UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escuela Técnica Superior de Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática

Universidad Politécnica de Valencia

Desarrollo, evaluación y modelado de un sistema de comunicaciones para multicópteros

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Fabra Collado, Francisco José

Tutor: Tavares Calafate, Carlos Miguel

Curso 2015-2016

RESUMEN

El uso de multicópteros se está extendiendo rápidamente debido a los muchos campos de aplicación que tienen, por lo que cada vez más hay riegos de colisión entre estos aparatos. Además, se están empezando a proponer soluciones donde enjambres de multicópteros logren solucionar problemas más complejos mediante cooperación. Ambos casos requieren que dichos multicópteros se comuniquen entre ellos para coordinar sus trayectorias, algo que actualmente aún se está realizando solo a nivel experimental, por lo que cabe realizar más estudios para averiguar cuál sería la tecnología de comunicación más idónea, y cuáles son los factores que afectan a las prestaciones.

El principal objetivo de este proyecto es el desarrollo de un entorno que permita estudiar las comunicaciones entre multicópteros. Para eso se han modificado multicópteros existentes para dotarlos de comunicaciones inalámbricas, así como para permitir comunicar con el sistema de navegación de cara a obtener los parámetros de vuelo en todo momento.

El principal elemento desarrollado ha sido una aplicación distribuida que permite el análisis de la calidad del enlace de comunicaciones entre dos drones conectados mediante una red WiFi Ad-hoc, la cual registra no solo la tasa de pérdida en el canal, sino también las posiciones de los drones y demás parámetros de vuelo. Como complemento a la herramienta propuesta, se han desarrollado una serie de scripts que permiten automatizar el análisis de datos y la generación de gráficas, para permitir detectar y cuantificar la influencia que pueden ejercer distintos factores ambientales y parámetros de configuración en la calidad del enlace.

La aplicación está completada en su totalidad y ha sido probada en campo, permitiendo constatar la influencia de la distancia entre los drones, la velocidad de rotación de los motores y el tamaño de los paquetes de datos en la calidad del enlace de comunicación. Además, el estudio realizado ha permitido también hallar conclusiones interesantes: dado que los actuales mandos de control remoto trabajan en la banda de los 2,4 GHz, interfieren con la señal WiFi en esta banda y tienen un impacto importante en las prestaciones, por lo que se recomienda utilizar la banda de los 5,9 GHz.

Palabras clave: Dron, Pixhawk, Raspberry Pi, WiFi, Ad-hoc, MAVLink

ABSTRACT

The use of multicopters is spreading rapidly due to their wide scope of application. This means that, as time goes by, the risk of collision between these devices significantly encreases. Moreover, solutions where swarms of multicopters are used to solve more complex problems through cooperation are being proposed. Both cases require multicopters to communicate among them to coordinate their flight, something that is only being done at an experimental level at this moment. Thus, so more studies are required to determine which would be the most appropriate communication technology, and which are the factors affecting performance.

The main objective of this project is to develop an environment that allows studying the communications among multicopters. For that purpose, existing multicopters have been modified to provide them wireless communication capabilities, and to make them able to communicate with the navigation system to retrieve flight parameters at any time.

The main element developed was a distributed application that allows analysing the quality of the communication link between two drones connected by means of an WiFi-based ad-hoc network, and that records not only the packet loss ratio, but also the drone's position and other flight parameters. As a complement to the proposed tool, a set of scripts that allow automating the data analysis and graphics generation were developed, thereby allowing to detect and quantify the influence that different environmental factors and configuration parameters have on link quality.

The application has reached its final development status and has been field tested, allowing to study the influence of the distance between the drones, the engine rotational speed and the packet size in the communication's link quality. Moreover, the study has also allowed to draw interesting conclusions: since most radio remote controllers work in the 2.4 GHz band, they strongly interfere with the WiFi signal in this band, and thus have a significant impact on performance, and so we recommend using the 5.9 GHz band.

Keywords: Drone, Pixhawk, Raspberry Pi, WiFi, Ad-hoc, MAVLink

Glosario

WiFi. Mecanismo de conexión inalámbrica entre dispositivos en las bandas de frecuencia de 2,4 y/o 5,9 GHz.

Red Ad-hoc. Red de comunicaciones descentralizada donde las estaciones comunican directamente entre sí.

Punto de acceso. Dispositivo que hace de puente de comunicación entre dispositivos inalámbricos y entre éstos y alguna otra tecnología, como puede ser Ethernet sobre cable.

Zigbee. Conjunto de protocolos diseñados para la transmisión inalámbrica a baja tasa de envío de datos y bajo consumo de energía.

Dron. Vehículo aéreo no tripulado controlado por radiofrecuencia, normalmente capaz de realizar vuelos programados.

Multirrotor. Variante de dron que se caracteriza por disponer de 3 o más rotores, lo cual le permite tener una elevada estabilidad en el aire.

Pixhawk. Dispositivo hardware diseñado para el control de vuelo y navegación.

ArduPilot. Plataforma software instalada en un dispositivo tipo Pixhawk para el control de drones.

Raspberry Pi 2. Microordenador de bajo coste basado en la plataforma ARM diseñado con el objeto de fomentar el aprendizaje de las ciencias de computación en las escuelas. Dispone de puertos USB, Ethernet y suficiente capacidad de cómputo para realizar tareas medianamente complejas.

Raspbian. Sistema operativo basado en Debian con el que funcionan los dispositivos Raspberry Pi.

Cabeceo o *pitch*. Ángulo de inclinación de una aeronave respecto al eje transversal, perpendicular a la dirección de desplazamiento. En el caso de los aviones corresponde, por ejemplo, a la elevación del morro durante el despegue.

Alabeo o *roll*. Ángulo de inclinación de una aeronave respecto al eje longitudinal (dirección de desplazamiento). En el caso de los aviones corresponde a la inclinación lateral cuando gira a la derecha o a la izquierda.

Guiñada o *yaw*. Ángulo de rotación de una aeronave respecto al eje vertical. En el caso de los aviones corresponde al rumbo.

MAVLink. Formato de mensajes y protocolo de comunicación diseñado para la transferencia de paquetes de control y de información desde y hacia drones.

AESA. Agencia Estatal de Seguridad Aérea, encargada de regular el vuelo de drones en el territorio español.

Índice de figuras

Índice de tablas

Índice de contenidos

1 Introducción

1.1 Motivación

Los drones son vehículos aéreos no tripulados (VANT) que realizan vuelos programados o dirigidos desde control remoto. Se trata de vehículos que han ganado gran popularidad a escala internacional durante los últimos años, siendo capaces de realizar gran variedad de tareas:

- Uso militar para reconocimiento y ataque
- Levantamientos topográficos
- Gestión de cultivos
- Lucha contra incendios
- Seguridad civil
- Rescate de personas
- Vigilancia de oleoductos
- Toma de muestras en volcanes
- Realización de filmaciones
- Uso recreativo

A medida que se extiende el uso de drones se detectan nuevas aplicaciones, entre las que se encuentran aquellas que requieren la actuación coordinada de varios de ellos. Por otro lado, la proliferación de drones también crea la necesidad de establecer protocolos y enlaces de comunicación entre los mismos para evitar su colisión cuando se encuentren relativamente cerca entre sí. Teniendo en cuenta estos factores, es más que pertinente estudiar tecnologías y protocolos mediante los cuales los drones puedan trabajar en equipo.

En este contexto, la tecnología WiFi surge como una opción viable ya que está ampliamente desarrollada y extendida, lo que la hace una alternativa adecuada para resolver la problemática planteada. Además, dado que los drones se desplazan por un volumen de espacio extenso durante sus vuelos, el uso de puntos de acceso (PA) queda totalmente descartado, ya que el enlace entre dron y PA podría perderse con facilidad. Por este motivo, la única alternativa viable a día de hoy es el uso de una red Ad-hoc sostenida por los mismos drones.

1.2 Objetivos y etapas del proyecto

En el contexto planteado, el presente trabajo tiene como **objetivo** el ofrecer un entorno que permita estudiar las comunicaciones entre multicópteros. Para lograrlo es necesario dotar a los multicópteros de comunicaciones inalámbricas, así como permitir comunicar con el sistema de navegación de cara a obtener los parámetros de vuelo en todo momento. Como pieza clave de cara a realizar los estudios necesarios, es necesario desarrollar una aplicación que permita medir la calidad del enlace entre dos drones conectados por una red WiFi Ad-hoc. Dicha aplicación debe ser robusta frente a fallos en la comunicación mediante un conjunto de pruebas que simulan las que se podrían realizar para cumplir la utilidad principal para la que se desarrolla

la aplicación. Además, cabe automatizar el análisis de resultados para permitir extraer las principales conclusiones a partir de un amplio conjunto de experimentos.

Para lograr el objetivo antes definido, los pasos seguidos durante el desarrollo del proyecto son los siguientes:

1. Establecimiento del enlace por puerto serie entre el sistema de navegación (Pixhawk) y el sistema embebido (Raspberry Pi 2).
2. Creación y configuración de una red Ad-hoc entre multicópteros.
3. Desarrollo de la aplicación Dronning y testeo simultáneo en el simulador.
4. Verificación del correcto funcionamiento de la aplicación en una red Ad-hoc.
5. Pruebas de campo.
6. Desarrollo de una serie de scripts para automatizar en análisis de datos y generación de gráficas.
7. Análisis de los datos obtenidos.

1.3 Estructura del documento

En el siguiente apartado se proporciona información sobre trabajos realizados en la presente área de investigación. A continuación, en el punto 3, se incluye información descriptiva sobre los drones o multirrotores empleados durante el desarrollo, así como sobre los trabajos necesarios para poder obtener información sobre el estado del dron desde el dispositivo que ejecuta la aplicación desarrollada.

En el punto 4 se detallan los pasos seguidos para configurar correctamente la red Ad-hoc entre los drones y un ordenador portátil, este último empleado para controlar cada una de las tomas de datos.

El punto 5 describe en profundidad la aplicación Dronning con un enfoque descendente. Empieza describiendo la estructura interna de la aplicación, y termina describiendo la interfaz gráfica y algunos detalles relevantes para el uso de la aplicación.

El punto 6 detalla las hipótesis consideradas para la planificación de las pruebas de funcionamiento, determina las variables que pueden influir en la calidad de la comunicación entre los drones, y establece el conjunto de pruebas a realizar para verificar si las variables consideradas realmente influyen.

En el punto 7 se explica el análisis de datos realizado, detallando la información obtenida de forma automática mediante macros de Microsoft Excel, y termina mostrando los resultados obtenidos de las pruebas realizadas.

En el punto 8 se indican las conclusiones inferidas del presente Trabajo Fin de Grado, y en el punto 9 se indican posibles vías de ampliación del estudio iniciado en este proyecto.

Tras la bibliografía se incluyen dos anexos. En el primer anexo se detalla el proceso seguido para utilizar un simulador de dron durante el *debugging* de la aplicación, y en el segundo anexo se justifican los cálculos del ángulo que forma el plano de sustentación de un dron respecto al otro dron en pleno vuelo y se incluye, como ejemplo, el conjunto de gráficos generados automáticamente por las macros anteriormente comentadas.

2 Trabajos existentes en el área

En relación con el uso de la tecnología IEEE 802.11, en [22] se demuestra mediante experimentos empíricos que la máxima productividad alcanzable está todavía lejos de su máximo teórico. Es más, se concluye que la adaptación automática de la tasa de envío que realizan los chipsets con el estándar 802.11n no se adapta lo suficientemente rápido a la elevada movilidad de los drones, y que la posición de la antena es crucial para lograr una buena calidad de enlace. Los mismos autores ampliaron más tarde el estudio [21] con mediciones detalladas en un entorno controlado y en campo, identificando muchos retos para las redes entre drones en términos de la capa física, MAC y de *routing*, incluyendo el análisis con varios tipos de antena y estrategias de envío de información. En [11] se presentan resultados experimentales sobre la eficiencia en comunicaciones con un salto entre drones y una estación en tierra, así como en comunicaciones con dos saltos (comunicación entre dos drones a través de la estación en tierra) comparando los modos infraestructura y mallada de 802.11a. Los autores informan de una gran varianza en la productividad, enfatizando la necesidad de realizar más mediciones y de buscar mejores soluciones para soportar mejor las conexiones dinámicas con varios saltos.

En relación al uso de tecnologías diferentes de IEEE 802.11, en [7] se estudia cómo los drones pueden adoptar el rol de eNodeB o de equipo de usuario (UE) para habilitar la comunicación entre drones, dejando en evidencia que las actuales redes LTE necesitan modificaciones importantes para lograr una mayor integración con drones que usen esta tecnología. En [30] se estudia la efectividad de interfaces radio compatibles con Zigbee para la comunicación entre drones, habiendo observado que factores medioambientales, la orientación de la antena, y la reflexión de la señal en el suelo son los aspectos que más afectan a la degradación de la señal.

Desde una perspectiva más general, en [57] se subraya que los canales de comunicación entre drones están concentrados principalmente en la línea de visión, teniendo la reflexión sobre el suelo un impacto mínimo en general. Además, el efecto Doppler se agrava por la velocidad relativa entre los drones, que potencialmente puede ser muy elevada. En general, los autores consideran que son necesarios estudios más profundos para identificar la tecnología más adecuada para el enlace entre drones.

3 Equipamiento utilizado

3.1 El multirrotor GRCQuad

Para la toma de datos se han empleado dos unidades de multirrotor GRCQuad.

En ambas unidades se ha instalado una Raspberry Pi 2 en su parte superior para dotarles de capacidad de comunicación Wi-Fi y de capacidad de cómputo suficiente para realizar la toma de datos.

3.1.1 Caracterización

Se trata de una aeronave multirrotor en configuración X4 Quadcopter [42], con peso inferior a 2 kg, de la marca Quaternium modelo GRCQuad v1. Este modelo usa el frame DJI F330, y está propulsado por 4 motores eléctricos de 150W de potencia, con control electrónico de velocidad de 15 A, y dispone de batería LiPo de 14,8 V y 3.300MAh de alta descarga para la alimentación de los propulsores. Dispone de control de eje de cabeceo /pitch, desplazamiento lateral o alabeo (roll) y guiñada (yaw), así como controladora de estabilización giroscópica y GPS. Se controla mediante transmisor de radio control de 7 canales en la banda de 2,4 GHz.

La aeronave ofrece 5 modos de vuelo: Modo estabilizado GPS, Modo de sensor de Altitud, Modo estabilizado GPS+altura (Loiter), Modo de vuelta a casa, y Modo misión (automático).

Como medidas de seguridad, dispone de un sistema de protección por pérdida de enlace radio y por bajo nivel de batería.

Las baterías empleadas no se dañan con facilidad cuando se descargan rápidamente durante su uso. Sin embargo, tienen como inconveniente una limitada autonomía, lo que ha condicionado la planificación de la toma de datos.

Tabla 1: Prestaciones del multirrotor GRCQuad.

AERONAVE	Fabricante	Modelo	Identificador	Categoría
	Quaternium	**GRCQuad**	**CSM1150001 CSM1150002**	**RPA < 2 kg**
MOTOR	Tipo	Modelo/Kv	Potencia	Alimentación
	Eléctrico	**SunnySky X2212-980v**	**250 W**	**DC 12 V**
VARIADOR ELECTRÓNICO	Tipo	Modelo	Amp. Cont/máx	Entrada DC
	ESC	**DJI**	**15 A**	**5,6 a 12 V**
HÉLICE	Fabricante	Diámetro	Paso	Material
	GenFan	**8”**	**4,5”**	**Nylon carbón reinforced**
PESOS	En vacío	En vacío + batería	Carga máxima	MTOW
	831 g	**1196 g**	**604 g**	**1800 g**
DIMENSIONES	Anchura total	Anchura diagonal	Altura	Distancia ejes motor
	560 mm	**360 mm**	**160 mm**	**250 mm**
PRESTACIONES	Velocidad máx. ascenso	Velocidad máx. descenso	Velocidad máx. giro horizontal	Velocidad máx. de vuelo
	6 m/s	**3 m/s**	**200 º/s**	**10 m/s**
	Ángulo máx. inclinación		Autonomía (bat. 3,3 Ah)	
	45ª		**7 minutos**	

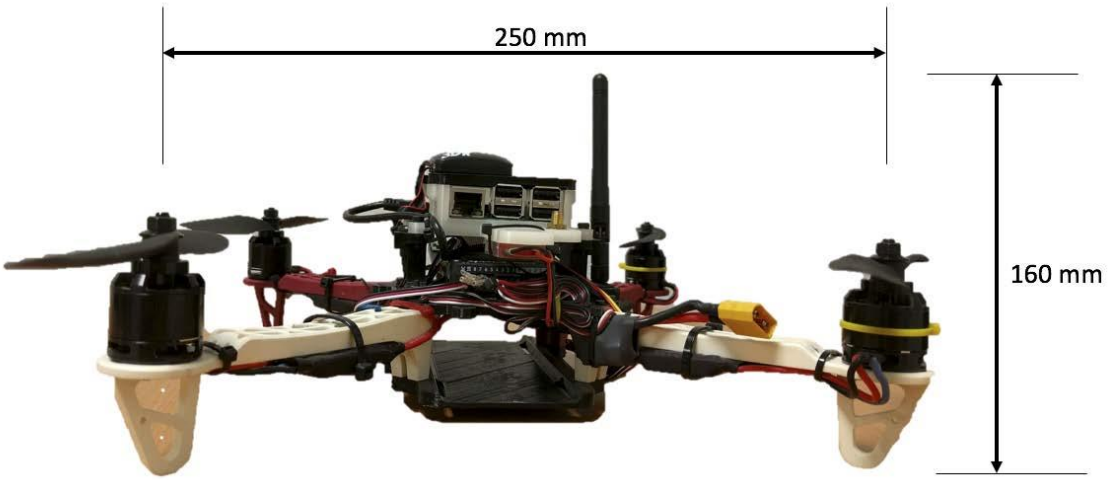


Figura 1: Vista lateral del multirrotor GRCQuad.

Figura 2: Vista superior del multirrotor GRCQuad.

Figura 3: Vista desde el motor delantero derecho del mutirrotor GRCQuad.

Figura 4: Vista desde el motor trasero izquierdo del mutirrotor GRCQuad.

3.1.2 Listado de componentes y equipos de la aeronave

Tabla 2: Componentes del multirrotor GRCQuad.

#	Descripción	Marca	Modelo	Datos
1	Propulsores eléctricos x4	SunnySky	X2212-980Kv	150 w, salida 15 A, empuje 0,96 Kg
2	Batería alimentación propulsor	Gens ace	4S1P – 3,3Ah	LiPo 3300 mAh 14,8V 25C 4S1P
3	Regulador electrónico de velocidad x4		15 A	Bec 5 V – 3 A, 15-20 A, 5,6-12 V
4	Controladora de vuelo	3DR	Pixhawk	3D ACC / Gyro / MAG / Baro
5	Compass	3DR	u-blox GPS with compass	u-blox NEO-7 module, 5 Hz update rate, HMC5883L compass
6	Telemetría	3DR	Radio Telemetry Kit 433 MHz	33 MHz Micro-USB port, 6-position DF12 connector, 100 mW máx., -117 dBm, RP-SMA connector, MAVLink protocol framing
7	Enlace radio	FrSky	X8R	8/16 Ch S.BUS ACCST Telemetry Receiver with Smart port
8	Sistema embebido	Raspberry Pi	RPi 2 model B+	900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, 4 USB, 40 GPIO pins, HDMI, Ethernet, CSI, DSI, Micro SD
9	Webcam	Raspberry Pi	RPi Camera module	5 MP, 1080p, SCI

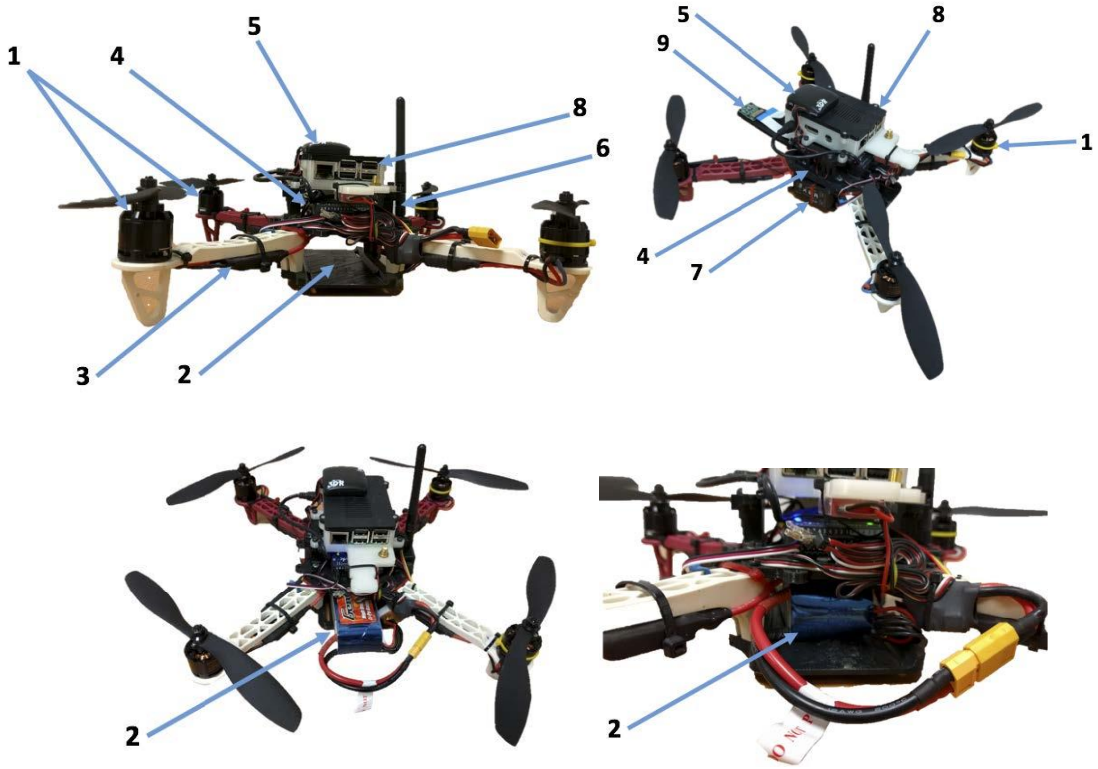


Figura 5: Ubicación de los componentes del mutirrotor GRCQuad.

3.1.3 Descripción del sistema de navegación

El sistema de telemetría 3DR 433MHz ofrece una conexión serie emulada mediante puerto USB, lo permite enlazar cualquier ordenador portátil o dispositivo móvil compatible con dicho dispositivo. Esto incluye a los sistemas operativos más comunes, incluyendo Windows, Linux y Android.

Para que el dispositivo móvil se convierta en un sistema de navegación, además de la interfaz de telemetría 3DR, debe tener instalada una aplicación de control de misión genérica. Actualmente, las más destacadas son APM Mission Planner 2 (para PC) y DroidPlanner 2 (para Android).

Estas aplicaciones ofrecen la funcionalidad básica de cualquier sistema de navegación, incluyendo:

- Datos de telemetría en tiempo real.
- Definición de rutas mediante introducción de *waypoints*.
- Selección de comandos de misión.
- Archivos de log de misión.

A continuación, se muestra la interfaz de cada una de estas aplicaciones.

APM Mission Planner 2

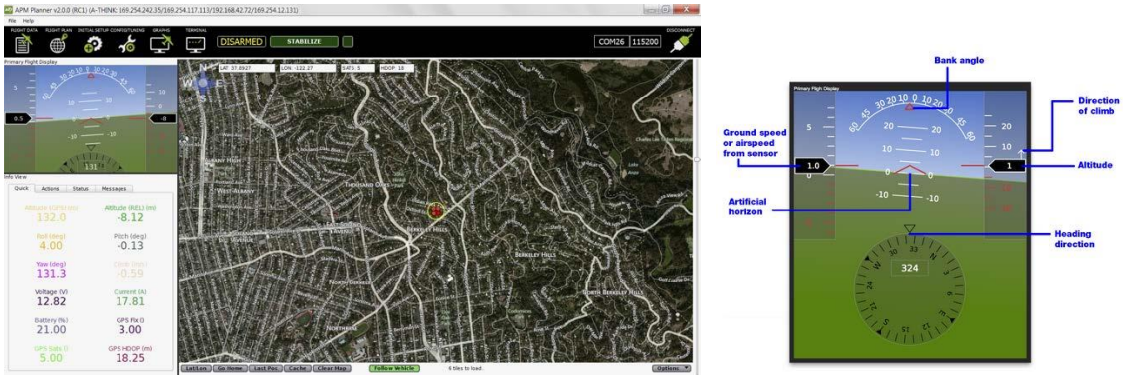


Figura 6: Interfaz gráfica de APM Mission Planner 2.

Características de APM Mission Planner 2:

- Introducción de *waypoints* mediante *point-and-click*, usando Google Maps/Bing/Open street maps o el servicio online de mapas de tu elección.
- Selección de comandos de misión mediante desplegables
- Descarga y análisis de los archivos de log de misión
- Configura los ajustes de APM para tu chasis
- Proporciona la interfaz con un simulador de vuelo de PC para crear un completo simulador UAV hardware-in-the-loop.
- Proporciona la salida del terminal serie de APM

DroidPlanner 2

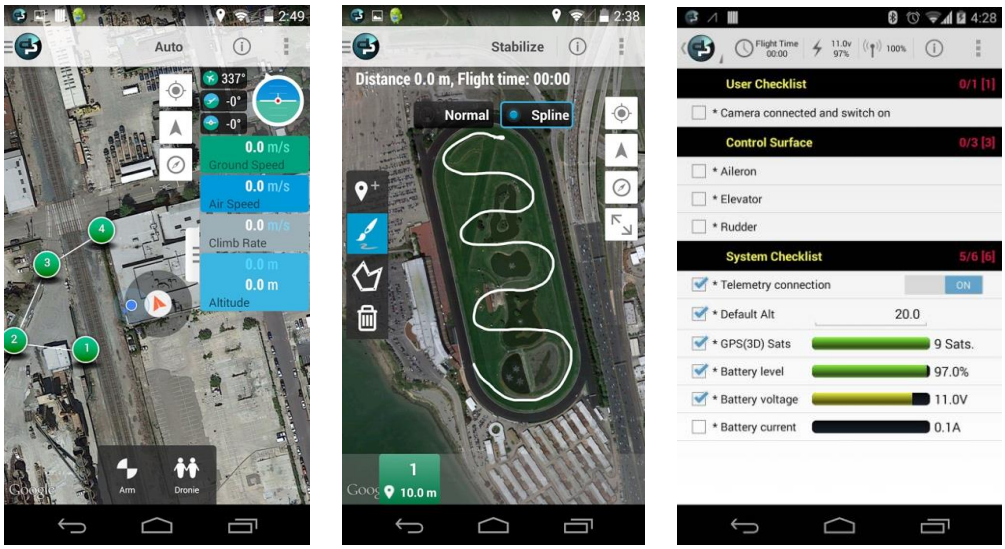


Figura 7: Interfaz gráfica de DroidPlanner 2.

Características de DroidPlanner 2:

- Especialmente diseñado para aeronaves 3DR e Iris
- Pantalla de telemetría con datos como HUD, batería, RSSI, distancia.
- Botones “Home”, “Land” y “Loiter” fáciles de usar.
- Nuevo modo de control con altitud variable.
- Cambio rápido entre modos.
- Nueva pantalla de planificación para generación rápida de misiones
- Herramientas para edición sencilla de misiones.
- Configuración básica de la TX radio.
- Lista de comprobación pre-vuelo.

3.1.4 Sistema de mando y control

Como equipo de radio control de la estación de tierra se utiliza un transmisor RC genérico que ha sido adaptado a la aeronave.

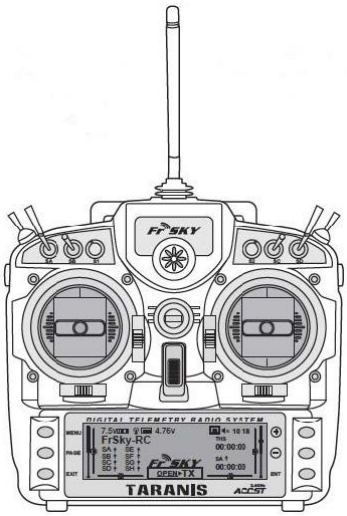


Figura 8: Mando de control remoto.

Especificaciones:

- Transmisor FrSky Taranis X9D Digital Telemetry Radio
- Frecuencia de transmisión: 2,4 GHz
- Número de canales: hasta 16
- Alimentación: 6 a 15V NiMh
- Corriente máxima: 260mA
- Temperatura de operación: -10~60°C
- Pantalla LCD retroiluminación: 212x64 píxeles monocromático
- Memorias: 60 (extensible mediante tarjeta SD)
- Modo de operación: Mode 2 (*Left hand throttle*)
- Alcance: entre 1,5 y 2 km

3.2 Software adicional

La aplicación Dronning se ha desarrollado en la plataforma Java y se ejecuta simultáneamente en las Raspberry Pi ubicadas en los drones y en un ordenador portátil que ejerce el rol de controlador de la prueba.

La realización de *debugging* requeriría mucho tiempo de realizarse directamente con los drones. Con objeto de reducir el tiempo necesario para las pruebas de desarrollo se instaló **Software In The Loop** (SITL) [6] en el ordenador portátil, un software capaz de emular el funcionamiento de un dron de características similares.

SITL consiste en una compilación en lenguaje C++ de ArduPilot, el software que se instala en la Pixhawk para controlar un dron. La instalación se puede realizar sobre Linux, Windows o sobre una máquina virtual, habiéndose escogido esta última opción para el presente trabajo. En el primer anexo se explica el proceso de instalación y manejo del simulador.

Wireshark [55] es un programa de análisis de protocolos de comunicaciones. Esta herramienta se utilizó también durante el proceso de *debugging* del software desarrollado para determinar si los paquetes de datos enviados y recibidos tenían el tamaño objetivo y si las direcciones y puertos empleados eran los correctos.

Microsoft Excel [27] es un software para el análisis matemático de datos. Se utilizó para analizar los datos tomados en campo, analíticamente y mediante gráficas. Para ello se desarrollaron macros en el lenguaje Visual Basic para Aplicaciones (VBA), tal y como se detalla en el apartado 7.

3.3 Protocolo MAVLink

El dispositivo Pixhawk que controla los drones se puede comunicar con el mando de control remoto y otros dispositivos mediante el protocolo MAVLink.

El protocolo MAVLink empaqueta datos de control y de información para la transmisión entre dispositivos. Está implementado a través de una librería muy ligera implementada en C++ y portada a otros lenguajes como Python y Java.

En la actualidad existen dos versiones, la 0.9 y la 1.0, aunque está previsto el lanzamiento de una versión 1.1 compatible con 1.0 que añadirá nuevos tipos de mensaje [41].

Además de los mensajes ya implementados en el estándar, el desarrollador puede definir sus propios mensajes en formato XML [37].

1B	1B	1B	1B	1B	1B	nB	1B
SOF	PLL	PS	SID	CID	MID	Payload	CRC

Figura 9: Estructura de un mensaje MAVLink v1.0.

…donde:

SOF: *Start of frame* (0xFE). Carácter especial necesario para identificar el comienzo del mensaje en la transmisión por el puerto serie.
PLL: *Payload length*. Longitud n de los datos incluidos en el mensaje.
PS: *Packet sequence*. Número de secuencia del mensaje desde el inicio de la comunicación entre la Pixhawk y cualquier otro dispositivo.
SID: *System ID*. Identificación del sistema emisor.
CID: *Component ID*. Identificación del componente del sistema que envía el mensaje.
MID: *Message ID*. Identificador del tipo de mensaje según el protocolo.
Payload: Datos incluidos en el mensaje.
CRC: *Checksum* del paquete para evitar errores de transmisión.

Como se observa en la figura 10, los mensajes MAVLink tienen longitud variable dependiendo del contenido de un mensaje.

Cuando se recibe un mensaje, el dispositivo correspondiente decodifica el *payload* considerando que el contenido se ajusta al mensaje del tipo especificado en el campo *Message ID*.

El *payload* del mensaje puede contener distintos tipos de datos, que requieren una longitud en bytes distinta para su almacenamiento. Para evitar problemas de alineación de los datos, éstos se reordenan antes de su codificación [39].

El CRC permite ignorar los mensajes con errores en la transmisión. En su cálculo, al final del mensaje se incluye un byte calculado como *checksum* del XML que define el tipo de mensaje, lo que evita que el origen y el destino utilicen distintas versiones del mismo tipo de mensaje. Esta mejora se introdujo en la versión 1.0 para impedir errores de interpretación cuando los dispositivos origen y destino utilizan distintas versiones del protocolo o de los mensajes.

Para analizar la información de telemetría proporcionada por el dron se ha desarrollado un programa en Java denominado Dronning. Mediante la compilación y uso de una librería, el programa transforma cada mensaje recibido en un objeto Java, lo que facilita considerablemente el acceso y la manipulación de la información.

La librería MAVLink [40] dispone de una descripción de cada tipo de mensaje en ficheros XML, aunque está concebida para su uso en el lenguaje de programación C. Mediante una utilidad adicional [14] se puede compilar una librería alternativa para su uso con Java, utilizando para ello los mensajes definidos en la librería diseñada para C.

El proceso de compilación de la librería se realiza en los siguientes pasos:

- Creación de un proyecto Eclipse con la utilidad ya indicada.
- Debido a un error de diseño de la utilidad, en el fichero “org.mavlink.library/pom.xml” es necesario englobar la etiqueta <*plugins*> en una nueva etiqueta denominada <*pluginManagement*>.

- Compilación de la utilidad.
- Copia de los ficheros “ardupilotmega.xml” y “common.xml” de la librería MAVLink en una carpeta y ejecución en la misma del programa compilado con los siguientes parámetros:

  java -jar generator.jar resources/v1.0/ target/ true true true true

  Los dos primeros parámetros indican las carpetas origen y destino de los ficheros xml y las librerías, respectivamente. Los cuatro booleanos indican que la codificación de los mensajes se realizará en *little endian*, que la compilación es para Embedded Java, que la versión del protocolo es la 1.0 y que se generarán los métodos *toString* en las clases Java para facilitar el *debugging*.
- Copia de las librerías generadas, org.mavlink.library.jar y org.mavlink.util-1.00.jar, al proyecto de trabajo.

3.4 Enlace entre Raspberry Pi 2 y Pixhawk

Para que el programa Dronning reciba información telemétrica de la Pixhawk [38] ha sido necesario crear un enlace entre Pixhawk y Raspberry Pi 2.

El puerto Telem1 de la Pixhawk envía telemetría en tiempo real al transmisor empleado para enlazar con el mando a distancia. El dispositivo dispone también de un puerto Telem2 adicional que se puede habilitar para transmitir la misma información. Para ello basta con conectar la Pixhawk al ordenador mediante USB y habilitar el envío de mensajes desde el programa APM Planner [35].

