

Máster Universitario en Robótica y Automatización
(2021 / 2022)

Trabajo Fin de Máster

“Mapeado y evitación de obstáculos para la navegación autónoma de un dron”

David Meira Pliego

Tutor

David Martín Gómez

Leganés, 6 julio 2022



Esta obra se encuentra sujeta a la licencia Creative Commons
Reconocimiento – No Comercial – Sin Obra Derivada

RESUMEN

Este Trabajo de Fin de Master (TFM) tiene como objetivo general el estudio y desarrollo del vuelo autónomo de drones. Teniendo en cuenta que se conoce el mapa global donde se va a realizar el vuelo del vehículo, se busca poder evitar obstáculos inesperados en tiempo real mediante un mapeado local de nube de puntos. Dicho estudio está centrado en una parte simulada, en la que se estudia el comportamiento de todos los componentes que forman parte del vehículo, y una parte real, en la que se pone a prueba el correcto funcionamiento obtenido en la fase de simulación.

Con el fin de alcanzar este objetivo de la forma más eficiente, se busca minimizar los riesgos, a la hora de evitar obstáculos, y la distancia de la trayectoria del vehículo aéreo, optimizando el tiempo de vuelo y la energía utilizada.

Se ha hecho uso de un dron multirrotor cuadricóptero de fabricación propia incorporado con el piloto automático de código abierto PX4, un ecosistema que proporciona soporte de hardware y software. Además, se ha hecho uso del láser lidar Benewake CE30-C, para conseguir una representación del entorno mediante una nube de puntos y de este modo obtener un mapa de ocupación por medio de OctoMap.

De esta forma y con vistas al futuro se intenta consolidar la idea del uso de los drones de forma segura en el día a día, bien sea desde monitorizar incendios forestales de gran alcance a la videovigilancia aérea de una zona concreta o a la entrega de paquetería en entornos urbanos.

Palabras clave: UAV; dron; obstáculos; PX4; lidar; navegación autónoma.

ABSTRACT

The aim of this Master's Thesis is the study and development of autonomous drone flight. Taking into account that the global map where the flight is going to be performed is known, the goal is to avoid unexpected obstacles in real time by means of a local point cloud mapping. This study is focused on a simulated part, in which the behavior of all the components that are part of the vehicle is studied, and a real part, in which the correct operation obtained in the simulation phase is tested.

In order to achieve this objective in the most efficient way, the aim is to minimize the risks when avoiding obstacles and the distance of the aerial vehicle trajectory by optimizing the flight time and the energy used.

An in-house manufactured quadcopter multicopter drone incorporated with the open source PX4 autopilot, an ecosystem that provides hardware and software support, has been used. In addition, use has been made of the Benewake CE30-C laser lidar, in order to achieve a representation of the environment by means of a point cloud and thus obtain an occupancy map by means of OctoMap.

In this way and with a view to the future, the idea of the safe use of drones in day-to-day life is being consolidated, from monitoring large-scale forest fires to the aerial video surveillance of a specific area or the delivery of parcels in urban environments.

Key words: UAV; dron; obstacle; PX4; lidar; autonomous navigation.

ÍNDICE GENERAL

1. INTRODUCCIÓN.....	1
1.1. Motivación del trabajo.....	1
1.2. Objetivos.....	2
2. ESTADO DEL ARTE	3
2.1. Vehículos aéreos actuales y sus aplicaciones.....	5
3. COMPONENTES DEL SISTEMA.....	9
3.1. Hardware	9
3.1.1. Estación de tierra.	9
3.1.2. Componentes del vehículo.....	10
3.2. Software.....	15
4. DESARROLLO DEL SISTEMA DE MAPEADO Y EVITACIÓN DE OBTÁCULOS	24
4.1. Diseño del sistema virtual.....	24
4.1.1. Representación del entorno en la simulación.	24
4.1.2. Modelos de dron en la simulación.....	43
4.2. Funcionamiento del código.	46
4.3. Puesta en marcha del vehículo aéreo.....	49
5. PRUEBAS Y RESULTADOS.	55
5.1. Fase simulada.	55
5.2. Fase real.....	59
6. CONCLUSIONES.....	64
6.1. Trabajos futuros.....	64
7. PRESUPUESTO Y MARCO REGULADOR.	67
7.1. Presupuesto.....	67
7.1.1. Mano de obra.....	67
7.1.2. Material.....	68
7.1.2. Presupuesto final.	68
7.2. Impacto socioeconómico.	69
7.3. Marco regulador.	70
BIBLIOGRAFÍA	71

ÍNDICE DE FIGURAS

Figura 1.1: Previsión en el mercado del dron 2020-2025. [1].....	2
Figura 2.1: Drones en la industria. [2].....	3
Figura 2.2: Arquitectura de drones. [16].	5
Figura 2.3: Ryan Firebee 1241. [17]	5
Figura 2.4: Predator C Avenger [18], Heron TP [19] y MQ-9B SkyGuardian. [20].	6
Figura 2.5: Matrice 300 RTK (DJI) [21] y Anafi Uusa (Parrot) [22].....	6
Figura 2.6: Conectores del modelo Pixhawk 2.1. [23]	7
Figura 2.7: Piloto automático U-Pilot. [24].....	8
Figura 2.8: Controlador Veronte 1X. [25].....	8
Figura 3.1: Simulador QGroundControl [27]	9
Figura 3.2: Estación de tierra del proyecto.....	9
Figura 3.3: Chasis Quadcopter S500. [28]	10
Figura 3.4: Pixhawk Cube Black. [29]	10
Figura 3.5: Modulo GPS en el dron.....	11
Figura 3.6: Puertos de la Jetson AGX Xavier.	12
Figura 3.7: Sensor lidar Benewake CE30-C. [30]	12
Figura 3.8: Batería Gens Ace 500.	12
Figura 3.9: Sistema regulador UBEC Duo.	13
Figura 3.10: ESC F55A Pro. [31].....	13
Figura 3.11: Grafico de la conexión en la batería.....	13
Figura 3.12: Motor del dron.	14
Figura 3.13: Dron utilizado para este proyecto.	15
Figura 3.14: Versión Linux en la Jetson a bordo del dron.	15
Figura 3.15: Representación Octree. [32].....	18
Figura 3.16: OctoMap obtenido en la simulación.	18
Figura 4.1: Página web para descargar mapas lidar a nivel nacional.	24
Figura 4.2: Recorte de la parcela deseada para extraer el mapa lidar.	25
Figura 4.3: Nubes de puntos (entorno ciudad).	25
Figura 4.4:Leyendo nubes de puntos en Matlab.....	26
Figura 4.5: Visualización de nubes de puntos en Matlab.	27
Figura 4.6: Generación de superficie a partir de nubes de punto.	28
Figura 4.7: Curvas de nivel de la superficie obtenida.	28
Figura 4.8: Generación de mallas a partir de nubes de punto.....	28
Figura 4.9: Perspectiva completa entorno de montaña (MeshLab).	30
Figura 4.10: Detalles del modelo de montaña.	31
Figura 4.11: Perspectiva completa entorno de ciudad (MeshLab).	32
Figura 4.12: Detalles del modelo ciudad.	33
Figura 4.13: Archivos model.config y model.sdf del mapa a representar.....	34
Figura 4.14: Desplazamiento en los ejes.	34
Figura 4.15: Archivo para dar textura al mapa.	35
Figura 4.16: Mapa sin textura. (Gazebo).....	35
Figura 4.17: Mapa texturizado. (Gazebo).....	37
Figura 4.18: Inicializar proceso en RenderDoc.	39
Figura 4.19: PID de GPU.	39

Figura 4.20: Inicializar la ventana con el PID correspondiente.	40
Figura 4.21: Vista aérea de la Universidad Carlos III en Google Maps.....	40
Figura 4.22: Captura obtenida en RenderDoc.	41
Figura 4.23: Importar modelo en Blender.	41
Figura 4.24: Modelo 3D obtenido en Blender.	42
Figura 4.25: Comparación en los modelos de dron reales [34] y simulados.....	43
Figura 4.26:Representación del lidar Benewake en el entorno simulado.....	43
Figura 4.27: Gráfico de los componentes creados para el dron simulado.	45
Figura 4.28: Cámara de profundidad en la simulación.....	46
Figura 4.29: Camino más corto.	47
Figura 4.30: Camino con menos riesgo.	47
Figura 4.31: Camino óptimo.....	48
Figura 4.32: Grafo de comunicaciones.....	49
Figura 4.33: Sentido de giro de las hélices.	49
Figura 4.34: Configuración de la batería.	50
Figura 4.35: Calibración Pixhawk.	51
Figura 4.36: Configuración IP lidar.....	52
Figura 4.37: Nube de puntos del sensor lidar.	52
Figura 4.38: Conexión Wifi.....	53
Figura 5.1: Ubicación del dron en la simulación.....	55
Figura 5.2: Boceto explicativo del funcionamiento del código.....	56
Figura 5.3: Simulación dron con sensor lidar.....	57
Figura 5.4: Mapa local del dron con sensor lidar incorporado.	58
Figura 5.5: Mapa de ocupación del dron con cámara de profundidad.	59
Figura 5.6: Grafo órdenes del código.	60
Figura 5.7: Errores de conexión.	60
Figura 5.8: Ping entre GCS y el dron.	61
Figura 5.9: GPS U-Blox.	62
Figura 5.10: Secuencia de vuelo en una misión.	63

ÍNDICE DE TABLAS

Tabla 3.1: Significado del parpadeo de las luces del GPS.	11
Tabla 3.2: Peso de los componentes del dron.....	14
Tabla 3.3: Versión correspondiente Ubuntu/ROS.....	16
Tabla 7.1: Cantidad de horas totales del proyecto.....	67
Tabla 7.2: Coste de mano de obra.	67
Tabla 7.3: Coste total de los componentes del proyecto.	68

ANEXO GLOSARIO

AESA	Agencia Estatal de Seguridad Aérea.
AR	Augmented Reality
CNIG	Centro Nacional de Información Geográfica.
CNN	Convolutional Neural Networks.
CPU	Central Processing Unit.
DJI	Dà-Jiang Innovations.
EKF	Extended Kalman filter
FAA	Federal Aviation Administration
FOV	Field of view.
GCS	Ground Control Station.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
HDMI	High-Definition Multimedia Interface.
LIDAR	Light Detection and Ranging.
LSI	Laboratorio de Sistemas Inteligentes
LTS	Long Term Support.
MAVLink	Micro Air Vehicle Link.
PIB	Producto Interior Bruto .
RD	Reglamento Delegado.
RE	Reglamento de Ejecución.
RNN	Recurrent Neural Networks.
ROS	Robot Operating System.
SDK	Software Development Kit.
SITL	Software In The Loop.
TFM	Trabajo Fin de Máster.
TOF	Time Of Flight.

UAV	Unmanned Aerial Vehicle.
UTM	Universal Transverse Mercator
UC3M	Universidad Carlos III de Madrid.
UE	Unión Europea.
USB	Universal Serial Bus.
WiFi	Wireless Fidelity.
3DR	3D Robotics.

1. INTRODUCCIÓN.

En este trabajo se da una respuesta a la necesidad de mejorar el sector aéreo de drones. Se aporta un método en el que se estudia y desarrolla el vuelo autónomo de un vehículo aéreo no tripulado capaz de esquivar obstáculos en tiempo real, creando su propio mapa local mediante el uso de un sensor lidar. Se busca dar una solución más económica a los vehículos aéreos que hay actualmente en el mercado, abriendo el abanico de posibilidades de cara a futuras implementaciones en los diferentes sectores.

Dicho estudio busca promover el uso de los drones en labores cotidianas y que las personas empiecen a familiarizarse con este medio de transporte que ya es una realidad y ha llegado para quedarse. Se busca dar una alternativa a métodos más convencionales aportando soluciones innovadoras y minimizar riesgos.

1.1. Motivación del trabajo.

En los últimos años, la industria de los UAV (Unmanned Aerial Vehicle) ha sufrido un extraordinario proceso de evolución, tanto en el mercado global como en el mercado nacional. El número de aplicaciones en este sector muestra un tremendo crecimiento a medio plazo gracias a la continua innovación y al desarrollo tecnológico de los tiempos actuales.

Debido a este continuo desarrollo, cada día aparecen nuevas aplicaciones en los vehículos aéreos que permiten dar un apoyo a las soluciones tradicionales. Los drones se postulan como una herramienta versátil e ideal para la mayoría de los sectores.

En la búsqueda por implementar la navegación autónoma de drones en los diferentes sectores, se han encontrado numerosos casos en los que ya se están utilizando, tales como: monitorizar zonas de seguridad, control de incendios, búsqueda de personas en catástrofes medioambientales... Es por ello por lo que en este trabajo se intenta dar mayor accesibilidad mediante el uso de un dron de fabricación propia con componentes de bajo coste.

Como se puede observar en la Fig.1.1, el mercado global de los UAV está en constante crecimiento; sin embargo, este crecimiento es más acentuado en los continentes con países desarrollados como Europa, América del Norte y Asia. Quiero destacar que, en el caso del continente asiático, es donde se encuentra una mayor subida en el mercado ya que cuenta en su territorio con países como China, Corea del Sur y Japón que son los mayores productores y consumidores de drones a nivel mundial.

DRONE MARKET SIZE AND FORECAST 2020-2025

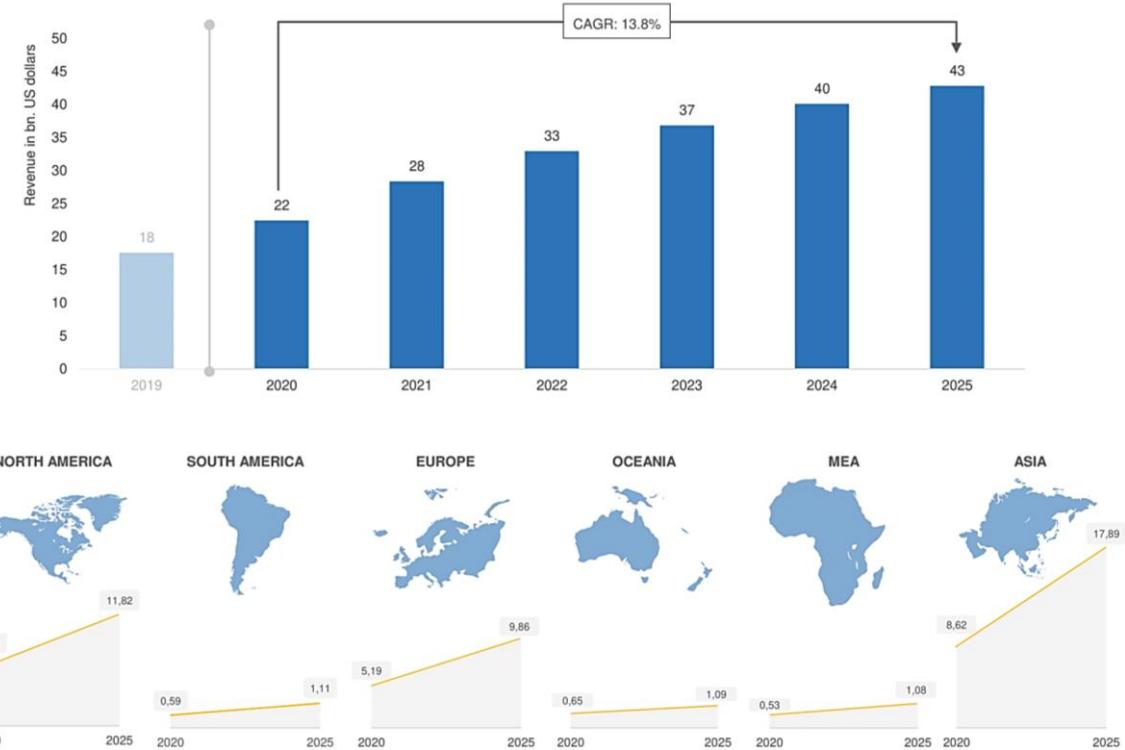


Figura 1.1: Previsión en el mercado del dron 2020-2025. [1]

1.2. Objetivos.

El objetivo general de este Trabajo de Fin de Master (TFM) es el desarrollo y estudio del vuelo autónomo de un vehículo aéreo no tripulado capaz de esquivar obstáculos en tiempo real, creando su propio mapa local mediante el uso de un sensor lidar. Se propone la construcción de un dron con piezas de bajo coste, versátil y capaz de aportar soluciones mediante su navegación autónoma. Al ser en código abierto su implementación es accesible para cualquier usuario y permite la edición en función del objetivo de la misión.

Para lograr este objetivo es necesario desarrollar una simulación en la que se ponga en práctica el código a bordo del dron. Para ello es necesario buscar una forma de representación de los mapas globales como entornos 3D y crear un modelo de dron 3D con características similares al dron real.

A la hora de realizar la navegación autónoma con evitación de obstáculos, se busca que su implementación sea sencilla, accesible y fácil de usar para el usuario. Se propone un método robusto y que se pueda implementar en diferentes modelos de UAV con la versatilidad de incorporar diferentes cámaras y sensores.

2. ESTADO DEL ARTE.

El término dron o UAV (Unmanned Aerial Vehicle) tienen su origen a mediados del siglo XIX, concretamente en el año 1849, cuando las tropas austrohúngaras, ante la impotencia de asediar la ciudad de Venecia, utilizaron un prototipo de globo no tripulado, cargado de explosivos, para bombardear la ciudad italiana.

