HERRAMIENTA DE SIMULACIÓN 3D PARA PATRULLA AUTÓNOMA CON DRON EN OPERACIONES DE SALVAMENTO

3D SIMULATION TOOL FOR AUTONOMOUS DRONE PATROL IN RESCUE OPERATIONS

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTOR
JESÚS RAMOS RODRÍGUEZ

DIRECTORES
SERGIO BERNABÉ GARCÍA
SANDRA CATALÁN PALLARÉS

CALIFICACIÓN: 10.0

GRADO EN DESARROLLO DE VIDEOJUEGOS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

HERRAMIENTA DE SIMULACIÓN 3D PARA PATRULLA AUTÓNOMA CON DRON EN OPERACIONES DE SALVAMENTO

3D SIMULATION TOOL FOR AUTONOMOUS DRONE PATROL IN RESCUE OPERATIONS

TRABAJO DE FIN DE GRADO EN DESARROLLO DE VIDEOJUEGOS

AUTOR
JESÚS RAMOS RODRÍGUEZ

DIRECTORES
SERGIO BERNABÉ GARCÍA
SANDRA CATALÁN PALLARÉS

CONVOCATORIA: SEPTIEMBRE 2025

GRADO EN DESARROLLO DE VIDEOJUEGOS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

9 DE SEPTIEMBRE DE 2025

AGRADECIMIENTOS

A mis padres, por su apoyo y por el esfuerzo económico que me permitió estudiar en Madrid y hacer posible mi situación actual.

A todo el personal académico que me encontré por el camino, a quienes les apasiona su trabajo aún tras años de ejercicio profesional. Que hacen hueco para añadir un par de alumnos más a sus TFGs, que encuentran tiempo para responder un e-mail a deshora o para agendar una tutoría fuera de su jornada. Que enseñan con entusiasmo, profesionalidad, disciplina y empatía. Y que son quienes nos permiten recordar con alegría este intenso período años después. Vuestro esfuerzo no es en vano.

A Sergio Bernabé y Sandra Catalán, por formar parte de este grupo de profesorado y ayudarme a dar un gran y anticipado punto final a mi vida académica.

RESUMEN

Herramienta de simulación 3D para patrulla autónoma con dron en operaciones de salvamento

Los drones han adquirido en los últimos años un papel destacado en sectores muy diversos, desde la agricultura, la logística o el medio ambiente, hasta el ámbito policial y militar, donde se utilizan intensivamente en tareas de reconocimiento y ataque. Entre sus múltiples aplicaciones, la búsqueda y rescate es una de las más prometedoras, ya que permite reducir los tiempos de respuesta, acceder a zonas de difícil alcance y mejorar notablemente la eficiencia y el coste operativo frente a los métodos convencionales utilizados hasta ahora.

Este Trabajo de Fin de Grado (TFG) presenta una aplicación desarrollada en Unity que, mediante la integración del simulador DroneKit-SITL, simula el vuelo autónomo de un dron equipado con cámara infrarroja para la localización de personas desaparecidas en mar o tierra. El usuario puede seleccionar cualquier zona del mundo y definir un área a patrullar. El sistema genera automáticamente una ruta eficiente, gestiona el consumo de batería, realiza recargas cuando es necesario, y retoma la misión hasta completarla, notificando en el proceso las fuentes de calor detectadas. La herramienta está pensada como un medio accesible orientada a cuerpos de rescate y demuestra el potencial de los drones en operaciones de salvamento.

Palabras clave

Dron, generación de rutas, algoritmo de patrullaje, simulación, búsqueda y rescate, Unity, DroneKit-SITL, vuelo autónomo, localización de personas.

ABSTRACT

3D Simulation Tool for Autonomous Drone Patrol in Rescue Operations

In recent years, drones have taken on a prominent role in a wide range of sectors, from agriculture, logistics, and the environment to law enforcement and the military, where they are used extensively for reconnaissance and attack missions. Among their many applications, search and rescue is one of the most promising, as it reduces response times, provides access to hard-to-reach areas, and significantly improves efficiency and operating costs compared to conventional methods used to date.

This Final Degree Project presents an application developed in Unity which, through the integration of the DroneKit-SITL simulator, is used to simulate the autonomous flight of a drone equipped with an infrared camera for locating missing persons both at sea and on land. The user can select any area of the world and define an area to patrol. The system automatically generates an efficient route, manages battery consumption, recharges when necessary, and resumes the mission until it is completed, notifying the user of any heat sources detected in the process. The tool is designed as an accessible resource for rescue teams and demonstrates the potential of drones in rescue operations.

Keywords

Drone, route generation, patrol algorithm, simulation, search and rescue, Unity, DroneKit-SITL, autonomous flight, people location.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Contexto y antecedentes

El avance tecnológico de las últimas décadas ha transformado profundamente la manera en que las sociedades enfrentan desafíos en sectores tan diversos como la agricultura, la vigilancia, la logística o la seguridad. En este contexto de innovación constante, los drones, también conocidos como vehículos aéreos no tripulados (UAV, del inglés Unmanned Aerial Vehicles), han pasado de ser herramientas experimentales a convertirse en soluciones prácticas, económicas y cada vez más sofisticadas. Su capacidad para operar de forma remota, acceder a zonas de difícil alcance y ofrecer una visión aérea en tiempo real ha impulsado su adopción en ámbitos tanto civiles como militares, donde ya forman parte habitual de estrategias operativas.

En el ámbito militar, los drones desempeñan un papel clave en tareas de reconocimiento, vigilancia y ataques de precisión. Más de 90 países han integrado estos dispositivos en sus operaciones, y se estima que el mercado global de drones militares superó los 36.000 millones de dólares en 2023, y ha continuado con crecimiento sostenido hasta la actualidad [1], [2]. En paralelo, el mercado civil continúa expandiéndose, con proyecciones que sitúan su valor por encima de los 120.000 millones para 2033, impulsado por aplicaciones como la cartografía, la agricultura de precisión, la inspección industrial y la respuesta ante emergencias [3].

Una de las aplicaciones con mayor proyección social es la búsqueda y rescate de personas, especialmente en situaciones de catástrofe o en zonas de difícil acceso. El uso de drones permite reducir significativamente los tiempos de respuesta y los costes operativos, al tiempo que minimiza el riesgo para los equipos humanos. Diversos estudios han demostrado que, en escenarios simulados, los drones han logrado localizar víctimas en menos tiempo que los métodos tradicionales, mejorando notablemente la eficiencia de las operaciones [4].

Además, el desarrollo de tecnologías complementarias, como sensores térmicos, cámaras infrarrojas, sistemas de navegación autónoma o técnicas de aprendizaje automático, está permitiendo diseñar drones cada vez más inteligentes y capaces de

adaptarse a diferentes entornos. En el ámbito académico, existen trabajos que aplican algoritmos de planificación de rutas, aprendizaje por refuerzo y redes neuronales para optimizar patrullajes en escenarios de búsqueda, logrando mejoras significativas en tiempos de cobertura y precisión [5].

Además de las fuentes académicas y profesionales mencionadas, durante el desarrollo de este Trabajo de Fin de Grado se consultaron diversos estudios y proyectos previos que guardan relación directa con la temática abordada. Estos trabajos han resultado útiles tanto para comprender el estado actual de la investigación en planificación de rutas y simulación de UAVs, como para identificar buenas prácticas.

En particular, cabe destacar:

- Simulation of Autonomous Flight of UAVs for Search and Rescue Missions. Alejandro-Daniel Díaz Román. [6].
- UAV-Based Forest Fire Patrol Path Planning Strategy. Yiqing Xu , Jiaming Li and Fuquan Zhang. [7].
- Energy-Efficient UAVs Coverage Path Planning Approach. Gamil Ahmed, Tarek Sheltami, Ashraf Mahmoud, Ansar Yasar. [8].
- Early Forest Fire Detection Using Drones and Artificial Intelligence. Diyana Kinaneva, Georgi Hristov, Jordan Raychev and Plamen Zahariev. [9].

1.2 Motivación

Las operaciones de búsqueda y rescate tradicionales se basan en recursos humanos y medios costosos, como helicópteros, embarcaciones o patrullas terrestres. Aunque efectivos, estos métodos presentan limitaciones: son lentos, peligrosos y, en muchos casos, ineficientes en grandes áreas o condiciones adversas. Los drones, por el contrario, ofrecen una alternativa más ágil, segura y económica, que permite intervenir rápidamente y sin poner en riesgo vidas humanas.

Su adopción en entornos reales va en aumento: se estima que más del 80 % de los operadores profesionales emplean drones en labores de búsqueda, rescate o gestión de emergencias [10]. Sin embargo, muchas soluciones tecnológicas actuales están pensadas para usuarios con formación técnica, lo que dificulta su implantación generalizada en equipos de intervención que no siempre disponen de esos conocimientos.

Este Trabajo de Fin de Grado nace con el objetivo de aportar en dos frentes. Por un lado, se plantea el desarrollo de un algoritmo capaz de generar patrullas eficientes que cubran áreas designadas en el menor tiempo posible. Por otro, se persigue demostrar que este tipo de sistemas pueden presentarse mediante herramientas accesibles, usables y aplicables a la realidad operativa, con una interfaz intuitiva y una experiencia centrada en el usuario. La combinación de eficiencia técnica y facilidad de uso busca favorecer la incorporación real de estas tecnologías en situaciones de emergencia.

1.3 Objetivos del proyecto

El objetivo general de este Trabajo de Fin de Grado es desarrollar una aplicación software que permita gestionar de forma realista y accesible la operación de un dron en escenarios simulados de búsqueda de una persona desaparecida. Para ello, se plantea el diseño de una interfaz intuitiva que facilite la configuración de parámetros clave del vuelo, la selección de zonas geográficas reales mediante coordenadas de latitud y longitud, y la visualización detallada de la misión en un entorno tridimensional. La finalidad última es demostrar que este tipo de herramientas pueden ser aplicadas de manera efectiva en situaciones reales de emergencia, combinando precisión técnica, automatización y facilidad de uso para operadores no expertos.

A partir de este objetivo general, se definen los siguientes objetivos específicos:

- **Permitir la selección de zonas geográficas reales** mediante la introducción de coordenadas de latitud y longitud, lo que ofrece flexibilidad total al usuario para definir cualquier ubicación del planeta como área de patrullaje.

- **Desarrollar un algoritmo de patrullaje eficiente** que genere rutas óptimas capaces de cubrir la totalidad del área designada, minimizando recorridos redundantes y optimizando el tiempo de misión.
- **Implementar una interfaz de usuario accesible y funcional**, que permita configurar parámetros clave del dron como la velocidad, así como visualizar en tiempo real datos como la altura, velocidad actual, nivel, consumo de batería y ETA (del inglés Estimated Time of Arrival).
- **Ofrecer una visualización 3D inmersiva** del entorno, que permita al usuario observar el vuelo del dron desde distintas perspectivas, incluyendo la posición de despegue, la trayectoria y los cambios de comportamiento autónomo durante la misión.
- **Integrar un sistema de mapeo térmico** que represente gráficamente las zonas ya cubiertas por el dron mediante un mapa de calor, así como un indicador de progreso porcentual que informe del grado de cobertura alcanzado respecto al área total.
- **Simular el comportamiento autónomo del dron** respecto a la gestión energética: monitorizando el nivel de batería, activando el retorno automático a una base predefinida para su recarga y reanudando la patrulla de forma automática una vez completada la recarga.
- **Permitir el envío de instrucciones manuales** desde la interfaz durante el transcurso de la misión, como forzar el retorno a la base en cualquier momento, independientemente del punto en el que se encuentre el dron, modificar la base del dron, o cargar una nueva zona de patrullaje que reemplace la anterior, adaptando la misión dinámicamente.
- **Incorporar un sistema de detección simulada** de personas desaparecidas mediante una cámara infrarroja virtual representada como una pirámide de visión descendente, capaz de identificar fuentes de calor durante el patrullaje.

1.4 Plan de trabajo

El desarrollo del proyecto se organizó en varias fases secuenciales, cada una con objetivos concretos que permitieron avanzar de forma estructurada hacia la versión final de la aplicación. A continuación se describen las principales etapas del trabajo:

- **Investigación preliminar:**
  - Estudio del uso de drones en tareas de búsqueda y rescate.
  - Análisis de sus limitaciones técnicas (autonomía, cobertura, precisión).
  - Evaluación comparativa de simuladores gratuitos disponibles.
  - Selección de DroneKit con ArduPilot-SITL (Software In The Loop) como simulador base por su realismo y compatibilidad.
- **Prueba de integración inicial:**
  - Desarrollo de una demo mínima para validar la conexión entre:
    - Unity (motor gráfico y entorno visual),
    - DroneKit-SITL (simulador del dron),
    - Servidor WebSocket (puente de comunicación entre ambos).
  - Verificación de transmisión de datos de telemetría y comandos.
- **Visualización y entorno gráfico:**
  - Implementación del sistema de mapas con datos cartográficos reales.
  - Control de cámara 3D y navegación libre por el escenario.
  - Representación visual del dron a partir de su telemetría (posición, orientación, etc.).
- **Diseño y desarrollo del algoritmo de patrullaje:**
  - Diseño del sistema que genera rutas óptimas para cubrir áreas completas.
  - Minimización de recorridos redundantes y maximización de eficiencia.
  - Ajuste del algoritmo para adaptarse dinámicamente a zonas nuevas.

- **Simulación y pruebas de vuelo:**
    - Validación del algoritmo en distintos terrenos simulados.
    - Evaluación de la cobertura y comportamiento autónomo del dron.
- **Diseño de la interfaz e interacción con el usuario:**
    - Selección inicial de la zona del mundo a patrullar mediante coordenadas de latitud y longitud.
    - Definición del área concreta de patrullaje mediante nodos sobre el mapa interactivo.
    - Configuración de variables de velocidad del dron y posición de la base de despegue.
    - Mapa de calor del recorrido y barra de progreso de la patrulla.
    - Panel informativo sobre los parámetros del dron.
- **Automatización del comportamiento del dron:**
    - Gestión de batería: retorno automático a base y recarga simulada.
    - Reanudación automática de la patrulla tras la recarga.
    - Posibilidad de enviar instrucciones manuales desde la interfaz:
        - Forzar regreso a base.
        - Modificar base.
        - Cambiar zona de patrullaje o cargar un nuevo recorrido.
        - Modificar la velocidad del dron en tiempo real.
- **Simulación de detección de personas:**
    - Implementación de un sistema de detección de fuentes de calor.
    - Uso de una cámara infrarroja simulada con un cono de visión descendente.
- **Validación y pulido final:**
    - Revisión funcional de todos los componentes.

- Mejora de la experiencia de usuario y del rendimiento general.
- Comprobación de estabilidad, accesibilidad y aplicabilidad real.

