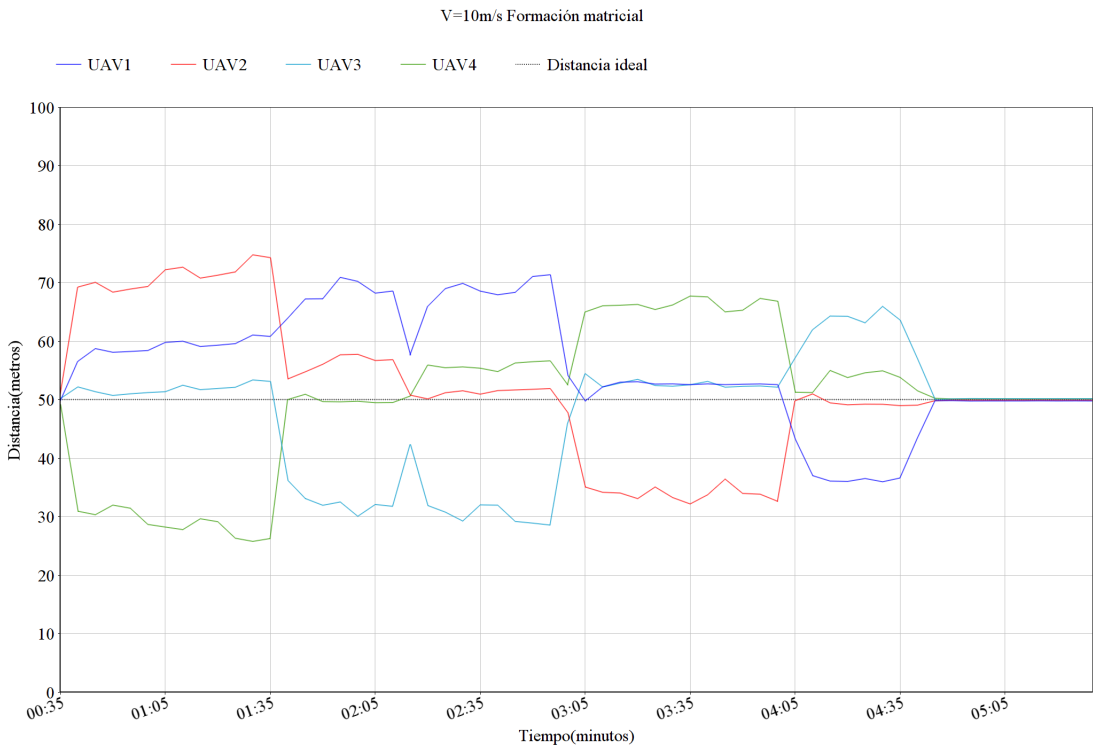

Figura 5.8: Resultados del experimento 8.

Tabla 5.8: Información del experimento 8.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 8	Matricial	10 m/s	5:39

A parte del cambio en el comportamiento de los drones respecto a su posición en la formación, en este experimento no se muestran diferencias muy grandes respecto al mismo experimento pero con formación circular. Se reduce el tiempo considerablemente aumentando la velocidad a 10 m/s, a cambio de una diferencia de 10 metros adicional en el desfase, pero sin sobrepasar una distancia peligrosa respecto al líder. Procede remarcar que consideramos que el valor de 10 m/s en la velocidad del líder es la combinación más óptima respecto al desfase de la distancia y el tiempo en completar la misión. También se observa una pequeña mejora en la distancia a la hora de aterrizar con esta formación