Con el paso de los años, la idea de los vehículos aéreos no tripulados ha ido evolucionando hasta el punto de que hoy en día se puede encontrar una amplia gama de drones con diferentes usos y fines. En la actualidad, el sector militar es el principal consumidor de este tipo de vehículos aéreos (Fig. 2.1) abarcando el 70% de la industria, seguido por el sector a nivel consumidor 17% y, por último, el sector comercial 13%.

En España, con la aprobación de la Ley 18/2014, se permitió el uso de un tipo específico de dron para algunas operaciones civiles. En otras palabras, esta ley permite realizar operaciones alejadas de zonas urbanas, fuera del espacio aéreo y dentro del alcance visual del piloto.

Como causa a que la Administración Federal de Aviación (FAA) está entregando permisos para el uso comercial de los drones, se espera que en el año 2050 la flota de drones industriales en Estados Unidos y Europa sea de 1 millón de unidades y genere en torno a 50 billones de dólares (41.607 millones de euros) [3].

A la hora de hablar sobre vehículos aéreos no tripulados, hay que mencionar a los vehículos autónomos terrestres para entender la complejidad de esta forma de navegación. En la actualidad, el sector de vehículos terrestres con capacidad de navegar de forma autónoma está en continuo crecimiento. Un claro ejemplo, que está presente cada vez en más viviendas, es el robot aspirador Roomba (coloquialmente denominado Roomba debido a la marca del primer fabricante). Este tipo de vehículos son capaces de navegar de forma autónoma por un espacio cerrado gracias a los sensores que lo componen, pudiendo evitar obstáculos y creando sus propios mapas locales. Sin embargo, el sector que sufrirá el mayor cambio en los próximos años es el automovilístico. En España, el sector de la automoción representa el 10% del PIB y el 19% de las exportaciones [4]; el crecimiento de este sector supone un crecimiento en la economía de nuestro país y es por ello por lo que se busca un continuo desarrollo, reflejado en la futura posibilidad de una conducción completamente autónoma. Aunque hoy en día todavía está en una fase de desarrollo, se espera que en un futuro cercano los coches sean capaces de transportar mercancía y personas sin necesidad de un conductor al volante.

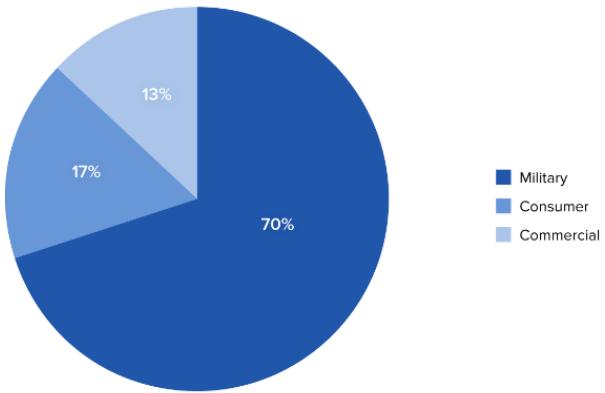


Figura 2.1: Drones en la industria. [2]

Dicho esto, los vehículos aéreos no tripulados tienen una dificultad añadida, operan en entornos de tres dimensiones con seis grados de libertad. Se necesita un control más preciso del vehículo para poder conocer su posición y orientación.

Teniendo en cuenta lo mostrado hasta este momento, se puede apreciar que los vehículos aéreos no tripulados tienen una gran capacidad para adaptarse a las aplicaciones civiles y militares. Por ejemplo, el uso de los drones para la videovigilancia aérea está en pleno auge, cuyo objetivo es la protección y la seguridad perimetral de grandes superficies. Para ello, los drones utilizan cámaras de alta resolución y cuentan con la posibilidad de programarse a ciertas horas permitiendo realizar tareas de forma automática. Algunos modelos cuentan con la opción de equiparse con visión nocturna, GPS, cámaras térmicas, reconocimiento facial... Hoy en día los drones se utilizan en un gran número de aplicaciones; por ejemplo, en el artículo [5], se puede observar cómo se hace uso de los drones para monitorizar el tráfico en las ciudades inteligentes y así reducir considerablemente la contaminación y los tiempos de espera en los atascos. Por otro lado, en [6], las tareas de prevención, vigilancia y extinción de incendios forestales hacen de los vehículos aéreos no tripulados un magnífico aliado, ya que ayudan a los guardias forestales a monitorizar áreas inaccesibles y controlar la propagación del fuego. Otro tema interesante es el visto en [7], donde mediante el uso de enjambres de drones, se vigila, cubre y rodea un objetivo con el fin de controlar el posible vertido de petróleo que un barco pueda derramar al mar. En un estudio de la Universidad de Cornell [8], se ha implementado el uso de la Inteligencia Artificial (AI) en los UAV para buscar y rescatar a personas en el mar Mediterráneo debido a la actual crisis migratoria en Europa.

Otro tema importante por mencionar, y que afecta a la navegación y posicionamiento del dron, es la ausencia de Sistemas Globales de Navegación por Satélite (GNSS). Para ello, una solución que se plantea es el uso de redes neuronales convencionales (CNN) como se puede observar en [9]. En esta investigación hacen uso de este tipo de redes para estimar la posición de un dron de carreras en relación con un punto determinado de la pista. Usando como apoyo la red Posenet, diseñada para la relocalización de la cámara en tiempo real, la cual se ha modificado con el objetivo de sacar un mayor rendimiento a los algoritmos de navegación autónoma y de este modo reducir la computación a bordo del vehículo.

Por otro lado, en el proyecto mostrado en [10] también hacen uso de las redes CNN, sin depender de la información suministrada por el GPS, para estimar la posición del dron por medio de la información recibida de una cámara a bordo. Por otra parte, usan varias rutas de navegación para un mayor entrenamiento de la red y crear una envoltura de navegación. Este tipo de navegación es ideal para viajes regulares por entornos conocidos. Siguiendo con el tema de las redes neuronales, en el artículo [11] se propone un método de estimación que utiliza una red neuronal recurrente (RNN) para permitir una estimación fiable de la posición y la velocidad de un dron en ausencia de señal de GPS. La RNN está entrenada en un conjunto de datos públicos recopilados con Pixhawk. Este piloto automático comercial de bajo costo registra las mediciones del sensor sin procesar (entradas de red) y las estimaciones EKF correspondientes (salidas de verdad del terreno).

Otra solución menos sofisticada a una posible pérdida de GPS es la que se presenta en [12]. Para ello, atan el dron con un cable a una estación de tierra y, según la tensión y posición tridimensional, estiman la localización mediante un filtro de Kalman con los datos de los motores eléctricos de a bordo del vehículo. Este tipo de soluciones hacen surgir otro tipo de problemas como es la dependencia de elementos externos al dron. Otro ejemplo sería el mostrado en [13] donde hacen uso de unos marcadores AR y una cámara para hacer un guiado autónomo en entornos cerrados y conocidos.

En la actualidad, el uso de los drones en las labores de búsqueda de supervivientes en zonas de catástrofe es muy demandado. En [14], se realiza un estudio sobre la detección de personas mediante una navegación basada en la reconstrucción del entorno con un sensor lidar para el mapeo global y una cámara para el mapeo local. Los entornos urbanos tienen la desventaja de sufrir vulnerabilidades con la conectividad del GPS. Es por ello por lo que en [15] abordan este problema de localización con el uso de una red convolucional entrenada por distritos y que toma como referencia edificios o monumentos característicos del propio lugar.

2.1. Vehículos aéreos actuales y sus aplicaciones.

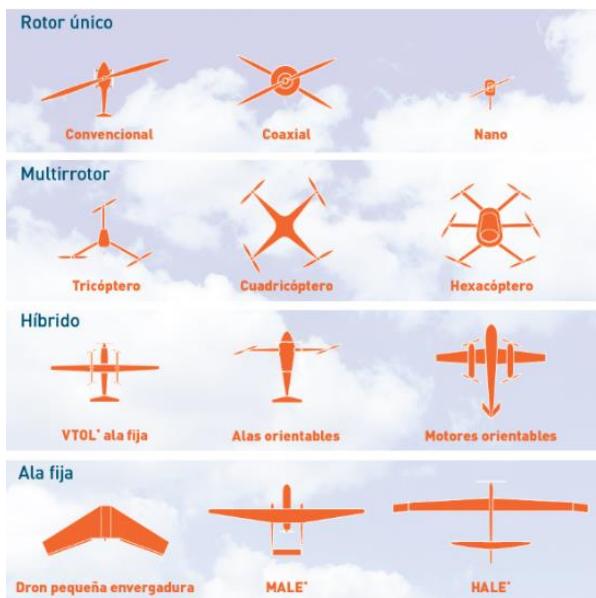


Figura 2.2: Arquitectura de drones. [16].

Una vez se tiene claro los diferentes tipos de drones que hay en el mercado, toca destacar cuáles son los más utilizados en cada sector.

Los drones en el sector militar han sufrido una evolución más destacable que en el sector civil. Esto se debe a la gran apuesta militar por estos vehículos, buscando minimizar riesgo en las misiones ya que estos vehículos aéreos no tripulados eliminan la posibilidad de que haya pérdidas humanas. El primer

A la hora de definir los diferentes tipos de drones según su arquitectura, se pueden distinguir dos grandes grupos: los de ala fija y los de ala rotatoria (existen modelos híbridos), Fig. 2.2. Por otro lado, según el grado de autonomía se dividen en: drones autónomos o drones por control remoto.

Esta amplia variedad de drones permite una mayor flexibilidad de prestaciones. De esta forma, los drones se manifiestan como herramientas muy versátiles capaces de adaptarse a diferentes sectores y situaciones.



Figura 2.3: Ryan Firebee 1241. [17]

UAV moderno utilizado en una guerra fue en 1973, de nombre Ryan Firebee 1241 Fig. 2.3, perteneciente al ejército israelí.

En la actualidad, el uso de drones ha revolucionado las tácticas militares en todo el mundo. Su gran capacidad de carga y su precisión permiten a los drones ser magníficos aliados en el campo de batalla. Los UAV de combate más populares hoy en día son el Predator C Avenger, Heron TP y MQ-9B SkyGuardian. Fig. 2.4.



Figura 2.4: Predator C Avenger [18], Heron TP [19] y MQ-9B SkyGuardian. [20].

Sin embargo, su uso no se restringe únicamente al combate. Otra de sus características principales es la autonomía de vuelo y la posibilidad de incorporar cámaras de gran calidad, lo cual hace que estos vehículos sean perfectos para misiones de reconocimiento o incluso para misiones señuelo en las que hacen de cebo para radares enemigos. Como se puede observar, la mayoría de los drones militares son de ala fija; esto se debe a que este tipo de vehículos tienen más carga útil, vuelan a alturas más elevadas y, sobre todo, tienen mucha mayor autonomía.

En cambio, el sector civil todavía está en pleno desarrollo. Su uso se extiende cada vez más y las industrias se está aprovechando de ello. Si bien es verdad que los drones militares suelen ser de ala fija, los drones civiles suelen ser de ala rotatoria. Esto es debido a que los drones de ala rotatoria no necesitan de un área de despegue, tienen mucha más estabilidad y su uso es más sencillo. Normalmente, este tipo de drones no suelen tener una carga útil muy elevada y, su amplia gama y bajo coste, permiten focalizar su uso para una tarea determinada; bien sea en temas de vigilancia perimetral, mantenimiento, competiciones de carreras, grabaciones en eventos...

Los drones civiles más utilizados suelen ser de la marca DJI o Parrot, Fig. 2.5. Destacan por su calidad, fiabilidad y uso sencillo. Si bien es cierto que hay drones más económicos en el mercado, estas empresas proporcionan garantías de calidad y un amplio abanico de posibilidades en su catálogo.



Figura 2.5: Matrice 300 RTK (DJI) [21] y Anafi Uusa (Parrot) [22].

Cuando se habla del término dron, hay que destacar los diferentes pilotos automáticos comerciales que existen en el mercado actual. Estos componentes, como el nombre bien indica, tienen la tarea de sustituir a un piloto a bordo del vehículo siendo capaces de controlar de forma autónoma el vuelo del vehículo. A continuación, se mostrarán unos cuantos ejemplos, haciendo hincapié en el controlador de vuelo Pixhawk que es el que se utiliza en este trabajo.

- **Pixhawk** : Es un proyecto basado en la creación de pilotos automáticos de bajo coste y gran calidad. El diseño está basado en un hardware libre e independiente que, junto con el sistema operativo NuttX, ejecutan PX4 o ArduPilot. Este sistema operativo destaca por ser en tiempo real, mostrar un riguroso criterio en los estándares técnicos y ser escalable a entornos de pequeño tamaño.

Algunos de los beneficios de estos controladores son: el soporte de software, la flexibilidad a la hora de conectar los diferentes tipos de hardware, la alta calidad del producto, la personalización de la forma física, las constantes actualizaciones del firmware...

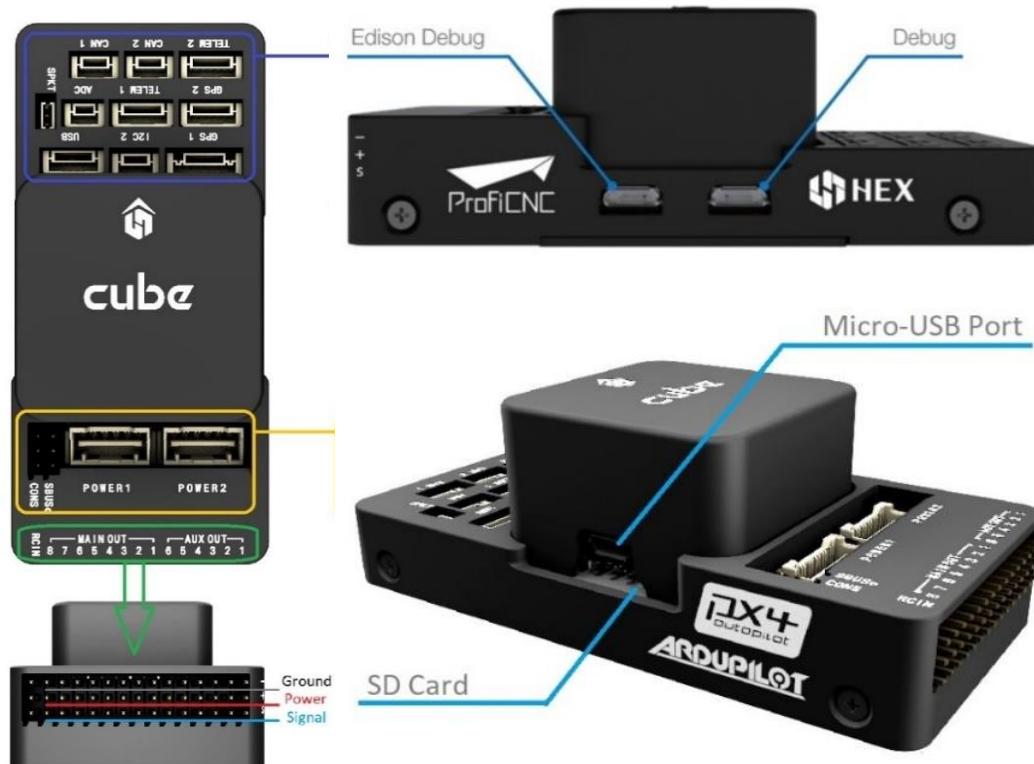


Figura 2.6: Conectores del modelo Pixhawk 2.1. [23]

En la Figura 2.6, se muestra el modelo de controlador que se va a utilizar en este trabajo; es el denominado Pixhawk 2.1 (actualmente conocido como Cube Black Flight Controller). Este controlador está diseñado para reducir el cableado gracias al acople de una placa de dominio específico. El sistema de energía admite una potencia máxima de 10 V y una corriente máxima de 10 A. En la Figura, también se puede apreciar todos los puertos de los que dispone este controlador.

- **U-Pilot:** Este hardware permite controlar de forma automática tanto aeronaves de ala rotatoria como fija. Cuenta con una doble CPU que permite cerrar el bucle de control a una alta velocidad (1 KHz) mientras maneja en paralelo las payload (cargas de pago). Por otro lado, opera con una frecuencia de 900 MHz (dependiendo de las restricciones) que permite alcanzar distancias de hasta 100 km. La mayor desventaja de este dispositivo es su alto coste, que puede llegar a alcanzar los 10.000€.



Figura 2.7: Piloto automático U-Pilot. [24]

- **Veronte Autopilot:** Son controladores miniaturizados de sistemas no tripulados, tanto aéreos como terrestres. Destaca por ser altamente configurable y por sus rutinas personalizables. Desde el despegue al aterrizaje, el control es totalmente autónomo. Los servidores de Veronte Cloud permiten tener un control online de los vehículos gracias a la integración del módulo M2M. Otro dato para tener en cuenta es que este controlador se puede obtener con triple redundancia para sistemas a prueba de fallos Failsafe. El equipo de Veronte ha desarrollado una interfaz de usuario intuitiva y personalizable para todos los públicos.



Figura 2.8: Controlador Veronte IX. [25].

3. COMPONENTES DEL SISTEMA.

Este apartado abarca los componentes que integran la parte de hardware y software del sistema utilizado para este trabajo.

3.1. Hardware.

En la parte del hardware, se muestran todos los elementos físicos que componen al dron y permiten que funcione de forma correcta.