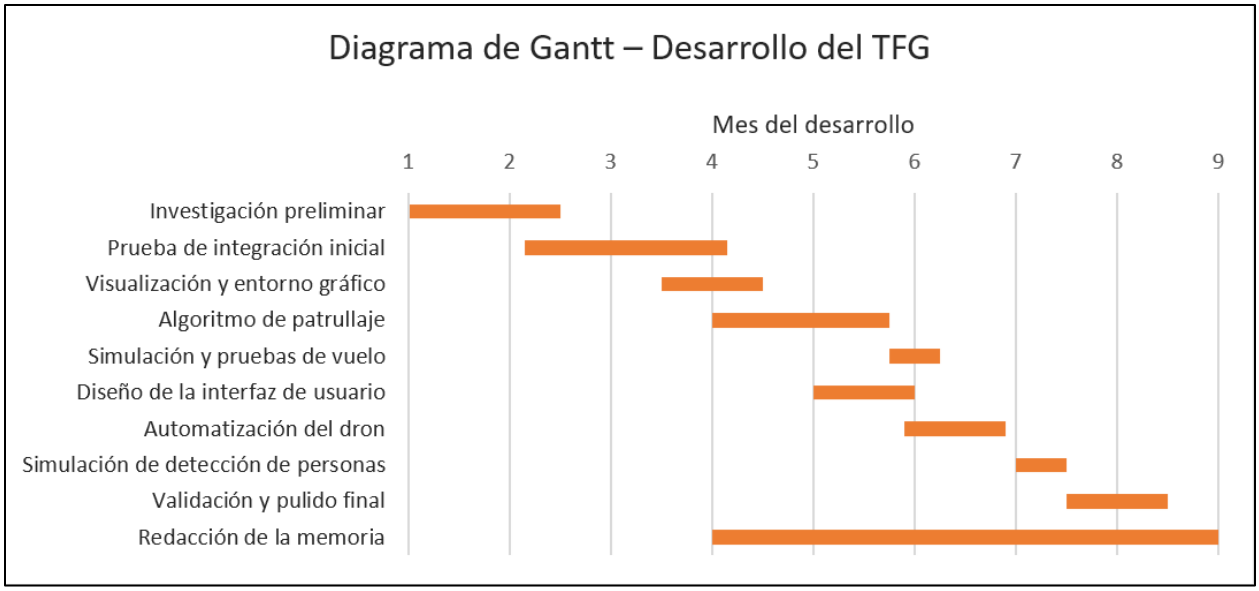

Figura 1-1. Diagrama de Gantt correspondiente al desarrollo del proyecto.

1.5 Estructura del documento

El documento se organiza en capítulos que siguen un hilo lógico desde la motivación y el contexto hasta los detalles de implementación y los recursos anexos:

- Capítulos 1 y 2 - Introducción: presenta el contexto y antecedentes del uso de drones en salvamento, la motivación del trabajo, los objetivos general y específicos, el plan de trabajo y esta estructura del documento.
- Capítulo 3 - Estado del arte: revisa la literatura relevante en los cuatro pilares del proyecto: planificación de rutas, uso de UAV en misiones SAR, detección térmica con cámaras infrarrojas y diseño de interfaces accesibles. Cierra con un resumen crítico que delimita el nicho que aborda la herramienta propuesta.
- Capítulo 4 - Arquitectura del sistema: describe la solución en tres bloques: la aplicación en Unity con Mapbox, la simulación DroneKit junto con ArduPilot-SITL y el servidor WebSocket. Además, detalla la integración y el flujo de datos. Incluye

la justificación de diseño, los problemas encontrados y sus soluciones, así como las limitaciones técnicas derivadas de las decisiones tomadas.

- Capítulo 5 - Algoritmo de patrullaje: explica con detalle el procedimiento de cobertura basado en offsets interiores cosidos en trayectoria boustrophedon, el postprocesado geométrico/altimétrico, la exportación a misión compatible con DroneKit, una comparación con alternativas y ejemplos de generación de rutas en geometrías diversas.
- Capítulo 6 - Simulación y entorno operativo: este capítulo aborda tres aspectos fundamentales del sistema: el modelo energético y de batería del servidor, con su telemetría e integración en la interfaz; la detección simulada de personas mediante visión infrarroja y lógica de notificación; y la interfaz de usuario, que incluye paneles, mapa 3D, modos de cámara y criterios de usabilidad y accesibilidad.
- Capítulo 7 - Resultados, discusión y conclusiones: presenta los resultados obtenidos en diferentes escenarios de validación, los analiza de forma crítica en relación con los objetivos planteados y las limitaciones identificadas, y concluye con un balance final del proyecto y posibles líneas de trabajo futuras.
- Bibliografía: reúne las referencias académicas y técnicas citadas a lo largo del texto.
- Apéndices: incluyen los scripts y componentes de apoyo referenciados en la memoria (p. ej., DroneWSClient.cs y ZoneSelector.cs), útiles para extender la implementación. Asimismo, se añaden los enlaces al repositorio en GitHub que contiene el código del proyecto, junto con otro enlace a un vídeo demostrativo que muestra la ejecución de la aplicación y el flujo de una misión simulada.

Esta organización permite que el lector avance desde el porqué y el qué del proyecto hasta el cómo se ha construido y validado la herramienta, dejando al final los recursos prácticos para su consulta.

Capítulo 2 - Introduction

2.1 Context and background

Technological advances in recent decades have profoundly transformed the way societies address challenges in sectors as diverse as agriculture, surveillance, logistics, and security. In this context of constant innovation, drones, also known as unmanned aerial vehicles (UAVs), have gone from being experimental tools to becoming practical, economical, and increasingly sophisticated solutions. Their ability to operate remotely, access hard-to-reach areas, and provide real-time aerial views has driven their adoption in both civilian and military fields, where they are now a regular part of operational strategies.

In the military sphere, drones play a key role in reconnaissance, surveillance, and precision strikes. More than 90 countries have integrated these devices into their operations, and it is estimated that the global market for military drones exceeded $36 billion in 2023 and has continued to grow steadily to date [1], [2]. At the same time, the civilian market continues to expand, with projections placing its value at over $120 billion by 2033, driven by applications such as mapping, precision agriculture, industrial inspection, and emergency response [3].

One of the applications with the greatest social impact is search and rescue, especially in disaster situations or in areas that are difficult to access. The use of drones significantly reduces response times and operating costs, while minimizing the risk to human teams. Several studies have shown that, in simulated scenarios, drones have been able to locate victims in less time than traditional methods, significantly improving the efficiency of operations [4].

In addition, the development of complementary technologies, such as thermal sensors, infrared cameras, autonomous navigation systems, and machine learning techniques, is enabling the design of increasingly intelligent drones capable of adapting to different environments. In the academic field, there are studies that apply route planning algorithms, reinforcement learning, and neural networks to optimize patrols in

search scenarios, achieving significant improvements in coverage times and accuracy [5].

In addition to the academic and professional sources mentioned above, during the development of this Final Degree Project, various previous studies and projects directly related to the topic addressed were consulted. These works have been useful both for understanding the current state of research in route planning and UAV simulation and for identifying best practices.

In particular, the following are noteworthy:

- Simulation of Autonomous Flight of UAVs for Search and Rescue Missions. Alejandro-Daniel Díaz Román. [6].
- UAV-Based Forest Fire Patrol Path Planning Strategy. Yiqing Xu , Jiaming Li and Fuquan Zhang. [7].
- Energy-Efficient UAVs Coverage Path Planning Approach. Gamil Ahmed, Tarek Sheltami, Ashraf Mahmoud, Ansar Yasar. [8].
- Early Forest Fire Detection Using Drones and Artificial Intelligence. Diyana Kinaneva, Georgi Hristov, Jordan Raychev and Plamen Zahariev. [9].

2.2 Motivation

Traditional search and rescue operations rely on human resources and costly equipment such as helicopters, boats, and ground patrols. Although effective, these methods have limitations: they are slow, dangerous, and, in many cases, inefficient in large areas or adverse conditions. Drones, on the other hand, offer a more agile, safer, and more economical alternative, allowing for rapid intervention without putting human lives at risk.

Their adoption in real-world environments is on the rise: it is estimated that more than 80% of professional operators use drones in search and rescue or emergency management tasks [10]. However, many current technological solutions are designed for

users with technical training, which hinders their widespread implementation in intervention teams that do not always have such knowledge.

This Final Degree Project aims to contribute on two fronts. On the one hand, it proposes the development of an algorithm capable of generating efficient patrols that cover designated areas in the shortest possible time. On the other hand, it seeks to demonstrate that these types of systems can be presented using accessible, usable tools that are applicable to operational reality, with an intuitive interface and a user-centered experience. The combination of technical efficiency and ease of use seeks to promote the real incorporation of these technologies in emergency situations.

2.3 Project objectives

The overall objective of this Final Degree Project is to develop a software application that allows for the realistic and accessible management of drone operations in simulated scenarios involving the search for a missing person. To this end, the design of an intuitive interface is proposed that facilitates the configuration of key flight parameters, the selection of real geographical areas using latitude and longitude coordinates, and the detailed visualization of the mission in a three-dimensional environment. The ultimate goal is to demonstrate that this type of tool can be effectively applied in real emergency situations, combining technical precision, automation, and ease of use for non-expert operators.

Based on this general objective, the following specific objectives are defined:

- **Enable the selection of real geographical areas** by entering latitude and longitude coordinates, offering users complete flexibility to define any location on the planet as a patrol area.
- **Develop an efficient patrol algorithm** that generates optimal routes capable of covering the entire designated area, minimizing redundant routes and optimizing mission time.
- **Implement an accessible and functional user interface** that allows key drone parameters such as speed to be configured, as well as real-time data such as

altitude, current speed, level, battery consumption, and ETA (Estimated Time of Arrival) to be displayed.

- **Provide an immersive 3D visualization of the environment**, allowing the user to observe the drone's flight from different perspectives, including takeoff position, trajectory, and changes in autonomous behavior during the mission.
- **Integrate a thermal mapping system** that graphically represents the areas already covered by the drone using a heat map, as well as a percentage progress indicator that reports the degree of coverage achieved in relation to the total area.
- **Simulate the drone's autonomous behavior with regard to energy management**: monitoring the battery level, activating automatic return to a predefined base for recharging, and automatically resuming patrol once recharging is complete.
- **Allow manual instructions to be sent** from the interface during the course of the mission, such as forcing a return to base at any time, regardless of the drone's location, modifying the drone's base, or loading a new patrol area to replace the previous one, dynamically adapting the mission.
- **Incorporate a simulated missing person detection system** using a virtual infrared camera represented as a downward-facing pyramid, capable of identifying heat sources during patrols.

2.4 Work plan

The project was organized into several sequential phases, each with specific objectives that allowed for structured progress toward the final version of the application. The main stages of the work are described below:

- **Preliminary research:**
    - Study of the use of drones in search and rescue tasks.
    - Analysis of their technical limitations (autonomy, coverage, accuracy).
    - Comparative evaluation of available free simulators.

- Selection of DroneKit with ArduPilot-SITL (Software In The Loop) as the base simulator due to its realism and compatibility.

- **Initial integration test:**
  - Development of a minimal demo to validate the connection between:
    - Unity (graphics engine and visual environment),
    - DroneKit-SITL (drone simulator),
    - WebSocket server (communication bridge between the two).
  - Verification of telemetry data and command transmission.
- **Visualization and graphical environment:**
  - Implementation of the mapping system with real cartographic data.
  - 3D camera control and free navigation through the scene.
  - Visual representation of the drone based on its telemetry (position, orientation, etc.).
- **Design and development of the patrol algorithm:**
  - Design of the system that generates optimal routes to cover entire areas.
  - Minimization of redundant routes and maximization of efficiency.
  - Adjustment of the algorithm to dynamically adapt to new areas.
- **Simulation and flight testing:**
  - Validation of the algorithm on different simulated terrains.
  - Evaluation of the drone's coverage and autonomous behavior.
- **Interface design and user interaction:**
  - Initial selection of the area of the world to be patrolled using latitude and longitude coordinates.
  - Definition of the specific patrol area using nodes on the interactive map.
  - Configuration of drone speed variables and takeoff base position.

- Heat map of the route and patrol progress bar.
- Information panel on drone parameters.

- **Automation of drone behavior:**
  - Battery management: automatic return to base and simulated recharging.
  - Automatic resumption of patrol after recharging.
  - Possibility of sending manual instructions from the interface:
    - Force return to base.
    - Modify base.
    - Change patrol area or load a new route.
    - Modify drone speed in real time.
- **Simulation of people detection:**
  - Implementation of a heat source detection system.
  - Use of a simulated infrared camera with a downward viewing cone.
- **Validation and final polishing:**
  - Functional review of all components.
  - Improvement of user experience and overall performance.
  - Checking stability, accessibility, and real applicability.

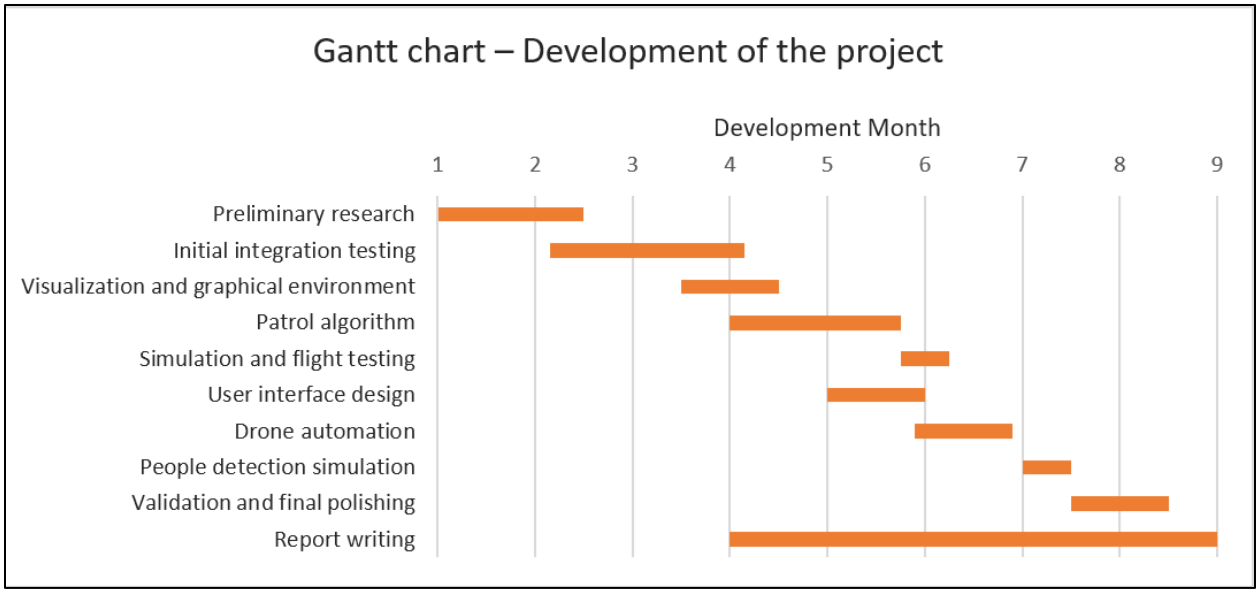

Figura 2-1. Gannt chart of the project development.