La Raspberry Pi 2 dispone de un puerto GPIO [44] (General Purpose Input/Output) de 40 pines en el que se puede habilitar un puerto serie al que conectar el puerto Telem2 de la Pixhawk.

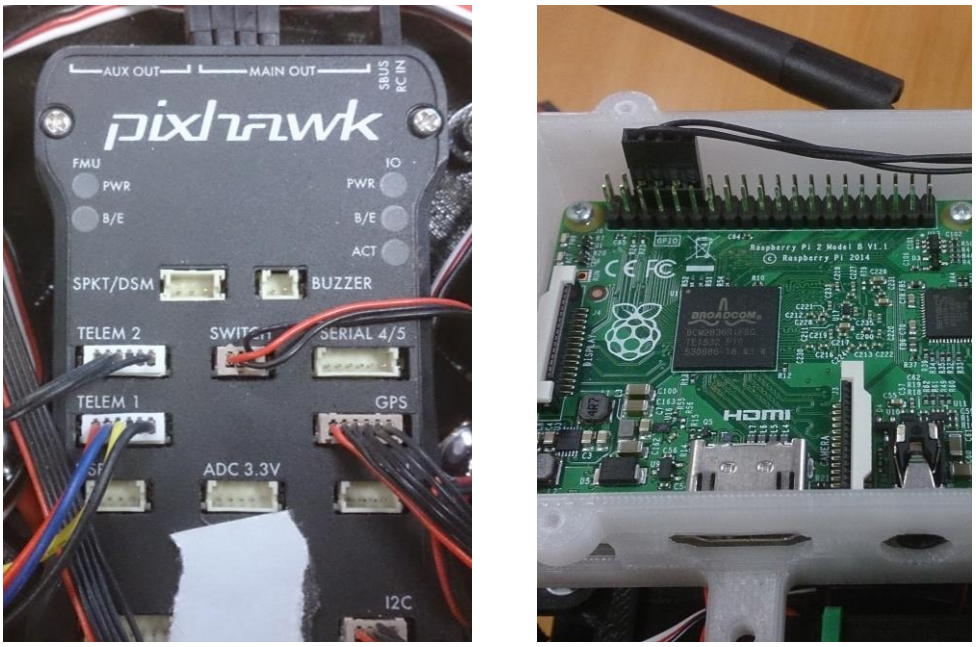


Figura 10: Puertos Telem2 y GPIO en Pixhawk y Raspberry Pi 2.

El esquema de conexión entre ambos dispositivos es sencillo [4]. Es importante tener en cuenta que, por defecto, la Raspberry Pi 2 envía información por el puerto serie, lo que podría provocar un comportamiento inadecuado de la Pixhawk. Para evitar este efecto indeseado hay que deshabilitar la salida de datos por el puerto serie mediante el siguiente comando y seleccionando la opción "*Advanced options*" en la utilidad que éste inicia:

```
sudo raspi-config
```

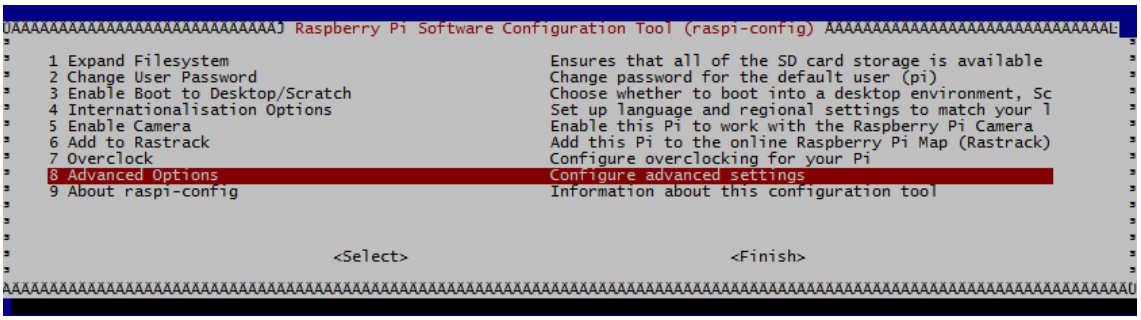


Figura 11: Utilidad raspi-config.

Una vez configurada la conexión entre ambos dispositivos es necesario emplear alguna librería para que el programa Dronning pueda leer la información desde el puerto serie.

Tras una búsqueda exhaustiva en Internet, ha sido posible localizar dos librerías para la comunicación por el puerto serie en Java, compatibles con la Raspberry Pi 2; PI4J [34] y RXTX [48].

Ambas librerías se testearon en programas de prueba, no apreciando ninguna diferencia en su funcionamiento. Se escogió la segunda opción por tener un proceso de instalación más sencillo, y por requerir incluir en el proyecto Eclipse un número de librerías mucho menor.

La instalación en la Raspberry Pi 2 requiere los siguientes pasos:

1. Inserción de la siguiente línea en el fichero "/etc/environment":
   ```
   JAVA_HOME="/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt"
   ```
2. Inserción de las siguientes líneas en el fichero "~/.bashrc":
   ```
   export JAVA_HOME="/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt"
   export PATH=$PATH:$JAVA_HOME/bin
   ```
3. Instalación de la versión 2.2pre1:
   ```
   sudo apt-get install librtx-java
   ```
4. Descarga del fichero "RXTXcomm.jar" desde la web oficial a la carpeta "/home/pi/javalibs/"
5. Creación de enlaces simbólicos al binario de la librería:
   ```
   mkdir /home/pi/lib
   ln -s /usr/lib/jni/librxtxSerial-2.2pre1.so
        /home/pi/lib/librxtxSerial.so
   ln -s /usr/lib/jni/librxtxSerial-2.2pre1.so /usr/lib/jvm/jdk-
        8-oracle-arm-vfp-hflt/jre/lib/arm/librxtxSerial.so
   ```
6. Inserción de la siguiente línea en el fichero "/etc/environment":
   ```
   CLASSPATH="/home/pi/javalibs/RXTXcomm.jar"
   ```
7. Inserción de las siguientes líneas en el fichero "~/.bashrc":
   ```
   export CLASSPATH=/home/pi/javalibs/ RXTXcomm.jar
   export LD_LIBRARY_PATH=/home/pi/lib/
   ```

Con los dos primeros pasos se registra la ubicación de Java y sus binarios [49] para que cualquier aplicación pueda usarlos. Como se observa, en la actual versión de Raspbian está instalada por defecto la versión 8.

Tras instalar la versión 2.2pre1 es necesario descargar una versión actualizada de la librería para Java. Tal y como se comenta en la página web de descarga [47], la versión de la librería Java y la versión de la librería nativa instalada en el dispositivo no coinciden, lo que genera por la consola un mensaje de advertencia cada vez que se utiliza RXTX, sin que ello suponga ningún riesgo ni produzca ningún tipo de error.

Con los pasos quinto y siguientes se hacen visibles la librería java y el binario que utiliza a todos los programas que se ejecuten en el entorno Java [58].

Se detectó un problema con la conexión serie entre Pixhawk y Raspberry Pi 2. Es necesario iniciar completamente la Pixhawk antes de conectar la Raspberry Pi 2 para que el enlace por el puerto serie se realice correctamente. Al parecer, si la Raspberry Pi 2 arranca y no detecta nada al otro lado de la conexión serie deshabilita el puerto correspondiente, impidiendo recibir ningún tipo de información del dron a partir de ese momento.

4 Creación de redes Ad-hoc: operación y requisitos

A medida que el uso de drones se generaliza, también se plantea la posibilidad de conseguir que trabajen en equipo mediante una red comunicaciones inalámbrica. Al estar en movimiento continuo, el uso de redes conectadas a un punto de acceso resulta totalmente inadecuado, pues los drones se alejarían con frecuencia del mismo hasta el punto de perder la conectividad. Por este motivo resulta especialmente interesante utilizar una red que no requiera infraestructura, y que se adapte dinámicamente a la disponibilidad de los dispositivos conectados a la misma.

Las redes Ad-hoc [53] son redes descentralizadas, sin punto de acceso, lo que las hace adecuadas para el contexto plateado. Se trata de redes donde la topología varía dinámicamente según la posición relativa de los nodos que integran la red. Para nuestros propósitos, se ha configurado una red Ad-hoc de la que forman parte los dos drones empleados en las pruebas y un ordenador portátil para controlar el proceso.

Con objeto de operar una red Ad-hoc, además de la configuración necesaria, hay que disponer de adaptadores WiFi capaces de funcionar en tal modo, ya que muchos adaptadores de bajo coste no disponen de esta característica. También se ha tenido en cuenta la conveniencia de utilizar adaptadores con antena externa, lo que mejora la recepción y permite observar la inclinación relativa de las antenas, factor que puede influir en la calidad de recepción de la señal. Por estos motivos se ha decidido utilizar el modelo TP-Link TL-WN722N, que funciona en la banda de los 2,4 GHz.

La configuración se ha realizado definiendo IPs estáticas, con un método diferente para los drones que para el ordenador portátil, dado que tienen sistemas operativos diferentes.

4.1 Configuración de la red Ad-hoc en los drones

Ambos drones funcionan con el sistema operativo Raspbian, una versión de Linux basada en la distribución Debian [43]. Dicho sistema ya tiene integrados los controladores necesarios para el adaptador WiFi escogido, por lo que tan solo es necesario configurar la red.

4.1.1 Configuración de la IP estática y del modo Ad-hoc

El adaptador WiFi tiene asociada la interfaz de red wlan0, por lo que la configuración a introducir en el fichero “/etc/network/interfaces” es la siguiente [2]:

```
auto wlan0
iface wlan0 inet static
      address 192.168.1.1
      netmask 255.255.255.0
      wireless-channel 4
      wireless-essid DRONNING
      wireless-mode ad-hoc
```

, donde la dirección IP es 192.168.1.1 para el dron servidor y 192.168.1.2 para el dron que ejerce el rol de cliente. A estos efectos, se ha denominado servidor al dron que envía paquetes de datos con el protocolo UDP, y cliente al dron que los recibe, detectando aquellos que se han perdido durante la transmisión. La IP 192.168.1.3 corresponde al ordenador portátil, y DRONNING es el nombre escogido para la red Ad-hoc.

4.1.2 Problemática encontrada

Aunque la configuración indicada debería ser suficiente, en el caso concreto del adaptador utilizado existe un problema adicional. Al arrancar la Raspberry Pi 2 y cargar los controladores del adaptador, ésta no es capaz de levantar correctamente el interfaz wlan0 en modo Ad-hoc (no hay ningún problema en modo infraestructura) y no le proporciona una IP.

Sin embargo, se ha comprobado que si se reinicia la red y se levanta de nuevo la interfaz funciona correctamente, es decir, el fallo se produce solamente la primera vez que se intenta poner el adaptador en modo Ad-hoc.

Para resolver el problema se modificó el fichero "~/.bashrc" incluyendo al final una orden que lanza el siguiente script [3] para realizar el proceso ya explicado cada vez que arranca el dispositivo:

```
iprango=192.168.1
ipbuena=`ifconfig wlan0 2>/dev/null|awk '/inet addr:/ {print $2}'|sed
     's/addr://'`
if [[ $ipbuena == $iprango* ]]
   then
        echo Tu IP ya era $ipbuena
   else
        sudo cp /etc/network/interface_ad-hoc_backup
             /etc/network/interfaces
        sudo systemctl daemon-reload
        sudo service networking restart
        sudo ifup wlan0
        sudo ifdown wlan0
        sudo ifup wlan0
        ipbuena=`ifconfig wlan0 2>/dev/null|awk '/inet addr:/ {print
             $2}'|sed 's/addr://'`
        if [[ $ipbuena == $iprango* ]]
          then
               echo Tu IP es $ipbuena
          else
               sudo ifdown wlan0
               sudo ifup wlan0
               ipbuena=`ifconfig wlan0 2>/dev/null|awk '/inet addr:/
                    {print $2}'|sed 's/addr://'`
               if [[ $ipbuena == $iprango* ]]
                    then
                        echo Tu IP en el segundo intento es $ipbuena
                    else
                        exit 1
               fi
        fi
fi
```

4.2 Configuración de la red Ad-hoc en ordenador portátil

El ordenador portátil funciona con el sistema operativo Windows 10. Una vez instalados los controladores proporcionados por el fabricante se intentó unir el ordenador a la red WiFi en modo Ad-hoc, pero en un primer momento no fue posible.

Microsoft ha reducido el soporte para redes Ad-hoc en los sistemas operativos Windows 8.1 y Windows 10 [13]. Las redes Ad-hoc no aparecen en la lista de redes disponibles, o aparecen pero no permiten conectarse a ellas.

La solución en el caso de Windows 10 consta de dos pasos:

1. Una sola vez, se crea manualmente el perfil de la red con el asistente disponible en el "Centro de redes y recursos compartidos", y después se ejecuta la siguiente orden en consola para cambiar el perfil de red de tipo infraestructura a tipo Ad-hoc:
   ```
   netsh wlan set profileparameter DRONNING connectiontype=ibss
        connectionmode=manual
   ```
2. Cada vez que se desea conectar a la red se ejecutan los siguientes comandos:
   ```
   netsh wlan connect DRONNING
   netsh interface ip set address name="Wi-Fi" static 192.168.1.3
        255.255.255.0 0.0.0.0
   ```

Cuando se desea desconectar de la red Ad-hoc se ejecutan los siguientes comandos:

```
netsh wlan disconnect
netsh interface ip set address name="Wi-Fi" source=dhcp
```

Como es lógico, los procesos de conexión y desconexión se han automatizado incluyendo las instrucciones ya descritas en sendos scripts.

Otro aspecto de compatibilidad digno de mención se refiere al hecho de que el ordenador con Windows 10 debe ser el último en entrar en la red Ad-hoc, pues se ha comprobado que cuando un dron entra en la red después que el ordenador, Windows no es capaz de localizarlo.

Además de los problemas de compatibilidad ya explicados, Windows 10 también tiene un error intermitente en el firewall, al menos cuando se utiliza este tipo de redes. El error impide a veces conectar con los drones. Dicho de otro modo, si el firewall de Windows 10 está activado, en algunas ocasiones no se puede realizar ni si quiera una petición de ping hacia los drones. Como consecuencia, antes de conectarse a la red Ad-hoc, se optó por desactivar temporalmente el firewall.

5 La aplicación Dronning

5.1 Descripción

Como ya se ha explicado en la introducción, en el presente proyecto se ha desarrollado una aplicación denominada Dronning para establecer comunicación entre dos drones con objeto de medir la calidad del enlace entre ambos para identificar y cuantificar los factores que más influyen en la misma.

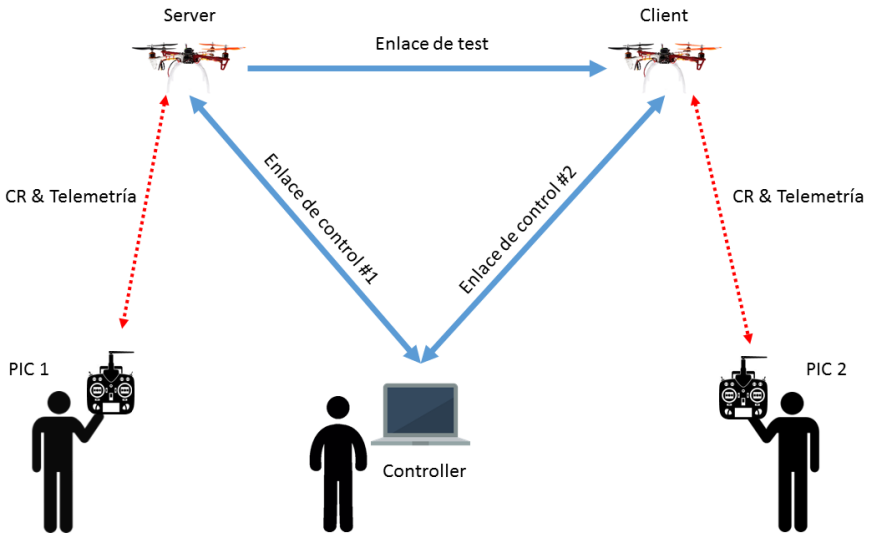


Figura 12: Esquema general de funcionamiento.

En la figura 13 se observa las condiciones en las que se utilizará el programa para analizar la calidad del enlace de test.

El programa se instala en el ordenador portátil y en las Raspberry Pi 2 acopladas a ambos drones, y que disponen de un adaptador WiFi cada una.

Un dron (*server*) ejerce el rol de servidor, enviando paquetes de datos de un determinado tamaño, durante un tiempo definido, y a una tasa constante mientras dura la prueba. El otro dron (*client*) recibe los paquetes de datos y los almacena identificándolos, junto a información de ambos drones. La información recogida permitirá determinar cuáles de los parámetros influye en la pérdida de paquetes. El proceso es controlado desde un ordenador portátil (*controller*).

El mecanismo de control de congestión del protocolo TCP reenvía un paquete cada vez que se detecta su pérdida, y además afecta a la velocidad de envío, lo que hace de TCP un protocolo inadecuado para realizar pruebas de comunicación a una tasa de envío constante. Por este motivo se ha utilizado UDP, un protocolo sin control de congestión ni control de flujo.

El programa diseñado permite transmitir los paquetes de datos por enlace de test tanto en modo *unicast* como *broadcast* [8]; este último modo permitiría realizar la prueba con más

drones en simultáneo en trabajos posteriores. Como aspecto negativo, el uso de este tipo de paquetes limita el ancho de banda a 1Mbps [56] o a 6 Mbps, dependiendo de la variante IEEE 802.11 utilizada, aunque se considera que dicha limitación no afecta al tipo de experimentación que se desea realizar. Además, es realista suponer que en un enjambre de drones habrá mensajes de tipo *broadcast* para su coordinación, por lo que se trata de un modelo de comunicación realista.

Se verificó que la dirección 255.255.255.255 no permite realizar envíos *broadcast* en el sistema operativo Raspbian, siendo necesario utilizar la dirección broadcast de la red, en este caso 192.168.1.255 (red 192.168.1.0/24).

Para el desarrollo de la aplicación se ha elegido el lenguaje de programación Java porque se trata de un lenguaje que genera aplicaciones multiplataforma, lo que facilita la ejecución del programa ya sea en el ordenador portátil bajo el sistema operativo Windows 10, o en los drones, en los cuales se ejecuta el sistema operativo Raspbian. A este respecto hay que hacer una matización, el acceso al puerto serie para recabar información de la Pixhawk se realiza con una librería nativa para Linux, aunque esto no afecta a la ejecución del programa en el ordenador personal, pues en este caso no se accede a ningún puerto serie.

5.2 Estructura interna de la aplicación

Dado que los tres roles de la aplicación tienen que comunicarse entre sí de forma asíncrona, resulta obvio que el planteamiento de la aplicación debe ir en la misma línea. Cuando un terminal con un rol se comunica con un terminal con otro rol, puede estar simultáneamente esperando información de otro terminal con rol distinto (ej/ *controller*), por lo que es necesario separar ambas tareas en sendos hilos de ejecución, lo que los independiza evitando así que el envío de un mensaje tenga que esperar a la recepción de otro mensaje, y viceversa.

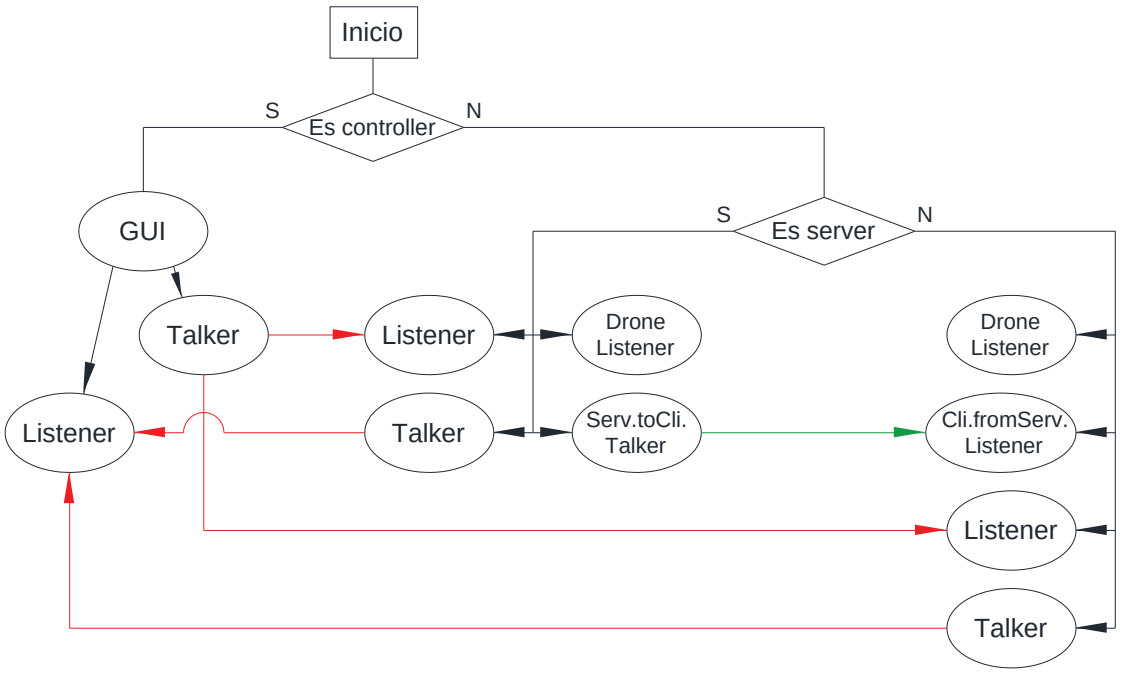


Figura 13: Estructura interna de la aplicación.

En la figura anterior se ilustran dos conceptos. Por un lado, enlazando con líneas de color negro se muestra el flujo de ejecución de la aplicación cuando se inicia, con la creación de los distintos hilos representados dentro de elipses. Por el otro se muestra en líneas de color rojo y verde la comunicación que se produce a través de los adaptadores WiFi entre los tres dispositivos conectados en la red.

En caso de que el programa ejerza el rol *controller*, el hilo principal lanza la Interfaz Gráfica de Usuario (GUI) y crea otros dos hilos, *Listener* y *Talker*, que se encargarán de escuchar los mensajes recibidos de los drones y de enviar mensajes de control y de respuesta, respectivamente.

Si el programa ejerce el rol *server*, el hilo principal lanza cuatro nuevos hilos, dos para la comunicación con el *controller*, uno para leer los datos proporcionados por la Pixhawk a través del puerto serie, y un último hilo para enviar datos al *client* mientras se realiza una toma de datos.

Por último, en cualquier otro caso, el programa ejerce el rol *client*. La estructura en hilos de ejecución es similar a la del *server*, aunque en este caso en vez de enviar únicamente recibe datos por el canal dedicado a la toma de datos.

Cabe destacar que, mientras se realiza una toma de datos, no se envía ningún mensaje por los canales de comunicación marcados con líneas rojas para no ocupar el medio, y que mientras se realizan las tareas de coordinación entre los tres dispositivos tampoco se envían datos por el canal dedicado a la toma de datos, representado con una línea de color verde.

5.3 Protocolo de comunicación entre dispositivos

Cuando se inicia una toma de datos hay tres instancias de la aplicación en ejecución en distintos dispositivos, más concretamente en ejecución asíncrona. Por este motivo es necesario establecer un protocolo de comunicación entre los tres dispositivos para coordinar las tareas que realizan los distintos hilos en ejecución.

El funcionamiento deseado del programa consta de los siguientes pasos:

1. Conexión de *client* a *controller*
2. Conexión de *server* a *controller*
3. Conexión de *server* a *client**
4. Orden de configuración de *controller* a *client*
5. Orden de configuración de *controller* a *server*
6. Orden de iniciar la prueba de *controller* a *client*
7. Orden de iniciar la prueba de *controller* a *server*
8. Aviso de finalización de la prueba de *client* a *controller*
9. Orden de finalizar la prueba de *controller* a *server*

* En realidad, esta operación solamente sería necesaria en caso de enviar paquetes en modo *unicast*.

Los pasos 1 a 3 se realizan automáticamente cuando se inician los dispositivos. Como se observa, el proceso no da comienzo hasta que el *client* intenta conectar al *controller*. El resto de pasos se repiten cada vez que se realiza una toma de datos. Una vez completado el tercer paso,

en la Interfaz Gráfica de Usuario (IGU) se actualiza información sobre la posición relativa de los drones y sobre su guiñada, excepto mientras se realiza la prueba, periodo en el cual no se envía ese tipo de información por la red para no afectar a los resultados de la misma.

En el diseño propuesto para las comunicaciones se observa que el control del proceso va cambiando entre el *client* y el *controller*. Al principio el control lo tiene el *client*, una vez establecidos los canales de comunicación pasa al *controller* y por último vuelve al *client*, que detiene la prueba una vez transcurrido el periodo de tiempo configurado para la misma. Tras detenerse la toma de datos el *controller* asume de nuevo el control para configurar la siguiente toma.

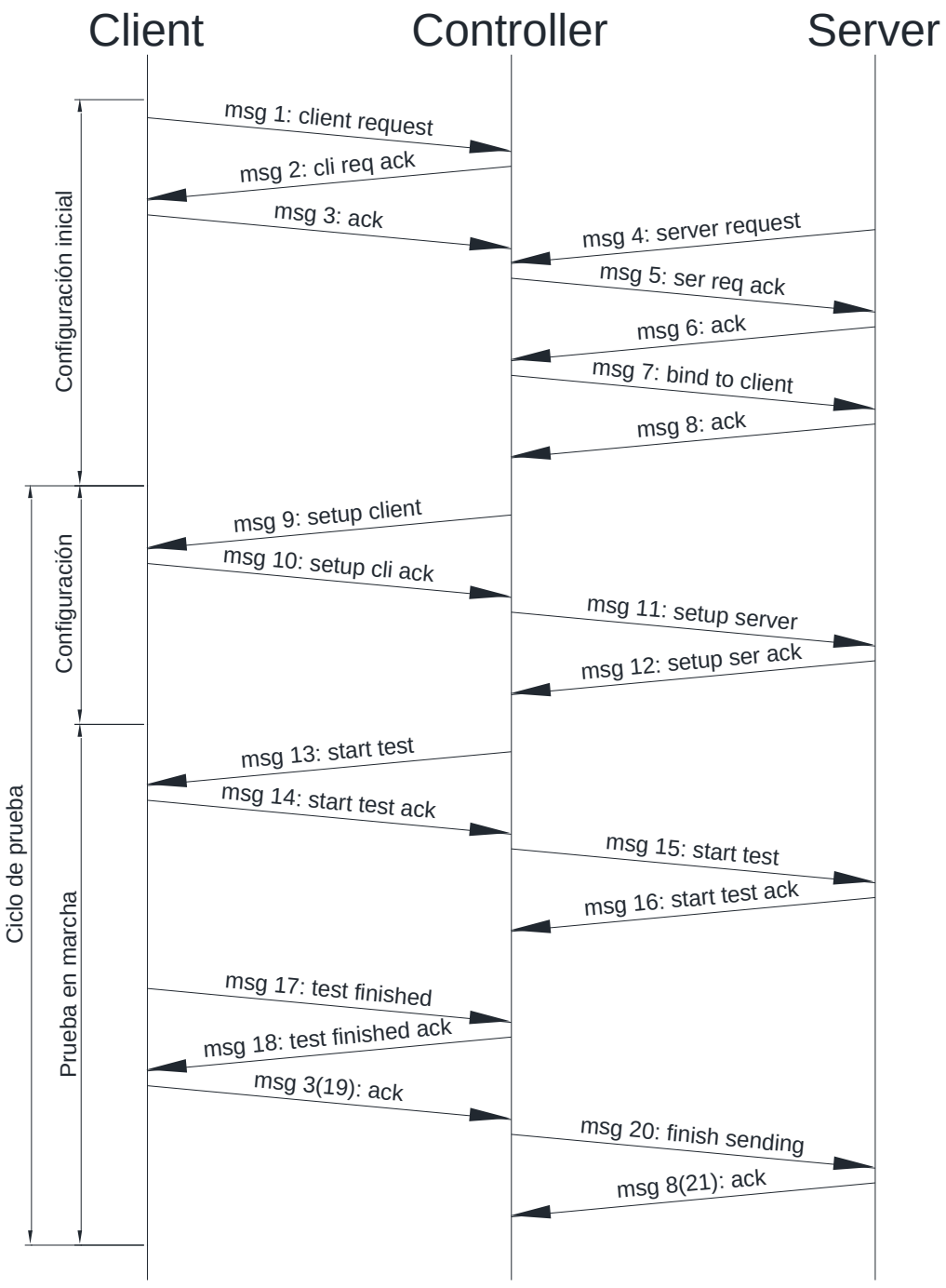


Figura 14: Protocolo de comunicación entre instancias de la aplicación.

El protocolo funciona fijando únicamente la IP del *controller* y los puertos de comunicación. De este modo, si en un futuro se amplían las pruebas con más drones, el programa ya estaría preparado para utilizar otras IPs en los clientes. Para ello, los drones se conectan a la IP conocida del *controller*, que toma nota de la IP de origen y, si procede, la proporciona al *server* para que conecte al *client* (en el caso de envíos *unicast*).

Dado que UDP es un protocolo con pérdida de paquetes, fue necesario hacer un protocolo robusto, capaz de funcionar correctamente incluso tras la pérdida de cualquier paquete de datos. La solución pertinente es el uso de paquetes de reconocimiento (ACK), la definición de estados de la aplicación, y la retransmisión de un paquete hasta que se recibe la confirmación de su recepción. La aplicación no debe cambiar de un estado al siguiente hasta haber recibido el paquete de datos que provoca tal evento. Como el estado afecta simultáneamente a varios hilos en ejecución asíncrona, la variable que determina el estado en que se encuentra la aplicación debe ser de tipo atómico, lo que evita posibles condiciones de carrera en el acceso concurrente por parte de varios hilos.

Los mensajes 3 y 8 sirven también para informar al *controller* sobre el resultado de la toma de datos anterior, luego tienen una doble funcionalidad, como ACK (mensajes tipo 3 y 8) y para proporcionar el resultado de la toma de datos (lo que serían mensajes tipo 19 y 21). Los mensajes 3, 8, 10 y 12 envían información en tiempo real sobre la posición y guiñada de los drones, información que se muestra en la interfaz gráfica en el periodo comprendido entre dos tomas de datos para poder ubicar correctamente los drones para la siguiente toma.

Los mensajes 9 y 11 contienen los parámetros de configuración que rigen la prueba que se desea efectuar. En caso de que entre dichos parámetros se indique una duración de la prueba con valor negativo, las tres instancias del programa determinan que se quiere detener los experimentos y proceden al cierre del programa. Las instancias que se ejecutan en los drones también realizan un cierre controlado del sistema.

En base a la figura anterior y a la variable atómica que regula el progreso del programa, se definen varios estados para cada instancia del programa. En las siguientes tablas se refleja también el mensaje de control que se está enviando mientras el programa permanece en cada estado.

Tabla 3: Estados de la instancia *controller.*

Estado	Mensaje siendo enviado	Mensaje a recibir	Valor de la variable de estado	
			Previo	Posterior
1	-*	msg 1: cli. req.	1 (start)	2 (client logging)
2	msg 2: cli. req. ack	msg 3: cli. req. ack	2 (client logging)	3 (client logged)
3	-*	msg 4: serv. req.	3 (client logged)	4 (server logging)
4	msg 5: serv. req. ack	msg 6: serv. req. ack	4 (server logging)	5 (server logged)
5	msg 7: bind to client	msg 8: bind ack	5 (server logged)	6 (server connected to client)
6	msg 9: setup cli.**	msg 10: setup cli. ack	6 (server connected to client)	7 (client configured)
7	msg 11: setup serv.	msg 12: setup serv. ack	7 (client configured)	8 (server configured)
8	msg 13: start cli.**	msg 14: start cli. ack	8 (server configured)	9 (client running)
9	msg 15: start serv.	msg 16: start serv. ack	9 (client running)	10 (server running)
10	-*	msg 17: cli. finished	10 (server running)	11 (client stopping)
11	msg 18: cli. finished ack	msg 3(19): ack	11 (client stopping)	12 (client stopped)
12	msg 20: serv. finish	msg 8(21): serv. finished ack	12 (client stopped)	13 (server finished)

* Bucle de espera a recibir el mensaje indicado en la tercera columna.
** El mensaje no se envía hasta que el usuario interaccione con la IGU ordenando que el proceso continúe.

La aplicación permanece en un estado enviando el mensaje indicado en la segunda columna hasta que recibe el mensaje indicado en la tercera columna. Los mensajes se repiten periódicamente por si no llegan a destino, lo que significa que también podrían llegar varias veces al destino, incluso después de cambiar de estado, situación en que serán descartados.

El cambio del estado 11 al 12 y del 12 al 6 se produce automáticamente en cuanto se recibe el primer mensaje de tipo 3 y 8, respectivamente.

Tabla 4: Estados de la instancia *client.*

Estado	Mensaje siendo enviado	Mensaje a recibir	Valor de la variable de estado	
			Previo	Posterior
1	msg 1: cli. req.	msg 2: cli. req. ack	1 (start)	2 (logging)
2	msg 3: cli. req. ack	msg 9: setup cli.	2 (logging)	3 (configured)
3	msg 10: setup cli. ack	msg 13: start cli.	3 (configured)	4 (running)
4	msg 14: start cli. ack**	-*	4 (running)	5 (stopping)
5	msg 17: cli. finished	msg 18: cli. finished ack	5 (stopping)	6 (stopped)

* Bucle de espera a recibir el mensaje indicado.
** El mensaje se envía durante un periodo de tiempo reducido para evitar que afecte a la prueba (el estado 4 corresponde a la ejecución de la misma).

Al completar estado 5 se pasa al estado 2

Tabla 5: Estados de la instancia *server.*

Estado	Mensaje siendo enviado	Mensaje a recibir	Valor de la variable de estado	
			Previo	Posterior
1	msg 4: serv. req.	msg 5: serv. req. ack	1 (start)	2 (logging)
2	msg 6: serv. req. ack	msg 7: bind to client	2 (logging)	3 (connected)
3	msg 8: bind ack	msg 11: setup serv.	3 (connected)	4 (configured)
4	msg 12: setup serv. ack	msg 15: start serv.	4 (configured)	5 (running)
5	msg 16: start test ack**	msg 20: serv. finish	5 (running)	6 (stopped)

* Bucle de espera a recibir el mensaje indicado.
** El mensaje se envía durante un periodo de tiempo reducido para evitar que afecte a la prueba (el estado 5 corresponde a la ejecución de la misma).

Al completar el estado 5 se pasa al estado 3

5.4 Comunicación entre hilos de ejecución

Ya se ha explicado con detalle la solución implementada para la comunicación entre los hilos de ejecución de distintas instancias del programa. En este apartado se explica el mecanismo utilizado para la comunicación entre los hilos de una misma instancia.

