UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

Propuesta de un algoritmo eficiente para el despegue de enjambres de UAVs tridimensionales

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTOR/A: Zhao, Haiyang

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

Cotutor/a: Wubben, Jamie

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Informática de Sistemas y Computadores

Universitat Politècnica de València

Propuesta de un algoritmo eficiente para el despegue de enjambres de drones tridimensionales

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

Autor: ZHAO HAIYANG

Tutor: Carlos Tavares De Araujo Cesariny

01/07/2024

Resum

Els drones s'utilitzen en una àmplia gamma d'aplicacions de la societat moderna, com l'agricultura, la filmació i la cinematografia, la inspecció d'infraestructures, el rescat d'emergència, la logística i la distribució, la protecció del medi ambient i l'ús militar. La demanda en aquests camps ha impulsat el ràpid desenvolupament i innovació de la tecnologia dels vehicles aeris no tripulats.

Aquesta tesi de màster (TFM) aborda el disseny i implementació d'un sistema de control d'eixams de UAV basat en el simulador ArduSim. Quan un gran nombre de UAV estan en l'aire al mateix temps, el risc de col·lisió augmenta significativament. Per garantir el funcionament segur i eficient dels UAV eixams, aquesta investigació es centra en l'aplicació d'estratègies d'evitació de col·lisions quan diversos UAV realitzen missions conjuntes. Això és degut a que els eixams de UAV poden demostrar una major eficiència i flexibilitat en missions complexes, com la monitorització mediambiental a gran escala, la recerca i rescat després de catàstrofes i els espectacles de llum d'eixams de UAV a gran escala.

El comportament de dos eixams de UAV en condicions de vol específiques es va avaluar mitjançant simulacions experimentals, assegurant la mateixa velocitat inicial. L'estat del vol es monitoritza en temps real i s'activa un algorisme d'evitació de col·lisions quan sorgeix un risc potencial, ajustant així dinàmicament les rutes i altituds dels UAV. Els resultats experimentals van mostrar que l'eixam de UAV era capaç de completar la seva missió de manera segura i eficient, tot i amb variacions mínimes en el temps de finalització a causa dels ajustos dinàmics per evitar col·lisions.

El simulador ArduSim proporciona una plataforma avançada per validar l'eficàcia dels algorismes de control en un entorn virtual segur. Aquest enfocament no només optimitza el disseny de l'algorisme, sinó que també redueix significativament el cost i el risc de les proves en el món real. L'estudi conclou que l'algoritme d'evitació de col • lisions és eficaç en la gestió de vols d'eixams de UAV, la qual cosa proporciona una valuosa referència per a futures investigacions sobre l'optimització d'algoritmes, la millora dels mecanismes de comunicació i l'ampliació dels escenaris d'aplicació. A través d'aquest treball, hem establit una important base teòrica i experimental per al futur desenvolupament i aplicació de la tecnologia UAV, destacant la importància de plataformes com ArduSim en el desenvolupament de sistemes de control d'eixams de UAV.

Paraules clau: ArduSim, eixam de UAV, estratègia anticollisió.

Summary

Drones are used in a wide range of modern society applications, such as agriculture, filming and cinematography, infrastructure inspection, emergency rescue, logistics and distribution, environmental protection, and military use. The demand in these fields has driven the rapid development and innovation of unmanned aerial vehicle (UAV) technology.

This master’s thesis (TFM) addresses the design and implementation of a UAV swarm control system based on the ArduSim simulator. When a large number of UAVs are in the air simultaneously, the risk of collision increases significantly. To ensure the safe and efficient operation of the UAV swarm, this research focuses on the application of collision avoidance strategies when multiple UAVs perform joint missions. This is because UAV swarms can

demonstrate greater efficiency and flexibility in complex missions, such as large-scale environmental monitoring, search and rescue after disasters, and large-scale UAV light shows.

The behavior of two UAV swarms under specific flight conditions was evaluated through experimental simulations, ensuring the same initial speed. The flight status is monitored in real-time, and a collision avoidance algorithm is activated when a potential risk arises, dynamically adjusting the UAVs' routes and altitudes. Experimental results showed that the UAV swarm was able to complete its mission safely and efficiently, with minimal variations in completion time due to dynamic adjustments to avoid collisions.

The ArduSim simulator provides an advanced platform to validate the effectiveness of control algorithms in a safe virtual environment. This approach not only optimizes algorithm design but also significantly reduces the cost and risk of real-world testing.The study concludes that the collision avoidance algorithm is effective in managing UAV swarm flights, providing a valuable reference for future research on algorithm optimization, improvement of communication mechanisms, and expansion of application scenarios. Through this work, we have laid an important theoretical and experimental foundation for the future development and application of UAV technology, highlighting the importance of platforms like ArduSim in the development of UAV swarm control systems.

Keywords: ArduSim, UAV swarm, collision avoidance strategy.

Resumen

Los drones se utilizan en una amplia gama de aplicaciones de la sociedad moderna, como la agricultura, la filmación y la cinematografía, la inspección de infraestructuras, el rescate de emergencia, la logística y la distribución, la protección del medio ambiente y el uso militar. La demanda en estos campos ha impulsado el rápido desarrollo e innovación de la tecnología de los vehículos aéreos no tripulados.

Esta tesis de máster (TFM) aborda el diseño e implementación de un sistema de control de enjambres de UAV basado en el simulador ArduSim. Cuando un gran número de UAVs están en el aire al mismo tiempo, el riesgo de colisión aumenta significativamente. Para garantizar el funcionamiento seguro y eficiente de los UAV enjambre, esta investigación se centra en la aplicación de estrategias de evitación de colisiones cuando varios UAV realizan misiones conjuntas. Esto se debe a que los enjambres de UAV pueden demostrar una mayor eficiencia y flexibilidad en misiones complejas, como la monitorización medioambiental a gran escala, la búsqueda y rescate tras catástrofes y los espectáculos de luz de enjambres de UAV a gran escala.

El comportamiento de dos enjambres de UAV en condiciones de vuelo específicas se evaluó mediante simulaciones experimentales, garantizando la misma velocidad inicial. El estado del vuelo se monitoriza en tiempo real y se activa un algoritmo de evitación de colisiones cuando surge un riesgo potencial, ajustando así dinámicamente las rutas y altitudes de los UAV. Los resultados experimentales mostraron que el enjambre de UAV era capaz de completar su misión de forma segura y eficiente, aunque con variaciones mínimas en el tiempo de finalización debido a los ajustes dinámicos para evitar colisiones.

El simulador ArduSim proporciona una plataforma avanzada para validar la eficacia de los algoritmos de control en un entorno virtual seguro. Este enfoque no sólo optimiza el diseño del algoritmo, sino que también reduce significativamente el coste y el riesgo de las pruebas en el mundo real. El estudio concluye que el algoritmo de evitación de colisiones es eficaz en la gestión de vuelos de enjambres de UAV, lo que proporciona una valiosa referencia para futuras

investigaciones sobre la optimización de algoritmos, la mejora de los mecanismos de comunicación y la ampliación de los escenarios de aplicación. A través de este trabajo, hemos sentado una importante base teórica y experimental para el futuro desarrollo y aplicación de la tecnología UAV, destacando la importancia de plataformas como ArduSim en el desarrollo de sistemas de control de enjambres de UAV.

Palabras clave: arduSim, enjambre de UAV, estrategia anticolisión.

Tabla de contenidos

1. Introducción

1.1 Antecedentes de la investigación

En los últimos años, el rápido desarrollo de la tecnología de drones ha emergido como un campo importante en la investigación científica y tecnológica moderna. Los drones no solo desempeñan un papel crucial en áreas tradicionales como militar, vigilancia y logística, sino que también han demostrado su valor único en campos emergentes como agricultura, cartografía, monitoreo ambiental y rescate en desastres. Especialmente, el concepto de enjambres de drones, en el cual múltiples drones trabajan de manera coordinada, ha proporcionado nuevas ideas para abordar tareas y operaciones complejas, abriendo nuevas perspectivas de investigación y aplicación en comparación con los drones individuales.

El sistema de control de enjambres de drones es la tecnología central para realizar este concepto. Implica el control coordinado de múltiples drones con el objetivo de lograr comportamientos y ejecutar tareas colectivas complejas. Sin embargo, este campo enfrenta muchos desafíos, especialmente en áreas como comunicación, algoritmos de coordinación, procesamiento en tiempo real y confiabilidad del sistema. Entre estos desafíos, el desarrollo y la prueba de algoritmos de control de grupos son especialmente críticos, pero las pruebas en el mundo real suelen estar limitadas por costos, seguridad y operatividad.

En el ámbito del control y coordinación de enjambres de drones, los algoritmos de evasión de colisiones son cruciales. Durante tareas como el seguimiento de terreno o las operaciones de búsqueda y rescate, los drones deben comunicarse y coordinar sus movimientos en tiempo real para evitar colisiones tanto entre ellos como con obstáculos, asegurando así el éxito en la realización de sus misiones. Además, debido a la incertidumbre y complejidad del ambiente, estos algoritmos deben ser capaces de adaptarse a condiciones de vuelo variables y a eventos imprevistos, tales como cambios bruscos en la velocidad del viento o la aparición y desaparición de otros drones en el área.

El uso de ArduSim permite a los investigadores explorar los límites, la eficacia y los posibles fallos de estos algoritmos en un entorno virtual seguro, acelerando así el desarrollo de la tecnología de control de enjambres de drones. Por ejemplo, al simular el desempeño de algoritmos de evasión de colisiones bajo diversas condiciones y escenarios, los investigadores pueden evaluar la viabilidad y eficiencia de sus diseños para su

aplicación práctica. Además, los resultados de la simulación de ArduSim pueden ayudar a optimizar el diseño de drones.

ArduSim, como una plataforma avanzada de simulación de drones, proporciona un entorno ideal para el desarrollo, prueba y validación de algoritmos de control de enjambres de drones. Simula condiciones de vuelo del mundo real, permitiendo a los investigadores probar el comportamiento de los drones sin riesgos de vuelo reales. Además, ArduSim puede simular hasta 256 drones simultáneamente, lo que proporciona una herramienta poderosa para estudiar comportamientos de grupos y protocolos de comunicación complejos.

Este estudio se centra en el simulador ArduSim y el lenguaje de programación Java, con el objetivo de diseñar y desarrollar un sistema de control de evasión de colisiones eficiente y confiable para enjambres de drones. Al combinar la capacidad avanzada de simulación de ArduSim con la flexibilidad y la amplia aplicación de Java, este estudio contribuirá no solo al avance de la tecnología de control de enjambres de drones, sino también proporcionará importantes referencias e inspiración para la investigación y aplicación en campos relacionados.

1.2 Importancia del Sistema de Control de Enjambres de drones

El sistema de control de enjambres de drones es de vital importancia en el contexto actual de avances tecnológicos y el creciente uso de drones. Con la amplia aplicación de drones en áreas como la vigilancia militar, la cartografía, la monitorización agrícola, la distribución logística y el rescate en desastres, la seguridad de su operación aérea se ha convertido en un foco clave de investigación y desarrollo. La operación de enjambres de drones, debido a su eficiencia y flexibilidad, permite completar tareas que individualmente serían difíciles de lograr, pero también presenta desafíos de control complejos, especialmente en lo que respecta a evitar colisiones entre drones. A continuación, se resumen algunos puntos que destacan la importancia del sistema de control de enjambres de drones:

a) Mejora de la capacidad de ejecución de tareas: Los enjambres de drones pueden asignar diferentes tareas a cada dron, lo que permite la ejecución simultánea de múltiples tareas, como vigilancia, recolección de datos, o operaciones de búsqueda y rescate, en áreas extensas. Esta colaboración aumenta la eficiencia y el alcance de la ejecución de tareas.

b) Aumento de la capacidad de adquisición y procesamiento de datos: En áreas como la monitorización ambiental y la cartografía, los enjambres de drones pueden recopilar datos desde múltiples ángulos y en grandes cantidades. Al integrar y procesar estos datos, se puede obtener información más completa y precisa, lo que respalda la toma de decisiones más precisa.

c) Flexibilidad y escalabilidad: El diseño del sistema de control de enjambres de drones permite ajustar de manera flexible el número y la configuración de los drones para adaptarse a diferentes necesidades de tarea. Esta escalabilidad significa que el sistema puede optimizarse según circunstancias específicas, lo que aumenta su aplicabilidad y adaptabilidad.

d) Mejora de la robustez y seguridad: Cuando un dron individual experimenta fallas u otros problemas, el sistema de control de enjambres de drones puede mantener la continuidad de las operaciones mediante la reasignación de tareas y recursos. Esto aumenta la robustez del sistema y reduce el riesgo de fracaso de la tarea.

e) Rentabilidad: A pesar de que el desarrollo y la implementación del sistema de control de enjambres de drones pueden requerir una inversión inicial significativa, a largo plazo pueden reducir costos totales al disminuir la necesidad de mano de obra y aumentar la eficiencia operativa.

f) Área de investigación innovadora: El desarrollo y la aplicación del sistema de control de enjambres de drones proporciona oportunidades para explorar nuevos algoritmos, protocolos de comunicación y estrategias colaborativas. Estas investigaciones no solo impulsan el avance de la tecnología de drones, sino que también pueden tener impacto en otras áreas como la robótica, la inteligencia artificial y las comunicaciones de red.

