UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de sistema de guiado para multicópteros en base a datos de contaminación ambiental

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Paul Mínguez, Javier

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

Director/a Experimental: WUBBEN, JAMIE

CURSO ACADÉMICO: 2021/2022

Resumen

El cambio climático es una de las principales problemáticas a las que se enfrenta el ser humano, y reducir la contaminación es una de las acciones que se pueden tomar para combatirlo. Por otro lado, durante los últimos años ha aumentado el interés y el uso de vehículos aéreos no tripulados, comúnmente conocidos como drones. En este proyecto se plantea unir estas dos ideas con el fin de utilizar dichos drones para la monitorización de los niveles de contaminación en un área definida por el usuario. Para este propósito, se ha desarrollado un sistema de guiado automático de multicópteros, de tal forma que sean capaces de obtener los datos de contaminación de dicha área de una forma eficiente, autónoma y precisa. El movimiento de estos vehículos se define durante la toma de medidas, en función de los datos que se miden en cada momento. Este movimiento hace hincapié en las zonas de máxima concentración de contaminantes, ofreciendo información precisa sobre los focos de dicha contaminación. Así pues, tras cada ejecución del sistema, se obtiene un gradiente entre los puntos de máxima y mínima concentración.

Palabras clave: ArduSim, Drones, Contaminación

Resum

El canvi climàtic és una de les principals problemàtiques a què s'enfronta l'humanitat, i reduir la contaminació és una de les accions que es poden prendre per combatre'l. D'altra banda, durant els últims anys ha augmentat l'interès i l'ús de vehicles aeris no tripulats, coneguts com a drons. En aquest projecte es planteja unir aquestes dues idees per tal d'utilitzar aquests drons per a la monitorització dels nivells de contaminació en una àrea definida per l'usuari. Per a aquest propòsit, s'ha desenvolupat un sistema de conducció automàtica de multicòpters, de manera que puguen obtenir les dades de contaminació d'aquesta àrea de manera eficient, autònoma i precisa. El moviment d'aquests vehicles es defineix en el moment que prenen les mesures, en funció de les dades mesurades a cada moment. Aquest moviment posa l'accent en les zones de màxima concentració de contaminants i ofereix informació precisa sobre els focus d'aquesta contaminació. Així, després de cada execució del sistema, s'obté un gradient entre els punts de màxima i mínima concentració.

Paraules clau: ArduSim, Drons, Contaminació

Abstract

Climate change is one of the main problems that humanity is facing, and reducing pollution is one of the actions that can be taken to fight it. On the other hand, in recent years the interest and use of unmanned aerial vehicles, commonly known as drones, has increased. This project proposes to join these two ideas in order to use these drones to monitor pollution levels in areas defined by the user. For this purpose, an automatic multicopter guidance system has been developed in such a way that they are capable of obtaining pollution data from an area in an efficient, autonomous and precise way. The movement of these vehicles is

defined when taking measurements, based on the data that is measured at each moment. This movement emphasizes the areas of maximum level of pollution, offering precise information on the contamination sources. Thus, after each execution of the system, a gradient is obtained between the points of maximum and minimum concentration.

Key words: ArduSim, Drones, Pollution

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

CAPÍTULO 1
Introducción

1.1 Motivación

En la actualidad, el medio ambiente es (o debería ser) una de las principales preocupaciones del ser humano. Teniendo en cuenta la situación global actual, debemos tomar todas las medidas necesarias para reducir la huella de carbono de la humanidad en el planeta. Con esta idea en mente, resulta interesante poder estudiar la contaminación existente en diferentes áreas para cuantificar los niveles de contaminación. Por ejemplo, no encontramos el mismo nivel de contaminación en un área urbana que en un área rural.

Por otro lado, también resulta interesante la localización de los puntos con contaminación máxima en una determinada área. De esta forma se pueden encontrar los focos que generan esta contaminación, información que se puede utilizar para tomar decisiones más informadas y mejor dirigidas. Esto no solo puede ayudar a que los trabajos de corrección tengan más impacto, sino que también puede disminuir la cantidad de trabajo de poca utilidad.

La idea detrás de esto es evolucionar de un modelo en el que se toman medidas de contaminación en momentos puntuales con fines estadísticos, a la sensorización de esta medición. Esto supone pasar de medidas diarias a una medición constante usando sensores. Con la aparición de estos sensores, sería posible implantar un sistema de alarmas que facilite la localización de los contaminantes con vistas a trabajar para reducir el impacto que este problema pueda tener.

Así pues, este proyecto puede ser de ayuda en dos casos diferentes:

- **La localización de los puntos más contaminantes en una zona determinada.** Por ejemplo, en una zona industrial con diversas empresas trabajando juntas, se podría localizar el punto del cual proviene la mayor carga de contaminantes. Esto conllevaría una única ejecución del vuelo de los drones para la recogida de datos, y una actuación posterior en consecuencia.

- **La monitorización continua de un área.** Por ejemplo, en una zona urbana con muchos residentes, se podría tener un dron realizando vuelos continuos para tomar medidas frecuentes. De esta forma se podría estudiar la evolución del punto de máxima contaminación, es decir, si cambia de lugar o sufre una evolución continua. También se podría controlar que los nive-

les de contaminación no pasen nunca de ciertos umbrales, actuando en el momento en el que se detecte un aumento de sustancias contaminantes.

1.2 Objetivos

El objetivo de este proyecto es desarrollar un sistema de vuelo autónomo para drones en función de la contaminación ambiental. Estos drones irían equipados con un sensor de contaminación y recogerían datos repetidamente durante su vuelo. Este vuelo debe estar guiado usando en el algoritmo PdUC-D [2], siendo implementado y probado en el simulador de vuelo ArduSim [3].

Para lograr este objetivo general podemos definir 6 objetivos específicos:

1. Investigar y entender la situación tecnológica actual, para una mejor contextualización del proyecto.
2. Estudiar y comprender el funcionamiento del simulador ArduSim. Familiarizarse con él para saber desarrollar un módulo capaz de ejecutar las pruebas necesarias de este proyecto. Este módulo ha de estar completamente cohesionado con las diferentes partes del simulador para facilitar su comprensión y posterior modificación.
3. Estudiar y comprender el algoritmo PdUC [1] y su versión discreta [2], ya que será el algoritmo que definirá el movimiento de los drones.
4. Especificar los requisitos que este nuevo módulo de ArduSim va a necesitar, facilitando así su posterior desarrollo.
5. Desarrollo del módulo basado en PdUC-D, y validación de su correcto funcionamiento.
6. Pruebas de eficiencia y calidad de los datos obtenidos. Una vez implementado el algoritmo, resulta interesante compararlo con diferentes modalidades de vuelo, ya sea vuelo aleatorio o un vuelo recorriendo el área entera. De esta forma se puede cuantificar la eficiencia del algoritmo, y se puede comparar si el error, en comparación con los datos reales, es significativo o no.

1.3 Estructura de la memoria

Este proyecto está compuesto por 8 capítulos principales:

- El capítulo 1, el actual, es el capítulo en el que se han presentado los puntos de partida a partir de los cuales ha surgido este proyecto, y sus objetivos.
- En el capítulo 2 se expone una visión general de la situación tecnológica y legal actual, el estado del arte. Aquí se explicará el concepto de UAV, y se hablará de las tecnologías y normativas que guardan relación con el presente trabajo.

- En el capítulo 3 se hablará en detalle del simulador de vuelo utilizado para este proyecto, ArduSim. Se expondrá de forma general cómo está implementado ArduSim y su estructura modular, lo que hace posible el desarrollo independiente de sus protocolos.
- En el capítulo 4 se expondrá el algoritmo implementado en el proyecto. Primero se explicará su base más puramente teórica (PdUC), y posteriormente su versión orientada a llevarla a la práctica (PdUC-D), que será la utilizada en este desarrollo.
- En el capítulo 5 se hará un análisis del problema y se estudiará de forma teórica la que posteriormente será la estructura del desarrollo. También se estudiarán los usuarios a los que puede ir dirigida la aplicación, y su forma de interactuar con la misma.
- En el capítulo 6 se hablará en profundidad del proceso de desarrollo del algoritmo. También se explicarán los puntos importantes del movimiento de los drones.
- En el capítulo 7 se harán pruebas en distintos niveles de implementación con el objetivo de comparar la eficiencia del protocolo. De esta forma se podrá hacer una comparativa en cuanto a calidad de los datos obtenidos y el tiempo de ejecución del protocolo.
- Por último, en el capítulo 8, se recopilarán las conclusiones una vez finalizado el proyecto, y se estudiarán posibles mejoras y trabajos futuros del mismo.

Tras los capítulos principales de la memoria se encuentra la bibliografía y anexo de objetivos de desarrollo sostenible (ODS).

CAPÍTULO 2
Estado del arte

2.1 Vehículos aéreos no tripulados (UAV)

Los vehículos aéreos no tripulados, más conocidos como drones o UAV (del inglés *Unmanned Aerial Vehicle*) son dispositivos voladores que no tienen piloto a bordo, por lo que pueden estar controlados remotamente, o realizar vuelos programados.

Figura 2.1: Multicóptero con cámara equipada.

2.1.1. Clasificación de drones

Según su uso, los UAV pueden clasificarse en dos grupos [6]:

- **Drones de uso civil**
  En este grupo se pueden encontrar drones de uso más cercano al recreativo. Se pueden encontrar drones que son incluso juguetes para niños pequeños, pensados para vuelo en interiores. En este grupo destacan los multicópteros de uso adulto amateur, donde se encuentran dispositivos de buena calidad equipados con todo tipo de sensores, siendo las cámaras de vídeo las predominantes. Como ejemplo, se puede observar el dron de la figura 2.1. Debido

a su popularidad, los UAV de este grupo están recibiendo grandes disminuciones de precio, facilitando su llegada al público en general.

- **Drones de uso militar**
  Estos son los precursores de lo que se conoce actualmente como un dron. Tal y como pasa con multitud de tecnologías que están actualmente al alcance del usuario general, los drones tienen origen militar, siendo usados como vehículos de combate o de espionaje.

Según su tipo de ala, se pueden clasificar en:

- **Drones de ala fija**
  Estos drones utilizan la aerodinámica para volar. Tienen un funcionamiento similar a los aviones de pasajeros; usan un sistema a propulsión, y se mantienen en el aire gracias a la fuerza de sustentación de las alas que el aire genera por la velocidad que obtienen. Un ejemplo se puede encontrar en la figura 2.2.

Figura 2.2: Dron de ala fija.

- **Drones de ala rotatoria o multicópteros**
  Estos son los drones hacia los que va dirigido este proyecto. Son, con diferencia, los más comunes. En su funcionamiento son similares a los helicópteros, es decir, cuentan con unas hélices que les permiten despegar y moverse de forma vertical, así como mantenerse en una altura deseada. Son los UAV más versátiles por la alta precisión en sus movimientos y su capacidad de quedarse estáticos en el aire. Un ejemplo de este tipo de drones se encuentra en la figura 2.1. Concretamente, el de la figura se trata de un cuadricóptero, pero los drones de ala rotatoria se pueden subdividir en varios subgrupos, dependiendo del número de motores que tengan:

  - Tricópteros (3 motores)
  - Cuadricópteros (4 motores)
  - Hexacópteros (6 motores)
  - Octacópteros (8 motores)

2.1.2. Drones en las futuras *Smart Cities*

Durante los últimos años ha crecido el interés y la investigación del uso de los drones en lo que se denomina Ciudades Inteligentes o *Smart Cities* [5]. Esta tendencia está teniendo lugar al mismo tiempo que aumenta el interés en la domótica (Hogar Inteligente), que consiste en sensorizar el hogar para tener controladas, en todo momento, variables como la temperatura o la humedad.

De forma análoga, se busca sensorizar las ciudades, pero debido a la gran superficie a medir, tener pequeños dispositivos voladores que se muevan libremente por ella resulta de gran utilidad. Los drones son capaces de adaptarse a los numerosos cambios que puede experimentar una ciudad, ofreciendo una flexibilidad que difícilmente se puede conseguir con otros medios. De esta forma se pueden tener controladas variables como el tráfico, la temperatura o, como es el caso de este proyecto, la contaminación.

Pero los drones en las *Smart Cities* no solo pueden servir como sensores móviles, sino que también pueden aportar otros usos muy diversos, como por ejemplo:

- **Entrega de paquetes**
  Gracias a su desplazamiento aéreo y al uso de algoritmos para recorrer la ciudad eficientemente, los UAV superan a los vehículos de combustión, reduciendo la contaminación y aumentando la velocidad de reparto.
- **Transporte de pasajeros**
  Aunque este uso todavía está en un estado muy temprano, ya hay prototipos funcionales, y se investiga para darles esta utilidad en un futuro.

2.2 Medidas de contaminación

La contaminación se puede encontrar presente en diversas partes del planeta, así como en los diversos estados de la materia. En este proyecto, la contaminación que se busca medir es la contaminación atmosférica, es decir, la que se encuentra en el aire.

La contaminación atmosférica consiste en la presencia de materia o energía en el aire que no debería encontrarse ahí naturalmente, o al menos en una concentración bastante más reducida. La presencia de estas sustancias puede suponer un riesgo para los seres humanos y la biodiversidad. Esta contaminación está teniendo cada vez más impacto sobre los ecosistemas, ya que desde la Revolución Industrial no ha dejado de aumentar.

Es por esto que se han desarrollado procedimientos estándar [8] para medir la concentración de dichas sustancias contaminantes, junto con los denominados *sistemas de medición de la calidad del aire* (SMCA) [10], para tomar medidas y determinar si el aire estudiado se considera contaminado o no. De esta forma, un SMCA genera datos sobre la concentración de los contaminantes y, posteriormente, estos datos se comparan con los estándares para determinar si se han rebasado. Es muy importante que, durante la obtención de datos, se sigan las normas y procedimientos de control y aseguramiento de calidad, para obtener datos correctos y sin adulterar.

2.2.1. Estándares límite de contaminantes

La tabla 2.1 muestra un ejemplo de estos estándares. En esta tabla se observan varios límites descritos en la Directiva de la UE [7], pero es competencia de cada país adaptarlos y decidir sus propios límites. En el caso de España, pasa a ser competencia de las comunidades autónomas.

Tabla 2.1: Contaminantes y los límites definidos por la UE.

Contaminante	**Período**	**Valores límite de la Directiva µg/m^3**	**Nº de veces en un año que se pueden superar las normas de la UE**
Dióxido de nitrógeno (NO_2)	1 año	40	-
	1 hora	200	18
Ozono (O_3)	1 hora	180	25
Partículas PM_{10}	1 año	40	-
	24 horas	50	35
Partículas $PM_{2,5}$	1 año	17	-
Dióxido de azufre (SO_2)	24 horas	125	3
	1 hora	350	24
Plomo (Pb)	1 año	0,35	-
Monóxido de carbono (CO)	8 horas	10000	-

2.2.2. Sensores de contaminación

Para cada sustancia contaminante es necesario utilizar una técnica de medición específica según las cualidades de dicha sustancia. Estos son los métodos que se utilizan en los sensores de contaminación para detectar cada contaminante [9] [10]:

- **Dióxido de nitrógeno (NO_2)**
  - Método: Espectroscopia de fluorescencia.
  - Toma de medidas: La muestra excitada puede emitir el exceso de energía excitada. Esta energía se puede medir en forma de fotones.
- **Ozono (O_3)**
  - Método: Absorción ultravioleta.
  - Toma de medidas: La medición se realiza mediante el cálculo de la absorción de radiación ultravioleta por parte de las moléculas de ozono.
- **Partículas PM_{10} y $PM_{2,5}$**
  - Método: Recolección de partículas.
  - Toma de medidas: Se utilizan filtros que no permiten el paso de estas partículas, para posteriormente determinar la masa utilizando el peso.

- **Dióxido de azufre (SO_2)**
  - Método: Espectrofotometría.
  - Toma de medidas: Se forman soluciones coloreadas mediante la mezcla de los reactivos con los contaminantes para medir la fluorescencia de estas moléculas.
- **Plomo (Pb)**
  - Método: Absorción atómica.
  - Toma de medidas: Se somete la muestra a radiación para posteriormente medir la radiación absorbida.
- **Monóxido de carbono (CO)**
  - Método: Absorción infrarroja.
  - Toma de medidas: La medición se realiza mediante el cálculo de la absorción de radiación infrarroja por parte de las moléculas de monóxido de carbono.