3.1.1. Estación de tierra.

La estación de tierra o GCS (Ground Control Station) permite la comunicación del dron con el ordenador. Los GCS suelen ser proporcionados por el fabricante del vehículo o ser de código abierto. En este trabajo, al ser un dron de fabricación propia, se ha optado por utilizar un GCS de código abierto por lo que se ha hecho uso del programa QGroundControl. Este simulador de código abierto da soporte a vehículos que ejecutan PX4 o ArduPilot y permite obtener por pantalla la ruta de vuelo y la posición del vehículo; así como planificar misiones para vuelos autónomos. Destaca por su uso sencillo y da soporte a funciones más desarrolladas como las vistas en [26]. Este programa se obtiene descargándolo desde la propia página de QGroundControl.



Figura 3.1: Simulador QGroundControl [27]

En la Fig. 3.2. se puede observar una fotografía realizada durante una prueba en la universidad. Quiero destacar que el dron debe de estar en una zona despejada de edificios y con una condición meteorológica favorable para un correcto funcionamiento del GPS.



Figura 3.2: Estación de tierra del proyecto.

3.1.2. Componentes del vehículo.

En este trabajo se ha hecho uso de un dron de fabricación propia. A continuación, se procede a explicar detalladamente cada uno de los componentes que forman a este vehículo.

- **Chasis Quadcopter S500:** El primer componente esencial a describir es el chasis que conforma la estructura del dron. Este chasis, fabricado a partir de fibra de vidrio y nylon, destaca por su sencillez y bajo coste; sus dimensiones de 480mm de ancho (diagonal entre motor y motor) y sus 170 mm de altura, hacen que llegue a alcanzar un peso de 425 gr.



Figura 3.3: Chasis Quadcopter S500. [28]

- **Pixhawk Cube Black:** Mencionado con anterioridad en el apartado 1.2., en este trabajo se ha decidido utilizar el modelo Cube Black. Este controlador de vuelo es un piloto automático que ejecuta PX4 y tiene un peso de 500 gr. Las conexiones en el lado superior son en GPS1, TELE1, POWER1. Por otro lado, las conexiones de los pines son las mostradas en color verde en la Fig. 3.4.



Figura 3.4: Pixhawk Cube Black. [29]

- **Módulo GPS/GNSS HERE 3:** Mientras que la potencia es suministrada directamente desde la batería y la telemetría va conectada por puerto USB a la Jetson, el GPS va conectado a un módulo GNSS Here 3 como el mostrado en la Fig. 3.5. Este módulo proporciona un posicionamiento de alta precisión y consta de acelerómetro, giroscopio, barómetro y brújula. Otra parte a tener en cuenta a la hora de volar el dron es el color que proporcionan las leds del GPS y que aportan información sobre su estado; existen dos modos: leds parpadeando y leds estables. En la tabla 3.1 se explica la relación de color con cada uno de los significados.



Figura 3.5: Modulo GPS en el dron.

LED Parpadeando		LED Estable	
■	Inicializar giroscopio	■	Armado, GPS lock
■	Desarmado, GPS lock	■	Armado, GPS activo
■	Desarmado, GPS activo	■	Fallo conexión
■	Fallo pre-armado	■	Firmware no detectado
■	Fallo seguridad GPS		
■	Fallo navegación EKF		
■	Fallo barómetro		

Tabla 3.1: Significado del parpadeo de las luces del GPS.

- **Jetson AGX Xavier:** Esta placa fabricada por NVIDIA está especializada en aplicaciones de robótica e inteligencia artificial. Su reducido tamaño (105 mm x 105 mm x 65 mm) y peso (600 gr), su amplio número de entradas/salidas y su tarjeta gráfica, hacen que sea ideal para máquinas autónomas.

Por otro lado, esta Jetson consta de: una GPU con 512 núcleos NVIDIA Volta y 64 núcleos Tensor, una CPU ARM de 8 núcleos de 64 bits, una RAM de 32 GB de 256 bits y un almacenamiento interno de 32 GB. El procesamiento de su tarjeta gráfica implica un valor añadido en su rendimiento.

Por último, en la Fig. 3.6 se pueden observar todas conexiones relevantes para este proyecto: ethernet RJ45, HDMI 2.0, USB-C 3.1 (x2), microUSB 2.0, antena WiFi, microSD y el jack de alimentación de 12V.

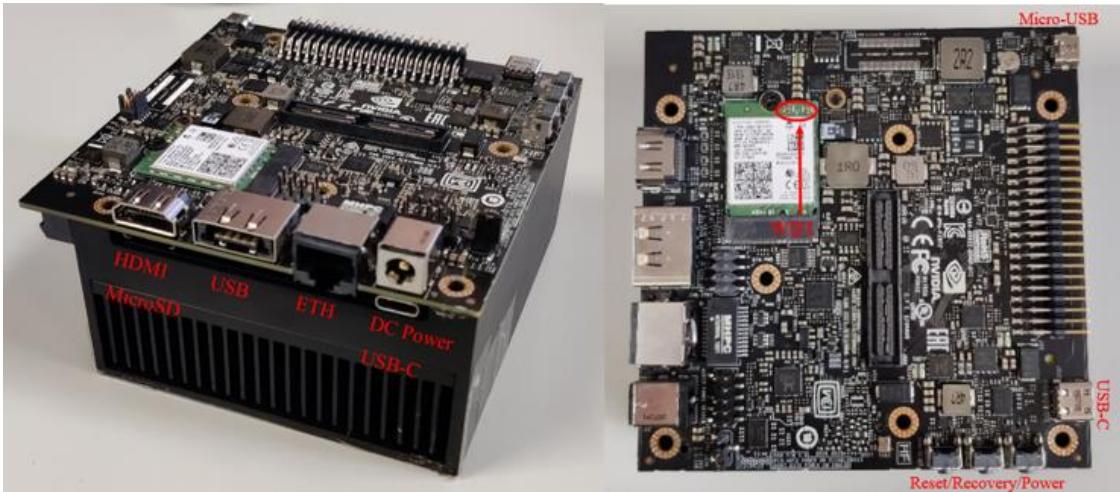


Figura 3.6: Puertos de la Jetson AGX Xavier.

- **Lidar Benewake CE30-C:** Este sensor lidar 3D de estado sólido TOF (Time Of Flight), técnica que estima la posición de los objetos en función del tiempo transcurrido entre la emisión y recepción de un haz de luz infrarroja, tiene una longitud máxima de 4 metros con un FOV (Field Of View) horizontal de 132° y un FOV vertical de 9°. Con un peso reducido de 220 gr, este sensor es una estupenda elección para incorporar a bordo del vehículo de estudio ya que permite obtener una clara nube de puntos en tres dimensiones y sus características físicas son las indicadas en relación a las dimensiones del dron.



Figura 3.7: Sensor lidar Benewake CE30-C. [30]

- **Baterías Gens Ace 5000:** La única fuente de alimentación que lleva este dron a bordo es una batería de LiPo de cuatro celdas, con un peso aproximado de 500 gr, que proporciona un voltaje entre 16,8 V ~ 14 V.



Figura 3.8: Batería Gens Ace 500.

Debido a estas altas magnitudes, es necesario disponer de otros elementos, como el UBEC (Fig. 3.9), que regulen la tensión suministrada a los equipos del vehículo. En este caso, tanto la Jetson como el sensor Lidar tienen una alimentación continua de 12V que obtienen a través de esta conexión.

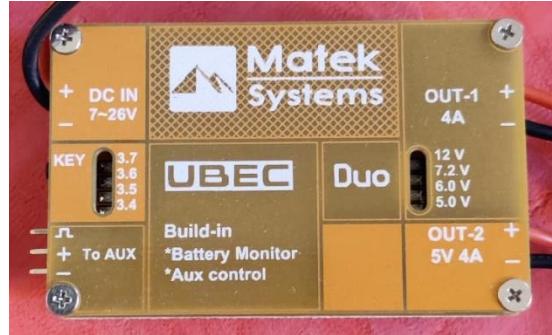


Figura 3.9: Sistema regulador UBEC Duo.

Por otro lado, el autopiloto está conectado directamente a la batería por el módulo suministrado por el fabricante. De este modo, el controlador de vuelo es capaz de monitorizar el estado de las baterías y los sensores que lo componen.

Por último, la conexión entre la batería y los motores de las hélices se realiza a través del ESC (Fig. 3.10), este se encarga de controlar la velocidad de giro de los motores y suministrar la corriente necesaria.



Figura 3.10: ESC F55A Pro. [31]

A modo de conclusión, en la Fig. 3.11 se puede observar un esquema gráfico de la conexión de la batería con los diferentes componentes que necesitan de alimentación a bordo del vehículo aéreo.

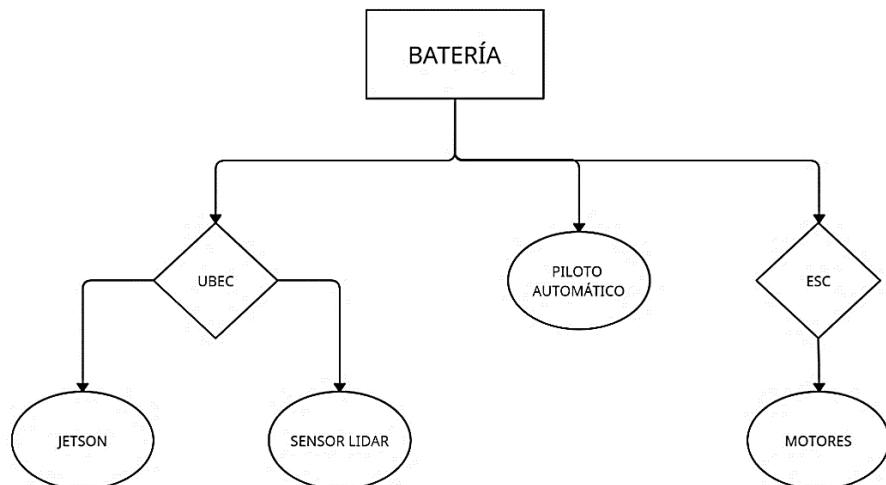


Figura 3.11: Gráfico de la conexión en la batería.

- **BrosTrend Linux Adaptador WiFi USB :** Debido a fallos de conexión por la antena Wifi proporcionada por Nvidia, se ha decidido incorporar a mayores este adaptador Wifi de la marca BrosTrend. Destaca por su compatibilidad con la mayoría de sistemas operativos y por su velocidad wifi de 433 Mbps en bandas de 5 GHz o de 200 Mbps en bandas de 2.4 GHz.
- **Motores AX-2810Q:** Teniendo en cuenta que el peso total aproximado del dron es de 2,5 Kg (desglosado en la tabla 3.2), se ha decidido hacer uso de los motores AX-2810Q (Fig. 3.12). El empuje de estos motores de 750KV es el suficiente como para levantar 1,5 kg cada uno, por lo que se cumple con la necesidad de que los motores deben de ser capaces de levantar el doble del peso del vehículo.

Componente	Peso (gr)
Chasis	425
Pixhawk	500
Jetson	600
Lidar	220
Batería	500 <ul style="list-style-type: none"> • UBEC • ESC • 32 • 17,5
GPS	50
Antena Wifi	5
Total:	2349,5 ~ 2500

Tabla 3.2: Peso de los componentes del dron.

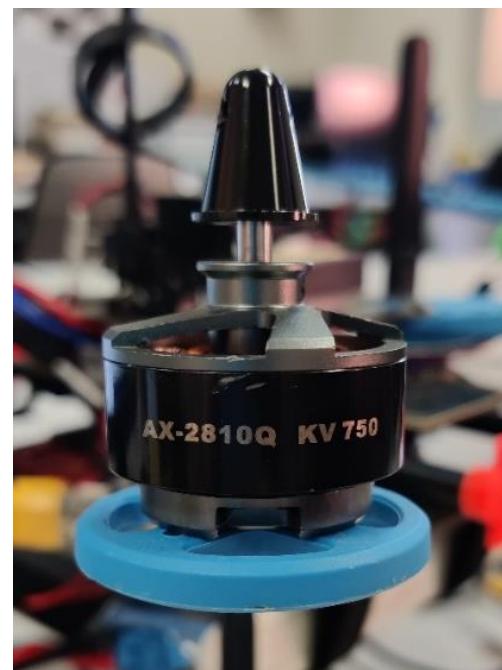


Figura 3.12: Motor del dron.

- **Hélices 1045:** Se han elegido este tipo de hélices debido a las limitaciones geométricas del dron. Cuentan con un longitud de 25,4 cm, un eje de 6 mm y un peso de 10 gr ajustándose a las especificaciones del motor y permitiendo el vuelo del vehículo.

A modo de ilustración, en la Fig. 3.13 se puede observar el montaje final con todos los componentes a bordo del dron explicados en este apartado y su correcto funcionamiento en una prueba de despegue.



Figura 3.13: Dron utilizado para este proyecto.

3.2. Software.

En este proyecto se hace uso del sistema operativo Ubuntu en su versión 18.04 LTS, tanto en el ordenador principal como en la Jetson AGX Xavier. Este sistema operativo es un software libre que pertenece a una distribución de GNU/Linux, el cual destaca por ser uno de los sistemas operativos más seguros del mercado, pudiendo obtener una nueva versión cada seis meses con un soporte técnico y actualizaciones de seguridad durante nueve meses. En caso de que la versión sea LTS (Long Term Support), el soporte técnico asciende hasta los cinco años de duración. Otro punto para tener en cuenta es la gran cantidad de paquetes de software, con licencia libre o código abierto, de la que se compone.

```
nvidia@jetson-0421419020930: ~
cnnvidia@jetson-0421419020930:~$ cat /proc/version
Linux version 4.9.108-tegra (buildbrain@mobile-u64-2565) (gcc version 6.4.1 2017
0707 (Linaro GCC 6.4-2017.08) ) #1 SMP PREEMPT Wed Oct 31 15:17:21 PDT 2018
```

Figura 3.14: Versión Linux en la Jetson a bordo del dron.

Por otro lado, se ha hecho uso de ROS (Robot Operating System), un software libre de ámbito robótico encargado de la gestión de datos, la mensajería, la gestión de las API, los servicios de aplicaciones y la autenticación. Basado en una arquitectura de grafos donde los protagonistas son los nodos que se encargan de enviar y recibir la información.

Para su instalación, es necesario tener previamente instalado en el ordenador un sistema UNIX (Ubuntu /Linux). Las versiones de Ubuntu y ROS compatibles son las siguientes:

Versión de Ubuntu	Versión de ROS
Ubuntu 14.04	ROS ÍNDIGO
Ubuntu 16.04	ROS KINETIC
Ubuntu 18.04	ROS MELODIC
Ubuntu 20.04	ROS NOETIC

Tabla 3.3: Versión correspondiente Ubuntu/ROS.

Para este trabajo se ha decidido utilizar la versión de Ubuntu 18.04 con la distribución de ROS Melodic ya que es el entorno recomendado para una mejor comunicación y experiencia entre PX4 y ROS.

Los pasos por seguir para crear el paquete de ROS en este trabajo han sido los siguientes:

Terminal Ubuntu
<pre>#Añadir los repositorios de ROS a Ubuntu sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu \$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list' sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net --recvkey 0xB01FA116 sudo apt-get update #Instalar la versión de escritorio sudo apt-get install ros-melodic-desktop-full echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc source ~/.bashrc sudo rosdep init rosdep update</pre>

```
sudo apt-get install python-rosinstall
```

```
mkdir -p ~/catkin_ws/src
```

#Crear el área de trabajo

```
cd ~/catkin_ws/src
```

```
rospack profile
```

```
roscore
```

```
catkin_init_workspace
```

```
cd ~/catkin_ws/
```

```
catkin build
```

```
source devel/setup.bash
```

#Crear el paquete

```
cd ~/catkin_ws/src
```

```
catkin_create_pkg tfm std_msgs rospy roscpp
```

Otro paquete por instalar es Octomap, un software libre de código abierto basado en la representación de objetos mediante estructuras de datos octree. Dichas estructuras se caracterizan por subdividir el espacio de cada nodo en ocho octantes como se puede observar en la Fig. 3.15, cada nodo tiene un valor que oscila entre el 0 y el 1; siendo 0 cuando el nodo está libre, 1 cuando el nodo está ocupado y un número de coma flotante que representa la probabilidad de que ese nodo esté ocupado.

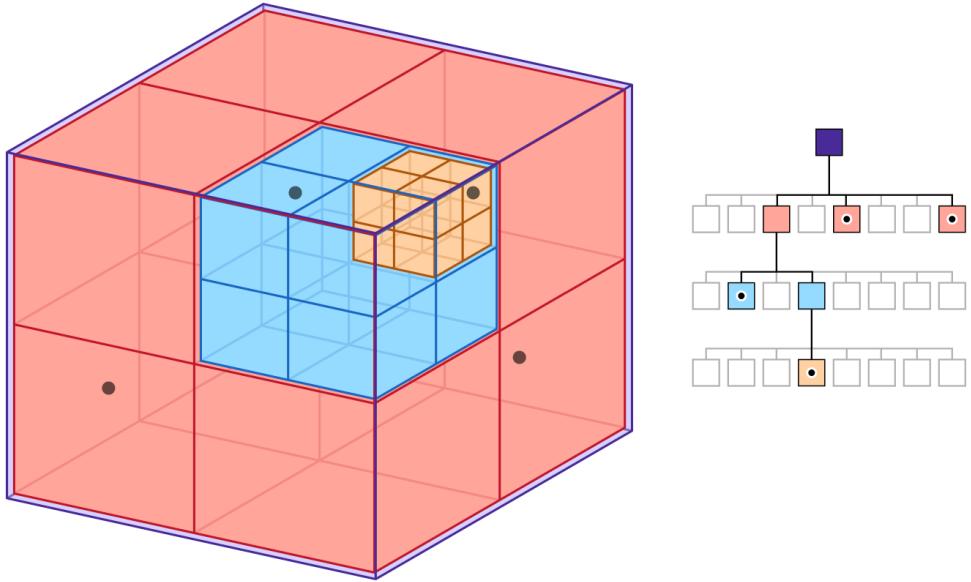


Figura 3.15: Representación Octree. [32]

Dicho esto, cada cuadrante representa la probabilidad de que la celda esté ocupada lo que hace que este software sea ideal para mapas de navegación. Si se quiere aumentar la resolución del mapa hay que disminuir el tamaño de las cuadriculas y viceversa. Es por ello por lo que es aconsejable disminuir la resolución en entornos de gran tamaño con el objetivo de disminuir el tiempo computacional. En la Fig. 3.16, se puede apreciar un mapa de ocupación obtenido de la simulación de este proyecto en el que se hace uso de Octomap.