2.5 Document structure

The document is organized into chapters that follow a logical thread from motivation and context to implementation details and attached resources:

- Chapters 1 and 2 - Introduction: presents the context and background of the use of drones in rescue operations, the motivation for the work, the general and specific objectives, the work plan, and the structure of this document.
- Chapter 3 - State of the art: reviews the relevant literature on the four pillars of the project: route planning, use of UAVs in SAR missions, thermal detection with infrared cameras, and accessible interface design. It concludes with a critical summary that defines the niche addressed by the proposed tool.
- Chapter 4 - System architecture: Describe the solution in three blocks: the Unity application with Mapbox, the DroneKit simulation together with ArduPilot-SITL, and the WebSocket server. In addition, detail the integration and data flow. Include the design rationale, problems encountered and their solutions, as well as technical limitations resulting from the decisions made.
- Chapter 5 - Patrol algorithm: explains in detail the coverage procedure based on interior offsets stitched in a boustrophedon trajectory, geometric/altimetric post-

processing, export to a DroneKit-compatible mission, a comparison with alternatives, and examples of route generation in various geometries.

- Chapter 6 - Simulation and operating environment: this chapter addresses three fundamental aspects of the system: the server's energy and battery model, with its telemetry and integration into the interface; simulated detection of people using infrared vision and notification logic; and the user interface, which includes panels, a 3D map, camera modes, and usability and accessibility criteria.
- Chapter 7 - Results, discussion, and conclusions: This chapter presents the results obtained in different validation scenarios, critically analyzes them in relation to the objectives set and the limitations identified, and concludes with a final assessment of the project and possible future lines of work.
- Bibliography: This chapter brings together the academic and technical references cited throughout the text.
- Appendices: Include the scripts and support components referenced in the report (e.g., DroneWSClient.cs and ZoneSelector.cs), which are useful for extending the implementation. Links to the GitHub repository containing the project code are also provided, along with another link to a demonstration video showing the application in action and the flow of a simulated mission.

This organization allows the reader to progress from the why and what of the project to how the tool was built and validated, leaving the practical resources for consultation at the end.

Capítulo 3 - Estado del arte

3.1 Introducción

En este capítulo se presenta una revisión de la literatura y las tendencias actuales en los cuatro pilares que sustentan este trabajo:

- Planificación de rutas para drones.
- Uso de UAV en misiones de búsqueda y rescate (SAR, del inglés Search and Rescue).
- Detección térmica mediante cámaras infrarrojas.
- Diseño de interfaces accesibles para operarios no expertos.

3.2 Planificación de rutas (algoritmos de patrullaje)

En la actualidad, los algoritmos de planificación se clasifican en tres grandes categorías: métodos clásicos (como A* o RRT), técnicas bioinspiradas o evolutivas y enfoques fundamentados en el aprendizaje profundo. Una reciente revisión sobre detección aérea de personas sintetiza estas familias y destaca la transición hacia soluciones de inteligencia artificial híbrida, capaces de combinar la eficiencia de los planificadores clásicos con la adaptabilidad de modelos de aprendizaje profundo [11].

Entre los avances más recientes sobresalen:

- AgilePilot, un planificador entrenado con deep reinforcement learning que mantiene planificación en tiempo real y seguridad frente a obstáculos dinámicos [12].
- FM-Planner, que integra foundation models (LLM/VLM) con percepción visual, logrando una navegación semántica y visualmente razonada en entornos complejos [13].

Estas líneas de investigación ilustran la tendencia a explotar aprendizaje profundo y modelos fundacionales para obtener trayectorias más robustas y adaptativas, aunque su integración en herramientas operativas accesibles sigue siendo limitada.

3.3 Uso de drones en búsqueda y rescate (SAR)

El despliegue de drones en emergencias ha crecido notablemente. Casos recientes, como el rescate de un hombre de 78 años en Malibú gracias a un dron con cámara térmica, muestran su efectividad para reducir tiempos de respuesta en condiciones adversas [14].

En el plano investigador, plataformas cooperativas como AutoSOS combinan varios UAV con IA ligera y computación en el borde para cubrir amplias zonas marítimas de forma coordinada [15]. Estas iniciativas confirman que los UAV son ya una herramienta estratégica en SAR, aunque la mayoría de soluciones siguen requiriendo operadores expertos y entornos de control especializados.

3.4 Drones utilizados en misiones de búsqueda y rescate

Tras consolidarse como herramienta estratégica en operaciones SAR, los drones se han diversificado en distintas tipologías según el entorno y los requerimientos de la misión. Desde multirrotores portátiles de despliegue inmediato hasta plataformas de ala fija para grandes coberturas o sistemas cautivos para vigilancia continua, cada categoría ofrece ventajas específicas que amplían las capacidades de búsqueda y rescate.

Multirrotores portátiles

Pequeños, fáciles de transportar y rápidos de desplegar. Útiles para la primera respuesta en campo.

- DJI Mavic 3 Thermal (M3T): 45 min, térmica 640×512, zoom híbrido 56× [16].
- Autel EVO II Dual 640T V3: 38 min, cámara dual RGB + térmica 640×512, opción costo-efectiva [17].

Multirrotores industriales

Más robustos y modulares, con sensores de alto rendimiento. Adecuados para operaciones prolongadas y condiciones adversas.

- DJI Matrice 30T: 41 min, cámara híbrida con térmica 640×512, telémetro láser [18].
- Skydio X10: 40 min, térmica Boson+ 640×512, autonomía avanzada con navegación 360° [19].

Ala fija/VTOL

Diseñados para cubrir grandes áreas con autonomía extendida, ideales en entornos rurales o marítimos.

- Autel Dragonfish: hasta 120 min, carga con térmica 640×512 [20].
- senseFly eBee X + Duet T: hasta 90 min, cámara dual RGB+térmica [21].

Sistemas tethered (cautivos)

Operan conectados por cable, lo que permite vuelos continuos durante horas o días, usados en vigilancia estática.

- Fotokite Sigma: 24+ h de operación, cámara dual EO/IR [22].
- Elistair ORION 2: hasta 24–50 h, cargas ISR intercambiables [23].

3.5 Detección térmica mediante cámaras infrarrojas

La literatura demuestra que las cámaras infrarrojas montadas en drones pueden localizar fuentes de calor humanas incluso de noche o con visibilidad reducida [24]. Investigaciones recientes optimizan modelos tipo YOLO para datos IR, aumentando la precisión en detección y seguimiento en tiempo real [25]. También se han planteado variantes ligeras como LRI-YOLO, diseñadas para funcionar en dispositivos de baja potencia, típicos en UAVs, demostrando alta eficiencia en detección en el dominio IR [26].

Desde el punto de vista tecnológico, la termografía permite "ver" variaciones térmicas en el entorno sin necesidad de iluminación visible, ya que los objetos más

cálidos, como el cuerpo human, destacan claramente sobre fondos más fríos, lo que facilita su identificación visual y automática [27].

3.6 Interfaces accesibles y visualización

No obstante, aunque existen aplicaciones recientes que buscan reducir la complejidad de operación, como Lifeseeker, empleado en búsqueda y rescate con una interfaz web accesible para equipos no especializados [28], o Responder de BRINC, que permite control remoto simplificado en escenarios de seguridad pública [29], siguen siendo escasas las propuestas que integran de forma conjunta visualización 3D, mapas de calor del recorrido y métricas de progreso claras. Plataformas como FlytBase, orientada a seguridad pública y gestión multi-dron [30], o proyectos académicos como RescueAR, basado en realidad aumentada para mejorar la cognición espacial en operaciones de emergencia [31], e ICARUS, que integra control desde una aplicación Android para misiones de búsqueda y rescate [32], demuestran un interés creciente en la accesibilidad y la simplificación de interfaces. Sin embargo, aún no consolidan una solución integral, lo que refuerza la identificación de un nicho para interfaces orientadas a la usabilidad operativa, donde se enmarca la presente propuesta.

3.7 Resumen

La revisión del estado del arte muestra una evolución acelerada en el desarrollo de algoritmos de planificación y en el despliegue de drones para operaciones de búsqueda y rescate. Los avances recientes en aprendizaje profundo, modelos fundacionales y visión artificial han demostrado resultados prometedores, pero en muchos casos dependen de recursos computacionales elevados, lo que puede limitar su implementación en drones ligeros o con restricciones energéticas.

Por otro lado, se observa un vacío en soluciones que combinen algoritmos de bajo coste computacional con interfaces accesibles para usuarios no técnicos. En este contexto, el uso de algoritmos matemáticos más simples —como los que no requieren aprendizaje automático ni procesamiento intensivo— representa una línea de trabajo

válida, al permitir implementaciones más eficientes energéticamente y adaptables a plataformas con capacidades reducidas, como drones pequeños o de bajo consumo.

Asimismo, las interfaces accesibles con visualización clara del progreso, mapeo del recorrido y control directo desde el entorno gráfico siguen siendo escasas, lo que refuerza la necesidad de desarrollar herramientas más usables y aplicables a la realidad operativa de los cuerpos de emergencia.

Capítulo 4 - Arquitectura del sistema

El sistema desarrollado se organiza en torno a una arquitectura formada por tres componentes principales: el motor gráfico Unity, que actúa como núcleo visual y de interacción; el stack de simulación compuesto por DroneKit y ArduPilot-SITL, encargado de reproducir el comportamiento físico y los modos de vuelo del dron; y un servidor de comunicación basado en WebSocket, que hace posible el intercambio de datos entre ambos. Esta estructura busca equilibrar realismo y control, permitiendo representar de manera visual las operaciones del dron mientras se mantiene la fidelidad de una simulación de vuelo.

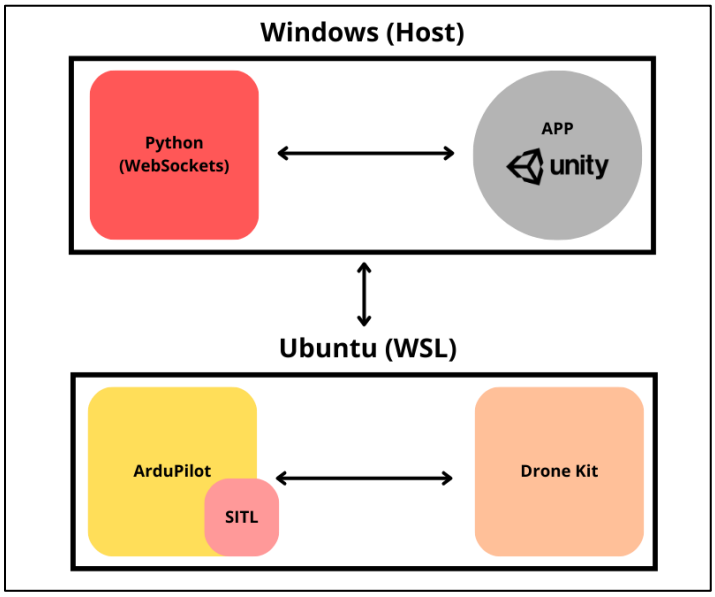

Figura 4-1. Esquema general de la arquitectura del sistema

4.1 Descripción general

Los tres componentes interactúan de forma bidireccional: ArduPilot-SITL, a través de DroneKit, genera datos de telemetría que son transmitidos al servidor y, desde allí, enviados a la aplicación de Unity para su representación. De manera recíproca, las órdenes del usuario introducidas en la aplicación se transmiten a través del servidor hasta DroneKit-SITL. El resultado es un flujo constante de información que mantiene sincronizadas la simulación física y la visualización en tiempo real.

4.2 Aplicación (motor gráfico y lógica principal)

La aplicación desarrollada en Unity constituye el núcleo del sistema, ya que concentra tanto la representación visual como la lógica general de la misión. En el plano gráfico, se apoya en el plugin Mapbox para Unity [33], que permite generar escenarios tridimensionales basados en datos cartográficos reales, con alturas y relieves precisos.

En paralelo, la aplicación integra los principales módulos de lógica:

- Algoritmo de generación de patrullas, encargado de calcular recorridos óptimos para cubrir el área definida.
- Gestión de la misión, que controla el estado del dron y coordina los eventos del vuelo.
- Procesos de apoyo a la autonomía, como el control del retorno a base, la reanudación tras la recarga y la adaptación dinámica del recorrido.

La aplicación también centraliza la interfaz de usuario, que ofrece tanto información en tiempo real como herramientas de control (véase Figura 4-2):

- Panel informativo de vuelo (I): altura, velocidad, ETA, nivel de batería, voltaje y corriente.
- Panel de progreso (II): porcentaje y barra de cobertura de la zona patrullada.
- Mapa 3D interactivo (III): visualización del dron con icono identificativo, seguimiento de cámara, pirámide de visión, límites de patrullaje, ruta activa y mapa de calor del área cubierta.

- Panel de control (IV): selector de velocidad, botones toggle para crear recorridos, fijar la base del dron o ubicar personas desaparecidas, así como controles para ejecutar el plan de vuelo o forzar el regreso inmediato a base.

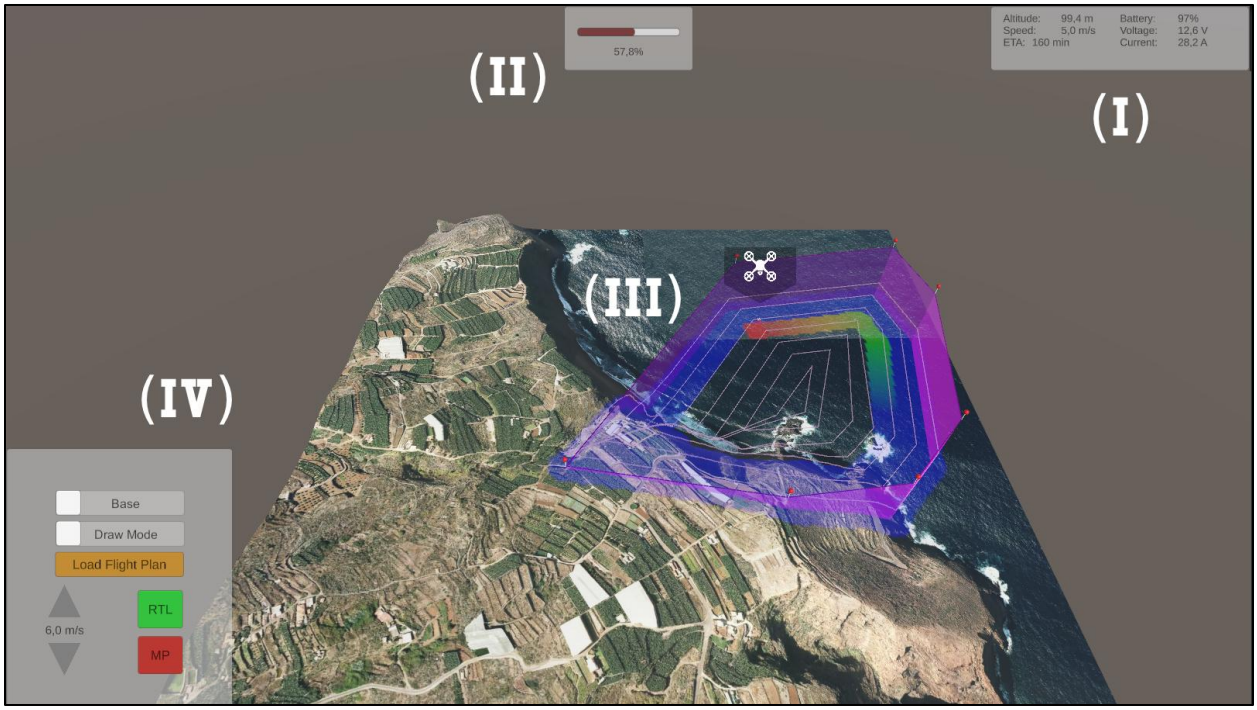

Figura 4-2. Captura de la aplicación con los destacados.