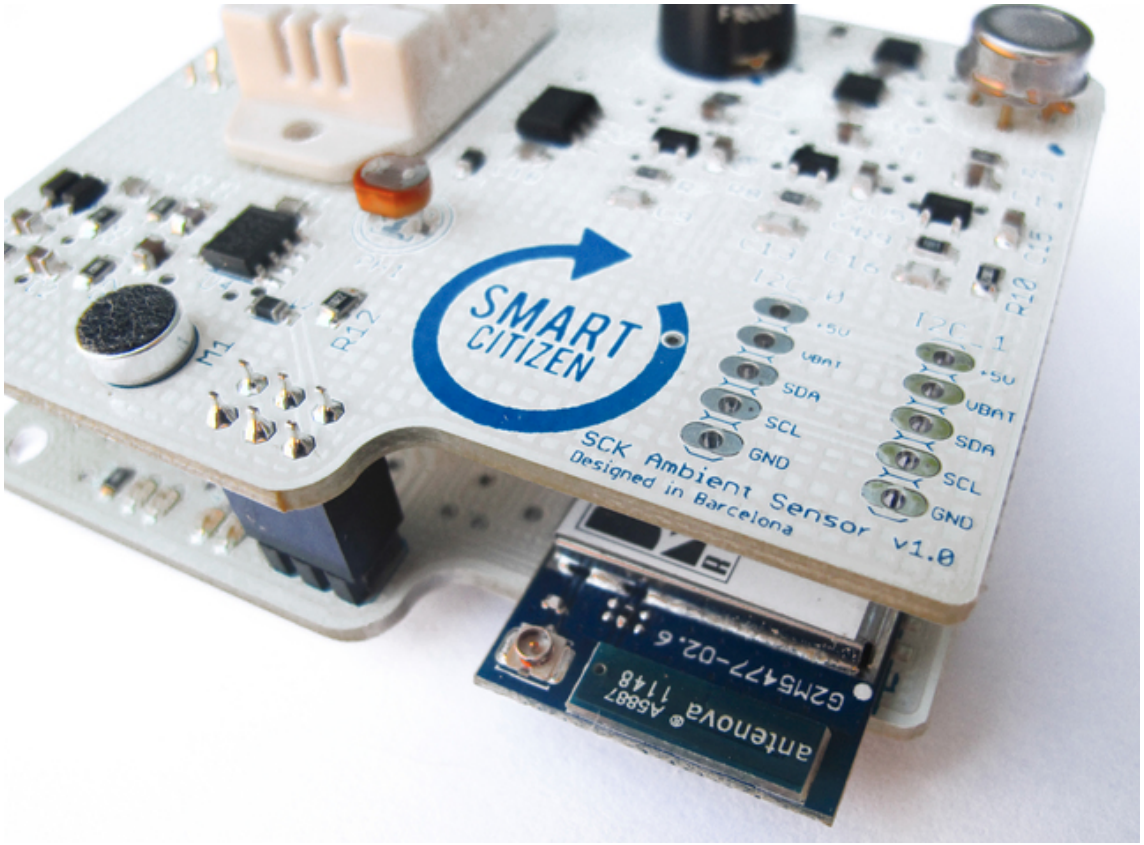


Figura 2.3: El sensor 'Smart Citizen' que mide la contaminación urbana. [12] [13]

En la figura 2.3 se muestra un ejemplo de placa con capacidades de sensorización múltiple, incluyendo partículas PM_{10} y $PM_{2,5}$, entre otros.

2.3 Simuladores de vuelo

Si bien es cierto que el precio de los drones está disminuyendo con los años y que están llegando a manos del público general, utilizar multicópteros reales puede suponer varios inconvenientes [14]:

- **Las leyes del país en el que se quiere volar** [15]
  La extensión del uso de estos pequeños vehículos voladores ha traído consigo una dura legislación al respecto. No resulta tan sencillo como comprarse un dron y hacerlo volar, sino que es obligatorio tener la habilitación correspondiente como piloto de drones para uso profesional, entre otros muchos requisitos.

- **Las condiciones meteorológicas**
  Aunque la tecnología de los UAV esté en mejora continua y cada vez sean capaces de afrontar más situaciones adversas, una mala meteorología supone siempre un problema para el vuelo de estos vehículos. Además, puede entorpecer algunos de sus usos al afectar a la medición de sus sensores.

- **La batería**
  Es un hecho que la batería de cualquier dispositivo siempre va a ser limitada. Es por esto que el uso de drones físicos siempre va a tener una limitación en cuanto al tiempo de uso y tiempo de experimentación, que irá ligada a la duración de la batería del mismo.

- **Experimentación con varios drones simultáneamente**
  Trabajar con un único dron puede ser asequible, pero cuando se necesita comprobar el funcionamiento de un enjambre de drones trabajando conjuntamente, la dificultad aumenta exponencialmente. Resulta mucho más complicado controlar un buen funcionamiento de los UAV, y comprobar que no sucede ningún percance.

Es por todo esto que resulta interesante el uso de un simulador de vuelo. Usar un simulador puede disminuir la parte recreativa de trabajar con drones, pero resultan muy útiles a la hora de realizar estudios. De todas formas, el uso de un simulador no necesariamente significa que no se vayan a usar drones reales; un simulador puede hacer de paso intermedio entre una fase inicial del protocolo donde tal vez no sea tan estable como se desea, y una fase final. Puede hacer de entorno de pruebas para depurar el protocolo desarrollado hasta que se considere que se tiene un producto suficientemente pulido. De esta forma se evitará el mayor número posible de incidencias.

CAPÍTULO 3
ArduSim

En este capítulo se hablará en profundidad del simulador de vuelo ArduSim [3]. Este simulador ha sido programado en la Universitat Politècnica de València (UPV), concretamente en el Departamento de Informática de Sistemas y Computadores (DISCA) por el grupo GRC. El código fuente se puede encontrar en en GitHub, donde se realiza el control de versiones [4].

ArduSim surge bajo la necesidad de simular varios drones a la vez. En la figura 3.1 se puede observar su interfaz y una simulación en la que muchos drones trabajan conjuntamente para dibujar el símbolo de un átomo.

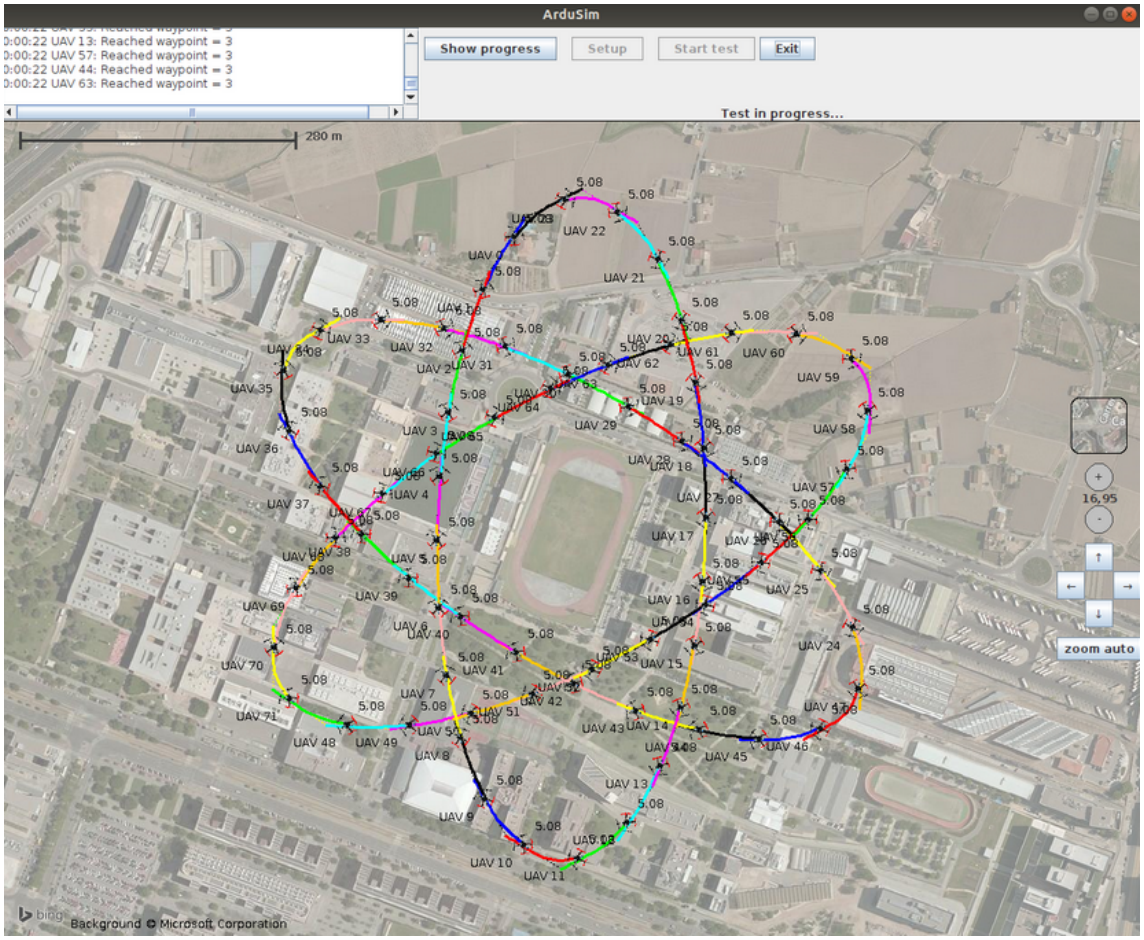


Figura 3.1: ArduSim.

3.1 Estructura y funcionamiento

ArduSim está programado en Java, y tiene una estructura modular. Está basado en el simulador SITL [16], el cual permite la simulación de diferentes tipos de UAV con bastante precisión. El problema principal de SITL es que solo permite la simulación de un único dron, por lo que ArduSim se encarga de actuar como una capa superior bajo la cual se ejecutan tantas instancias de SITL como sean necesarias. Se ejecuta una instancia por cada UAV, siendo ejecutadas en procesos independientes, permitiendo así la simulación de enjambres de drones.

Cada UAV virtual está compuesto por un agente encargado de controlar el dron. Los agentes a su vez incluyen un hilo, el controlador, que es el encargado de enviar los comandos necesarios al multicóptero, así como de recibir la información que este genera. Es decir, es el controlador el que se comunica con la lógica del protocolo que estamos ejecutando.

Este simulador también se encarga de las comunicaciones entre UAV (*drone-to-drone* o D2D). Lo realiza mediante tecnología WiFi, bajo el estándar IEEE 802.11a (Banda de 5GHz). Es en los protocolos donde se configura esta comunicación. Requiere de un mínimo de dos hilos, uno para el envío de paquetes (*Talker*) y otro para la recepción (*Listener*).

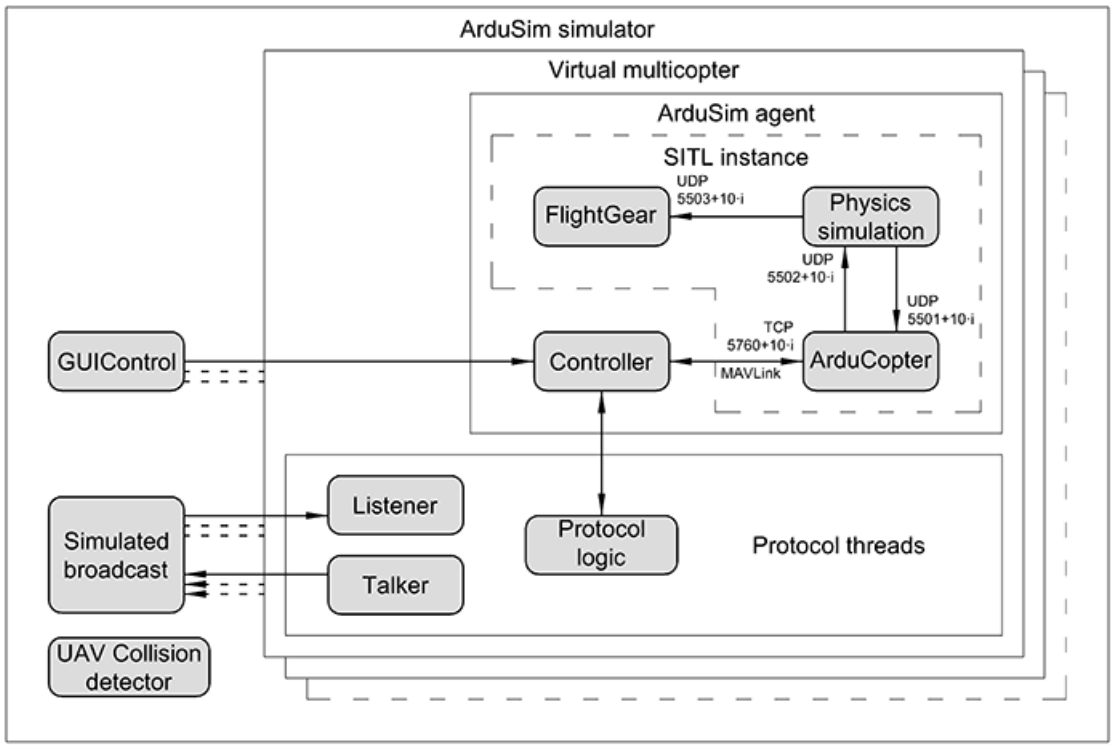


Figura 3.2: Estructura interna del simulador ArduSim.

3.1.1. ArduSim con UAVs reales

Adicionalmente, ArduSim también está implementado de tal forma que se pueda instalar en una Raspberry Pi conectada a un multicóptero, y este funcio-

naría de forma prácticamente idéntica a un dron simulado. En este caso se cambiaría la instancia SITL por un UAV real, pero manteniendo el controlador como parte del agente ArduSim. El controlador seguiría siendo el encargado de la comunicación con el protocolo. En la figura 3.3 se puede observar la estructura del simulador cuando se utilizan drones reales.

Además, en este caso aparece un nuevo concepto, el *PC Companion*. Como su nombre indica, se trata de un PC externo que actuará como receptor de los mensajes y logs que el UAV necesite enviar. No se profundizará más en este apartado, ya que queda fuera del alcance de este proyecto.

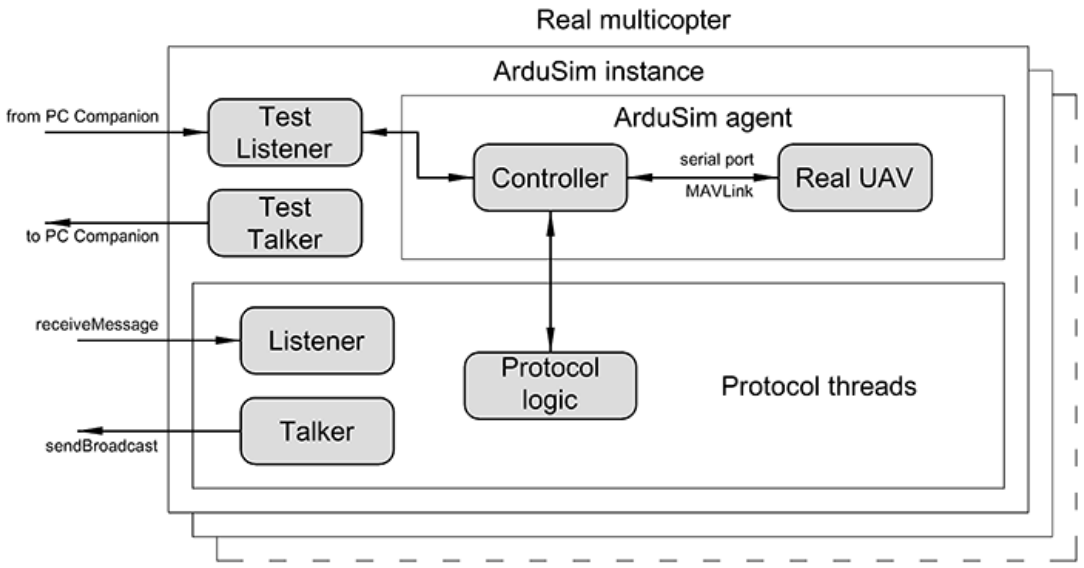


Figura 3.3: Estructura interna de ArduSim con drones reales.

3.2 Implementación de protocolos

Los protocolos en ArduSim son las piezas de código en las que se especifica el comportamiento que los drones deben tener. Se programan de forma totalmente modular al código propio del simulador, haciéndolos completamente independientes del mismo. De esta forma, se puede optar por no disponer de todos los protocolos instalados, o crear uno nuevo sin modificar nada de los anteriores ni del simulador.

Dentro del proyecto de Java, se puede diferenciar dos rutas en las que se guardan datos de cada protocolo:

- **ArduSim/src/main/resources/protocols**
  En esta ruta se crea un directorio por cada protocolo con su propio nombre. Aquí es donde se almacenarán archivos de datos de cada protocolo, como por ejemplo archivos *.properties* con las variables de ejecución por defecto. También será en esta carpeta donde se almacenen los ficheros *.fxml* en caso de requerir un entorno gráfico para modificar las variables de ejecución. En el caso específico de este proyecto, también se utilizará esta ruta para almacenar los datos que leerá el sensor de contaminación simulado.

- **ArduSim/src/main/java/com/protocols/**
  Será en esta ruta en la que se almacene toda la lógica de la aplicación. Como en el caso anterior, se crea un directorio por cada protocolo con su propio nombre, y a su vez este directorio puede contener hasta 3 subdirectorios:

  - **gui**. Aquí es donde se almacenará toda la lógica del entorno gráfico de cada protocolo, en caso de necesitarlo. En los protocolos existentes, está desarrollada usando javafx.
  - **logic**. Aquí se encontrará el grueso del protocolo. Cada protocolo debe tener implementado un *ProtocolHelper.java*, que hará de puente entre el protocolo y el código interno de ArduSim. También será en este directorio donde se almacenen los *ProtocolThread.java* en caso de utilizarlo. Básicamente será en esta carpeta donde se encuentre la funcionalidad de los drones.
  - **pojo**. Será en esta carpeta donde se almacenen las clases de java útiles para la lógica del protocolo.

3.2.1. Protocolos disponibles

Actualmente, en ArduSim se puede encontrar una variada selección de protocolos ya implementados y funcionales:

- **Mission.** Este protocolo permite que un grupo de multicópteros sigan una misión previamente definida.
- **MBCAP.** Este protocolo está implementado para el estudio de técnicas de prevención de colisiones entre varios multicópteros que siguen una misión previamente definida [17].
- **MUSCOP.** Con este protocolo se consigue que un enjambre de drones sigan una misión que se encuentra almacenada en uno de ellos, el dron máster. Los multicópteros se mueven siguiendo una formación, y es resistente al fallo de cualquiera de los drones que forman el enjambre [18].
- **FollowMe.** En este caso, un enjambre sigue a un dron controlado manualmente por un piloto [19].
- **Vision.** Usando este protocolo, se necesitará un UAV equipado con una cámara. El dron utilizará la información que recoja de la cámara para realizar un aterrizaje seguro [20].
- **Shakeup.** Este protocolo permite que un enjambre de drones se reconfigure para reducir las posibilidades de colisión [21].
- **CompareTakeOff.** Este protocolo está implementado para estudiar diferentes algoritmos de despegue de forma segura.