Por lo tanto, el sistema de control de enjambres de drones no solo es un área de investigación candente en la actualidad, sino también clave para el futuro desarrollo de la tecnología de drones. Con el continuo avance tecnológico y la expansión de los escenarios de aplicación, se espera que los enjambres de drones puedan ofrecer su valor único en más áreas, contribuyendo aún más al desarrollo de la sociedad humana. La aplicación de plataformas de simulación como ArduSim promoverá en gran medida la investigación y el desarrollo de la tecnología de enjambres de drones, impulsando la industria de drones hacia niveles tecnológicos más altos.

1.3 Objetivos de la Investigación y Resumen de la Estructura del Documento

1.3.1 Objetivos de la investigación

Con el rápido desarrollo y la creciente aplicación de la tecnología de drones, el potencial y los desafíos de la operación de enjambres de drones (UAV) también se hacen cada vez más evidentes. La operación de enjambres de drones implica el trabajo coordinado de múltiples drones, lo que puede mostrar una eficiencia y flexibilidad incomparables para llevar a cabo tareas complejas. Sin embargo, a medida que aumenta la complejidad de la operación, asegurar el vuelo seguro de cada dron en el enjambre, especialmente evitando colisiones entre ellos de manera efectiva, se convierte en un problema que necesita ser abordado con urgencia.

Con base en esto, los principales objetivos de esta investigación pueden resumirse en los siguientes puntos:

a) Diseñar algoritmos de control de grupo eficientes: desarrollar algoritmos eficientes aplicables a enjambres de drones para lograr una coordinación y control precisos, asegurando la finalización exitosa de las tareas. Esta investigación tiene como objetivo explorar y diseñar un mecanismo efectivo de evasión de colisiones entre enjambres de drones. A través de un análisis integral y mejoras en las tecnologías existentes, se desarrollarán algoritmos inteligentes que se adapten a diferentes entornos y requisitos de tareas. Además, considerando los desafíos específicos que enfrentan los enjambres de drones en diferentes etapas de vuelo (como despegue, crucero y aterrizaje), se trabajará en resolver los riesgos de colisión en estas situaciones para mejorar la seguridad y eficiencia general de los enjambres de drones.

b) Realizar simulaciones y pruebas en tiempo real de enjambres de drones: utilizando el simulador ArduSim como herramienta principal de investigación, se simularán las condiciones de vuelo y escenarios de colisión de enjambres de drones en el mundo real para validar la efectividad de los algoritmos propuestos de evasión de colisiones. Al reproducir posibles situaciones de colisión en un entorno controlado, se analizarán las respuestas de los enjambres de drones en diferentes condiciones, así como la contribución de los algoritmos para mejorar la coordinación y seguridad del grupo. Esto no solo ayudará a optimizar el diseño de los algoritmos, sino que también reducirá significativamente el riesgo y los costos de las pruebas prácticas.

c) Evaluar el rendimiento y la confiabilidad del sistema: mediante una serie de experimentos de simulación, evaluar el rendimiento y la confiabilidad del sistema en diferentes condiciones y escenarios.

1.3.2 Descripción general de la estructura del documento

La estructura de este documento tiene como objetivo mostrar de manera integral el proceso de diseño, desarrollo y prueba del sistema de control de enjambres de drones. La organización de este documento es la siguiente:

1.Introducción: Se presenta el contexto de la investigación, la importancia del sistema de control de enjambres de drones, los objetivos del estudio, y la estructura general del documento.

2.Revisión de la literatura: Se analiza el progreso actual en el campo del control de enjambres de drones, con un enfoque especial en la aplicación de simuladores en la investigación del control de enjambres de drones.

3.Descripción del simulador ArduSim: Se proporciona una descripción detallada de las características y funciones del simulador ArduSim, así como su aplicación en este estudio.

4.Diseño del algoritmo: Se describe en detalle el principio del algoritmo de detección de colisiones, el mecanismo de detección de colisiones, el proceso de implementación del algoritmo, etc.

5.Diseño del sistema: Se detalla el diseño arquitectónico del sistema de control de enjambres de drones, la estrategia y lógica de detección de colisiones, y el diseño de la interfaz de usuario.

6.Desarrollo e implementación del sistema: Se describe el proceso de desarrollo del sistema, incluida la implementación de la programación, la configuración del entorno de simulación, y el desarrollo de la interfaz de usuario.

7.Experimentación y análisis de resultados de simulación: Se muestran una serie de experimentos de simulación realizados con ArduSim, así como el análisis y la discusión de los resultados experimentales.

8.Conclusiones y perspectivas: Se resumen los logros de la investigación, se discuten las limitaciones del estudio, y se sugieren direcciones para futuras investigaciones.

9.Referencias: Se enumeran todas las referencias citadas en el documento.

10.Apéndice: Se proporciona información adicional relevante, como fragmentos de código, datos adicionales y gráficos, entre otros.

A través de esta estructura, el documento tiene como objetivo presentar de manera integral y sistemática el proceso de investigación del sistema de control de enjambres de drones, desde la exploración teórica hasta la aplicación práctica, para proporcionar referencia y orientación para futuras investigaciones en este campo.

2. Revisión de literatura

2.1 Avances en la tecnología de evitación de colisiones entre enjambres de drones

En la actualidad, con el rápido desarrollo de la tecnología de drones, los sistemas de evitación de colisiones son cruciales para el funcionamiento seguro de los enjambres de drones. Con la expansión de las aplicaciones de los drones, especialmente en entornos complejos como el espacio aéreo urbano y las misiones de rescate en zonas de desastre, la tecnología efectiva de evitación de colisiones se vuelve aún más importante. En el estudio realizado por Raja y otros en 2019, se demostró cómo implementar la evitación de colisiones entre drones en un entorno grupal mediante la aplicación de la técnica de aprendizaje profundo Q-learning. Esta tecnología no solo aumentó la autonomía de los drones, sino que también mejoró su capacidad de adaptación a entornos desconocidos[1].

El aprendizaje profundo Q-learning, como un método avanzado de aprendizaje automático, es especialmente relevante en la aplicación a enjambres de drones. Este método ajusta continuamente la estrategia de vuelo de los drones para adaptarse a los cambios dinámicos del entorno, previniendo así colisiones potenciales. Este aprendizaje se basa en la retroalimentación obtenida de experiencias de vuelo pasadas, permitiendo a los drones autoaprender y optimizar sus trayectorias y comportamientos de vuelo.

Además, en 2021, Wei y otros realizaron una revisión exhaustiva de las tecnologías existentes de evitación de colisiones, señalando varias direcciones potenciales para el desarrollo futuro, como la integración de sistemas de percepción más avanzados y algoritmos más complejos para lograr una evitación de obstáculos más precisa[2]. Estos avances tecnológicos no solo pueden mejorar la seguridad de los drones en operaciones independientes, sino también aumentar la eficiencia y seguridad operativa de todo el enjambre de drones durante trabajos colaborativos[3].

Otro aspecto clave de la tecnología de evitación de colisiones es la mejora en la capacidad de percepción del entorno. Equipados con sensores avanzados y cámaras, los drones pueden percibir con mayor precisión el entorno circundante, incluyendo la posición y el estado de movimiento de otros drones. Esta información es procesada rápidamente mediante algoritmos avanzados, asegurando que, incluso en condiciones extremas, los drones puedan tomar decisiones de vuelo rápidas y precisas.

2.2 Estrategias de control e integración de sistemas para enjambres de drones

En la investigación sobre el control de enjambres de drones, los científicos enfrentan el desafío de garantizar que estos sistemas puedan operar de manera eficiente y segura sin interferencias externas. Para enjambres de drones a gran escala, el estudio de Li y sus colaboradores destaca particularmente la importancia del control preciso y la evitación de colisiones en entornos complejos[4]. Su enfoque se centra en lograr una comunicación y estrategias de control efectivas entre los drones para mantener la forma y la funcionalidad del enjambre, al mismo tiempo que asegura que cada dron pueda responder de manera independiente a los cambios ambientales cuando sea necesario.

Equipos de investigación como el de Asaamoning y sus colaboradores exploran la posibilidad de tratar a los enjambres de drones como sistemas de control de red, proponiendo un método que integra tecnologías de red y computación[5]. Esto no solo optimiza la eficiencia operativa de los drones, sino que también mejora la fiabilidad de las operaciones del enjambre. Este enfoque integrado permite que los drones sean más autónomos en la ejecución de tareas, reduciendo así la dependencia del control humano.

Otro campo de investigación clave es la capacidad de autogestión de los drones, incluyendo la identificación y respuesta automática a amenazas potenciales de colisión. Esto requiere que los drones posean capacidades avanzadas de sensado y procesamiento para analizar datos del entorno en tiempo real y tomar decisiones rápidas[6]. Al mejorar el nivel de inteligencia de los drones, los investigadores esperan reducir los riesgos asociados con el vuelo en enjambre en condiciones complejas o extremas.

Teniendo en cuenta que los drones pueden operar en diferentes condiciones geográficas y ambientales, cómo ajustar y optimizar los algoritmos de control para adaptarse a entornos cambiantes es otro enfoque de investigación. Por ejemplo, el vuelo de drones en entornos urbanos enfrentará más obstáculos y fuentes potenciales de interferencia, lo que requiere que sus sistemas de control puedan adaptarse rápidamente a nuevas condiciones de vuelo, asegurando seguridad y eficiencia.

2.3 Aplicación y futuras direcciones de la detección de colisiones en enjambres de drones

En aplicaciones específicas, la tecnología de evitación de colisiones en enjambres de drones ha demostrado una enorme contribución a misiones cruciales como la búsqueda

y rescate. Por ejemplo, Khalil et al. (2022) desarrollaron un modelo de comunicación de enjambre de drones basado en el aprendizaje automático para aplicaciones de búsqueda y rescate, que puede mejorar eficazmente la eficiencia y precisión de la búsqueda[7]. Además, a través de algoritmos de aprendizaje automático, los drones pueden ajustar sus rutas de vuelo en tiempo real para adaptarse a los cambios en el entorno y situaciones de emergencia, aumentando así la tasa de éxito de la misión.

Al mismo tiempo, los enjambres de drones han demostrado una notable capacidad de trabajo en equipo en operaciones de rescate complejas. Paez et al. (2021) optimizaron el rendimiento de sistemas multi-robot en la ejecución de tareas de búsqueda y rescate aplicando un algoritmo de optimización por enjambre de partículas distribuido[8]. Este enfoque no solo mejora la flexibilidad operativa de los drones en entornos inciertos, sino que también refuerza la comunicación y eficiencia de toma de decisiones entre el enjambre.

Además, Zaini (2020) investigó el desempeño de los enjambres de drones frente a desafíos de evitación de colisiones y limitaciones de comunicación[9]. El estudio señaló que ajustando las estrategias de comunicación y los parámetros de vuelo entre los drones, se puede reducir eficazmente el riesgo de colisiones mientras se mantiene la eficiencia general y la velocidad de respuesta del enjambre[10].

Con el avance de la tecnología, se espera que los enjambres de drones logren mayores avances en autonomía e inteligencia en el futuro. Los investigadores están trabajando en el desarrollo de sistemas de percepción y marcos de decisión más avanzados, lo que permitirá a los drones ejecutar tareas en entornos más complejos y dinámicos, como la evaluación rápida y las operaciones de rescate en sitios de desastre[11]. El desarrollo futuro de estas tecnologías podría incluir el aumento de la velocidad y precisión de los algoritmos, así como mejorar la capacidad de los drones para adaptarse a los cambios ambientales y predecir eventos futuros.

3. El simulador Ardusim

3.1 Introducción a Ardusim

ArduSim es un simulador de vuelo en tiempo real diseñado específicamente para el desarrollo de protocolos de coordinación de vuelo de drones de rotor múltiple (UAVs), que permite simular el comportamiento de múltiples drones simultáneamente. La aparición de esta herramienta tiene como objetivo abordar los problemas de coste y riesgo de seguridad asociados con la prueba del comportamiento de enjambres de drones en el mundo real. Al simular con precisión el vuelo de los drones en un entorno virtual, ArduSim proporciona a investigadores y desarrolladores una plataforma de experimentación segura, controlada y rentable.

El diseño de ArduSim considera la comunicación inalámbrica ad-hoc entre los drones, lo cual es crucial para lograr una coordinación efectiva entre ellos. Puede generar la trayectoria de movimiento de cada dron en formato OMNeT++ y NS2, lo que proporciona un importante soporte de datos para la investigación de redes de drones. Además, mediante el uso del protocolo MAVLink, ArduSim puede comunicarse con controladores de vuelo abiertos existentes, simplificando así el proceso de implementación de nuevos protocolos en drones reales

ArduSim fue desarrollado por Francisco Fabra durante su etapa como estudiante de doctorado en el Grupo de Redes de Computadoras (GRC), bajo la supervisión de Carlos T. Calafate. Actualmente, el proyecto está siendo mantenido por Jamie Wubben en el mismo grupo de investigación. El desarrollo de ArduSim refleja la alta actividad investigadora en la Universidad Politécnica de Valencia, y su aplicación se ha extendido desde la investigación académica hasta aplicaciones prácticas.