4.3 Simulador de vuelo: DroneKit + ArduPilot-SITL

El subsistema de simulación se fundamenta en ArduPilot, que se ejecuta en modo SITL (Software In The Loop) para emular el funcionamiento del firmware de un piloto automático real en un entorno puramente software. En este contexto, ArduPilot se encarga de procesar la dinámica de vuelo, los modos de operación y la gestión de sensores virtuales.

Sobre esta base, DroneKit proporciona una API de alto nivel en Python que facilita la interacción con el autopiloto a través del protocolo MAVLink. Esta librería permite tanto la consulta de variables internas (por ejemplo, estado de los sensores, posición, orientación, nivel de batería, modo de vuelo) como el envío de órdenes de control (armado, despegue, navegación a puntos de ruta o retorno automático).

En conjunto, ambos componentes ofrecen un entorno simulado que reproduce con fidelidad el comportamiento de un dron real, generando datos de telemetría (posición GPS, altitud, velocidad, actitud y otros parámetros de estado) que son consumidos por el servidor y la aplicación.

Sin embargo, debido a limitaciones de esta herramienta, algunas funciones han sido externalizadas:

- **Altura**, gestionada desde la aplicación, ya que el modelo de elevación de Mapbox no coincide con el de DroneKit.
- **Batería**, simulada desde el servidor de WebSocket, aunque la lógica de recarga y reanudación de misión se gestiona en la aplicación.

El resto de variables son simuladas por DroneKit-SITL, garantizando un comportamiento globalmente coherente con un dron físico.

4.4 Servidor de comunicación: WebSocket

El servidor de WebSocket constituye la capa intermedia de comunicación, permitiendo el intercambio de datos entre DroneKit-SITL y la aplicación. Por un lado, recibe la telemetría del simulador y la transmite a la aplicación para actualizar la representación visual y la interfaz. Por otro, canaliza los comandos enviados desde la aplicación, como órdenes de regreso a base o ejecución de nuevos recorridos, hacia el simulador.

Además de esta función de puente, el servidor asume la simulación del consumo energético de la batería, actualizando en tiempo real el porcentaje restante y comunicando los eventos asociados al retorno y recarga en base. Esta decisión se tomó para superar las limitaciones de DroneKit y representar un comportamiento realista acorde a un dron usado en este tipo de misiones así como para centralizar el control de la gestión energética, haciéndolo más flexible.

4.5 Integración y flujo de datos

La integración entre la aplicación, el servidor WebSocket y el sistema de simulación se apoya en un flujo de mensajes en formato JSON transmitidos de forma asíncrona sobre WebSockets. Este diseño facilita la interoperabilidad entre distintos lenguajes de programación (C# en Unity y Python en el servidor) y permite una comunicación en tiempo real con baja latencia.

El funcionamiento del sistema sigue un ciclo continuo:

1. ArduPilot-SITL (a través de DroneKit) genera telemetría en tiempo real.
2. El servidor WebSocket recibe dicha información y la envía a la aplicación.
3. La aplicación actualiza la representación gráfica, la interfaz y la lógica asociada al recorrido.
4. El usuario puede emitir comandos desde la interfaz.
5. Estos comandos son transmitidos al servidor y, desde allí, enviados a DroneKit-SITL, cerrando el ciclo.

Este bucle asegura la coherencia entre simulación física, representación visual y control de misión. Además, la lógica energética permite que el dron interrumpa automáticamente su patrulla, regrese a la base cuando alcanza un nivel crítico de batería y reanude la misión desde el punto exacto donde se detuvo.

Como podemos observar en la Figura 4-3, la comunicación se organiza de manera bidireccional:

- **Telemetría hacia la aplicación**: el servidor recibe de ArduPilot-SITL variables de estado (posición, orientación, velocidad, nivel de batería, modo de vuelo, etc.) y las transmite en JSON a Unity, que actualiza la escena 3D y los paneles de la interfaz.
- **Comandos hacia el simulador:** la aplicación envía órdenes de control (ajuste de velocidad, carga de misión, fijación de base, retorno, gestión de batería) en JSON al servidor, que las traduce a mensajes MAVLink interpretados por DroneKit-SITL.
- **Estado de misión:** la aplicación combina la telemetría con su lógica interna (recorrido, mapa de calor, reanudación tras recarga), mientras que el servidor

gestiona en paralelo el consumo de batería y notifica eventos como batería baja, retorno y recarga.

En conjunto, esta arquitectura asegura la sincronización de todos los módulos y constituye la base sobre la que se justifican las decisiones de diseño del sistema.

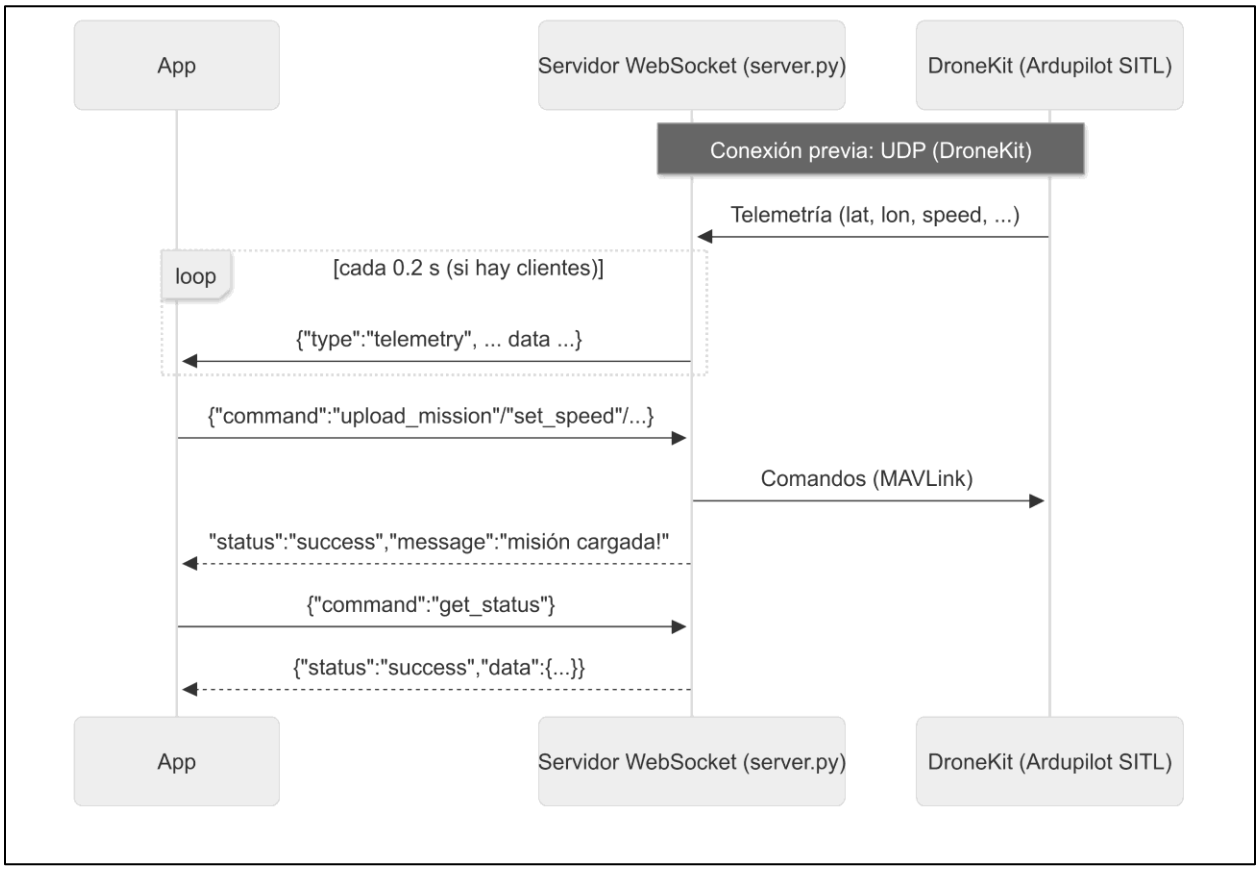

Figura 4-3. Diagrama de la comunicación entre aplicación, servidor y simulador

4.6 Justificación de diseño

La arquitectura planteada responde a criterios de equilibrio entre realismo, control y accesibilidad. La separación en tres componentes principales, aplicación, servidor y simulador, permite distribuir responsabilidades de forma clara, simplificando tanto el desarrollo como la validación del sistema. Esta organización facilita además la incorporación de nuevas funcionalidades en fases posteriores, ya que cada bloque mantiene una función bien definida.

La elección de Unity como motor gráfico y entorno de desarrollo responde a varias razones estratégicas. En primer lugar, Unity ofrece un ecosistema maduro y ampliamente documentado, lo que facilita tanto la integración con librerías externas (como Mapbox) como la interoperabilidad con otros lenguajes a través de protocolos estándar (JSON, WebSocket). Esta flexibilidad ha sido clave para conectar de forma estable la lógica de la simulación (DroneKit-SITL en Python) con la representación visual y la interfaz desarrollada en C#.

En segundo lugar, Unity proporciona potentes capacidades de renderizado 3D en tiempo real que permiten generar escenarios inmersivos basados en datos cartográficos reales. Gracias a ello, ha sido posible construir una visualización clara y detallada de la misión, incluyendo la posición del dron, su pirámide de visión infrarroja, el mapa de calor del área cubierta y la interacción con el entorno. Todo esto resulta esencial para que la herramienta sea comprensible y operativa para usuarios sin formación técnica avanzada.

Además, Unity facilita la implementación de un sistema de control directo por parte del usuario, permitiendo mover la cámara, ajustar el nivel de zoom, alternar perspectivas y seleccionar zonas de patrullaje mediante interacción con el entorno tridimensional. Esta capacidad de integrar de manera natural la navegación y la manipulación de elementos en la escena refuerza la usabilidad de la aplicación y contribuye a que pueda ser utilizada de forma intuitiva por operarios sin necesidad de formación técnica previa.

El uso de Mapbox integrado en Unity aporta realismo geográfico y flexibilidad para situar misiones en cualquier parte del mundo, mientras que la delegación de ciertas funciones, como la gestión de altura o el modelado de la batería, se ha realizado estratégicamente para maximizar el control y la estabilidad del sistema. Gracias a estas decisiones, el proyecto combina de manera eficaz un motor de simulación realista (ArduPilot-SITL/DroneKit) con una aplicación accesible y altamente configurable.

4.7 Problemas encontrados y soluciones

1. **Diferencias en los modelos de altura**
    - Problema: el mapa de elevación de Mapbox no coincidía con el modelo de altitud gestionado por DroneKit, generando inconsistencias visuales.
    - Solución: la altura pasó a ser controlada directamente desde la aplicación, imponiendo coherencia con el terreno representado.
2. **Gestión limitada de la batería**
    - Problema: además de que DroneKit-SITL no modela una descarga energética realista, tampoco se disponía de un dron simulado con las prestaciones de un dron de rescate policial (capacidad, número de celdas y umbrales operativos), con el fin de evaluar misiones verosímiles.
    - Solución: Se implementó un modelo propio.
        - **Potencia y energía:** se calcula $P = V \cdot I$ y se integra en el tiempo $\Delta E \approx \sum P \cdot \Delta t$ para actualizar el estado de carga (SoC) respecto a la capacidad nominal.
        - **Parámetros del pack:** 12 celdas, $Vnom, celda = 3{,}7\ V, C = 22\ Ah \rightarrow$ $energía\ nominal\ E_{nom} = 12 \times 3{,}7 \times 22 = 976{,}8\ Wh$

          Este valor se alinea con las especificaciones de drones profesionales empleados por cuerpos policiales y de emergencias.

          El modelo principal de referencia ha sido el DJI Matrice 400 que incorpora packs TB100 de ~977 Wh, prácticamente idénticos al modelo simulado en este proyecto, y está específicamente orientado a seguridad pública, policía y operaciones de rescate [34].
        - **Umbrales operativos:** se aplican umbrales de aviso y RTH(del inglés Return to Home) en función de SoC y/o voltaje por celda para evitar operación por debajo del corte seguro.
        - **Recarga y reanudación:** cuando el estado de carga (SoC) desciende por debajo del umbral de retorno, la aplicación ordena de forma automática el regreso a base. El dron interrumpe la patrulla, navega hasta la base,

aterriza, y a continuación se inicia la recarga simulada (incremento controlado del SoC hasta el objetivo establecido). Completada la recarga, el sistema reanuda la misión de manera autónoma desde el último waypoint pendiente, preservando el orden del recorrido y la cobertura global del área.

- **Resultado:** Un comportamiento de descarga/recarga coherente con multirrotores de rescate, que mejora la fidelidad frente al esquema básico del simulador y permite análisis de autonomía y estrategia de retorno más realistas.

3. **Sincronización en tiempo real**
    - Problema: la transmisión frecuente de datos podía generar retardo en la visualización.
    - Solución: el servidor emite telemetría a intervalos de 0,2 s. En la aplicación se decidió no interpolar posiciones entre mensajes (sin *lerp*), de manera que el dron se representa exactamente en los puntos transmitidos. Con ello se prioriza la fidelidad de los datos reales recibidos frente a la suavidad visual.
4. **Compatibilidad entre C# y Python**
    - Problema: las APIs de Unity y DroneKit están en lenguajes distintos.
    - Solución: se definió un protocolo propio en JSON, con campos estandarizados (lat, lon, alt, yaw, groundspeed, batería, estado de armado, modo de vuelo), lo que simplificó la interoperabilidad.

4.8 Limitaciones técnicas

El sistema presenta también algunas limitaciones técnicas que condicionan su escalabilidad y aplicabilidad:

- **Escalabilidad a múltiples drones**. La arquitectura ha sido diseñada inicialmente para un dron único. La extensión a una flota requeriría rediseñar la gestión de estados en el servidor y en la aplicación, así como el protocolo de mensajes, para mantener sincronización y evitar conflictos en la comunicación.

- **Realismo limitado en sensores**. La detección se representa de forma simplificada mediante un cono de visión y un mapa de calor, sin procesamiento de imagen ni simulación avanzada de cámaras infrarrojas. Si bien es suficiente para el objetivo de este proyecto, no reproduce las limitaciones físicas de sensores reales (ruido, interferencias, falsas detecciones).
- **Dependencia de DroneKit-SITL.** Aunque DroneKit-SITL proporciona telemetría básica y comportamientos de vuelo coherentes, no reproduce todas las dinámicas físicas del dron (como turbulencias, fallos de motor o interacción con condiciones meteorológicas). Además, carece de modelos avanzados de batería o de sensores complejos, lo que obliga a delegar parte de la simulación a la aplicación y al servidor.
- **Integración entre sistemas operativos**. La ejecución actual implica lanzar DroneKit-SITL bajo un entorno Linux (WSL) mientras que la aplicación principal corre en Windows, con el servidor WebSocket como intermediario. Esta solución híbrida es funcional, pero puede dificultar su despliegue para usuarios finales no técnicos. Para un uso comercial o en campo sería recomendable empaquetar todo el sistema en un único entorno o contenedor multiplataforma.
- **Consumo de recursos**. La integración de mapas 3D de Mapbox, junto con la simulación en paralelo de DroneKit y el servidor, exige un equipo con cierta capacidad gráfica y de procesamiento. Aunque se mantiene fluido en hardware de gama media, su uso en equipos menos potentes podría verse limitado.