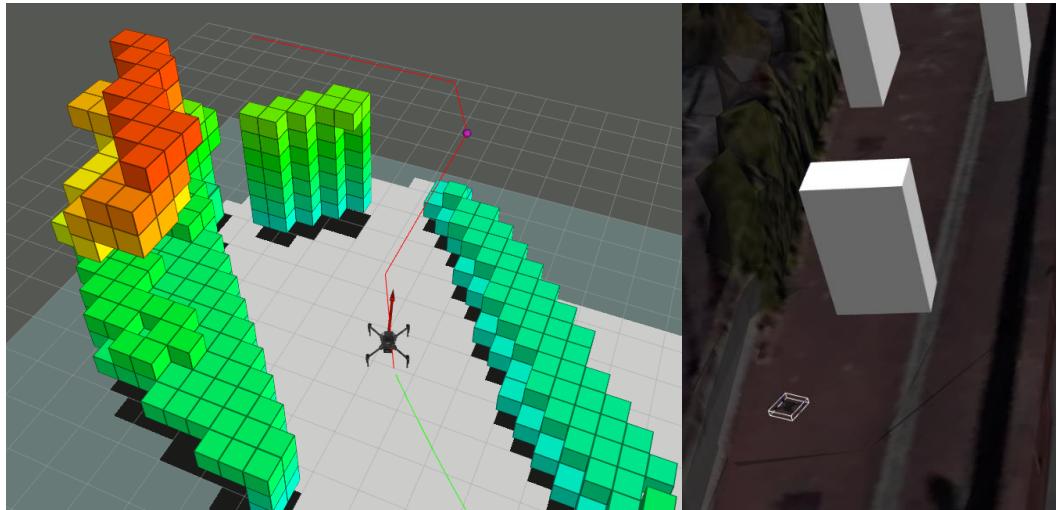


Figura 3.16: OctoMap obtenido en la simulación.

El método para instalar Octomap y utilizar junto con ROS es el siguiente:

Terminal Ubuntu
sudo apt-get install ros-melodic-octomap

Para simular el entorno se ha hecho uso de Gazebo, un simulador 3D de código abierto especializado en el campo de la robótica. Cuenta con motores de física que permiten una representación realista del entorno, así como texturas, sombras e iluminación. Destaca por dar soporte a la simulación de sensores y actuadores.

El método de instalación es el siguiente.

Terminal Ubuntu

```
sudo apt install ros-melodic-gazebo9*
```

Por otro lado, en las comunicaciones se ha usado MAVLink (Micro Air Vehicle Link), un protocolo diseñado para realizar las comunicaciones entre una estación de control de tierra y vehículos aéreos no tripulados. Los mensajes que se envían y publican son archivos en formato XML que permiten la comunicación interna y externa.

MAVLink destaca por ser muy fiable, muy eficiente y por su adaptabilidad con los diferentes lenguajes de programación y sistemas operativos.

El método de instalación es el siguiente:

Terminal Ubuntu

```
sudo apt install python3-lxml libxml2-utils //necesario para los archivos XML  
git clone https://github.com/mavlink/mavlink.git
```

A mayores, se ha utilizado Mavros, un nodo de ROS que permite mantener las comunicaciones con una estación de tierra (.QGroundControl) mediante MAVLink.

Para su instalación es necesario seguir las siguientes directrices:

Terminal Ubuntu

```
sudo apt install ros-melodic-mavros ros-melodic-mavros-extras  
  
wget  
https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh
```

```
chmod +x install_geographiclib_datasets.sh
```

```
sudo ./install_geographiclib_datasets.sh
```

Como controladora de vuelo se ha utilizado el software de código abierto PX4, un proyecto que proporciona a los desarrolladores una cantidad considerable de herramientas y así poder crear soluciones personalizadas para diferentes aplicaciones. PX4 pertenece a Dronecode, una organización sin ánimo de lucro que busca fomentar el uso de código abierto en vehículos voladores.

Las principales características que destacan en PX4 son:

- **Fuente abierta:** PX4 sigue una metodología de desarrollo global. Su enfoque es un conjunto de herramientas general, bastante utilizado y escalable dentro de la industria.
- **Configurabilidad:** Dronecode también alberga SDK por lo que PX4 dispone de API y SDK optimizados para los desarrolladores. Los módulos del dron se pueden intercambiar y son independientes unos de otros. Destaca por su sencillez en la implementación.
- **Pila de autonomía:** Uno de los puntos fuertes de PX4, es que está diseñado para integrarse con la visión por computador y las operaciones autónomas. Facilita la elaboración de algoritmos para la localización y detección de obstáculos.
- **Arquitectura modular:** Una de las características más interesantes es la escalabilidad, tanto a nivel de software como de hardware. Su arquitectura basada en puertos permite no perder rendimiento ni solidez.

Las empresas y organizaciones han empezado a utilizar PX4 debido a su interoperabilidad y licencia permisiva. Por otro lado, cuenta con unas fuertes funciones de seguridad y ha sido validado por Realworld.

Como experiencia personal, quiero destacar la excelente calidad del soporte técnico y la gran comunidad de desarrolladores que crece día a día. A continuación, se muestra los pasos a seguir para su implementación en el sistema.

Terminal Ubuntu - Dependencias

#Requisitos:

- ROS Melodic
- Gazebo9
- Mavros

```
sudo apt-get remove modemmanager
```

```
sudo apt-get update
```

```
sudo apt-get install git zip cmake build-essential genromfs ninja-build exiftool  
python-pip python-dev
```

```
which xxd || sudo apt install xxd -y || sudo apt-get install vim-common --no-install-recommends
```

```
sudo pip3 install wheel setuptools
```

```
sudo pip3 install argparse argcomplete coverage cerberus empy jinja2  
matplotlib==3.0.* numpy nunavut packaging pkgconfig pyros-genmsg pyulog  
pyyaml requests serial six toml psutil pyulog Wheel
```

```
sudo pip install --upgrade pip
```

```
sudo pip install wheel setuptools
```

```
sudo pip install argcomplete argparse catkin_pkg catkin-tools cerberus coverage  
empy jinja2 matplotlib==2.2.* numpy pkgconfig px4tools pygments pymavlink  
packaging pyros-genmsg pyulog pyyaml requests rosdep rospkg serial six toml  
pandas pyserial
```

```
sudo apt-get install ninja-build
```

```
wget https://www.eprosima.com/index.php/component/ars/repository/eprosima-fast-rtps/eprosima-fast-rtps-1-7-1/eprosima_fastrtps-1-7-1-linux-tar-gz -O eprosima_fastrtps-1-7-1-linux.tar.gz
```

```
tar -xzf eprosima_fastrtps-1-7-1-linux.tar.gz eProsima_FastRTPS-1.7.1-Linux
```

```
tar -xzf eprosima_fastrtps-1-7-1-linux.tar.gz requiredcomponents
```

```
tar -xzf requiredcomponents/eProsima_FastCDR-1.0.8-Linux.tar.gz
```

```
cd eProsima_FastCDR-1.0.8-Linux && ./configure --libdir=/usr/lib && make -j2 && sudo make install
```

```
cd eProsima_FastRTPS-1.7.1-Linux && ./configure --libdir=/usr/lib && make -j2 && sudo make install
```

Terminal Ubuntu - Firmware

```
git clone https://github.com/PX4/Firmware.git
```

```
cd ~/Firmware
```

```
DONT_RUN=1 make px4_sitl_default gazebo
```

```
echo 'export
```

```
GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:~/Firmware/build/px4_sitl_default/build_gazebo' >> ~/.bashrc
```

```
echo 'export  
GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/Firmware/Tools/sitl_gazebo/models' >> ~/.bashrc
```

```
echo 'export  
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/Firmware/build/px4_sitl_default/build_gazebo' >> ~/.bashrc
```

```
echo 'export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:~/Firmware' >>
~/.bashrc

echo 'export
ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:~/Firmware/Tools/sitl_gazebo'
>> ~/.bashrc

source ~/.bashrc
```

Por último, MavSDK es un conjunto de librerías que permiten la comunicación con MavLINK. En este trabajo se han utilizado para llevar a cabo las misiones del dron junto con el lenguaje de programación python3.

Terminal Ubuntu

```
pip3 install mavsdk
```

4. DESARROLLO DEL SISTEMA DE MAPEADO Y EVITACIÓN DE OBTÁCULOS.

En este aparto se muestra el estudio y desarrollo que se ha llevado a cabo para poder realizar las simulaciones de vuelo y la puesta en marcha real del vehículo aéreo.

4.1. Diseño del sistema virtual.

Para realizar las pruebas en un entorno simulado es necesario obtener un modelo 3D del entorno con el mayor detalle posible; para ello, se han probado varios métodos basados en la obtención de nubes de puntos y que se explican a continuación en el apartado 4.1.1.

Por otro lado, es necesario conseguir un modelo de dron, de tamaño y funcionalidad similares al dron real, que cargue con un sensor de especificaciones idénticas al lidar usado en las pruebas reales y que se explica en detalle en el apartado 4.1.2.

4.1.1. Representación del entorno en la simulación.

En este apartado se explican diferentes métodos para obtener un modelo del entorno en 3D lo más realista posible.

4.1.1.1. Implementación de mapas Lidar en Matlab.

La primera opción planteada para este trabajo fue implementar los datos de un sensor lidar en el programa de Matlab, para ello se ha hecho uso de una base de datos que recoge la mayor parte del territorio español en formato de nube de puntos.

El primer paso que se debe llevar a cabo es ir a la página web del CNIG (Centro Nacional de Información Geográfica) [33]. En esta web se puede encontrar una gran variedad de información en relación con la orografía del territorio español.

En este caso, el apartado que nos interesa es el de modelos digitales de elevaciones. En esta sección se puede encontrar información altimétrica que representa el relieve del territorio nacional, y en el caso de los datos Lidar, también de los elementos que se encuentran sobre él. Dentro de los modelos digitales de elevaciones se selecciona el Lidar 2ª Cobertura; esta selección permitirá obtener ficheros digitales de nubes de puntos 3D en formato .LAZ.



Figura 4.1: Página web para descargar mapas lidar a nivel nacional.

Una vez dentro del mapa, se selecciona la parcela deseada. Nota: Las parcelas proporcionadas por el CNIG tienen unas dimensiones establecidas de 2000 x 2000 metros.



Figura 4.2: Recorte de la parcela deseada para extraer el mapa lidar.

Una vez se tiene el archivo .LAZ, se procede a editar la parcela y seleccionar las regiones de interés. Para ello, se ha hecho uso del programa CloudCompare que permite el procesamiento de nubes de puntos 3D.



Figura 4.3: Nubes de puntos (entorno ciudad).

Se ha recortado una parcela con cierta elevación y edificios altos para ver el comportamiento de las trayectorias generadas por el sistema.

A continuación, se procesará el mapa de nube de puntos 3D en MATLAB. Para ello, el primer paso es definir la ruta de nuestro archivo .LAZ/.LAS.

Código

```
path = 'C:\Users\DMelirap\Desktop\TFM\Mapas\OurenseCut.las';
lasReader = lasFileReader(path);
```

Acto seguido, se hace uso de la clase ptCloud para leer los datos y almacenar las variables de la nube de puntos.

Código

```
ptCloud = readPointCloud(lasReader);
pcshow(ptCloud.Location);
```

```
>> path = 'C:\Users\DMelirap\Desktop\TFM\Mapas\OurenseCut.las'
path =
'C:\Users\DMelirap\Desktop\TFM\Mapas\OurenseCut.las'
>> lasReader = lasFileReader(path)
lasReader =
lasFileReader with properties:

    FileName: 'C:\Users\DMelirap\Desktop\TFM\Mapas\OurenseCut.las'
    Count: 218002
    LasVersion: '1.2'
    XLimits: [5.9322e+05 5.9365e+05]
    YLimits: [4.6882e+06 4.6887e+06]
    ZLimits: [101.9300 207.8600]
    GPSTimeLimits: [1.5677e+08 sec    1.571e+08 sec]
    NumReturns: 4
    NumClasses: 13
>> ptCloud = readPointCloud(lasReader)
ptCloud =
pointCloud with properties:

    Location: [218002x3 single]
    Count: 218002
    XLimits: [5.9322e+05 5.9365e+05]
    YLimits: [4.6882e+06 4.6887e+06]
    ZLimits: [101.9300 207.8600]
    Color: [218002x3 uint8]
    Normal: []
    Intensity: [218002x1 uint8]
>> pcshow(ptCloud.Location)
```

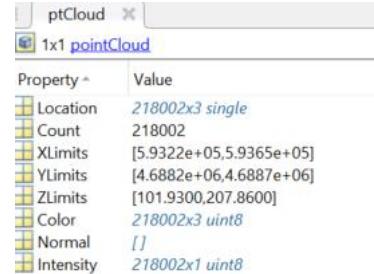


Figura 4.4:Leyendo nubes de puntos en Matlab.

Como resultado, se obtiene el siguiente mapa: (Las coordenadas mostradas en el mapa son en formato UTM).

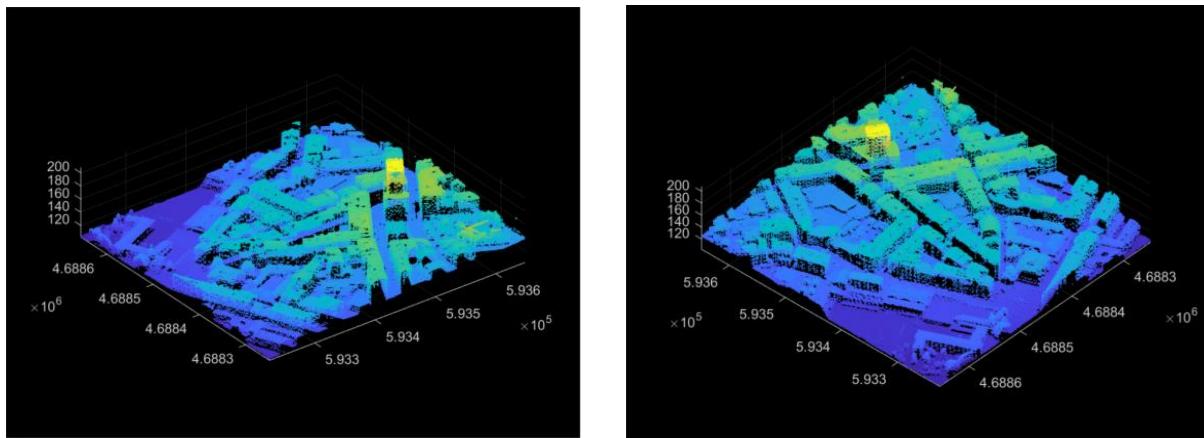


Figura 4.5: Visualización de nubes de puntos en Matlab.

Una vez se tiene la nube de puntos 3D se pueden sacar las mallas de la superficie y las curvas de nivel. Para ello, el primer paso es definir el máximo y el mínimo de las variables X e Y.

Código

```
minX=min(x);
minY=min(y);
maxX=max(x);
maxY=max(y);
```

Acto seguido, se generan los vectores de las coordenadas X e Y y se procede a generar el entorno de rejillas 3D.

Código

```
CoordX=linspace(minX,maxX,100);
CoordY=linspace(minY,maxY,100);
[Mx,My]=meshgrid(CoordX,CoordY);
Mz=griddata(x,y,z,Mx,My);
```

Con estas variables, se puede obtener la superficie de rejillas del mapa y las curvas de nivel.

Código

```
surf(Mz);
```

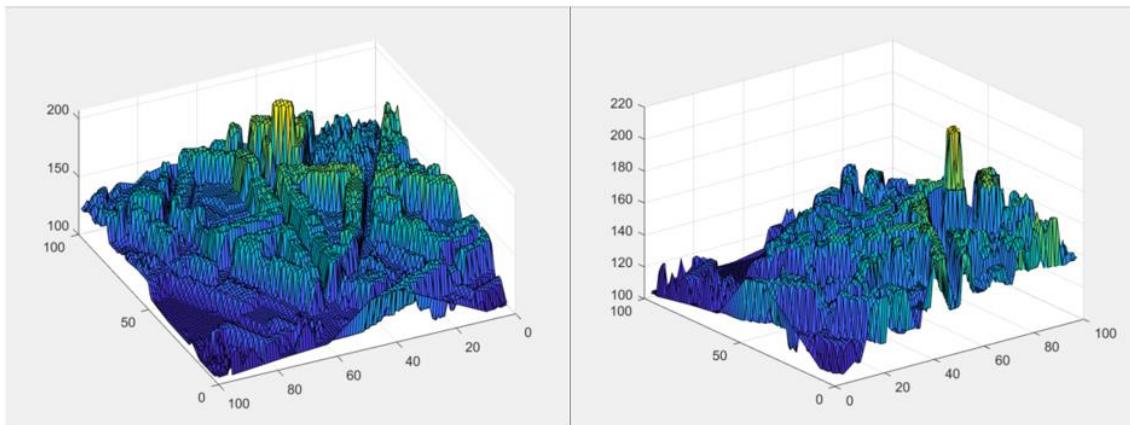


Figura 4.6: Generación de superficie a partir de nubes de punto.