CAPÍTULO 4
Control de UAVs en función de la contaminación

Bajo la idea de monitorizar la contaminación en cualquier área, y la búsqueda de los puntos con máxima contaminación, surge el algoritmo *Pollution-driven UAV Control* (PdUC) [1]. Posteriormente, se aumentó la eficiencia del mismo en entornos reales al evitar perder tiempo tomando muestras en puntos muy cercanos. Esta mejora se consiguió al discretizar el anterior algoritmo, creando así una nueva versión: PdUC-D (PdUC-Discreto) [2].

4.1 Pollution-driven UAV Control (PdUC)

4.1.1. Base teórica

Este algoritmo se basa en la metaheurística quimiotaxis (*chemotaxis* en inglés) [22]. Esta metaheurística está basada en el movimiento de las bacterias, que se define por los estímulos químicos que reciben. Se sienten atraídas hacia zonas con mayor concentración de algunas sustancias, como por ejemplo la comida, mientras se sienten repelidas por otros químicos, como el veneno.

En cada paso del movimiento, la dirección de la siguiente posición viene definida por el incremento de concentración de un determinado componente químico entre la posición anterior y la actual. Este movimiento se puede generalizar en las siguientes ecuaciones:

$$\vec{P}_j^i = \begin{pmatrix} x_{j-1}^i \\ y_{j-1}^i \end{pmatrix} + \begin{pmatrix} d^i \times cos(\theta_j^i) \\ d^i \times sin(\theta_j^i) \end{pmatrix}, \tag{4.1}$$

$$\vec{\theta}_j^i = \begin{cases} \theta_{j-1}^i + \alpha_j^i, & p_j^i \geq p_{j-1}^i, \\ -\theta_{j-1}^i + \beta_j^i, & p_j^i < p_{j-1}^i. \end{cases} \tag{4.2}$$

En la ecuación 4.1 se define $\vec{P}_j^i$, que representa la nueva posición a la que se deben dirigir. Esta posición es calculada mediante la posición anterior, representada por x_{j-1}^i y y_{j-1}^i, sumando las distancia a que se va a mover, representado como d^i, y aplicando una dirección aleatoria θ_{j-1}^i. Esta dirección se define en la

ecuación 4.2, y es calculada en base al incremento de concentración $\triangle p^i$. De esta forma, se pueden definir dos tipos de movimiento: *Run* y *Tumble*:

- **Movimiento *Run*.** Ocurre cuando ha habido un incremento positivo de la sustancia buscada. El movimiento continúa en la misma dirección que anteriormente, sumándole un ángulo α. De esta forma se avanzará hacia otra posición en la que se supone que la concentración será todavía más alta. La aparición del ángulo α introduce variabilidad y aumenta el gradiente, facilitando la búsqueda de la zona de mayor concentración del químico.
- **Movimiento *Tumble*.** Al contrario que el anterior, ocurre cuando hay un incremento negativo de la sustancia buscada. En este caso, el movimiento cambia hacia la dirección opuesta, sumándole un ángulo β. De esta forma se evitará avanzar hacia puntos de menor concentración. La aparición del ángulo β tiene la misma funcionalidad que en el caso anterior. Gracias a este ángulo se evita volver exactamente al paso anterior y se aumenta el gradiente del movimiento. El tamaño de estos ángulos se puede ver gráficamente en la figura 4.1.

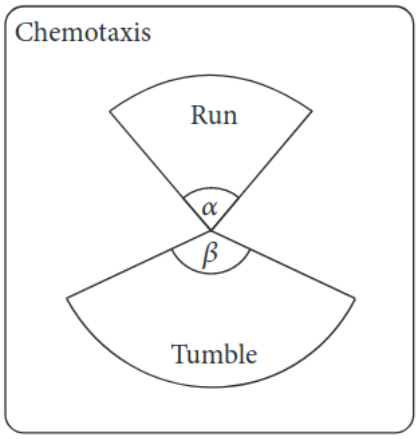


Figura 4.1: Ángulos límite para los los movimientos *Run* y *Tumble*.

4.1.2. Solución propuesta

En esta solución se busca que un dron equipado con un sensor de calidad del aire pueda imitar el movimiento de las bacterias de cara a determinar el punto de máxima contaminación, así como la distribución de ese contaminante en el área circundante. El algoritmo final propuesto se puede subdividir en dos fases:

- **Fase de búsqueda.** En esta fase el dron utiliza el concepto de quimiotaxis para buscar el punto con mayor concentración de contaminación. Si en la dirección en la que avanza aumentan los niveles de contaminación, continuará. Si disminuyen, cambiará la dirección. Para determinar cuándo se ha encontrado un máximo, se utiliza un contador TTL (*time-to-live*). Cada vez que se encuentra un valor más alto, el TTL se resetea, y mientras no se encuentre otro valor todavía mayor, se seguirá buscando hasta que el tiempo se termine. Cuando esto ocurre, se cambia a la fase de exploración ya que se considera que se ha encontrado el máximo.

- **Fase de exploración.** En esta fase se parte del punto de contaminación máxima encontrado en la fase anterior. Aquí, el UAV se mueve en forma de espiral para recorrer el área alrededor del punto máximo. Este movimiento continuará hasta haber cubierto el área entera o hasta encontrar otro punto de máxima contaminación, volviendo entonces a la fase de búsqueda. Mientras no se encuentre otro máximo, la distancia entre los puntos que recorre va aumentando hasta llegar a un valor de 3 veces la distancia recorrida inicialmente, por lo que cada vez la precisión de la toma de medidas disminuye. El movimiento en está fase está ilustrado en la figura 4.2.

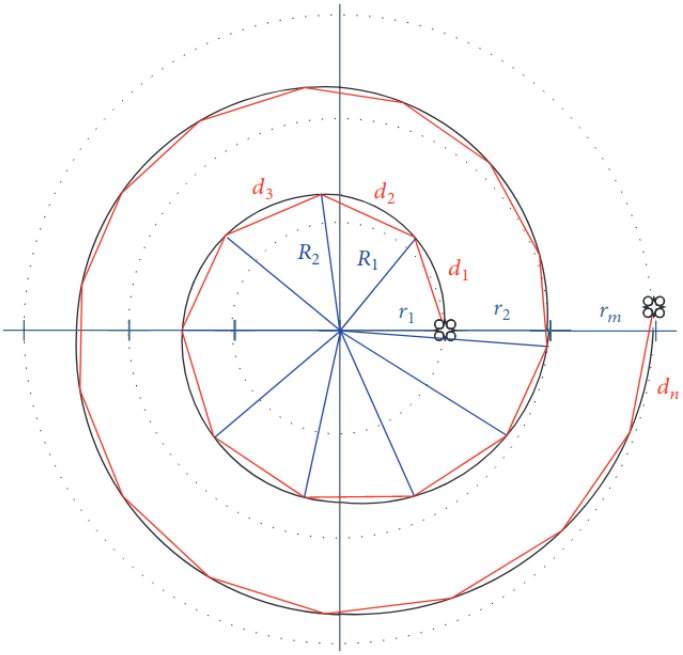


Figura 4.2: Fase de exploración de PdUC.

Cada vez que el dron toma una medida la almacena. También almacena los anteriores puntos máximos, y el radio de la espiral que ha llegado a recorrer. De esta forma se mantiene información de las áreas ya visitadas, y de los valores de cada punto medido, para maximizar la eficiencia.

Esta información se puede utilizar para optimizar la fase de exploración. El dron siempre evitará entrar en zonas que se consideren ya visitadas o ya medidas. Además, otra optimización utilizada es el cambio de sentido de la espiral, para evitar realizar desplazamientos muy largos entre dos puntos sucesivos. Esto ocurre cuando parte de la espiral que se está realizando en ese momento colisiona con los límites del área o con un área ya visitada. Estas casuísticas se pueden ver más claramente en la figura 4.3. En la parte de la izquierda de la figura se puede ver un ejemplo en el que el dron llega a los límites del área, por lo que, en lugar de avanzar a la parte superior del mismo y seguir la espiral en el mismo sentido, opta por girar y continuar la espiral al revés. En la parte derecha se observa otro ejemplo en el que al vehículo le correspondería entrar en un área ya visitada, pero en cambio la evita.

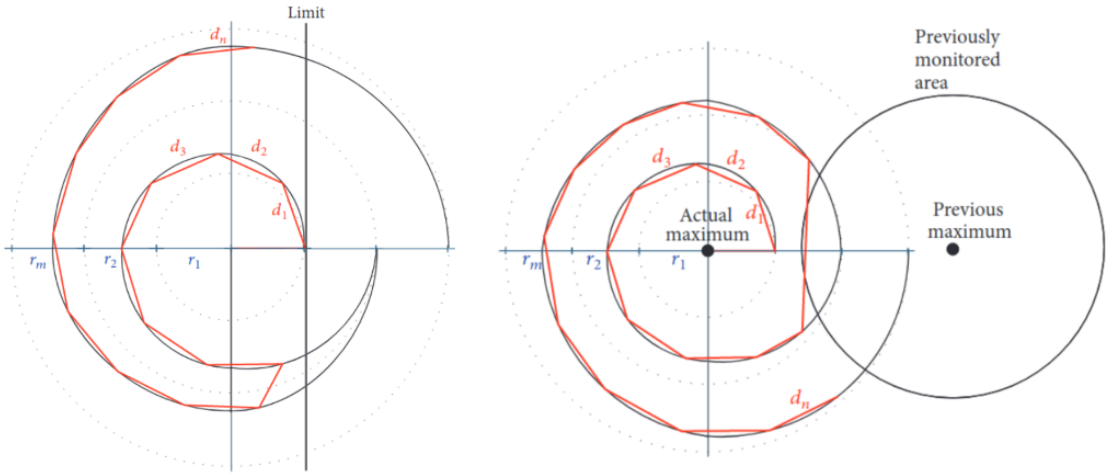


Figura 4.3: Fase de exploración cuando colisiona con los límites o con áreas ya visitadas.

Posteriormente se puede encontrar en los algoritmos 4.1 y 4.2 la formalización de este proceso, diferenciando entre cada una de las fases.

Algorithm 4.1 PdUC: Fase de búsqueda.

while $isSearching$ **do**
  $p_2^{pollution} \leftarrow CurrentPollution(p_2)$
  $\nabla poll \leftarrow p_2^{pollution} - p_1^{pollution}$
  $p_1 \leftarrow p_2$
  if $\nabla poll > 0$ **then**
    $ttl \leftarrow 0$
    $p_2 \leftarrow Run(p_1)$
    $p_{max} \leftarrow p_2$
  else
    $p_2 \leftarrow Tumble(p_1)$
    $ttl \leftarrow ttl + 1$
    $p_2 \leftarrow AdjustPSO(p_2, p_{max})$
  end if
  if $isInsideArea(p_2)$ **then**
    $MoveTo(p_2)$
  else
    $p_2 \leftarrow Tumble(p_1)$
  end if
  if $ttl > ttl_{max}$ **then**
    $isSearching \leftarrow false$
    $isExploring \leftarrow true$
  end if
end while

Algorithm 4.2 PdUC: Fase de exploración.

while isExploring **do**
  $round \leftarrow round + 1$
  $round_{size} \leftarrow 2\pi \cdot (round + round_{next})/2$
  $round_{direction} \leftarrow -previous_{direction}$
  $angle_{count} \leftarrow 2\pi/(round_{size}/d$
  $step \leftarrow 0$
  $angle \leftarrow 0$
  while $step < round_{size} and isExploring$ **do**
    if $isInsideArea(p_2) and isNotMonitored(p2)$ **then**
      $p_2^{pollution} \leftarrow CurrentPollution(p_2)$
      if $p_2 > p_max$ **then**
        $store(p_max)$
        $store(round)$
        $isExploring = false$
        $isSearching = true$
        $p_{max} = p_2$
      else
        $\nabla poll \leftarrow p_2^{pollution} - p_1^{pollution}$
        $MoveTo(p_2)$
      end if
    end if
    $step \leftarrow step + d$
    $angle \leftarrow anfle + angle_{count} \times round_{direction}$
    $p_1 \leftarrow p_2$
    $p_2 \leftarrow NextPoint(p_1, angle, step)$
    $previous_{direction} \leftarrow round_{direction}$
  end while
end while

4.1.3. Prueba del algoritmo

En la figura 4.4 se puede observar una traza de ejecución del algoritmo. Se puede ver coloreado en el fondo los niveles de contaminación, sobre los cuales se ha movido el dron. Se puede observar los cambios de fase del algoritmo cada vez que detecta un nuevo máximo, así como los cambios de sentido del movimiento en espiral.

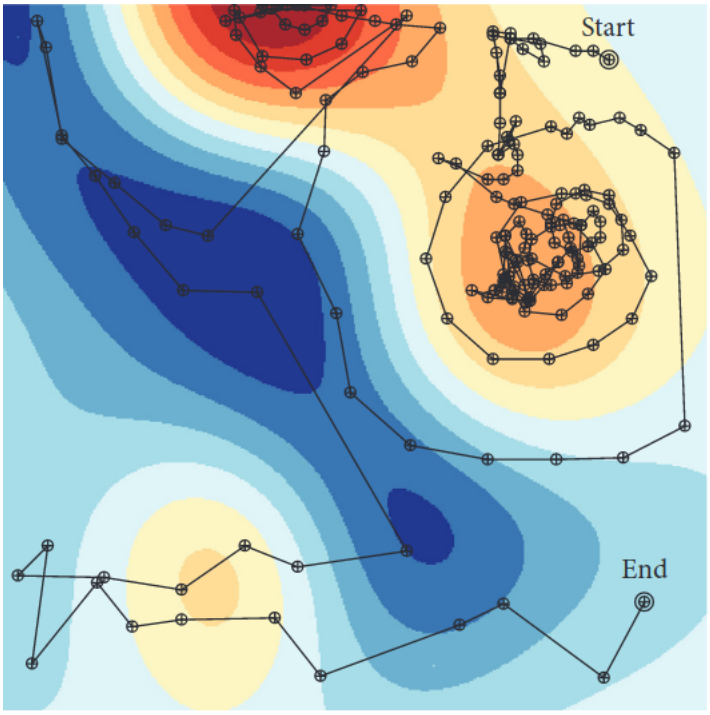


Figura 4.4: Simulación del algoritmo PdUC.

4.2 PdUC-Discretized (PdUC-D)

Tras la puesta en marcha y la prueba en profundidad del algoritmo PdUC, se observó que, pese a que funcionaba correctamente y ofrecía unos resultados adecuados, requería demasiado tiempo. Esto resulta un problema crítico debido a la limitada batería que equipan los multicópteros, lo cual significa que no siempre es posible realizar una ejecución completa del algoritmo. Es por esto que surge la necesidad de desarrollar un protocolo más eficiente.

4.2.1. Solución propuesta

Como solución a la problemática del algoritmo PdUC, se optó por discretizar el área a medir. La discretización es uno de los enfoques matemáticos más eficientes en cuanto a la optimización de un sistema. De esta forma se transforma un sistema contínuo en uno discreto, realizando una división de sus componentes.

Para este proceso, se ha asumido como el estándar para este protocolo un área de 4x4 Km de tamaño. Esta superficie se divide en una cuadrícula en la que cada celda tiene un tamaño de 100m de largo, ya que se supone que en una subárea de ese tamaño, no hay una gran variación en cuanto a la contaminación ambiental dentro de la misma. De esta forma, el dron únicamente tiene que desplazarse entre los centros de las celdas, y tomar una única medida por celda. Esto también ayuda a evitar tomar medidas de puntos que ya se han visitado, al especificar qué se considera como visitado.

El nuevo algoritmo funcionaría de forma análoga a PdUC, únicamente siendo necesario adaptarlo a la nueva forma de movimiento dentro del área. Los principales cambios son estos:

- **Cambios en la fase de búsqueda:**
  De nuevo, en esta fase, si se detecta un incremento en la contaminación, se avanzará a la siguiente casilla en la misma dirección (Movimiento *Run*). Por lo contrario, si la contaminación no crece o disminuye, el multicóptero se moverá al rededor de la celda anteriormente detectada como un máximo (Movimiento *Tumble*), priorizando la celda más cercana. Cuando todas las celdas alrededor de la celda máxima han sido ya visitadas es cuando se avanza a la fase de exploración. En la figura 4.5 se puede observar la adaptación de la metaheurística *chemotaxis* de esta fase. Con este cambio, es mucho más fácil determinar los movimientos del UAV.