Capítulo 5 - Algoritmo de patrullaje

El algoritmo de patrullaje convierte el polígono definido por el usuario en una ruta de cobertura completa y compatible con el simulador. La implementación reside en la aplicación y se articula en cuatro fases: (I) definición de la zona, (II) generación de anillos de cobertura mediante *offset* interior, (III) construcción de una trayectoria continua tipo boustrophedon (alternando sentido por anillo), y (IV) postprocesado geométrico/altimétrico para producir *waypoints* finales y (V) exportarlos al backend.

5.1 Descripción detallada del algoritmo

I) Entrada: polígono y parámetros operativos

El usuario delimita la zona con clics colocando nodos sobre el mapa 3D (Mapbox) desde la opción *Draw mode*. Al acabar, el polígono se cierra, se normalizan nodos, se levantan "paredes" verticales para visualizar límites, como podemos observar en la Figura 5-1, y se dispara el proceso de generación de ruta (descrito en el siguiente apartado y en el Apéndice A). Los parámetros clave expuestos al usuario son:

- Altura sobre el terreno (*AGL, del inglés Above Ground Level*), intensidad de adaptación al relieve *adaptationIntensity*, y espaciado de muestreo *sampleSpacing*.
- Simplificación en planta *simplificationFactor*, salto vertical máximo *maxVerticalStep*, y umbral de inserción de WPs por pendiente *slopeWaypointThreshold*.
- Límite de waypoints para DroneKit *maxWaypoints* (≤ 650, valor máximo de Ardupilot).

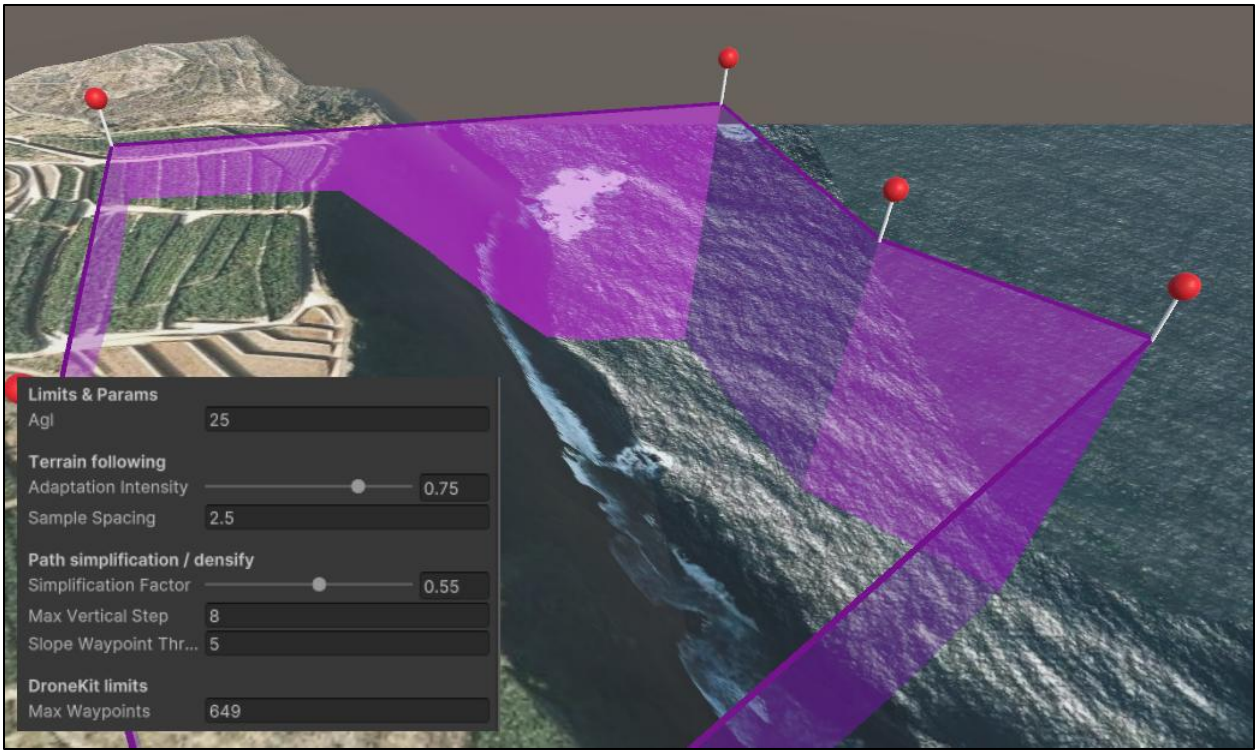

Figura 5-1. Visualización de parámetros y delimitación del área de trabajo del dron.

II) Cobertura por offset interior (anillos)

Formalización y objetivo.

El proceso de cobertura se basa en la construcción de anillos concéntricos mediante la operación de offset negativo aplicada al polígono inicial. Formalmente, Sea $P0 \subset R2$ el polígono definido por el usuario en el plano de escena (eje X - Z de Unity). El algoritmo genera una familia finita de subpolígonos $\{Pi\}_{i=0}^{k}$ por offset negativo con paso constante $\mathrm{d} > 0$:

$$P_{i+1} = Offset(Pi, -d), \qquad para\ i = 0, \dots, k-1,$$

hasta que P_{i+1} es vacío o pierde viabilidad. Cada P_i se convierte en un anillo de cobertura; el recorrido alterna sentidos (CW/CCW) y se cose entre anillos con el punto de inicio más próximo al último vértice visitado, minimizando transiciones.

Elección de d a partir del Field of View y AGL.

Se aproxima el lado efectivo de cobertura en planta usando el valor de la altura de vuelo (AGL) definida:

$$fov_{side} = 1{,}2 \cdot AGL, \qquad d = 0{,}8 \cdot fov_{side}$$

lo que garantiza solape entre pasadas contiguas. Este d se sincroniza con el *sampleSpacing* para evitar sobre-densidades innecesarias en el postprocesado.

Implementación del offset.

La operación $Offset(\cdot)$ se delega en una librería robusta de geometría poligonal (p. ej., Clipper mediante *InflatePaths*), ampliamente utilizada en CAD/GIS, lo que evita lidiar con casos degenerados y asegura estabilidad numérica en polígonos con concavidades pronunciadas. En este trabajo no se modifica el algoritmo interno de offset; se parametriza el *join type* y *end type* adecuados a contornos cerrados, con tolerancias acordes a la escala del mapa.

Criterios de terminación y limpieza.

- **Terminación**: cuando el offset devuelve vacío o cuando el área de P_{i+1} cae por debajo de un umbral ε_{area} (evita anillos minúsculos).
- **Filtrado de fragmentos**: si el offset produce múltiples polígonos (p. ej., por estrechamientos), se ordena por área y se mantiene el mayor o se concatenan por tamaño. Los fragmentos subcríticos se descartan. Estos fragmentos se refieren a los subpolígonos de área muy reducida (por debajo de τ_{area}) que no resultan operativamente útiles, ya que generarían trayectorias residuales de escasa extensión, aumentarían el número de *waypoints* y complicarían la misión sin aportar una cobertura significativa. Por este motivo, dichos fragmentos se descartan en la fase de limpieza, garantizando así un recorrido compacto, eficiente y ejecutable por el dron.
- **Regularización opcional**: si apareciera auto-intersección (raro con librerías robustas), se aplica unión interna antes de continuar.

Complejidad aproximada.

Si n es el número de vértices de P_i , el offset y clipping interno suelen escalar en $O(n \log n)$ por anillo. El número de anillos k es acotado por el radio inradius y el paso d (aprox. $k \approx \frac{inradius(P_0)}{d}$). En la práctica, el coste total se controla mediante d, *sampleSpacing* y la simplificación posterior.

III) Trayectoria boustrophedon y conexión entre anillos

Para minimizar maniobras, cada anillo se recorre en sentido alterno (CW/CCW), cosiendo la ruta entre anillos desde el punto del nuevo anillo más cercano al último punto visitado (rotación de la lista para comenzar en ese vértice y cerrar la vuelta). Esto produce una única polilínea cerrada que recorre todos los anillos con transiciones cortas entre ellos.

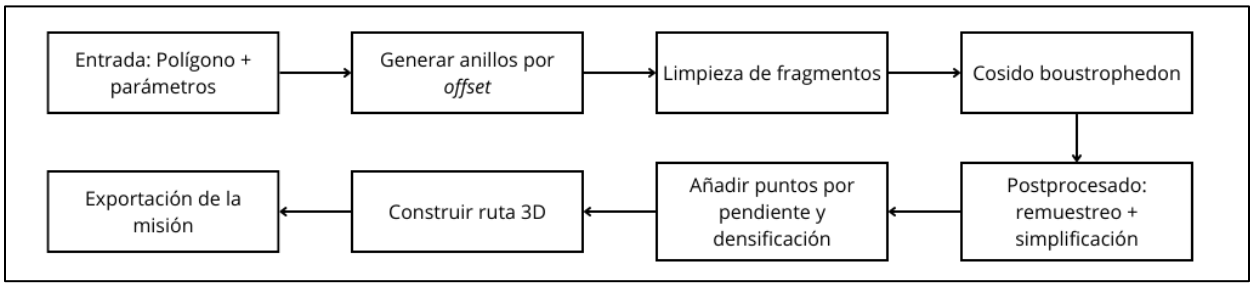

Figura 5-2. Diagrama de flujo de la generación del recorrido de patrulla.

IV) Postprocesado geométrico y altimétrico

Una vez generada la trayectoria inicial, esta se somete a un proceso de refinamiento diseñado para equilibrar la densidad de puntos, la suavidad de las transiciones y la adaptación al relieve. Dicho postprocesado se compone de cuatro etapas secuenciales:

1. **Remuestreo por longitud de segmento (ResamplePath).**

   Se interpola la trayectoria para que la distancia entre puntos consecutivos no exceda el valor definido en *sampleSpacing*. Este paso garantiza una discretización uniforme y controlada.

2. **Simplificación en planta (Douglas–Peucker).**

   Sobre la proyección 2D se aplica el algoritmo de Douglas–Peucker con una tolerancia $\varepsilon = fov_{side} \cdot simplificationFactor$, eliminando vértices redundantes sin comprometer la cobertura del área.

3. **Inserción por pendiente (AddSlopeBreaks).**

   Cuando la diferencia de altitud entre dos muestras sucesivas supera el umbral *slopeWaypointThreshold*, se introduce un punto intermedio cuya cota se obtiene del modelo de elevación de Mapbox. Este mecanismo evita saltos bruscos en la altitud.

4. **Densificación mixta (DensifyForHeight).**

Se imponen simultáneamente dos restricciones: distancia horizontal máxima entre puntos (≤ sampleSpacing) y salto vertical máximo (≤ maxVerticalStep). Así se asegura una transición progresiva y segura en el espacio tridimensional.

Finalmente, la aplicación incluye una función (descrita en el Apéndice A) que asigna a cada vértice una altitud definida como la cota del terreno más la altura AGL fijada, aplicando un suavizado vertical controlado por adaptationIntensity (0 = vuelo completamente plano, 1 = seguimiento exacto del relieve). El resultado es un recorrido tridimensional consistente, representado gráficamente en la escena que se almacena como referencia para su posterior exportación, aspecto documentado en el Apéndice A.

V) Exportación a misión (compatibilidad DroneKit)

El último paso consiste en transformar el recorrido refinado en una misión ejecutable por el simulador DroneKit-SITL. Para ello, *una función específica de la aplicación* realiza las siguientes operaciones:

- **Conversión de coordenadas.** Cada punto de la ruta en el plano de Unity (XZ) se proyecta a coordenadas geográficas (latitud, longitud) mediante Mapbox. La altitud se obtiene como la suma de la cota del terreno (MSL) y la altura AGL seleccionada.

- **Control del número de waypoints.** Si la misión supera el límite impuesto por DroneKit (*maxWaypoints* ≤ *650*), se aplica un muestreo proporcional que conserva la distribución espacial de los puntos, reduciendo el tamaño de la lista sin alterar la cobertura esencial.
- **Envío al servidor.** Los waypoints resultantes se empaquetan en una estructura JSON y se transmiten al simulador a través del cliente WebSocket, que los reenvía a DroneKit para su validación y ejecución.

Este procedimiento asegura la compatibilidad plena con la pila DroneKit/SITL, evitando errores de sobrecarga de misión y garantizando que el dron virtual ejecute un plan de vuelo coherente con la ruta calculada en la aplicación.

5.2 Comparación breve con alternativas

Diversos enfoques han sido explorados en la literatura y en aplicaciones prácticas para la cobertura de áreas con drones. A continuación se presentan las principales alternativas, destacando sus ventajas e inconvenientes en relación con la estrategia implementada. También podemos ver la comparativa de forma más concreta en la Tabla 5-1.

- **Barrido por líneas paralelas (line-sweep clásico).**

  Equivalente al patrón *lawnmower* ortogonal, alineado a un eje fijo. Requiere recortar cada pasada contra los límites del polígono y manejar de forma explícita concavidades o "islas". Aunque ofrece rutas regulares en polígonos convexos, su implementación es más compleja en geometrías irregulares.

- **Patrones aleatorios o espirales.**

  Útiles en escenarios de exploración sin mapa previo, donde la cobertura completa no es un requisito estricto. Sin embargo, en contextos donde se exige garantizar el 100 % de la cobertura generan redundancias significativas y tiempos de misión superiores, lo que los hace poco adecuados para tareas sistemáticas de búsqueda.

- **Descomposición en celdas boustrophedon (GRF/BCD).**

  Este método divide el área en subregiones simples y calcula un recorrido de tipo boustrophedon dentro de cada celda, generando trayectorias muy limpias incluso en polígonos complejos. No obstante, en el contexto de este proyecto resultaba innecesariamente costoso: requiere una segmentación topológica previa y la gestión de transiciones entre celdas mediante "puertas", lo que complica la implementación y aumenta la probabilidad de fallos en geometrías irregulares o con concavidades pronunciadas.

  Frente a ello, el enfoque adoptado, basado en offsets interiores cosidos alternativamente, produce de forma directa una única polilínea continua, evita operaciones adicionales de segmentación y se integra de manera más sencilla con DroneKit, que consume misiones lineales de waypoints. Además, dado que el objetivo no es maximizar la optimalidad matemática de la ruta, sino garantizar cobertura completa dentro del límite de 650 puntos, la estrategia por anillos ofrece un mejor equilibrio entre robustez, simplicidad y compatibilidad.

- **Planificación sobre grafos (A, PRM, etc.).**

  Permite modelar costes anisótropos (por ejemplo, riesgo, consumo energético o presencia de obstáculos dinámicos). Son técnicas potentes en entornos parcialmente conocidos o con restricciones adicionales, pero su complejidad es innecesaria en escenarios abiertos cuyo objetivo principal es la cobertura uniforme.