La ejecución asíncrona de los hilos requiere coordinar su funcionamiento. Además de la variable atómica empleada para definir los estados de la aplicación, se han utilizado otras cuatro variables para organizar el orden de ejecución de los hilos.

También es necesario utilizar otras variables atómicas para almacenar información consultada y/o escrita por distintos hilos. Para ello se ha creado una clase denominada "DatosDron" cuyos objetos contienen los atributos: temperatura, alabeo, cabeceo, guiñada, velocidad con que varían los tres ángulos indicados, latitud, longitud, altitud, altitud relativa a la cota con que comenzó la toma de datos, velocidad en los tres ejes del espacio, y ángulo de orientación respecto al norte.

La Pixhawk proporciona al programa datos de tipo int, long y float. Java dispone de variables atómicas en los dos primeros casos, pero no en el último. Por lo tanto, ha sido necesario crear un tipo de dato atómico para números float [24].

En el caso del *controller*, el hilo *Listener* (ver figura 14) recibe estos datos de ambos drones mediante los mensajes del tipo 3, 8, 10 y 12 para mostrar la información por pantalla.

El *server* también envía esta información al *client*, que los recibe en su hilo *Cli.fromServ Listener* (ver figura 14). El mismo hilo combina los datos obtenidos del propio dron mediante el hilo *Drone listener*, con los recibidos y los almacena en un fichero en formato CSV para su posterior análisis.

En un orden menos relevante cabe mencionar que también se han utilizado variables atómicas para actualizar los datos que se muestran en la GUI del *controller*, dado que los datos los recibe en un hilo distinto del que actualiza la interfaz y podrían producirse condiciones de carrera.

5.5 Funcionamiento interno de los hilos de ejecución

En el presente apartado se comenta someramente el funcionamiento interno de los hilos de ejecución reflejados en la figura 14.

Teniendo en cuenta que la potencia de las Raspberry Pi 2 es limitada, y con objeto que el tiempo de procesamiento no influya en las pruebas realizadas con la aplicación, se realizaron test de velocidad de transferencia de un objeto tipo "DatosDron", el mismo que se transmite de un dron al otro, mientras se realiza una prueba. El test se realizó en el interfaz de *loopback* del mismo ordenador empleado como *controller*:

Tabla 6: Comparativa de tasas de envío.

Alternativa:	**Tasa máxima (paquetes/segundo)**
Serialización manual de los atributos	13.900
Serialización del objeto Java	23.500
Serialización del objeto mediante Kryo	29.500
Escritura directa de los datos en un buffer Kryo	48.300

Las tres primeras alternativas utilizan un ObjectOutputStream y un ObjectInputStream para enviar y recibir los datos, mientras que la última utiliza un buffer directamente.

La primera alternativa consiste en aprovechar que los atributos del objeto son serializables para introducirlos en el ObjectStream uno a uno. La segunda consiste en hacer el objeto del tipo "DatosDron" serializable y aplicar el mismo procedimiento, pero con un solo objeto. En la tercera alternativa se utiliza una librería externa denominada Kryo [10] que es mucho más eficiente serializando objetos que Java [18]. Ésta envía un identificador de la clase del objeto, definido previamente, en vez de transmitir la definición de la clase junto al mismo, lo que reduce el tamaño de los datos enviados y elimina el tiempo necesario para serializar la estructura de la clase.

La última alternativa consiste en reutilizar un buffer vaciando el contenido y rellenándolo cada vez que se desea enviar un paquete. Los objetos Input y Output de la librería Kryo se limitan a gestionar la lectura y escritura de un buffer de forma eficiente, lo que los hace especialmente interesantes al utilizar esta opción. La implementación se ha realizado de esta forma por ser la más eficiente, consumiendo por ello menos recursos del sistema.

Además del buffer de escritura o lectura, también se reutilizan los objetos Input, Output y DatagramPacket entre envíos, lo que mejora la eficiencia al no tener que crear los objetos cada vez, y evita basura que tendría que recoger el *Garbage Collector*.

Dado que la opción elegida resulta ser la más eficiente, se decidió utilizar también dicho método de empaquetado en los paquetes de control que se envían entre *controller*, *server* y *client*.

Otro aspecto a considerar es la tasa de envío de paquetes.

Si se establece un bucle sin más, los paquetes de datos se envían a la máxima velocidad que es capaz de proporcionar el sistema y la tarjeta adaptadora. Si se pretende enviar los datos

a intervalos regulares es necesario forzar al hilo de ejecución a una espera de tal modo que, sumada al tiempo de ejecución del código, implique un envío con el periodo buscado. La elección obvia para alcanzar este objetivo es utilizar el método java.lang.Thread.sleep(). Tan solo bastaría calcular el tiempo de espera entre dos paquetes para llamar a la función.

Sin embargo, en la realidad la solución planteada no funciona correctamente. Al no tratarse de un sistema en tiempo real, la granularidad en la definición del tiempo de espera está limitada por la duración del periodo de interrupción del programador del procesador [17], es más, la implementación Java del método que considera nanosegundos se limita a redondear al milisegundo más próximo y llamar a la función equivalente en milisegundos [46], lo que no mejora la solución.

Por otro lado, aunque proporcionaría la precisión requerida, queda totalmente descartada la espera activa por el desmedido consumo de CPU y energía que supondría. Esta solución consistiría en utilizar el método System.nanotime() para obtener el tiempo actual y comprobar si se rebasa el tiempo correspondiente al periodo de espera deseado. El método indicado proporcionaría una precisión suficiente para la frecuencia de envío de paquetes que se pretende alcanzar durante las pruebas.

La tasa de envío de paquetes de control entre los tres dispositivos se ha controlado mediante el método java.util.concurrent.TimeUnit.NANOSECONDS.sleep(nanosegundos), que a su vez llama a java.lang.Thread.sleep(), ya que el intervalo de envío de paquetes nunca es menor que 200 ms.

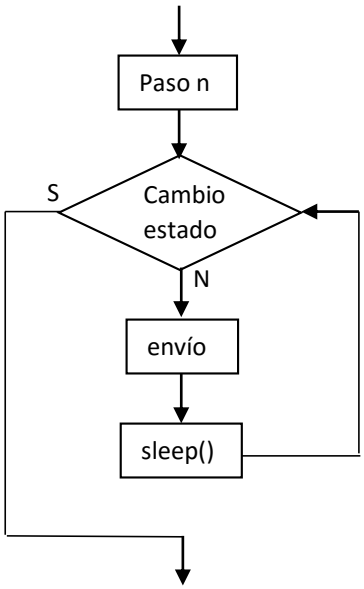


Figura 15: Lógica de los hilos de ejecución tipo *Talker*.

Como se observa en la figura, una vez se alcanza el estado durante el cual se tiene que enviar el mensaje en concreto, y mientras la instancia en ejecución no cambie de estado, se envía un paquete de datos y el hilo entra en suspensión hasta que sea necesario enviar un nuevo paquete. La decisión de detener el envío viene condicionada por el estado de una variable atómica compartida que modifican otros hilos.

Sin embargo, para el envío de paquetes durante una prueba, se requiere mayor granularidad en el tiempo de envío, y por eso se ha planteado una solución que implica tiempos

de espera nunca menores de 100 ms. Esta solución se explica más adelante en el apartado 5.5.4 por tratarse de una situación especial en el hilo de ejecución “Server to Client Talker”.

A continuación, se comentan algunas particularidades de varios hilos.

5.5.1 Hilo GUI

El hilo que construye la interfaz gráfica del *controller* es también el hilo principal del programa. Además de mantener la GUI, se encarga de crear los hilos de ejecución según se observa en la figura 14, y luego espera a que terminen la ejecución.

5.5.2 Hilos tipo Listener

Son los hilos de ejecución que reciben paquetes UDP de otros dispositivos.

Estos hilos podrían no terminar su ejecución nunca si se utilizase una espera bloqueante para la recepción de datos. Por este motivo se ha configurado los sockets con un *timeout* que permite comprobar periódicamente si la duración de la prueba es excesiva:

```
java.net.DatagramSocket.setSoTimeout(tiempoDeEspera)
```

El rol *controller* utiliza también este hilo para actualizar la interfaz gráfica con los datos recibidos de las Raspberry Pi 2. Dado que Swing no es *thread-safe* [19], hay que utilizar el código incluido a continuación para poder actualizar la interfaz gráfica sin riesgo desde un hilo distinto del que la mantiene:

```
javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
            //Código a ejecutar en el hilo de la GUI
      }
});
```

Entre otros datos, hay que actualizar la distancia entre los dos drones en metros, aunque las Pixhawk proporcionan la posición de los drones en coordenadas geográficas. La distancia entre los drones en metros se ha obtenido a partir de las coordenadas UTM, por lo que ha sido necesario construir métodos auxiliares para transformar latitud y longitud en coordenadas X e Y [50].

5.5.3 Hilos tipo Talker

En estos hilos se comprueba periódicamente una variable compartida para detectar si el tiempo transcurrido supera el esperable para la toma de datos, del mismo modo que en el caso anterior.

Los datos que se envían se escriben en un buffer Kryo, tal y como se ha comentado al principio del apartado 5.5.

5.5.4 Hilo Server to Client Talker

En el apartado 5.5 también se ha descrito el uso de la espera pasiva cada vez que se envía un paquete de control entre los tres dispositivos implicados en la comunicación. Sin embargo, el envío de paquetes de datos durante una prueba debe realizarse a una tasa constante y definida, lo que requiere una precisión en la espera entre dos paquetes consecutivos que no puede proporcionar una Raspberry Pi 2 con el método Thread.sleep(). En la siguiente figura se observa la solución empleada.

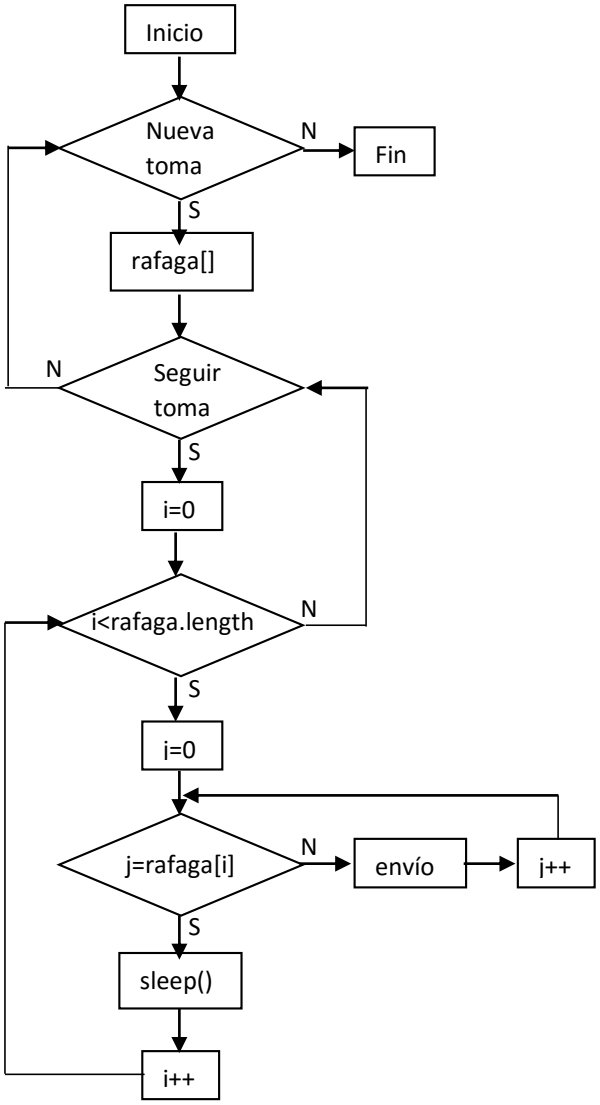


Figura 16: Lógica del hilo de ejecución *Server to Client Talker.*

Se decidió que el tiempo mínimo de espera empleado en el método sleep() sería de 100 ms para evitar el problema detectado relativo a la limitación de granularidad ya explicada. Por lo tanto, los datos se envían en como máximo 10 ráfagas por segundo.

En base a las hipótesis ya definidas, se crea un array “rafaga” de tamaño el número de ráfagas a enviar y valor el número de paquetes a enviar en cada ráfaga. El reparto de los paquetes a enviar en cada ráfaga es equitativo.

En la siguiente tabla se muestran ejemplos de cómo se repartirían los paquetes según la tasa de envío que se emplee en cada prueba.

Tabla 7: Ejemplo de reparto de paquetes en ráfagas.

Tasa (paquetes/segundo)	**Número de ráfaga dentro del mismo segundo**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
7	1	1	1	1	1	1	1	-	-	-
12	2	2	1	1	1	1	1	1	1	1
25	3	3	3	3	3	2	2	2	2	2
50	5	5	5	5	5	5	5	5	5	5
321	33	32	32	32	32	32	32	32	32	32

Mientras deba continuar la toma de datos los paquetes se envían en ráfagas, reduciendo así el número de veces que se emplea el método sleep() y aumentando el tiempo de espera que se le pasa como parámetro.

Se considera que el tamaño del paquete de datos puede influir en la probabilidad de que se pierda al ser enviado de un dron al otro. Por lo tanto, uno de los parámetros que se fijan al realizar la prueba es el tamaño de paquete.

Con la ayuda del software Wireshark se monitorizaron pruebas realizadas en el interfaz de *loopback* del ordenador con objeto de determinar cuántos bytes ocupan los datos transmitidos entre los drones. Se verificó que esta cantidad es de 86 bytes, la cual coincide con la suma de los bytes necesarios teniendo en cuenta el tipo de cada uno de los datos enviados. Teniendo en cuenta esta cifra y el tamaño estándar de la trama Ethernet, se impuso en la interfaz gráfica que solamente se puedan realizar pruebas con tamaño de trama entre 100 y 1500 bytes.

Para enviar un paquete de tamaño superior al ocupado por los datos basta con indicar la longitud del buffer que se desea enviar cuando se reasignan los datos al datagrama con el siguiente método:

```
java.net.DatagramPacket.setData(buffer, 0, longEnvio)
```

El *padding* enviado se ignora en destino, pues se leen únicamente los primeros bytes del datagrama recibido.

5.5.5 Hilo Client from Server Listener

Este es el hilo encargado de registrar en un fichero CSV información sobre el estado de los drones.

La Pixhawk envía información a las Raspberry Pi 2 cada 500 o 250 ms, según tipo de mensaje. Para realizar la prueba podría especificarse una tasa de envío muy elevada, por lo que al guardar la información de todos los paquetes de datos se guardaría información repetida, utilizando recursos de CPU y E/S innecesariamente.

Sin embargo, hay que registrar los datos al menos al doble de la frecuencia con que se actualiza la información para garantizar que no se pierde ninguna información, según el teorema

de Nyquist-Shannon [54]. Dado que una muestra cada 250 ms supone 4 muestras por segundo, se decidió tomar 10 muestras por segundo, sea cual sea la tasa de envío programada.

5.5.6 Hilos tipo Drone Listener

La instancia de cada uno de los drones dispone de un hilo para registrar la información que la Pixhawk proporciona a través del puerto serie de la Raspberry Pi 2.

La información se recibe en formato MAVLink, se interpreta mediante la librería RXTX, ya explicada en el apartado 3.3, y se almacena en un objeto tipo "DatosDron", construido a base de variables atómicas para que puedan ser accedidas simultáneamente por otros hilos.

En concreto, el hilo está a la escucha de los siguientes tipos de mensajes MAVLink:

- GLOBAL_POSITION_INT. Contiene información sobre posición y velocidad.
- ATTITUDE. Registra los ángulos que definen la posición del dron.
- SCALED_PRESSURE. Contiene la temperatura ambiente, entre otros.

5.6 Información transmitida

Cada vez que se envía un paquete durante una toma de datos, el emisor (*server*) incluye en el mismo la información recopilada sobre su estado, como se comenta en el apartado anterior. Como se explica en el apartado 5.5, los datos se escriben uno a uno en un buffer y luego se asigna una parte del mismo con la longitud que se desea alcanzar al datagrama que se envía. Los datos se escriben en el siguiente orden:

1 short	1 long	1 int	6 float	4 long	3 int	1 int
id	numSeq	temp	attitude	position	speed	heading

…donde:

id. Identificador del dron que transmite el mensaje
numSeq. Número de secuencia del mensaje. Necesario para identificar paquetes perdidos
temp. Temperatura ambiente
attitude. Posición del dron. Incluye guiñada, alabeo, cabeceo y velocidad con que varían dichos ángulos
position. Coordenadas geográficas, altitud y altitud respecto al suelo
speed. Velocidad respecto a los tres ejes cartesianos del dron
heading. Guiñada u orientación respecto al norte geográfico

A continuación, se detalla la información transmitida en cada uno de los mensajes definidos en el protocolo de comunicaciones descrito en el apartado 5.3. Se detallan solamente los mensajes que contienen datos adicionales a la mera identificación del mismo, representada por el campo *type*.

msg 1:

type	port

port. Puerto en el que el *client* escuchará los mensajes de control del *controller*

msg 3:

type	latitude	longitude	altitude	yaw

latitude. Latitud del *client*
longitude. Longitud del *client*
yaw. Guiñada del *client*

msg 4:

type	port

port. Puerto en el que el *server* escuchará los mensajes de control del *controller*

msg 7:

type	client IP	client port

client IP. IP a la que el *server* enviará paquetes durante la prueba en caso de realizarse por *unicast*
client port. Puerto al que en *server* enviará paquetes durante la prueba

msg 8:

type	latitude	longitude	altitude	yaw	real sent ratio

latitude, *longitude*, *altitude*, *yaw*. Ya descritos
real sent ratio. Tasa a la que se han enviado realmente los paquetes durante la toma de datos

msg 9:

type	filename	duration

filename. Nombre base para el fichero donde se guardará información
duration. Duración que tendrá la prueba en segundos

msg 10:

type	latitude	longitude	altitude	yaw

parámetros ya descritos

msg 11:

type	filename	duration	ratio	broadcast	packet size

filename, *duration*. Ya descritos
ratio. Tasa a la que el *server* debe enviar los paquetes durante la toma de datos
broadcast. Determina si los paquetes se enviarán por *unicast* o *broadcast*
packet size. Tamaño en bytes que deberán tener los paquetes transmitidos

msg 12:

type	latitude	longitude	altitude	yaw

Parámetros ya descritos

msg 17:

type	loss ratio	total lost packets

loss ratio. Porcentaje de paquetes perdidos durante la toma de datos
total lost packets. Número total de paquetes perdidos durante la toma de datos

5.7 Interfaz gráfica

La instancia del programa ejecutada con el rol *controller* es la única que dispone de interfaz gráfica, ya que las otras dos instancias se lanzan en los drones mientras están volando.

Dado que el software desarrollado no está orientado a la venta se decidió centrar el diseño de la interfaz en la sencillez y la funcionalidad, dando menor importancia a la parte estética.

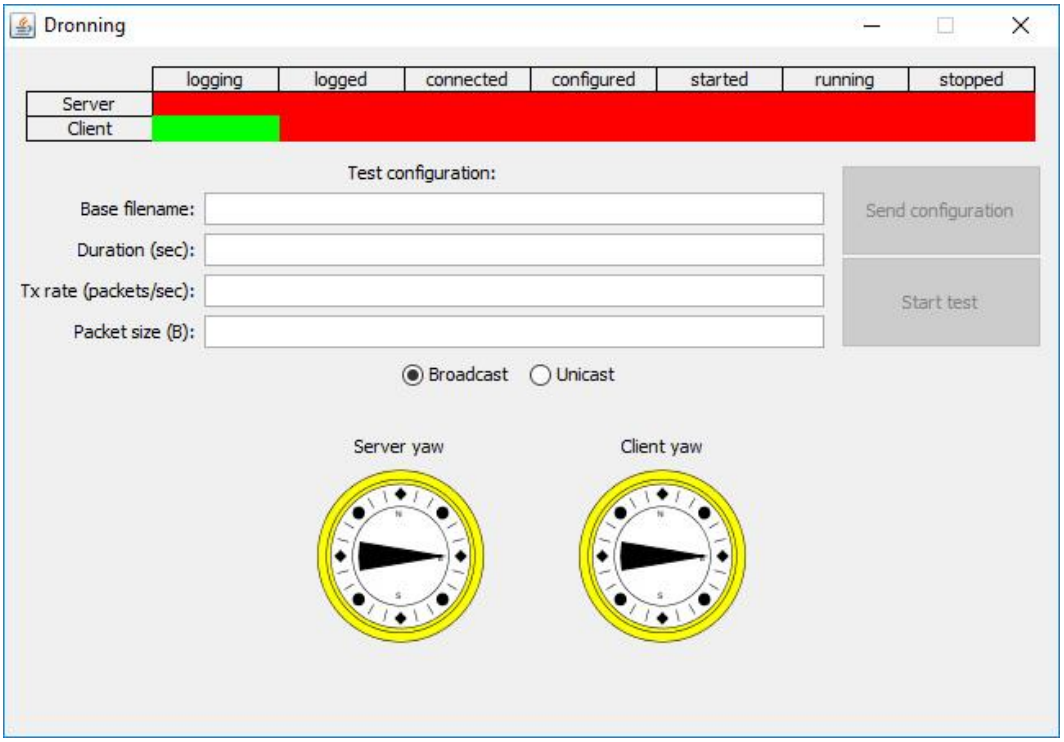


Figura 17: Interfaz gráfica durante el inicio del entorno de pruebas.

En la imagen anterior se puede observar los principales elementos que forman la interfaz:

Panel de estados. En la parte superior se observa una matriz de celdas. Para cada dron se refleja en color verde las fases del protocolo de comunicaciones por las que ya ha pasado la instancia del programa que ejecuta. Este panel fue especialmente útil durante el proceso de *debugging* para detectar algunos fallos de lógica que impedían el progreso de la toma de datos.

Test configuration. En este panel se introducen los parámetros de la prueba. Se incluyen los parámetros duración de la prueba, tasa de envío de paquetes, tamaño de paquete y modalidad de envío: unicast o broadcast. También se incluye el nombre base del fichero en el que el dron almacenará la información, para facilitar la identificación posterior de cada toma de datos.

A la derecha se ubican dos botones, para pasar la configuración a los drones y para iniciar la toma de datos.

En la parte inferior se muestra la guiñada de ambos drones u orientación en el plano horizontal respecto al norte. Esta información no se actualiza hasta que ambos drones se han conectado al *controller*.

La representación gráfica de la guiñada se realizó con la intención de facilitar la interpretación de la posición relativa entre ambos drones antes de realizar cada toma de datos. Ambos dibujos son imágenes generadas mediante la librería JFreeChart [32] y representan una brújula.

Aunque se pude escoger distintos tipos de aguja y distintos colores, se decidió utilizar una aguja de gran grosor y color negro sobre blanco para proporcionar alto contraste, dado que la pantalla del ordenador portátil utilizado muestra poca visibilidad a plena luz del día.

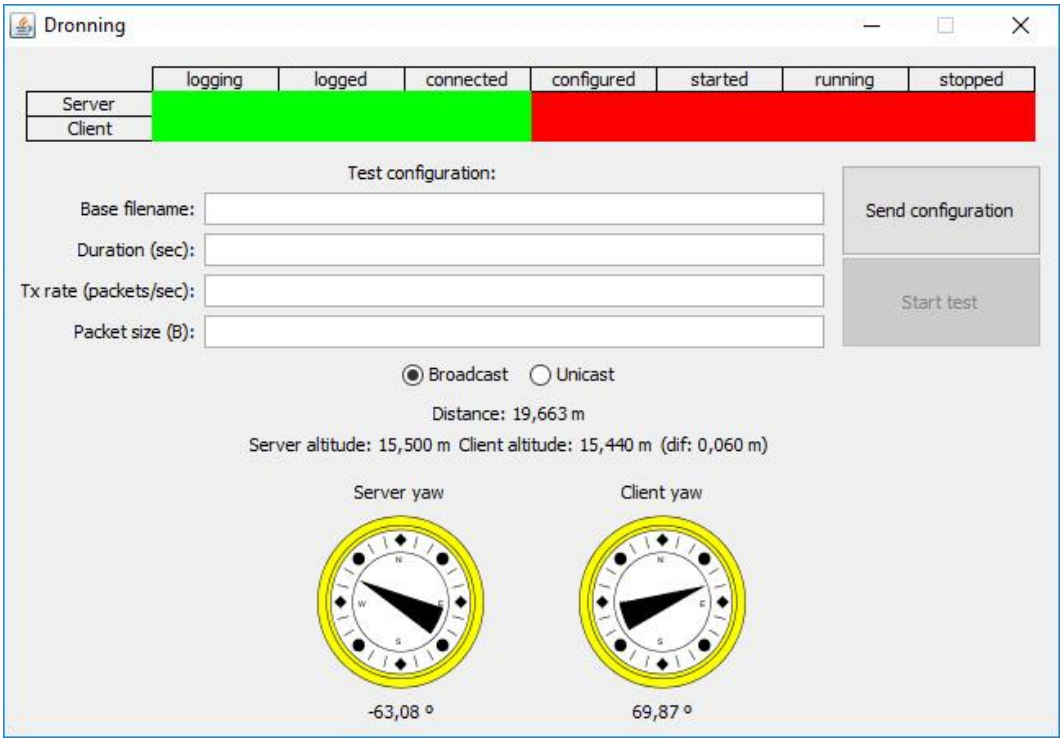


Figura 18: Interfaz gráfica lista para realizar una toma de datos.

En la anterior imagen se observa la información proporcionada por ambos drones.

La altitud, la guiñada y la distancia entre los drones permite posicionarlos fácilmente en las condiciones deseadas para la prueba, para luego iniciarla.

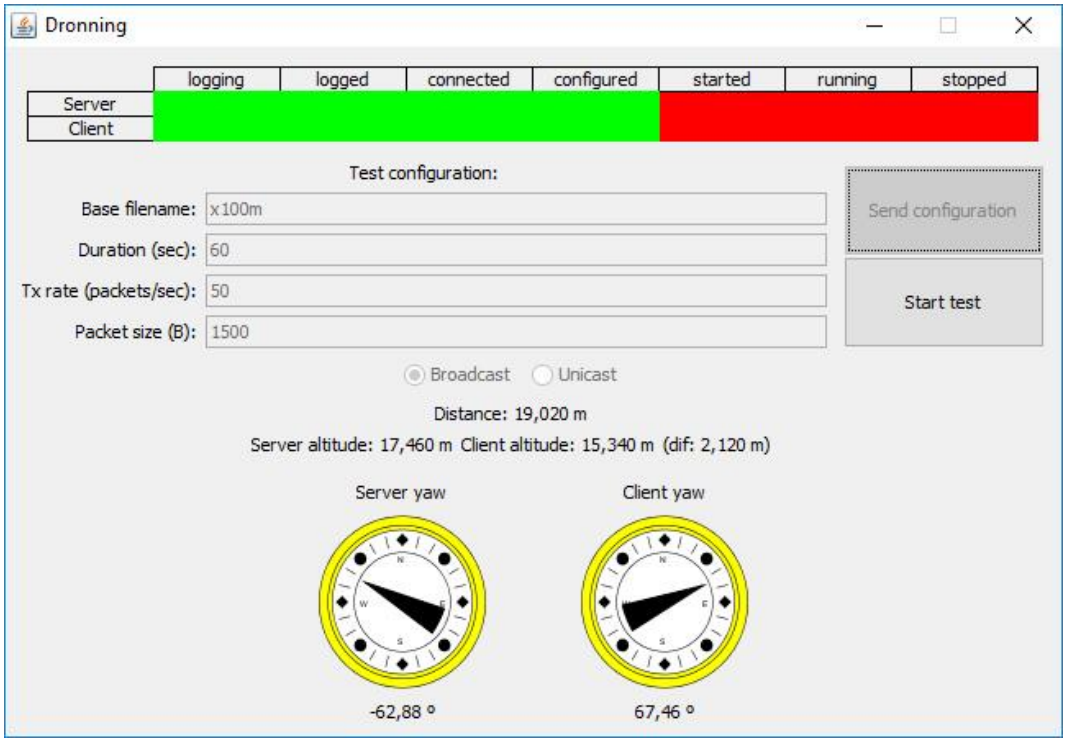


Figura 19: Interfaz gráfica con una toma de datos configurada.

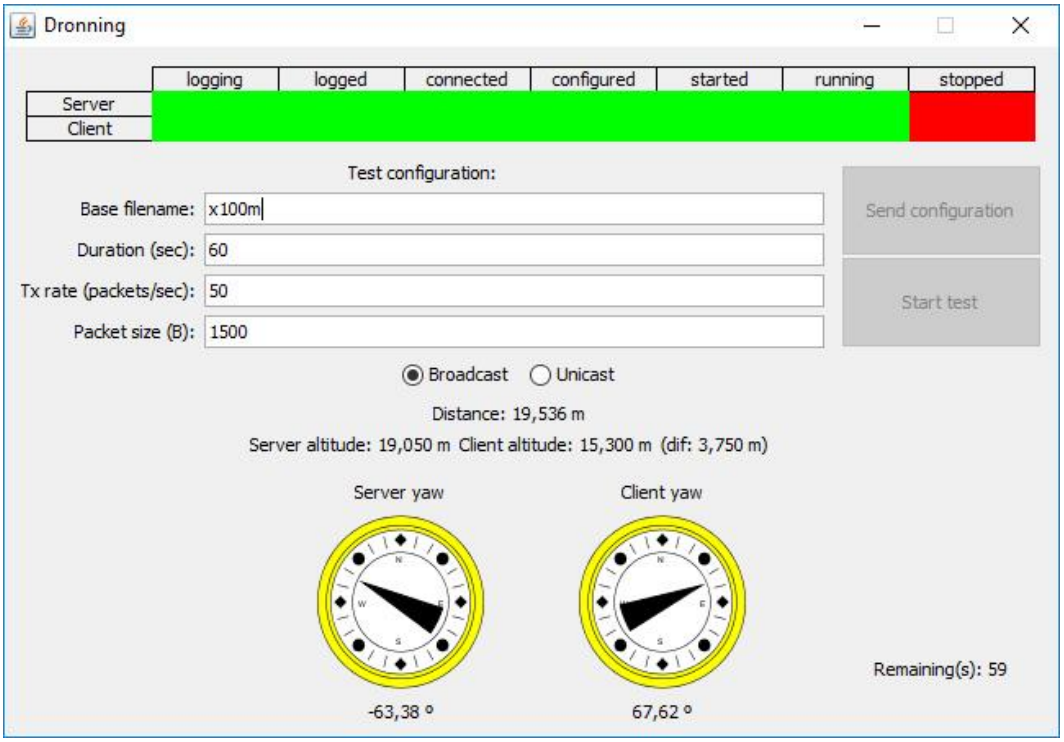


Figura 20: Interfaz gráfica mientras se realiza una toma de datos.

Mientras se realiza la toma se muestra el tiempo restante hasta su finalización. La información relativa a los drones deja de actualizarse, pues no se recibe información suficiente. Se podría enviar paquetes de datos con esta información, pero no es conveniente ocupar el medio, pues podría influir en los resultados. Además, la información está siendo registrada en el dron que realiza el rol *client*.

Los parámetros para la siguiente toma de datos se pueden introducir antes de que la anterior termine. Esta característica se buscó expresamente dado que la batería de los drones dura unos pocos minutos y es muy perjudicial dedicar tiempo de vuelo a otra cosa que no sea realizar tomas.

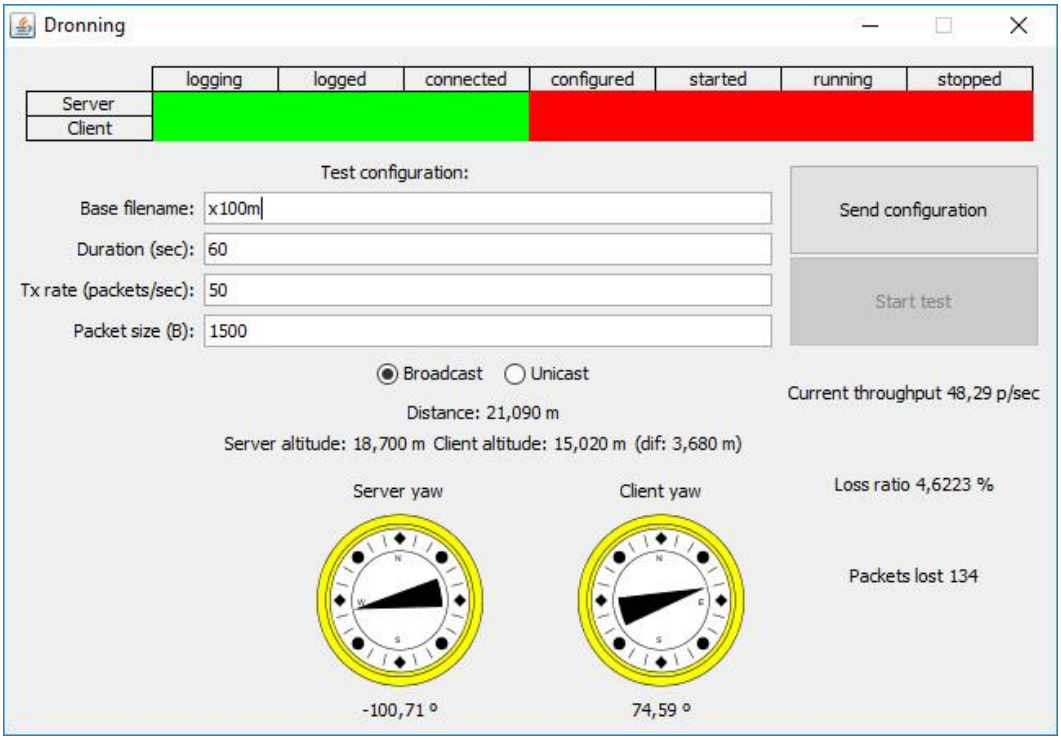


Figura 21: Interfaz gráfica cuando termina una toma de datos.

En la anterior imagen se observa que al terminar la toma se muestra en pantalla la tasa real a la que se han enviado los paquetes de datos, el porcentaje de paquetes que se han perdido en la transmisión y su número.

Al mismo tiempo, se reanuda la actualización de la posición de los drones con objeto de facilitar su reubicación para la siguiente toma.

5.8 Tareas adicionales

5.8.1 Inicio de la aplicación

El mismo programa se ejecuta en tres dispositivos diferentes. Sin embargo, en los tres casos debe tener un comportamiento totalmente diferente. Para ello, al ejecutar el programa se especifica el rol que ejerce la aplicación; *client*, *server* o *controller*.

En el caso de los drones se configuró el sistema para que inicie la aplicación automáticamente en cada inicio [25]. Para ello se introdujo la siguiente línea en el fichero "~/.config/lxsession/LXDE/autostart":

```
@/usr/bin/lxterminal -e /home/pi/Desktop/cliente.sh
```

, lo que ejecuta un script con la siguiente orden:

```
java -jar /home/pi/Desktop/dronning.jar client 2>&1 | tee –a
     /home/pi/Desktop/log.txt
```

El comando anterior lanza la aplicación con el rol especificado, *client* en este caso, junta la salida estándar y la salida de error, y las envía a un fichero de *log* empleado como control por si en algún momento se produce algún problema con la ejecución del programa.