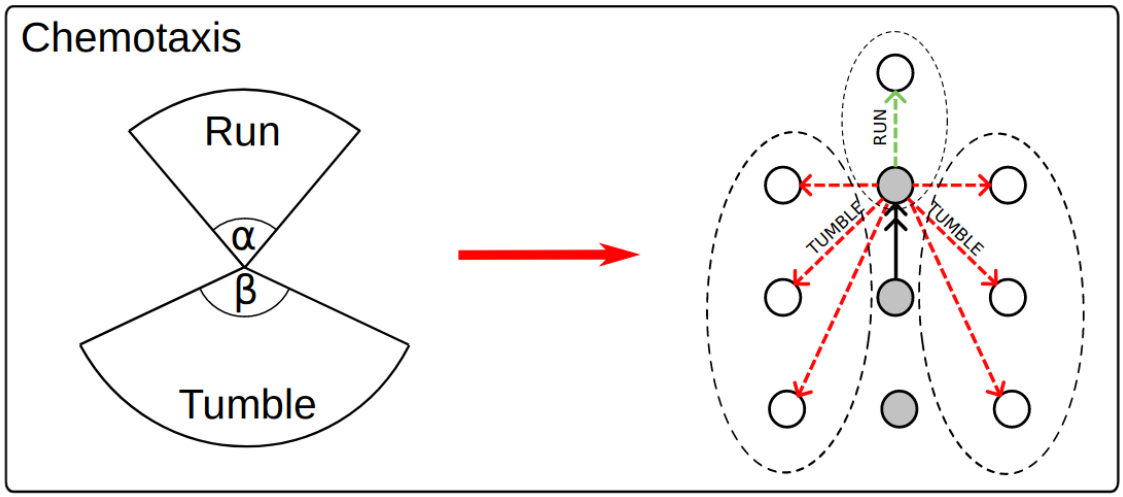


Figura 4.5: Diferencias en la fase de búsqueda entre PdUC y PdUC-D.

- **Cambios en la fase de exploración:**
  En esta fase los cambios son menores. Únicamente es necesario adaptar el desplazamiento del dron, pero manteniendo el funcionamiento. Al moverse por celdas cuadradas, la espiral pasa a tener forma cuadrada también. En la figura 4.6 se puede observar esta adaptación de forma gráfica. Para este movimiento se han de tener las siguientes consideraciones:

  1. En cada ronda de la espiral, se salta un número creciente de celdas sin medir. En la primera ronda la espiral tiene un radio de 3 celdas, y se

salta 1 celda por movimiento. En la segunda ronda, tiene un radio de 5 celdas y se salta 2 celdas por movimiento, y así sucesivamente.

2. Se considerará como visitado todo el interior del cuadrado creado al terminar cada ronda de la espiral.

3. Se mantiene las consideraciones del algoritmo PdUC en cuanto al cambio de sentido de la espiral en el caso de llegar a un borde o a un área ya visitada.

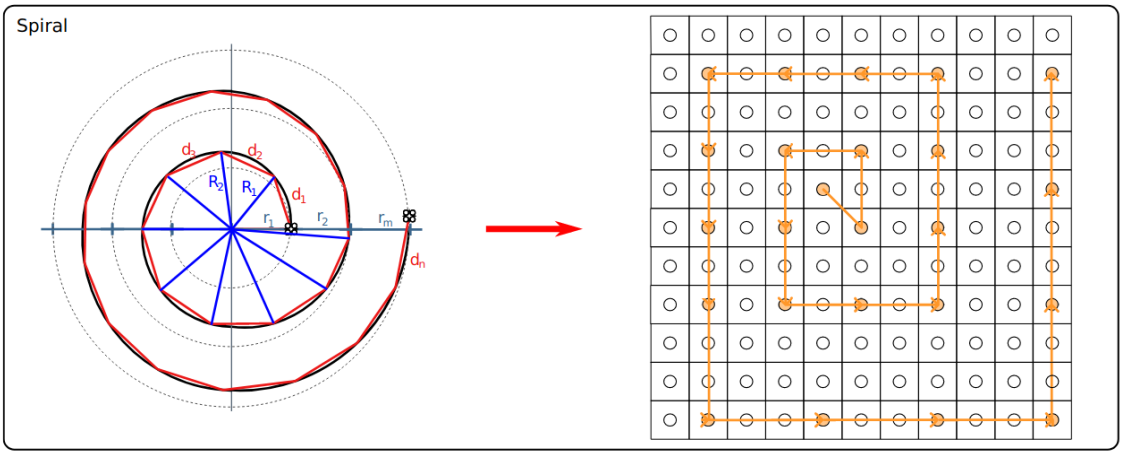


Figura 4.6: Diferencias en la fase de exploración entre PdUC y PdUC-D.

Durante la ejecución de este algoritmo, se mantendrá un control de los datos y de las áreas visitadas mediante el uso de dos matrices $P_{m,n}$ y $B_{m,n}$ en las que se almacenarán los datos medidos y las celdas monitorizadas, respectivamente. Para esto se asumirá una cuadrícula de n x m celdas.

$$P_{m,n} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{pmatrix} \tag{4.3}$$

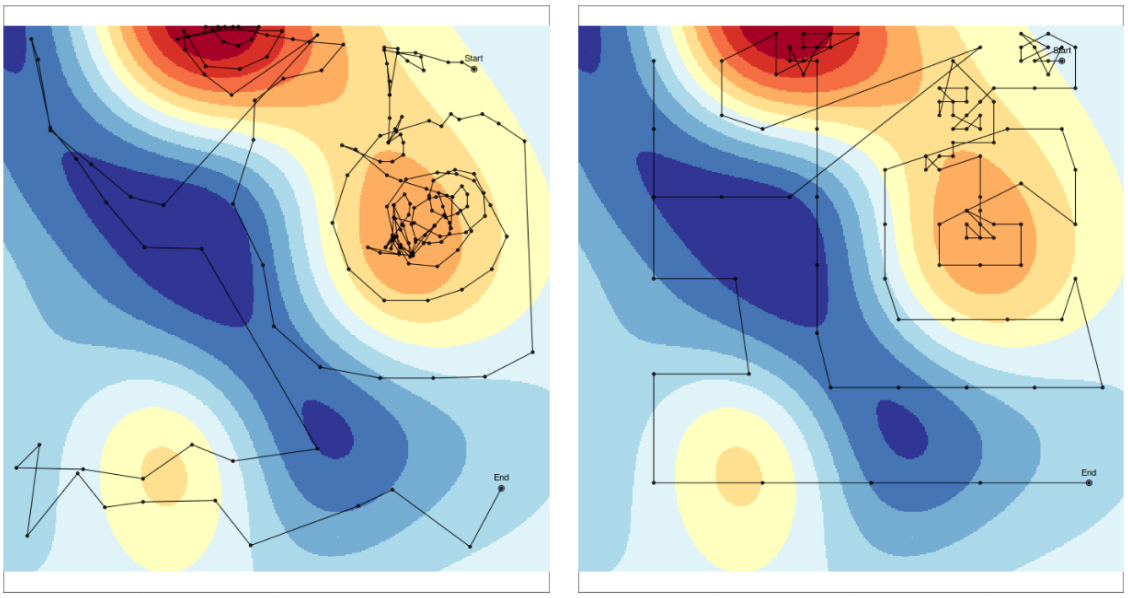
$$B_{m,n} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,n} \end{pmatrix} \tag{4.4}$$

La matriz P se inicializará con valores *null*, mientras que la matriz B se inicializará con ceros. Cada vez que se tome una medida, en un punto $t_{i,j}$, se almacenará el valor de esa medida en $P_{i,j}$, mientras que en $B_{i,j}$ se almacenará un 1. Adicionalmente, al terminar una ronda de la espiral, todas las celdas interiores de la misma se marcan como visitadas en B.

4.2.2. Prueba del algoritmo

En la figura 4.7 se observa la traza de ejecución de ambos algoritmos. Igual que en caso anterior, se puede ver coloreado los niveles de contaminación y el movimiento realizado por el dron.

A simple vista se puede diferenciar que el recorrido realizado por el multicóptero es menor en el caso del segundo algoritmo, disminuyendo también notablemente la cantidad de mediciones hechas por el UAV. Esto es lo que al final se traduce en un menor tiempo de vuelo, pero sin renunciar a la fiabilidad de los datos ya que, como se puede observar, el dron sigue haciendo hincapié sobre las zonas de mayor concentración de contaminantes.



(a) PdUC (b) PdUC-D

Figura 4.7: Comparativa en una traza de ejecución de los algoritmos PdUC y PdUC-D.

CAPÍTULO 5
Análisis del problema

En este capítulo se estudiará el problema con vistas a facilitar su posterior desarrollo. Será en este punto donde se establecerá de forma teórica los requisitos que el proyecto ha de cumplir. Además, se estudiará la forma más eficiente de desarrollarlo, creando una imagen mental previa a la programación.

5.1 Actores

Se definen como actores en un sistema cualquier parte que interactúa con el mismo. Puede tratarse de una persona, un dispositivo involucrado o una actividad que desempeña un rol. En este desarrollo, se definen los siguientes actores:

- **Dron**. Se trata del multicóptero encargado de realizar el vuelo y tomar las medidas de contaminación, para posteriormente enviar los datos al sistema para que puedan ser interpretados por el usuario.

- **Usuario**. Se trata de la persona encargada de definir las variables de la ejecución. Por ejemplo, define el lugar y el tamaño de la superficie a medir, para posteriormente supervisar que el vuelo se realiza correctamente e interpretar los datos finales.

- **ArduSim**. Se trata del software encargado de darle las órdenes al dron, que definirán su movimiento. Hará de intermediario entre el protocolo implementado y el dron. Posteriormente se encargará de tratar los datos recibidos para ofrecérselos al usuario de una forma legible, para que este los pueda interpretar.

5.2 Especificación de requisitos

En este apartado estableceremos los requisitos que necesitará nuestra aplicación. Diferenciaremos entre requisitos funcionales y no funcionales.

5.2.1. Requisitos funcionales

Estos requisitos son los que están relacionados con la funcionalidad del proyecto. Aquí es donde se define qué acciones debe ser capaz de llevar a cabo cada actor.

- **Dron**

  1. **Tomar datos de contaminación (D-01).**
     El dron utiliza su sensor para medir la contaminación en un punto y mandar los datos al protocolo de ArduSim.
  2. **Recibir nueva órden (D-02).**
     El dron recibe una señal con su nuevo destino.
  3. **Desplazarse al lugar indicado (D-03).**
     El dron se desplaza hacia su nuevo destino, volviendo a realizar la medida.

- **Usuario**

  1. **Introducir los parámetros de la simulación (U-01).**
     El usuario introduce parámetros como la localización y el área a medir y da comienzo al vuelo.
  2. **Comenzar la simulación (U-02)**
     Tras introducir los parámetros, el usuario decide comenzar la simulación.
  3. **Recibir los datos (U-03).**
     El usuario obtiene los datos una vez la simulación ha terminado y los interpreta.

- **ArduSim**

  1. **Calcular nuevo destino del dron (A-01).**
     ArduSim utiliza los datos recibidos para calcular la próxima posición a la que debe desplazarse el dron.
  2. **Guardar los resultados de la simulación (A-02).**
     ArduSim guarda los resultados de la simulación de una forma legible para el usuario.

5.2.2. Requisitos no funcionales

Estos son los requisitos que debería cumplir el proyecto, pero no tienen relación con la funcionalidad del mismo. Se trata de requisitos más generales de la aplicación en su conjunto.

1. **Portabilidad.**
   Este protocolo, al igual que los otros que componen ArduSim, ha de ser ejecutable en cualquier plataforma. Además, debe ser capaz de usarse indistintamente de si se está probando en un dron real o en un dron simulado.

2. **Mantenibilidad.**
   El protocolo debe ser implementado usando un mecanismo de control de versiones, facilitando su mantenibilidad y su recuperación en caso de pérdida.

3. **Eficiencia.**
   El protocolo debe obtener un buen nivel de detalle en los datos, sin necesidad de utilizar tiempos de ejecución demasiado largos.

4. **Usabilidad.**
   La aplicación debe ser sencilla para el usuario. Será el sistema el que se encargue de todas las partes complejas, necesitando poca actuación del usuario.

5. **Cohesión.**
   El protocolo, al ser una parte añadida al simulador ArduSim, debe estar completamente cohesionado con el mismo. No debe parecer un desarrollo externo al simulador, sino que se debe sentir como una parte más.

6. **Escalabilidad.**
   El protocolo debe ser capaz de medir y de tomar datos de áreas de todos los tamaños. Se debe comportar de igual forma en un área pequeña y en un área grande, sin afectar a la precisión de los datos.

5.3 Diagrama de clases

En este diagrama se expondrán las diferentes clases que conpondrán el sistema, junto con sus atributos y las relaciones entre ellas. Este diagrama está compuesto por las siguientes clases:

1. **PollutionProtocol.** Esta clase representa el protocolo implementado. Será la encargada de almacenar el tamaño de la cuadrícula e información sobre los puntos más relevantes de la misma. También será la encargada de controlar los datos sobre los puntos ya visitados.

2. **UAV.** Esta clase hace referencia al multicóptero encargado de tomar las medidas.

3. **DataPoint.** Esta es la clase encargada de definir las coordenadas de cada punto en la cuadrícula, así como el valor de contaminación medido si es el caso.

4. **ValueSet.** Esta clase está destinada a agrupar los puntos ya visitados, así como el valor máximo encontrado o el valor mínimo.

Este diagrama se encuentra en la figura 5.1, donde las relaciones son las siguientes:

- Un UAV **se ubica en** un DataPoint.

- El PollutionProtocol **simula** un UAV. También **almacena puntos visitados** en ValueSets.
- Un ValueSet **está compuesto de** DataPoints.

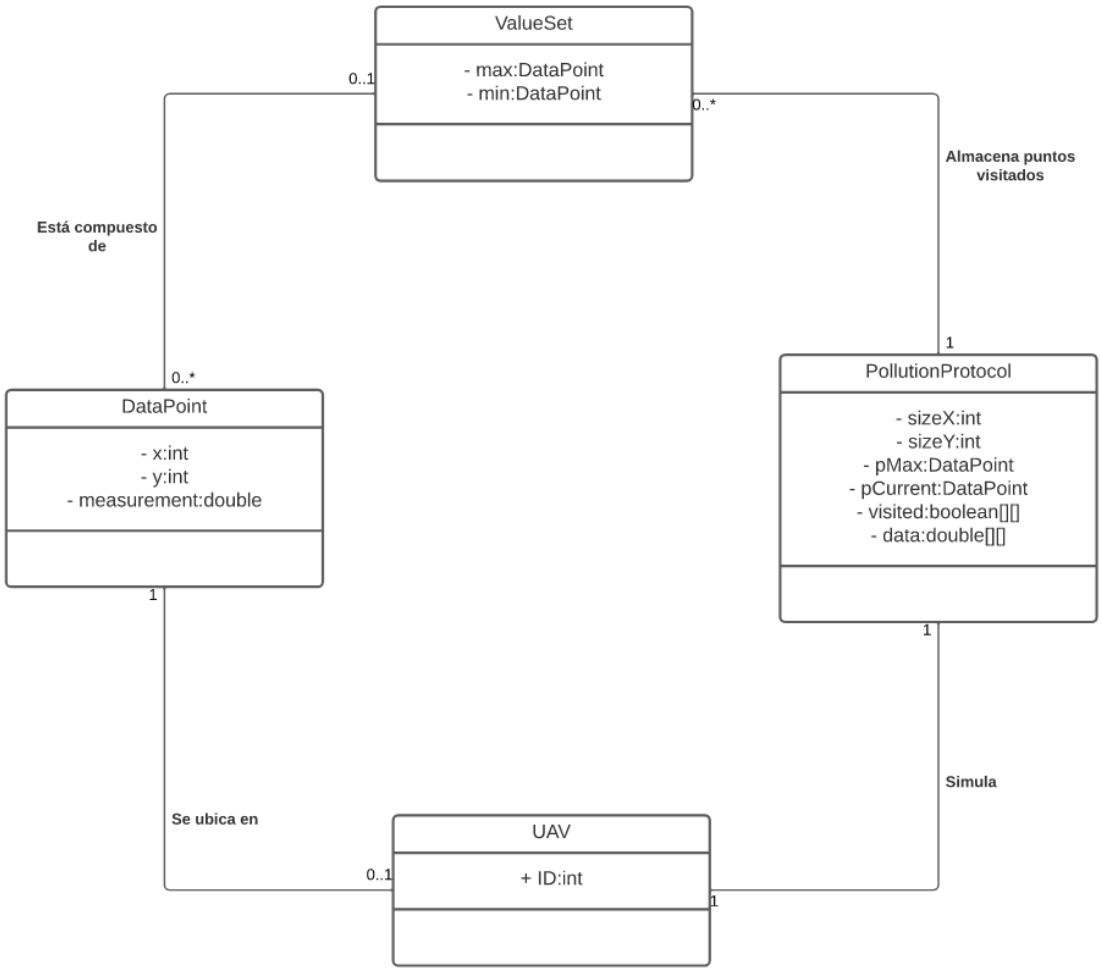


Figura 5.1: Diagrama de clases del protocolo *pollution*.

5.4 Diagrama de casos de uso

En este diagrama se muestra de forma gráfica las diferentes acciones que pueden llevar a cabo los actores. Cabe destacar las siguientes relaciones entre las acciones:

- Cuando una acción **incluye** a una segunda, significa que la segunda no se puede realizar sin haberse realizado la primera antes.
- Cuando una acción tiene una segunda que la **extiende**, significa que aporta algo a la primera, pero no siempre que se realice la primera es necesario que se realice la segunda.