- **Balance**

  La estrategia desarrollada en este trabajo, basada en offsets interiores concatenados mediante cosido alterno, ofrece un compromiso adecuado entre simplicidad de implementación, robustez ante polígonos arbitrarios y eficiencia práctica. El postprocesado geométrico y altimétrico contribuye a mantener un número de waypoints compatible con las limitaciones de DroneKit (≤ 650) sin sacrificar cobertura.

  Como limitación, este enfoque puede generar solapes en cuellos estrechos, algo que se mitiga parcialmente mediante los parámetros de

simplificación y densificación. En contrapartida, se evita la complejidad de la segmentación en celdas o la planificación sobre grafos, lo que lo convierte en una solución idónea para el objetivo de simulación de misiones de patrullaje en áreas definidas por el usuario.

MÉTODO	COMPLEJIDAD	COBERTURA Y ROBUSTEZ	EFICIENCIA Y ADECUACIÓN
BARRIDO PARALELO	Media	Cobertura completa en convexos, limitado en concavidades	Bueno en simples, pobre en irregulares
ALEATORIO/ESPIRAL	Baja	No garantiza cobertura	Poco eficiente, escasa adecuación
CELDAS BOUSTROPHEDON	Alta	Cobertura y robustez muy buenas	Preciso pero costoso
OFFSETS INTERIORES (ESTE TRABAJO)	Media	Cobertura garantizada, robusto salvo cuellos estrechos	Ruta compacta, bien en irregulares
GRAFOS (A*, PRM, ETC.)	Alta	Cobertura flexible y robusta	Variable, sobrecoste innecesario

Tabla 5-1. Comparativa de métodos de generación de rutas

5.3 Ejemplos de generación de rutas

Con el fin de ilustrar de manera práctica el funcionamiento del algoritmo de patrullaje desarrollado, a continuación se presentan varios ejemplos aplicados a zonas de distinta geometría. Estos casos permiten observar cómo el procedimiento basado en offsets interiores y cosido alterno se adapta a polígonos simples y complejos, garantizando la cobertura completa del área y manteniendo la coherencia con los parámetros de altura y densidad de muestreo establecidos por el usuario.

- **Rectángulo**. Los offsets generan anillos rectangulares concéntricos; el cosido alterno minimiza transiciones y la simplificación elimina vértices superfluos. Cobertura uniforme con solape controlado por AGL.

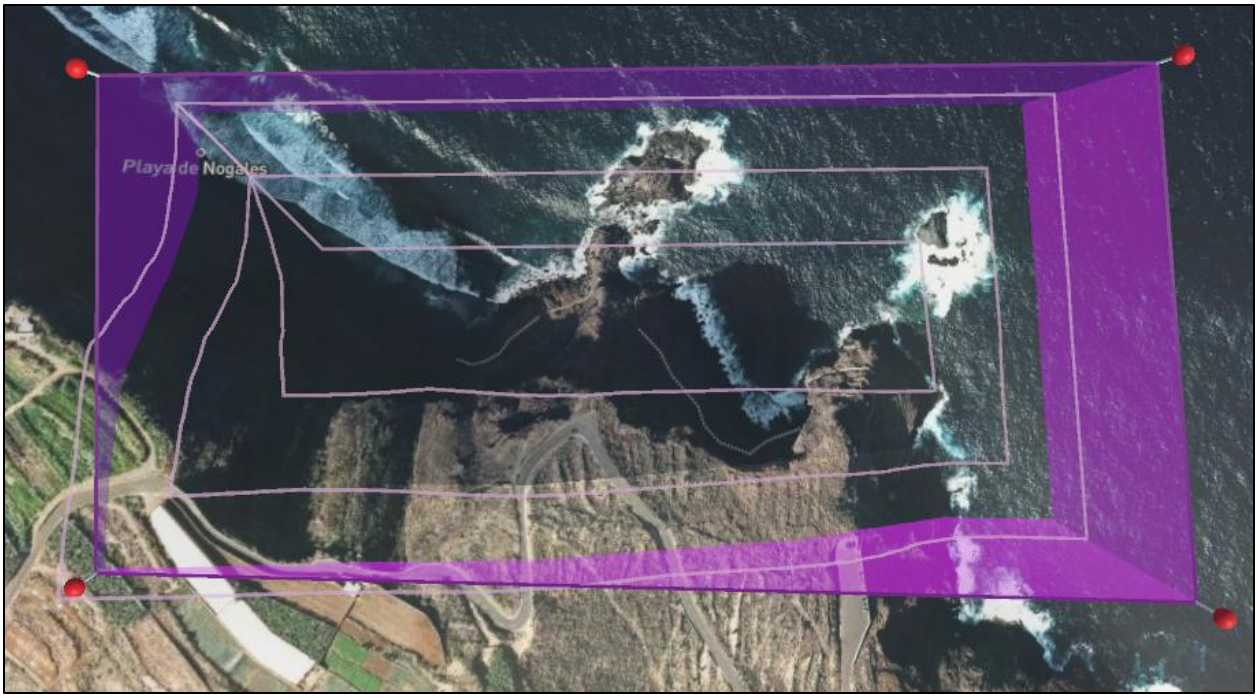

Figura 5-3. Ejemplo de generación de ruta en polígono rectangular.

- **Polígono irregular con concavidades**. Los offsets respetan la forma; la inserción por pendiente añade puntos donde el relieve cambia bruscamente, evitando saltos de altura. Pequeños solapes pueden aparecer en cuellos estrechos, mitigados por simplificationFactor.

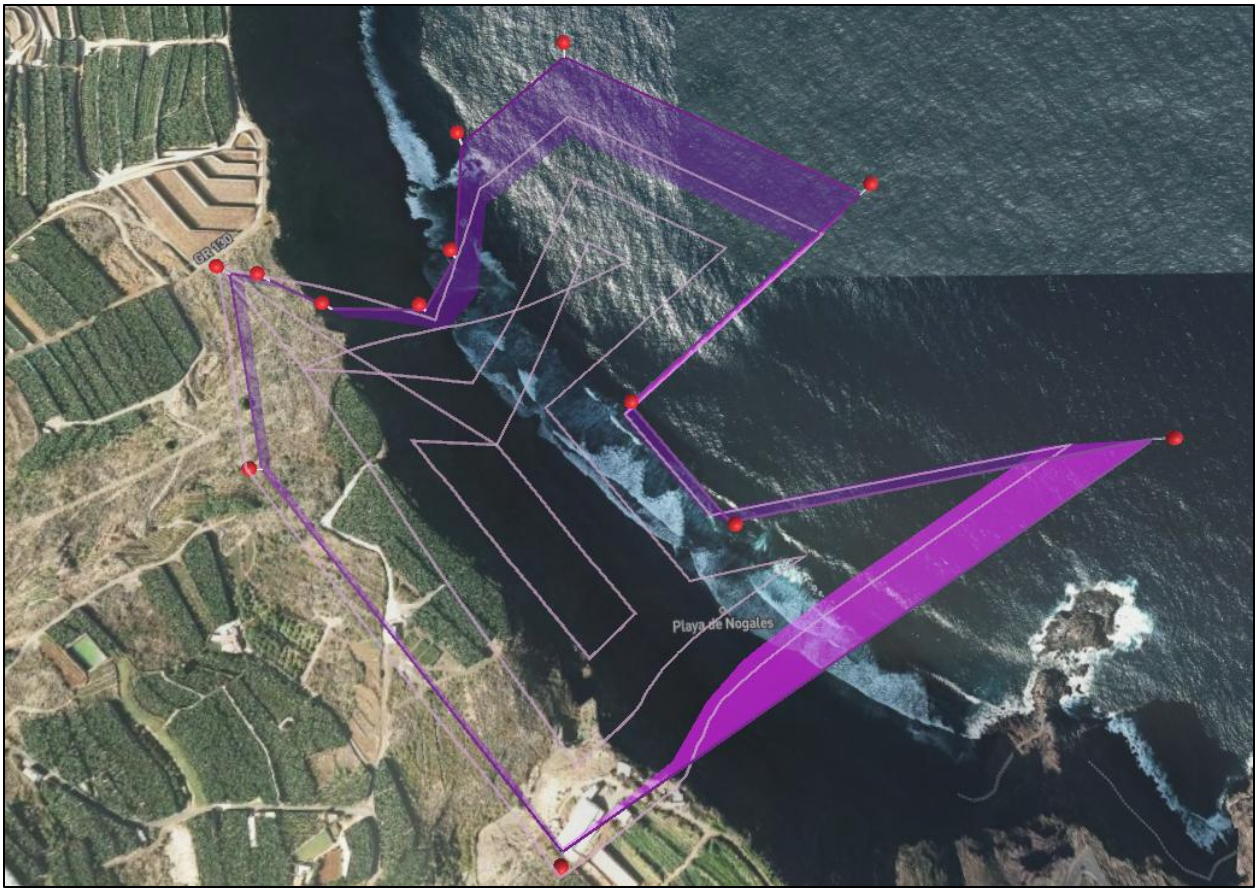

Figura 5-4. Ejemplo de generación de ruta en polígono complejo.

Capítulo 6 - Simulación y entorno operativo

6.1 Modelado energético y de batería

Objetivo y alcance.

La simulación energética se implementa en el servidor WebSocket (Python) para centralizar un modelo de batería realista pero ligero y exponerlo a la aplicación Unity mediante telemetría periódica. El cliente Unity consume estos datos para visualización y lógica de misión.

Parámetros físicos y capacidad.

El servidor define una batería 12S, 22 Ah, con tensión nominal por celda 3,7 V. La energía nominal se calcula como:

$$E_{nom} = N_{celdas} \cdot V_{celda} \cdot Ah = 12 \cdot 3{,}7 \cdot 22 \approx 977\ Wh.$$

Este valor inicializa el estado energético disponible *battery_wh_remaining* y sirve de base para estimar el SoC (estado de carga) como porcentaje

Telemetría y frecuencia.

Cada 0,2 s el servidor emite un paquete JSON con: posición, orientación, velocidad en tierra, y un bloque de batería con voltaje, corriente, nivel (%), energía restante (Wh) y ETA (min). Esta cadencia alinea la fidelidad temporal de la energía con la posición/actitud del dron.

Modelo de descarga por potencia ($\boldsymbol{P = V \cdot I}$).

Mientras el vehículo está armado, el consumo energético se integra a partir de la potencia instantánea:

$$P(t) = V(t) \cdot I(t), \qquad \Delta E = P(t) \cdot \frac{\Delta t}{3600}$$

En cada iteración, la energía restante se actualiza como

$$E_{rest} \leftarrow max(0,\ E_{rest} - \Delta E),$$

y el **SoC** se recalcula como $\% = 100 \cdot E_{rest}/E_{nom}$. La **ETA** se estima con $ETA \approx 60 \cdot E_{rest}/P(t)$ cuando hay medidas de *V* e *I*.

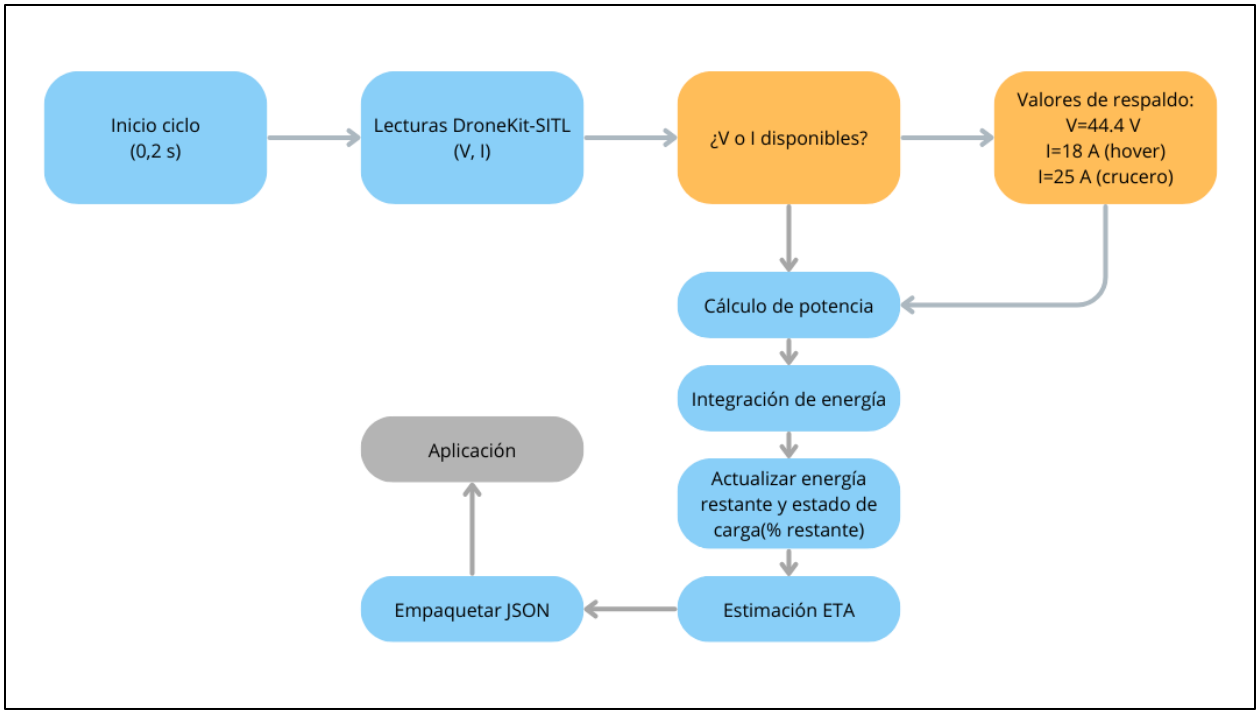

Figura 6-1. Diagrama de flujo del sistema de batería del dron.

Fuentes de V e I y valores de respaldo.

El servidor utiliza las lecturas de DroneKit-SITL (*vehicle.battery.voltage/current*). En ausencia de medidas, se aplican valores de respaldo realistas:

- **Voltaje**: se emplea un valor nominal de 44,4 V (12·3,7 V), acotado entre 42,0 V y 50,4 V para representar el rango operativo seguro de un pack 12S.
- **Corriente**: se define un respaldo diferenciado según el perfil de vuelo, en lugar de un valor fijo.
  - **18 A** en vuelo estacionario o baja velocidad (hover).
  - **25 A** en crucero normal (velocidad ≥ 2 m/s).

Este mecanismo garantiza continuidad en la simulación incluso si la pila de SITL no reporta correctamente la batería, evitando estimaciones irreales de autonomía.

Exposición al cliente y UI.

El paquete de telemetría incluye *battery.voltage, battery.current, battery.level (SoC %) y eta_min.* El cliente Unity deserializa estos campos y los muestra en la interfaz (nivel %, voltaje, corriente y ETA), además de utilizarlos en la lógica superior.

Control manual y pruebas.

Para escenarios de prueba, el cliente puede inyectar un nivel de batería simulado (*set_battery_level*) que recalcula internamente la energía restante de forma consistente con el SoC deseado. Esto permite reproducir condiciones de batería baja/alta sin depender de un vuelo largo.

Limitaciones del modelo.