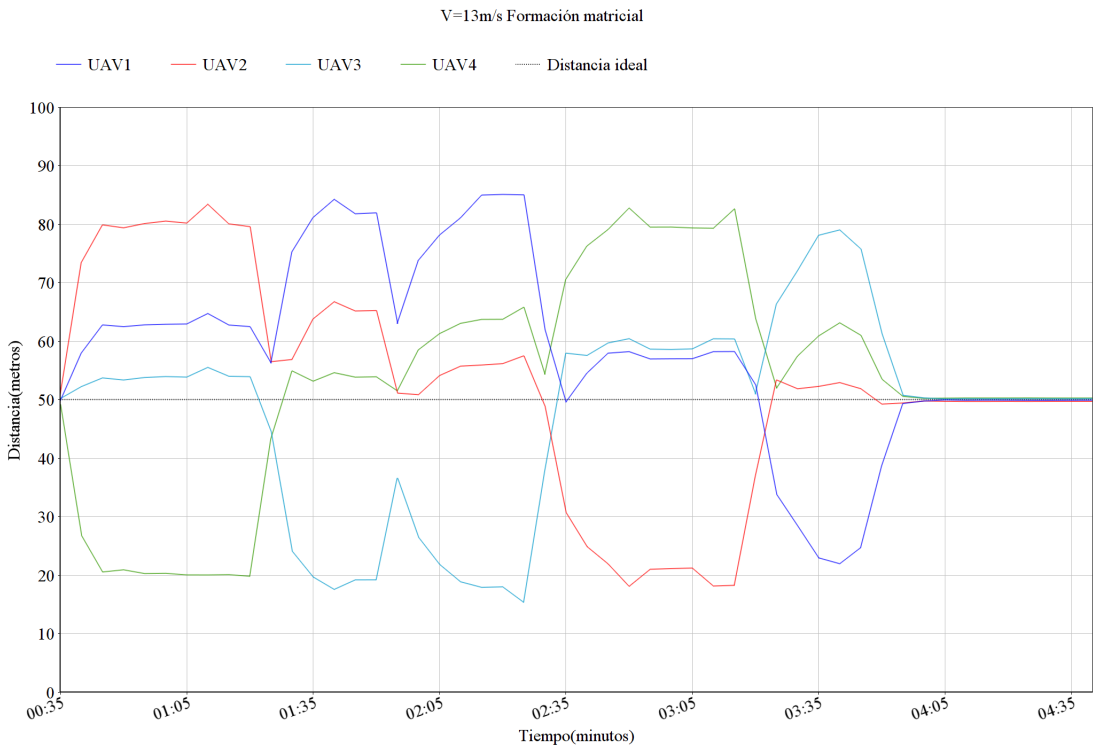

Figura 5.9: Resultados del experimento 9.

Tabla 5.9: Información del experimento 9.

	Formación	**Velocidad líder**	**Tiempo(minutos)**
Experimento 9	Matricial	13 m/s	4:47

Como último experimento, tomamos una formación matricial con velocidad de líder de 13 m/s, mostrando un peor desfase que el mismo experimento, pero con formación circular. Se sobrepasan los límites de seguridad con el dron líder en varias instancias y con fallos de inconsistencia en la formación, teniendo muy pocas zonas donde los drones alcanzan la distancia ideal. De todos los experimentos, este demuestra ser un peor caso, haciendo más ideal la formación circular si se desea aumentar la velocidad del líder.

Con todos estos datos recaudados, podemos afirmar que la mejor forma de utilizar el protocolo PlaneFollower depende de qué estrategia plantear en la misión. Si se desea utilizar pocos drones para realizar un trabajo, la formación lineal es la mas idónea. Sin embargo, si se desea utilizar más drones, entonces la formación matricial se adapta y asegura más la consistencia de la formación, pero esto solo ocurrirá en casos donde no se sobrepase la velocidad del líder en 10 m/s, haciendo que la formación circular sea más adecuada en casos donde se tenga que aumentar esta velocidad.

Para tener una vista general de la tendencia resultante de todo el conjunto de experimentos, se ha realizado una media de cada experimento, cogiendo el valor absoluto de todas las instancias de desfase de cada dron y calculando su media, volviendo a calcularla con todos los drones participantes, obteniendo una media general de cada experimento:

Tabla 5.10: Desfase medio de los experimentos de formación lineal.

Experimento	1	2	3
Valor Medio	5,110759494	9,097666667	11,8855

Tabla 5.11: Desfase medio de los experimentos de formación circular.

Experimento	4	5	6
Valor Medio	5,123006329	8,284	10,9252

Tabla 5.12: Desfase medio de los experimentos de formación matricial.