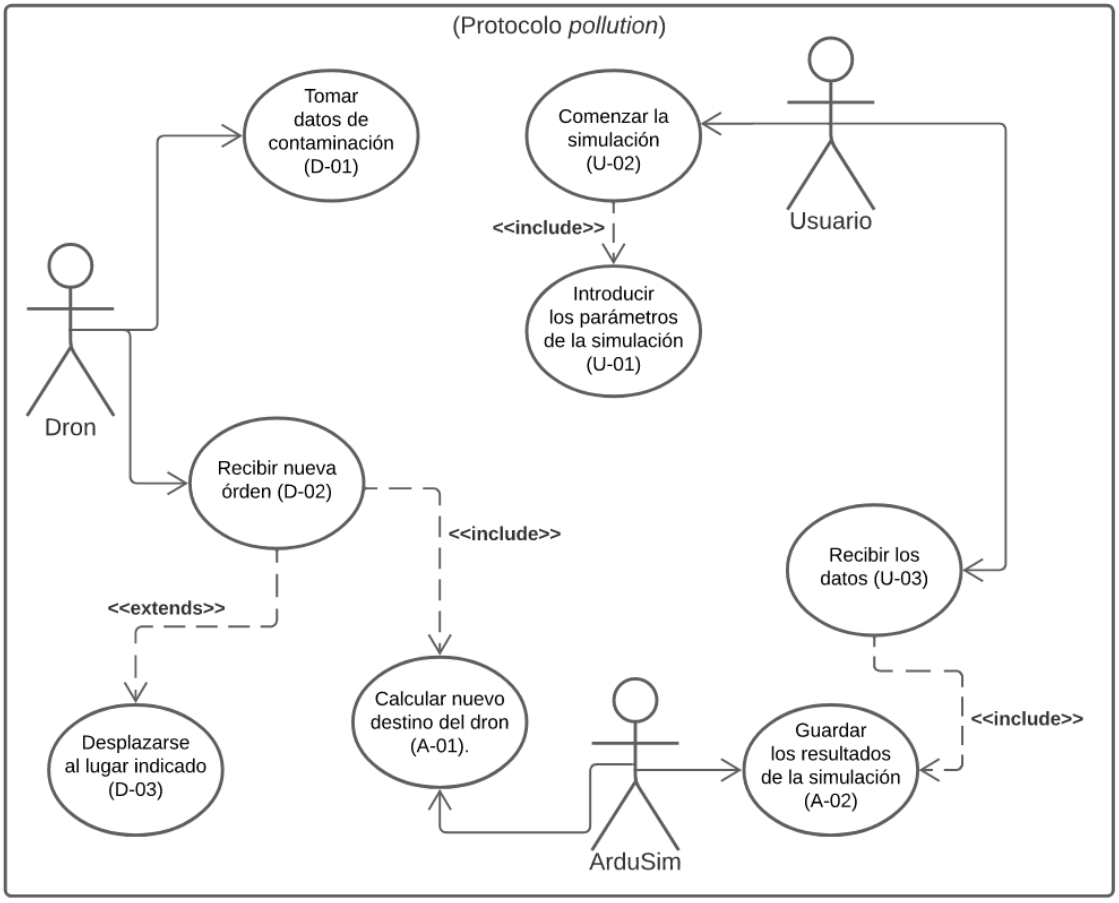


Figura 5.2: Diagrama de casos de uso del protocolo *pollution*.

En la figura 5.2 se puede encontrar el diagrama de casos de uso. A continuación se detallan las diferentes acciones en el sistema.

Tabla 5.1: Caso de uso D-01.

Caso de uso	Tomar datos de contaminación (D-01).
Actores	Dron.
Resumen	El dron se detiene el tiempo necesario para que su sensor incorporado tome una medida de contaminación en su localización actual.
Precondiciones	El dron debe situarse en un punto que no haya medido anteriormente, dentro de los límites del área.
Postcondiciones	La medida se debe almacenar en el sistema, así como información del lugar donde se ha hecho el muestreo.
Incluye	-
Extiende	-

Tabla 5.2: Caso de uso D-02.

Caso de uso	Recibir nueva órden (D-02).
Actores	Dron.
Resumen	El dron recibe una nueva órden de ArduSim.
Precondiciones	ArduSim debe haber calculado la posición a la que se debe mover y habérsela enviado.
Postcondiciones	El dron dispone del que debería ser su siguiente punto.
Incluye	Calcular nuevo destino del dron (A-01).
Extiende	Desplazarse al lugar indicado (D-03).

Tabla 5.3: Caso de uso D-03.

Caso de uso	Desplazarse al lugar indicado (D-03).
Actores	Dron.
Resumen	El dron se mueve hacia la ubicación que ha recibido anteriormente.
Precondiciones	La nueva posición debe ser accesible para el dron y sin obstáculos.
Postcondiciones	El dron debe ubicarse en la posición destino.
Incluye	-
Extiende	-

Tabla 5.4: Caso de uso U-01.

Caso de uso	Introducir los parámetros de la simulación (U-01).
Actores	Usuario.
Resumen	El usuario decide qué parámetros debe tener la simulación y las introduce al sistema.
Precondiciones	El usuario debe tener claro qué quiere simular.
Postcondiciones	El sistema debe mostrar la interfaz principal del simulador, permitiéndole iniciar la simulación.
Incluye	-
Extiende	-

Tabla 5.5: Caso de uso U-02.

Caso de uso	Comenzar la simulación (U-02).
Actores	Usuario.
Resumen	El usuario decide comenzar la simulación.
Precondiciones	Los parámetros introducidos anteriormente han de ser correctos.
Postcondiciones	La simulación comienza.
Incluye	-
Extiende	-

Tabla 5.6: Caso de uso U-03.

Caso de uso	Recibir los datos (U-03).
Actores	Usuario.
Resumen	El usuario recibe los datos tratados por ArduSim.
Precondiciones	ArduSim debe haber terminado correctamente la simulación y haber preparado los datos.
Postcondiciones	El usuario trata los datos.
Incluye	Guardar los resultados de la simulación (A-02).
Extiende	-

Tabla 5.7: Caso de uso A-01.

Caso de uso	Calcular nuevo destino del dron (A-01).
Actores	ArduSim.
Resumen	ArduSim calcula la nueva posición a la que el dron debe dirigirse.
Precondiciones	El dron debe haber tomado correctamente la medida del punto en el que se encuentra actualmente.
Postcondiciones	El dron recibe su nueva orden.
Incluye	-
Extiende	-

Tabla 5.8: Caso de uso A-02.

Caso de uso	Guardar los resultados de la simulación (A-02).
Actores	ArduSim.
Resumen	ArduSim trata los datos tomados por el dron y los guarda de forma comprensible para el usuario.
Precondiciones	La simulación ha terminado correctamente y ArduSim ha recibido todos los datos necesarios.
Postcondiciones	El usuario obtiene los datos finales.
Incluye	-
Extiende	-

5.5 Diagramas de actividad

En los diagramas de actividad es dónde veremos el flujo de ejecución de las dos fases del algoritmo. En la figura 5.3 se encuentra el diagrama de flujo de la fase de búsqueda.

Por su parte, en la figura 5.4 se encuentra el diagrama de la fase de exploración.

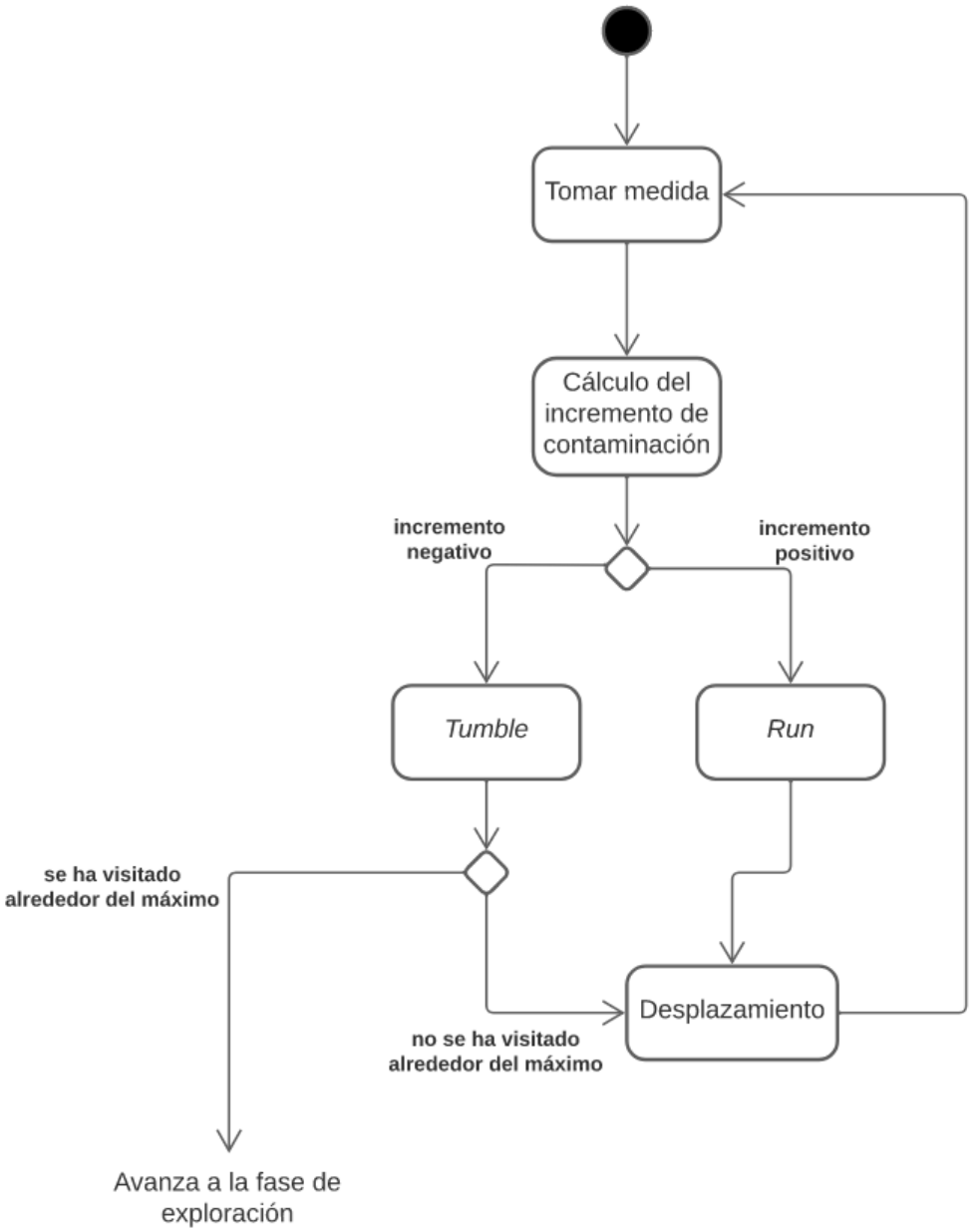


Figura 5.3: Diagrama de actividad de la fase de búsqueda.

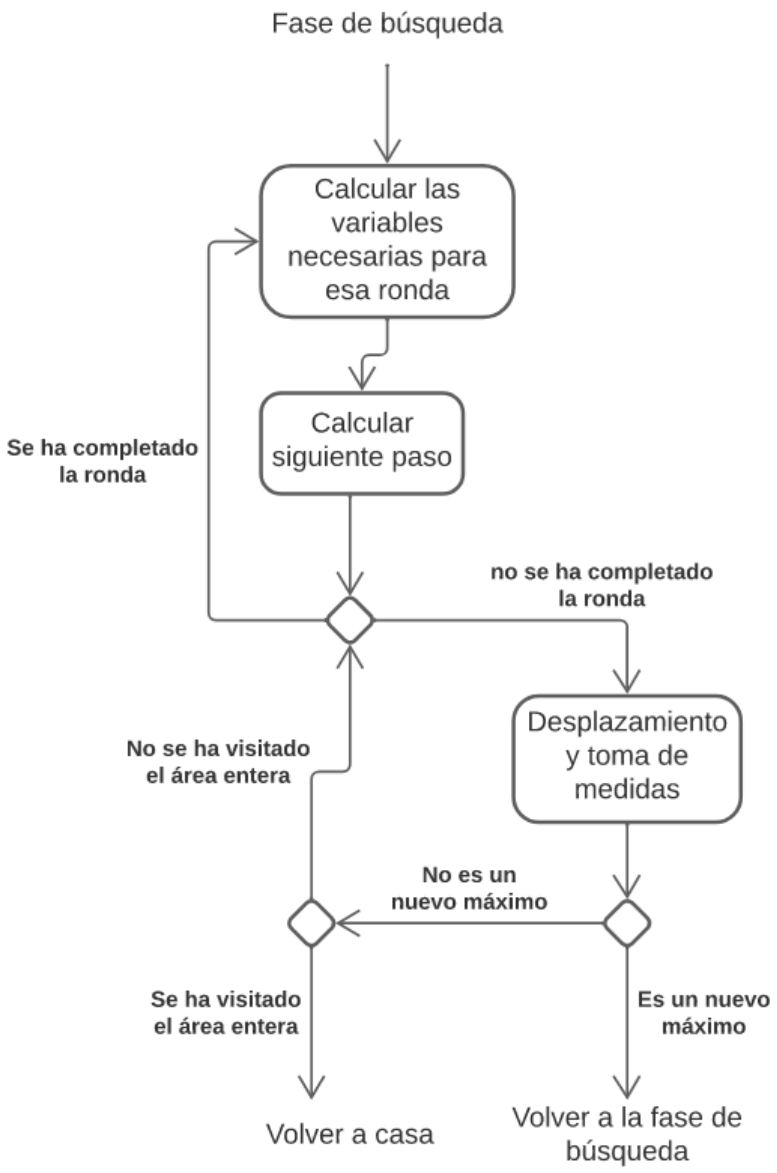


Figura 5.4: Diagrama de actividad de la fase de exploración.

CAPÍTULO 6
Desarrollo

6.1 Tecnologías usadas, mantenimiento y control de versiones

El desarrollo de este proyecto se ha realizado íntegramente en Linux, bajo la distribución Ubuntu [23]. Se ha utilizado la metodología tradicional incremental. Al no tratarse de un desarrollo excesivamente extenso, y al estar formado por una única persona, no se ha optado por las metodologías ágiles. Se ha ido construyendo el proyecto de manera progresiva, añadiendo funcionalidades e intentando pulirlas lo máximo posible antes de comenzar con la siguiente. De esta forma, ha sido posible hacer pruebas en cada hito del desarrollo, para comprobar si el movimiento de los drones estaba siendo el esperado. Los cambios y mejoras se han añadido y depurado usando un controlador de versiones, en este caso GitHub.

Las tecnologías utilizadas han sido las siguientes:

- Lenguaje de programación: Java
- Simulador de vuelo: ArduSim
- IDE: Eclipse [24]
- Diseño de interfaces: Scene Builder [25]

6.2 Movimiento del dron

Para realizar el movimiento del dron, se han desarrollado los siguientes métodos:

1. El método *move(DataPoint p)*, que simplemente realiza una llamada al método siguiente, cogiendo los valores del DataPoint que recibe como argumento.
2. El método *move(int x, int y)*, que se encarga de calcular la siguiente posición del dron y enviarle la orden para que se mueva.

3. El método *moveAndRead(DataPoint p)*, que realiza una llamada al primer método, haciendo que el dron se mueva, mientras que ejecuta una lectura del sensor, almacenando el dato leído.

Este es el código de los anteriores métodos:

```
//PollutionThread.java
private void move(DataPoint p) { //(1)
  move(p.getX(), p.getY());
}

private void move(int x, int y) { //(2)
  Double xTarget = PollutionParam.origin.x + (x *
      PollutionParam.density);
  Double yTarget = PollutionParam.origin.y + (y *
      PollutionParam.density);
  MoveTo moveTo = copter.moveTo(new Location3D(xTarget, yTarget,
      PollutionParam.altitude), new MoveToListener() {
      (...)
  });
  moveTo.start();
  try {
     moveTo.join();
  } catch (InterruptedException ignored) {}
}

private void moveAndRead(DataPoint p) { //(3)
  double m;
  move(p);
  m = PollutionParam.sensor.read();
   (...)
  visited[p.getX()][p.getY()] = true;
  p.setMeasurement(m);
}
```

Se han omitido diversas partes del código al resultar de poco interés. En cambio, sí que resulta más importante destacar la forma en la que el método *move(int x, int y)* calcula el destino:

1. Obtiene las coordenadas base del área, que se sitúan en la esquina suroeste de la misma.

2. Calcula el incremento relativo de las coordenadas respecto al origen. Esto se calcula mediante la posición en la cuadrícula del siguiente punto multiplicado por el tamaño de cada celda.

3. Para terminar, suma ese incremento a las coordenadas origen, y le da al dron la orden de dirigirse exactamente a esas nuevas coordenadas.

6.3 Movimiento en espiral (método explore)

Este método es el encargado del movimiento durante la fase de exploración del algoritmo. Pese a ser la segunda fase, se ha decidido comenzar por esta, al resultar más sencilla. De esta forma también sirve de ayuda a la comprensión del funcionamiento de ArduSim, resultando más fácil la depuración. Aquí se pueden encontrar las partes más importantes del método:

```
//PollutionThread.java
private boolean explore() {
  // Initial round. Initial radius = 3 to take into account Tumble
  int skip = 2;
  int radius = 3;
  ValueSet points;
  (...)
  boolean newMax = false;
    // Measure until radius covers all the grid or a new max is found
  while(!newMax && (radius < sizeX || radius < sizeY)) {
     (...)
     points = generatePoints(radius, skip); //(2)
     // Iterate until all points have been visited or a new maximum is
         found
     while(!points.isEmpty() && !newMax) {
        // ---- Read closest point
        pCurrent = findClosestPoint(points);
        moveAndRead(pCurrent);
        points.remove(pCurrent);
        // ---- If the point is a new max, exit explore and return to
            run & tumble
        if(pCurrent.getMeasurement() - pMax.getMeasurement() >
            PollutionParam.pThreshold) {
           newMax = true; //(3)
        }
     }
     if(points.isEmpty()) {
        markExploredArea(radius, skip); //(4)
     }
     if(newMax) {
        // Set pMax to pCurrent, keep both the same so algorithm goes
            to tumble on next step
        pMax = new DataPoint(pCurrent);
     }
     radius += skip; //(1)
     skip++;
  }
  return newMax;
}
```

Del funcionamiento de este método cabe destacar las siguientes partes:

1. Tal y como se mencionó en el apartado 4.2.1, el radio de la espiral se incrementa en cada ronda, igual que lo hace el número de casillas por las que se pasa sin parar a tomar medida.