En el ordenador portátil se creó un script que simplemente lanza la siguiente orden:

```
java -jar dronning.jar controller
```

5.8.2 Trasvase de datos desde y hacia los dispositivos

Cada vez que se genera una nueva versión de la aplicación es necesario transferirla a la Raspberry Pi 2 instalada en los drones. Para ello se decidió utilizar la herramienta gratuita WinSCP [36], que permite la copia segura de ficheros entre Windows y otros sistemas por SSH.

La misma herramienta se utilizó también para transferir la información recopilada durante las tomas de datos desde los drones al ordenador personal donde se realizó el análisis de los datos.

En casos puntuales se utilizó también la versión gratuita de MobaXterm [29], que permite disponer de una terminal remota mediante conexión SSH. Por ejemplo, se utilizó para realizar modificaciones en ficheros de configuración, o para realizar un apagado controlado de la Raspberry Pi 2 con objeto de evitar daños en la tarjeta microSD que contiene el sistema. En estas situaciones se realizaba una conexión SSH a través de la red Ad-hoc y se realizaban las acciones pertinentes.

6 Planificación de la toma de datos

Una vez desarrollado el software es necesario validarlo.

Para ello se realizaron una serie de tomas de datos con las que se comprobó el correcto funcionamiento del software. Además, la información obtenida ha permitido sacar algunas conclusiones muy interesantes sobre las condiciones en que se debe utilizar el programa, tal y como se detalla más adelante en el apartado 7.3.

En la figura 13 del apartado 5.1 se observa que, para realizar tomas de datos, hace falta un mínimo de tres personas, dos pilotos para controlar los drones, y una para establecer los parámetros de la toma en el ordenador portátil que ejerce el rol *controller*.

La normativa actual para el vuelo de drones establecida por la Agencia Estatal de Seguridad Aérea (AESA) [1] es muy restrictiva en cuanto a las condiciones que se deben cumplir para poder volar un dron:

- El piloto debe superar un curso de formación autorizado por la AESA para el rango de pesos en el que se incluye el dron que pretende volar. Además, debe acreditar los conocimientos suficientes para pilotar el modelo de dron concreto que pretende utilizar.
- Está prohibido volar por la noche, en zona urbana, en espacio aéreo restringido (aeropuertos, …), sobre aglomeraciones de gente, en condiciones climatológicas adversas y a más de 120 m de altitud.

A lo largo de 2016 está previsto que salga una nueva ley [9] menos restrictiva que permitirá, entre otras cosas, volar dentro de espacio urbano o en espacio aéreo restringido, eso sí, cumpliendo condiciones especiales.

En el caso concreto que nos atañe, las pruebas se deben realizar en espacio abierto y con buenas condiciones climatológicas, para que la lluvia no dañe a los drones y el viento no influya significativamente en los resultados.

La batería de 3.300 mAh permite una duración útil de vuelo menor a 10 minutos, lo que condiciona considerablemente la planificación de las tomas de datos, que deben realizarse en serie reduciendo al mínimo posible el tiempo empleado entre tomas para modificar adecuadamente las condiciones de la misma. Por este motivo, antes de salir a campo, se debe planificar detalladamente las tomas de datos que se desea realizar.

Para determinar las tomas a realizar es necesario identificar primero los parámetros que se desea considerar para identificar correctamente los factores que pueden influir de forma significativa en la calidad de la transmisión de datos entre los drones.

Como hipótesis de partida se consideró que pueden influir los siguientes factores:

- Distancia entre los drones
- Altitud respecto al suelo
- Guiñada u orientación relativa entre los drones
- Potencia de los motores de propulsión
- Tamaño de los paquetes de datos
- Inclinación relativa entre las antenas de los adaptadores WiFi
- Viento (por oscilaciones en las anteriores variables)

A partir de estas hipótesis se decidió medir la tasa de pérdidas de paquetes de datos en las siguientes condiciones:

- Variación de la distancia en tres situaciones; con los drones apoyados en el suelo y los motores parados, apoyados en el suelo con los motores encendidos al 25% de potencia y en vuelo estático a 4 m de altitud respecto al suelo.
- Variación de la altitud hasta alcanzar los 100 m sobre el nivel del suelo, estando ubicados a 20 m de distancia.
- Variación de la guiñada en incrementos de 90º, girando cada dron en sentido inverso, hasta realizar una vuelta completa. La distancia entre los drones es de 20 m.
- Variación simultánea de la potencia de los motores de ambos drones del 0 al 100% en incrementos del 25% estando fuertemente lastrados para imposibilitar totalmente su ascensión. De nuevo, la distancia entre los drones es de 20 m. Pretende medir la posible interferencia electromagnética inducida por los motores.
- Variación del tamaño de los paquetes de datos, tomando los valores: 300, 1000 y 1500 bytes, estando los drones ubicados a 20 m de distancia. Pretende determinar la tasa de error en el canal por ruido en base a la variación de pérdida según tamaño del paquete.

Se decidió valorar la inclinación relativa de la antena a partir de los datos tomados en las pruebas anteriores, pues la única forma de valorarla en pleno vuelo habría sido atar los drones al suelo y forzarlos a desplazarse, lo que los habría inclinado lateralmente. Se consideró una práctica extremadamente peligrosa, pues va en contra de la programación del sistema de control y, además, por tratarse de una práctica a todas luces contraria a la filosofía de la normativa de vuelo vigente.

La duración de todas las tomas de datos se estableció en 60 segundos.

Durante la toma de datos se encontraron múltiples problemas que aumentaron considerablemente el tiempo necesario para esta fase.

Primero, los drones se estrellaron en varias ocasiones por problemas en la calibración del magnetómetro e incluso por otras causas que no pudo diagnosticar el mismo fabricante. Como consecuencia, no fue posible realizar pruebas en intervalos de tiempo irregulares que en conjunto supusieron casi un mes de retraso en total.

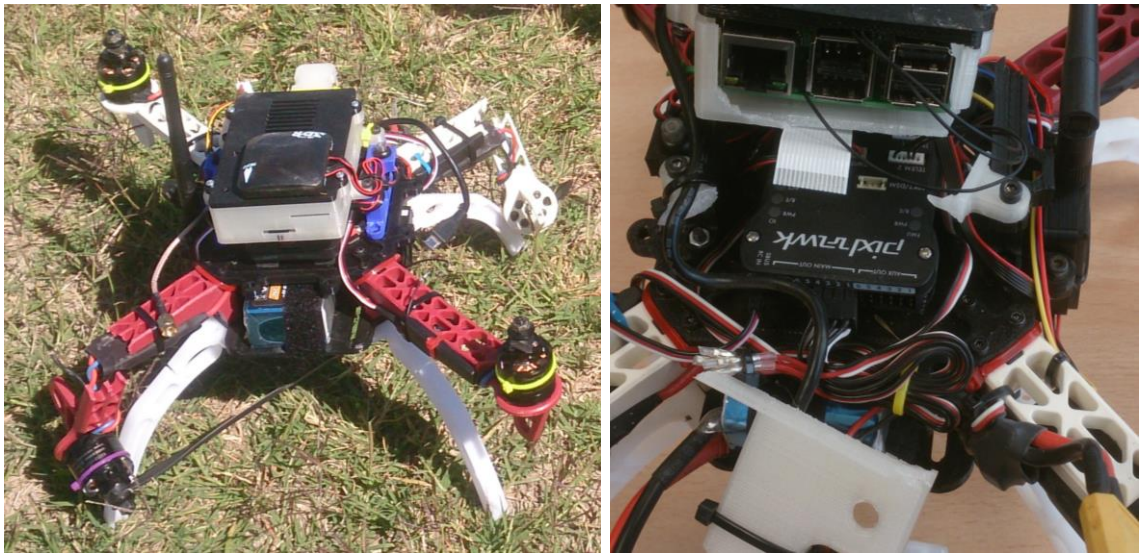


Figura 22: Resultado de varios accidentes.

Segundo, incluso a pesar de salir a tomar datos cuando la previsión meteorológica era favorable, en ocasiones había viento racheado que impedía a los drones mantener una posición estable, lo que hacía variar más de lo deseable algunos de los parámetros que se pretendía medir.

Tercero y último, durante las primeras tomas de datos hubo algunos problemas de coordinación y se detectaron necesidades que no estaban previstas, como la visualización en la aplicación de información sobre la ubicación de los drones, para facilitar su posicionamiento para cada toma.

7 Análisis de los datos

Cada una de las tomas de datos genera un fichero en formato CSV que contiene dos tipos de líneas, uno que solamente identifica un paquete recibido con el número de secuencia y otro que, además, incluye toda la información de los dos drones (*client* y *server*) ya comentada.

Con objeto de poder analizar los datos obtenidos es necesario procesarlos previamente. Dado que se inicialmente se desconoce cuáles son los parámetros que más influyen en la perdida de datos, realizar cálculos y gráficas manualmente sería muy costoso, ya que tiempo después podría ser necesario rehacer los cálculos en base a otros criterios, con la consecuente pérdida de tiempo. Por este motivo se decidió realizar un análisis inicial de forma automatizada, de forma que, en caso de buscar otro tipo de resultados, baste con modificar el programa, cargar de nuevo los datos, y procesarlos.

Se decidió programar macros en Excel 2016. Se escogió Excel porque, además de utilizar Visual Basic para Aplicaciones (VBA) que es un lenguaje sencillo, se trata de un entorno muy flexible, donde se puede realizar fácilmente cálculos sobre los resultados.

Se crearon dos plantillas de Microsoft Office Excel:

1. Análisis de una toma de datos.
2. Análisis de una serie de tomas de datos. Se trata del análisis de cómo varía la pérdida de datos al modificar un parámetro de la toma, por ejemplo, la distancia entre los drones.

Por motivos de eficiencia, antes de ejecutar código VBA se desactiva la actualización de la pantalla y los eventos, el cálculo automático y los delimitadores de página. De esta forma, el proceso de cálculo se acelera aproximadamente un orden de magnitud. Antes de terminar la ejecución de una macro se activan de nuevo las funciones indicadas, para devolver Excel al funcionamiento normal.

7.1 Desarrollo Excel para el análisis de una toma de datos

La plantilla consta de una única hoja Excel en blanco, que contiene dos botones. El primer botón permite importar ficheros de datos en formato CSV. Tras crear varias hojas que servirán para incluir gráficos, cada fichero se carga en una nueva hoja Excel [12]. El segundo botón permite procesar la información y genera los gráficos.

Se calcula el número de muestras de datos de la toma, el número de paquetes enviados, recibidos y perdidos, el porcentaje de paquetes perdidos, el tamaño mínimo y máximo de las ráfagas de paquetes perdidos de forma consecutiva, y el número de dichas ráfagas. Además, se calcula la media, desviación típica y el intervalo de confianza para el 95% de probabilidad para las siguientes variables:

- Tamaño medio de ráfaga de paquetes perdidos
- Temperatura ambiente
- Alabeo, cabeceo, guiñada y velocidad con que varían estos tres ángulos*
- Latitud, longitud, altitud, altitud relativa y coordenadas UTM*
- Velocidad según los tres ejes cartesianos del dron y velocidad en 3D*
- Orientación*
- Distancia entre los drones
- Diferencia de altitud entre los drones
- Ángulo formado por el plano de sustentación de los drones, con y sin considerar la diferencia de cota entre los mismos*
- Suma del valor absoluto del ángulo anterior de ambos drones

**Parámetros calculados tanto para el dron client como el que ejerce el rol server.*

La desviación estándar [23] y el intervalo de confianza [26] se han calculado sobre la población, y no sobre la muestra de la población, tomando el conjunto de datos disponibles.

La latitud y la longitud permiten obtener las coordenadas UTM [51] de manera similar a como se hizo en la aplicación Dronning.

La velocidad en 3D del dron se obtiene de forma trivial como módulo de la suma vectorial, puesto que se dispone de la velocidad del dron según los tres ejes cartesianos del mismo.

En el apartado 11.2 se detalla el cálculo realizado para obtener el ángulo entre el plano de sustentación de los drones.

Las gráficas generadas automáticamente [33] son las siguientes:

- Histograma del tamaño de ráfagas de paquetes perdidos
- Tamaño de las ráfagas de paquetes perdidos en función del porcentaje de progreso de la toma de datos
- Histogramas de la posición de cada dron según sus ejes cartesianos X, Y y Z
- Plano en planta de la posición del dron (ejes X e Y)
- Velocidad en 3D del dron en función del porcentaje de progreso de la toma de datos
- Ángulo entre los planos de sustentación de los drones, considerando y sin considerar la diferencia de cota entre los mismos

En el anexo II se incluye un ejemplo con la información generada automáticamente por el documento Excel.

7.2 Desarrollo Excel para el análisis de una serie de tomas de datos

Como en el caso anterior, la plantilla Excel consta de una única hoja en blanco con dos botones. El primer botón permite abrir un grupo de ficheros resultado del apartado anterior. Al hacerlo se cargan de esos ficheros todos los datos necesarios para realizar el análisis automático. El segundo botón procesa la información y genera los siguientes gráficos:

- Porcentaje de pérdidas en base a la distancia entre los drones
- Porcentaje de pérdidas en base a la altitud de uno de los drones
- Porcentaje de pérdidas en base a la diferencia de cota entre ambos drones
- Porcentaje de pérdidas en base a la distancia y la altitud
- Porcentaje de pérdida en base a la distancia y la diferencia de cota
- Tamaño de ráfaga perdida en base a la distancia entre los drones*
- Tamaño de ráfaga perdida en base a la altitud de uno de los drones*
- Tamaño de ráfaga perdida en base a la diferencia de cota entre los drones*
- Tamaño de ráfaga perdida en base a la distancia y la altitud
- Tamaño de ráfaga perdida en base a la distancia y la diferencia de cota
- Histogramas de la posición de cada dron según sus ejes cartesianos X, Y y Z
- Plano en planta de la posición del dron (ejes X e Y)

** Al disponer de información suficiente, además del valor medio se han dibujado barras de error [20] para representar el intervalo de confianza para una probabilidad del 95%.*

Como los ficheros se cargan en un orden aleatorio, para poder dibujar las gráficas correctamente hay que ordenar los datos cargados en memoria según la variable a estudiar en cada gráfica: distancia, altitud o diferencia de cota. Para ello se cargan los datos de la hoja Excel en una matriz bidimensional, y se utiliza el algoritmo QuickSort [31] para ordenarlo por una de sus columnas.

Aunque gráficos como el porcentaje de pérdidas y el tamaño de ráfaga perdida se generan siempre de forma automática, no siempre son útiles. Los gráficos basados en la distancia entre los drones serán útiles solamente si la diferencia entre las tomas de datos de la serie es la distancia entre los drones. Es decir, se podrán utilizar si la serie de tomas de datos hace referencia a esa variable, siendo conveniente borrar los gráficos en cualquier otro caso.

Es muy probable que en cada toma de muestras los drones estén ubicados en una posición diferente, por lo que los histogramas de posición y el plano de planta de posición no se podrían trazar correctamente sin realizar alguna transformación. La solución consiste en restar a las coordenadas X e Y de cada uno de los ficheros cargados el valor medio, lo que ubica al dron en la posición media (0, 0) y permite comparar el movimiento relativo entre las distintas tomas de datos.

En el anexo II se incluye un ejemplo con la información generada automáticamente por el documento Excel.

7.3 Análisis de los datos obtenidos

Una vez realizadas las tomas de datos, y habiéndolas preprocesado con los scripts Excel anteriormente descritos, es necesario realizar comparaciones de las gráficas obtenidas para averiguar las posibles conclusiones que se puedan obtener.

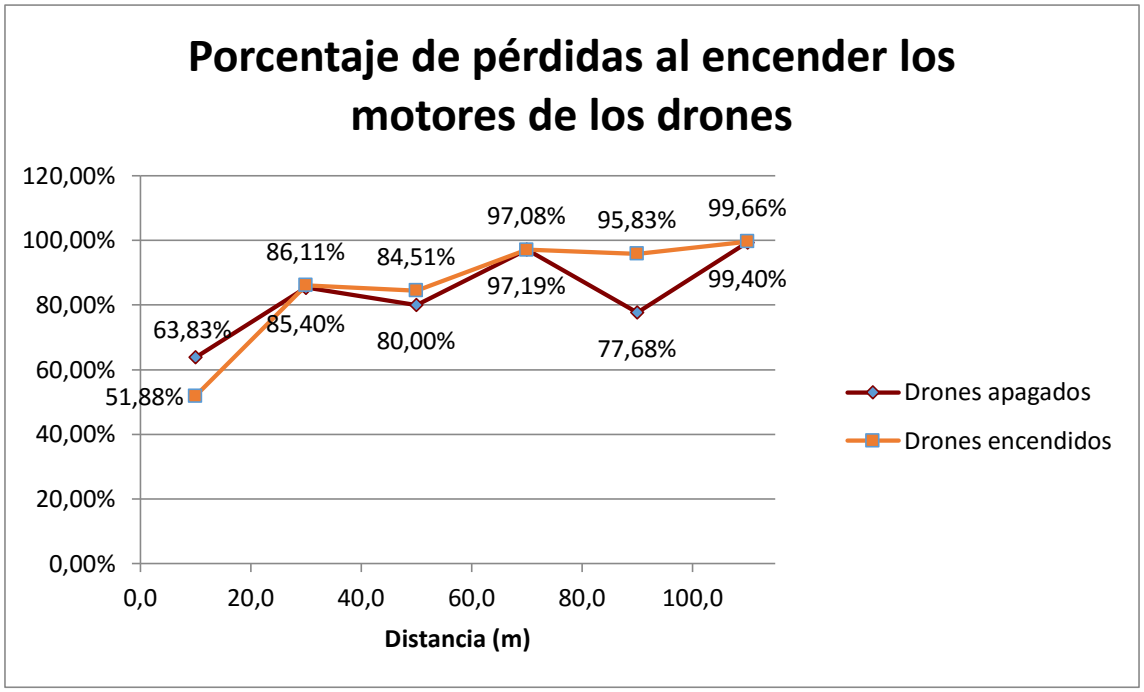


Figura 23: Pérdidas al encender motores al 25% de potencia.

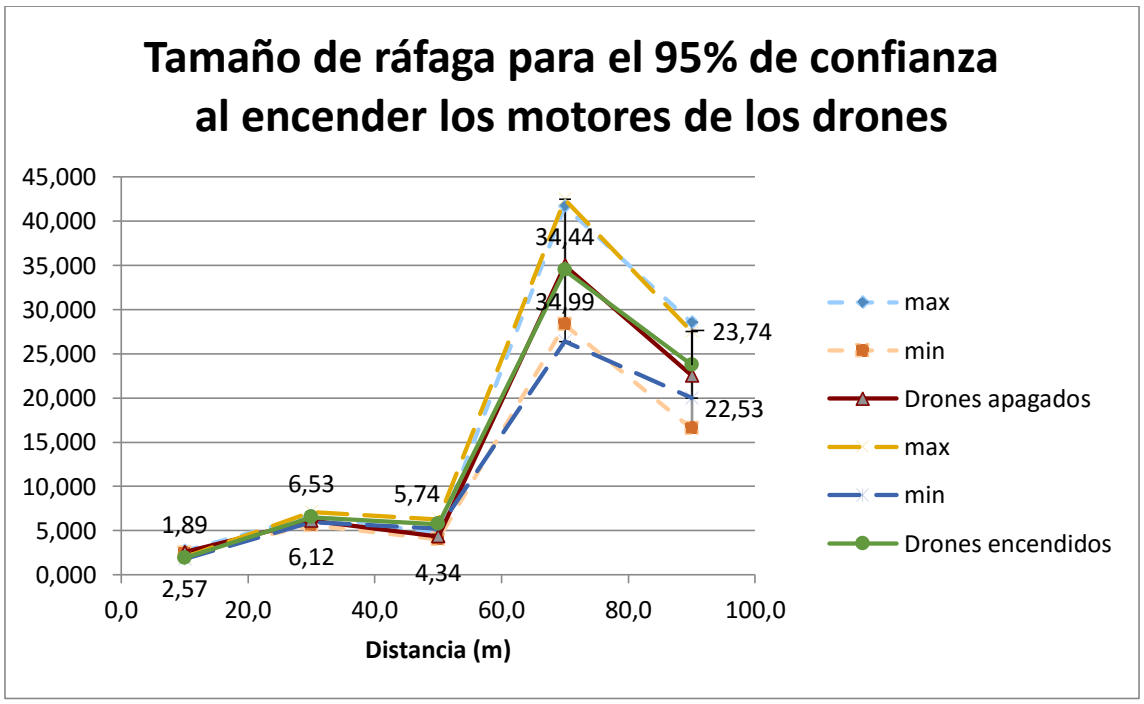


Figura 24: Tamaño de ráfaga al encender motores al 25% de potencia.

Mediante las figuras 23 y 24 se comparan las pérdidas de datos que se producen al separar los drones estando apoyados en el suelo, con la única diferencia que en una serie los drones tienen los motores apagados y en la otra, encendidos.

Se puede concluir que las pérdidas aumentan con la separación entre los drones, y que la diferencia en el tamaño de las ráfagas de paquetes perdidos no es significativa, pues los intervalos de confianza para el 95% de probabilidad se solapan.

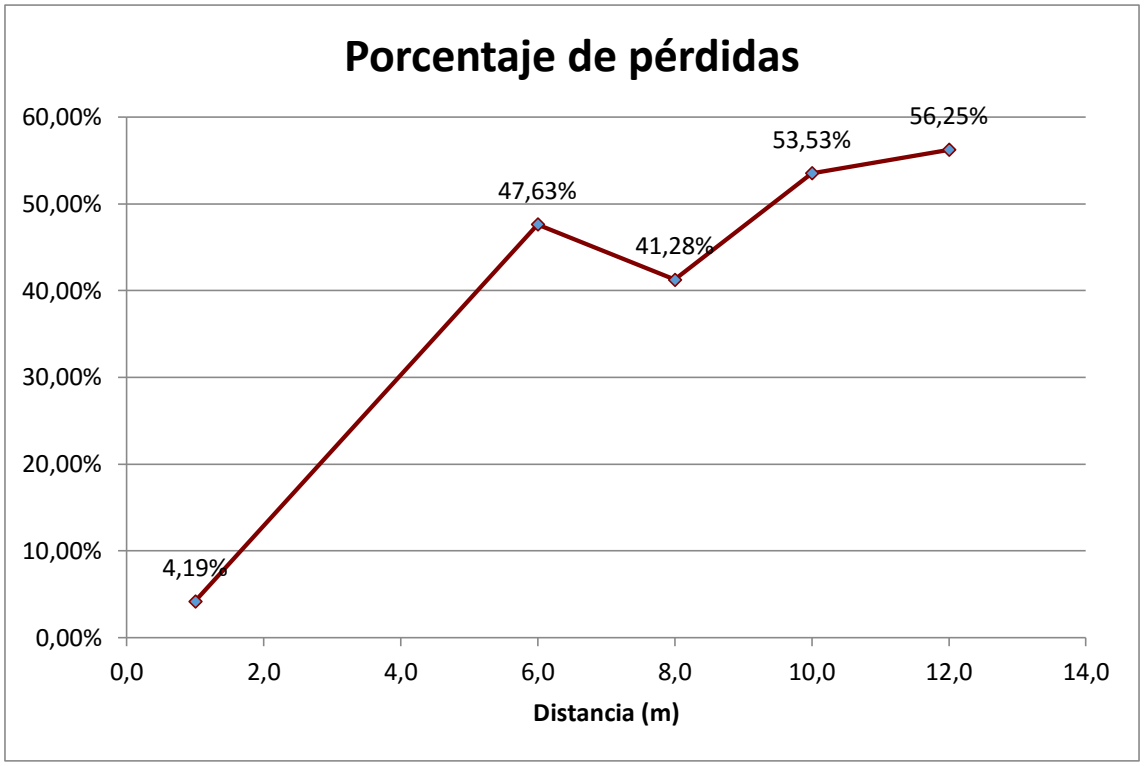


Figura 25: Pérdidas a corta distancia.

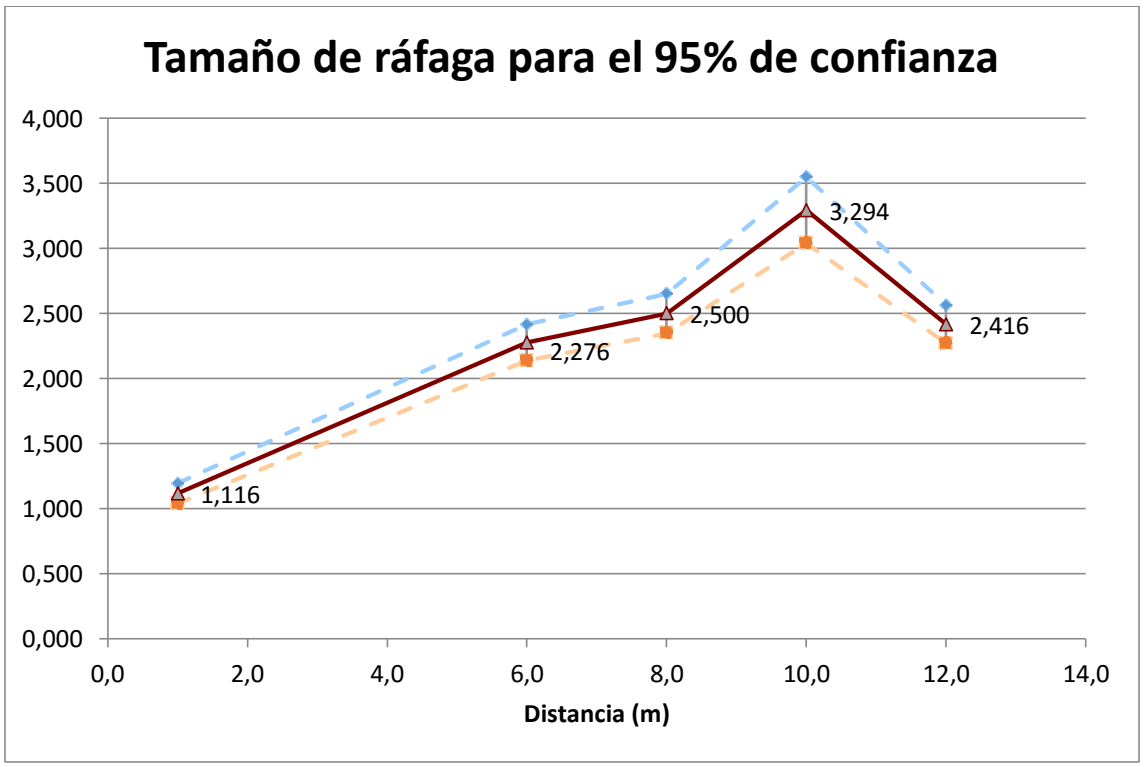


Figura 26: Tamaño de ráfaga a corta distancia.

Se repitió la prueba anterior con los motores apagados y a corta distancia. En las figuras 25 y 26 se observa que las pérdidas aumentan rápidamente al alejar los drones, lo que en principio sugiere que la distancia es un factor muy influyente en las pérdidas.

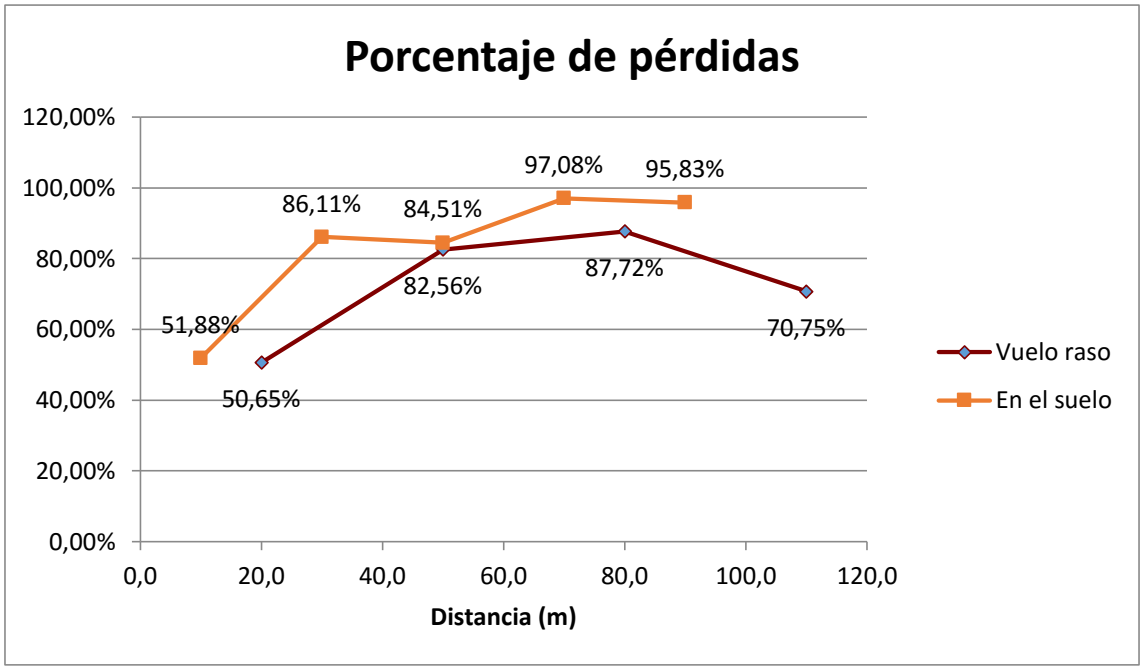


Figura 27: Pérdidas al elevar los drones.

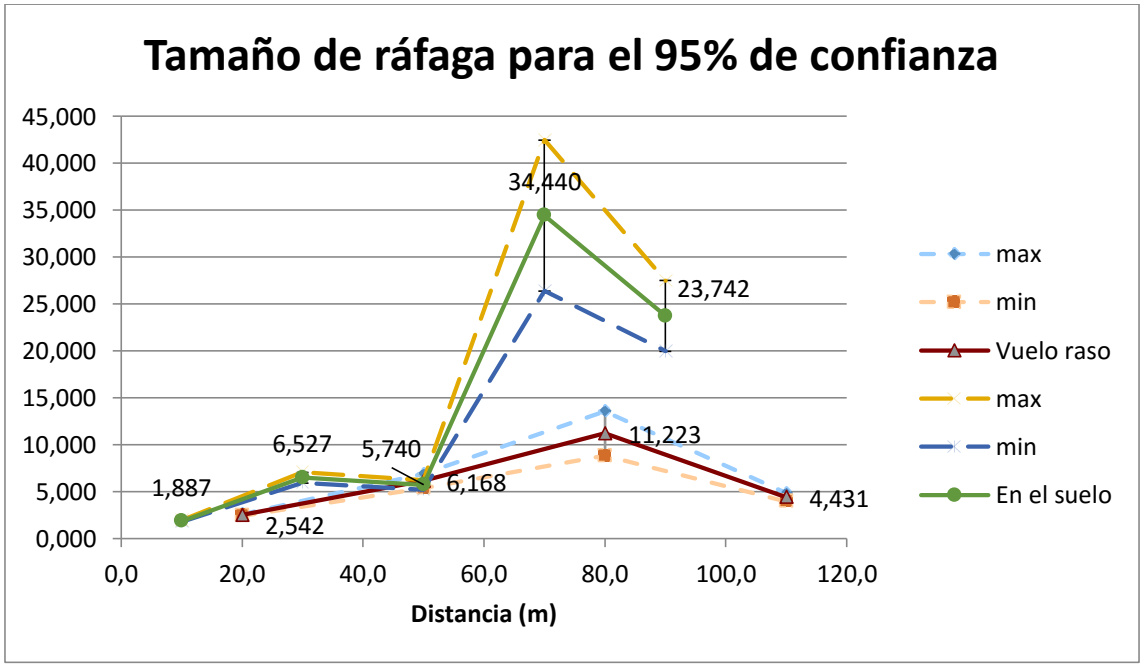


Figura 28: Tamaño de ráfaga al elevar los drones.

En las figuras 27 y 28 se compara el comportamiento volando a 4 m de altura o estando en el suelo, pero con los motores encendidos.

Los resultados parecen ser contrarios a lo esperable, pues las pérdidas son superiores estando apoyado en el suelo que en pleno vuelo. Sin embargo, las pérdidas son tan altas que indudablemente hay algún factor que distorsiona los resultados.

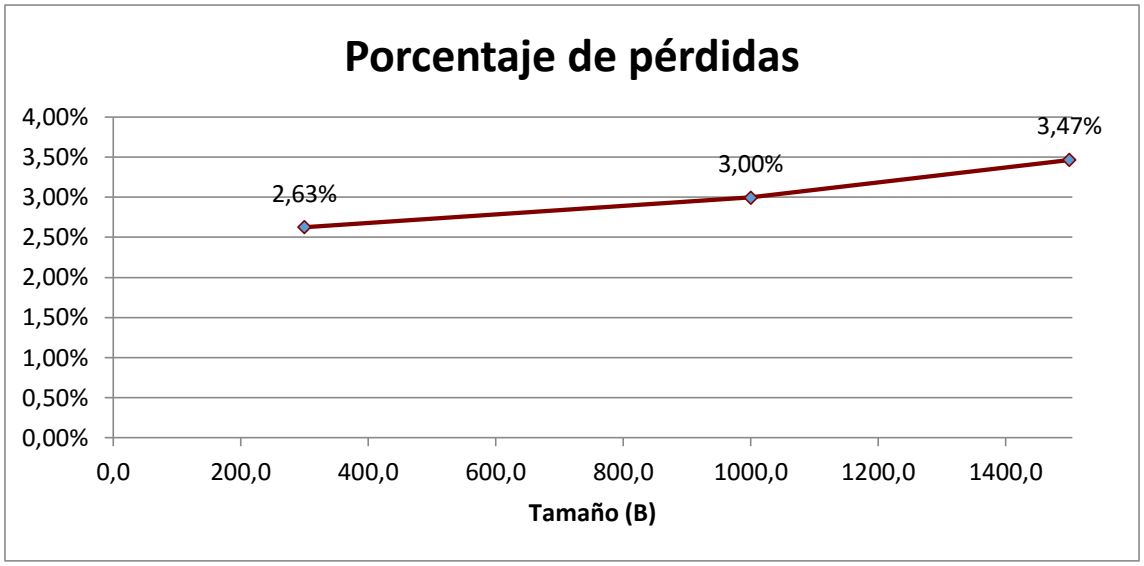


Figura 29: Pérdidas con distinto tamaño de paquete.