Experimento	7	8	9
Valor Medio	5,000601266	8,379166667	13,2276

La gráfica resultante de plasmar estos datos es la siguiente:

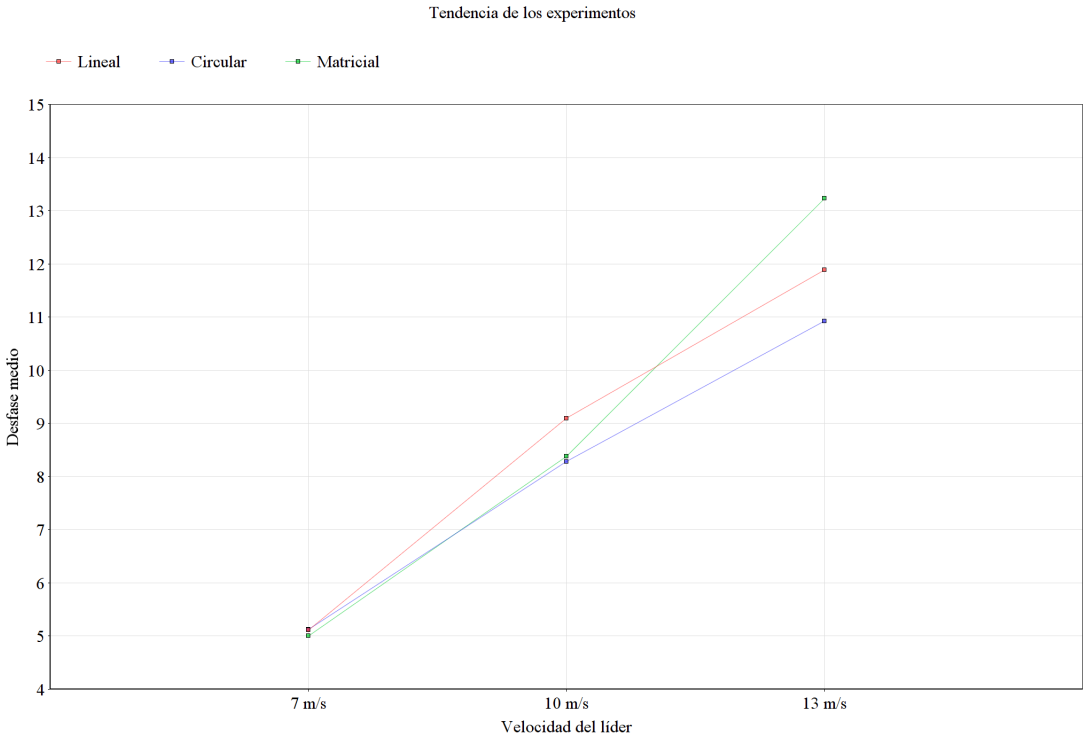

Figura 5.10: Tendencia del desfase medio de los experimentos

Analizando la media general del desfase de cada experimento, podemos destacar que cuando la velocidad del líder es de 7 m/s, las tres formaciones tienen medias similares. Cuando esta velocidad aumenta a 10 m/s, los valores ya varían entre ellos, siendo la formación lineal la que más desfase contiene, con las dos formaciones restantes teniendo valores semejantes. Por último, con la velocidad a 13 m/s ya se notan grandes aumentos entre las formaciones, siendo la circular la que menos error percibe de las tres, y la formación matricial la que más perjudicada se ve.

Con esta visión general, podemos concluir que la formación circular es la que mejor se adapta al protocolo, teniendo una tendencia de desfase menor a las demás formaciones. También podemos afirmar que la formación matricial es la que más se ve afectada a velocidades mayores, siendo la peor formación para casos donde la velocidad del líder supera los 10 m/s; sin embargo, presenta un menor desfase que la formación linear en velocidades menores e iguales a 10.

CAPÍTULO 6
Conclusiones y trabajo futuro

Con todos los avances tecnológicos que está recibiendo el sector de los drones de hoy en día, su uso aumenta continuamente, lo cual enfatiza su gran potencial. Este trabajo ha servido para contribuir en pequeña parte a estos avances, además de ser una puerta de entrada para futuros proyectos.