2. Para el cálculo del siguiente movimiento, primero se obtiene un *HashMap* con todos los puntos que componen la siguiente iteración de la espiral, descartando los puntos ya visitados, y los puntos fuera del área a medir. De esto se encarga el método *generatePoints(int radius,int skip)*. Mientras todavía queden puntos por visitar de esa ronda, se elegirá siempre el punto más cercano al dron.

3. Si ha habido un incremento positivo en la contaminación que supera el umbral, esta fase termina y se pasa a la fase de búsqueda.

4. Al terminar una ronda de la ejecución, se marca toda la zona interna como visitada y se pasa a la siguiente ronda. Se marcan en el método *markExploredArea(int radius, int skip)*.

5. En el caso de que este método termine sin haber encontrado un nuevo máximo, el algoritmo termina.

6.4 Búsqueda del punto con contaminación máxima (método runAndTumble)

Este es el método encargado de la primera fase del algoritmo, la fase de búsqueda. Aquí se pueden encontrar las partes más importantes del método:

```
 //PollutionThread.java
private void runAndTumble() {
   DataPoint pTemp;
   ValueSet points = new ValueSet();
   boolean finished = false;
   while(!finished) {
   if(pCurrent.getMeasurement() - pMax.getMeasurement() >
       PollutionParam.pThreshold) {//(1) // Run (2)
         //We move in the same direction as the pollution has increased
         pTemp = new DataPoint(pMax);
         pMax = new DataPoint(pCurrent);
         pCurrent.add(pCurrent.distVector(pTemp));
         points = new ValueSet();
         if(pCurrent.isInside(sizeX, sizeY) &&
             !(visited[pCurrent.getX()][pCurrent.getY()])) {
            moveAndRead(pCurrent);
         } else {
            pCurrent = new DataPoint(pMax);
         }
      } else { // Tumble (3)
         //We create the points that we need to visit in tumble phase
         points = generatePoints(1, 1);
         if(points.isEmpty()) { //(4)
```

```
            finished = true;
         } else {
            // ---- Read closest point
            pCurrent = findClosestPoint(points);
            moveAndRead(pCurrent);
            points.remove(pCurrent);
         }
      }
   }
}
```

Se pueden destacar los siguientes puntos:

1. Tal y como se explicó en el capítulo 4.2.1, primero se comprueba la diferencia entre la contaminación del punto actual y el punto máximo.

2. En el caso de que esta diferencia sea positiva y supere cierto umbral definido por el usuario, se realiza un movimiento *Run*. Este movimiento implica moverse en la misma dirección en la que ha habido el incremento, siempre y cuando esa nueva dirección no esté fuera del área o ya esté visitada. Esta dirección se calcula a partir del vector diferencia entre el punto máximo anterior y el punto actual, que pasa a ser el nuevo máximo.

3. En el caso de que esta diferencia sea negativa o no llegue al umbral, se realiza un movimiento *Tumble*. Este movimiento implica el recorrido de todas las celdas alrededor del punto de contaminación máxima, en busca de algún incremento. Igual que en el método anterior, se crea un *HashMap* con todos los puntos que hay que visitar, el cual se genera mediante el método *generatePoints(int round, int skip)* y se avanza directamente al más cercano.

4. Este método se ejecuta indefinidamente mientras siga encontrando puntos máximos. En el momento en el que visita todos los puntos alrededor del máximo sin encontrar incremento positivo, pasa a la siguiente fase.

6.5 Bucle principal del algoritmo

Las dos fases del algoritmo son lanzadas desde un bucle dentro de el método principal encargado de crear el hilo de ejecución. Aquí se pueden observar las partes más importantes:

```
 //PollutionThread.java
public void run() {
   sizeX = PollutionParam.width / PollutionParam.density;
   sizeY = PollutionParam.length / PollutionParam.density;
   visited = new boolean[sizeX][sizeY];
   PollutionParam.data = new double[sizeX][sizeY]; //(1)
   (...)
   /* Start Algorithm */
   pMax = new DataPoint(sizeX / 2, sizeY / 2); //(2)
   //Make the first move
```

```
    pCurrent = new DataPoint(pMax);
    pCurrent.addY(1);
    boolean newMax = true;
    try {
       moveAndRead(pMax);
       moveAndRead(pCurrent);
       //Main loop.
       while(newMax) { //(3)
          /* Phase 1: Looking for the point with max pollution */
          runAndTumble();
          /* Phase 2: Exploring the area surrounding the point with max
              pollution */
          newMax = explore();
       }
    } catch (LocationNotReadyException e) {
       e.printStackTrace();
       endExperiment("Unable to calculate the target coordinates.");
    }
    endExperiment("Experiment ended successfully."); //(4)
}
```

Donde destacan los siguientes puntos:

1. Primero se inicializan las matrices y se obtiene el tamaño en celdas del área.
2. Se coloca el dron en el centro para comenzar la simulación, y se hace el primer movimiento desde fuera del bucle principal.
3. Al entrar al bucle principal, es el método *runAndTumble()* el que se encargará de determinar si ha habido incremento positivo o negativo, y comenzar la ejecución.
4. Una vez el método *explore()* termina correctamente, termina el bucle principal y el protocolo llega a su final con el método *endExperiment(String msg)*. Este método es el encargado de hacer que el dron vuelva a la posición inicial y aterrice de forma segura, terminando así con el algoritmo y con la ejecución.

CAPÍTULO 7
Pruebas y análisis de los resultados

En este capítulo será donde realmente se ponga en práctica el desarrollo realizado, pudiendo comprobarse la eficiencia y la calidad del algoritmo implementado. Se harán comparaciones con fases previas del desarrollo para verificar la evolución, así como comparativas con otras ejecuciones en las que no se tenga en cuenta la calidad del aire.

En todas las ejecuciones se comenzará por el punto central del área y se utilizarán los siguientes parámetros de simulación:

Tabla 7.1: Parámetros de las simulaciones.

Simulación 1 (Realista)		**Simulación 2** (Reducida)	
Parámetro	Valor	Parámetro	Valor
Área	4x4 Km	Área	500x500 m
Tamaño de celda	100 m	Tamaño de celda	50 m
Número de celdas	40x40 = 1600	Número de celdas	10x10 = 100
Umbral del error	5	Umbral del error	5
Tiempo de muestreo	5 s	Tiempo de muestreo	4 s
Velocidad máxima	15 m/s	Velocidad máxima	15 m/s
Altitud	15 m	Altitud	10 m

En la primera simulación se utilizarán parámetros realistas, muy cercanos a una ejecución fuera del simulador. Se tomará medidas en un área de 4x4 Km de tamaño, dividida en una cuadrícula de 40x40 celdas. En la segunda, se utilizarán parámetros más reducidos, donde la superficie pasa a ser de 500x500 m, y se subdivide en 10x10 celdas. En ambas simulaciones se utilizarán mapas de valores diferentes, para enriquecer el resultado de estas pruebas.

Posteriormente, se obtendrá un mapa de calor de los datos medidos. En estos gráficos, los puntos con más contaminación se representan con colores más cercanos al amarillo, mientras los puntos con menor contaminación, tienden al morado.

7.1 2D *kriging*

Dado que en la mayoría de ejecuciones no se obtendrán los datos de todos los puntos, es necesaria una técnica de interpolación de los mismos. Para este propósito se utilizará el método *kriging*, un método de inferencia espacial que permite estimar los valores de los datos en los lugares no muestreados. Para ello, el método utiliza los valores que sí que se le proporcionan, devolviendo el mejor estimador lineal no sesgado, con una varianza mínima.

Para el cálculo del *kriging* en este proyecto, así como la generación de los mapas de calor con los resultados, se utilizará la herramienta *PyKrige* [26]. Esta herramienta está desarrollada en *Python* y soporta tanto *kriging* en 2D como en 3D, aunque en este caso únicamente se utilizará el 2D.

PyKrige soporta la interpolación de los datos siguiendo diferentes variogramas como, por ejemplo, el esférico o el exponencial, pero en este caso se obta por seguir un variograma lineal. Este variograma se define de la siguiente manera:

$$s \cdot d + n \tag{7.1}$$

Donde s representa la pendiente o el peso, d representa los valores de la distancia sobre los cuales se calcula el variograma, y n representa el *nugget*. Este último es el encargado de eliminar el ruido en las medidas, representando las desviaciones aleatorias que se pueden encontrar en una muestra de datos. El término *nugget* es una alusión a los orígenes del *kriging*. Anteriormente utilizado en la búsqueda de oro, el *nugget* hace referencia a la posibilidad de encontrar una pepita de oro aleatoriamente, aumentando enormemente el valor de un punto en comparación a su alrededor.

Se ha optado por este variograma siguiendo la idea de que la contaminación se concentra en su foco, donde es máxima, y disminuye linealmente a medida que se toman muestras más alejadas de dicho foco.

7.2 Barrido completo del área

En esta prueba, el dron comenzará desplazándose hacia una esquina e irá celda por celda tomando medidas de la superficie completa, sin tener en cuenta ningún tipo de medida de contaminación. Finalmente volverá a la ubicación central y aterrizará. Esta prueba obviamente será la que más tiempo necesite, pero también la que mayor precisión aportará, por lo que es una prueba fundamental para comprobar el funcionamiento del algoritmo. Los resultados de esta prueba serán los que se usarán para comparar la efectividad de las siguientes.

Tabla 7.2: Resultados en un barrido del área completa.

Simulación 1 (Realista)		**Simulación 2** (Reducida)	
Tiempo de simulación	11h, 17min y 20s (40640s)	Tiempo de simulación	32min y 51s (1971s)
Puntos medidos	1600 (100 %)	Puntos medidos	100 (100 %)

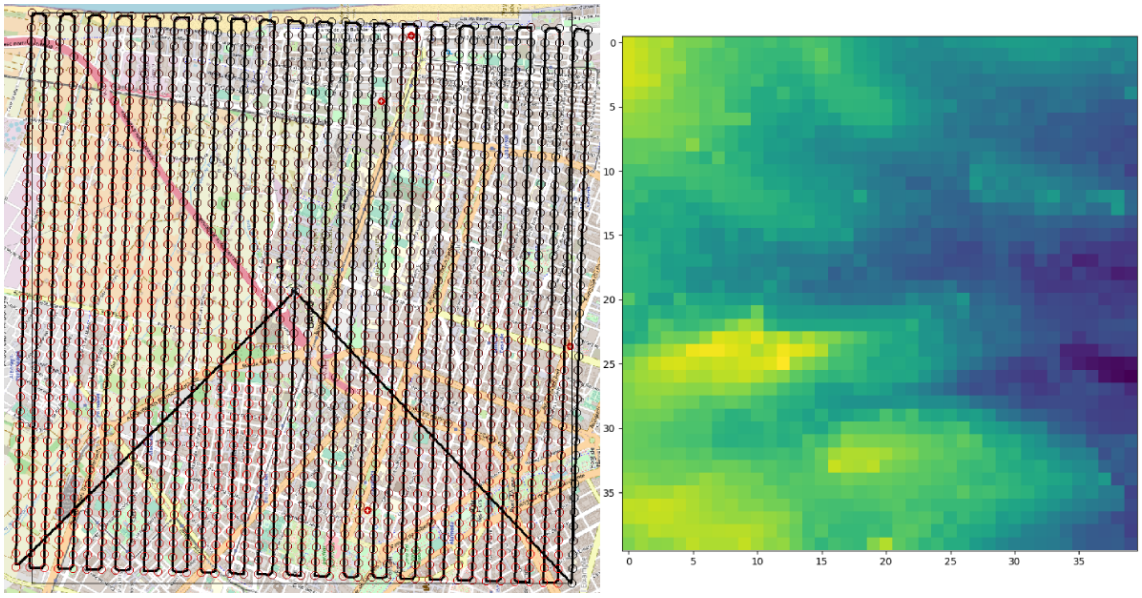


Figura 7.1: Recorrido seguido por el dron en un barrido del área completa. Simulación 1.

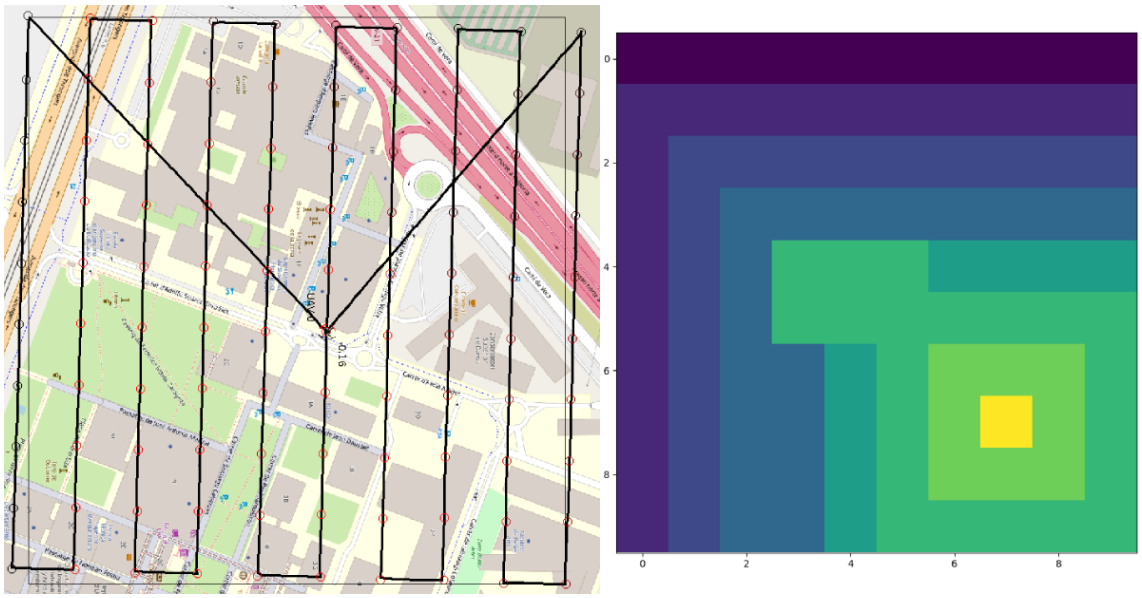


Figura 7.2: Recorrido seguido por el dron en un barrido del área completa. Simulación 2.

7.3 Vuelo aleatorio

En esta prueba, el dron simplemente se dirige por el área de forma completamente aleatoria, sin tener el cuenta ningún tipo de medida de contaminantes. Para que sea una comparación más justa, el experimento terminará una vez recorrido el 25 % del área, evitando semejanzas con la prueba anterior. Además, el dron puede moverse a cualquier celda situada en un radio de 3 celdas alrededor de la actual. Si esto no es posible, se aumentará el radio sucesivamente.

Tabla 7.3: Resultados tras un movimiento aleatorio.

Simulación 1 (Realista)		**Simulación 2** (Reducida)	
Tiempo de simulación	4h, 48min y 7s (17287s)	Tiempo de simulación	12min y 50s (770)
Puntos medidos	400 (25 %)	Puntos medidos	25 (25 %)

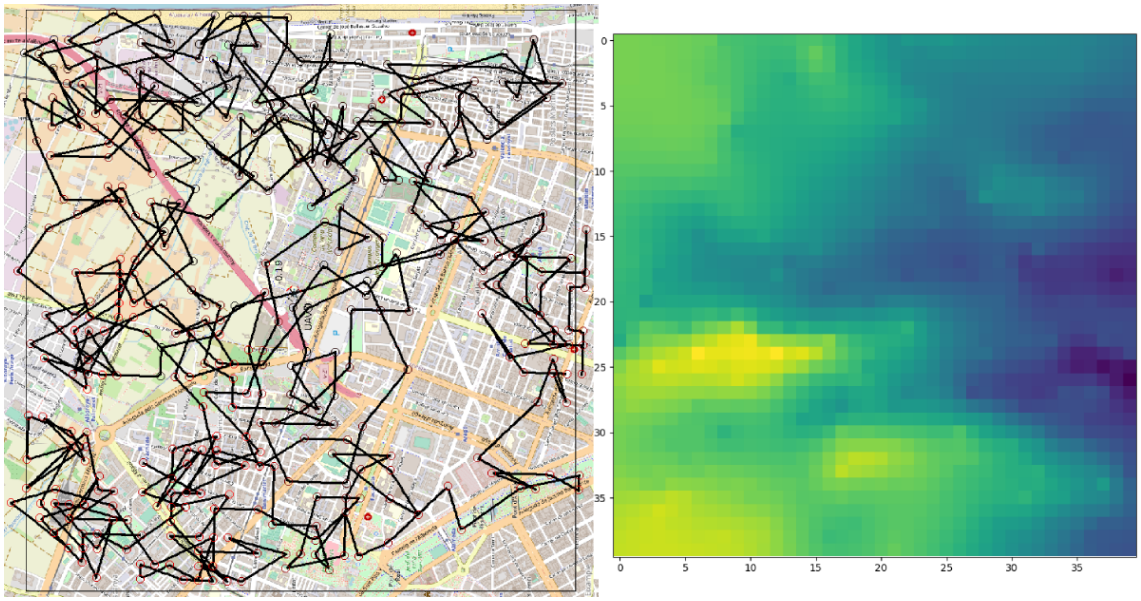


Figura 7.3: Recorrido seguido por el dron con un movimiento aleatorio. Simulación 1.