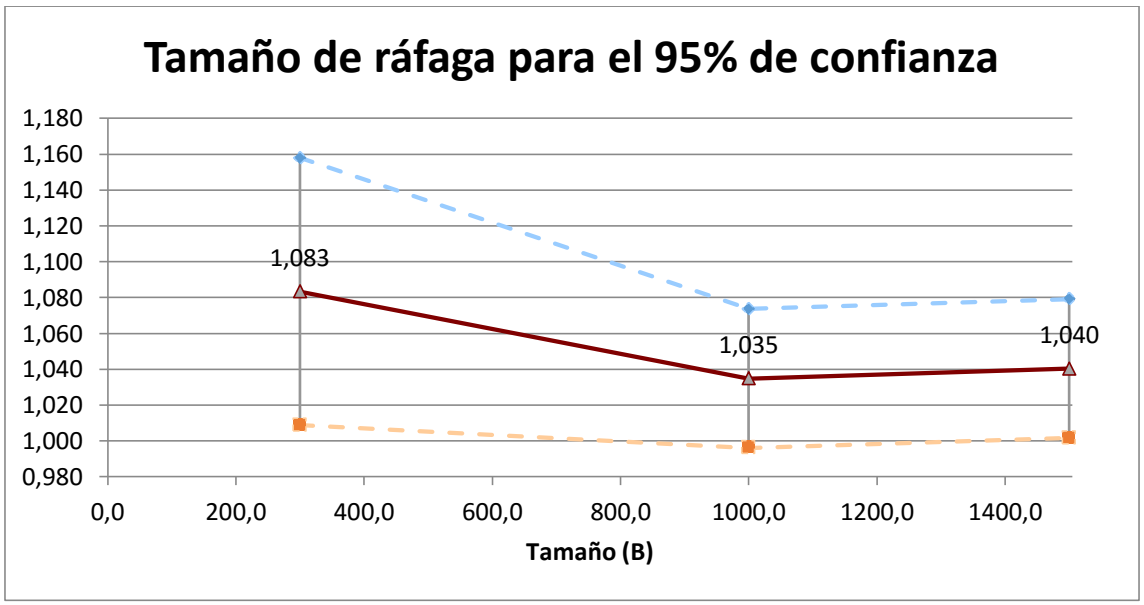


Figura 30: Tamaño de ráfaga con distinto tamaño de paquete.

En las figuras 29 y 30 se estudia cómo evolucionan las pérdidas a medida que aumenta el tamaño de paquete de datos, en concreto se ha probado con los tamaños 300, 1.000 y 1.500 bytes. Los drones estaban directamente apoyados sobre el suelo a una distancia de 20 metros y con los motores apagados.

De la primera gráfica se concluye que, con un tamaño de paquete mayor, el tiempo de ocupación del medio es mayor y, por lo tanto, también lo es la probabilidad de que se produzcan errores de transmisión.

Cabe señalar que las pérdidas son mucho menores que las mostradas en las otras pruebas. Más adelante en este mismo apartado se justifica dicha diferencia.

De la segunda gráfica se concluye que el tamaño de paquete no afecta significativamente al tamaño de ráfaga de paquetes perdidos, pues los intervalos de confianza para el 95% de probabilidad se solapan.

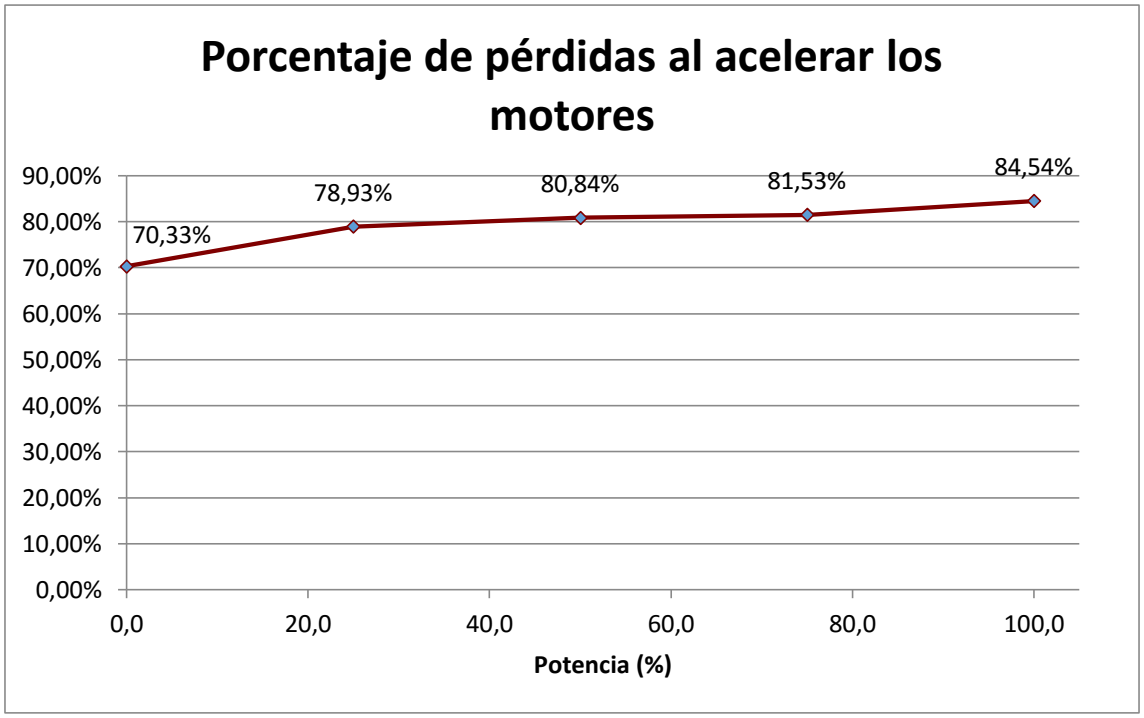


Figura 31: Pérdidas al acelerar los motores.

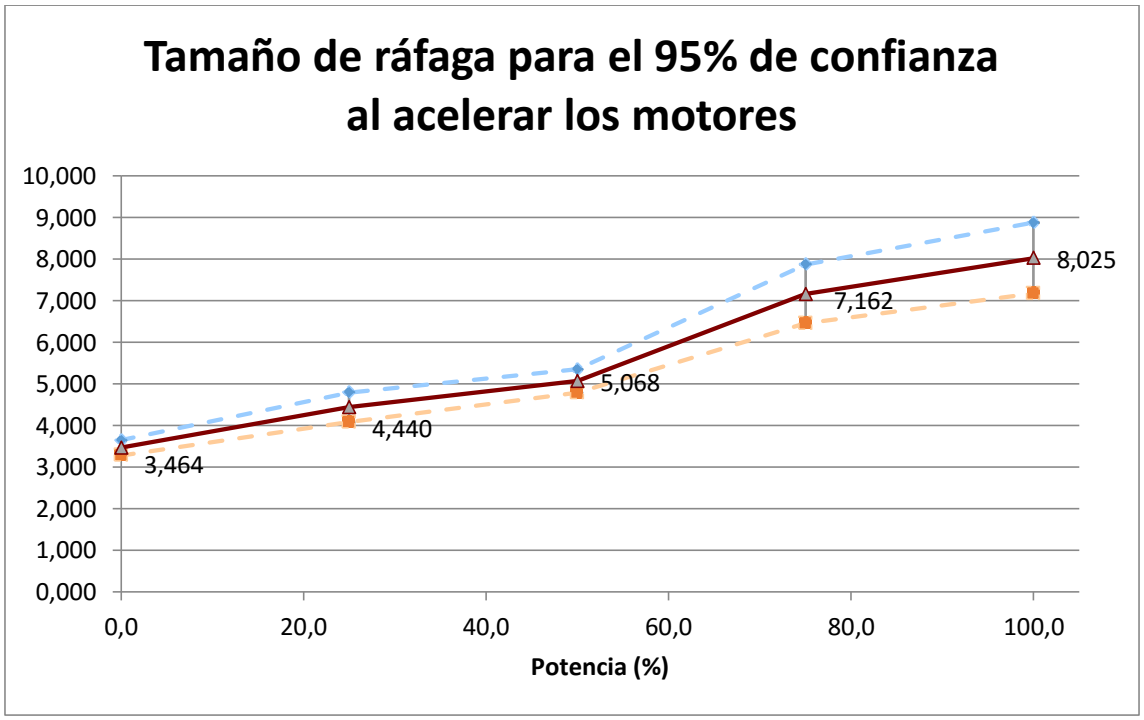


Figura 32: Tamaño de ráfaga al acelerar los motores.

Las figuras 31 y 32 muestran los resultados obtenidos con los drones separados 20 m y apoyados en el suelo. A continuación, se puso en marcha los motores a distintos niveles de potencia y se realizaron las mediciones. De dichas figuras se desprende que el aumento de potencia de los motores influye significativamente en la tasa de pérdida de paquetes. Sin embargo, en futuros estudios sería conveniente repetir el ensayo, pues los resultados pueden estar condicionados por el hecho de que los drones estaban sujetos al suelo y no en pleno vuelo.

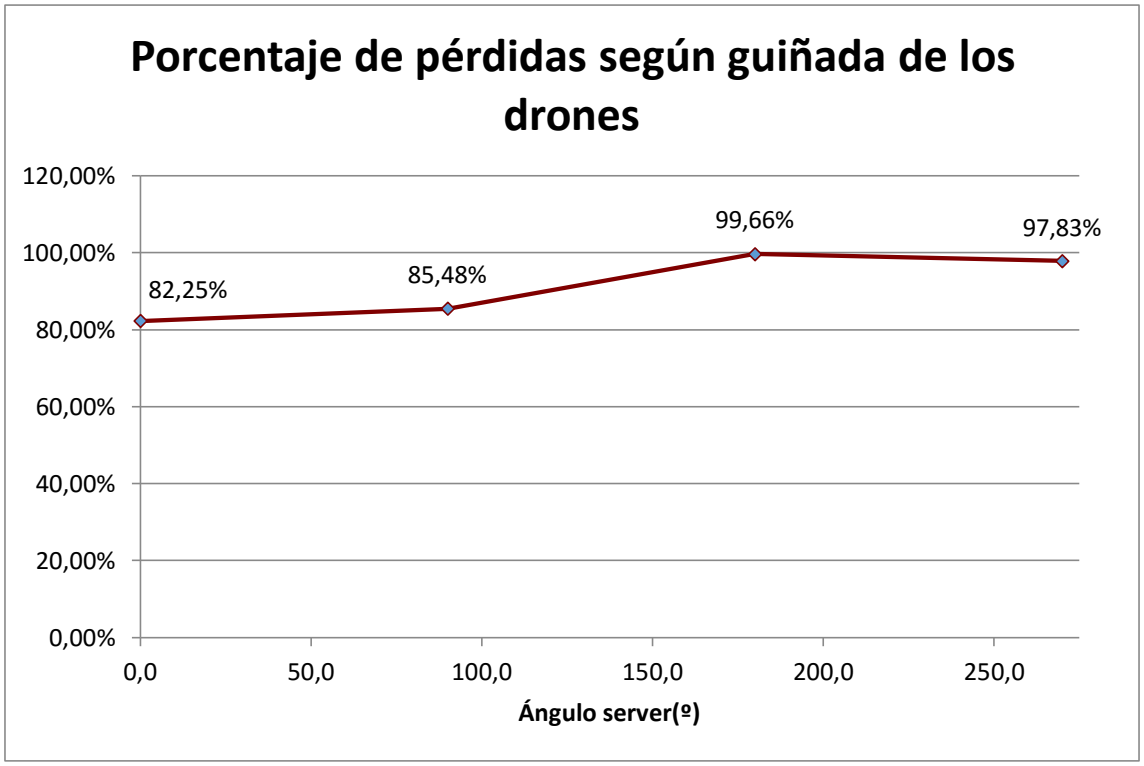


Figura 33: Pérdidas según la guiñada de los drones.

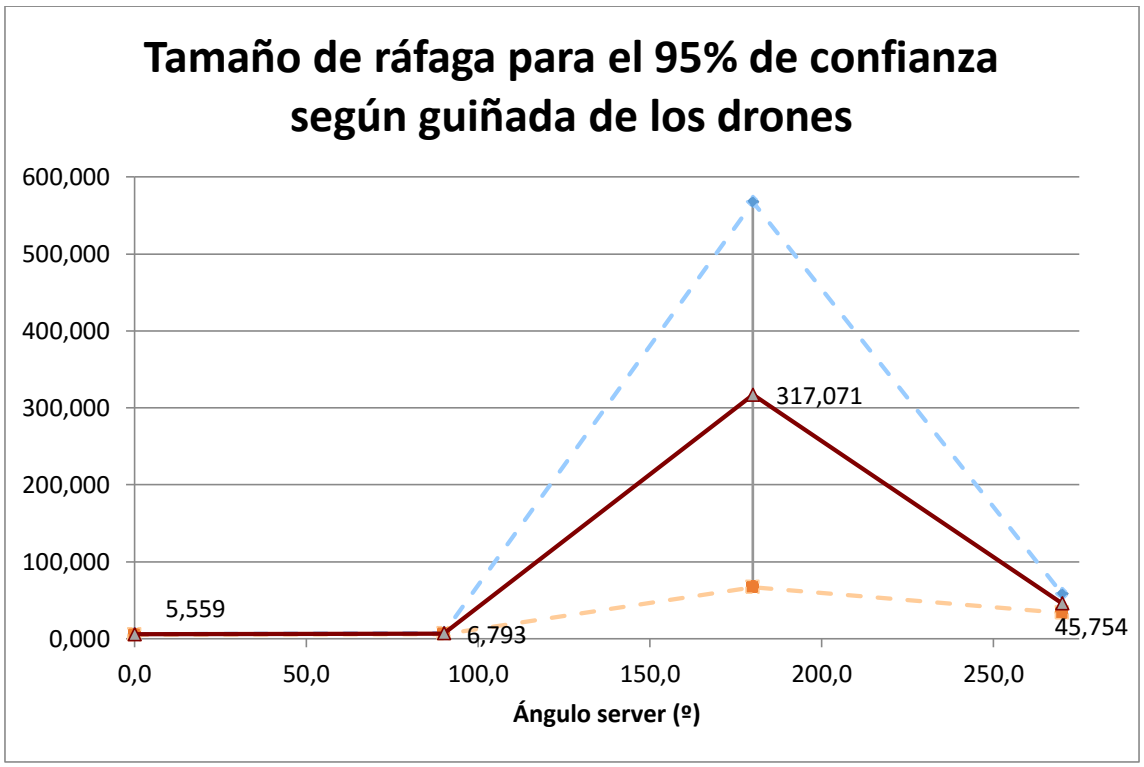


Figura 34: Tamaño de ráfaga según la guiñada de los drones.

Las figuras 33 y 34 muestran los resultados tras ubicar los drones a 20 m de distancia y girarlos 90 grados en cada toma de datos, cada dron en sentido opuesto. La prueba se realizó en pleno vuelo a 4 m de altura respecto al suelo. De las figuras indicadas no puede concluirse

claramente que la guiñada influya en la tasa de pérdidas una vez que ésta es tan elevada que los resultados no son significativos.

Como se puede observar en las gráficas mostradas hasta ahora, la tasa de pérdida de paquetes es muy elevada en general, hasta el punto de hacer difícilmente utilizable el enlace de datos.

Teniendo en cuenta las condiciones en que se realizó la prueba variando el tamaño de paquete se plantearon varias hipótesis para justificar tamaña variación.

Aunque hay factores externos que influyen seguramente en la tasa de pérdidas, como la presencia de otras redes WiFi, redes eléctricas próximas, etc., se observó que el mando a distancia empleado para el control de los drones funciona también en la frecuencia de 2,4 GHz, como la red WiFi, así que se decidió realizar algunas pruebas con los drones en el suelo y sin encender los mandos de control.

Los resultados fueron sorprendentes. Los mandos a distancia dañan considerablemente la señal WiFi, provocando una tasa de pérdidas excesiva.

El mando FrSky usa como protocolo de transmisión *Advanced Continuous Channel Shifting Technology* (ACCST), una variante de FHSS [45], protocolo utilizado por la inmensa mayoría de los mandos de control remoto existentes en el mercado. Se trata de un protocolo de salto de frecuencia con el que se transmite en toda la banda de los 2,4 GHz, lo que provoca interferencia con las transmisiones WiFi de forma masiva.

Se puede concluir de manera fehaciente que, siempre que se utilice mandos a distancia en la frecuencia de los 2,4 GHz es totalmente desaconsejable utilizar WiFi a 2,4 GHz como tecnología de enlace de comunicación entre los drones, siendo extremadamente aconsejable saltar a la banda de los 5,9 GHz.

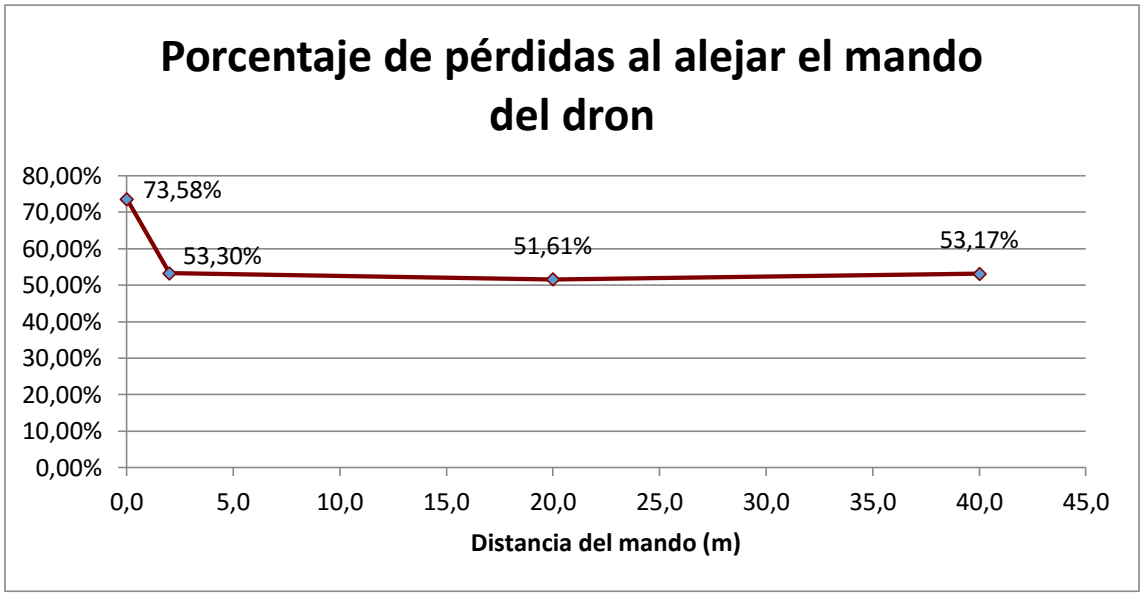


Figura 35: Pérdidas según la distancia del mando de control remoto.

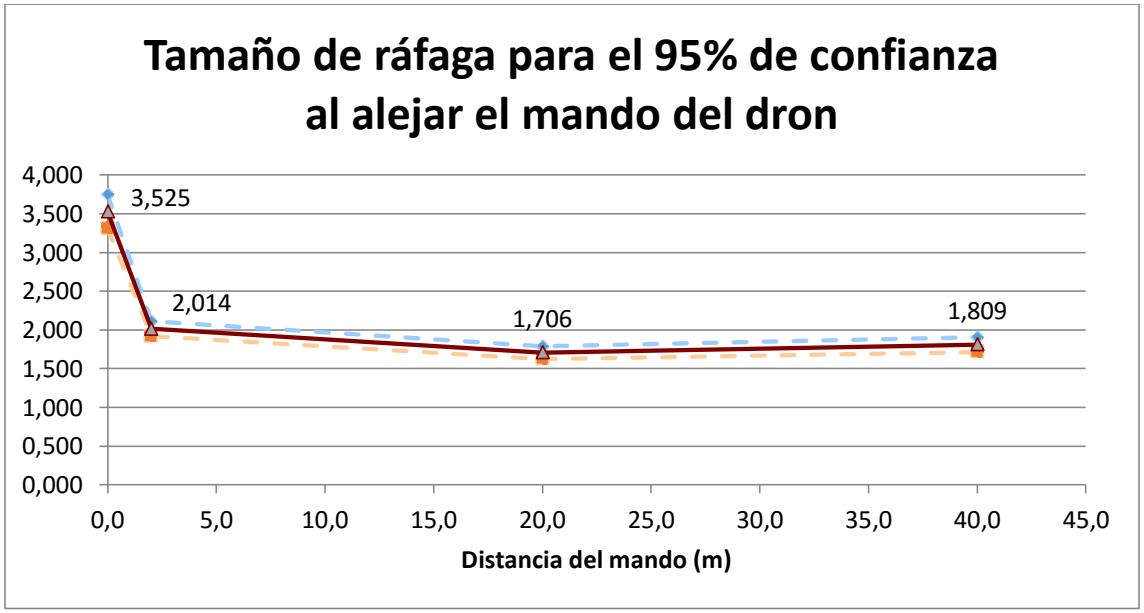


Figura 36: Tamaño de ráfaga según la distancia del mando de control remoto.

Una vez descubierto el problema con el mando a distancia, era lógico suponer que la distancia del mismo a los drones influye también en la tasa de pérdidas.

En la serie mostrada en las figuras 35 y 36 se probó con los dos drones directamente apoyados en el suelo a una distancia de 20 m y con los motores apagados. A continuación, se realizaron medidas ubicando los mandos a varias distancias.

Se observa que la proximidad del mando influye solamente a distancias muy cortas que nunca se darán durante el vuelo de los drones, por lo que esta variable no es significativa.

Sin embargo, se puede concluir que el problema de ocupación del medio lo genera el transmisor equipado en el dron y que se encarga de transmitir la información al mando, transmisor ese que va a estar siempre presente y encendido.

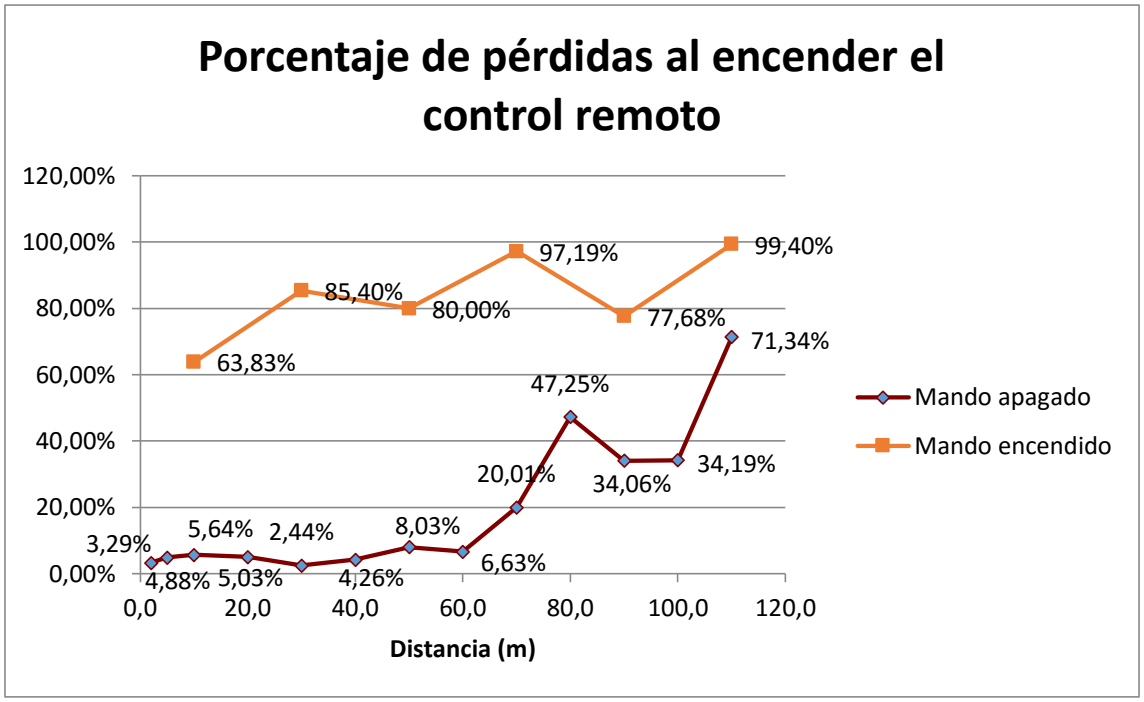


Figura 37: Pérdidas al encender el control remoto.

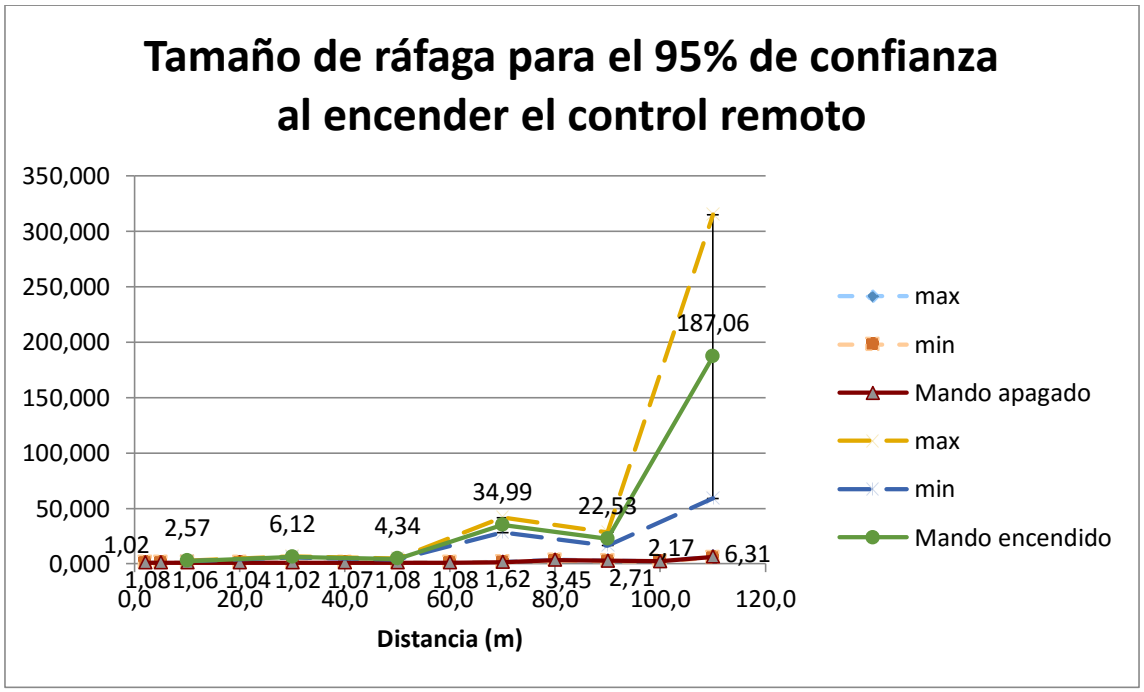


Figura 38: Tamaño de ráfaga al encender el control remoto.

En la prueba de las figuras 37 y 38 se comprobó la evolución de las pérdidas al encender y apagar los mandos, a medida que se alejaban los drones estando los motores apagados.

Los resultados obtenidos demuestran sobradamente la influencia negativa de la señal de control de los mandos en la señal del enlace WiFi. A partir de unos 100 m, con el mando encendido, la señal WiFi queda prácticamente inutilizable, ya que se alcanzaban pérdidas superiores al 99%.

El experimento realizado se considera un éxito, ya que se ha detectado un factor, la señal del control remoto, que condiciona drásticamente el empleo de la aplicación Dronning para futuros experimentos, así como el desarrollo de protocolos de comunicación entre drones en redes Ad-hoc. Como ya se ha indicado, se deduce que es más que recomendable realizar futuros desarrollos en la banda de los 5,9 GHz.

También se ha comprobado que otros factores como el tamaño de paquete, la potencia ejercida por los motores, y la distancia entre los drones influyen en la calidad del enlace WiFi.

Por otro lado, a pesar del tiempo invertido en calcular el ángulo relativo entre los planos de sustentación de los drones, en esta primera prueba no se halló ninguna correlación entre la pérdida de paquetes de datos y dicho ángulo, que puede equipararse al ángulo formado por las antenas de los adaptadores WiFi. Sin embargo, es posible que dicha correlación se detecte en futuros estudios, tras eliminar el factor del control remoto, pues enmascara la influencia de otros factores.

También se ha verificado la robustez de la aplicación Dronning, pues el programa ha seguido comportándose correctamente incluso con una tasa de pérdida de paquetes superior al 99%.

8 Conclusiones

El uso cada vez más extendido de multicópteros, usualmente denominados simplemente como drones, genera una serie de nuevos desafíos para desarrolladores, entre los que destaca la necesidad de comunicación entre dichos drones con propósitos tales como evitar choques o permitir realizar vuelos sincronizados con enjambres de drones, entre otros. Por ello, cabe estudiar con más detalle la comunicación entre dichos dispositivos para averiguar la eficacia y prestaciones de diferentes tecnologías alternativas.

En este proyecto se propone un entorno que permite estudiar las comunicaciones entre multicópteros, concretamente usando redes ad-hoc basadas en WiFi. La solución propuesta incluye la plena integración de sistemas embebidos Raspberry Pi 2 en los multicópteros, el desarrollo de una aplicación para simplificar y automatizar las pruebas, y una serie de scripts que permiten realizar análisis estadísticos de los datos recolectados, así como generar diversas gráficas de interés.

Globalmente, se han cumplidos los objetivos del proyecto, pues se ha desarrollado completamente el software Dronning para el análisis de la conectividad de redes WiFi Ad-hoc, y se ha verificado su correcto funcionamiento incluso en condiciones muy adversas, cuando la calidad del enlace de comunicaciones es muy baja, lo cual complica el funcionamiento del protocolo de sincronización propuesto por la presencia de pérdidas de paquetes.

Además, se han realizado pruebas sobre una red Ad-hoc formada por dos drones conectados mediante WiFi en la banda de los 2,4 GHz, hallando conclusiones interesantes. Por un lado, se ha constatado que el uso de la banda de los 2,4 GHz es totalmente inadecuado por la interferencia que producen los mandos habitualmente utilizados para el control de este tipo de dispositivos, siendo más que recomendable adquirir y utilizar adaptadores WiFi en la banda de los 5,9 GHz. Por otro lado, se ha confirmado la influencia que tienen la distancia entre drones, la velocidad de los motores y el tamaño de los paquetes de datos en la tasa de pérdidas de paquetes.

9 Trabajos futuros

En base a los resultados de este proyecto se abre un amplio abanico de posibilidades que permitirán profundizar en áreas como:

- Comunicaciones: estudiar las prestaciones en la banda de 5.9 GHz; estudiar el uso de otros tipos de antena (planas, polarización circular), y de antenas con distintas ganancias.
- Red: estudiar la viabilidad de distintas técnicas y algoritmos de encaminamiento para redes ad-hoc basadas en multicópteros (*Flying Ad Hoc Networks*).
- Control: desarrollar soluciones que permiten explotar el potencial de las comunicaciones entre multicópteros para crear *UAV Swarms* (enjambres de multicópteros) y para evitar colisiones entre ellos.
- Aplicaciones: proponer soluciones basadas en uno o varios multicópteros de cara a cubrir una determinada área de estudio de la manera más eficiente posible (por ejemplo, midiendo la calidad de aire en un sector mediante multicópteros).
- Modelado: El actual desarrollo permitirá modelar las comunicaciones entre drones, lo que a su vez servirá de base para analizar la corrección de protocolos y aplicaciones distribuidas ante posibles errores de comunicación.

Bibliografía

[1] AGENCIA ESTATAL DE SEGURIDAD AÉREA. *Drones – Trabajos aéreos* <http://www.seguridadaerea.gob.es/lang_castellano/cias_empresas/trabajos/rpas/default.aspx> [Consulta: 6 de junio 2016]

[2] ALEJANDRO ESQUIVA RODRÍGUEZ. *Tutorial Raspberry Pi - Wireless Ad-hoc Network* <https://geekytheory.com/tutorial-raspberry-pi-wireless-ad-hoc-network/> [Consulta 1 de junio 2016]

[3] ANTHONY GEOGHEGAN. *Linux bash script to extract IP address* <http://stackoverflow.com/questions/21336126/linux-bash-script-to-extract-ip-address> [Consulta 3 de junio 2016]

[4] ARDUPILOT DEV TEAM. *Communicating with Raspberry Pi via MAVLink* <http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html> [Consulta: 1 de junio 2016]

[5] ARDUPILOT. *Setting up SITL using Vagrant* <http://ardupilot.org/dev/docs/setting-up-sitl-using-vagrant.html> [Consulta: 24 de mayo 2016]

[6] ARDUPILOT. *SITL Simulator (Software in the Loop)* <http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html> [Consulta: 24 de mayo 2016].