Las características principales de ArduSim incluyen su compatibilidad con Windows, Linux y macOS, lo que asegura una amplia accesibilidad y flexibilidad. Los usuarios pueden personalizar la configuración experimental a través del archivo de configuración ardusim.ini, incluyendo la selección de mapas de fondo como Bing Maps o Open Street Maps, así como ajustes de parámetros y comportamiento de vuelo de los drones. Además, ArduSim también admite la implementación directa de protocolos de vuelo en hardware real, como Raspberry Pi 3 B+, lo que brinda una gran conveniencia para la investigación y desarrollo de drones.

ArduSim no es solo un simulador, también es una herramienta de investigación poderosa que puede ayudar a los investigadores a innovar en diferentes tareas y estrategias de vuelo. Por ejemplo, al abordar el problema de evitar colisiones en enjambres de drones, ArduSim puede simular el efecto de diferentes estrategias, como cambiar la altitud de vuelo o ajustar la ruta de vuelo para evitar conflictos potenciales. Esta capacidad convierte a ArduSim en una plataforma ideal para investigar el comportamiento dinámico de enjambres de drones[12].

En resumen, ArduSim se ha convertido en un recurso valioso para la investigación y desarrollo de drones debido a su alta configurabilidad, entorno de simulación similar al mundo real, y soporte para protocolos de comunicación de drones. Proporciona un sólido apoyo para el avance de la tecnología de drones y la expansión de sus aplicaciones, siendo una herramienta muy valorada tanto en el ámbito académico como en el industrial.

3.2 Arquitectura de Ardu Sim

ArduSim es un simulador altamente flexible y rico en funciones para drones de múltiples rotores, diseñado para proporcionar un entorno de simulación preciso y en tiempo real para la investigación y desarrollo de sistemas de control de drones. A continuación se presentan los principales componentes de la arquitectura de ArduSim:

a) Motor de simulación central: El núcleo de ArduSim es su motor de simulación, que puede simular la dinámica de vuelo de los drones y la interacción con el entorno. Este motor utiliza principios de física para garantizar la precisión y realismo de la simulación.

b) Módulo de comunicación: ArduSim incluye un módulo de comunicación avanzado que permite el intercambio de datos entre drones simulados y entre drones y sistemas de control. Este módulo es crucial ya que soporta la implementación de algoritmos de coordinación y control complejos.

c) Interfaz de usuario: ArduSim proporciona una interfaz de usuario intuitiva que permite a los usuarios configurar fácilmente los parámetros de simulación, iniciar y monitorear la simulación, y analizar los datos resultantes.

d) Sistema de plugins: Para aumentar la flexibilidad y la capacidad de expansión, ArduSim ha sido diseñado con un sistema de plugins que permite a los investigadores agregar nuevas funcionalidades o modificar las existentes para adaptarse a necesidades de investigación específicas.

e) Registro y reproducción de datos: ArduSim es capaz de registrar todos los datos clave durante el proceso de simulación y proporcionar funcionalidades de reproducción para facilitar el análisis y la validación posterior.

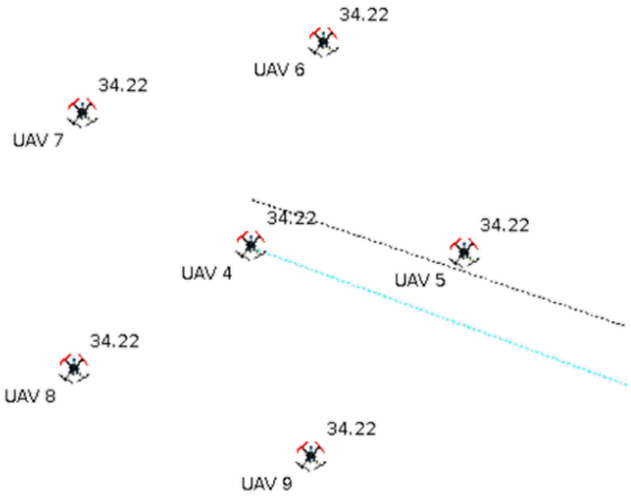

Figura 3.1 La interfaz de inicio de Ardusim

3.3 Las funciones de ArduSim

El propósito del diseño de ArduSim es proporcionar una herramienta completa y práctica para la investigación en sistemas de control de enjambres de drones. Sus principales características incluyen:

a) Simulación de múltiples drones: ArduSim puede simular simultáneamente el comportamiento de hasta cientos de drones, lo cual le convierte en una herramienta ideal para estudiar el comportamiento y el control coordinado de grupos.

b) Simulación en tiempo real: esta plataforma ofrece capacidades de simulación casi en tiempo real, permitiendo a los investigadores observar y analizar los cambios dinámicos en el comportamiento de los drones.

c) Pruebas de redes y protocolos de comunicación: ArduSim es compatible con la simulación de diversos protocolos de comunicación inalámbrica, permitiendo a los investigadores probar y validar la efectividad de estos protocolos en el control de enjambres de drones.

d) Validación de algoritmos de control de vuelo y navegación: ArduSim permite a los investigadores probar y validar diversos algoritmos de control de vuelo y navegación en un entorno de simulación, asegurando su eficacia y fiabilidad en aplicaciones prácticas.

e) Simulación de colisiones y evasión de obstáculos: mediante la simulación de diferentes condiciones de vuelo y entornos, ArduSim puede probar la capacidad de los drones para evitar colisiones y obstáculos en entornos complejos.

f) Personalización y escalabilidad: el sistema de complementos y la arquitectura flexible de ArduSim permiten su personalización y expansión según las necesidades específicas de la investigación.

En resumen, ArduSim proporciona una plataforma potente que no solo puede simular el comportamiento complejo de enjambres de drones, sino que también puede probar y validar una variedad de algoritmos y protocolos de comunicación utilizados en el control de grupos. Esto lo convierte en una herramienta valiosa para la investigación en sistemas de control de enjambres de drones.

3.4 El simulador Ardusim y su aplicación en el control de drones con Java

ArduSim es una plataforma de simulación avanzada diseñada para aplicaciones de drones, especialmente para la simulación de drones de multirrotores. Se utiliza principalmente para investigar y probar algoritmos de control de vuelo de drones, protocolos de comunicación y estrategias de coordinación de grupos. ArduSim puede simular el comportamiento de múltiples drones simultáneamente, proporcionando un entorno de prueba realista y rentable. Este simulador es especialmente adecuado para el desarrollo de sistemas de control de enjambres de drones complejos, ya que permite probar y validar la comunicación y coordinación entre drones sin necesidad de drones reales.

Java, como un lenguaje de programación ampliamente utilizado, desempeña un papel importante en el desarrollo de sistemas de control de drones. Su capacidad multiplataforma permite que los algoritmos de control de drones se ejecuten en diferentes hardware y sistemas operativos, lo que mejora la compatibilidad y la escalabilidad del sistema. Las características avanzadas de programación que ofrece Java, como la programación orientada a objetos, el sólido soporte de bibliotecas, y la buena gestión de la memoria, hacen posible el desarrollo de lógicas de control de drones complejas.

La combinación de ArduSim y el lenguaje Java proporciona importantes ventajas en el desarrollo de sistemas de control de drones. ArduSim ofrece una plataforma de experimentación que permite a los desarrolladores probar y optimizar algoritmos de

control escritos en Java en un entorno libre de riesgos. Al simular entornos reales en ArduSim, los desarrolladores pueden identificar y solucionar problemas potenciales, como retrasos en la comunicación, pérdida de datos o problemas de eficiencia en los algoritmos. Este enfoque reduce significativamente los costos y riesgos asociados con las pruebas reales, y acelera el proceso desde el concepto hasta la aplicación práctica.

En aplicaciones prácticas, los sistemas de control de drones desarrollados utilizando ArduSim y Java han demostrado su potencial en diversos campos, incluyendo respuesta a desastres, monitoreo agrícola, y recolección de datos de sistemas de información geográfica (SIG). Los investigadores utilizan ArduSim para simular diferentes escenarios y condiciones de vuelo, optimizando rutas de vuelo de drones, mecanismos de coordinación de enjambres y estrategias de gestión de energía. Por ejemplo, algunos estudios utilizan ArduSim para probar la capacidad de evasión de obstáculos de drones en entornos complejos, así como la eficiencia del trabajo en enjambres de drones.

En el futuro, es probable que la aplicación de ArduSim y el campo del control de drones se expanda a más escenarios y funcionalidades. Con el avance continuo de la tecnología de drones, la demanda de sistemas de control de drones más inteligentes y eficientes está en aumento. La combinación de ArduSim y Java podría impulsar la innovación en este campo, especialmente en áreas como la inteligencia artificial de drones, vuelo autónomo y colaboración entre múltiples drones. Además, con la mejora de las regulaciones de drones y la madurez del mercado, se espera que las aplicaciones comerciales basadas en estas tecnologías se difundan más ampliamente.

4. Diseño de algoritmos

En el rápido desarrollo de la tecnología de drones modernos, la investigación y aplicación de sistemas de drones en enjambre (UAV Swarm Systems) se ha convertido en un área candente. Estos sistemas, mediante la coordinación del comportamiento de múltiples drones, pueden llevar a cabo tareas complejas como búsqueda y rescate, monitoreo ambiental y reconocimiento militar. Sin embargo, a medida que aumenta la aplicación de enjambres de drones en entornos complejos, evitar colisiones entre diferentes enjambres se ha convertido en un problema que necesita ser resuelto de manera eficiente. Para abordar este desafío, este artículo propone una solución basada en el paradigma de líder-seguidor, y que consiste en un protocolo de evasión de colisiones para enjambres (Swarm Collision Avoidance Protocol). En este artículo, se analizarán en detalle los conceptos básicos de este protocolo.

4.1 El principio de funcionamiento

El principio de implementación del algoritmo de prevención de colisiones se basa en el paradigma "Líder-Seguidor", utilizando diferentes formaciones de vuelo en el aire, como formaciones matriciales, circulares y lineales, para mantener la posición relativa y la distancia de seguridad entre los UAV. La idea principal es que el líder determine la dirección y la velocidad del movimiento de todo el grupo, mientras que los seguidores ajustan su posición según las indicaciones del líder. El proceso principal de implementación es el siguiente:

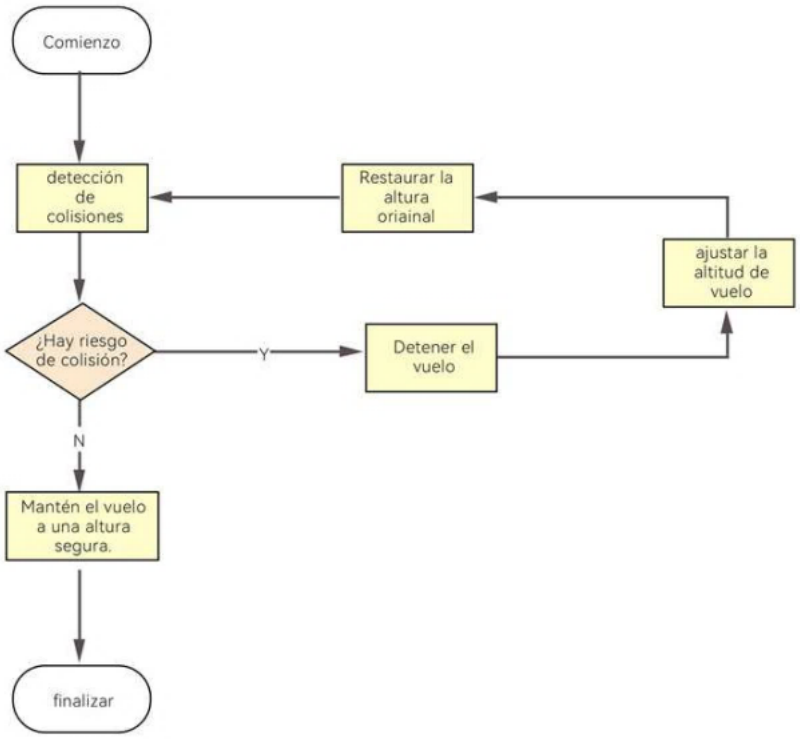

Figura 4.1 Diagrama de flujo de trabajo

a) Determinación del límite de seguridad: El líder del grupo primero calcula la distancia al UAV más lejano y agrega un margen de seguridad para determinar un círculo de seguridad.

b) Detección de conflictos de grupos: Cuando los límites de seguridad de dos grupos se superponen en el tiempo y el espacio, se considera que existe un riesgo de conflicto. En este momento, si la diferencia de altitud entre dos grupos es inferior a 10 metros, se confirma aún más la posibilidad de conflicto.