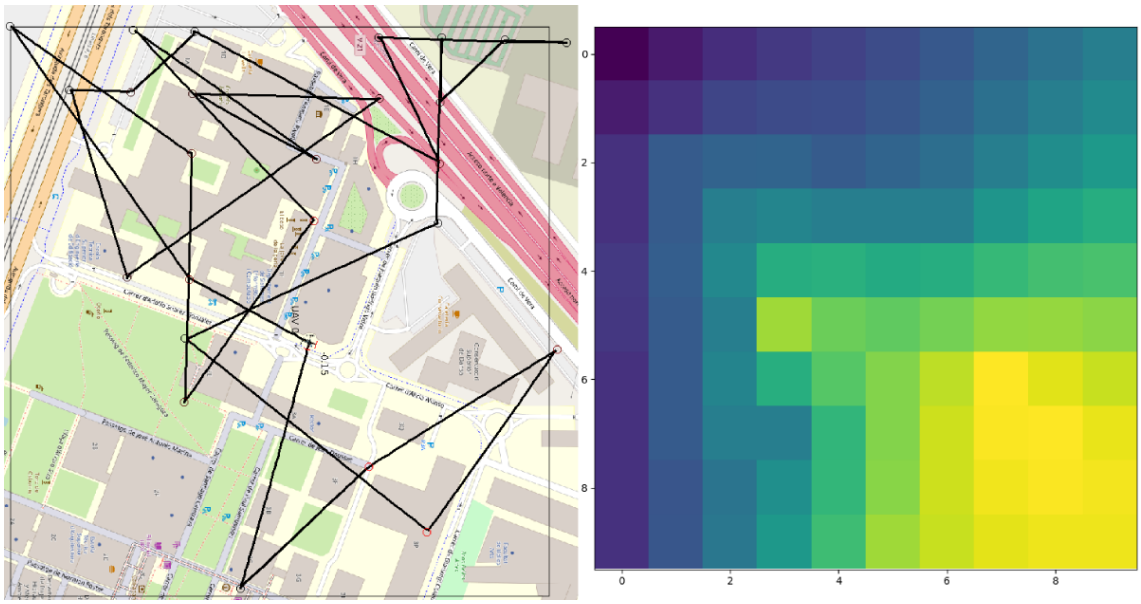


Figura 7.4: Recorrido seguido por el dron con un movimiento aleatorio. Simulación 2.

7.4 Recorrido en espiral

En esta prueba, el dron ejecutará únicamente su método de *Explore*. Esto significa que siempre realizará el mismo recorrido independientemente de la con-

taminación medida. Se espera que sea la prueba que lleve menos tiempo, pero también se espera perder precisión en el mapa de calor. Esta precisión perdida se espera que sea especialmente notable en el punto de contaminación máxima. Se puede destacar en el recorrido del dron los cambios de dirección de la espiral. Esto es debido a su implementación y a su selección por el punto más cercano.

Tabla 7.4: Resultados tras un movimiento en espiral.

Simulación 1 (Realista)		**Simulación 2** (Reducida)	
Tiempo de simulación	1h, 16min y 47s (4607)	Tiempo de simulación	12min y 8s (728)
Puntos medidos	97 (6 %)	Puntos medidos	28 (28 %)

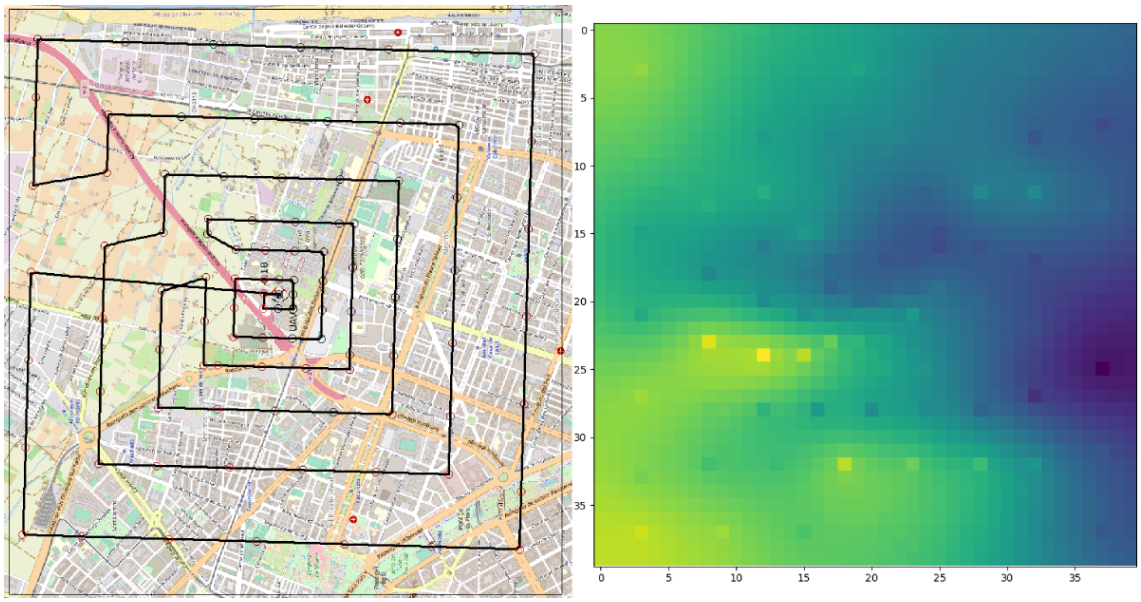


Figura 7.5: Recorrido seguido por el dron con un movimiento en espiral. Simulación 1.

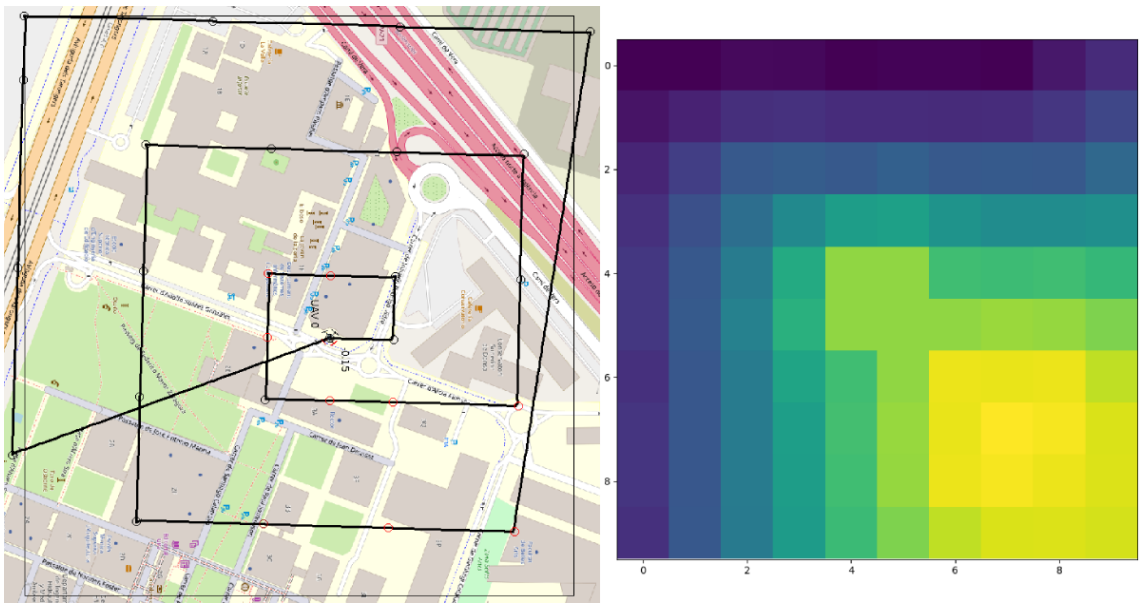


Figura 7.6: Recorrido seguido por el dron con un movimiento en espiral. Simulación 2.

7.5 Vuelo inteligente

En este punto es donde se pondrá a prueba la totalidad del proyecto. Aquí el dron utilizará ambas fases del algoritmo PdUC-D para realizar un vuelo informado y tomar decisiones en cuanto a la dirección de sus movimientos. Se espera obtener un buen balance entre el tiempo de ejecución y precisión de los datos.

Tabla 7.5: Resultados tras ejecutar el algoritmo PdUC-D.

Simulación 1 (Realista)		**Simulación 2** (Reducida)	
Tiempo de simulación	1h, 29 min y 1s (5341s)	Tiempo de simulación	10 min y 7 s (607)
Puntos medidos	116 (7,25 %)	Puntos medidos	25 (25 %)

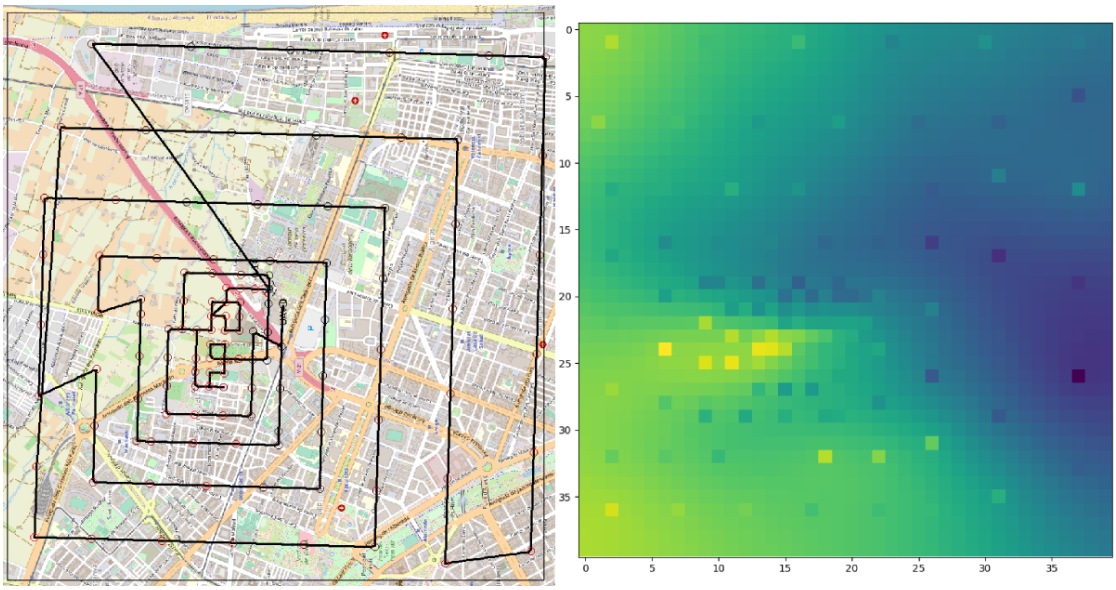


Figura 7.7: Recorrido seguido por el dron usando el algoritmo PdUC-D. Simulación 1.

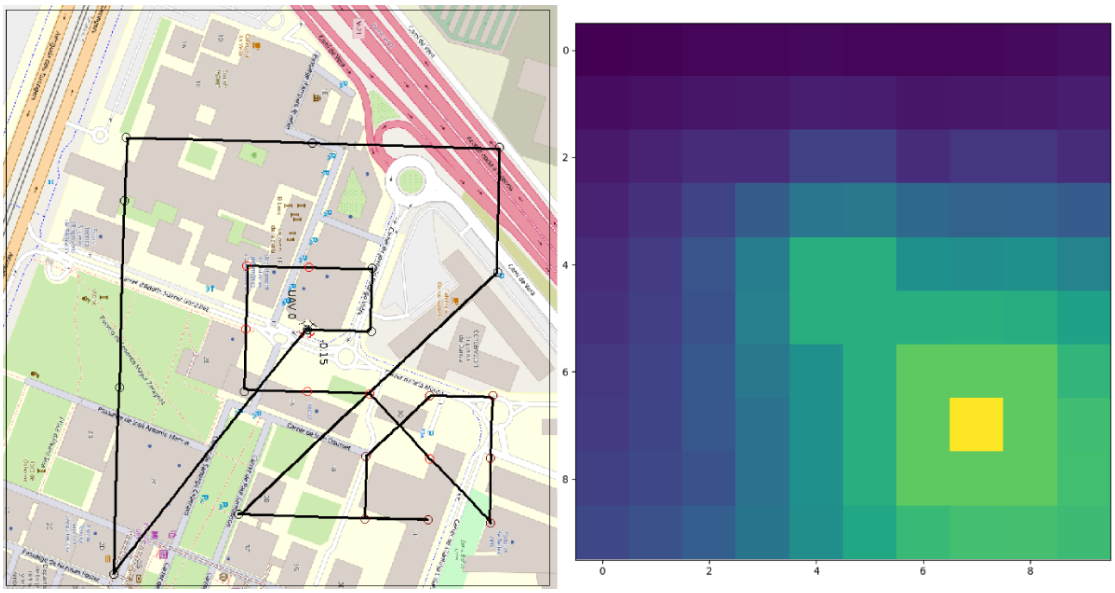


Figura 7.8: Recorrido seguido por el dron usando el algoritmo PdUC-D. Simulación 2.

7.6 Análisis de los resultados

7.6.1. Simulación 1

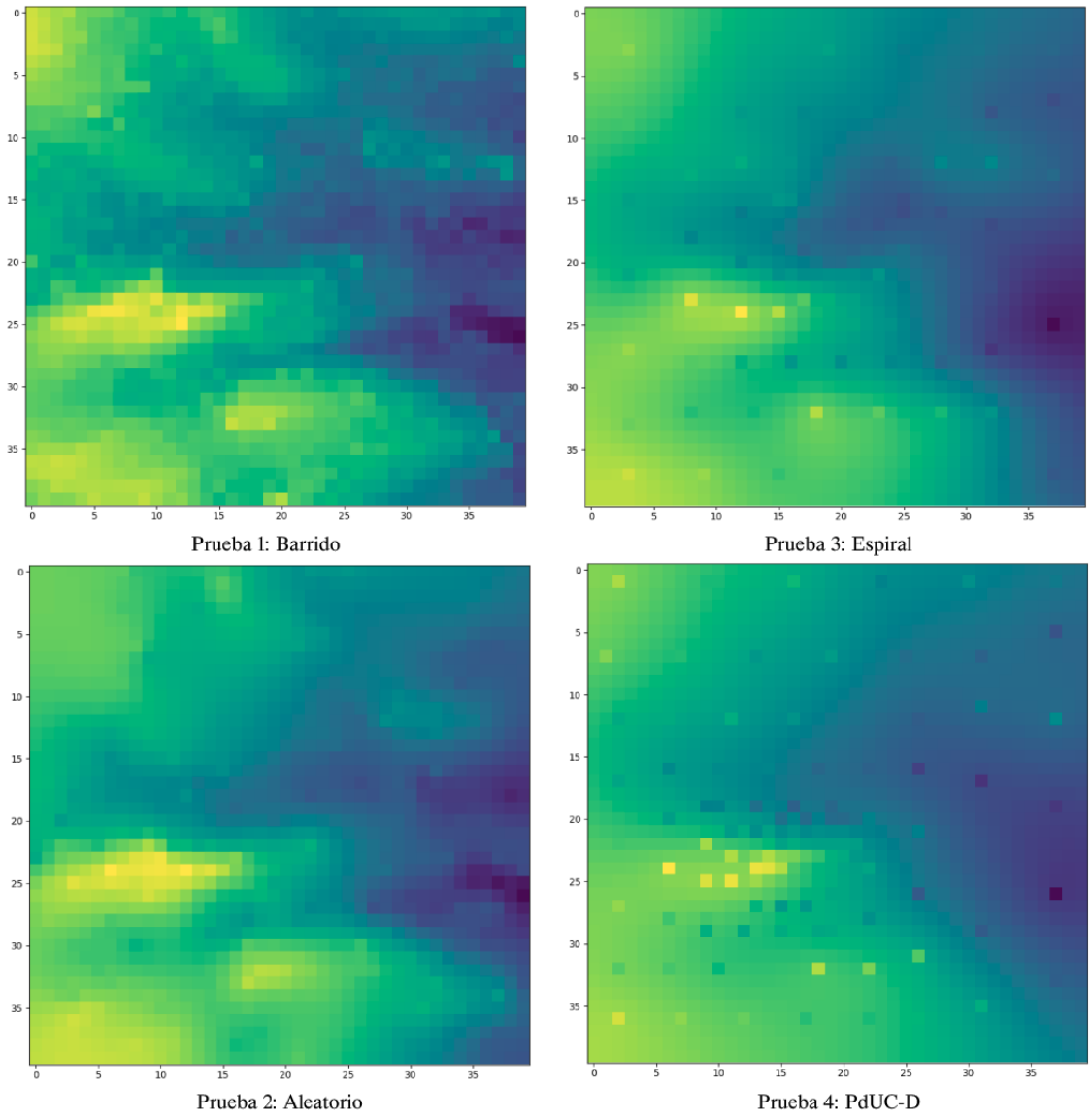


Figura 7.9: Comparativa entre las pruebas. Simulación 1.

En esta simulación, tanto la prueba 1 como la prueba 2 son completamente inviables en un escenario real. El tiempo de vuelo es demasiado alto para las baterías de los drones. Cabe destacar la precisión del mapa de calor del vuelo aleatorio, por lo que se podría asumir que con una medición del 25 % de los datos se pueden obtener datos aceptables. Aún así, el uso de un protocolo de vuelo aleatorio no sería admisible en este proyecto, ya que no se tendría ningún tipo de garantía de la fiabilidad de los datos. Es posible que en otra ejecución se hubieran obtenido datos completamente imprecisos.