[7] B. V. D. BERGH, A. CHIUMENTO, y S. POLLIN, *"LTE in the Sky: Trading off Propagation Benefits with Interference Costs for Aerial Nodes"*, en *IEEE Communications Magazine*, VOL. 54, NO. 5, PP. 44–50, MAY 2016

[8] *Broadcast* <http://www.tack.ch/multicast/broadcast.shtml> [Consulta: 3 de junio 2016]

[9] DRONE SPAIN. *Nueva ley de drones – Marzo 2016* <http://www.dronespain.pro/nueva-ley-de-drones-en-espana-marzo-2016/> [Consulta: 6 de junio 2016]

[10] ESOTERICSOFTWARE. *EsotericSoftware/kryo: Java serialization and cloning: fast, efficient, automatic* <https://github.com/EsotericSoftware/kryo> [Consulta: 3 de junio 2016]

[11] E. YANMAZ, S. HAYAT, J. SCHERER, y C. BETTSTETTER, *"Experimental Performance Analysis of Two-Hop Aerial 802.11 Networks"*, en *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, abril 2014, PP. 3118–3123

[12] FIONNUALA. *Macro to prompt user to select CSV files for import into existing sheet in workbook* <http://stackoverflow.com/questions/9975705/macro-to-prompt-user-to-select-csv-files-for-import-into-existing-sheet-in-workb> [Consulta: 6 de junio 2016]

[13] GARRETT GUSTAFSON. *Windows 8.1 and Windows 10 AdHoc network support solution* <https://globalcache.zendesk.com/entries/82172789-FAQ-Windows-8-1-and-Windows-10-AdHoc-network-support-solution> [Consulta: 1 de junio 2016]

[14] GHELLE. *MAVLinkJava: A Java code generator and a Java library for MAVLink* <https://github.com/ghelle/MAVLinkJava> [Consulta: 31 de mayo 2016]

[15] *Git for Windows* <https://git-for-windows.github.io/> [Consulta: 24 de mayo 2016]

[16] HASHICORP. *Vagrant by HashiCorp* <https://www.vagrantup.com/> [Consulta: 24 de mayo 2016]

[17] *How to suspend a Java thread for a small period of time, like 100 nanoseconds?* <http://stackoverflow.com/questions/11498585/how-to-suspend-a-java-thread-for-a-small-period-of-time-like-100-nanoseconds> [Consulta: 5 de junio 2016]

[18] JAMES SUTHERLAND. *Java Persistence Performance* <http://java-persistence-performance.blogspot.com.es/2013/08/optimizing-java-serialization-java-vs.html> [Consulta 3 de junio 2016]

[19] JAVIN PAUL. *Swing is not thread-safe in Java – What does it mean?* <http://javarevisited.blogspot.com.es/2013/08/why-swing-is-not-thread-safe-in-java-Swingworker-Event-thread.html> [Consulta: 5 de junio 2016]

[20] JON PELTIER. *Custom Error Bars in Excel Charts* <http://peltiertech.com/custom-error-bars-in-excel-charts/> [Consulta: 7 de junio 2016]

[21] M. ASADPOUR, B. V. DEN BERGH, D. GIUSTINIANO, K. A. HUMMEL, S. POLLIN, y B. PLATTNER, *"Micro Aerial Vehicle Networks: An Experimental Analysis Of Challenges and Opportunities"* en *IEEE Communications Magazine*, VOL. 52, NO. 7, PP. 141–149, Julio 2014

[22] M. ASADPOUR, D. GIUSTINIANO, y K. A. HUMMEL, *"From ground to aerial communication: Dissecting WLAN 802.11N for the drones"*, en *Proceedings of the 8th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, SER. WINTECH '13, 2013, PP. 25–32

[23] MACROPTION. *Population vs. Sample Variance and Standard Deviation* <http://www.macroption.com/population-sample-variance-standard-deviation/> [Consulta: 6 de junio 2016]

[24] MAKOTO YUI. *Atomic Float* <http://www.java2s.com/Code/Java/Threads/AtomicFloat.htm> [Consulta: 3 de junio 2016]

[25] MATT. *How To Autostart Apps In Raspbian LXDE Desktop* <http://www.raspberrypi-spy.co.uk/2014/05/how-to-autostart-apps-in-rasbian-lxde-desktop/> [Consulta: 3 de junio 2016]

[26] MICROSOFT. *Descripción de las funciones estadísticas de confianza en Excel* <https://support.microsoft.com/es-es/kb/828124> [Consulta: 6 de junio 2016]

[27] MICROSOFT. *Programas de software de hoja de cálculo* <https://products.office.com/es-es/excel> [Consulta: 24 de mayo 2016]

[28] *MinGW | Minimalist GNU for Windows* <http://www.mingw.org/> [Consulta: 24 de mayo 2016]

[29] MOBATEK. *MobaXterm free Xserver and tabbed SSH client for Windows* <http://mobaxterm.mobatek.net/> [Consulta: 6 de junio 2016]

[30] N. AHMED, S. S. KANHERE, y S. JHA, *"On the Importance of Link Characterization for Aerial Wireless Sensor Networks"*, en *IEEE Communications Magazine*, VOL. 54, NO. 5, PP. 52–57, MAY 2016

[31] NILE. *Sorting a multidimensional array in VBA* <http://stackoverflow.com/questions/4873182/sorting-a-multidimensionnal-array-in-vba> [Consulta: 7 de junio 2016]

[32] OBJECT REFINERY LIMITED. *JFreeChart* <http://www.jfree.org/jfreechart/index.html> [Consulta: 6 de junio 2016]

[33] PELTIER TECHNICAL SERVICES, INC. *Quick Excel Chart VBA Examples* <http://peltiertech.com/Excel/ChartsHowTo/QuickChartVBA.html> [Consulta: 7 de junio 2016]

[34] PI4J. *The PI4J Project* < http://pi4j.com/> [Consulta: 1 de junio 2016]

[35] *Pixhawk and Telem2 port on Raspberry Pi* <http://discuss.ardupilot.org/t/pixhawk-and-telem2-port-on-raspberry-pi/3151> [Consulta: 1 de junio 2016]

[36] pRIKRYL. *WinSCP* <https://winscp.net/eng/docs/lang:es> [Consulta: 6 de junio 2016]

[37] *Programando y desarrollando – Protocolo MAVLink* <https://noescomolocuentan.wordpress.com/2014/10/18/programando-y-desarrollando-protocolo-mavlink/> [Consulta: 31 de mayo 2016]

[38] PX4 AUTOPILOT. *Pixhawk Autopilot. Pixhawk Flight Controller Hardware Project* <https://pixhawk.org/modules/pixhawk> [Consulta: 1 de junio 2016]

[39] QGROUNDCONTROL. *Field Reordering and CRC Extra Calculation* <http://qgroundcontrol.org/mavlink/crc_extra_calculation> [Consulta: 31 de mayo 2016]

[40] QGROUNDCONTROL. *MAVLink micro air vehicle marshalling/communication library* <https://github.com/mavlink/mavlink> [Consulta: 31 de mayo 2016]

[41] QGROUNDCONTROL. *MAVLink Release 1.1 Compatibility and Features* <http://qgroundcontrol.org/mavlink/release_11> [Consulta: 31 de mayo 2016].

[42] QUATERNIUM (2015). *GRCQuad. Caracterización de la aeronave*. Documento proporcionado con el dron en su adquisición.

[43] RASPBERRY PI FOUNDATION. *Download Raspbian For Raspberry Pi* <https://www.raspberrypi.org/downloads/raspbian/> [Consulta 1 de junio 2016]

[44] RASPBERRY PI FOUNDATION. *GPIO: Models A+, B+, Raspberry Pi 2 B and Raspberry Pi 3 B* <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/README.md> [Consulta: 1 de junio 2016]

[45] RC MODEL REVIEWS. *Review: FrSky/FriSky 2.4GHz FHSS (Getting Technical)* <http://www.rcmodelreviews.com/frskyreview2.shtml> [Consulta: 7 de junio 2016]

[46] REVEREND GONZO. *What's the purpose of sleep(long millis, int nanos)?* <http://stackoverflow.com/questions/6553225/whats-the-purpose-of-sleeplong-millis-int-nanos> [Consulta: 5 de junio 2016]

[47] RXTX. *Download - RXTX* <http://rxtx.qbang.org/wiki/index.php/Download> [Consulta: 1 de junio 2016]

[48] RXTX. *Rxtx* <http://rxtx.qbang.org/wiki/index.php/Main_Page> [Consulta: 1 de junio 2016]

[49] SZATI. *How to get Serviio running on Raspbian on a Raspberry Pi* <http://wiki.serviio.org/doku.php?id=howto:linux:install:raspbian> [Consulta: 1 de junio 2016]

[50] USER2548538. *Java, convert lat/lon to UTM* <http://stackoverflow.com/questions/176137/java-convert-lat-lon-to-utm> [Consulta 5 de junio 2016]

[51] VÍCTOR CRUZ. *Pedazo de código: Conversión Grados Sexagesimales a UTM* <http://vmcruz.tumblr.com/post/78178763080> [Consulta: 6 de Junio 2016]

[52] VIRTUALBOX. *Oracle VM Virtualbox* <https://www.virtualbox.org/> [Consulta: 24 de mayo 2016]

[53] WIKIPEDIA. *Red ad hoc inalámbrica* <https://es.wikipedia.org/wiki/Red_ad_hoc_inal%C3%A1mbrica> [Consulta: 1 de junio 2016]

[54] WIKIPEDIA. *Teorema de muestreo de Nyquist-Shannon* <https://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon> [Consulta 6 de junio 2016]

[55] WIRESHARK. *Wireshark. Go deep* <https://www.wireshark.org/> [Consulta: 24 de mayo 2016]

[56] YLEARN. *Why does my UDP Broadcast wireless communication is capped at 1MBs?* <http://networkengineering.stackexchange.com/questions/1782/why-does-my-udp-broadcast-wireless-communication-is-capped-at-1mbs> [3 de junio 2016]

[57] Y. ZENG, R. ZHANG, y T. J. LIM, *"Wireless Communication with Unmanned Aerial Vehicles: Opportunities and Challenges"*, en *IEEE Communications Magazine*, VOL. 54, NO. 5, PP. 36– 42, MAY 2016

[58] ZUI. *Java and RXTX* <https://www.raspberrypi.org/forums/viewtopic.php?t=12452> [Consulta: 1 de junio 2016]

Anexo I: Instalación y manejo del simulador

El simulador SITL permite realizar pruebas con el software desarrollado para el presente trabajo sin necesidad de instalarlo en un dron real.

Instalación de SITL en una máquina virtual [5]

Antes de proceder a la instalación en Windows es necesario instalar previamente el siguiente software:

- VirtualBox [52]. Es un software de virtualización de licencia GPL. Permite crear la máquina virtual sobre la que funciona SITL.
- Vagrant [16]. Facilita la creación de un entorno de desarrollo sobre una máquina virtual, es decir, facilita la instalación de SITL en una máquina virtual Linux.
- Git para Windows [15]. Sistema de control de versiones. Esta herramienta permite descargar Ardupilot del repositorio público en el que se encuentra, para realizar la instalación.
- SSH. Es necesario disponer del programa de línea de comandos para conexiones seguras, ya que la comunicación entre el *host* y la máquina virtual se realiza mediante conexión segura y por línea de comandos. Para ello se optó por instalar MinGW [28], dado que incluye SSH entre otras muchas utilidades.

Una vez clonado el repositorio, basta con entrar en la carpeta “Vagrant” del proyecto Ardupilot y lanzar el comando “vagrant up” para arrancar la máquina virtual con Ardupilot dentro del sistema de ficheros de la misma.

Manejo del simulador

Los comandos son muy sencillos. Tras poner en marcha la máquina virtual y de conectar por SSH a su consola, tan solo queda poner en marcha el dron virtual con la configuración adecuada.

El dron simulado envía datos de telemetría al host mediante una conexión UDP por la tarjeta de red de la máquina virtual. Al encenderlo hay que especificar la IP y puerto UDP del *host* en la red virtual, dirección a la que enviará la información indicada.

```
vagrant up
vagrant ssh
sudo /vagrant/Tools/autotest/sim_vehicle.sh -L Poli --out 192.168.56.1:14550
```

Desde la consola de la máquina virtual se puede dar instrucciones al dron para cambiar su comportamiento como si se hiciese desde el mando de radio control.

Una vez realizadas las pruebas pertinentes, para detener la ejecución del dron simulado se pulsan las teclas “Ctrl+d”, se sale de la consola de la máquina virtual con la orden “exit” y por último se detiene la máquina virtual con la orden “vagrant suspend”.

La recepción de la telemetría por UDP permite realizar pruebas de software, así como lanzar la aplicación APM Planner 2 para visualizar el comportamiento del dron simulado.

Anexo II. Cálculo del ángulo entre los drones, en el plano vertical que los contiene

En el apartado 6 se considera como hipótesis de partida que la inclinación de un dron respecto al otro puede influir en la calidad del enlace de comunicación.

En el presente apartado se detalla la justificación matemática de la obtención del ángulo formado por el plano de sustentación de un dron y la línea que une los dos drones.

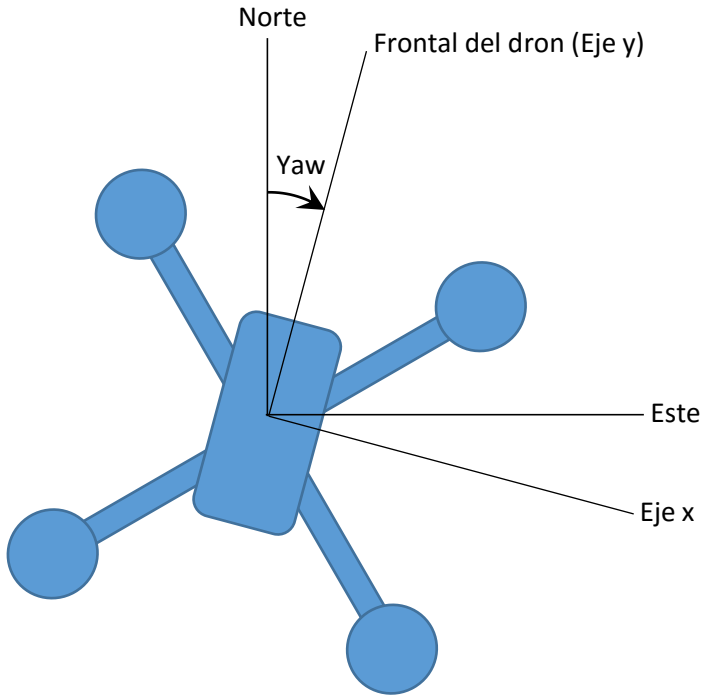


Figura 39: Vista esquemática cenital del dron.

En la anterior figura se observan los ejes norte y este, definidos por las coordenadas geográficas obtenidas del GPS del dron, así como los ejes X e Y definidos arbitrariamente como referencia para realizar las transformaciones de ángulos.

Como se observa, es el eje y el que utiliza internamente la Pixhawk como referencia para medir la desviación del vuelo respecto al norte geográfico, lo que se conoce como guiñada (*yaw*).

Durante el vuelo, el plano horizontal del dron observado en la figura adoptará inclinaciones según los ejes x (cabeceo o *pitch*) e y (alabeo o *roll*). En el primer caso el giro será positivo si el dron eleva la parte frontal, mientras que en el segundo lo será también si se inclina lateralmente hacia la derecha.

Además de los tres ángulos ya comentados, para analizar la inclinación del dron respecto del dron con el que se está comunicando, será necesario conocer también su posición relativa, de la cual se podrá obtener el ángulo que forman en la horizontal y el ángulo que forman en la vertical.

A partir de todos estos datos iniciales proporcionados por la Pixhawk se puede determinar el ángulo entre el plano horizontal del dron, con su inclinación, y la línea que une los dos drones.

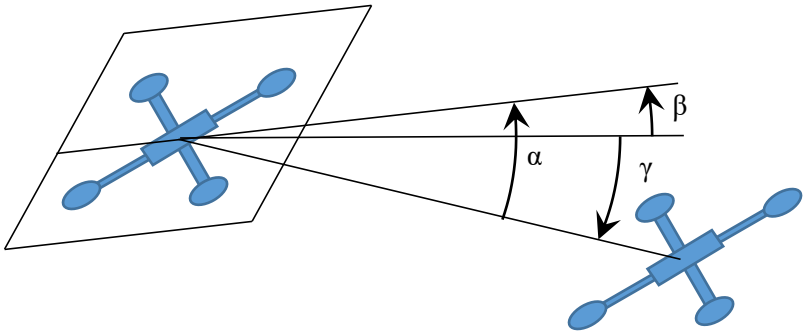


Figura 40: Vista de perfil según la proyección UTM.

El ángulo α entre la recta 3D que une los drones y el plano de sustentación del dron será la combinación de dos ángulos:

- β entre el plano de sustentación del dron y el plano horizontal con la misma cota que el mismo.
- γ entre el plano horizontal con la misma cota que el dron y la línea que une los drones.

Obtención de β

a) Obtención del plano de sustentación del dron respecto a los ejes X e Y.

La ecuación del plano la podemos obtener a partir de tres puntos cualesquiera, conocidos los ángulos δ (cabeceo) y ε (alabeo).

$$P_0\ (0, 0, 0)$$
$$P_1\ (0, 1, tg\delta)$$
$$P_2\ (1, 0, -tg\varepsilon)$$

A partir de los tres puntos se obtienen dos rectas contenidas en el plano:

$$V_1 = P_1 - P_0 = (0, 1, tg\delta)$$
$$V_2 = P_2 - P_0 = (1, 0, -tg\varepsilon)$$

La ecuación del vector normal al plano se obtiene como el producto vectorial de ambos vectores:

$$Vn = V_1 \times V_2 = (-tg\varepsilon, tg\delta, -1)$$

Por último, dado que el plano pasa por el origen de coordenadas, la ecuación general del plano tiene la forma:

$$-tg\varepsilon \cdot x + tg\delta \cdot y - z = 0$$

b) Obtención del ángulo Θ que forma la proyección horizontal de la recta que une los drones con el eje X.

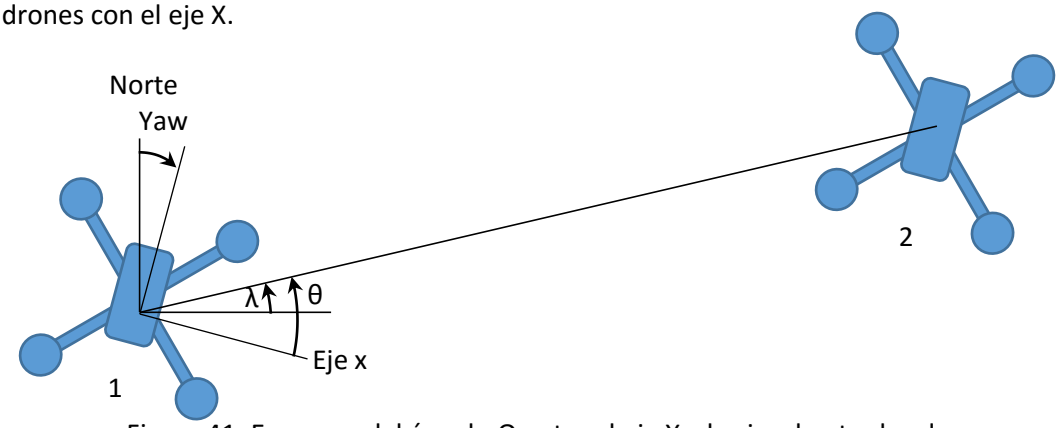


Figura 41: Esquema del ángulo Θ entre el eje X y la visual entre los drones.

El cálculo se puede realizar a partir de las coordenadas UTM de ambos drones y de la guiñada del dron que se está estudiando, en este caso el 1:

$$\Delta x = x_2 - x_1$$
$$\Delta y = y_2 - y_1$$
$$\lambda = \text{atan}(\Delta y/\Delta x)$$
$$\Theta = \lambda + \text{Yaw}$$

Hay que hacer notar que siempre existen dos ángulos distanciados π radianes que dan la misma tangente.

Por este motivo, es necesario analizar el cuadrante en el que se ubica el ángulo λ e incrementar el resultado de la fórmula. De este modo, se evita que el ángulo Θ esté mal representado, lo que repercute en posteriores cálculos:

- $\Delta x \geq 0$ e $\Delta y \geq 0$. El valor es el calculado.
- $\Delta x \geq 0$ e $\Delta y < 0$. $\lambda = \lambda + 2\pi$.
- Otros casos. $\lambda = \lambda + \pi$.

c) Obtención de la proyección horizontal de la línea que une ambos drones.

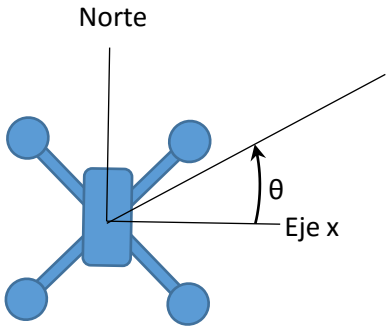


Figura 42: Proyección sobre el plano horizontal de la visual entre los drones.

Para z = 0, la ecuación de la línea se puede obtener directamente a partir del ángulo:

$$tg\Theta = y/x$$
$$y = tg\Theta \cdot x$$

d) Proyección de un punto de la recta anterior sobre el plano de sustentación del dron.

Arbitrariamente, cogemos el punto de x = 1:

$$P_{recta}\ (1,\ tg\Theta,\ z_{recta})$$

La proyección sobre el plano se consigue sustituyendo las coordenadas x e y en la ecuación del mismo y despejando la coordenada z:

$$-tg\varepsilon \cdot 1 + tg\delta \cdot tg\Theta - z = 0$$
$$z = -tg\varepsilon + tg\delta \cdot tg\Theta$$
$$P'_{recta}\ (1,\ tg\Theta,\ -tg\varepsilon + tg\delta \cdot tg\Theta)$$

e) Obtención de β.

El ángulo entre el plano de sustentación del dron y el plano horizontal de cota 0 respecto al dron vendrá dado por el ángulo formado por la recta que une P'_{recta} con el origen de coordenadas y el plano horizontal:

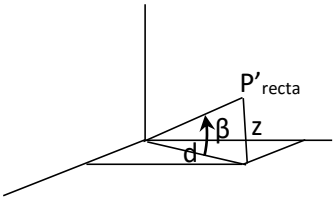


Figura 43: Esquema para el cálculo de β.

$$tg\beta = z/d$$
$$\beta = \operatorname{atan}\left(\frac{z}{d}\right) = \operatorname{atan}\left(\frac{-tg\varepsilon + tg\delta \cdot tg\theta}{\sqrt{1^2 + tg\theta^2}}\right)$$

Si x tiene un valor negativo, entonces es necesario cambiar el signo del numerador de la división, dado que en tal caso hay que tomar como referencia el siguiente punto de la recta:

$$P_{recta}\ (-1,\ -tg\Theta,\ z_{recta})$$
$$P'_{recta}\ (-1,\ -tg\Theta,\ tg\varepsilon - tg\delta \cdot tg\Theta)$$

Obtención de γ

El ángulo γ se puede obtener directamente a partir de las coordenadas de ambos drones.

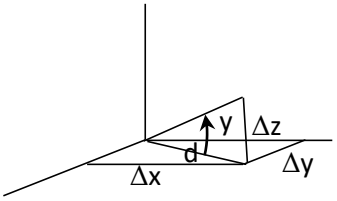


Figura 44: Esquema para el cálculo de γ.

$$tg\gamma = \Delta z/d$$
$$\gamma = \operatorname{atan}\left(\frac{\Delta z}{d}\right) = \operatorname{atan}\left(\frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}}\right)$$

Resultado final

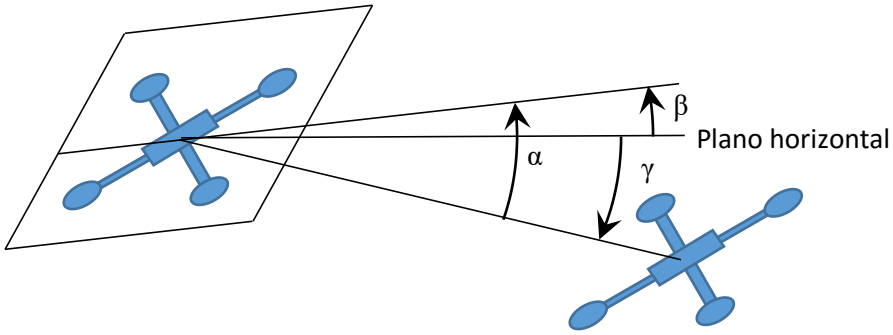
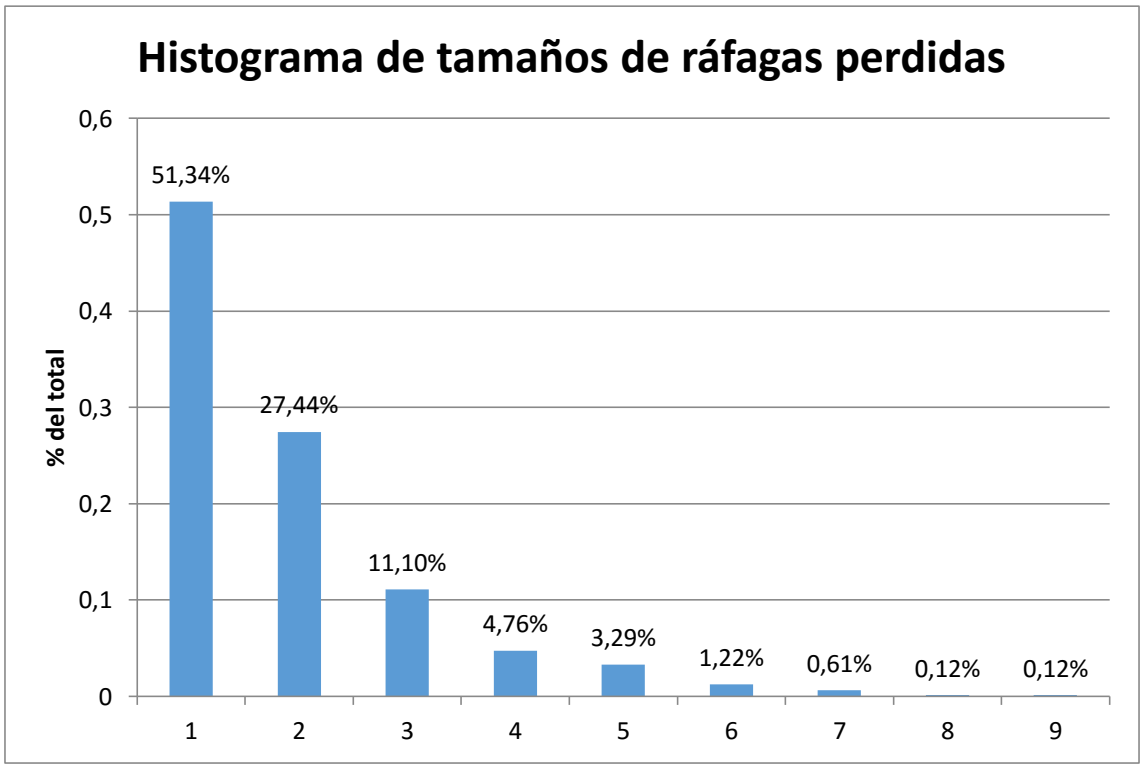


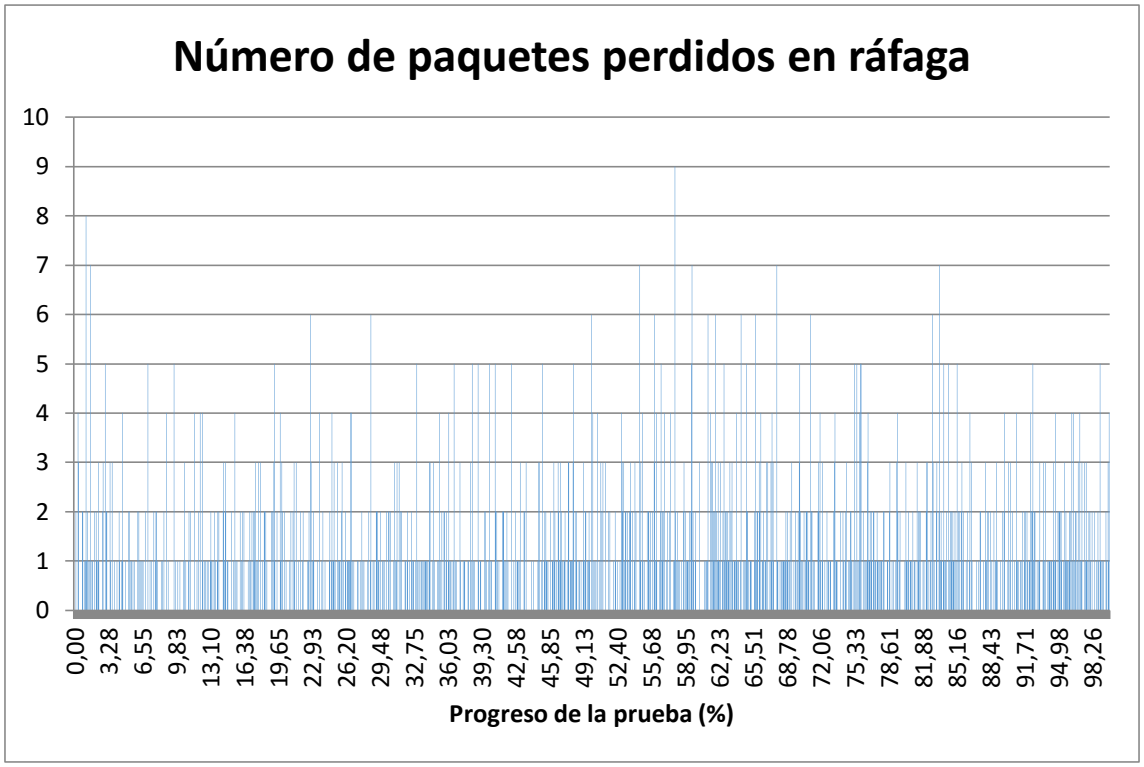
Figura 45: Obtención del ángulo α.

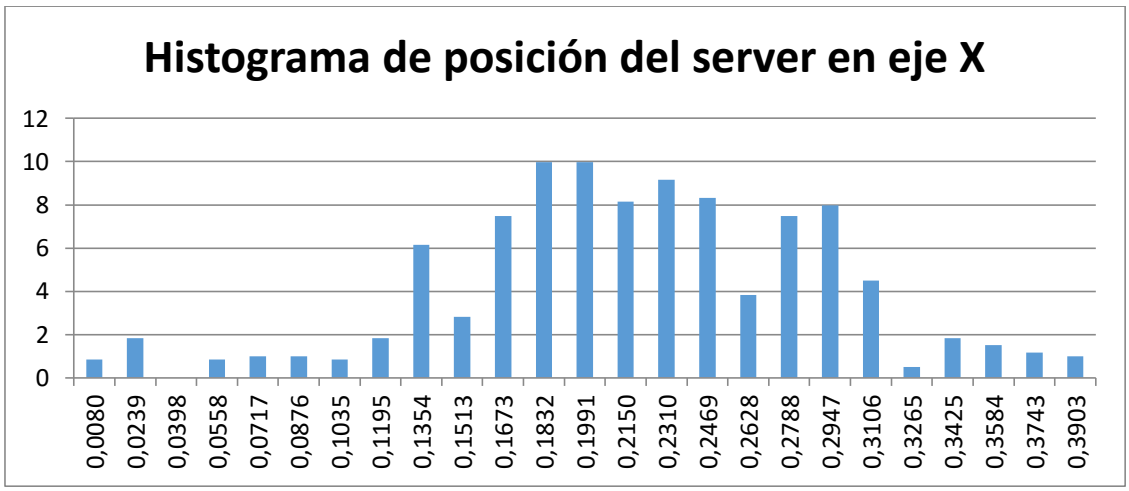
$$\alpha = \beta - \gamma$$

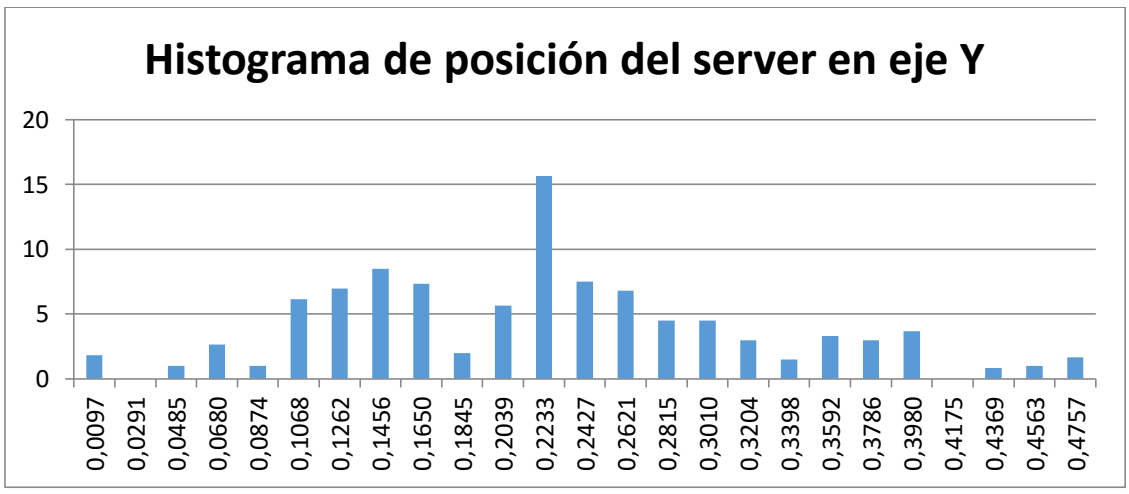
Ejemplo de análisis de una toma de datos

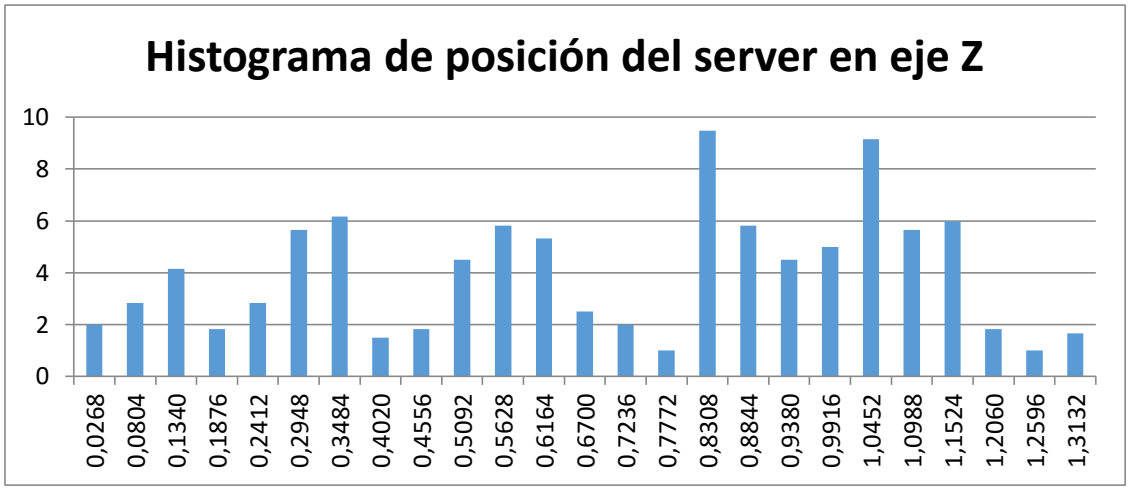
En este anexo se incluye, meramente a modo de ejemplo, los gráficos obtenidos mediante macros Excel para el análisis de los ficheros CSV generados por los drones durante las tomas de datos.