En el ámbito de la simulación, se ha propuesto crear un protocolo nuevo que permita la cohesión entre varios tipos de drones, creando un enjambre híbrido que permita la entrada a futuras mejoras e implementaciones. Con algunos avances más se podrá incluso lograr la ejecución del protocolo en un caso real, con drones multirrotores físicos y uno de ala fija.

Con una visión general y teórica, se ha podido organizar, diseñar y desarrollar una solución a la propuesta ideal vigente, con mejoras y cambios a medida que aparecían problemas que al principio del desarrollo no se contemplaron.

Explicando paso a paso los detalles de creación de un protocolo de vuelo, se ha conseguido con esfuerzo el lanzamiento del protocolo PlaneFollower, que mediante la programación y comunicación mediante hilos, se ha podido desarrollar para lograr un entendimiento estable por parte de los dos tipos de drones.

Con varias pruebas y experimentos, se ha calculado la eficiencia en la consistencia de la formación gracias al desfase en la distancia de los drones del enjambre con el líder. Esto ha demostrado las condiciones necesarias para ejecutar el protocolo de forma óptima, y también se ha observado qué situaciones han sido desfavorables. Siendo un factor que afecta el resultado del experimento, la velocidad configurada para el líder es crítica a la hora de mantener las distancias con el resto de elementos del enjambre. Por otro lado, la formación también ha sido importante para verificar cual recibía más inconsistencias a medida que aumentaba dicha velocidad.

Como último punto, este proyecto podría beneficiarse de futuras mejoras que podrían ampliar el funcionamiento al protocolo, dando paso a futuros proyectos que podrían desarrollarse con base a este protocolo, o incluso optimizarlo más aún:

- La primer mejora a realizar sería permitir la inclusión general de cualquier tipo de librerías relacionadas con el tipo de vehículo no tripulado aéreo dentro del simulador ArduSim. Mediante clases de comunicación generales, se podría establecer un lenguaje común para la comunicación entre todo tipo de drones, y que permitiesen cualquier tipo de cohesión de enjambre en el simulador.

- Para probar como se comportaría este protocolo de forma alternativa, se podría plantear la misma ejecución pero invirtiendo el tipo de dron del líder con el enjambre, es decir, ver como actuaría el protocolo cuando un enjambre de drones de ala fija son los que decidirán comunicarse y mantener un seguimiento de un dron multirrotor líder.

- Pasando ahora a la experimentación física, un trabajo futuro podría ser probar este protocolo en un entorno real. Concretamente, se podría utilizar un número de drones multirrotores y uno de ala fija en una pista de vuelo, y realizar distintos tipos de pruebas para medir como se desenvuelve este protocolo en situaciones reales.

- Como mejoras al protocolo, una de las importantes a destacar sería la mejora en la comunicación de los drones, disminuyendo el desfase en la distancia a un caso ideal. Aunque no se pueda conseguir del todo este objetivo, hay formas y mecanismos de comunicación que podrían ser más avanzadas, y de mejor utilidad.

Bibliografía

[1] Francisco Fabra y col. «ArduSim: Accurate and real-time multicopter simulation». En: *Simulation Modelling Practice and Theory* 87 (2018), págs. 170-190. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2018.06.009. URL: https://www.sciencedirect.com/science/article/pii/S1569190X18300893.

[2] Francisco Fabra y col. «Automatic system supporting multicopter swarms with manual guidance». En: *Computers & Electrical Engineering* 74 (2019), págs. 413-428. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2019.01.026. URL: https://www.sciencedirect.com/science/article/pii/S0045790618323577.

[3] Francisco Fabra y col. «MUSCOP: Mission-Based UAV Swarm Coordination Protocol». En: *IEEE Access* 8 (2020), págs. 72498-72511. DOI: 10.1109/ACCESS.2020.2987983.