El modelo es determinista y de primer orden (energía por potencia). No contempla:

- Resistencia interna ni caída de tensión dependiente de la corriente (sag).
- No linealidad por ley de Peukert ni envejecimiento/temperatura.
- Dependencia explícita de consumo con la aerodinámica o el perfil de velocidad vertical (se asume que SITL modula I). Estas simplificaciones se aceptan por su bajo coste computacional y por ofrecer métricas robustas para la simulación de misión (SoC, ETA). La integración con SITL asegura que la corriente refleje, en lo posible, el estado dinámico del vuelo.

Consistencia con DroneKit/SITL y misión.

El servidor empaqueta la batería dentro de la misma telemetría que posición/velocidad/modo, por lo que la UI siempre visualiza información sincronizada. Además, las órdenes de control (p. ej., cambios de velocidad de navegación o reanudación de misión) no alteran directamente el modelo energético; su impacto llega a través de la corriente que SITL reporta, preservando la coherencia física.

6.2 Implementación de la detección simulada de personas

Objetivo y enfoque.

La detección se modela mediante una pirámide de base cuadrada invertida anclada al dron, que representa el campo de visión de una cámara infrarroja de orientación nadiral. Este volumen constituye el área de exploración: todo objeto situado dentro de la pirámide se considera potencialmente visible para el sensor. El modelo busca un equilibrio entre simplicidad geométrica, bajo coste computacional y verosimilitud respecto a un sensor real en vuelo.

Representación del volumen.

El volumen de visión es una pirámide cuadrada, como podemos ver en la Figura 6-2 de color rojo. Este volumen actúa como un frustum invertido, con la base cuadrada proyectada sobre el terreno y el vértice en la posición del dron.

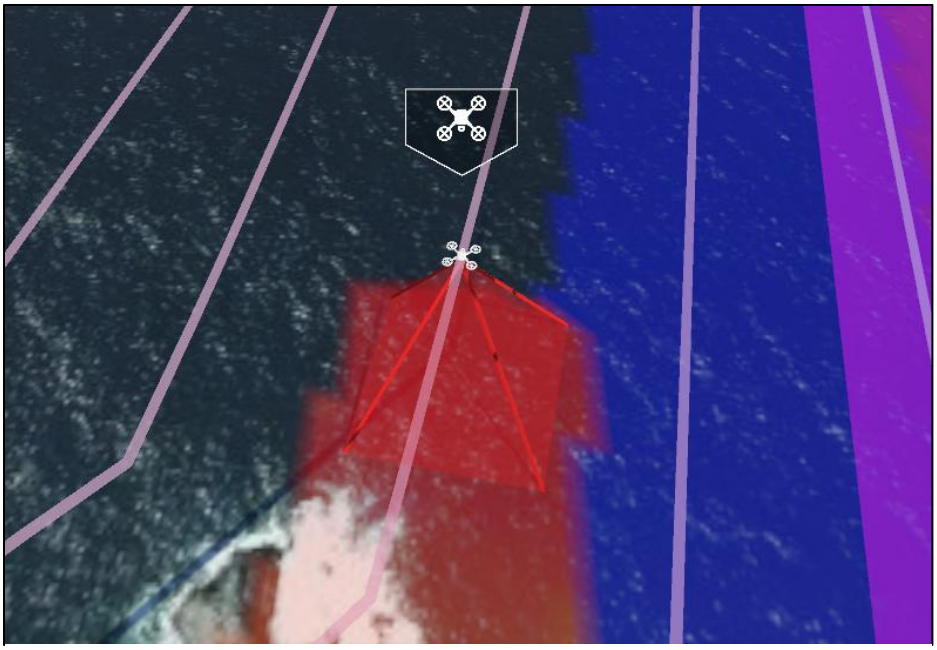

Figura 6-2. Campo de visión simulado durante una patrulla.

Intersección con el terreno y actualización del mapa de cobertura.

En cada iteración de la aplicación, si la cámara está activa, se calcula la caja englobante de la pirámide y se consulta a qué elementos del terreno afecta mediante *detección de colisiones por el motor de físicas de la aplicación*. Para cada malla intersectada, se recorren sus triángulos y se evalúa si alguno de sus vértices está contenido en el volumen de la pirámide. En caso afirmativo, los vértices se marcan como "pintados" y se notifica a *la interfaz* para incrementar el porcentaje de cobertura. De este modo, la pirámide actúa como una brocha que proyecta sobre el terreno la huella de observación.

Criterio de inclusión de vértices.

El método desarrollado combina un filtrado preliminar basado en la caja englobante con una comprobación más precisa sobre el contorno de la pirámide. Si la distancia entre el vértice analizado y el punto más cercano de la superficie es inferior a 0,01 m, el vértice se considera incluido en el volumen. Este procedimiento garantiza robustez en tiempo real y evita recurrir a cálculos analíticos complejos de intersección triángulo–pirámide.

Detección de personas simuladas.

El método de detección de personas desaparecidas aplica el mismo principio: se realiza una comprobación espacial sobre la capa correspondiente a posibles objetivos. Si se identifica un candidato, se valida que su posición central se encuentre dentro del volumen de búsqueda. Una detección positiva activa un aviso en la interfaz y, en caso de que el dron esté en vuelo armado y no en modo de regreso automático, se emite la orden de retorno a la base a través del cliente de comunicación.

Colocación de objetivos.

Desde la aplicación se permite al usuario instanciar objetivos en el mapa mediante clic. Cada objetivo queda asociado a coordenadas lat/lon calculadas con Mapbox. Esta información se usa posteriormente cuando el dron detecta a nuestra persona y notifica en la interfaz con el mensaje: "Heat signal detected! Possible person found at lat: XX lon: YY". Esta vinculación facilita tanto la trazabilidad como la validación del subsistema de detección.

Limitaciones.

- El criterio de "algún vértice dentro de la pirámide" puede sobreestimar la cobertura en bordes o, en el caso de triángulos grandes, omitir intersecciones parciales.
- No se contemplan oclusiones por obstáculos 3D intermedios: cualquier persona bajo la base de la pirámide se considera visible, incluso si estuviera detrás de un objeto.
- La base cuadrada de la pirámide se mantiene fija; no se ajusta dinámicamente a variaciones de altura del terreno, lo que puede introducir pequeñas imprecisiones en pendientes fuertes.

6.3 Interfaz de usuario

Elementos implementados.

La aplicación ofrece una interfaz gráfica diseñada en Unity que combina controles operativos, indicadores de telemetría y herramientas de visualización. Los principales elementos se organizan en varios paneles:

- Panel de control (esquina inferior izquierda). Incluye los toggles para fijar la base de operaciones y activar el modo dibujo (definición de polígono), el botón para instanciar un objetivo de persona desaparecida (Missing Person), el control para cargar y ejecutar el plan de vuelo (Load Flight Plan), el botón de retorno inmediato a base (RTL), y el control incremental para ajustar la velocidad de crucero del dron.
- Panel de telemetría (esquina superior derecha). Muestra en tiempo real los valores principales transmitidos por el servidor: altura, velocidad, tiempo de batería restante (ETA), nivel de batería, voltaje y corriente. Estos indicadores proporcionan al usuario una visión clara del estado del dron durante toda la misión.
- Panel de progreso (parte superior central). Representa gráficamente, mediante una barra y un porcentaje, el avance de la cobertura en el área patrullada. Este

panel facilita el seguimiento del cumplimiento de la misión y permite evaluar la eficiencia del recorrido.

- Panel de notificaciones (parte superior central). Ante la detección de una persona desaparecida, se activa un banner informativo que indica la localización y alerta al usuario de la situación. Este componente refuerza la inmediatez de la respuesta ante eventos críticos.
- Mapa 3D interactivo. Constituye el núcleo de la interfaz, en el que se representan:
    - El icono del dron en movimiento.
    - La pirámide de visión infrarroja en color rojo, que delimita el volumen de detección.
    - Las paredes del polígono de patrulla en violeta.
    - La ruta calculada por el algoritmo en verde azulado.
    - El mapa de calor proyectado sobre el terreno, que evoluciona desde colores cálidos (zonas detectadas recientemente) a fríos (zonas ya patrulladas), ofreciendo una representación visual del grado de cobertura.

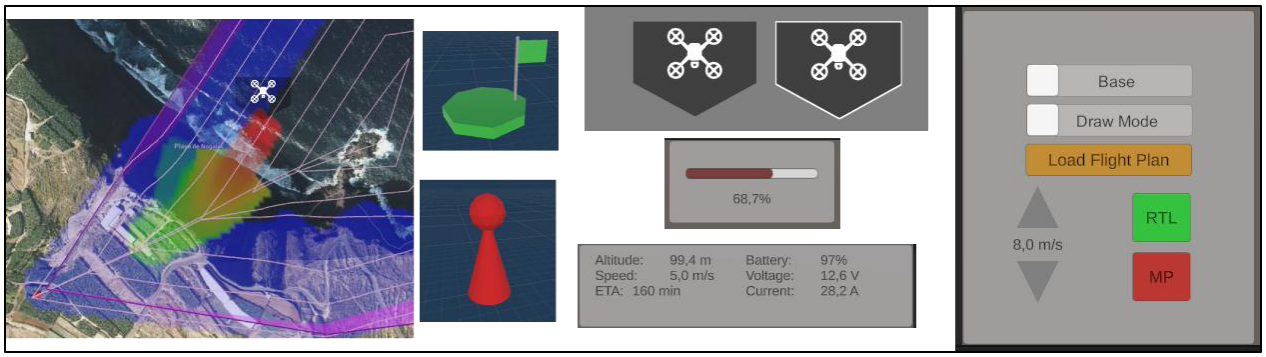

Figura 6-3. Elementos de la interfaz de usuario.

Controles de cámara.

- Movimiento libre: mantener clic derecho y arrastrar permite desplazar la cámara por el escenario.

- Zoom: con la rueda del ratón se ajusta el nivel de zoom, lo que facilita tanto la inspección detallada como la supervisión panorámica del área.
- Elevación: al mantener clic central se controla la altura de la vista, modificando el ángulo de observación.
- Seguimiento del dron: al hacer clic sobre el icono del dron en el mapa 3D, la cámara pasa a modo seguimiento, centrando la vista automáticamente en él durante el vuelo. Cualquier otra interacción en este modo cancela el seguimiento.

Consideraciones de usabilidad y accesibilidad.

La interfaz se ha diseñado bajo criterios de claridad y simplicidad, para ser utilizada por operarios sin formación técnica avanzada:

- Agrupación funcional. Los controles se ubican en un panel independiente, evitando solapamiento con la zona de visualización del mapa.
- Consistencia visual. Los colores de botones y elementos gráficos siguen convenciones estándar (verde para acciones seguras como RTL, rojo para estados críticos como Missing Person).
- Retroalimentación inmediata. Cada acción del usuario genera un cambio visible (p. ej., aparición de paredes al trazar el polígono, actualización porcentual de cobertura, mensajes de detección), lo que refuerza la transparencia del sistema.
- Legibilidad. Los paneles de telemetría utilizan tipografía clara sobre fondo semitransparente, asegurando visibilidad en distintos tipos de terreno.
- Accesibilidad operativa. La interacción se limita a clics y botones, sin necesidad de comandos manuales complejos, favoreciendo la adopción por equipos de rescate en entornos reales.

Capítulo 7 - Resultados y Discusión

7.1 Resultados obtenidos

El sistema desarrollado fue validado en distintos escenarios representativos, que abarcaron desde entornos costeros con acantilados hasta zonas planas y polígonos de geometría simple y compleja. Esta diversidad permitió comprobar tanto la robustez del algoritmo de patrullaje como la estabilidad de la integración entre el simulador DroneKit-SITL y la aplicación en Unity.

En todos los casos, el algoritmo generó rutas de cobertura completas y coherentes, adaptadas a la geometría definida por el usuario. En áreas simples, los recorridos se mantuvieron eficientes y sin redundancias apreciables, mientras que en geometrías más complejas se preservó la continuidad de la trayectoria, resolviendo correctamente situaciones en las que un trazado manual habría resultado tedioso o poco eficiente.

La simulación energética implementada en el servidor mostró un comportamiento autónomo verosímil: el dron interrumpía la patrulla al alcanzar un umbral crítico de batería, regresaba automáticamente a la base, simulaba el proceso de recarga y reanudaba la misión desde el punto exacto en el que la había dejado. Este ciclo validó la lógica de automatización planteada y aportó realismo al sistema.

La detección de personas, representada mediante un área de colisión simplificada, funcionó de manera consistente en todos los escenarios de prueba. El sistema notificó adecuadamente los objetivos simulados, reflejando su localización en la interfaz y proyectando el área de detección sobre el mapa 3D.

Finalmente, la interfaz de usuario resultó clave en la validación. Su diseño accesible permitió configurar patrullas, monitorizar variables críticas y controlar manualmente el dron sin necesidad de conocimientos técnicos específicos. Este aspecto confirmó uno de los objetivos centrales del proyecto: reducir la barrera de entrada para que la herramienta pueda ser utilizada por equipos de rescate no especializados.

El flujo completo de una misión, desde la configuración inicial hasta la finalización de la patrulla con detección de objetivos y gestión de batería, se ilustra en el vídeo complementario incluido en el Anexo, que acompaña a esta memoria como material adicional de validación.

7.2 Discusión crítica

Los resultados obtenidos muestran que el sistema cumple con los tres pilares definidos en la motivación del proyecto: eficiencia en la planificación de rutas, automatización del ciclo operativo y accesibilidad mediante una interfaz intuitiva.

Comparado con propuestas académicas revisadas en el estado del arte, que a menudo se apoyan en algoritmos de aprendizaje automático o en configuraciones multi-dron, la solución aquí presentada adopta un enfoque pragmático: algoritmos geométricos de bajo coste computacional y una arquitectura modular que prioriza la estabilidad y la facilidad de uso. Aunque este planteamiento limita el realismo en ciertos aspectos, como la fidelidad de los sensores o la escalabilidad a flota, también refuerza la viabilidad de la herramienta en entornos donde los recursos computacionales y humanos son reducidos.

Las validaciones en distintos escenarios confirmaron que el sistema es capaz de adaptarse a geometrías arbitrarias, gestionar de manera autónoma situaciones críticas de energía y proporcionar información clara al usuario. No obstante, deben señalarse algunas limitaciones: la detección de personas es todavía una abstracción muy simplificada y no refleja la complejidad de cámaras infrarrojas reales; la simulación se restringe a un único dron, sin soporte para misiones cooperativas; y la ejecución distribuida entre distintos entornos operativos (Windows y Linux) puede suponer una dificultad para usuarios sin experiencia técnica.

A pesar de estas limitaciones, la propuesta destaca por su aplicabilidad práctica. Mientras que muchas investigaciones permanecen en el ámbito experimental o dependen de recursos especializados, este trabajo ha dado lugar a una aplicación funcional que puede emplearse como herramienta formativa, de experimentación y como base para futuros desarrollos en operaciones de rescate.

7.3 Conclusiones

El desarrollo de este Trabajo de Fin de Grado ha permitido alcanzar los objetivos planteados inicialmente:

- Se ha implementado un algoritmo de patrullaje eficiente capaz de generar rutas óptimas para áreas arbitrarias, garantizando cobertura completa y continuidad en la trayectoria.
- Se ha construido un sistema totalmente automatizado, donde la gestión energética, el retorno a base y la reanudación de la misión ocurren sin intervención del usuario.
- Se ha diseñado una aplicación con interfaz accesible que elimina la necesidad de conocimientos técnicos avanzados, acercando la tecnología de simulación de drones a potenciales equipos de búsqueda y rescate.