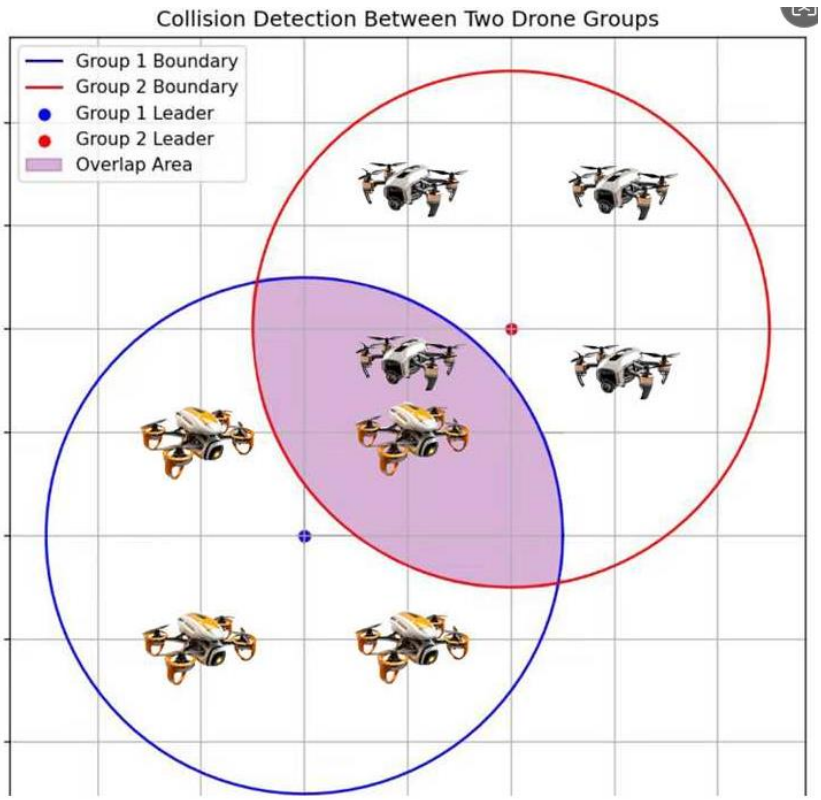

Figura 4.2 Confilic Risk

c) Mecanismo de difusión del líder: Para evitar colisiones, Cuando la distancia entre los límites de dos grupos es inferior a 50 metros, se activa la maniobra de evasión de colisiones.

d) Mecanismo de ajuste de altitud: Cuando se detecta un riesgo potencial de colisión, el líder hace que su grupo se detenga, y compara la altitud de vuelo con la del líder del otro grupo. Según el resultado de la comparación, el grupo más alto aumentará su altitud en 10 metros (no más de 120 metros), mientras que el grupo más bajo disminuirá su altitud en 10 metros (sin tocar el suelo).

e) Recuperación de vuelo y regreso a la altitud original: Cuando la diferencia de altitud entre dos grupos es superior a 10 metros, se considera que se ha resuelto el conflicto, y los grupos pueden reanudar el vuelo. Cuando no hayan más conflictos potenciales, los grupos pueden regresar a su altitud de vuelo original.

El diseño de este algoritmo tiene en cuenta la dinámica y complejidad del vuelo en grupo, logrando una prevención eficiente de colisiones en el aire mediante un intercambio de información efectivo, y ajustes de altitud adaptativos. Su ventaja radica en su capacidad para adaptarse dinámicamente a los cambios en la posición relativa entre los

grupos, y responder rápidamente a posibles conflictos de vuelo, mejorando así la seguridad y confiabilidad del vuelo.

4.2 Detalles de implementación

En un enjambre de drones, el núcleo del algoritmo de prevención de colisiones radica en implementar un protocolo de evasión de obstáculos eficiente y confiable. Este protocolo se basa en el paradigma de "líder-seguidor" y, a través de un mecanismo de comunicación cuidadosamente diseñado, asegura que diferentes enjambres de drones (llamados "grupos") mantengan una distancia de seguridad en el aire para evitar colisiones mutuas.

4.2.1 El rol y las responsabilidades del líder

Cada grupo tiene un líder designado que se encarga de medir y determinar la distancia al dron más alejado dentro de su grupo, y sobre esa base agrega un margen de seguridad para formar un círculo de protección. Además, el líder también debe transmitir regularmente información clave, incluyendo las cuatro coordenadas de los límites del círculo de protección, la altitud de vuelo, y el vector de velocidad. Cuando los círculos de protección de dos grupos se superponen en el espacio y el tiempo, se considera que existe un posible conflicto.

4.2.2 Detección y respuesta de conflictos

Determinará si hay riesgo de colisión según los límites de los círculos de protección y la información de vuelo propia. Si descubre que la distancia entre los límites de los círculos de protección de ambos grupos es inferior a 50 metros, comenzará a ejecutar las maniobras de evasión de colisiones. En primer lugar, el líder ordenará a su grupo detenerse y comparará las alturas de vuelo con el líder del otro grupo para decidir si aumentar o disminuir la altitud de vuelo para evitar el conflicto.

4.2.3 Estrategia de ajuste de altura

El ajuste de altitud es crucial para evitar colisiones. Si la diferencia de altitud entre dos grupos de vuelo es menor a 10 metros, se requiere realizar un ajuste de altitud. Las reglas específicas de ajuste son las siguientes:

Si la altitud de nuestro propio grupo de vuelo es mayor que la del otro grupo y aún no ha alcanzado el límite máximo (como 120 metros), el grupo completo aumentará su altitud en 10 metros.

Por otro lado, si la altitud de nuestro propio grupo de vuelo es menor que la del otro grupo y el ajuste no resulta en contacto con el suelo, se reducirá la altitud en 10 metros.

El propósito de este procedimiento es asegurar que la diferencia de altitud entre los dos grupos sea de al menos 10 metros, garantizando así una distancia segura en el espacio tridimensional.

4.2.4 Resolución de conflictos y reanudación de vuelos

Una vez que se haya garantizado la distancia de seguridad entre los dos grupos mediante ajustes de altura, se considerará que el conflicto ha sido resuelto. En este punto, los grupos pueden reanudar su misión de vuelo original. Para evitar futuros conflictos, los grupos también deben tener cuidado de ajustar los vectores de velocidad al regresar a la altura de vuelo original, asegurándose de que los movimientos verticales no generen nuevas amenazas de seguridad.

4.2.5 Volver a la altitud de vuelo original.

Una vez que el conflicto se haya resuelto por completo, el grupo debe volver gradualmente a su altitud de vuelo original. Este proceso se logra ajustando el componente vertical del vector de velocidad, o si el grupo utiliza navegación basada en waypoints, simplemente moviéndose directamente al waypoint establecido. Durante este proceso, el líder debe continuar monitoreando el entorno circundante para detectar cualquier nuevo conflicto que pueda surgir.

4.3 Ventajas y Aplicaciones

El algoritmo de prevención de colisiones cuidadosamente diseñado proporciona una sólida garantía para la operación segura de un grupo de drones, siendo que no solo mejora la seguridad de las misiones aéreas, sino que también aumenta la flexibilidad y eficiencia en la ejecución de las tareas. En primer lugar, este algoritmo implementa el paradigma líder-seguidor, logrando una gestión ordenada dentro del grupo y una comunicación efectiva entre grupos, reduciendo en gran medida el riesgo de colisión potencial debido a una comunicación deficiente. Además, mediante la estrategia de ajuste dinámico de la altitud de vuelo, los drones pueden reaccionar rápidamente ante conflictos, evitando

eficazmente obstáculos, y garantizando la seguridad de todo el enjambre. Este mecanismo de ajuste de altitud no solo considera la necesidad de evitar colisiones.

En cuanto a su aplicación, el potencial del algoritmo de prevención de colisiones es enorme. No solo puede aplicarse en tareas coordinadas de enjambres de drones comerciales y militares, como la entrega logística, la vigilancia ambiental, la búsqueda y rescate, y la patrulla de fronteras, sino que también puede extenderse a otros tipos de grupos de vehículos aéreos autónomos, como los autos voladores o los taxis aéreos no tripulados. En la entrega logística, los enjambres de drones pueden utilizar este algoritmo para completar eficientemente tareas de entrega a gran escala, reduciendo los tiempos de entrega y mejorando la eficiencia del servicio. En las tareas de vigilancia ambiental y búsqueda y rescate, el algoritmo permite que los enjambres de drones realicen búsquedas y recopilación de datos de manera flexible en terrenos complicados, mejorando significativamente la cobertura de la misión y la precisión de los datos. En el ámbito militar, los enjambres de drones pueden utilizar este algoritmo para realizar patrullajes y vigilancia precisos de fronteras, mejorando la capacidad de vigilancia y la capacidad de respuesta rápida ante amenazas.

En resumen, el algoritmo de prevención de colisiones no solo optimiza la seguridad y eficiencia de los enjambres de drones, sino que también proporciona un sólido soporte técnico para la implementación de diversos escenarios de aplicación, mostrando amplias perspectivas de aplicación y un impacto significativo. Con el continuo avance de la tecnología de drones y la expansión de los campos de aplicación, la importancia y el valor de este algoritmo serán cada vez más reconocidos y valorados por las personas.

5. Desarrollo e implementación del sistema

5.1 Entorno de desarrollo y herramientas

En este estudio, el sistema de control de drones se implementó en un entorno de desarrollo integral, utilizando principalmente las siguientes herramientas y plataformas:

a) Entorno de Desarrollo Integrado (IDE): IntelliJ IDEA

IntelliJ IDEA es un IDE eficiente y ampliamente utilizado que proporciona un soporte completo para el desarrollo en Java. Este entorno cuenta con funciones avanzadas de edición de código, depuración, refactorización de código e integración de control de versiones, entre otras. Elegimos IntelliJ IDEA debido a su asistencia inteligente en la codificación y características de automatización, que pueden mejorar la eficiencia de desarrollo y simplificar la gestión de proyectos complejos.

b) Java Development Kit (JDK): JDK 11

JDK 11 es una versión del kit de desarrollo de Java que proporciona el entorno necesario para ejecutar programas Java. No solo incluye el entorno de ejecución de Java (JRE), sino también herramientas de desarrollo como el compilador (javac) y el generador de documentación de Java (Javadoc). JDK 11, al ser una versión de soporte a largo plazo, ofrece estabilidad y características avanzadas, lo cual es crucial para garantizar la fiabilidad y la innovación del sistema de control de drones.

c) Plataforma de simulación: ArduSim

ArduSim es un simulador preciso y en tiempo real diseñado para drones multirotor. Permite simular y controlar múltiples drones simultáneamente. ArduSim ofrece un entorno de simulación de vuelo realista, que incluye la simulación de la comunicación por WiFi entre los drones. Utilizar ArduSim como plataforma principal de simulación permite validar y optimizar las estrategias de planificación de rutas y control de enjambres de drones sin necesidad de realizar pruebas de vuelo reales.

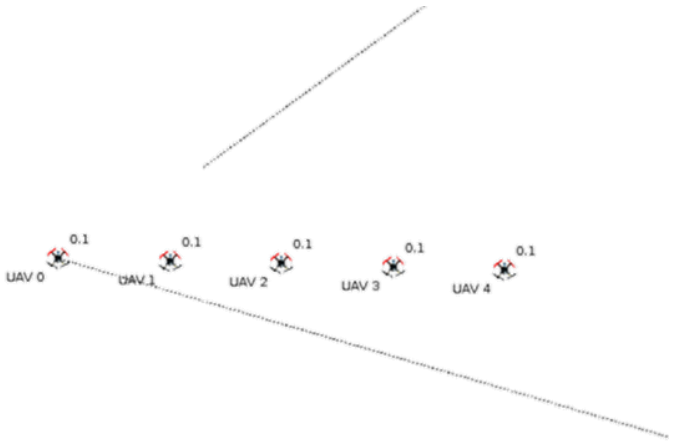

Figura 5.3 Simulador de drones ArduSim.

Además, para garantizar la calidad del código y la eficiencia en la colaboración del equipo, este proyecto integra sistemas de control de versiones como Git, así como varias herramientas de gestión de calidad de código y análisis estático de código. A través de la aplicación integral de estas herramientas, se puede asegurar que el proceso de desarrollo del sistema de control de drones sea eficiente, ordenado y que el sistema producido sea estable y confiable.

5.2 Implementación del protocolo de prevención de colisiones entre enjambres de drones

Nuestro protocolo adopta el paradigma de líder-seguidor, donde el líder es responsable de definir los parámetros de vuelo para el grupo de drones al que pertenece, incluyendo velocidad, dirección, altura y formación. Al definir la formación, se consideran diferentes formaciones 2D para los enjambres, como matrices, círculos y formaciones lineales. El dron líder también debe calcular la distancia al dron seguidor más alejado, y agregar un margen de seguridad sobre esta distancia para definir un círculo (o rectángulo) de seguridad, y transmitir su posición, el radio del círculo, la altura de vuelo, y el vector de velocidad a los otros drones.

5.2.1 Inicialización y advertencia de colisión

a) Inicialización

La inicialización es el primer paso del arranque del sistema, encargado de establecer los parámetros básicos de la misión de vuelo del grupo de drones. En el constructor de la clase CollisionWarningThread, la inicialización incluye la configuración de la altura

objetivo de la misión, la nueva altura de evasión de colisiones, el número de drones, los puntos de ruta objetivo, la formación de vuelo, y la posición específica de cada dron en la formación. Estos parámetros proporcionan la información necesaria para las siguientes medidas de advertencia y evasión de colisiones. El proceso de inicialización también inicia hilos, comenzando a monitorear el estado de vuelo de los drones y sentando las bases para advertencias de colisión inmediatas.

En la etapa de inicialización, el sistema también calcula el margen de seguridad para cada grupo de drones. Esto se logra considerando la distancia máxima entre cada dron del grupo y el dron central del grupo, más la distancia de seguridad predefinida (por ejemplo, 20 metros). Este margen de seguridad garantiza que los drones tengan suficiente espacio durante el vuelo para realizar maniobras de evasión, reduciendo así el riesgo de colisiones.