[4] Wenjian He, Xiaogang Qi y Lifang Liu. «A novel hybrid particle swarm optimization for multi-UAV cooperate path planning». En: *Applied Intelligence* 51.10 (oct. de 2021), págs. 7350-7364. ISSN: 1573-7497. DOI: 10.1007/s10489-020-02082-8. URL: https://doi.org/10.1007/s10489-020-02082-8.

[5] Hanno Hildmann y col. «Nature-Inspired Drone Swarming for Real-Time Aerial Data-Collection Under Dynamic Operational Constraints». En: *Drones* 3.3 (2019). ISSN: 2504-446X. DOI: 10.3390/drones3030071. URL: https://www.mdpi.com/2504-446X/3/3/71.

[6] Eric W. Justh y Perinkulam S. Krishnaprasad. «A Simple Control Law for UAV Formation Flying». En: *Institute for Systems Research Technical Reports (ISR)* (2002). URL: http://hdl.handle.net/1903/6274.

[7] Robert L. Lidowski, Barry E. Mullins y Rusty O. Baldwin. «A novel communications protocol using geographic routing for swarming UAVs performing a Search Mission». En: *2009 IEEE International Conference on Pervasive Computing and Communications*. 2009, págs. 1-7. DOI: 10.1109/PERCOM.2009.4912764.

[8] Alex Kushleyev y col. «Towards a swarm of agile micro quadrotors». En: *Autonomous Robots* 35.4 (nov. de 2013), págs. 287-300. ISSN: 1573-7527. DOI: 10.1007/s10514-013-9349-9. URL: https://doi.org/10.1007/s10514-013-9349-9.

[9] R. Purta y col. «A Testbed for Investigating the UAV Swarm Command and Control Problem Using DDDAS». En: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, págs. 2018-2027. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.05.371. URL: https://www.sciencedirect.com/science/article/pii/S1877050913005140.

[10] Sotirios Spanogianopoulos, Qian Zhang y Sarah Spurgeon. «Fast Formation of Swarm of UAVs in Congested Urban Environment». En: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, págs. 8031-8036. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.1228. URL: https://www.sciencedirect.com/science/article/pii/S2405896317317342.

[11] Zain Anwar Ali y Han Zhangang. «Multi-unmanned aerial vehicle swarm formation control using hybrid strategy». En: *Transactions of the Institute of Measurement and Control* 43.12 (2021), págs. 2689-2701. DOI: 10.1177/01423312211003807. eprint: https://doi.org/10.1177/01423312211003807. URL: https://doi.org/10.1177/01423312211003807.

[12] Francisco Fabra y col. *ArduSim: Accurate and real-time multicopter simulation*. Accedido el 4 de Septiembre, 2022. URL: https://github.com/GRCDEV/ArduSim.

[13] Juan Roberto Robles Gómez. «Sistema para el control en tiempo real de un enjambre de drones basado en pilotaje manual». Tesis doct. Universitat Politècnica de València, 2021. URL: http://hdl.handle.net/10251/173533.

[14] *Java | Oracle*. Accedido el 4 de Septiembre, 2022. URL: https://www.java.com.

[15] *IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains*. Accedido el 4 de Septiembre, 2022. URL: https://www.jetbrains.com/idea/.

[16] *Ubuntu: Enterprise Open Source and Linux*. Accedido el 4 de Septiembre, 2022. URL: https://ubuntu.com/.

[17] *Introduction - MAVLink Developer Guide*. Accedido el 4 de Septiembre, 2022. URL: https://mavlink.io/en/.

[18] *Wireshark · Go Deep.* Accedido el 4 de Septiembre, 2022. URL: https://www.wireshark.org/.

[19] *Mission Planner Home - ArduPilot*. Accedido el 4 de Septiembre, 2022. URL: https://ardupilot.org/planner/.

[20] *ArduPilot - Versatile, Trusted, Open*. Accedido el 4 de Septiembre, 2022. URL: https://ardupilot.org/ardupilot/index.html.

[21] *Plane Home — Plane documentation - ArduPilot*. Accedido el 4 de Septiembre, 2022. URL: https://ardupilot.org/plane/.