En el caso de la prueba 3, se observa una clara pérdida de precisión en el mapa de calor. En este caso, este error resulta asumible, ya que el punto de máxima concentración se encuentra cercano al centro, que es donde más precisión consi-

gue este movimiento. No ocurre lo mismo en la simulación 2, como se verá más adelante. El tiempo de simulación ya se sitúa en unos márgenes aceptables para un uso real, necesitando solo el 11,34 % del tiempo de hacer un barrido completo.

En el caso de la prueba 4 (algoritmo PdUC-D) se puede ver mejor definición en los puntos de contaminación máxima. Ofrece una precisión ligeramente superior que la prueba 3, necesitando el 13,14 % del tiempo del barrido, es decir, es un poco más lento que la prueba anterior.

7.6.2. Simulación 2

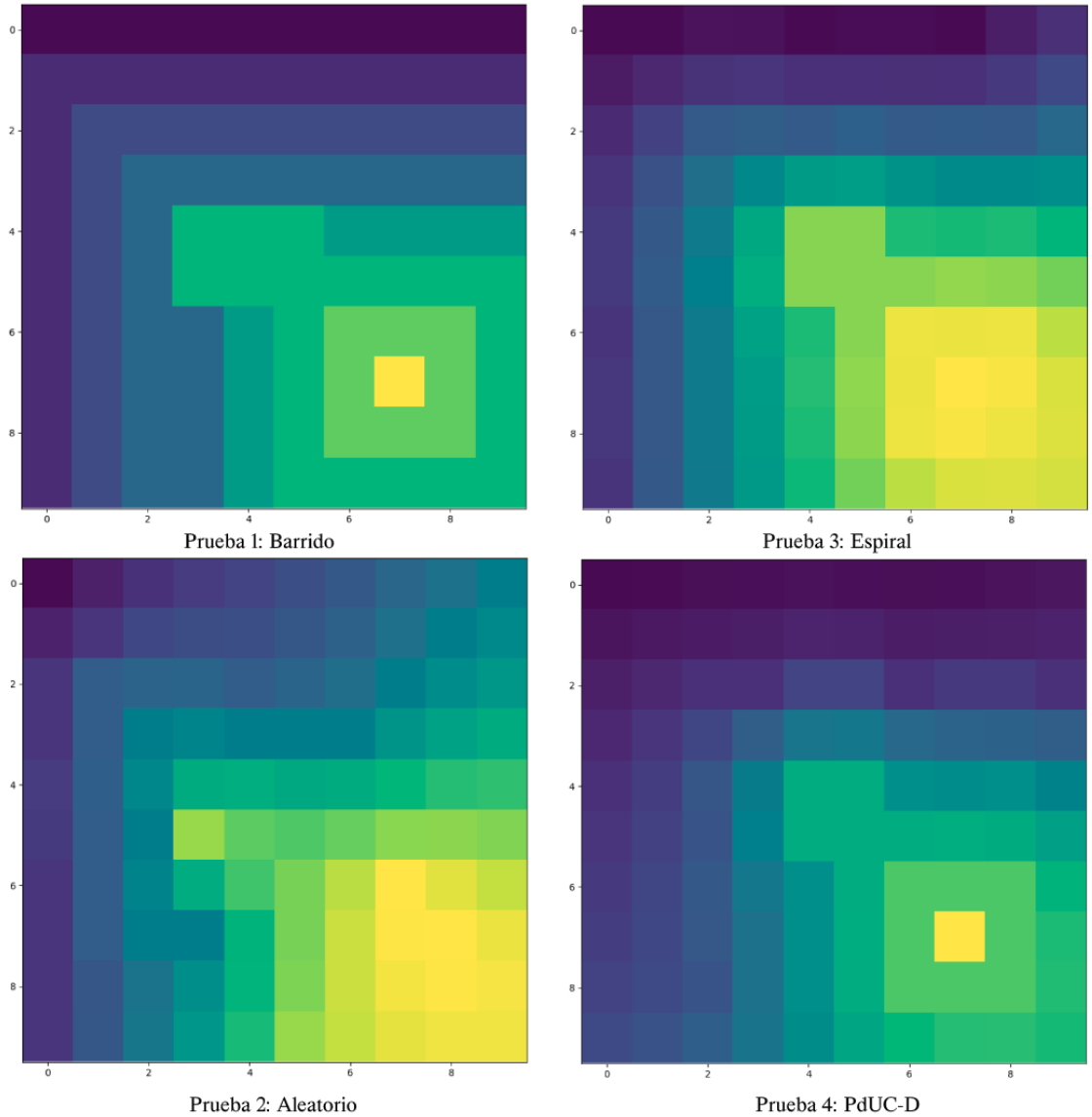


Figura 7.10: Comparativa entre las pruebas. Simulación 2.

En este caso, la prueba aleatoria obtiene la peor precisión en los datos. Como ya se ha comentado en el punto anterior, esto puede variar enormemente entre ejecuciones, lo que hace esta forma de movimiento inviable, aunque claramente consigue un buen tiempo.

La prueba 3 continua obteniendo unos tiempos bastante cortos, suponiendo un 36,94 % del tiempo del barrido. En este caso el foco se encuentra alejado del punto central, por lo que tal y como se ha comentado en el apartado anterior, la precisión es bastante baja. Ni en esta prueba ni en la prueba 2 se puede diferenciar correctamente el foco contaminante.

Para terminar, la prueba de PdUC-D se trata de la ganadora indiscutible. Se trata de la única prueba que muestra correctamente el foco de contaminación, de una forma clara y definida. Además, contrario a lo esperado, también se trata de la prueba que obtiene una mayor eficiencia, necesitando únicamente el 30,79 % del tiempo de barrido.

En esta simulación el tiempo de ejecución pierde importancia, ya que al tratarse de un área tan reducida todos los tiempos son asumibles en ejecuciones con drones físicos.

CAPÍTULO 8
Conclusiones y trabajos futuros

En este proyecto se ha desarrollado un protocolo para el uso de UAV en la medición de datos de contaminación atmosférica. Estos vehículos se desplazan de forma autónoma siguiendo las directrices definidas por el algoritmo PdUC-D. De esta forma, se busca usar los drones para combatir el cambio climático.

Tras finalizar el proyecto, se puede confirmar que se han cumplido todos los objetivos definidos en el primer capítulo. Se ha completado el desarrollo de un protocolo para el simulador ArduSim, lo cual ha involucrado una muy importante primera fase de investigación y búsqueda de información, una segunda fase de desarrollo, y una tercera fase de pruebas. Las tres fases han sido completadas satisfactoriamente.

Observando las pruebas realizadas, se puede confirmar el buen funcionamiento del algoritmo desarrollado, así como su cohesión dentro del simulador ArduSim. Se ha demostrado que es capaz de conseguir resultados altamente precisos con un tiempo de uso bastante limitado.

8.1 Trabajos futuros

Como posibles trabajos futuros se pueden diferenciar dos ramas:

- **Mejoras del algoritmo**
  Pese a los buenos resultados obtenidos con este algoritmo, todavía se puede mejorar su eficiencia.

  Una mejora clara consiste en calcular la forma más eficiente de recorrer los puntos durante la espiral. Actualmente, durante la espiral, el dron siempre se mueve al punto más cercano. Esta aproximación ofrece buenos resultados de forma general, pero se podría mejorar si primero se tomaran todos los puntos que debe recorrer en cada iteración y se calculara el camino más corto.
  Otra posible mejora sería extender el algoritmo para que opere en 3D, de cara a obtener información de contaminación tridimensional y no solo en un mismo plano, para aquellos escenarios donde la contaminación se esté generando al nivel del suelo, y sea de interés comprobar como reduce sus niveles con la altura.

- **Añadir funcionalidades**
  Durante las pruebas del protocolo, se ha observado la necesidad de añadir diversas funcionalidades extra. Por ejemplo, la posibilidad de comenzar la simulación en cualquier punto seleccionado por el usuario, en vez de comenzar siempre por el centro.

  Otra funcionalidad sería la integración del script de PyKrige en el protocolo, para no depender de una aplicación externa para obtener el mapa de calor con los resultados.

  Por último, también sería interesante establecer un límite de tiempo de ejecución, para que la cantidad de batería que vaya a ser utilizada (teóricamente), pueda ser controlada también por el usuario.

Además, también queda pendiente una prueba con drones reales ya que, gracias al funcionamiento de este simulador, se podría utilizar este protocolo sin hacerle ningún cambio. Simplemente haría falta instalar ArduSim en una Raspberry Pi montada en un dron, y así validar que el protocolo se comporta como es debido en entornos reales.

Bibliografía

[1] O. Alvear, N. R. Zema, E. Natalizio, C. T. Calafate. Using UAV-Based Systems to Monitor Air Pollution in Areas with Poor Accessibility. *Journal of Advanced Transportation*, 10.1155/2017/8204353, agosto, 2017.

[2] Alvear-Alvear, Ó.; Tavares De Araujo Cesariny Calafate, CM.; Zema, N.; Natalizio, E.; Hernández-Orallo, E.; Cano, J.; Manzoni, P. (2018). A Discretized Approach to Air Pollution Monitoring Using UAV-based Sensing. *Mobile Networks and Applications.*, 23(6):1693-1702, 2018.

[3] Fabra Collado, FJ.; Tavares De Araujo Cesariny Calafate, CM.; Cano, J.; Manzoni, P. (2018). ArduSim: Accurate and real-time multicopter simulation. *Simulation Modelling Practice and Theory.*, 87:170-190, junio, 2018.

[4] F. Fabra, J. Wubben y C. T. Calafate. *Repositorio ArduSim.* https://github.com/GRCDEV/ArduSim.

[5] C. Giannini, A. A. Shaaban, C. Buratti, R. Verdone. Delay Tolerant Networking for smart city through drones. *Proceedings of the International Symposium on Wireless Communication Systems*, 10.1109/ISWCS.2016.7600975, octubre, 2016.

[6] Todos los tipos de drones según el uso, diseño o control. *Consultado en* https://www.adslzone.net/reportajes/drones/tipos-drones/.

[7] DIRECTIVA 2008/50/CE DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 21 de mayo de 2008 relativa a la calidad del aire ambiente y a una atmósfera más limpia en Europa. *Consultado en* https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32008L0050&from=en.

[8] ¿Cómo se mide la contaminación del aire? *Consultado en* https://enviraiot.es/como-se-mide-la-contaminacion-del-aire/.

[9] La medición de contaminantes. *Consultado en* https://cidta.usal.es/riesgos/CD1/control_contaminacion_aire/www.cepis.ops-oms.org/bvsci/e/fulltext/orienta2/cap7c.pdf.

[10] Sistemas que sirven para medir y analizar la contaminación del aire. *Consultado en* https://ecologiahoy.net/medio-ambiente/sistemas-que-sirven-para-medir-y-analizar-la-contaminacion-del-aire/.

[11] Ventajas del uso de sensores para medir la calidad del aire exterior. *Consultado en* https://enviraiot.es/medir-calidad-aire-exterior-sensores/.

[12] Premio 'Ciudades Inteligentes' para un sensor 'made in Barcelona' que mide la contaminación urbana. *Consultado en* https://www.elmundo.es/ciencia/2013/11/23/529095c363fd3de25a8b4570.html.

[13] Smart Citizen - Sensores ciudadanos. *Crowdfunding:* https://www.goteo.org/project/smart-citizen-sensores-ciudadanos.

[14] Fabra Collado, FJ. Flight coordination solutions for multirotor unmanned aerial vehicles. *Universitat Politècnica de València.* http://hdl.handle.net/10251/147857.

[15] Real Decreto 1036/2017, de 15 de diciembre, por el que se regula la utilización civil de las aeronaves pilotadas por control remoto. *Consultado en* https://www.boe.es/boe/dias/2017/12/29/pdfs/BOE-A-2017-15721.pdf.

[16] A. D. Team. *SITL Simulator (Software in the Loop).* http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[17] Fabra F, Zamora W, Sangüesa J, Calafate CT, Cano J-C, Manzoni P. A Distributed Approach for Collision Avoidance between Multirotor UAVs Following Planned Missions. *Sensors (2019).* https://doi.org/10.3390/s19102404.

[18] Fabra Collado, FJ.; Zamora, W.; Reyes, P.; Sangüesa, JA.; Tavares De Araujo Cesariny Calafate, CM.; Cano, J.; Manzoni, P. MUSCOP: Mission-Based UAV Swarm Coordination Protocol. *IEEE Access (2020).* http://hdl.handle.net/10251/164054.

[19] Fabra Collado, FJ.; Zamora, W.; Masanet, J.; Tavares De Araujo Cesariny Calafate, CM.; Cano, J.; Manzoni, P.. Automatic system supporting multicopter swarms with manual guidance. *Computers & Electrical Engineering. (2019)* http://hdl.handle.net/10251/156662.

[20] Wubben, J.; Fabra Collado, FJ.; Tavares De Araujo Cesariny Calafate, CM.; Krzeszowski, T.; Márquez Barja, JM.; Cano, J.; Manzoni, P. Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. *Electronics (2019).* http://hdl.handle.net/10251/144317.

[21] Wubben, J.; Aznar, P.; Fabra Collado, FJ.; Tavares De Araujo Cesariny Calafate, CM.; Cano, J.; Manzoni, P. Toward secure, efficient, and seamless reconfiguration of UAV swarm formations. *IEEE. 1-7 (2020).* http://hdl.handle.net/10251/179834.

[22] I. Boussa ïD, J. Lepagnot, and P. Siarr. A survey on optimization metaheuristics. *Information Sciences. An International Journal*, pp. 82–117, 2013.

[23] Canonical. *Distribución ubuntu.* https://ubuntu.com/.

[24] Eclipse Foundation. *Eclipse IDE.* https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-developers.

[25] Gluon. *Scene Builder.* https://gluonhq.com/products/scene-builder/.

[26] PyKrige developers. *Kriging Toolkit for Python.* https://geostat-framework.readthedocs.io/projects/pykrige/en/stable/.

APÉNDICE A
Objetivos de Desarrollo Sostenible

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

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

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

En este proyecto se ha buscado una solución para facilitar la localización de la contaminación atmosférica. Desde un primer momento se ha tenido en mente el objetivo 3 (Salud y bienestar), ya que dicha contaminación ha ido creciendo exponencialmente durante los últimos años, convirtiéndose en un gran problema para la salud pública. Es de vital importancia tomar las medidas necesarias para reducir la contaminación ambiental, por lo que la búsqueda de los focos más contaminantes es de gran ayuda para tomar dichas medidas de una forma informada y eficaz.

Para conseguir esta mejora de la calidad de vida de las personas, este proyecto se ha centrado en el objetivo 11 (Ciudades y comunidades sostenibles). El primer paso para mejorar la salud de las personas es trabajar sobre los lugares donde viven. No es admisible encontrar altos niveles de contaminación en los núcleos urbanos, por lo que esto es algo que necesita ser investigado y controlado a fondo.

Este TFG también está relacionado con los objetivos 9 (Industria, innovación e infraestructuras) y 12 (Producción y consumo responsables), ya que puede ser utilizado para comprobar la sostenibilidad de las industrias y la infraestructura de las mismas. Un dron puede estar encargado de medir los contaminantes que se generan mediante el proceso de producción industrial, asegurándose así de que en ningún momento se rebasan los límites establecidos de contaminación. Esto ayudaría también si en algún momento hay algún aumento en la contaminación, facilitando la detección y posterior actuación, ya sea mediante multas a las empresas involucradas, o mediante una reducción de los niveles de producción hasta encontrar un punto estable.

Pero no solo se ha tenido en cuenta la salud de los humanos, sino que también se ha tenido en mente la salud del propio planeta, trabajando en el objetivo 13 (Acción por el clima). El cambio climático es un problema cada vez mayor, y la culpa de esto reside en la contaminación generada por los humanos. La temperatura global no deja de crecer a un ritmo cada vez mayor y aparecen fenómenos atmosféricos con cada vez más frecuencia. Si bien es cierto que actualmente resultaría imposible deshacer todo hecho hasta ahora, el objetivo debería ser reducir dicha contaminación que se genera a lo mínimo posible, intentando disminuir al mínimo el cambio climático.

Por otro lado, la vida de los ecosistemas terrestres es una de las más afectadas por la contaminación atmosférica. Es por esto que el objetivo 15 también cobra gran importancia en este proyecto. También es importante tener los niveles de contaminación controlados en medios rurales, ya que al final, la contaminación ambiental puede afectar de igual forma tanto a los animales como a las personas. De igual forma, aunque en menor medida, ocurre con la vida submarina (objetivo 14). Está demostrado que parte de la contaminación presente en el aire termina siendo absorbida por mares y ríos, terminando en el organismo de los seres vivos. Debido a eso, el objetivo 6 (Agua limpia y saneamiento) también resulta cercano a este trabajo, ya que reduciendo los niveles de contaminación atmosférica también se lograría reducir un poco la contaminación presente en el agua.

Adicionalmente, el calentamiento global está afectando a la temperatura de los mares, destruyendo los ecosistemas acuáticos. Todo esto sin contar el deshielo, una problemática también en auge que actualmente está destruyendo la vida en los ecosistemas gélidos, pero que en un futuro comenzará a afectar tanto a los ecosistemas marinos como terrestres, al aumentar los niveles del mar.

Para terminar, también se encuentra relación con el objetivo 7 (Energía asequible y no contaminante), ya que los drones utilizan energía eléctrica, mucho más sostenible que otros vehículos de combustión, y en menor medida debido a su tamaño. Además, también pueden ser utilizados para monitorizar los procesos de generación de energía, para asegurar, al igual que con el resto de indústrias, que se haga de forma sostenible.