Código

```
contour(Mz,100);
```

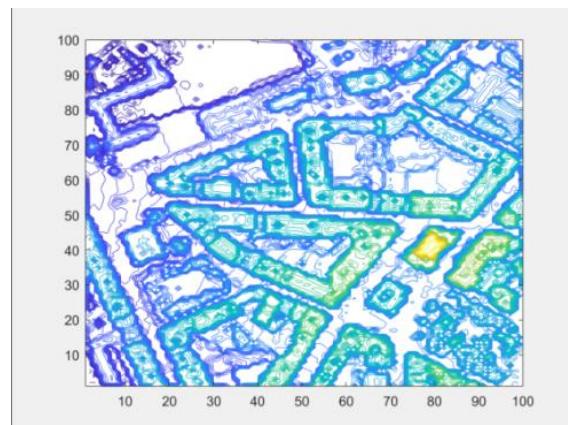


Figura 4.7: Curvas de nivel de la superficie obtenida.

Por otro lado, también es interesante aplicar la función `alphaShape(x,y,z)` que permite transformar el mapa obtenido en un volumen creado a partir de polígonos, manteniendo las coordenadas UTM. Esto puede ser útil ya que se puede llegar a transformar el volumen obtenido en un terreno y crear las trayectorias en base a estos parámetros.

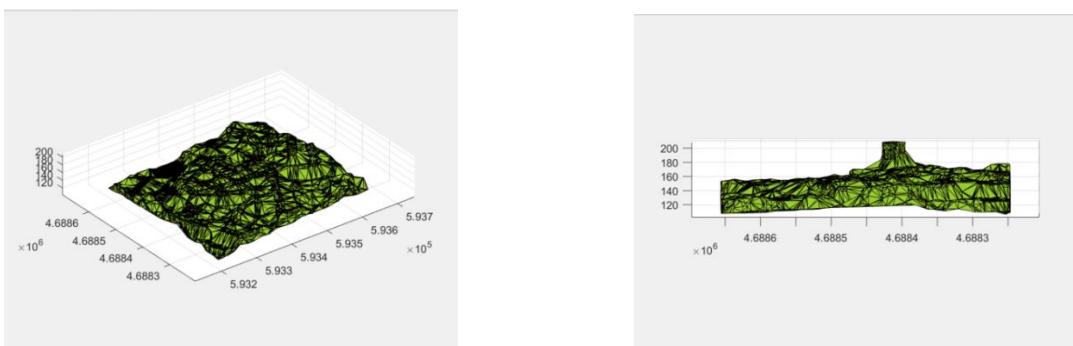


Figura 4.8: Generación de mallas a partir de nubes de punto

4.1.1.2. Implementación de mapas Lidar en el entorno ROS/Gazebo.

Debido a la impotencia de crear un mapa de ocupación 3D en Matlab a partir de las nubes de puntos obtenidas en la página web del CNIG, se ha optado por utilizar directamente el entorno ROS-Gazebo.

Para ello, el primer paso ha sido transformar las nubes de puntos en mallas. Previamente, con el programa CloudCompare, se transforman los archivos .las/.laz obtenidos del CNIG en archivos .ply. Posteriormente, se hace uso del programa MeshLab para transformar el archivo .ply en un archivo .obj y aportar textura al objeto. Este archivo .obj es el que se utilizará para cargar el mapa en el entorno de Gazebo. Los pasos por seguir son los siguientes:

Directrices MeshLab
<ul style="list-style-type: none">▪ Filters >> Point Set >> Point Cloud Simplification.▪ Filters >> Point Set >> Compute normals for point sets. (20/8)▪ Filters >>Remeshing, Simplification and Reconstruction >> Screened Poisson Surface Reconstruction.▪ Filters >> Selection >> Select Faces with edges longer than▪ Filters >> Cleaning and Repairing >> Remove isolated pieces (wrt Face Num.).▪ Filters >> Selection >> Select non Manifold Vertices.▪ Filters >> Texture >> Per Vertex Texture Function.▪ Filters >> Texture >> Convert PerVertex UV into PerWedge UV.

- Filters >> Texture >> Parametrization: trivial Per-triangle. (0/4096/0/Basic).
- File >> Export Mesh As...
- Filters >> Texture >> Transfer: Vertex Color to Texture. (4096/4096/Assign texture/Fill texture).

En las siguientes figuras se puede observar unos cuantos ejemplos de entornos de monte y entornos urbanos obtenidos con los parámetros descritos en la tabla superior. Se puede apreciar que en los entornos de monte se obtiene un mejor resultado debido a la simplicidad del terreno ya que la acumulación de edificios en los entornos urbanos dificulta a la hora de obtener una malla más detallada.

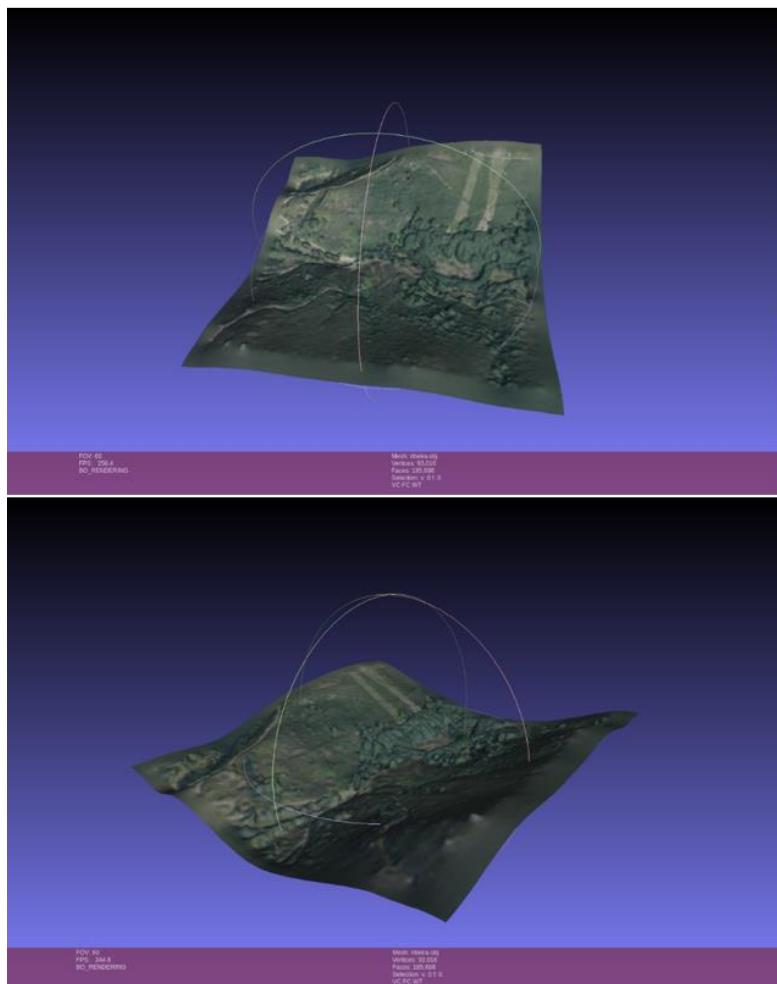


Figura 4.9: Perspectiva completa entorno de montaña (MeshLab).

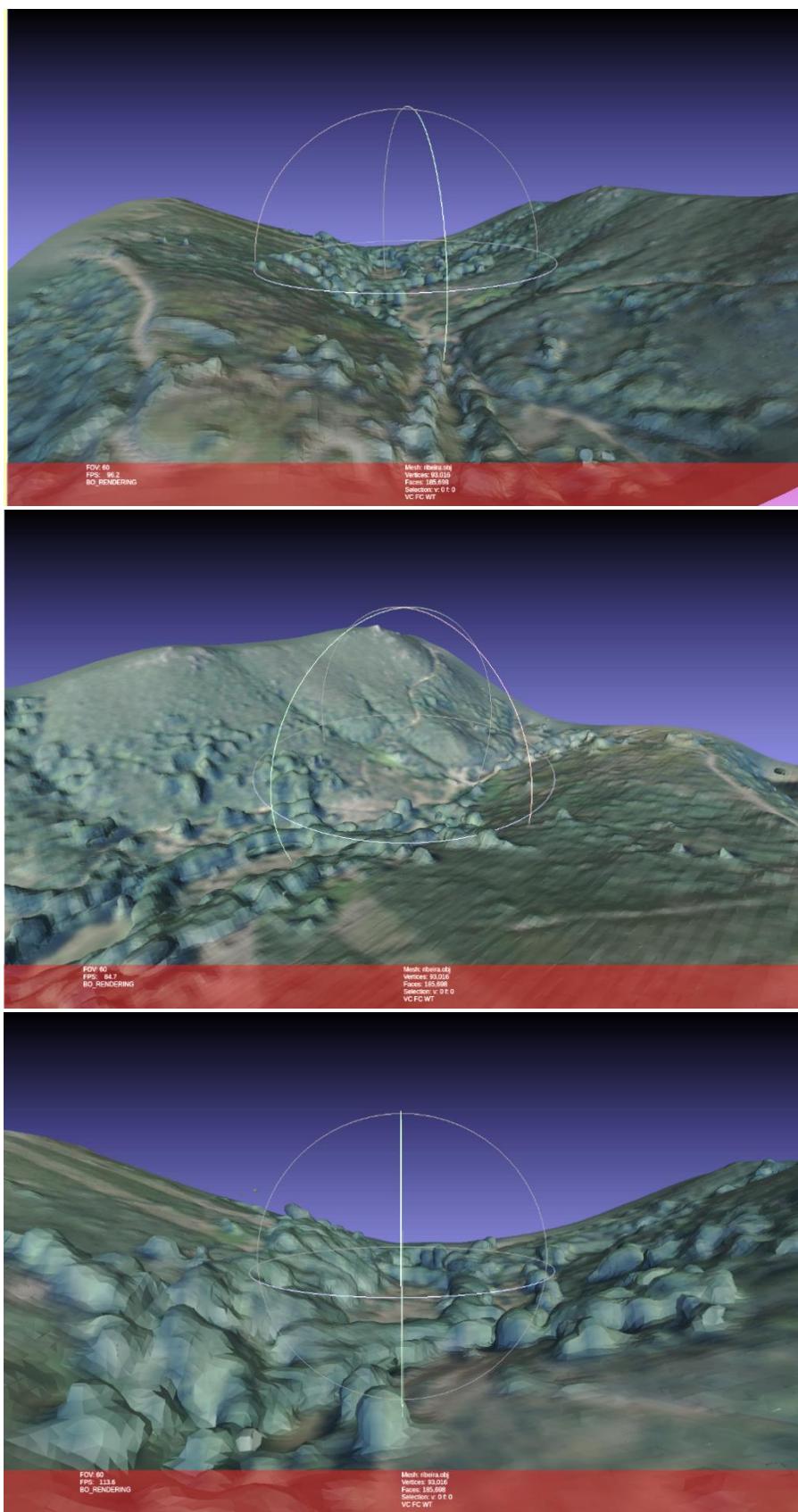


Figura 4.10: Detalles del modelo de montaña.

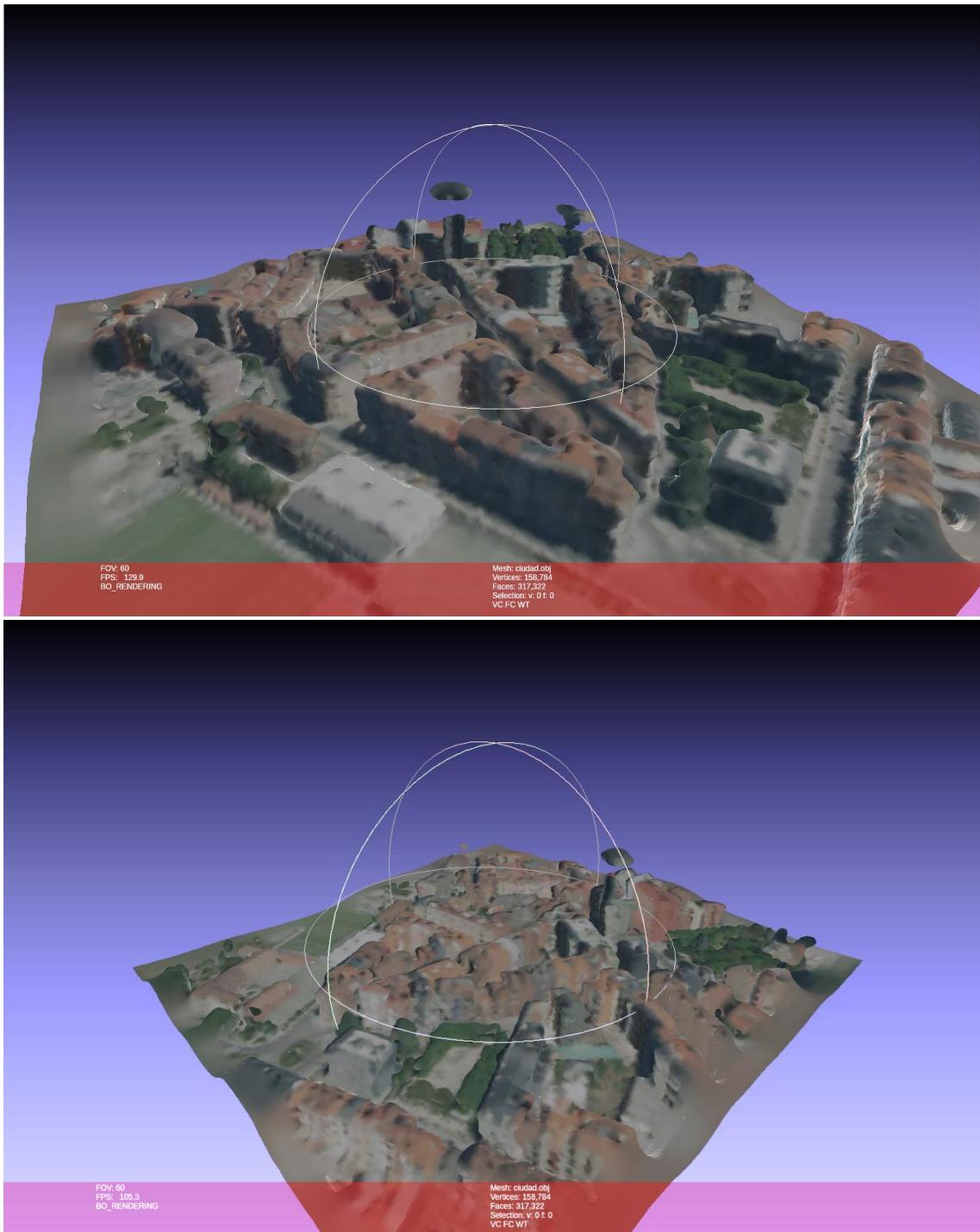


Figura 4.11: Perspectiva completa entorno de ciudad (MeshLab).