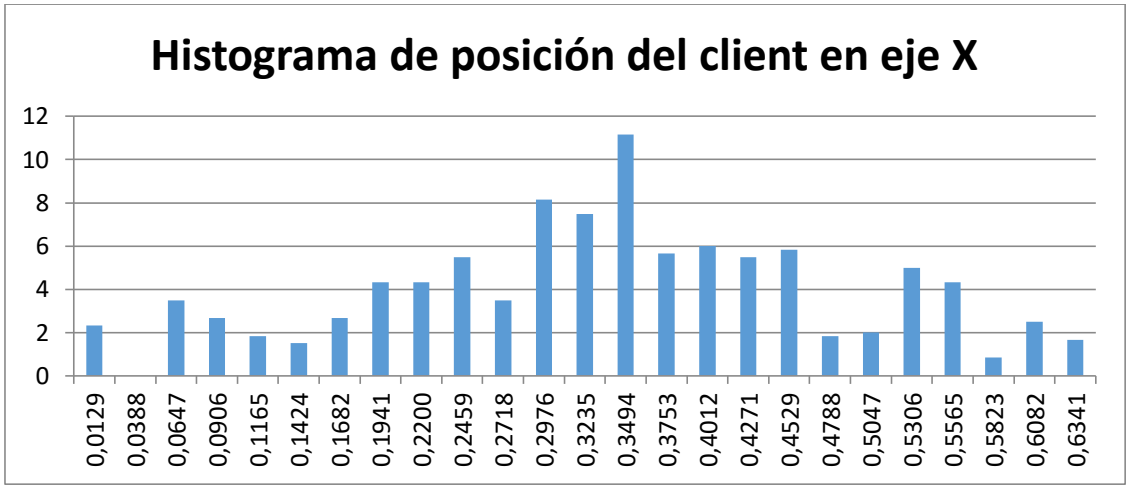


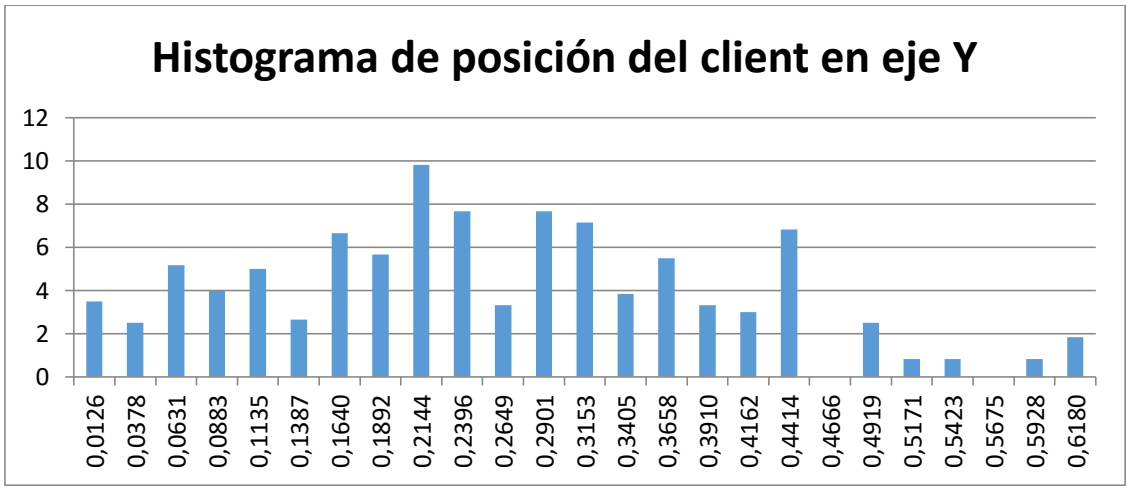

Número de paquetes perdidos en ráfaga
10
9
8
7
6
5
4
3
2
1
0
0,00
3,28
6,55
9,83
13,10
16,38
19,65
22,93
26,20
29,48
32,75
36,03
39,30
42,58
45,85
49,13
52,40
55,68
58,95
62,23
65,51
68,78
72,06
75,33
78,61
81,88
85,16
88,43
91,71
94,98
98,26
Progreso de la prueba (%)

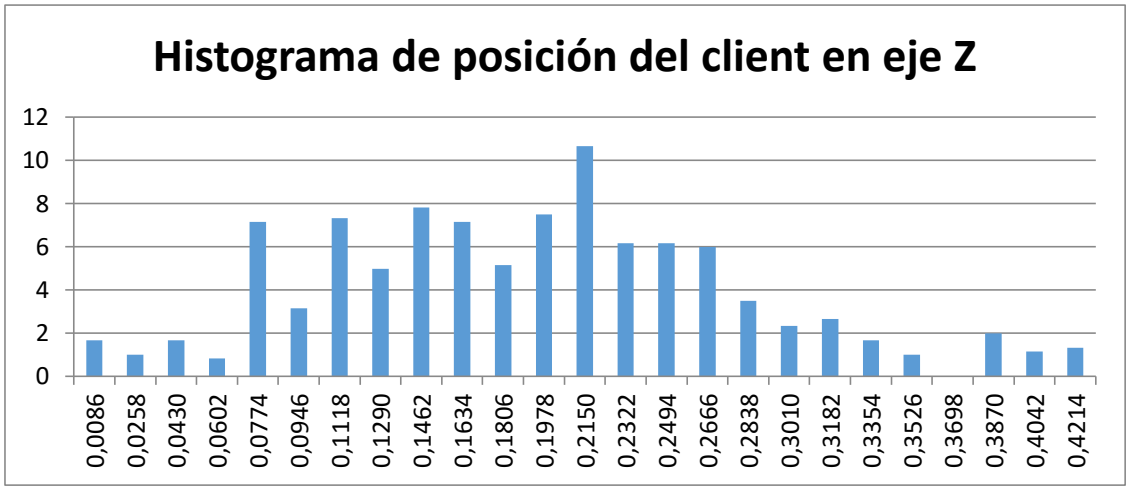

Histograma de posición del server en eje X
12
10
8
6
4
2
0
0,0080
0,0239
0,0398
0,0558
0,0717
0,0876
0,1035
0,1195
0,1354
0,1513
0,1673
0,1832
0,1991
0,2150
0,2310
0,2469
0,2628
0,2788
0,2947
0,3106
0,3265
0,3425
0,3584
0,3743
0,3903

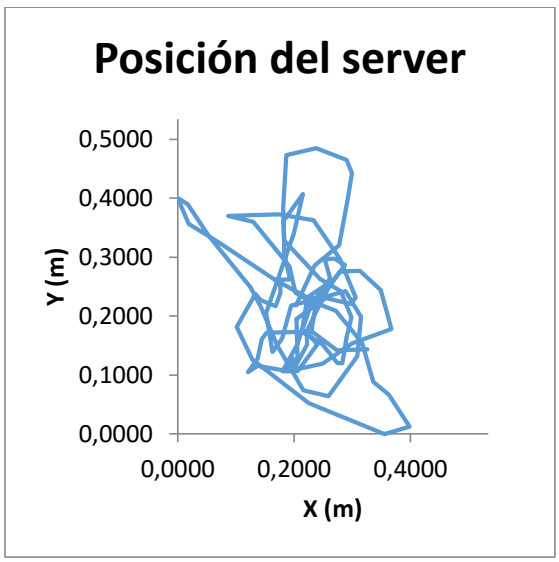

Histograma de posición del server en eje Y
20
15
10
5
0
0,0097
0,0291
0,0485
0,0680
0,0874
0,1068
0,1262
0,1456
0,1650
0,1845
0,2039
0,2233
0,2427
0,2621
0,2815
0,3010
0,3204
0,3398
0,3592
0,3786
0,3980
0,4175
0,4369
0,4563
0,4757

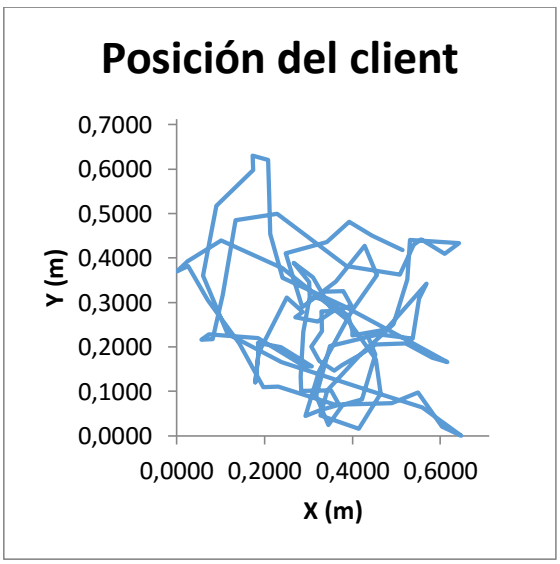

Histograma de posición del server en eje Z
10
8
6
4
2
0
0,0268
0,0804
0,1340
0,1876
0,2412
0,2948
0,3484
0,4020
0,4556
0,5092
0,5628
0,6164
0,6700
0,7236
0,7772
0,8308
0,8844
0,9380
0,9916
1,0452
1,0988
1,1524
1,2060
1,2596
1,3132

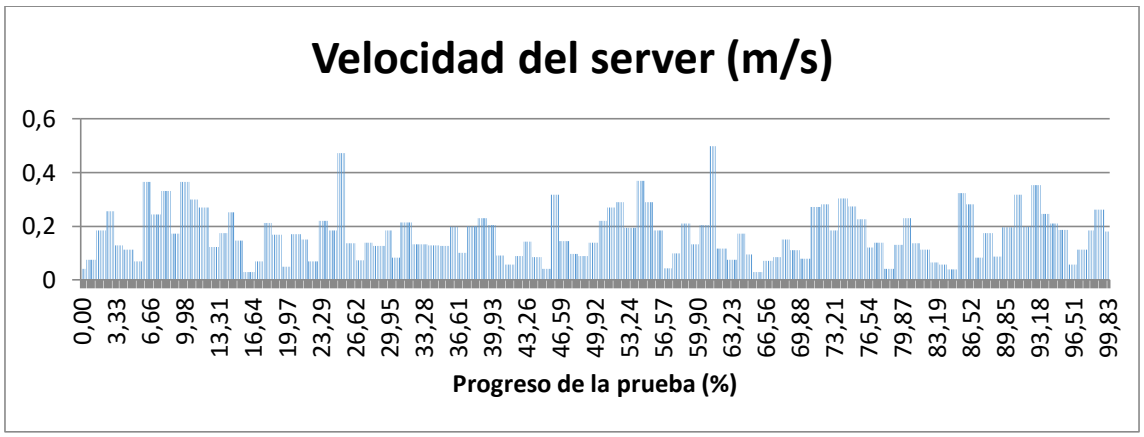

Histograma de posición del client en eje X
12
10
8
6
4
2
0
0,0129
0,0388
0,0647
0,0906
0,1165
0,1424
0,1682
0,1941
0,2200
0,2459
0,2718
0,2976
0,3235
0,3494
0,3753
0,4012
0,4271
0,4529
0,4788
0,5047
0,5306
0,5565
0,5823
0,6082
0,6341

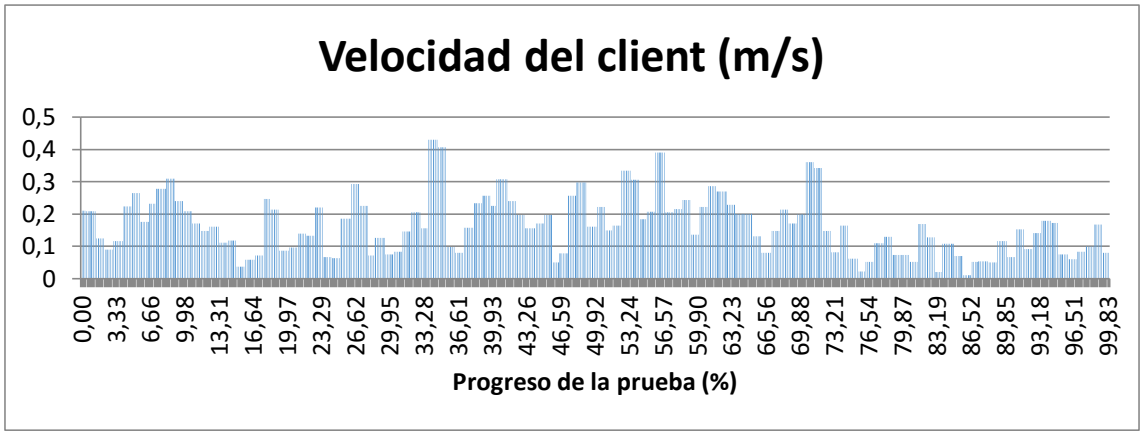

Histograma de posición del client en eje Y
12
10
8
6
4
2
0
0,0126
0,0378
0,0631
0,0883
0,1135
0,1387
0,1640
0,1892
0,2144
0,2396
0,2649
0,2901
0,3153
0,3405
0,3658
0,3910
0,4162
0,4414
0,4666
0,4919
0,5171
0,5423
0,5675
0,5928
0,6180

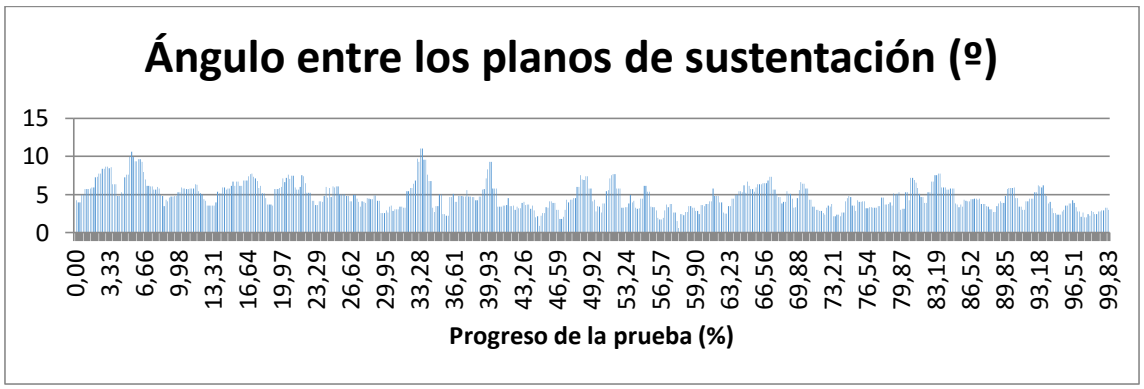

Histograma de posición del client en eje Z
12
10
8
6
4
2
0
0,0086
0,0258
0,0430
0,0602
0,0774
0,0946
0,1118
0,1290
0,1462
0,1634
0,1806
0,1978
0,2150
0,2322
0,2494
0,2666
0,2838
0,3010
0,3182
0,3354
0,3526
0,3698
0,3870
0,4042
0,4214

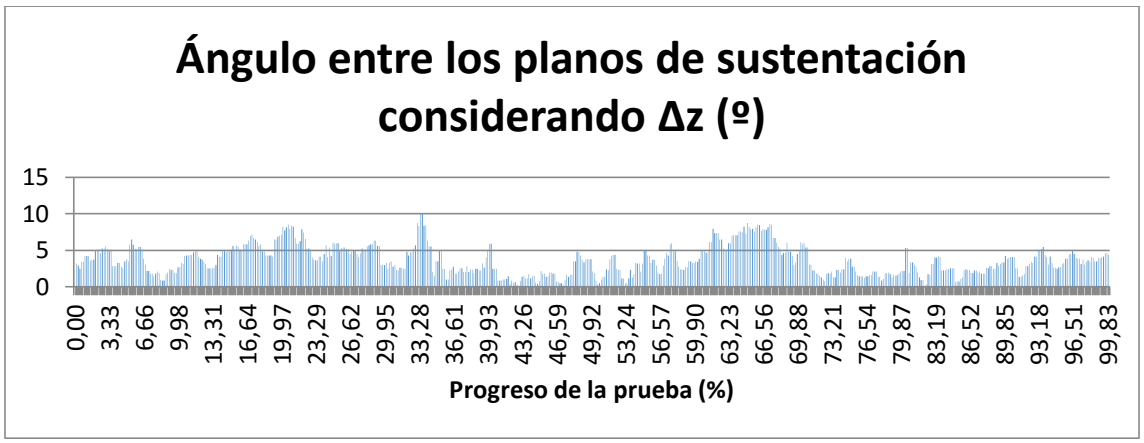

Posición del server
0,5000
0,4000
0,3000
0,2000
0,1000
0,0000
Y (m)
0,0000
0,2000
0,4000
X (m)

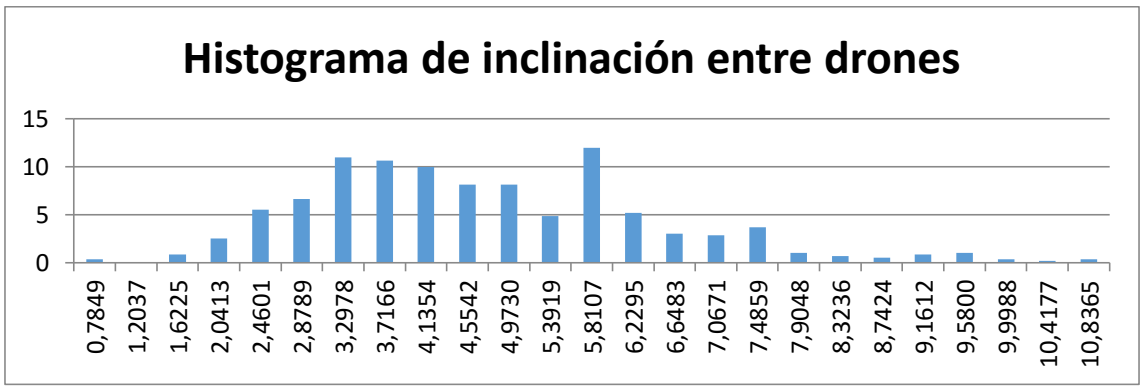

Posición del client
0,7000
0,6000
0,5000
0,4000
0,3000
0,2000
0,1000
0,0000
Y (m)
0,0000
0,2000
0,4000
0,6000
X (m)

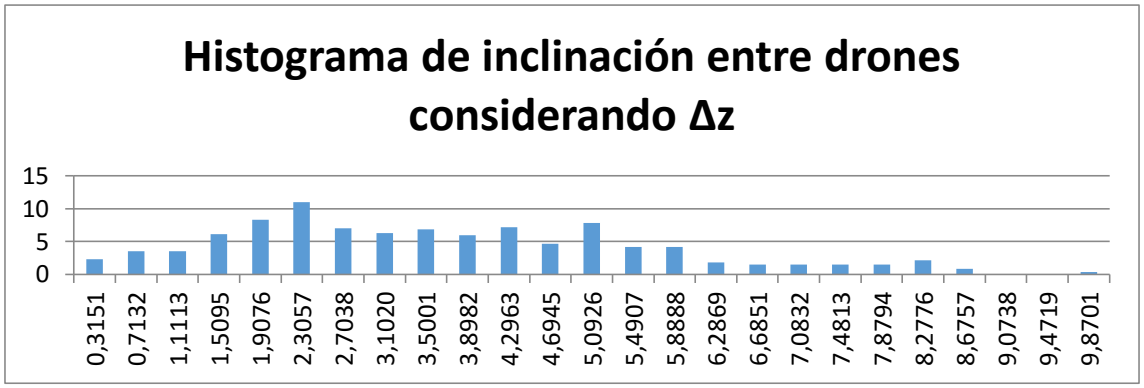

Velocidad del server (m/s)
0,6
0,4
0,2
0
0,00
3,33
6,66
9,98
13,31
16,64
19,97
23,29
26,62
29,95
33,28
36,61
39,93
43,26
46,59
49,92
53,24
56,57
59,90
63,23
66,56
69,88
73,21
76,54
79,87
83,19
86,52
89,85
93,18
96,51
99,83
Progreso de la prueba (%)


Velocidad del client (m/s)
0,5
0,4
0,3
0,2
0,1
0
0,00 3,33 6,66 9,98 13,31 16,64 19,97 23,29 26,62 29,95 33,28 36,61 39,93 43,26 46,59 49,92 53,24 56,57 59,90 63,23 66,56 69,88 73,21 76,54 79,87 83,19 86,52 89,85 93,18 96,51 99,83
Progreso de la prueba (%)


Ángulo entre los planos de sustentación (º)
15
10
5
0
0,00 3,33 6,66 9,98 13,31 16,64 19,97 23,29 26,62 29,95 33,28 36,61 39,93 43,26 46,59 49,92 53,24 56,57 59,90 63,23 66,56 69,88 73,21 76,54 79,87 83,19 86,52 89,85 93,18 96,51 99,83
Progreso de la prueba (%)


Ángulo entre los planos de sustentación considerando Δz (º)
15
10
5
0
0,00 3,33 6,66 9,98 13,31 16,64 19,97 23,29 26,62 29,95 33,28 36,61 39,93 43,26 46,59 49,92 53,24 56,57 59,90 63,23 66,56 69,88 73,21 76,54 79,87 83,19 86,52 89,85 93,18 96,51 99,83
Progreso de la prueba (%)

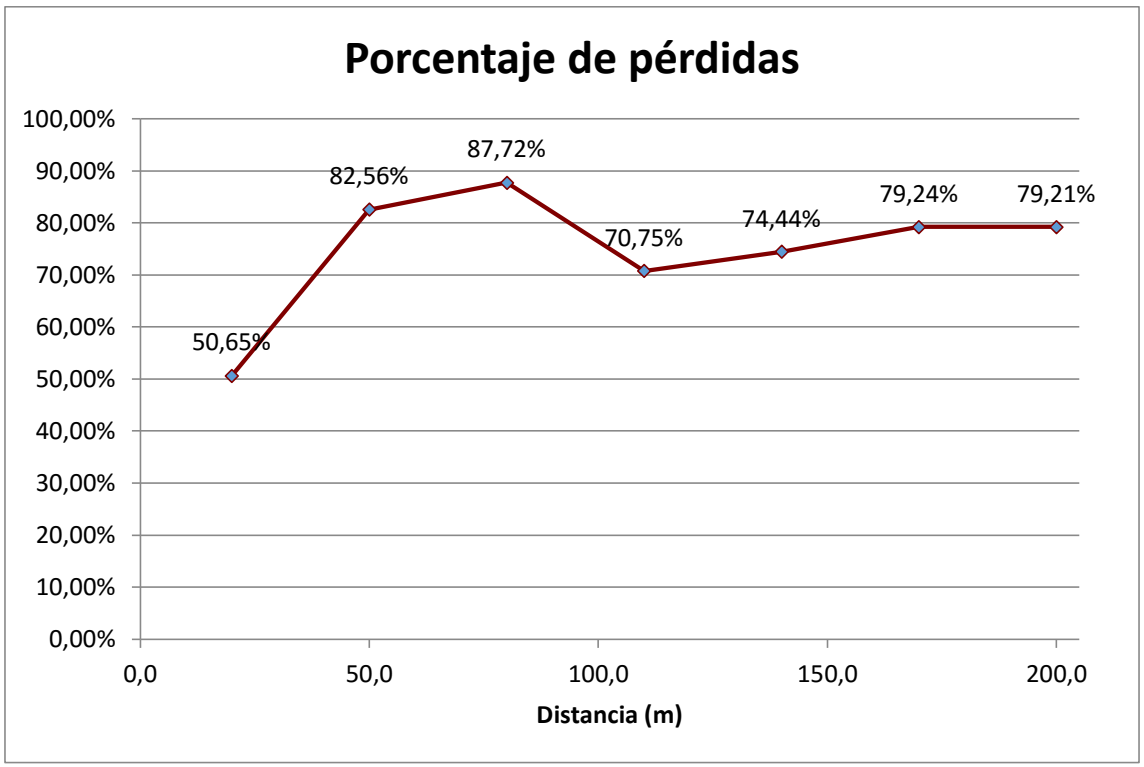

Histograma de inclinación entre drones
15
10
5
0
0,7849 1,2037 1,6225 2,0413 2,4601 2,8789 3,2978 3,7166 4,1354 4,5542 4,9730 5,3919 5,8107 6,2295 6,6483 7,0671 7,4859 7,9048 8,3236 8,7424 9,1612 9,5800 9,9988 10,4177 10,8365

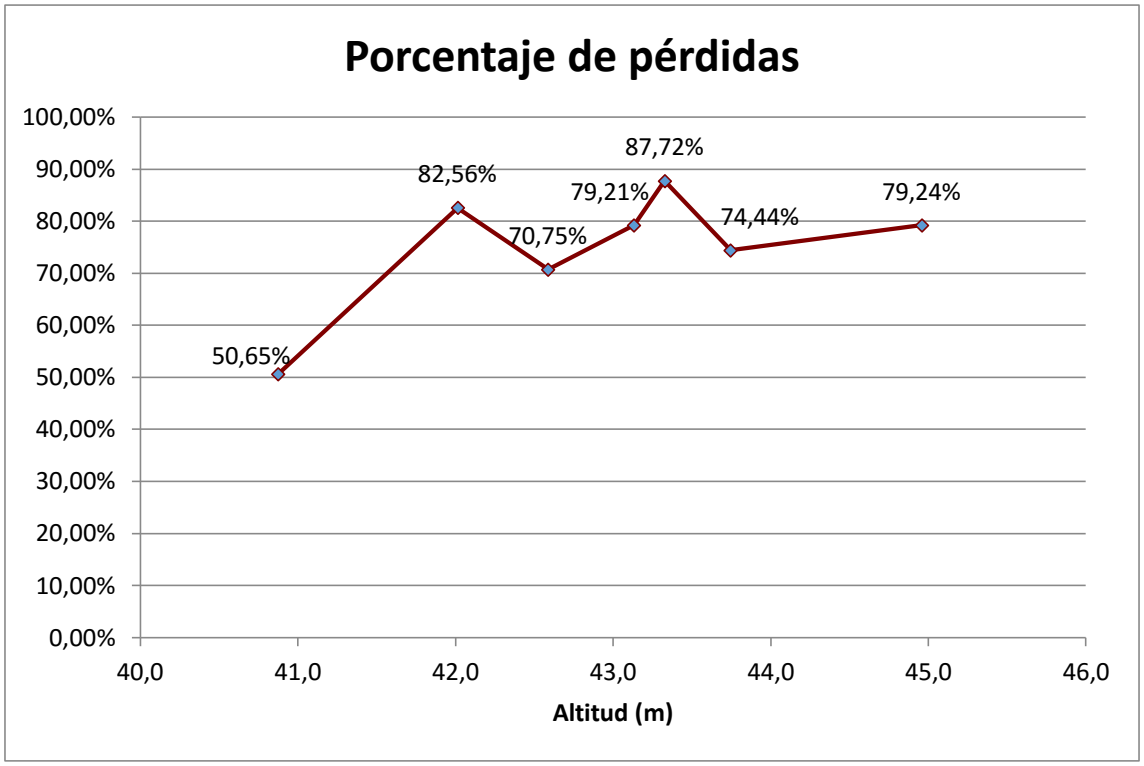


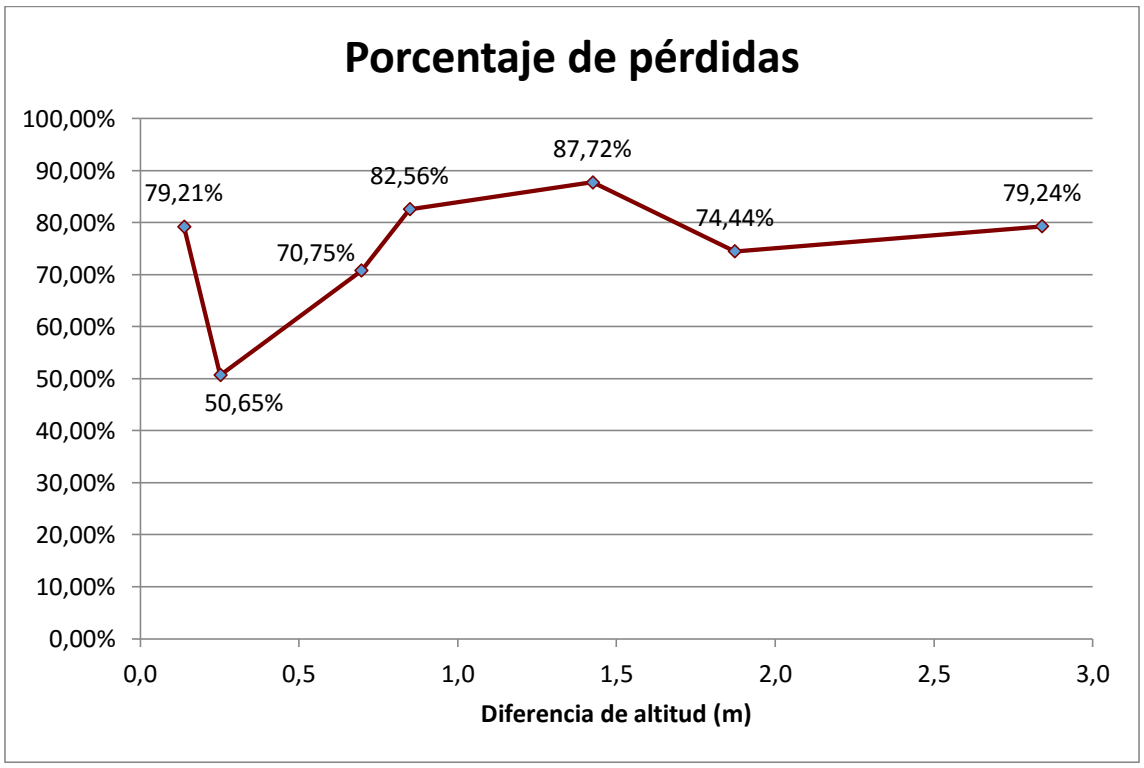
Ejemplo de análisis de una serie de tomas de datos

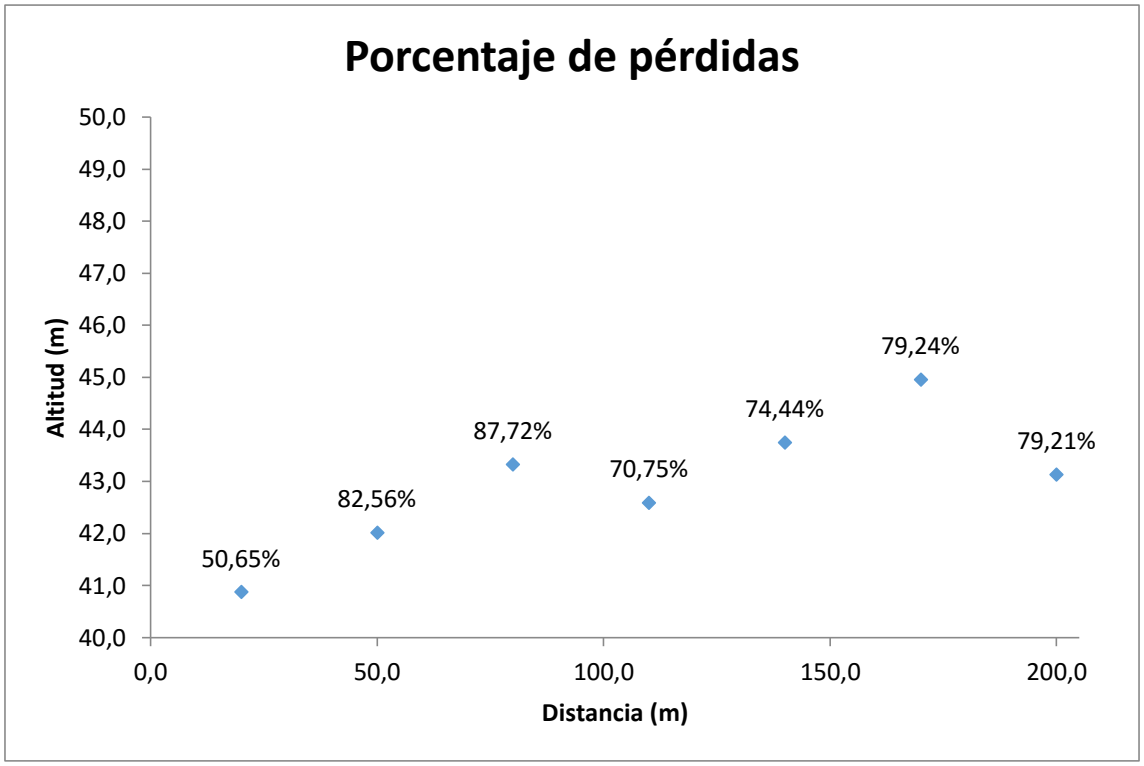
En este anexo se incluye, a modo de ejemplo, los gráficos obtenidos automáticamente mediante macros Excel para una serie de tomas de datos, a partir de un conjunto de ficheros de tomas de datos como el mostrado en el apartado anterior.

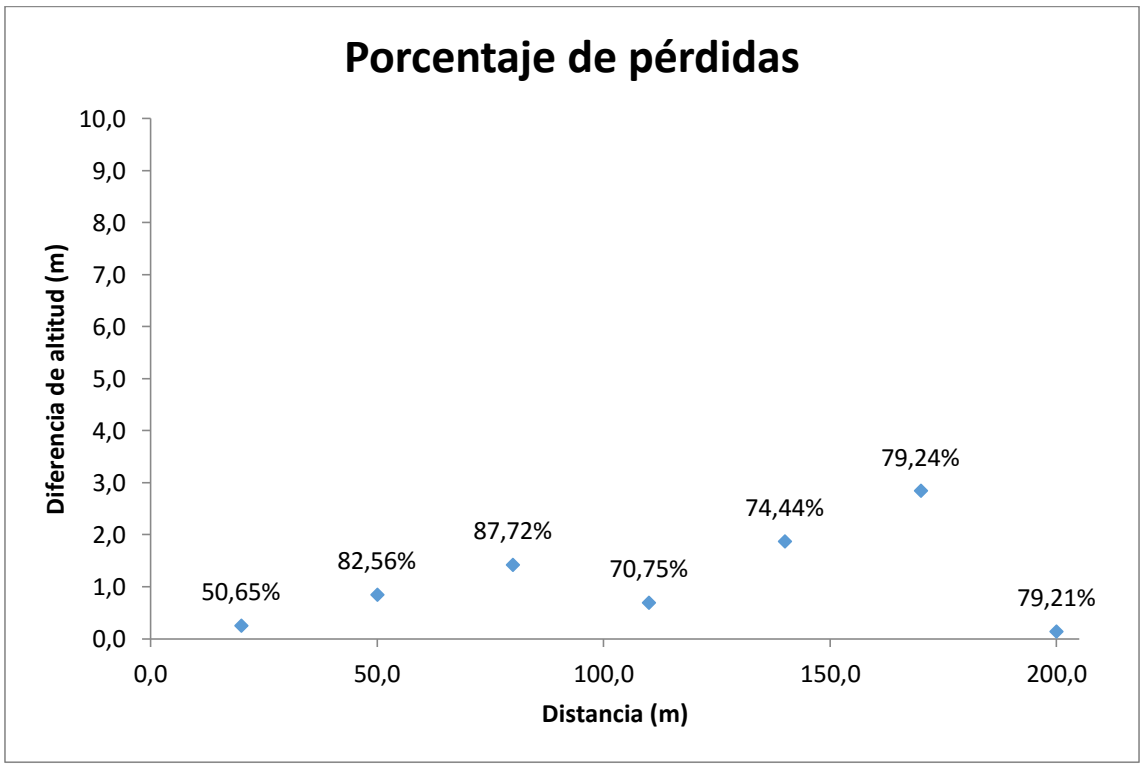
En los gráficos relativos a la posición X, Y y Z de los drones se observa cómo al acumular mediciones de distintas tomas de datos la distribución se aproxima gradualmente a una gausiana. Del mismo modo, el plano de planta de la posición de cada dron representa un trazado considerablemente circular, aun cuando los ficheros originales mostraban claramente una tendencia según un eje inclinado, como puede observarse en el apartado anterior.