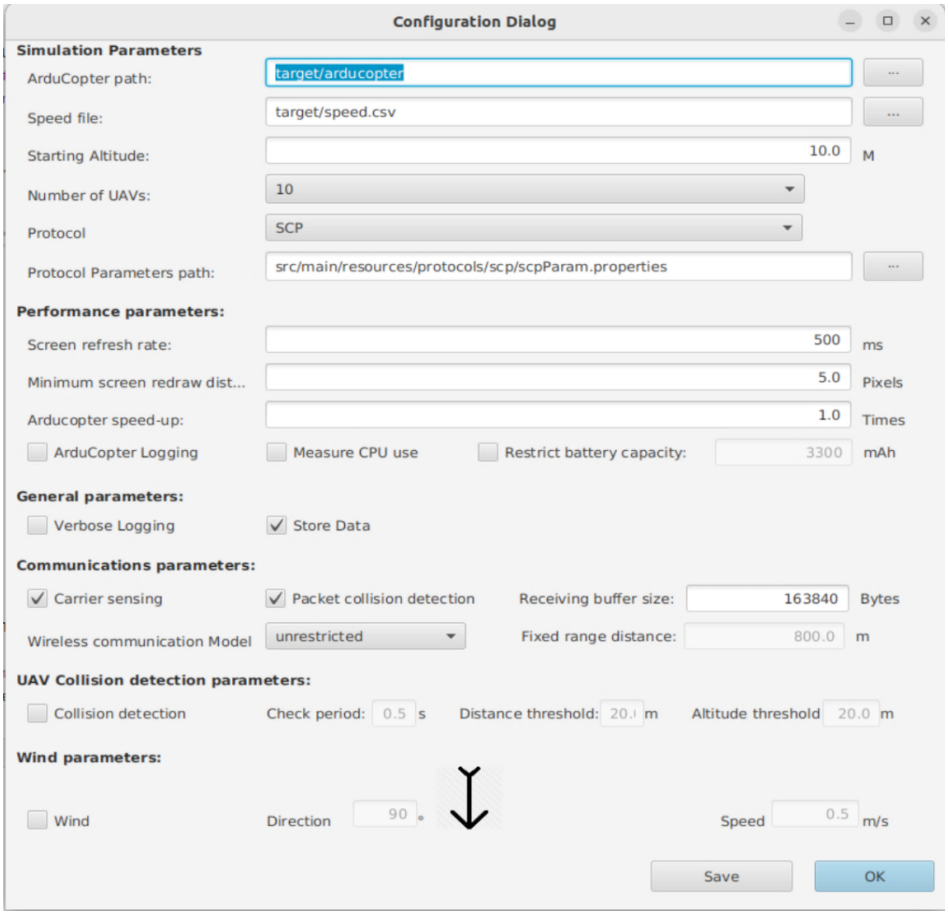

Figura 5.4 Configuración de parámetros.

b) Advertencia de colisión

La advertencia de colisión es un proceso continuo de monitoreo y evaluación durante la operación del sistema, utilizado para detectar en tiempo real la posición relativa y la diferencia de altitud entre drones, con el fin de evaluar el riesgo de colisión. A través del

método de ejecución en el bucle run, el sistema verifica periódicamente la información de posición de los drones, calculando la distancia y la diferencia de altitud entre los drones clave de los dos grupos. Cuando la suma de las distancias entre los radios de protección de los dos enjambres de drones en la dirección horizontal es menor que la suma de los márgenes de seguridad de ambas partes, y la diferencia de altitud relativa es inferior a 10 metros, el sistema determina que existe un riesgo de colisión.

Una vez detectado el riesgo potencial de colisión, el sistema activa de inmediato el mecanismo de advertencia de colisión, suspendiendo primero la tarea de vuelo actual de los drones afectados, y luego ajustando su altitud de vuelo según la lógica predefinida para evitar una posible colisión. La implementación de este mecanismo de advertencia depende de la comunicación y capacidad de control en tiempo real de los drones, lo que garantiza que los drones puedan reaccionar rápidamente en situaciones de emergencia, mejorando efectivamente la seguridad del vuelo de los enjambres de drones.

A través de esta serie de iniciativas de inicialización y mecanismos de advertencia de colisión refinados, los enjambres de drones pueden lograr una coordinación eficiente en entornos de vuelo complejos, reduciendo significativamente la probabilidad de accidentes por colisión y asegurando la continuidad y seguridad de la ejecución de las misiones. Esto no solo refleja la tecnología avanzada de los drones modernos, sino que también proporciona una garantía sólida de seguridad para la aplicación de enjambres de drones.

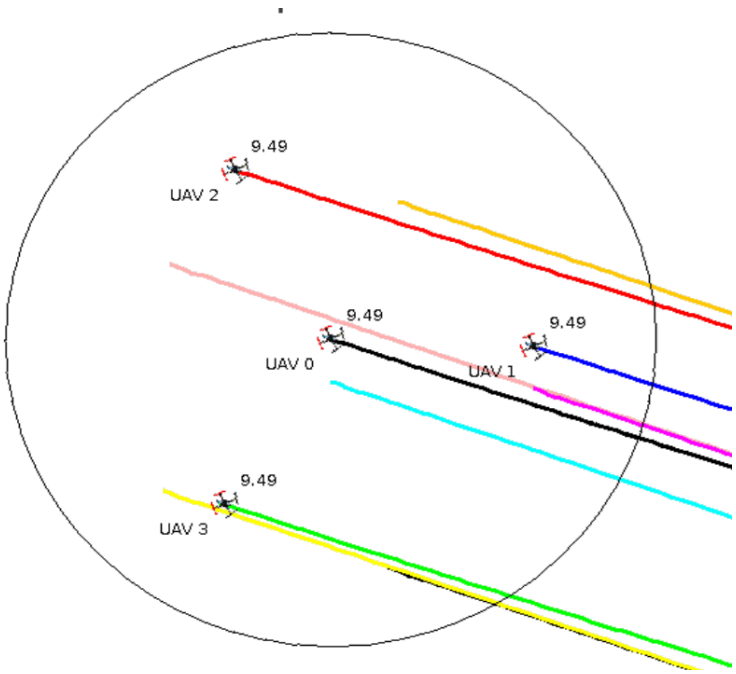

Figura 5.5 Zona de alerta de seguridad.

La figura 5.5 muestra el área de seguridad para un grupo de drones. En el gráfico, cuatro drones forman una unidad, con UAV0 como líder del grupo. El área circular alrededor del líder se considera el área de seguridad para los drones. Cuando dos enjambres de drones ingresan al área de seguridad del otro, se activa una alarma de seguridad y se toman medidas de seguridad.

5.2.2 Detección y respuesta ante posibles colisiones

En los sistemas de drones no tripulados (UAV), el diseño de un mecanismo de detección y respuesta a colisiones es crucial para garantizar que el conjunto de drones pueda trabajar de manera coordinada y segura durante la ejecución de tareas, evitando colisiones en el aire entre ellos. Este artículo, basado en el fragmento de código mencionado anteriormente, describe detalladamente un proceso práctico de detección y respuesta a colisiones, que abarca todo el proceso desde la detección de posibles colisiones, la toma de decisiones, hasta la ejecución de acciones de evasión.

a) Detección de colisiones

En los sistemas de múltiples drones, la monitorización del riesgo de colisión se realiza mediante el seguimiento continuo y la evaluación de la posición actual, la altitud de vuelo y la ruta programada de cada dron. En cuanto a la implementación del código, se realiza un bucle de ejecución de la función run para detectar la distancia relativa y la diferencia de altitud entre los drones. Los parámetros clave involucrados en este proceso incluyen la información de posición de los drones líderes de cada grupo (n1, n2), así como sus márgenes de seguridad respectivos (safetyMarginSwarm1, safetyMarginSwarm2). Cuando la distancia horizontal entre dos drones o enjambres de drones es menor que la suma de sus márgenes de seguridad, y la diferencia de altitud relativa es menor a 10 metros, el sistema determina que existe un riesgo potencial de colisión.

b) Mecanismo de respuesta

En el protocolo, la estrategia principal para evitar colisiones es ajustar la altitud de vuelo de los enjambres de drones involucrados. Calculando una nueva altitud de misión (newMissionAltitude), el sistema ordena a cada dron ajustar su altura hacia arriba o hacia abajo para asegurar que la diferencia de altitud relativa entre dos enjambres de drones permanezca dentro de un rango seguro. Esta operación de ajuste de altitud se realiza cambiando el modo de vuelo del dron a GUIDED, y moviéndolo hacia la nueva altitud. Una vez completado el ajuste de altitud, el dron continuará ejecutando su misión de vuelo original en la nueva altitud segura.

6. Simulación experimental y análisis de resultados

En este experimento, nuestro objetivo es verificar la efectividad y eficiencia del protocolo anti-colisión mencionado anteriormente. Al simular el comportamiento de vuelo de diferentes enjambres de drones (UAV) en entornos complejos mientras realizan tareas, podemos analizar el papel del protocolo de anti-colisión en mejorar la seguridad aérea y reducir el riesgo de colisiones. El diseño experimental incluirá la configuración del experimento, los pasos de ejecución, y los tipos de datos que esperamos recopilar.

6.1 Diseño del experimento

a) Configuración de parámetros

El experimento se realiza en la plataforma de simulación Ardusim, desarrollada en Java, que puede simular el comportamiento de vuelo de drones, procesos de comunicación e interacción con el entorno. La configuración de los parámetros de simulación se basa en los indicadores de rendimiento de los drones reales, que incluyen, entre otros, la velocidad de vuelo, el tiempo de respuesta, el retardo de comunicación.

Cantidad y configuración inicial de drones: El experimento simula dos grupos, cada uno con 5 drones. La altitud inicial de vuelo se establece en 20 metros, asegurando que al inicio ambos grupos estén en la misma capa del espacio.

Misión de vuelo y rutas: Se establecen rutas de vuelo cruzadas para cada grupo, asegurando que exista un riesgo potencial de colisión entre los grupos durante el proceso de simulación.

Margen de seguridad y velocidad de vuelo: Según las características de vuelo de los drones y las condiciones ambientales, se establece un margen de seguridad adecuado para cada grupo. La velocidad de vuelo de todos los drones se establece en un valor uniforme para simplificar las condiciones del experimento.

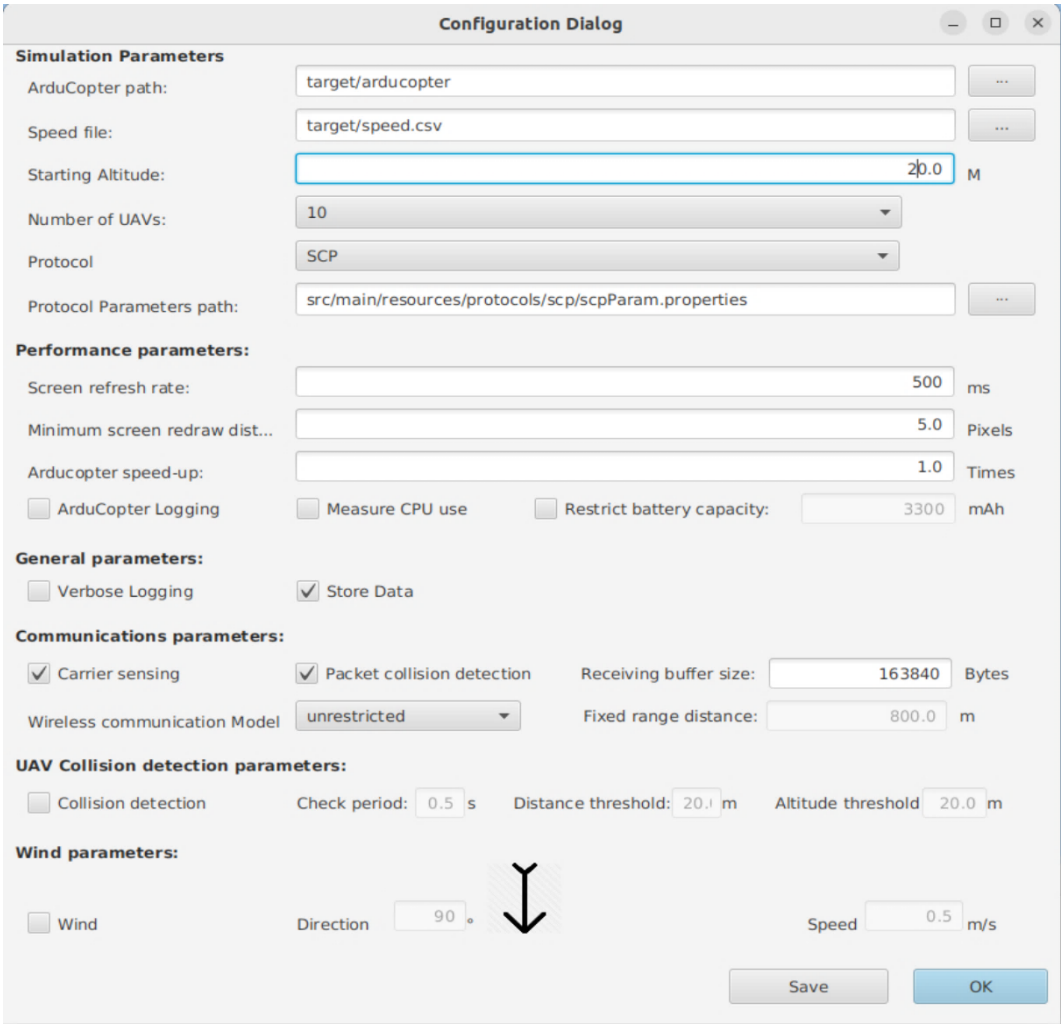

Figura 6.1 Inicialización de parámetros en ArduSim.

b) Mecanismos de comunicación y monitoreo

Elección de líderes: Cada grupo elige un líder. El líder es responsable de monitorear los riesgos potenciales de colisión.