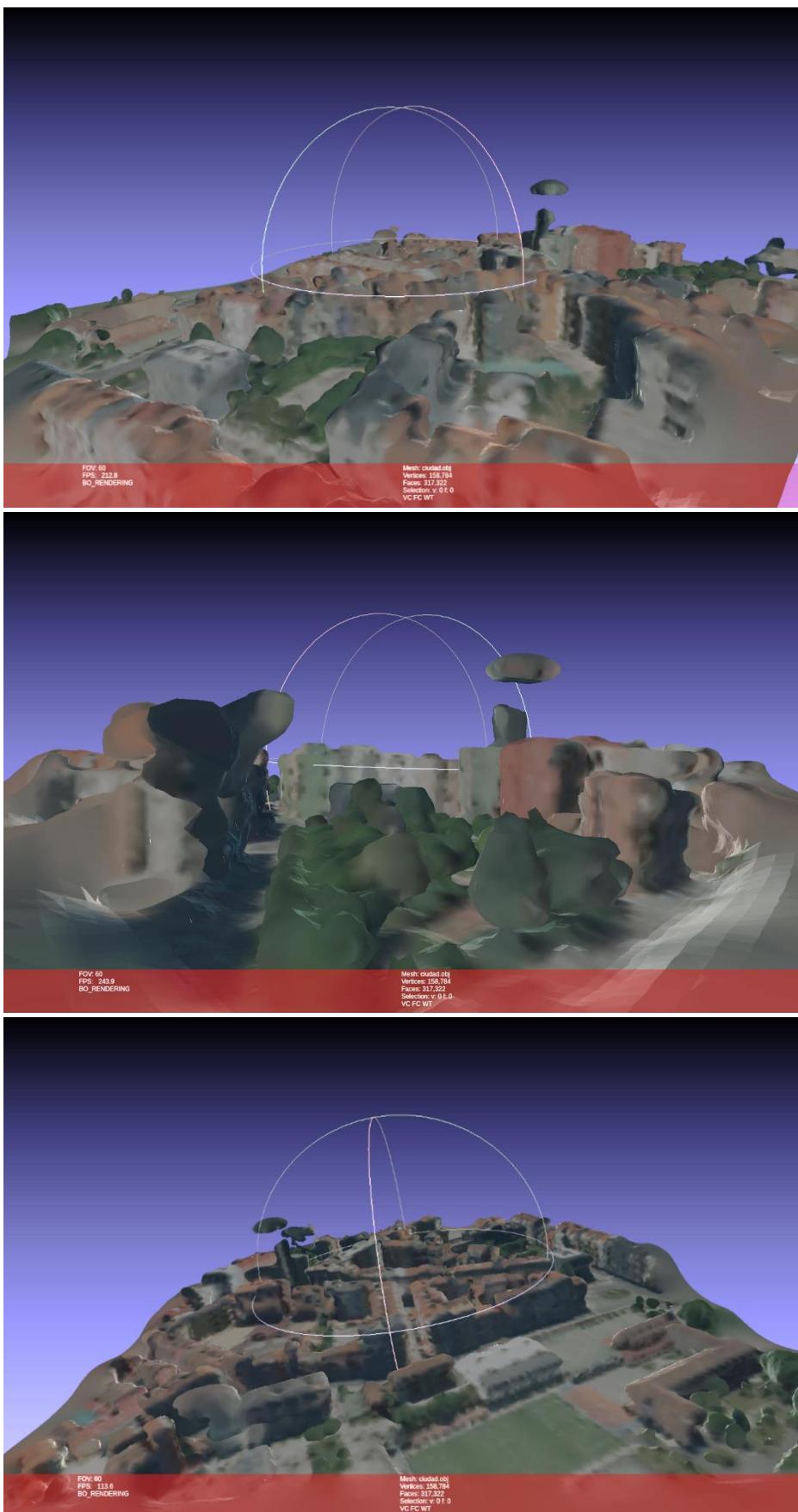


Figura 4.12: Detalles del modelo ciudad.

Una vez se tiene el archivo .obj creado junto con su textura, se procede a crear el modelo en la carpeta de gazebo (~/.gazebo/models). Para ello, se crea una carpeta con el nombre del objeto que vamos a utilizar (en este caso monte : ~/.gazebo/models/monte). Acto seguido, se crean los archivos model.config y model.sdf que permiten importar el archivo a Gazebo.

```
<?xml version="1.0"?>
<model>
  <name>Monte</name>
  <version>1.0</version>
  <sdf version="1.5">model.sdf</sdf>

  <author>
    <name>David</name>
    <email>meirapliego@gmail.com</email>
  </author>

  <description>
    Ribeira
  </description>
</model>
```



```
<?xml version="1.0" ?>
<sdf version="1.5">
  <model name="ribeira">
    <pose>-654154.187500 -4656699 -808.547485 0 0 0</pose>
    <static>0</static>
    <link name="body">
      <visual name="visual">
        <cast_shadows> true </cast_shadows>
        <pose>-654154.187500 -4656699 -808.547485 0 0 0</pose>
        <geometry>
          <mesh><uri>model://monte/media/ribeira.obj</uri></mesh>
        </geometry>
        <material>
          <script>
            <uri>model://monte/media/textures</uri>
            <uri>model://monte/media/scripts</uri>
            <name>monteMat/Diffuse </name>
          </script>
        </material>
      </visual>
      <collision name="collision">
        <pose>-654154.187500 -4656699 -808.547485 0 0 0</pose>
        <geometry>
          <mesh><uri>model://monte/media/ribeira.obj</uri></mesh>
        </geometry>
      </collision>
    </link>
  </model>
</sdf>
```

Figura 4.13: Archivos model.config y model.sdf del mapa a representar.

[*Ojo con la <pose> de cada objeto, el programa CloudCompare indica el desplazamiento en los ejes (x,y,z) al importar la nube de puntos].

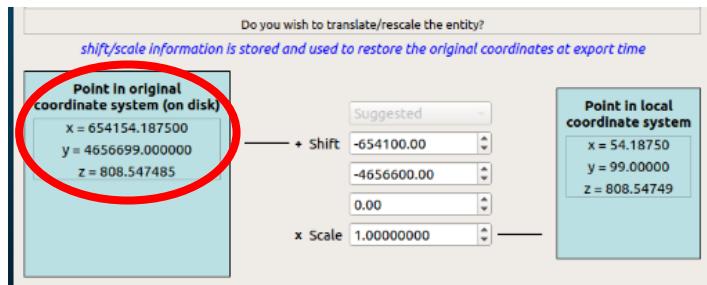


Figura 4.14: Desplazamiento en los ejes.

Una vez se tienen los archivos model.config y model.sdf creados, se crea una carpeta en la que se introduce el archivo .obj las texturas (/textures) y el script (/scripts) monte.material que permite al objeto cargar las texturas en Gazebo.



```
material monteMat/Diffuse
{
    receive_shadows off
    technique
    {
        pass
        {
            texture_unit
            {
                texture ribeira_tex.png
            }
        }
    }
}
```

Figura 4.15: Archivo para dar textura al mapa.

Una vez se tiene el modelo configurado con todos sus archivos, solo falta abrir el programa Gazebo y cargar el mapa (Fig. 4.16 & Fig. 4.17).

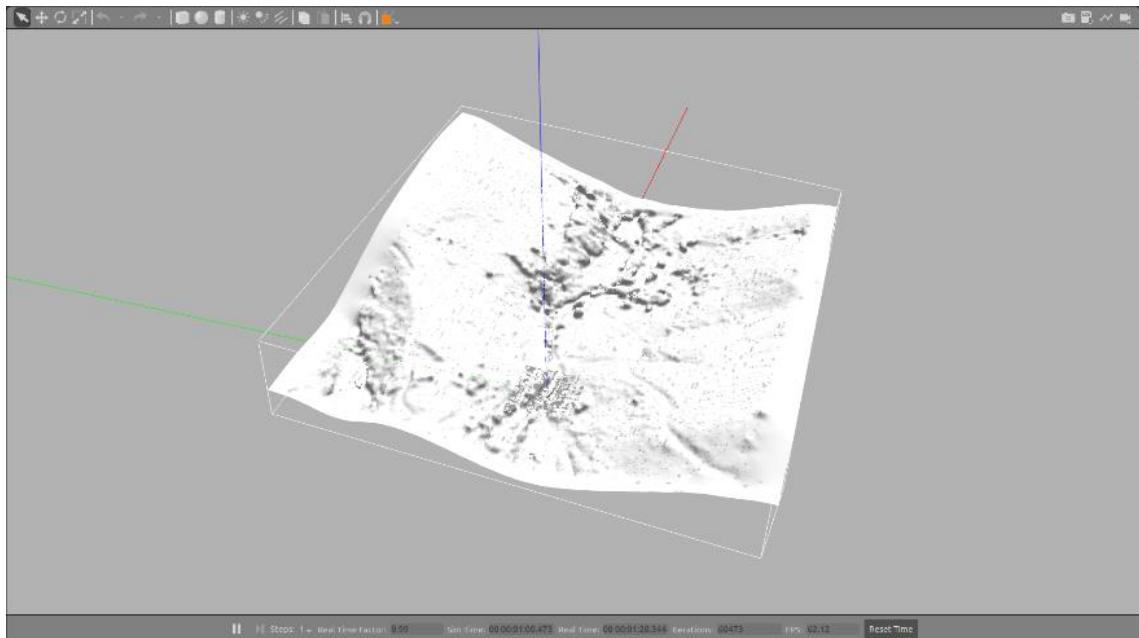
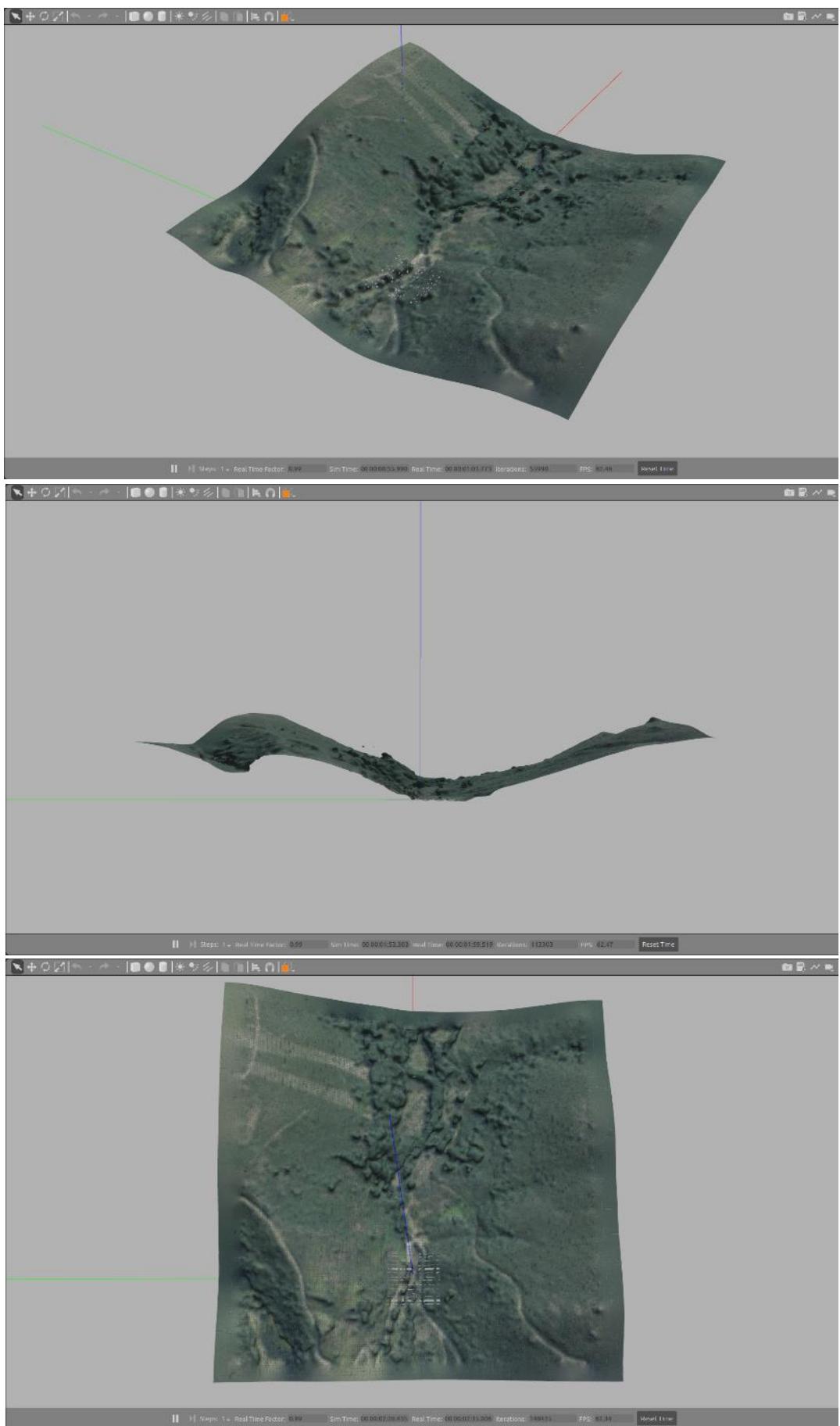


Figura 4.16: Mapa sin textura. (Gazebo)



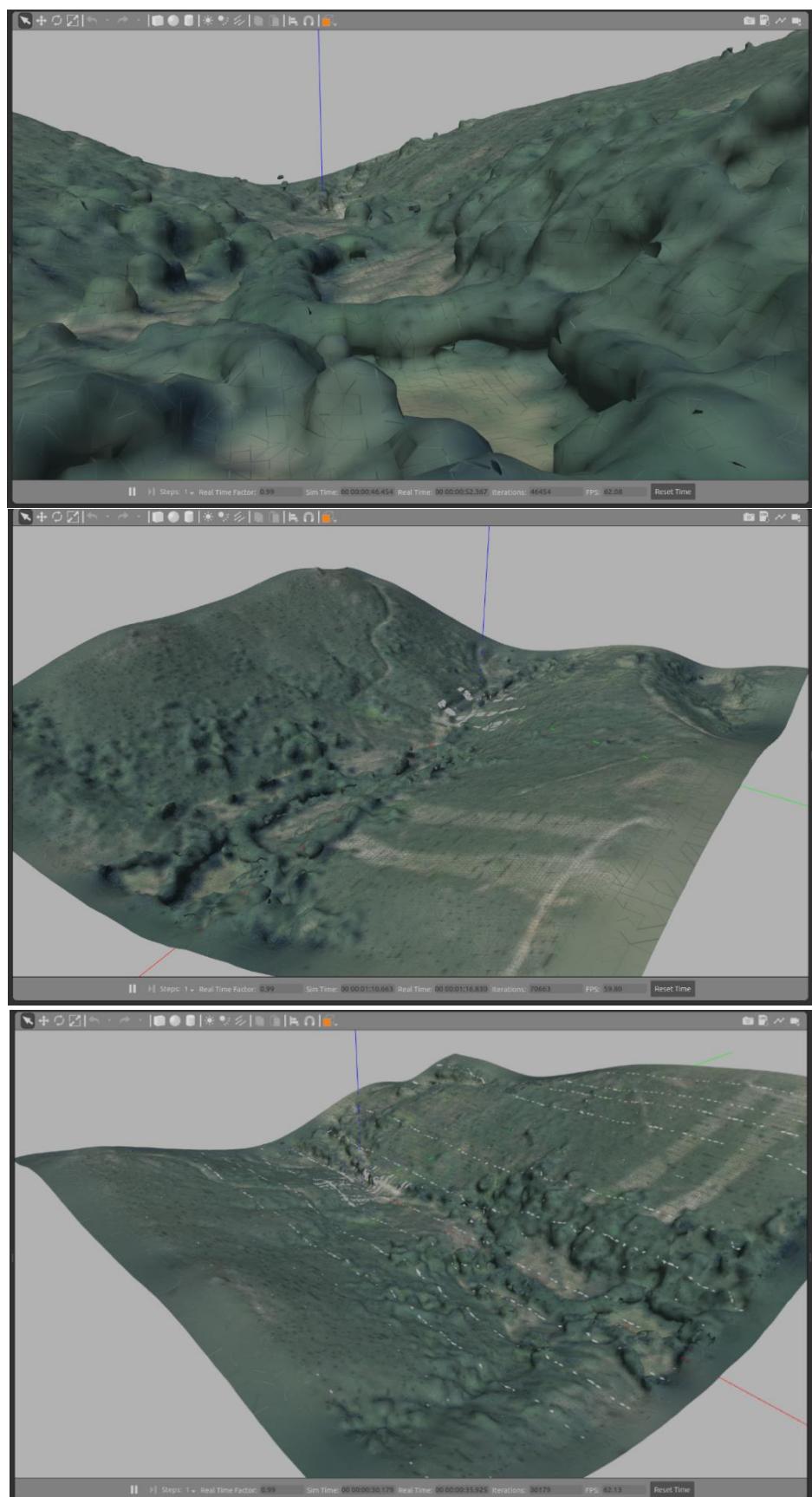


Figura 4.17: Mapa texturizado. (Gazebo)

4.1.1.3. Entornos 3D de Google Maps en ROS/Gazebo.

Este método basa su funcionamiento gracias a una extensión del programa de diseño Blender que permite importar modelos 3D a partir de información recogida en los mapas actuales de Google Maps. Gracias a este método, se pueden obtener mapas de cualquier zona geográfica, con una calidad en el detalle muy destacable y una actualización actual.

Para entrar en contexto, primero es necesario conocer el entorno de Google Maps. A simple vista, puede parecer que Google Maps está compuesto de imágenes de calles y edificios; sin embargo, hoy en día, la mayoría de las ciudades y entornos montañosos se encuentran modelados en 3D. Lamentablemente, estos diseños 3D pertenecen a Google lo que impide el poder obtener el modelo 3D.

Sin embargo, después de una exhaustiva búsqueda, se ha encontrado una forma de obtener estos modelos 3D deseados. Para ello es necesario contar con el sistema operativo Windows y las siguientes aplicaciones:

1. Blender

Blender es un software multiplataforma gratuito especializado en la representación de modelos 3D. Sus potentes motores gráficos (Interno, Cycles y FreeStyle) y su continuo desarrollo construyen una experiencia de usuario única.



2. RenderDoc

RenderDoc es un programa de código abierto gratuito que permite depurar y monitorizar la GPU. Su punto fuerte es poder inspeccionar paso a paso todas las llamadas de la API gráfica y sus argumentos.



3. Google Chrome

Google Chrome es un navegador web gratuito y desarrollado por Google que ocupa la primera posición a nivel mundial como el navegador más utilizado llegando a alcanzar los 900 millones de usuarios.



Una vez mostrados los programas a utilizar, se procede a explicar su funcionamiento.

El primer paso que seguir es habilitar ciertos permisos de la aplicación RenderDoc en Google Chrome. Para ello hay que introducir los siguientes comandos en sus respectivas rutas:

```
C:\Windows\System32\cmd.exe /c  
"SET RENDERDOC_HOOK_EGL=0 && START "  
C:\Program Files (x86)\Google\Chrome\Application\chrome.exe^  
"--disable-gpu-sandbox --gpu-startup-dialog"
```

El siguiente paso es abrir la aplicación de RenderDoc e ir a *File >> Inject into process*.