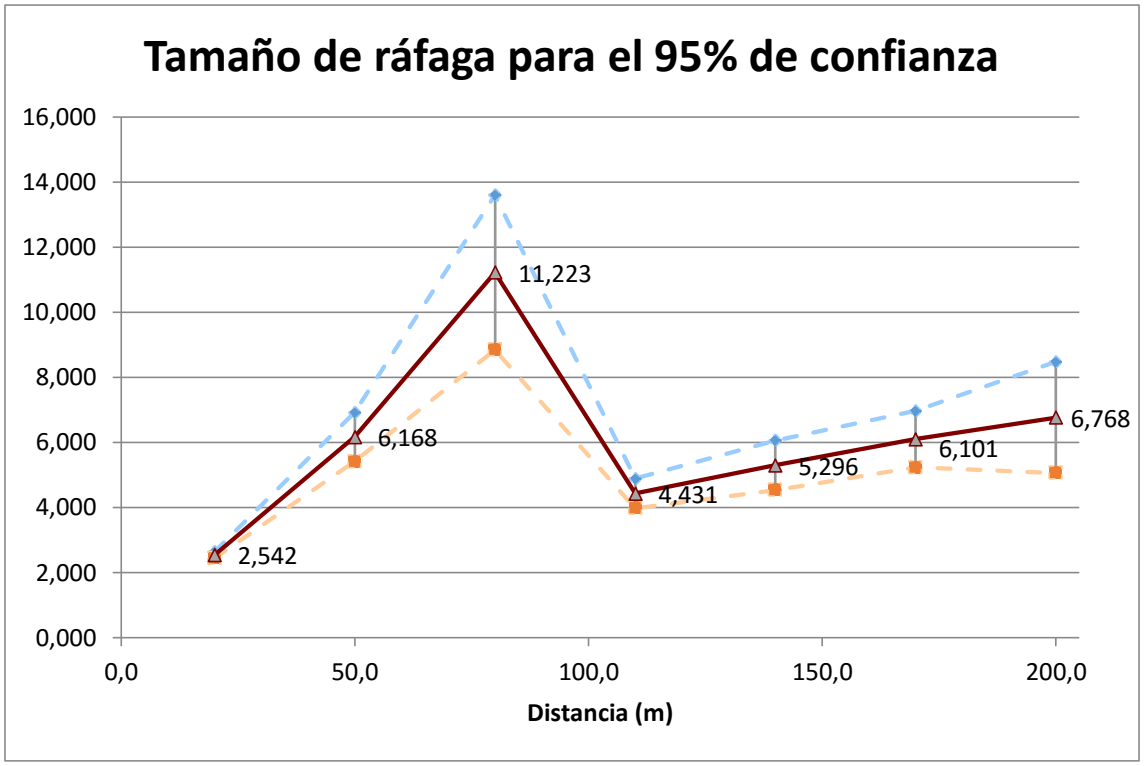


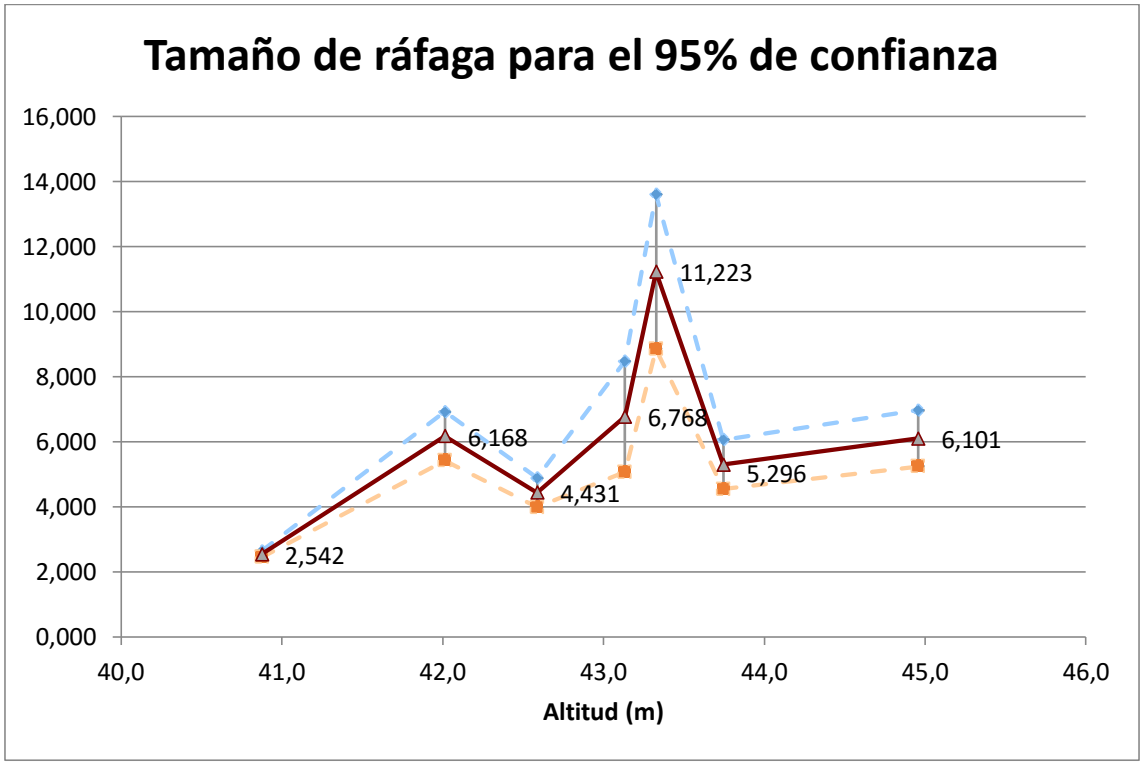

Porcentaje de pérdidas
100,00%
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
40,0
41,0
42,0
43,0
44,0
45,0
46,0
Altitud (m)
50,65%
82,56%
70,75%
79,21%
87,72%
74,44%
79,24%

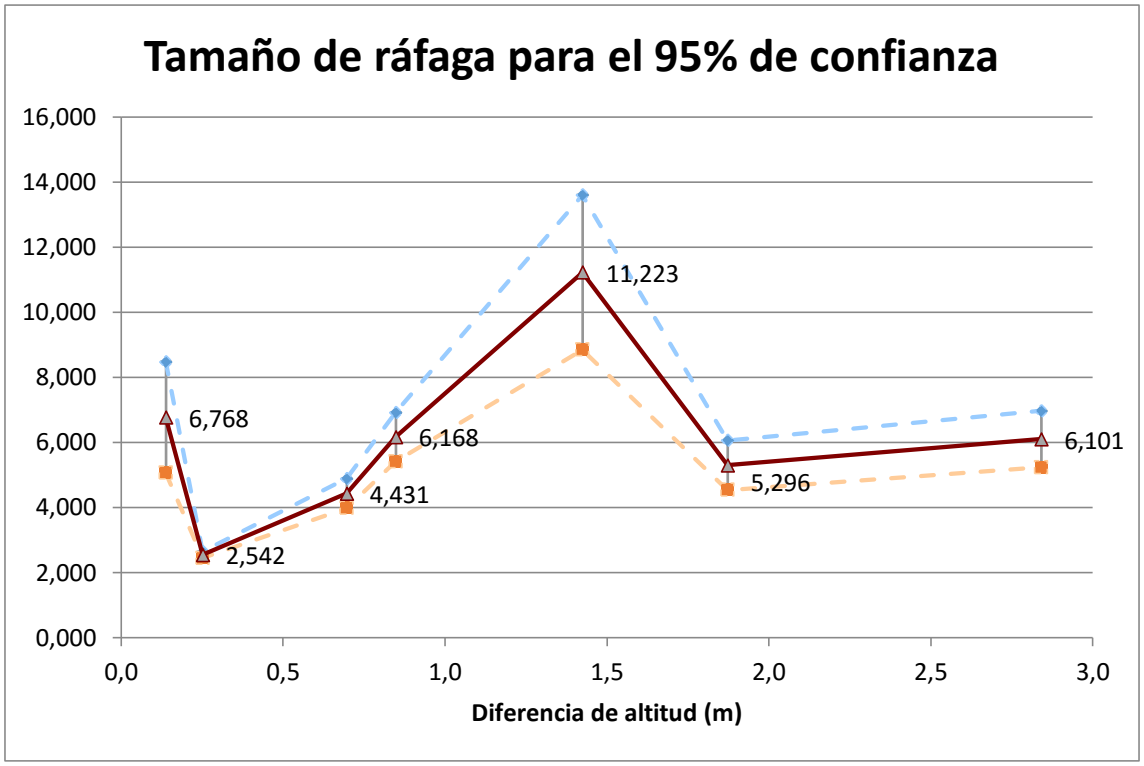

Porcentaje de pérdidas
100,00%
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
0,0
0,5
1,0
1,5
2,0
2,5
3,0
Diferencia de altitud (m)
79,21%
50,65%
70,75%
82,56%
87,72%
74,44%
79,24%

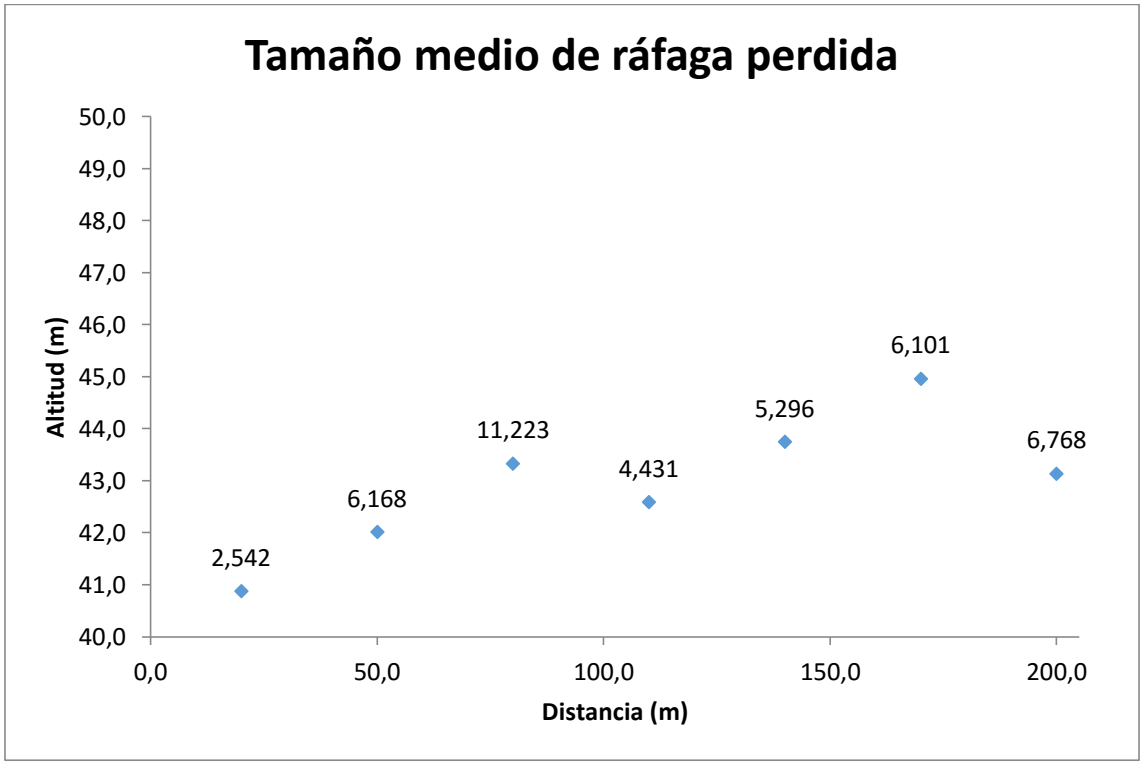

Porcentaje de pérdidas
Altitud (m)
50,0
49,0
48,0
47,0
46,0
45,0
44,0
43,0
42,0
41,0
40,0
0,0
50,0
100,0
150,0
200,0
Distancia (m)
50,65%
82,56%
87,72%
70,75%
74,44%
79,24%
79,21%

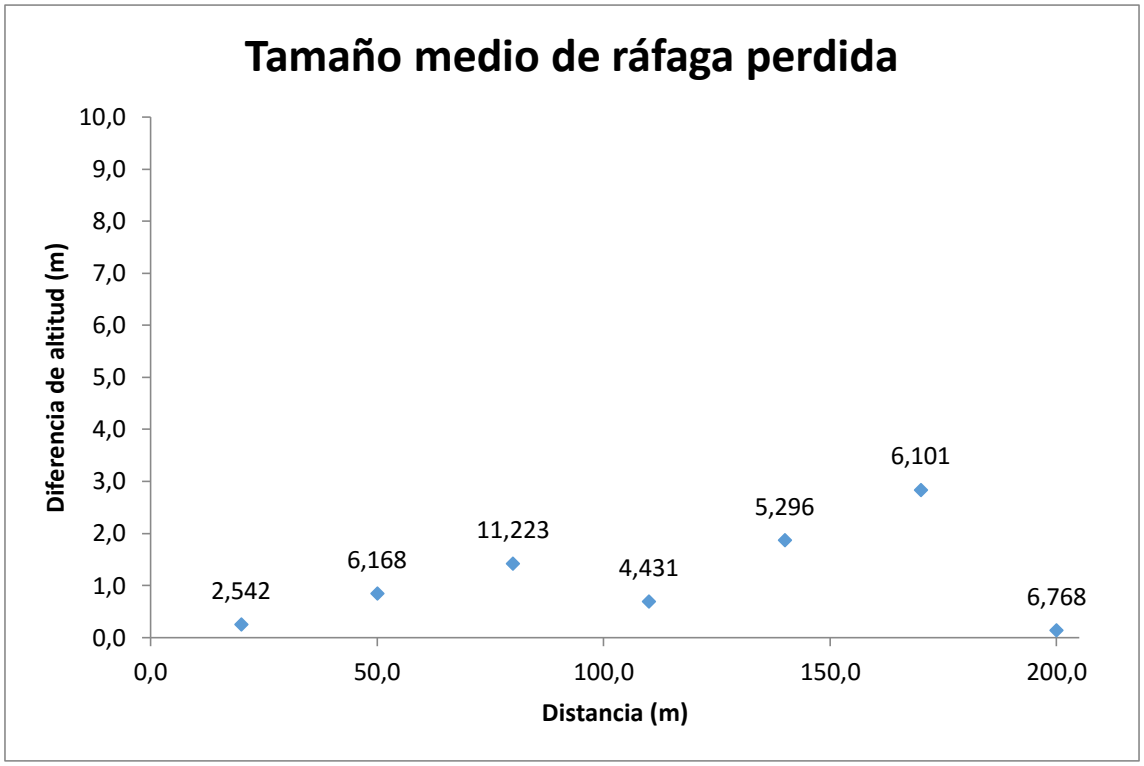

Porcentaje de pérdidas
Diferencia de altitud (m)
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
0,0
50,0
100,0
150,0
200,0
Distancia (m)
50,65%
82,56%
87,72%
70,75%
74,44%
79,24%
79,21%

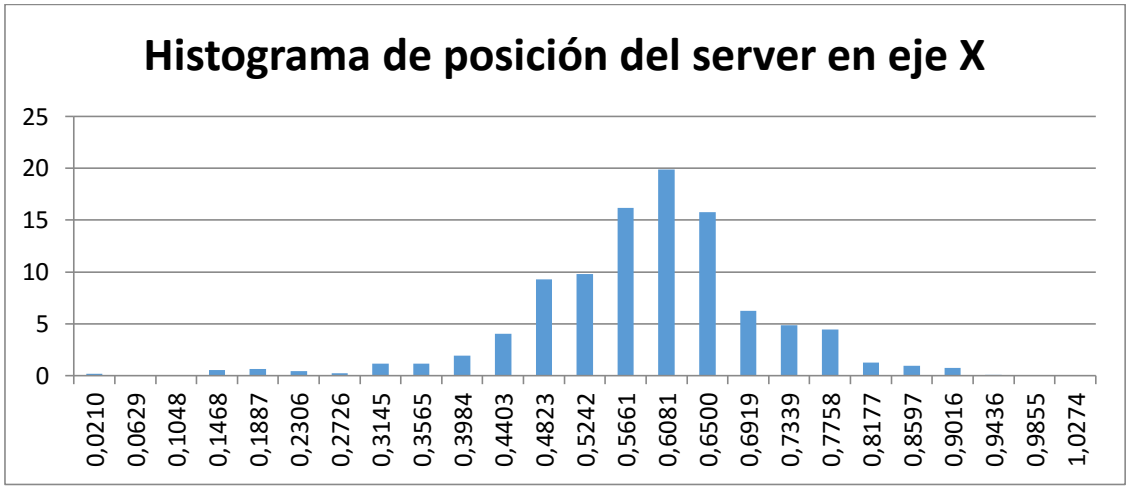

Tamaño de ráfaga para el 95% de confianza
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
0,0
50,0
100,0
150,0
200,0
Distancia (m)
11,223
6,168
2,542
4,431
5,296
6,101
6,768

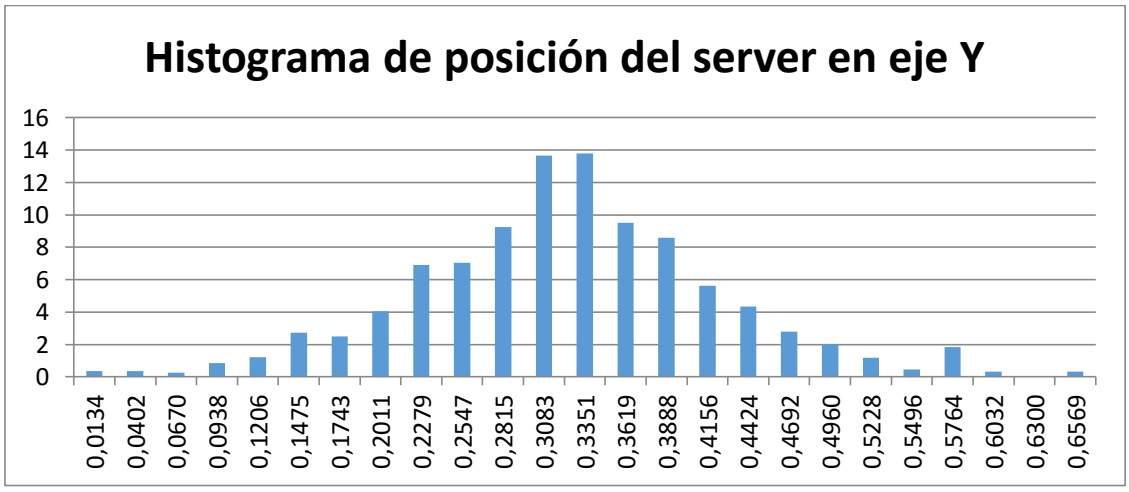

Tamaño de ráfaga para el 95% de confianza
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
40,0
41,0
42,0
43,0
44,0
45,0
46,0
Altitud (m)
11,223
6,168
6,768
6,101
5,296
4,431
2,542

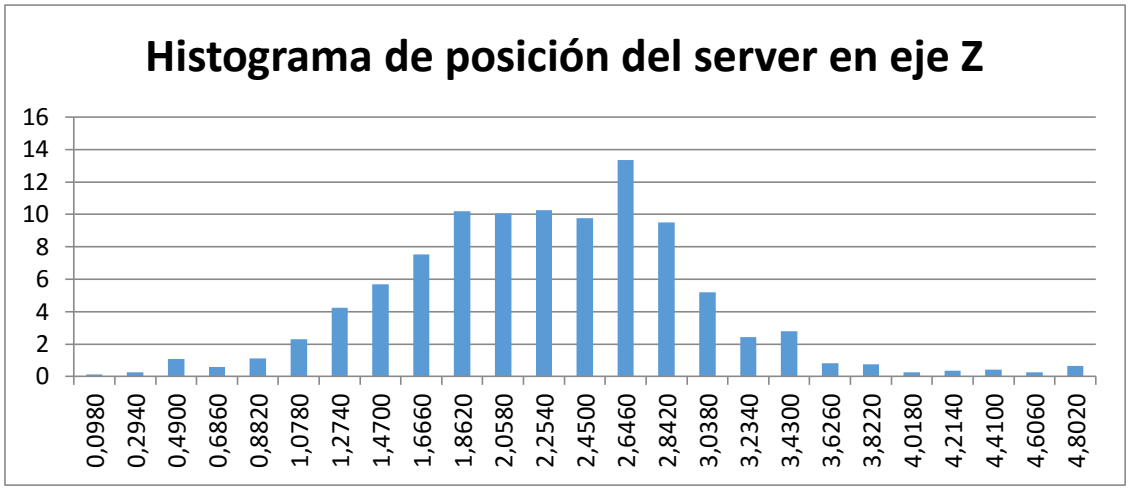

Tamaño de ráfaga para el 95% de confianza
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
0,0
0,5
1,0
1,5
2,0
2,5
3,0
Diferencia de altitud (m)
6,768
2,542
6,168
4,431
11,223
5,296
6,101

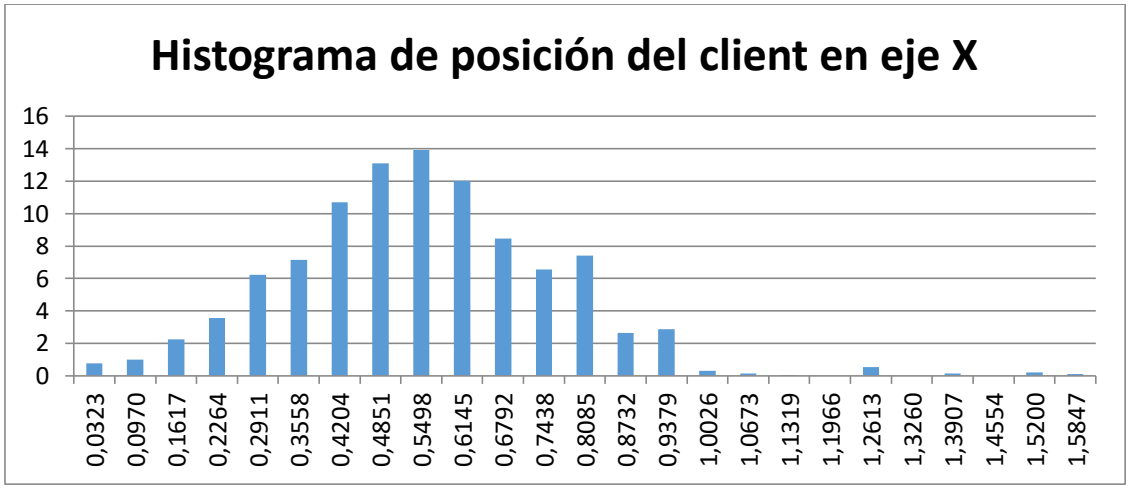

Tamaño medio de ráfaga perdida
Altitud (m)
50,0
49,0
48,0
47,0
46,0
45,0
44,0
43,0
42,0
41,0
40,0
0,0
50,0
100,0
150,0
200,0
Distancia (m)
2,542
6,168
11,223
4,431
5,296
6,101
6,768

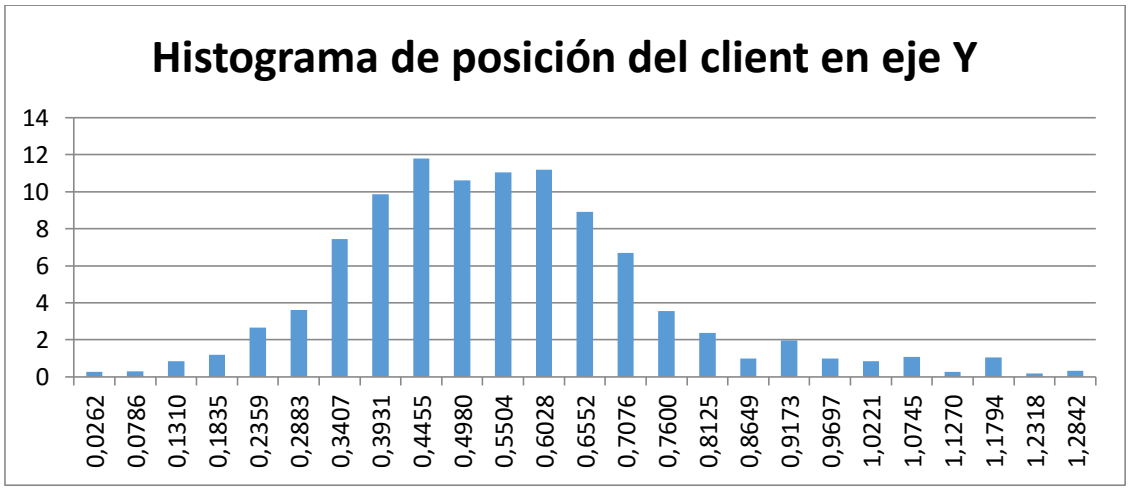

Tamaño medio de ráfaga perdida
Diferencia de altitud (m)
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
2,542
6,168
11,223
4,431
5,296
6,101
6,768
0,0
50,0
100,0
150,0
200,0
Distancia (m)

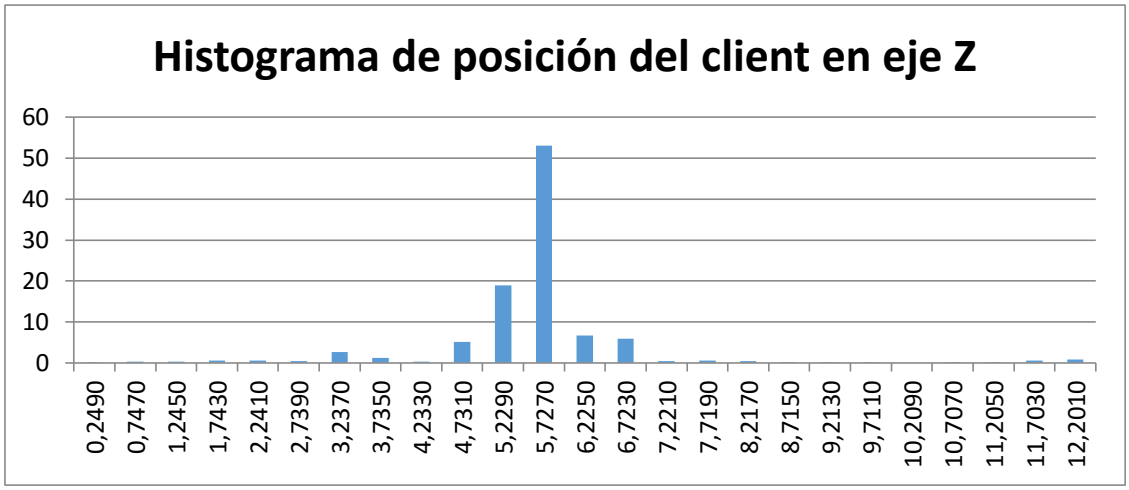

Histograma de posición del server en eje X
25
20
15
10
5
0
0,0210
0,0629
0,1048
0,1468
0,1887
0,2306
0,2726
0,3145
0,3565
0,3984
0,4403
0,4823
0,5242
0,5661
0,6081
0,6500
0,6919
0,7339
0,7758
0,8177
0,8597
0,9016
0,9436
0,9855
1,0274

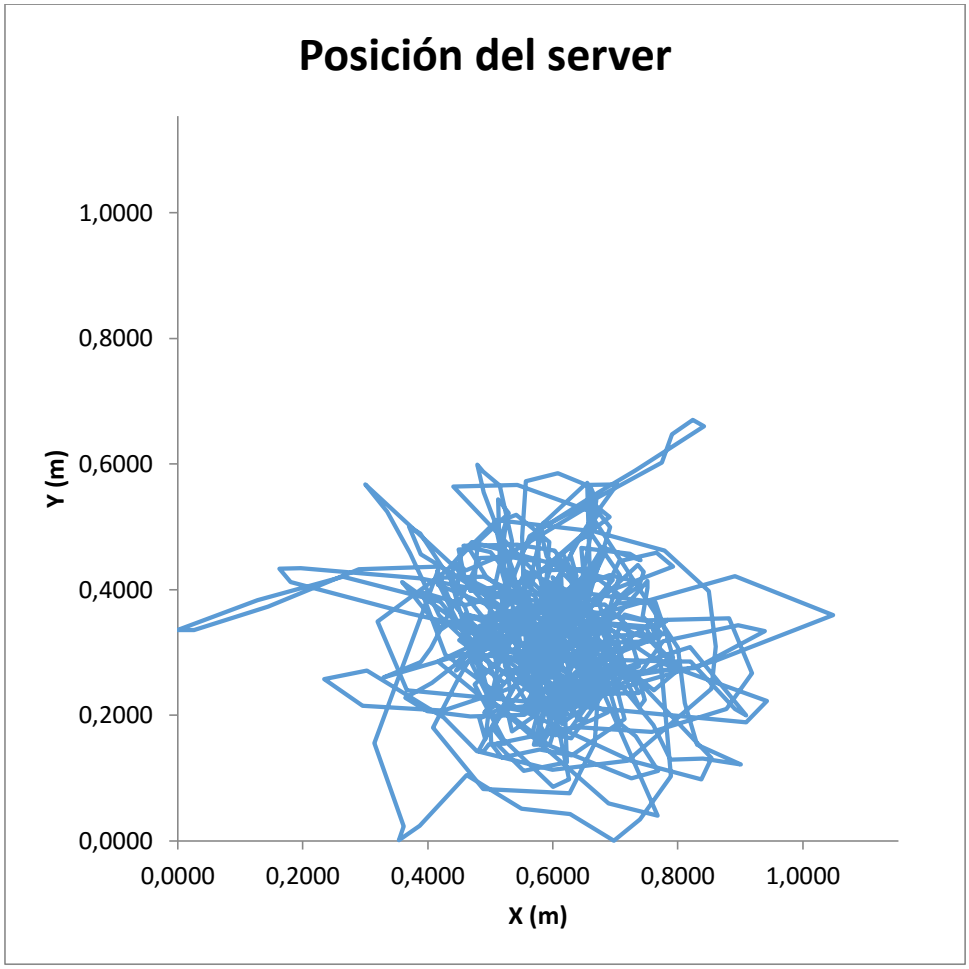

Histograma de posición del server en eje Y
16
14
12
10
8
6
4
2
0
0,0134
0,0402
0,0670
0,0938
0,1206
0,1475
0,1743
0,2011
0,2279
0,2547
0,2815
0,3083
0,3351
0,3619
0,3888
0,4156
0,4424
0,4692
0,4960
0,5228
0,5496
0,5764
0,6032
0,6300
0,6569

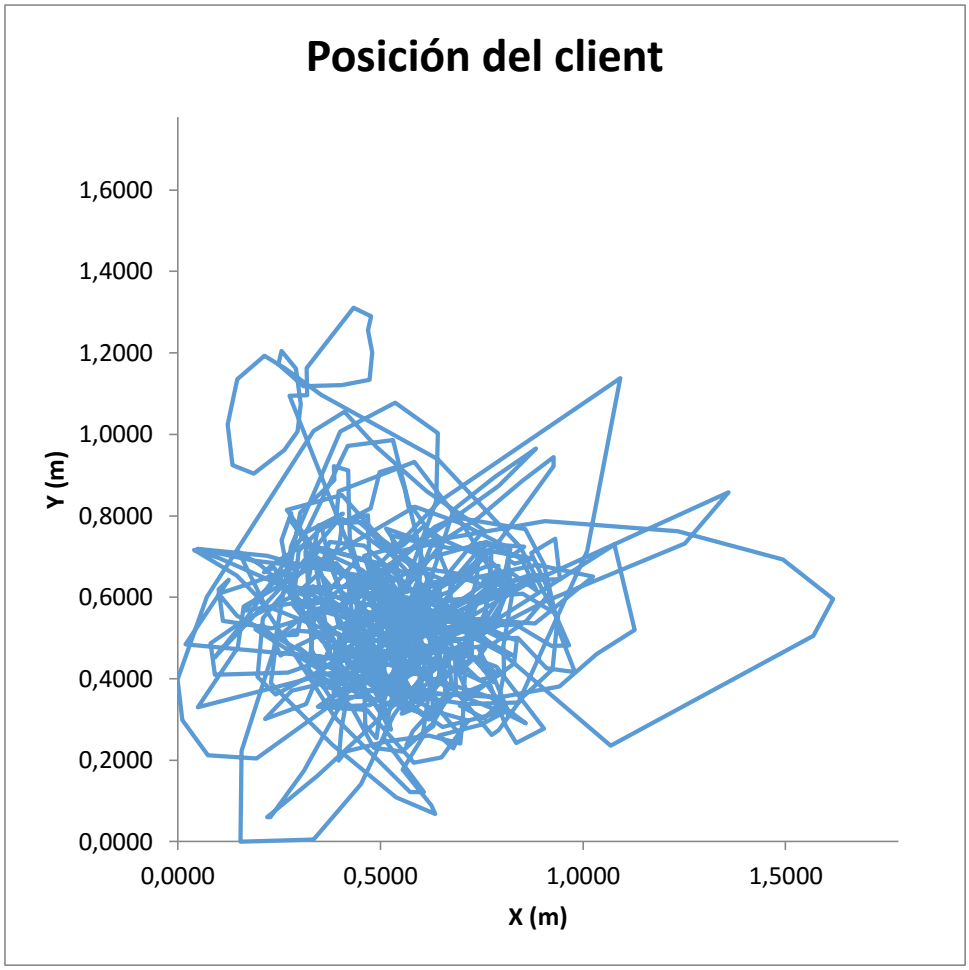

Histograma de posición del server en eje Z
16
14
12
10
8
6
4
2
0
0,0980
0,2940
0,4900
0,6860
0,8820
1,0780
1,2740
1,4700
1,6660
1,8620
2,0580
2,2540
2,4500
2,6460
2,8420
3,0380
3,2340
3,4300
3,6260
3,8220
4,0180
4,2140
4,4100
4,6060
4,8020


Histograma de posición del client en eje X
16
14
12
10
8
6
4
2
0
0,0323
0,0970
0,1617
0,2264
0,2911
0,3558
0,4204
0,4851
0,5498
0,6145
0,6792
0,7438
0,8085
0,8732
0,9379
1,0026
1,0673
1,1319
1,1966
1,2613
1,3260
1,3907
1,4554
1,5200
1,5847


Histograma de posición del client en eje Y
14
12
10
8
6
4
2
0
0,0262
0,0786
0,1310
0,1835
0,2359
0,2883
0,3407
0,3931
0,4455
0,4980
0,5504
0,6028
0,6552
0,7076
0,7600
0,8125
0,8649
0,9173
0,9697
1,0221
1,0745
1,1270
1,1794
1,2318
1,2842


Histograma de posición del client en eje Z
60
50
40
30
20
10
0
0,2490
0,7470
1,2450
1,7430
2,2410
2,7390
3,2370
3,7350
4,2330
4,7310
5,2290
5,7270
6,2250
6,7230
7,2210
7,7190
8,2170
8,7150
9,2130
9,7110
10,2090
10,7070
11,2050
11,7030
12,2010


Posición del server
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
Y (m)
0,0000
0,2000
0,4000
0,6000
0,8000
1,0000
X (m)


Posición del client
1,6000
1,4000
1,2000
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
Y (m)
0,0000
0,5000
1,0000
1,5000
X (m)