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**			X	
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**			X	
ODS 7. **Energía asequible y no contaminante.**		X		
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**			X	
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**			X	
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**		X		
ODS 16. **Paz, justicia e instituciones sólidas.**			X	
ODS 17. **Alianzas para lograr objetivos.**				X

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Empezando el primer punto de los ODS, fin de la pobreza, mi trabajo de fin de grado se relaciona con el uso aéreo de drones en un simulador, cosa que no se relaciona mucho con esta temática, ya que su uso no soluciona los problemas críticos económicos de países pobres.

Después, sobre el hambre cero no procede tampoco, la utilización de un protocolo de simulación no afecta físicamente a la hambruna generada en diferentes regiones. Sin embargo, se podría adjuntar el ODS de salud y bienestar en cierta parte, permitiendo a este protocolo aéreo ser experimentado con varias misiones relacionadas con la salud pública.

Con el ODS de educación de calidad, opino que no procede, ya que se tratan temas tecnológicos que no son necesarios para aprender una base de fundamentos necesaria para la educación de jóvenes, sino que se trata más de una especialización universitaria.

Los drones en un simulador no contribuyen a la igualdad de género en ningún aspecto que pueda ser importante, esto se debe a que el objetivo de este proyecto es el facilitar una cohesión de diferentes tecnologías de drones.

Respecto a el agua limpia y saneamiento, podría relacionarse mi trabajo si se decidiera utilizar el protocolo en un caso físico, incorporando diferentes tecnologías para calcular los niveles de contaminación del agua desde el aire, pero no se podría incorporar ninguna manera de sanear esta agua a los drones.

Seguido con el ODS de trabajo decente y crecimiento económico, puede estar ligeramente relacionado, creando nuevas oportunidades a los programadores para realizar nuevos protocolos a partir del establecido en el TFG, y también crear plazas para ingenieros que traten de pasar este proyecto a un ámbito real.

También se podría relacionar con el ODS de industria, innovación e infraestructura, gracias a la versatilidad de los drones a la hora de efectuar diferentes trabajos, como podría ser analizar el terreno desde el aire para un estudio posterior de obras o incluir drones a alguna cadena de trabajo que necesite transportar pequeños materiales a largas distancias.

Como los drones solo sirven como aspecto tecnológico, no creo que afecte a reducir ni aumentar ninguna desigualdad en las sociedades.

Por otra parte, si podría relacionarse con las ciudades y comunidades sostenibles, enviando drones con este protocolo desarrollado a estudiar cambios en la meteorológica y poder prepararse para distintos tipos de desastres, pero como bien se ha comentado, este proyecto se ha llevado a cabo en un simulador y para poder hacer esto primero habría que realizar distintas pruebas en un ámbito real.

El ODS producción y consumo responsable puede estar relacionado con el desarrollo y producción de componentes tecnológicos para el uso de drones, pero como todo este proyecto se ha hecho en un entorno simulado, no perjudica ningún aspecto en este tema.

Con un enjambre de drones se podría analizar varios aspectos del clima para ver como se ha visto afectado por el cambio climático, pero volviendo a remarcar, habría que pasar este protocolo a una situación real para poder estudiar esta temática.

Respecto a la vida submarina, se puede analizar las diferentes especies que viven en la superficie acuática desde una vista aérea, pero a la hora de estudiar la vida subacuática no podemos utilizar drones aéreos para estudiarla, ya que no pueden sumergirse en el agua.

Por otro lado, sí que se puede estudiar la fauna y la flora desde una vista de águila, haciendo que el tema ODS de vida de ecosistemas terrestres pueda estar fuertemente ligado a este proyecto si se decide mejorar el protocolo de la simulación al entorno real.

No hay mucho que pueda ayudar los drones en el tema de paz, justicia y e instituciones sólidas. Podría emplearse para patrullar en diferentes aspectos relacionados con la policía, pero según mi opinión esto podría causar un problema de invasión de privacidad de los residentes.

Por último, no pienso que el desarrollo de un protocolo de vuelo en un simulador pueda apoyar a las relaciones políticas y sus uniones para lograr objetivos, marcando este ODS como no relacionado y terminando de relacionar todos los aspectos en los objetivos de desarrollo sostenible.