Intercambio de información de posición y altura: Los grupos intercambian información periódicamente sobre la posición del líder, la altura de vuelo, y los márgenes de seguridad para monitorear los riesgos potenciales de colisión.

Mecanismo de detección de colisiones: Basándose en la información recopilada, el líder determina si existe riesgo de colisión. Se considera un estado de colisión potencial cuando hay superposición en las áreas de margen de seguridad de dos enjambres, y la diferencia de distancia es menor a 10 metros.

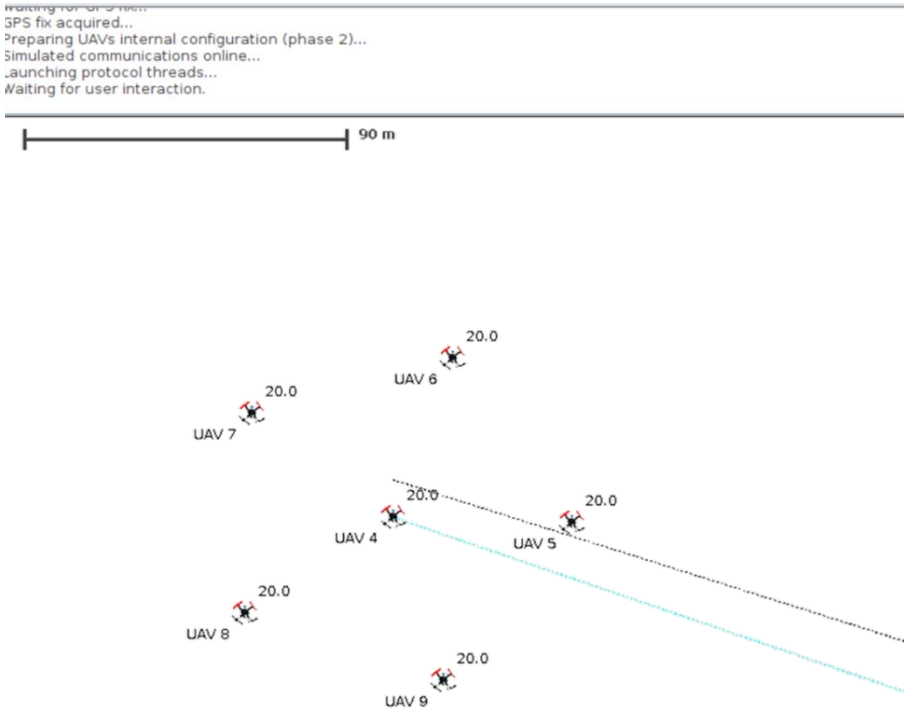

Figura 6.2 Inicialización de la interfaz.

6.2 Ejecución del Experimento

Una vez completada la configuración experimental, se entra en la etapa de ejecución del experimento, la cual constituye el núcleo de todo el proceso experimental. Esta etapa incluye principalmente la monitorización en tiempo real de la dinámica del enjambre de drones, la ejecución de estrategias de evasión de colisiones, y el registro de datos experimentales.

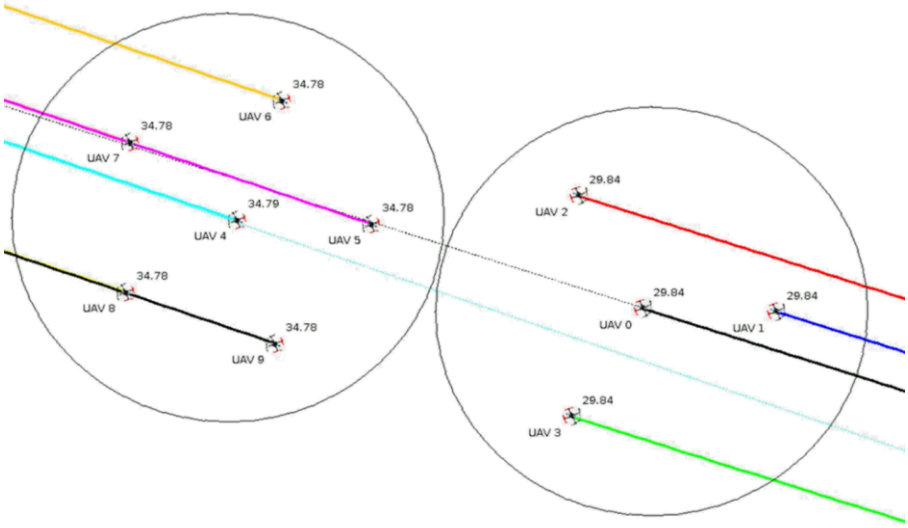

Figura 6.3 Detección de colisiones

La ejecución de la estrategia de evitación de colisiones se divide en varios pasos clave. En primer lugar, el sistema ordena la pausa de la misión de vuelo actual de todos los drones dentro de las áreas de conflicto potencial. Luego, basándose en la diferencia de altitud relativa entre los dos grupos, determina dinámicamente si cada grupo necesita ascender o descender para evitar la colisión. Para los enjambres de drones que necesitan ajustar su altitud de vuelo, el sistema calcula una nueva altitud de vuelo y, a través de la interfaz de control de vuelo de los drones, ordena que los drones se trasladen suavemente a la nueva altitud. Durante este proceso, el modo de vuelo de los drones se establece en modo GUIADO, asegurando que los drones puedan alcanzar con precisión la nueva altitud especificada.

Una vez que todos los drones potencialmente en conflicto han sido trasladados con éxito a la nueva altitud de vuelo, el sistema continúa monitoreando la distancia y la altitud entre los enjambres para confirmar si se ha evitado efectivamente el riesgo de colisión. Una vez que el sistema determina que los dos enjambres se han alejado de manera segura del área de conflicto potencial, ordena que los drones se restauren a su misión de vuelo y altitud original. En este punto, los drones ajustan gradualmente su trayectoria de vuelo y altitud hasta regresar a su estado de vuelo programado original.

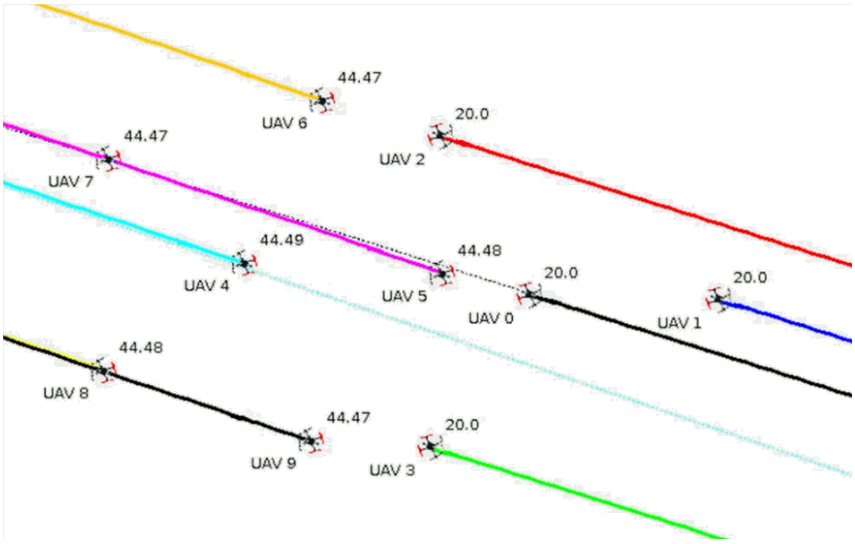

Figura 6.4 Ajuste de la altitud de vuelo.

Durante todo el proceso de ejecución del experimento, el sistema registra en tiempo real datos clave del experimento, incluyendo la trayectoria de vuelo de los drones, cambios de altitud, tiempo de ejecución de la evasión de colisiones, y el tiempo de recuperación del estado de vuelo original, entre otros. Estos datos son crucialmente

importantes para el análisis posterior de los resultados del experimento. No solo se pueden utilizar para evaluar la eficacia y la temporalidad del algoritmo de prevención de colisiones, sino que también se pueden utilizar para analizar el comportamiento de vuelo y el rendimiento del grupo de drones al ejecutar estrategias de evasión de colisiones.

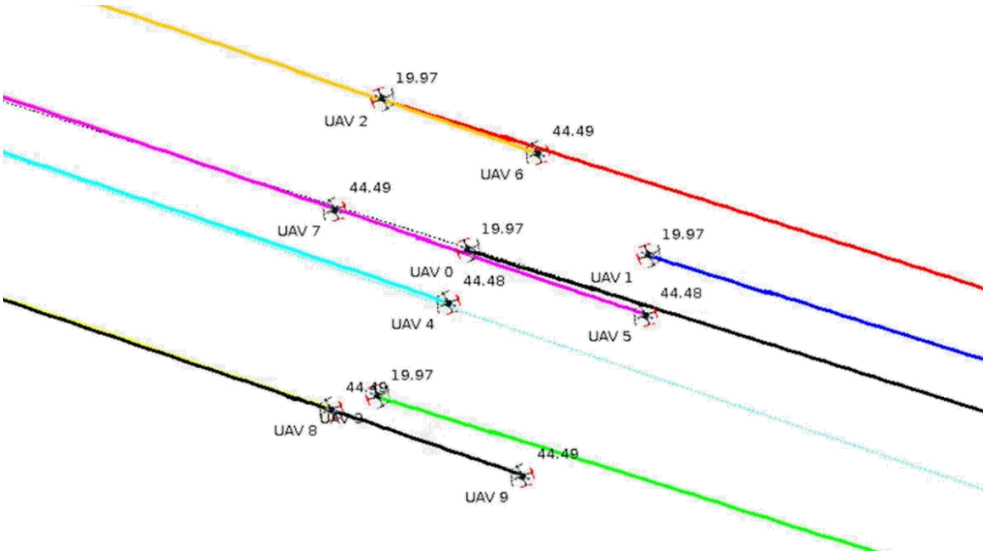

Figura 6.5 Reanudar vuelos normales

A través de una ejecución rigurosa de experimentos, y de un registro de datos detallado, se garantiza la confiabilidad y precisión de los resultados experimentales, proporcionando así una base sólida para el análisis posterior y la optimización de algoritmos.

6.3 Los resultados y el análisis del experimento son los siguientes

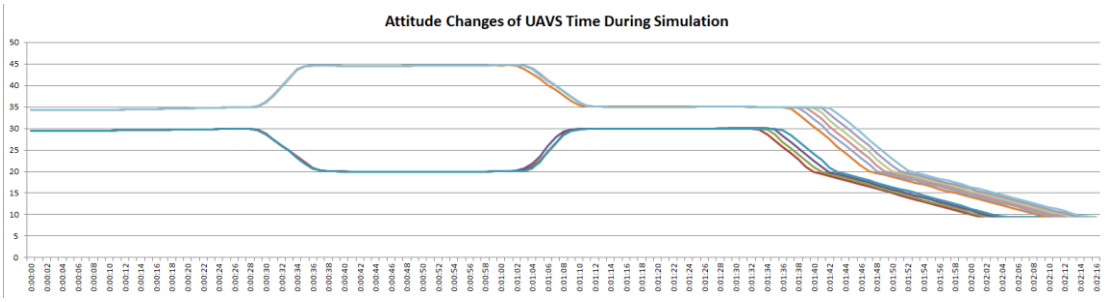

Figura 6.3 Experimento completado.

A medida que avanzaba el experimento, el enjambre de drones voló durante 2 minutos y 16 segundos, con una pausa de 26 segundos para ajustar su altitud, y una pausa para recuperar la altitud de alrededor de 1 minuto.

Además, con el aumento del número de drones, el tiempo de finalización de la tarea muestra una tendencia general al aumento, lo que podría reflejar que los drones que se

inician más tarde en la tarea probablemente necesiten realizar más ajustes en la fruta de vuelo para evitar colisiones con drones que se iniciaron antes o con otros obstáculos.

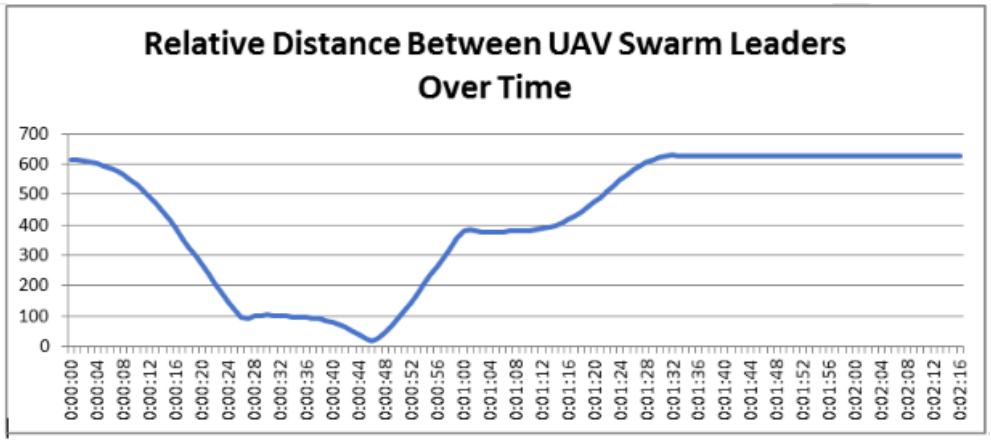

Figura 6.4 Distancia relativa entre líderes a lo largo del tiempo

Esta gráfica muestra la distancia relativa entre los líderes de un grupo de drones a lo largo del tiempo. Inicialmente, la distancia disminuye gradualmente hasta aproximadamente 300 metros, marcando el punto más bajo alrededor de las 1:00. Posteriormente, la distancia comienza a aumentar, alcanzando alrededor de 600 metros cerca de las 2:15, y se mantiene en un nivel elevado. Esta tendencia indica ajustes espaciales por parte de los drones durante la misión para evitar posibles colisiones en el aire, implementando estrategias de vuelo correspondientes.