En conjunto, el proyecto demuestra que es posible combinar rigor técnico con usabilidad, ofreciendo una herramienta que, aunque todavía simplificada en ciertos aspectos, constituye un paso intermedio entre los modelos académicos complejos y las necesidades operativas de los cuerpos de emergencia.

Como línea futura, resultaría de gran interés explorar la extensión del sistema a misiones multi-dron, incorporar sensores simulados más realistas (incluyendo modelos térmicos avanzados) y empaquetar la solución en un entorno multiplataforma más accesible. Estas mejoras ampliarían tanto la fidelidad de la simulación como su aplicabilidad en escenarios reales.

7.4 Conclusions

The development of this Final Degree Project has enabled the initial objectives to be achieved:

- An efficient patrolling algorithm has been implemented that is capable of generating optimal routes for arbitrary areas, ensuring complete coverage and continuity in the trajectory.
- A fully automated system has been built, where energy management, return to base, and mission resumption occur without user intervention.
- An application with an accessible interface has been designed that eliminates the need for advanced technical knowledge, bringing drone simulation technology closer to potential search and rescue teams.

Overall, the project demonstrates that it is possible to combine technical rigor with usability, offering a tool that, although still simplified in certain aspects, constitutes an intermediate step between complex academic models and the operational needs of emergency services.

As a future line of research, it would be of great interest to explore extending the system to multi-drone missions, incorporating more realistic simulated sensors (including advanced thermal models), and packaging the solution in a more accessible multi-platform environment. These improvements would increase both the fidelity of the simulation and its applicability in real-world scenarios.

BIBLIOGRAFÍA

[1] Yahoo Finance, Global Military Drone Market Size & Share Analysis, 2024. [En línea]. Disponible en: https://finance.yahoo.com/news/global-military-drone-market-size-130000306.html [Último acceso: 27 07 2025]

[2] Financial Times, Drone warfare: How autonomous drones are reshaping conflict, 2024. [En línea]. Disponible en: https://www.ft.com/content/ef4feab1-cb26-4631-b59d-e22e778b025c [Último acceso: 27 07 2025]

[3] Straits Research, Drone Market Forecast 2024–2033, 2024. [En línea]. Disponible en: https://straitsresearch.com/report/drone-market [Último acceso: 27 07 2025]

[4] DJI Enterprise, Drones in Search and Rescue: Efficacy Report, 2022. [En línea]. Disponible en: https://enterprise-insights.dji.com/en/en/learning-center/drone-efficacy-report-sar [Último acceso: 27 07 2025]

[5] arXiv, Reinforcement Learning for Human Search and Rescue Missions with UAVs, mayo 2024. [En línea]. Disponible en: https://arxiv.org/abs/2405.12800 [Último acceso: 27 07 2025]

[6] Simulation of Autonomous Flight of UAVs for Search and Rescue Missions. Alejandro-Daniel Díaz Román. [En línea]. Disponible en: https://docta.ucm.es/entities/publication/5819aa16-32ee-4d48-85ab-2fcedd291876

[7] Xu, Y.; Li, J.; Zhang, F. A UAV-Based Forest Fire Patrol Path Planning Strategy. Forests 2022, 13,1952. [En línea]. Disponible en: https://doi.org/10.3390/f13111952 [Último acceso: 28 06 2025]

[8] Energy-Efficient UAVs Coverage Path Planning Approach. Gamil Ahmed, Tarek Sheltami, Ashraf Mahmoud, Ansar Yasar. 2022. [En línea]. Disponible en: https://www.techscience.com/CMES/v136n3/51821 [Último acceso: 28 06 2025]

[9] Early Forest Fire Detection Using Drones and Artificial Intelligence. Diyana Kinaneva, Georgi Hristov, Jordan Raychev and Plamen Zahariev. 2019. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/8756696 [Último acceso: 29 06 2025]

[10] FlytBase, How Drones Are Powering Law Enforcement Agencies, 2025. [En línea]. Disponible en: https://www.flytbase.com/blog/drones-for-law-enforcement [Último acceso: 27 07 2025]

[11] G. Zhang, H. Tang, Y. Zhou, Y. Li y L. Zhang, "Survey on Path Planning Algorithms for Unmanned Aerial Vehicles," Remote Sensing, vol. 2024, art. no. 0474, 2024. [En línea]. Disponible en: https://spj.science.org/doi/10.34133/remotesensing.0474 [Último acceso: 30 07 2025]

[12] A. Kumar, V. K. Singh, A. Mishra y S. Deb, "AgilePilot: Deep Reinforcement Learning-based UAV Agent for Real-Time Motion Planning," arXiv preprint, arXiv:2502.06725, 2025. [En línea]. Disponible en: https://arxiv.org/html/2502.06725v1 [Último acceso: 30 07 2025]

[13] J. Xiao, S. Liu, C. Huang, Y. Sun, Y. Liu y H. Zhao, "FM-Planner: Foundation Model Guided Path Planning for Autonomous Drone Navigation," arXiv preprint, arXiv:2505.20783, 2025. [En línea]. Disponible en: https://arxiv.org/abs/2505.20783 [Último acceso: 30 07 2025]

[14] NBC Los Angeles, "Heat-Detecting Drone Helps Find Missing 78-Year-Old Man in Malibu," NBC Los Angeles, 2024. [En línea]. Disponible en: https://www.nbclosangeles.com/news/local/drone-rescue-missing-man-malibu/3589271/ [Último acceso: 30 07 2025]

[15] Y. Wang, F. Gao, L. Meng y D. Wang, "AutoSOS: Towards Multi-UAV Systems Supporting Maritime Search and Rescue with Lightweight AI and Edge Computing," Procedia Computer Science, vol. 2025, art. no. 1195103625000035. [En línea]. Disponible en: https://www.researchgate.net/publication/341231303_AutoSOS_Towards_Multi-UAV_Systems_Supporting_Maritime_Search_and_Rescue_with_Lightweight_AI_and_Edge_Computing [Último acceso: 30 07 2025]

[16] Specs - DJI Mavic 3 Enterprise - DJI Enterprise Disponible en: https://enterprise.dji.com/mavic-3-enterprise/specs

[17] EVO II Dual 640T Rugged Bundle V3 – Autel Robotics. Disponible en: https://shop.autelrobotics.com/products/evo-ii-dual-640t-rugged-bundle-v3

[18] Specs - Matrice 30 Series - DJI Enterprise. Disponible en: https://enterprise.dji.com/matrice-30/specs

[19] Skydio X10 Technical Specs | Skydio. Disponible en: https://www.skydio.com/x10/technical-specs

[20] Autel Dragonfish VTOL Drone Platform – Long Endurance UAS – Advexure. Disponible en: https://advexure.com/pages/autel-dragonfish

[21] eBee X mapping drone - Drones | AgEagle Aerial Systems Inc. Disponible en: https://ageagle.com/drones/ebee-x/

[22] Actively Tethered Drones for Emergency Response I Fotokite Disponible en: https://fotokite.com/

[23]elistair-unveils-orion-2-tethered-drone-for-defense-and-security. Disponible en: https://elistair.com/company-news/product-releases/elistair-unveils-orion-2-tethered-drone-for-defense-and-security/

[24] Thermal Image Tracking for Search and Rescue Missions with a Drone, Seokwon Yeom, 2024. [En línea]. Disponible en: https://www.mdpi.com/2504-446X/8/2/53 [Último acceso: 24 06 2025]

[25] M. Qiu, L. Zheng y Y. Yang, "A Lightweight YOLO-Based Object Detection Framework for UAVs in Infrared Search and Rescue Missions," Infectious Diseases Now, vol. 55, no. 1, 2025. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1195103625000035 [Último acceso: 30 07 2025]

[26] A Lightweight Real-Time Infrared Object Detection Model Based on YOLOv8 for Unmanned Aerial Vehicles [En línea]. Disponible en: https://www.mdpi.com/2504-446X/8/9/479 [Último acceso: 24 06 2025]

[27] Thermography - Wikipedia [En línea]. Disponible en: https://en.wikipedia.org/wiki/Thermography [Último acceso: 24 06 2025]

[28] Centum, "Lifeseeker – A mission-critical system for search and rescue," Centum Solutions, 2023. [En línea]. Disponible en: https://centum.com/en/products/lifeseeker [Último acceso: 02 07 2025]

[29] BRINC, "Responder Drone – Rapidly deployable, easy-to-use drone for first responders," BRINC Drones, 2023. [En línea]. Disponible en: https://brincdrones.com/responder [Último acceso: 02 07 2025]

[30] FlytBase, "Drone automation for public safety – FlytNow," FlytBase, 2023. [En línea]. Disponible en: https://www.flytbase.com/public-safety [Último acceso: 02 07 2025]

[31] Y. Liu and S. Shen, "Augmented reality interaction interface for autonomous drone," arXiv preprint arXiv:2008.02234, 2020. [En línea]. Disponible en: https://arxiv.org/abs/2008.02234 [Último acceso: 02 07 2025]

[32] D. A. G. Sánchez, J. A. G. Fajardo, and J. C. S. Parra, "ICARUS: UAV system with Android app for search and rescue operations," arXiv preprint arXiv:2308.14994, 2023. [En línea]. Disponible en: https://arxiv.org/abs/2308.14994 [Último acceso: 02 07 2025]

[33] Maps SDK for Unity: 3D worlds, AR, & POIs | Mapbox. Disponible en: https://www.mapbox.com/unity

[34] DJI Matrice 400 - Specs - DJI Enterprise. Disponible en: https://enterprise.dji.com/matrice-400/specs

[35] GitHub - AngusJohnson/Clipper2: Polygon Clipping and Offsetting - C++, C# and Delphi. Disponible en: https://github.com/AngusJohnson/Clipper2

APÉNDICES

Apéndice A - Scripts y componentes referenciados en la memoria

A.1 Cliente de comunicación (DroneWSClient.cs)

Este componente, desarrollado en Unity (C#), implementa un cliente WebSocket encargado de la comunicación con el servidor. La clase principal (DroneWSClient) mantiene la conexión (WebSocket ws), recibe telemetría en formato JSON y envía comandos hacia el simulador.

Telemetría recibida

La estructura de datos DroneData define los parámetros deserializados:

- lat, lon, alt: coordenadas GPS y altitud.
- yaw: orientación en radianes.
- groundspeed: velocidad respecto al suelo.
- mode: modo de vuelo actual.
- armed: estado de armado.
- battery: objeto BatteryData con voltage, current, level y eta_min.

La información se actualiza en cada mensaje recibido en OnWsMessage y se refleja en la interfaz mediante los elementos infoTextL e infoTextR.

Comandos disponibles

El cliente expone métodos públicos para enviar instrucciones codificadas en JSON al servidor, todos ellos precedidos por un campo command:

- SendSetSpeed(float speed) → { "command": "set_speed", "speed": <valor> }
- SendMission(IEnumerable<MissionWaypoint> waypoints) → { "command": "upload_mission", "waypoints": [...] }

- SendSetHome(double lat, double lon, double alt) → { "command": "set_home", "lat": ..., "lon": ..., "alt": ... }
- SendReturnToLaunch() → { "command": "return_to_launch" }
- SendSetBatteryLevel(float level) → { "command": "set_battery_level", "level": ... }
- SendResumeMission() → { "command": "resume_mission" }

Cada método comprueba previamente la conexión y, en caso de estar activa, envía el mensaje serializado al servidor.

Gestión de la conexión

La conexión se establece mediante Connect(), que inicializa el WebSocket con la URL configurada. Los eventos principales (OnWsOpen, OnWsMessage, OnWsError, OnWsClose) gestionan la apertura, recepción de datos, errores y cierre de la comunicación. Al destruir el objeto (OnDestroy()), se cierran y desuscriben todos los callbacks asociados.

A.2 Selección de zona y generación de ruta (ZoneSelector.cs)

El componente ZoneSelector implementa la lógica para la definición del polígono de patrullaje, la construcción de rutas de cobertura y su exportación como misión.

Métodos principales

Interacción y edición del polígono:

- ToggleDrawingMode(bool) → activa/desactiva el modo de dibujo; al cerrarse con ≥3 nodos invoca ClosePolygon().
- ToggleBase(bool) → habilita la colocación de la base de operaciones.
- AddPoint(Vector3) → añade un nodo al polígono y re-nivela todos los vértices para mantener coherencia visual.
- DeleteNodes() y DeletePath() → eliminan elementos gráficos y resetean las colecciones.

Cierre y validación del polígono:

- ClosePolygon() → cierra el área definida, ajusta vértices extremos si están próximos y genera las paredes con CreateWalls(). Posteriormente invoca GenerateCoveragePath().

Generación de ruta:

- GenerateCoveragePath() → implementa la estrategia de cobertura tipo lawn-mower, incluyendo:
    - Construcción de anillos interiores mediante Clipper2Lib [35].
    - Alternancia de sentido de pasadas para minimizar desplazamientos.
    - Re-muestreo con ResamplePath().
    - Simplificación con DouglasPeucker().
    - Inserción de puntos intermedios por pendiente (AddSlopeBreaks()).
    - Densificación según distancia y desnivel (DensifyForHeight()).
    - Proyección en 3D (BuildPath3D()), aplicando el suavizado controlado por adaptationIntensity.
- El resultado se almacena en lastCoveragePath y se visualiza con LineRenderer.

Exportación de misión:

- ExportMission() → convierte cada punto de lastCoveragePath en waypoints (MissionWaypoint), incorporando elevación del terreno vía map.QueryElevationInUnityUnitsAt().
- Si la ruta excede maxWaypoints, aplica un muestreo proporcional para ajustarla.
- Finalmente envía la misión al servidor a través de client.SendMission().

Funciones auxiliares:

- SampleGroundY(Vector2) → obtiene la cota del terreno en coordenadas Unity.
- DouglasPeucker(List<Vector2>, float) → aplica el algoritmo de simplificación.

- AddSlopeBreaks(...), DensifyForHeight(...), ResamplePath(...) → rutinas de ajuste de la trayectoria para garantizar continuidad y seguridad.
- ToPath64(...) y ToVector2(...) → conversión entre estructuras propias de Unity y Clipper2Lib.

Apéndice B - Repositorio del proyecto

El código fuente desarrollado en este Trabajo de Fin de Grado se encuentra disponible en un repositorio en línea, organizado en tres bloques principales:

- App_code: scripts en C# de la aplicación en Unity, que incluyen la lógica de control del dron, el algoritmo de generación de rutas, la interfaz y los módulos de simulación de eventos.
- Server_code: servidor de comunicación en Python, encargado de la transmisión de telemetría y de la simulación energética de la batería.
- Simulator_code: scripts auxiliares para la configuración y ejecución del simulador DroneKit-SITL junto con ArduPilot.

El repositorio puede consultarse en la siguiente dirección:

https://github.com/jeramauni/TFG-Drone-Simulation-Code.git

Además del código, en este otro repositorio se incluye un vídeo demostrativo que muestra la ejecución de la aplicación y el flujo completo de una misión simulada:

https://drive.google.com/file/d/1tBPUEOgqvQTbyVQFg8QRZ0XXt_Z78tkd/view?usp=sharing