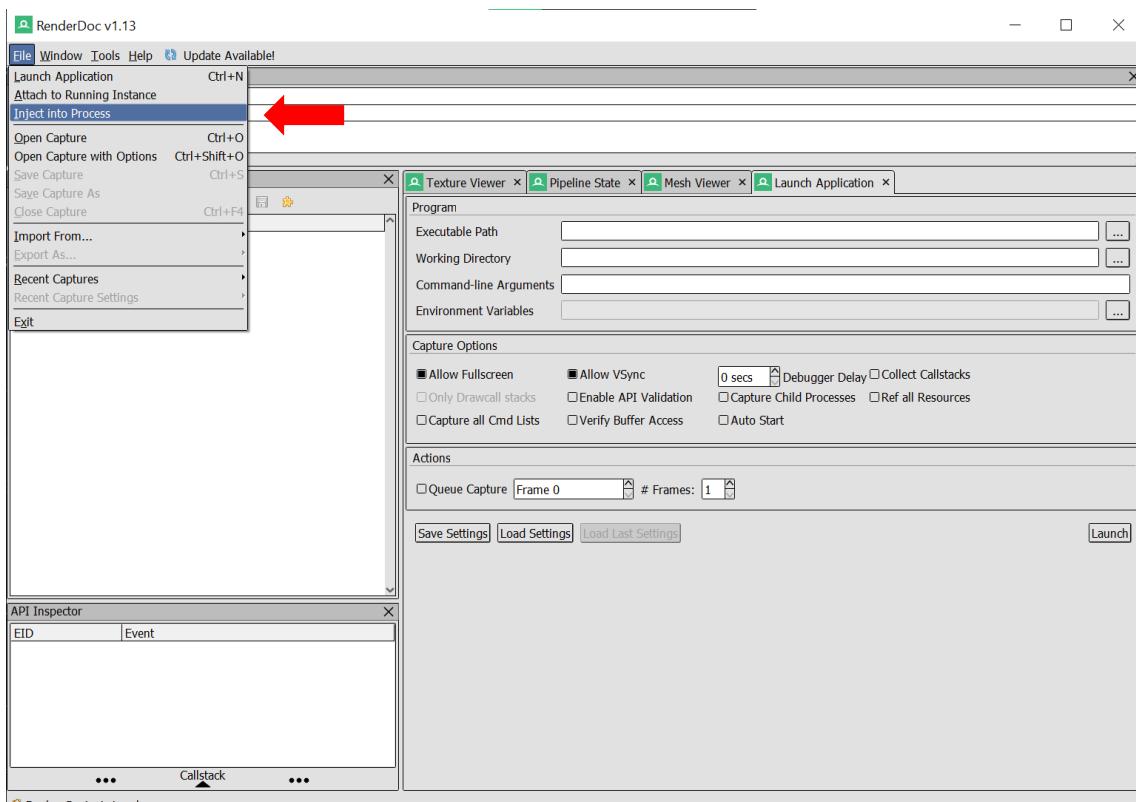


Figura 4.18: Inicializar proceso en RenderDoc.

Acto seguido, hay que abrir la aplicación de Google Chrome de la cual emergerá una ventana que indica la inicialización de la GPU y su código PID. Es importante no aceptar todavía esta ventana.

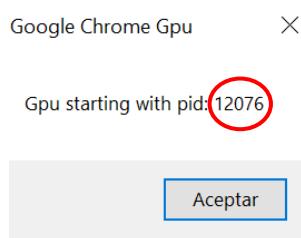


Figura 4.19: PID de GPU.

Volviendo a RenderDoc, se busca la barra de *Filter process list by PID or name* e introducimos la palabra Chrome, la cual nos permite hacer un filtro de los procesos abiertos. Se selecciona la fila con el PID correspondiente y se pulsa *Inject*. Ahora sí, es el momento de cerrar la pequeña ventana de Chrome con el PID.

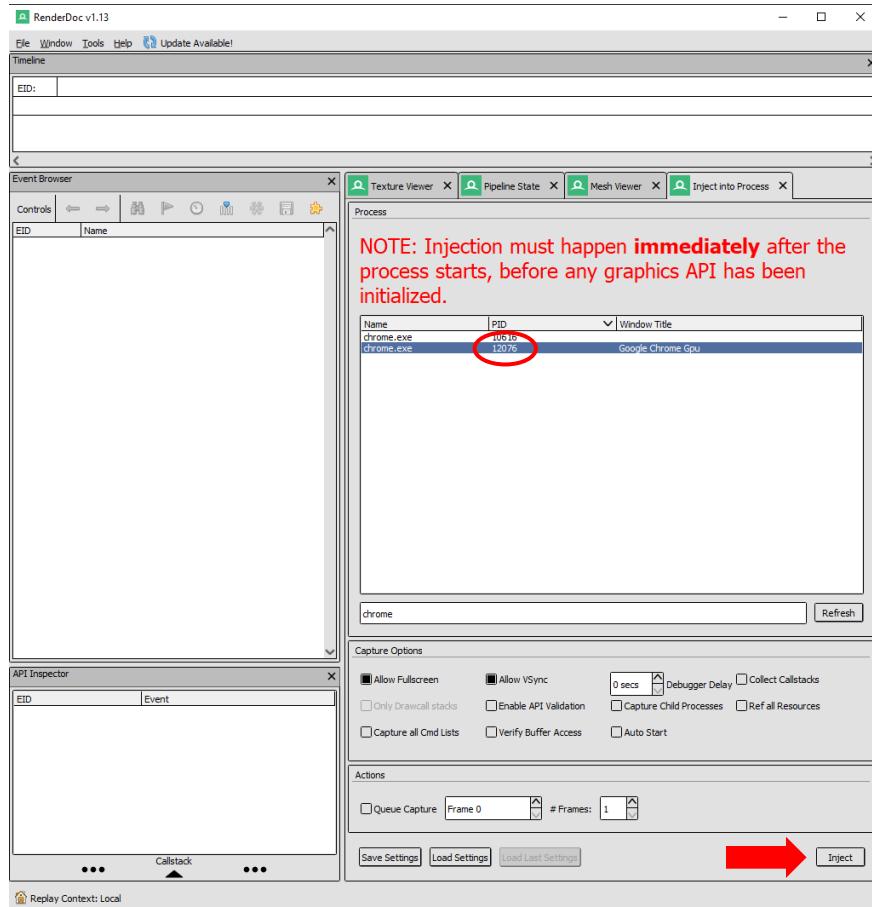


Figura 4.20: Inicializar la ventana con el PID correspondiente.

Una vez se ha abierto Google Chrome, hay que dirigirse a Google Maps y escoger una localización para el modelado 3D. En este caso se ha escogido el campus de Leganés de la Universidad Carlos III de Madrid (UC3M).

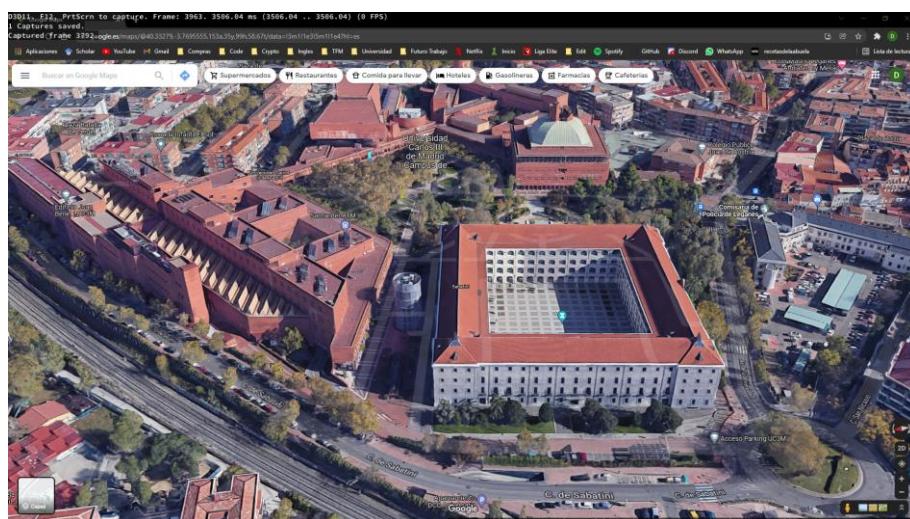


Figura 4.21: Vista aérea de la Universidad Carlos III en Google Maps.

Cuando se está en la localización deseada, se hace una captura de pantalla con ‘F12’ o con la tecla ‘Imp pant’. Acto seguido, debe de aparecer la captura en RenderDoc, se guarda la captura y se cierra el programa.

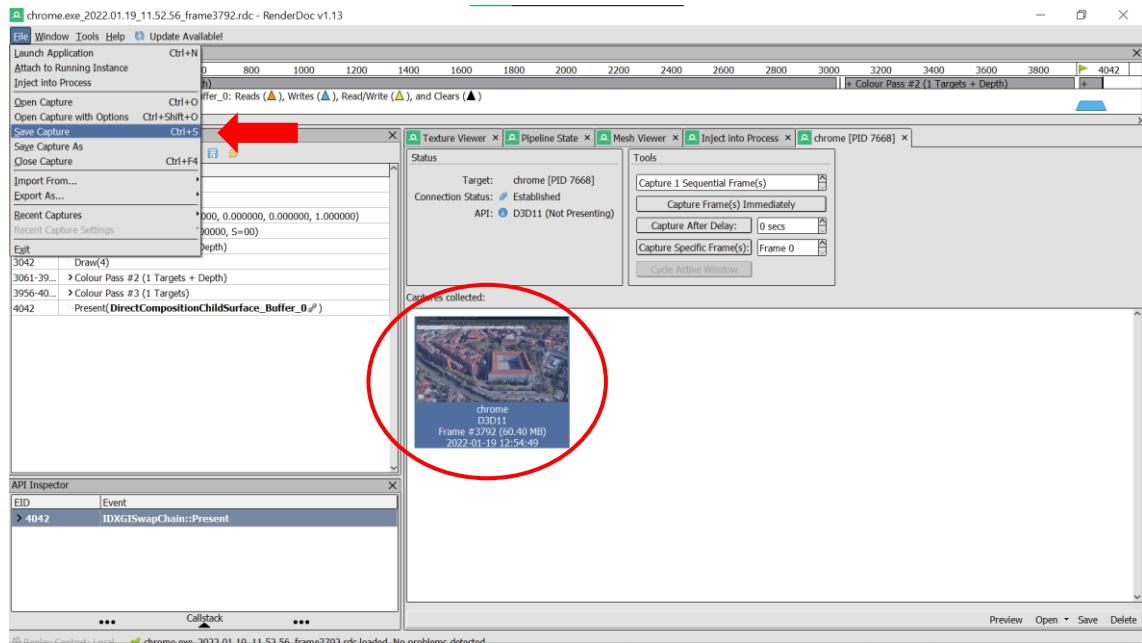


Figura 4.22: Captura obtenida en RenderDoc.

El último paso es abrir el programa Blender e instalar el complemento que permite abrir archivos de RenderDoc. Se importa la captura obtenida en RenderDoc y se obtiene el modelo 3D deseado.

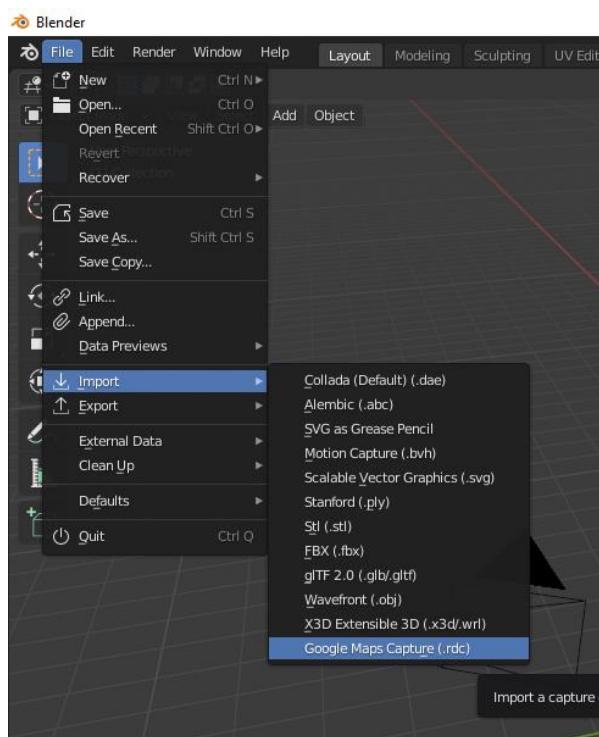


Figura 4.23: Importar modelo en Blender.

Al tener cargado el modelo 3D en Blender, solo falta guardar el modelo en formato “.dae” en una carpeta con todas las texturas que proporciona el programa. El uso de las texturas es opcional, su uso es meramente visual e implica un tiempo de computación mayor. Finalmente, para cargar el modelo en la simulación de gazebo hay que seguir los pasos previos explicados en el apartado 4.1.1.2 con relación a la creación de los archivos .sdf y .config.

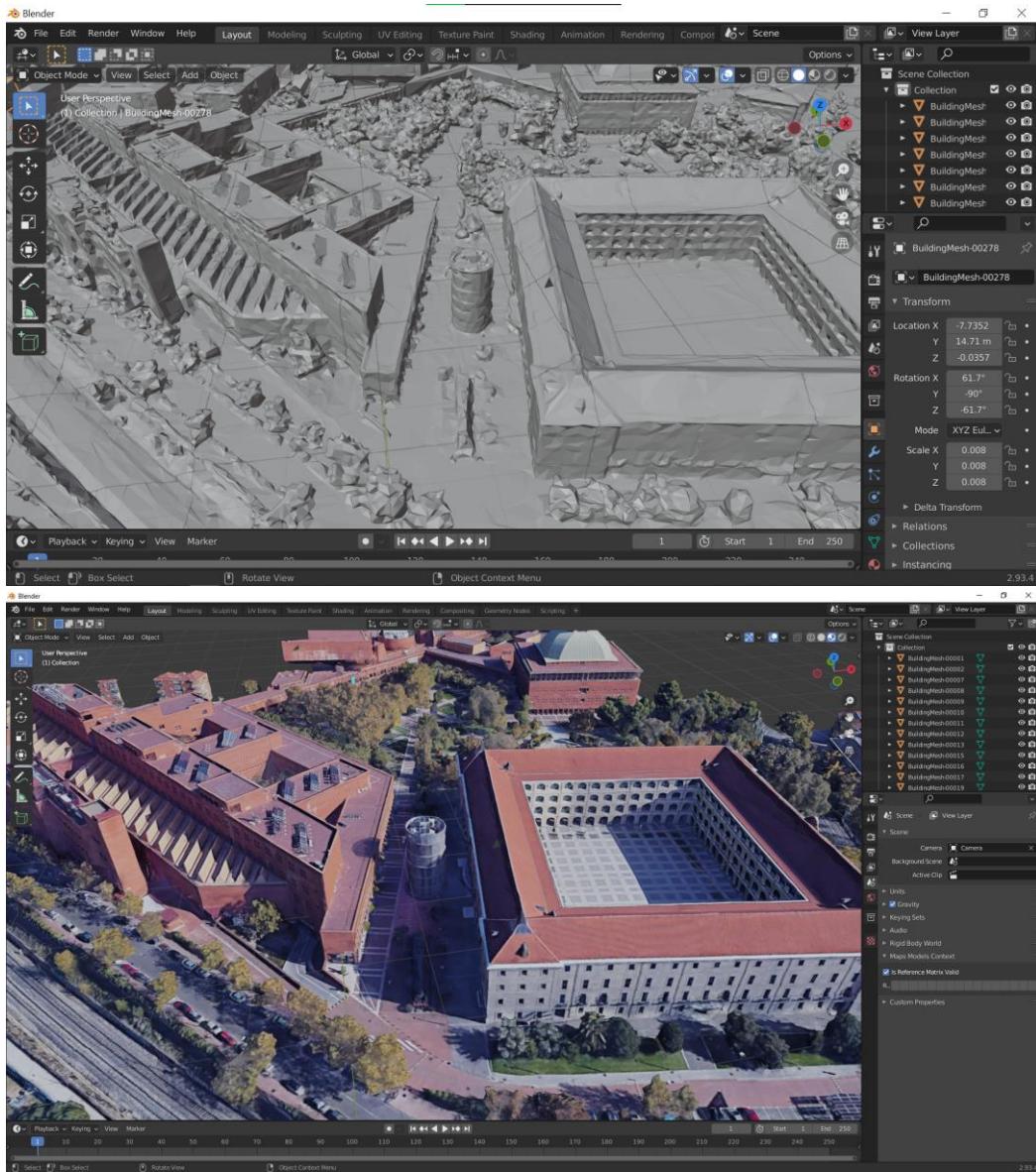


Figura 4.24: Modelo 3D obtenido en Blender.

A modo de conclusión, se ha considerado que el método que obtiene un modelo más realista es el obtenido mediante Google Maps y RenderDoc. Es un método sencillo y que facilita la representación de entornos para la simulación; por el contrario, los otros métodos resultaban más laboriosos y el resultado no era del todo satisfactorio. La calidad del entorno 3D obtenido (Fig.4.24) es excelente y la posibilidad de poder editarlo con Blender permite incorporar obstáculos al mapa para realizar las pruebas de evitación de obstáculos.