La simulación de este proceso verifica la efectividad del algoritmo de evitación de colisiones. A través del monitoreo en tiempo real y el ajuste de la altura de vuelo, se puede asegurar la seguridad de los drones en entornos complejos. Los resultados experimentales muestran que incluso en entornos operativos densos para drones, las estrategias adecuadas de coordinación y control pueden evitar colisiones de manera efectiva y asegurar la finalización exitosa de las tareas. Este estudio proporciona una referencia valiosa para la optimización futura de los algoritmos de control de enjambres de drones y también demuestra la importancia de ArduSim en la simulación y control de drones.

Este experimento no solo valida la efectividad del algoritmo, sino que también proporciona una base teórica y soporte técnico para la operación segura de drones en más escenarios de aplicación práctica en el futuro. Esto tiene un papel importante en el impulso del desarrollo y la aplicación de la tecnología de drones.

En general, los resultados experimentales muestran que los algoritmos de prevención de colisiones en el grupo de drones pueden gestionar y ajustar eficazmente las rutas de vuelo de los drones para hacer frente a entornos de vuelo complejos y riesgos potenciales

de colisión. Aunque existen diferencias en los tiempos de finalización de la tarea, en general, mediante mecanismos de comunicación y coordinación efectivos, los enjambres de drones pueden completar sus tareas programadas de manera segura y eficiente.

7. Conclusiones y Perspectivas

7.1 Conclusiones

Este estudio, mediante un experimento simulado, investiga profundamente las estrategias de prevención de colisiones en grupos de drones durante la ejecución de tareas comunes. El diseño experimental se centró en analizar el tiempo que los drones completan las tareas para evaluar la efectividad del algoritmo anti-colisiones. Todas las velocidades iniciales de los drones fueron igualadas para eliminar la influencia de las diferencias de velocidad en los resultados del experimento. Esta configuración permitió que las diferencias en los tiempos de finalización reflejaran más precisamente el impacto directo de las estrategias anti-colisiones. A través del experimento, pudimos observar cómo el algoritmo gestionaba eficazmente las rutas de vuelo y las alturas de los grupos de drones, asegurando que completaran las tareas de manera segura y eficiente.

Los resultados experimentales revelaron pequeñas diferencias en los tiempos de finalización de las tareas entre los grupos de drones, que principalmente fueron causadas por ajustes dinámicos en las rutas de vuelo y la implementación de estrategias para evitar colisiones. Esto indica que, incluso con el control de la velocidad de vuelo y las condiciones básicas, los ajustes dinámicos realizados durante la ejecución de las tareas aún afectan la eficiencia general y los tiempos de finalización de los drones.

A pesar de los resultados positivos, el experimento también tuvo algunas limitaciones. Por ejemplo, se asumió que todas las características operativas y de respuesta de los drones eran completamente idénticas, lo cual puede no ser el caso en aplicaciones reales. Los experimentos futuros podrían incluir drones de diferentes modelos y capacidades para simular más precisamente la dinámica de grupo en entornos reales. Además, este experimento no tomó en cuenta completamente factores ambientales como la velocidad del viento y las condiciones climáticas que pueden afectar el rendimiento de vuelo. Investigaciones futuras deberían considerar estos factores externos para evaluar completamente la adaptabilidad y estabilidad de los algoritmos anti-colisiones en entornos complejos. Finalmente, los problemas de seguridad y confidencialidad de la comunicación entre drones también merecen un estudio más profundo para asegurar la seguridad de los datos y la confidencialidad de las comunicaciones cuando los grupos de drones ejecuten tareas sensibles.

7.2 Perspectivas

La investigación futura puede profundizar en las siguientes direcciones:

Optimización de algoritmos: continuar optimizando los algoritmos de prevención de colisiones, mejorando su velocidad de respuesta y eficiencia de decisión, reduciendo las discrepancias de tiempo necesarias para que los drones completen tareas, y así mejorar aún más la eficiencia general del grupo.

Aumento de la complejidad del entorno: probar los algoritmos de prevención de colisiones en entornos de simulación más complejos, como la introducción de obstáculos dinámicos, cambios en la velocidad y dirección del viento, etc., para evaluar el rendimiento de los algoritmos en entornos más cercanos a la realidad.

Validación experimental de hardware: además de los experimentos de simulación, también se pueden realizar pruebas de campo bajo condiciones controladas, utilizando drones reales para experimentos de vuelo en grupo, para verificar la efectividad y confiabilidad de los algoritmos en aplicaciones prácticas.

Expansión de los escenarios de aplicación: explorar el potencial de los algoritmos de prevención de colisiones en escenarios de aplicación más amplios, como la gestión del tráfico aéreo urbano, búsqueda y rescate en caso de desastres, monitoreo ambiental, etc., para impulsar la aplicación social y el desarrollo industrial de la tecnología de drones.

En resumen, mediante el análisis experimental y la discusión de resultados de esta investigación, se ha confirmado la efectividad y necesidad de los algoritmos de prevención de colisiones en la gestión del vuelo de enjambres de drones. La investigación futura se centrará en la optimización de algoritmos, la mejora de los mecanismos de comunicación, y la expansión de los escenarios de aplicación práctica, con el objetivo de mejorar aún más la eficiencia operativa y la seguridad de los enjambres de drones, y abrir perspectivas más amplias para el desarrollo y aplicación de la tecnología de drones

Referencias bibliográficas

[1] Raja G, Anbalagan S, Narayanan V S, et al. Inter-UAV collision avoidance using Deep-Q-learning in flocking environment[C]//2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, 2019: 1089-1095.

[2] Wei Z, Meng Z, Lai M, et al. Anti-collision technologies for unmanned aerial vehicles: Recent advances and future trends[J]. IEEE Internet of Things Journal, 2021, 9(10): 7619-7638.

[3] Agnel Tony L, Ghose D, Chakravarthy A. Unmanned aerial vehicle mid-air collision detection and resolution using avoidance maps[J]. Journal of Aerospace Information Systems, 2021, 18(8): 506-529.

[4] Li J, Fang Y, Cheng H, et al. Large-scale fixed-wing UAV swarm system control with collision avoidance and formation maneuver[J]. IEEE Systems Journal, 2022, 17(1): 744-755.

[5] Asaamoning G, Mendes P, Rosário D, et al. Drone swarms as networked control systems by integration of networking and computing[J]. Sensors, 2021, 21(8): 2642.

[6] Kobzystyi M. Drone collision warning system[J]. 2023.

[7] Khalil H, Rahman S U, Ullah I, et al. A UAV-Swarm-Communication Model Using a Machine-Learning Approach for Search-and-Rescue Applications[J]. Drones, 2022, 6(12): 372.

[8] Paez D, Romero J P, Noriega B, et al. Distributed particle swarm optimization for multi-robot system in search and rescue operations[J]. IFAC-PapersOnLine, 2021, 54(4): 1-6.

[9] Zaini A H. UAV swarming with collision avoidance and communication constraints[J]. 2020.

[10] Stolfi D H, Danoy G. An evolutionary algorithm to optimise a distributed UAV swarm formation system[J]. Applied Sciences, 2022, 12(20): 10218.

[11] Huang Y, Tang J, Lao S. Cooperative multi-UAV collision avoidance based on a complex network[J]. Applied Sciences, 2019, 9(19): 3943.

[12] https://github.com/GRCDEV/ArduSim

Apéndice

fragmento de código package com.protocols.scp.logic;

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import com.api.API;
import com.api.MoveTo;
import com.api.MoveToListener;
import com.api.formations.Formation;
import com.api.pojo.FlightMode;
import com.api.pojo.location.Waypoint;
import com.protocols.scp.gui.ScpSimProperties;
import com.uavController.UAVParam;

import es.upv.grc.mapper.DrawableCirclesGeo;
import es.upv.grc.mapper.GUIMapPanelNotReadyException;
import es.upv.grc.mapper.Location2DGeo;
import es.upv.grc.mapper.Location2DUTM;
import es.upv.grc.mapper.Location3D;
import es.upv.grc.mapper.Mapper;

public class CollisionWarningThread extends Thread {

   int missionAltitude;
   int newMissionAltitude;
   int numUAV;
   int n1=ScpSimProperties.getSwarm1IDsFirstFromSet();
   int n2=ScpSimProperties.getSwarm2IDsFirstFromSet();
   double
```

```
safteyMarginSwarm1=calculateSafteyMargin(ScpSimProperties.getSwarm1IDsFirstF
romSet(),ScpSimProperties.swarm1IDs);
        double
safteyMarginSwarm2=calculateSafteyMargin(ScpSimProperties.getSwarm2IDsFirstF
romSet(),ScpSimProperties.swarm2IDs);
        Formation f;
        Waypoint wp ;
        int index;
        volatile boolean collision=false;
        public CollisionWarningThread(int missionAltitude,int newMissionAltitude,int
numUAV,Waypoint wp,Formation f,int index) {
                this.missionAltitude=missionAltitude;
                this.newMissionAltitude=newMissionAltitude;
                this.numUAV=numUAV;
                this.f=f;
                this.wp=wp;
                this.index=index;
                this.start();
        }

        public void run() {
                while(API.getCopter(numUAV).isFlying()) {
                        drawCircle();
                        double
distance=UAVParam.uavCurrentData[n1].getUTMLocation().distance(UAVParam.ua
vCurrentData[n2].getUTMLocation());
                        double
altitudesdis=Math.abs(UAVParam.uavCurrentData[n1].getZRelative()-
UAVParam.uavCurrentData[n2].getZRelative());

        if(distance<(safteyMarginSwarm1+safteyMarginSwarm2)&&altitudesdis<10
&&!collision) {
                                collision=true;
                                //pause uavs
```

```
                        if(API.getCopter(numUAV).getMissionHelper().pause())
{
                                //change altitude
                                changeAltitude(newMissionAltitude);
                                //move with a certain height difference"
                                move(0);
                                System.out.println("move with a certain height
difference");
                                monitorForCollisions();
                        }
                }

                try {
                        Thread.sleep(500);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
        }
    }
    boolean monitor=false;
    public void monitorForCollisions() {
                while(!monitor) {
                        double
dis=UAVParam.uavCurrentData[n1].getUTMLocation().distance(UAVParam.uavCurr
entData[n2].getUTMLocation());
                        //restore altitude
                        if(dis>(safteyMarginSwarm1+safteyMarginSwarm2)*2)
{

    if(API.getCopter(numUAV).getMissionHelper().pause()) {
                                changeAltitude(missionAltitude);
                                //restore
                                move(1);
                                System.out.println("restore altitude");
```

```
                    }
                    monitor=true;
                }
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        public void changeAltitude(double altitude) {
            if(API.getCopter(numUAV).setFlightMode(FlightMode.GUIDED)) {
                MoveTo moveTo = API.getCopter(numUAV).moveTo(new
Location3D(API.getCopter(numUAV).getLocationGeo(), altitude), new
MoveToListener() {
                    public void onFailure() {
                    }
                    public void onCompleteActionPerformed() {
                    }
                });
                moveTo.start();
                System.out.println("change atti");
                try {
                    moveTo.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        public void drawCircle() {
            if(numUAV!=n1&&numUAV!=n2)return;
            int n = 0;double safteyMarginSwarm=0.0;
            if(numUAV==n1) {
                n=n1;
```

```
                safteyMarginSwarm=safteyMarginSwarm1;
            }else if(numUAV==n2) {
                n=n2;
                safteyMarginSwarm=safteyMarginSwarm2;
            }
            List<Location2DGeo> circles = new ArrayList<>();
            circles.add(UAVParam.uavCurrentData[n].getGeoLocation());
            DrawableCirclesGeo current =
ScpSimProperties.predictedLocation[n].getAndSet(null);
            if (current == null) {
                try {

        ScpSimProperties.predictedLocation[n].set(Mapper.Drawables.addCirclesGeo
(3, circles, safteyMarginSwarm,
                                        Color.BLACK, new BasicStroke(1f)));
                } catch (GUIMapPanelNotReadyException e) {
                    e.printStackTrace();
                }
            } else {
                try {
                    Mapper.Drawables.removeDrawable(current);
                } catch (GUIMapPanelNotReadyException e) {
                    e.printStackTrace();
                }
            }
        }
        public void move(double altitude) {
            if(API.getCopter(numUAV).setFlightMode(FlightMode.GUIDED)) {
                API.getGUI(numUAV).logUAV("Moving to WP: 1");
                if(altitude==0) {
                    altitude=newMissionAltitude;
                }else if(altitude==1) {
                    altitude=missionAltitude;
                }
```

```
                try {
                        Location2DUTM locInSwarm =
f.get2DUTMLocation(wp.getUTM(),index);
                        Location3D loc = new
Location3D(locInSwarm.getGeo(),altitude);
                        Thread t = API.getCopter(numUAV).moveTo(loc, new
MoveToListener() {
                                public void onCompleteActionPerformed() {
                                }
                                public void onFailure() {

                                }
                        });
                        t.start();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

    }

    public double calculateSafteyMargin(int index,Set<Integer> swarmIDs) {
        double max=0;
        Location2DUTM
p1=UAVParam.uavCurrentData[index].getUTMLocation();
        for(int numUAV:swarmIDs) {
            double
temp=p1.distance(UAVParam.uavCurrentData[numUAV].getUTMLocation());
            if(temp>max) {
                max=temp;
            }
        }
        return max+30;
    }
```

}

实 验 页 面 :

```
package com.protocols.scp.gui;

import com.api.API;
import com.api.ArduSimTools;
import com.api.formations.Formation;
import com.setup.Param;
import com.setup.Text;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TextField;
import javafx.scene.control.TextFormatter;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Properties;
import java.util.ResourceBundle;

public class ScpDialogController {

    private final ResourceBundle resources;
    private final Stage stage;
    private final ScpSimProperties properties;
```

```
        @FXML
        private TextField missionFile;
        @FXML
        private Button missionFileButton;
        @FXML
        private Button okButton;
        @FXML
        private TextField minDistance;
        @FXML
        private ChoiceBox<String> formation;

        public ScpDialogController(ResourceBundle resources, ScpSimProperties
properties, Stage stage){
            this.resources = resources;
            this.properties = properties;
            this.stage = stage;
        }

        @FXML
        public void initialize() {
            missionFile.setDisable(true);
            for(Formation.Layout l: Formation.Layout.values()){
                formation.getItems().add(l.name());
            }

formation.getSelectionModel().select(resources.getString("formation").toUpperCase()
);
            minDistance.setTextFormatter(new
TextFormatter<>(ArduSimTools.doubleFilter));

            missionFileButton.setOnAction(e -> searchMissionFile());
            okButton.setOnAction(e->{
                if(ok()){
                    Platform.setImplicitExit(false); // so that the application does not close
```

```
                Param.simStatus = Param.SimulatorState.STARTING_UAVS;
                okButton.getScene().getWindow().hide();
            }else{
                ArduSimTools.warnGlobal(Text.LOADING_ERROR,
Text.ERROR_LOADING_FXML);
            }
        });
    }

    private boolean ok(){
        Properties p = createProperties();
        return properties.storeParameters(p,resources);
    }

    private Properties createProperties(){
        Properties p = new Properties();
        Field[] variables = this.getClass().getDeclaredFields();
        for(Field var:variables){
            String annotation = var.getAnnotatedType().getType().getTypeName();
            if(annotation.contains("javafx")) {
                try {
                    Method getValue = null;
                    if (annotation.contains("TextField")) {
                        getValue = var.get(this).getClass().getMethod("getCharacters");
                    }
                    if(getValue != null) {
                        String value = String.valueOf(getValue.invoke(var.get(this)));
                        p.setProperty(var.getName(), value);
                    }
                } catch (IllegalAccessException | InvocationTargetException |
NoSuchMethodException e) {
                    e.printStackTrace();
                }
            }
```

```
        }
        return p;
    }

    private void searchMissionFile(){

        FileChooser fileChooser = new FileChooser();
        fileChooser.setInitialDirectory(new
File(API.getFileTools().getSourceFolder() + "/main/resources"));
        fileChooser.setTitle(Text.MISSIONS_DIALOG_TITLE_1);
        FileChooser.ExtensionFilter extFilterKML = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_1,
"*."+Text.FILE_EXTENSION_KML);
        FileChooser.ExtensionFilter extFilterWaypoints = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_2,
"*."+Text.FILE_EXTENSION_WAYPOINTS);
        fileChooser.getExtensionFilters().addAll(extFilterKML,extFilterWaypoints);

        List<File> missionPath = fileChooser.showOpenMultipleDialog(stage);
        if(missionPath != null && missionPath.size()>0) {
            String text = "";
            if(missionPath.size() > 1){
                for(File mission : missionPath){
                    text = text + mission.getAbsolutePath() + ";";
                }
            }else{
                text = missionPath.get(0).getAbsolutePath();
            }
            Path absolute = Paths.get(text);
            Path base = API.getFileTools().getResourceFolder();
            missionFile.setText(base.relativize(absolute).toString());
        }else{
            missionFile.setText("");
        }
```

```
    }

} package com.protocols.scp.gui;

import com.api.API;
import com.api.ArduSimTools;
import com.api.formations.Formation;
import com.setup.Param;
import com.setup.Text;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TextField;
import javafx.scene.control.TextFormatter;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Properties;
import java.util.ResourceBundle;

public class ScpDialogController {

    private final ResourceBundle resources;
    private final Stage stage;
    private final ScpSimProperties properties;
```

```
        @FXML
        private TextField missionFile;
        @FXML
        private Button missionFileButton;
        @FXML
        private Button okButton;
        @FXML
        private TextField minDistance;
        @FXML
        private ChoiceBox<String> formation;

        public ScpDialogController(ResourceBundle resources, ScpSimProperties
properties, Stage stage){
            this.resources = resources;
            this.properties = properties;
            this.stage = stage;
        }

        @FXML
        public void initialize() {
            missionFile.setDisable(true);
            for(Formation.Layout l: Formation.Layout.values()){
                formation.getItems().add(l.name());
            }

formation.getSelectionModel().select(resources.getString("formation").toUpperCase()
);
            minDistance.setTextFormatter(new
TextFormatter<>(ArduSimTools.doubleFilter));

            missionFileButton.setOnAction(e -> searchMissionFile());
            okButton.setOnAction(e->{
                if(ok()){
```

```
                Platform.setImplicitExit(false); // so that the application does not close
                Param.simStatus = Param.SimulatorState.STARTING_UAVS;
                okButton.getScene().getWindow().hide();
            }else{
                ArduSimTools.warnGlobal(Text.LOADING_ERROR,
Text.ERROR_LOADING_FXML);
            }
        });
    }

    private boolean ok(){
        Properties p = createProperties();
        return properties.storeParameters(p,resources);
    }

    private Properties createProperties(){
        Properties p = new Properties();
        Field[] variables = this.getClass().getDeclaredFields();
        for(Field var:variables){
            String annotation = var.getAnnotatedType().getType().getTypeName();
            if(annotation.contains("javafx")) {
                try {
                    Method getValue = null;
                    if (annotation.contains("TextField")) {
                        getValue = var.get(this).getClass().getMethod("getCharacters");
                    }
                    if(getValue != null) {
                        String value = String.valueOf(getValue.invoke(var.get(this)));
                        p.setProperty(var.getName(), value);
                    }
                } catch (IllegalAccessException | InvocationTargetException |
NoSuchMethodException e) {
                    e.printStackTrace();
                }
```

```
            }
        }
        return p;
    }

    private void searchMissionFile(){

        FileChooser fileChooser = new FileChooser();
        fileChooser.setInitialDirectory(new
File(API.getFileTools().getSourceFolder() + "/main/resources"));
        fileChooser.setTitle(Text.MISSIONS_DIALOG_TITLE_1);
        FileChooser.ExtensionFilter extFilterKML = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_1,
"*."+Text.FILE_EXTENSION_KML);
        FileChooser.ExtensionFilter extFilterWaypoints = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_2,
"*."+Text.FILE_EXTENSION_WAYPOINTS);
        fileChooser.getExtensionFilters().addAll(extFilterKML,extFilterWaypoints);

        List<File> missionPath = fileChooser.showOpenMultipleDialog(stage);
        if(missionPath != null && missionPath.size()>0) {
            String text = "";
            if(missionPath.size() > 1){
                for(File mission : missionPath){
                    text = text + mission.getAbsolutePath() + ";";
                }
            }else{
                text = missionPath.get(0).getAbsolutePath();
            }
            Path absolute = Paths.get(text);
            Path base = API.getFileTools().getResourceFolder();
            missionFile.setText(base.relativize(absolute).toString());
        }else{
            missionFile.setText("");
```

```
        }

    }

} package com.protocols.scp.gui;

import com.api.API;
import com.api.ArduSimTools;
import com.api.formations.Formation;
import com.setup.Param;
import com.setup.Text;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TextField;
import javafx.scene.control.TextFormatter;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Properties;
import java.util.ResourceBundle;

public class ScpDialogController {

    private final ResourceBundle resources;
    private final Stage stage;
```

```
    private final ScpSimProperties properties;

    @FXML
    private TextField missionFile;
    @FXML
    private Button missionFileButton;
    @FXML
    private Button okButton;
    @FXML
    private TextField minDistance;
    @FXML
    private ChoiceBox<String> formation;

    public ScpDialogController(ResourceBundle resources, ScpSimProperties
properties, Stage stage){
        this.resources = resources;
        this.properties = properties;
        this.stage = stage;
    }

    @FXML
    public void initialize() {
        missionFile.setDisable(true);
        for(Formation.Layout l: Formation.Layout.values()){
            formation.getItems().add(l.name());
        }

formation.getSelectionModel().select(resources.getString("formation").toUpperCase()
);
        minDistance.setTextFormatter(new
TextFormatter<>(ArduSimTools.doubleFilter));

        missionFileButton.setOnAction(e -> searchMissionFile());
        okButton.setOnAction(e->{
```

```
            if(ok()){
                Platform.setImplicitExit(false); // so that the application does not close
                Param.simStatus = Param.SimulatorState.STARTING_UAVS;
                okButton.getScene().getWindow().hide();
            }else{
                ArduSimTools.warnGlobal(Text.LOADING_ERROR,
Text.ERROR_LOADING_FXML);
            }
        });
    }

    private boolean ok(){
        Properties p = createProperties();
        return properties.storeParameters(p,resources);
    }

    private Properties createProperties(){
        Properties p = new Properties();
        Field[] variables = this.getClass().getDeclaredFields();
        for(Field var:variables){
            String annotation = var.getAnnotatedType().getType().getTypeName();
            if(annotation.contains("javafx")) {
                try {
                    Method getValue = null;
                    if (annotation.contains("TextField")) {
                        getValue = var.get(this).getClass().getMethod("getCharacters");
                    }
                    if(getValue != null) {
                        String value = String.valueOf(getValue.invoke(var.get(this)));
                        p.setProperty(var.getName(), value);
                    }
                } catch (IllegalAccessException | InvocationTargetException |
NoSuchMethodException e) {
                    e.printStackTrace();
```

```
                }
            }
        }
        return p;
    }

    private void searchMissionFile(){

        FileChooser fileChooser = new FileChooser();
        fileChooser.setInitialDirectory(new
File(API.getFileTools().getSourceFolder() + "/main/resources"));
        fileChooser.setTitle(Text.MISSIONS_DIALOG_TITLE_1);
        FileChooser.ExtensionFilter extFilterKML = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_1,
"*."+Text.FILE_EXTENSION_KML);
        FileChooser.ExtensionFilter extFilterWaypoints = new
FileChooser.ExtensionFilter(Text.MISSIONS_DIALOG_SELECTION_2,
"*."+Text.FILE_EXTENSION_WAYPOINTS);
        fileChooser.getExtensionFilters().addAll(extFilterKML,extFilterWaypoints);

        List<File> missionPath = fileChooser.showOpenMultipleDialog(stage);
        if(missionPath != null && missionPath.size()>0) {
            String text = "";
            if(missionPath.size() > 1){
                for(File mission : missionPath){
                    text = text + mission.getAbsolutePath() + ";";
                }
            }else{
                text = missionPath.get(0).getAbsolutePath();
            }
            Path absolute = Paths.get(text);
            Path base = API.getFileTools().getResourceFolder();
            missionFile.setText(base.relativize(absolute).toString());
        }else{
```

```
            missionFile.setText("");
        }

    }

}
```

Los datos del experimento

Global: Total time: 0:02:16 UAV 0: 0:02:03 UAV 1: 0:02:03 UAV 2: 0:02:04 UAV 3: 0:02:05 UAV 4: 0:02:11 UAV 5: 0:02:12 UAV 6: 0:02:13 UAV 7: 0:02:14 UAV 8: 0:02:15 UAV 9: 0:02:16 Simulation parameters: Number of UAVs: 10 Initial speed (m/s): 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0 Performance parameters: Screen refresh rate: 500 ms Minimum screen redraw distance: 5.0 pixels Enable arducopter logging: disabled Restrict battery capacity: no Measure CPU use: disabled General parameters: Enable verbose logging: disabled UAV synchronization protocol: SCP UAV to UAV communications parameters: Carrier sensing enabled: true Packet Collision detection enabled: true Receiving buffer size: 163840 bytes Wireless communications model: unrestricted Total sent packets: 0 UAV Collision detection parameters: Enable collision detection: no Wind: no Additional parameters (ardusim.ini): KMLMINALTITUDE=5.0 KMLOVERRIDEALTITUDE=false KMLMISSIONEND=unmodified KMLWAYPOINTDELAY=0 Swarm formation parameters: GROUNDFORMATION=CIRCLE GROUNDDISTANCE=10.0 AIRFORMATION=LINEAR AIRDISTANCE=50.0 LANDDISTANCE=2.5