4.1.2. Modelos de dron en la simulación.

Para la simulación de este proyecto se han creado dos modelos de dron. Ambos modelos tienen los mismos componentes, diferenciándose en que uno tiene un sensor lidar y el otro una cámara de profundidad. Se ha querido realizar la simulación con estos dos vehículos para poder tener dos referencias en la toma de datos y estudiar el comportamiento en cada uno de ellos.

La base en los modelos de dron se obtiene directamente de un paquete de PX4 creado para importar modelos a Gazebo. Este modelo es un quadcopter basado en el modelo 3DR Iris, Fig. 4.25.



Figura 4.25: Comparación en los modelos de dron reales [34] y simulados.

El sensor utilizado en el primer modelo de dron cumple con las especificaciones dadas por el fabricante del lidar Benewake CE30-C, sensor que se utilizará en las pruebas reales. Para ello, se ha creado un modelo .sdf con los valores indicados en la hoja de especificaciones. A continuación se muestra el archivo que define los parámetros del lidar y su apariencia en Gazebo (Fig 4.26).

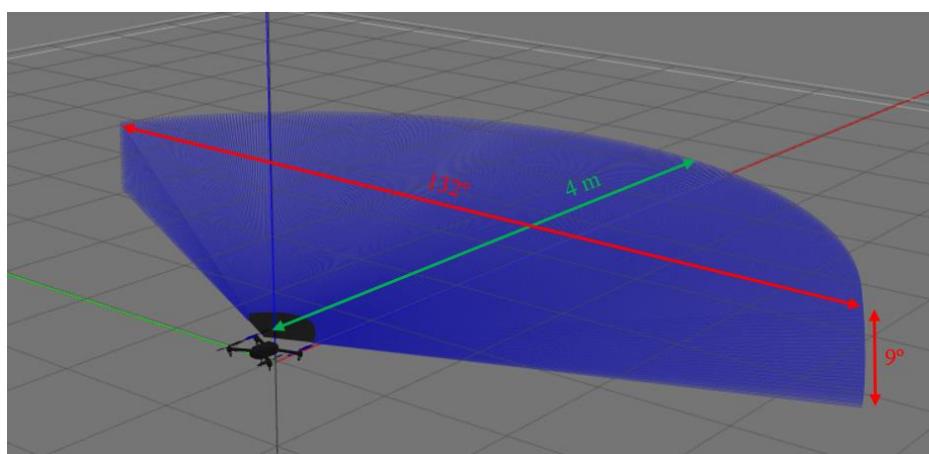


Figura 4.26: Representación del lidar Benewake en el entorno simulado.

lidar_benewake.sdf

```
<sdf version="1.5">
<model name="lidar_benewake"> #Nombre del modelo
<link name="link">

<inertial>
<pose>0 0 0 0 0 0</pose>
<mass>0.219</mass> #Peso del sensor
<inertia> #Valores predeterminados
<ixx>4.15e-6</ixx>
<ixy>0</ixy>
<ixz>0</ixz>
<iyy>2.407e-6</iyy>
<iyz>0</iyz>
<izz>2.407e-6</izz>
</inertia>
</inertial>

<sensor name='lidar_benewake' type='ray'> #Se define el tipo de sensor
<ray>
<scan>
<horizontal>
<samples>320</samples> #Muestras horizontales
<resolution>1</resolution>
<min_angle>-1.15192</min_angle> #Campo de visión horizontal en rad. (-66°)
<max_angle>1.15192</max_angle> #Campo de visión horizontal en rad. (+66°)
</horizontal>
<vertical>
<samples>16</samples> #Muestras verticales
<min_angle>0</min_angle> #Campo de visión vertical en rad. (0°)
<max_angle>0.15708</max_angle> #Campo de visión vertical en rad. (9°)
</vertical>
</scan>
<range> #Distancia en metros
<min>0.35</min> #Se aumenta la distancia mínima debido a la proximidad de las hélices
```

```

<max>4</max> #Distancia máxima

<resolution>0.01</resolution>

</range>
</ray>

<plugin name='3D_laser' filename='libgazebo_ros_velodyne_laser.so'> #Paquete compatible para
gazebo

<topicName>/lidar_benewake_scan</topicName>
<frameName>lidar_benewake_link</frameName>
<gaussianNoise>0.008</gaussianNoise> #Ruido para dar realismo

</plugin>
<always_on>1</always_on>
<update_rate>10</update_rate>
<visualize>1</visualize>
</sensor>

</link>
</model>
</sdf>

```

Finalmente, para unir la base del dron (iris.sdf) con el sensor (lidar_benewake.sdf), se ha creado un archivo (iris_lidar_benewake.sdf) que permite definir la posición del sensor en el dron y crear un modelo que englobe los dos componentes (Fig. 4.27).

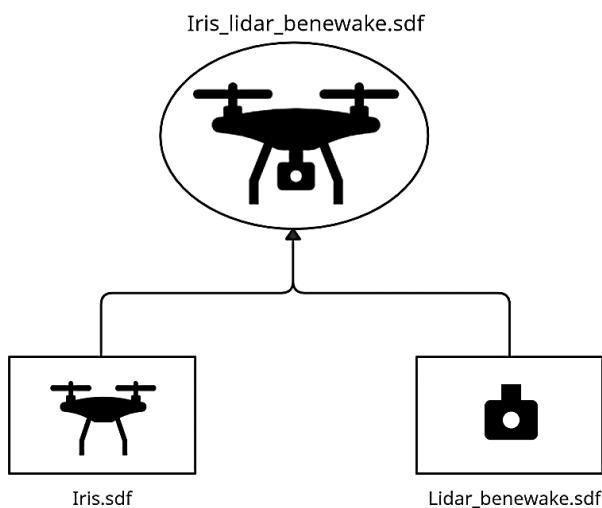


Figura 4.27: Gráfico de los componentes creados para el dron simulado.

El segundo modelo de dron utilizado incorpora una cámara de profundidad diseñada en base al modelo Intel Realsense. Este archivo .sdf viene incorporado en la carpeta de sitl_gazebo del Firmware PX4- Ha sido elegido en base a su rendimiento en las pruebas

de simulación y a su uso extendido dentro de la comunidad, facilitando la respuesta a dudas que han surgido durante la elaboración del proyecto. En las imágenes de la parte inferior (Fig. 4.28) se puede observar que el reconocimiento de objetos funciona de forma satisfactoria.

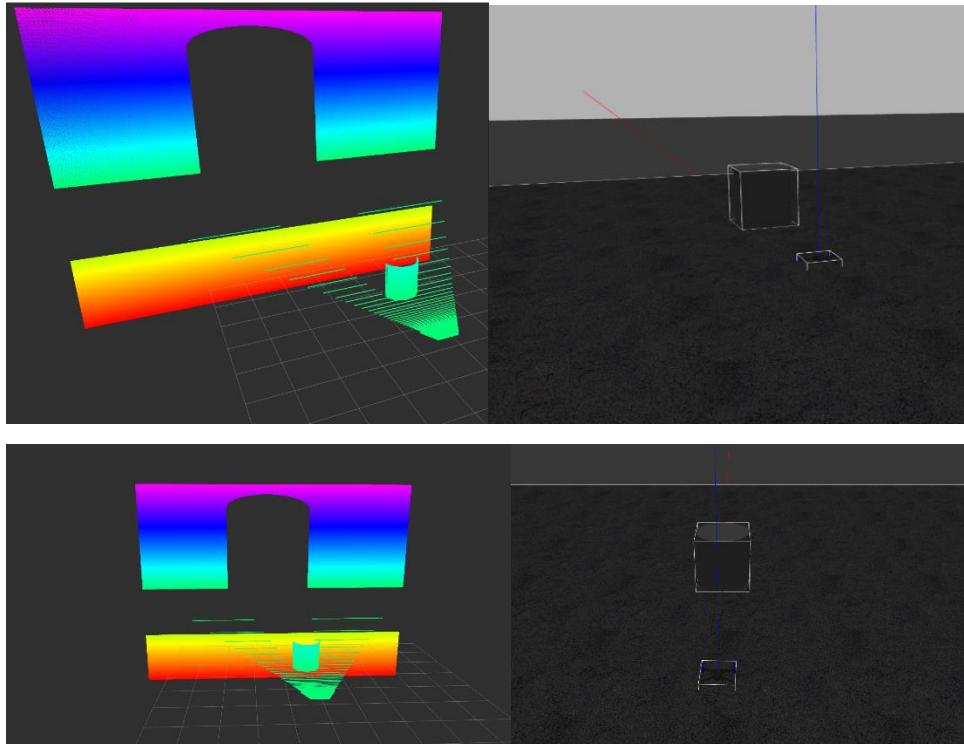


Figura 4.28: Cámara de profundidad en la simulación.

Cabe destacar que este modelo simplemente ha sido utilizado a modo de entrenamiento en la simulación para entender mejor el funcionamiento del programa. Debido a que las cámaras de profundidad suelen dar problemas en lugares exteriores, se ha decidido optar por la vía del sensor lidar. Otro inconveniente sería el tamaño y peso de este tipo de cámaras, imposibilitando su conexión a bordo del vehículo aéreo.

4.2. Funcionamiento del código.

Este proyecto sigue la metodología de código abierto y se puede encontrar en el repositorio de GitHub [dmeirap/tfm_dmeirap](https://github.com/dmeirap/tfm_dmeirap)¹. El funcionamiento para evitar obstáculos está basado en el estudio proporcionado por PX4 [35].

A la hora de volar un dron se busca reducir tres factores: el tiempo de vuelo, la energía utilizada y el riesgo de la trayectoria. Cuando se habla de distancia, se puede definir en base al tiempo y la energía utilizada en la trayectoria del camino. En el caso de los drones,

¹https://github.com/dmeirap/tfm_dmeirap

la energía y tiempo que se utiliza para desplazarse en el eje horizontal es equivalente; sin embargo, en el eje vertical se requiere más coste para subir que para bajar. Dicho esto, uno de los principales problemas a resolver en este trabajo es encontrar el camino más corto entre el punto inicial y el punto final de la misión. Este problema encontró respuesta en 1959 de la mano de Edsger Dijkstra [36]. En la actualidad existen algoritmos más complejos y con menor tiempo de resolución; sin embargo, lo que se busca en este caso es la simplicidad de código y su pequeño tiempo de computación. El algoritmo de Dijkstra se basa en un método de expansión de los nodos manteniendo siempre la mínima distancia con el nodo inicial.

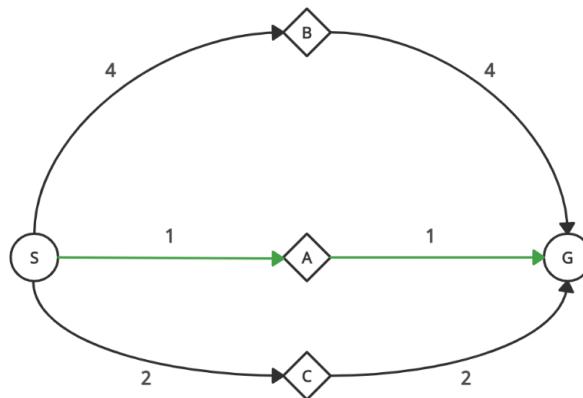


Figura 4.29: Camino más corto.

En la práctica, el camino más corto no siempre es el mejor camino. Hay que tener en consideración otros factores importantes como el riesgo de fallo que puede tener el dron en función del camino elegido; bien puede considerarse un riesgo la falta de batería como encontrarse el camino bloqueado.

El riesgo en la trayectoria es uno de los factores más complicados a calcular. Dado que las probabilidades dependen de la ocupación de los voxels dados por OctoMap, el dron podría llegar a navegar a través de un voxel ocupado sin necesidad de estrellarse por lo que puede interferir en la creación de trayectorias óptimas.

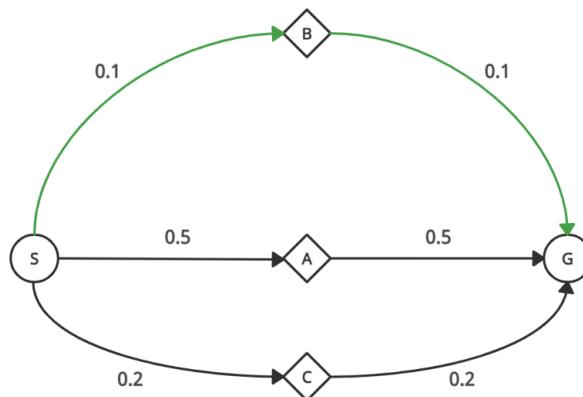


Figura 4.30: Camino con menos riesgo.

Por otro lado, se busca que el camino sea fluido y tenga el menor número de giros posibles. Al igual que con la distancia, una trayectoria fluida implica una reducción de tiempo y energía.

Es por todo esto por lo que el mejor camino posible intenta minimizar la distancia en la trayectoria y minimizar el riesgo de fallo.

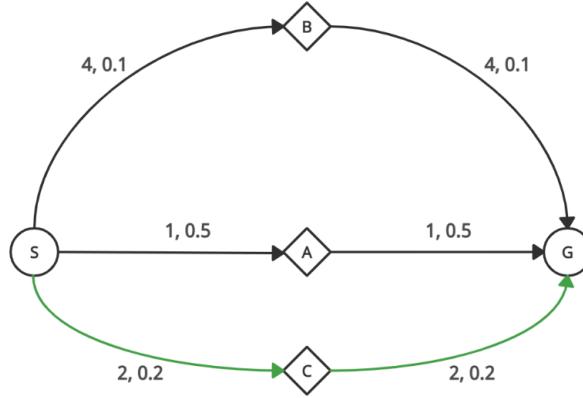


Figura 4.31: Camino óptimo.

Visto estos tres factores, este código intenta minimizar el coste usando heurísticas con el algoritmo A*. A diferencia de Dijkstra, en A* se tiene en consideración la distancia con el nodo final y se define en el código mediante el uso de heurísticas.

En el caso de la distancia, se hace uso de la ecuación de distancia Manhattan junto con movimientos diagonales, donde u es la posición actual y t es el objetivo. Los movimientos diagonales solo están permitidos en el eje horizontal.

$$h_d(u) = |u_x - t_x| + |u_y - t_y| + |u_z - t_z| - (2 - \sqrt{2}) \min(|u_x - t_x|, |u_y - t_y|)$$

En el caso del riesgo en la trayectoria, se toma como heurística tomar el camino más corto en un espacio desconocido. De esta forma, se asume el coste desconocido de todos los voxels que aparezcan en la trayectoria del camino más corto.

Para el caso de encontrar una trayectoria más fluida y teniendo en cuenta que la función de rotación es cuadrática, se ha decidido tomar giros de 45° para alcanzar el objetivo y, de este modo, minimizar el coste.

Las comunicaciones de este sistema siguen la estructura del grafo mostrado en la Fig. 4.32. Mientras que por un lado OctoMap y Gazebo cuentan con nodos específicos de ROS para enviar y recibir información, PX4 utiliza el nodo de Mavros para la comunicación.

Por otro lado, el nodo de global planner se centra en recibir la posición del dron, la posición final de la misión y la posición de los obstáculos gracias a los datos recibidos del lidar. Se crean nuevos caminos en función de la posición actual del dron y los posibles nuevos obstáculos. También se mantiene un registro del camino realizado hasta el momento por cuestiones de seguridad.

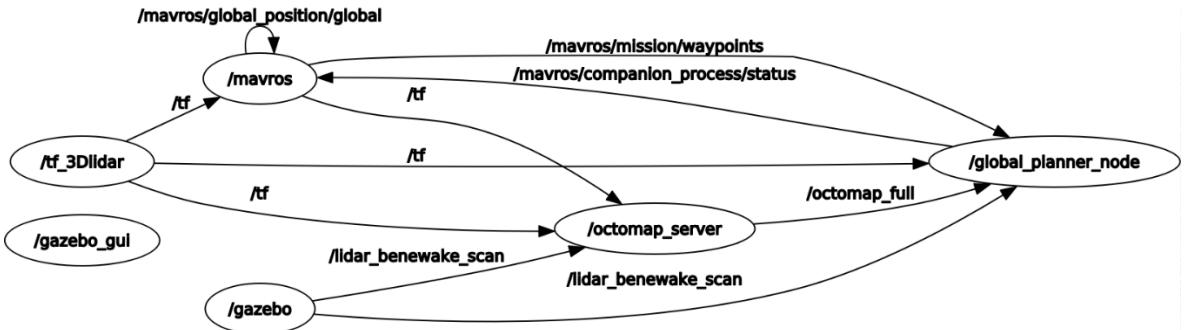


Figura 4.32: Grafo de comunicaciones.

4.3. Puesta en marcha del vehículo aéreo.

Para poner en funcionamiento el dron, es necesario que todos los componentes estén conectados de forma adecuada. A continuación, se aporta información imprescindible para cada componente.

- **Hélices:** Los motores de las hélices deben de estar en buen estado, asegurando que la energía suministrada a cada uno de ellos es proporcional y equitativa, ya que el fallo de uno de estos puede llevar a graves problemas de vuelo.

Por otro lado, es imprescindible tener en cuenta el sentido de las hélices, estas deben de coincidir en diagonal con su par (Fig. 4.33). Teniendo como referencia el sentido de la marcha del dron, la diagonal que va de la parte superior izquierda a la parte inferior derecha del dron debe de girar en sentido horario y la diagonal que va de la parte superior derecha a la parte inferior izquierda del dron debe de girar en sentido antihorario. Si estos no se cumplen, el dron no será capaz de levantar el vuelo.

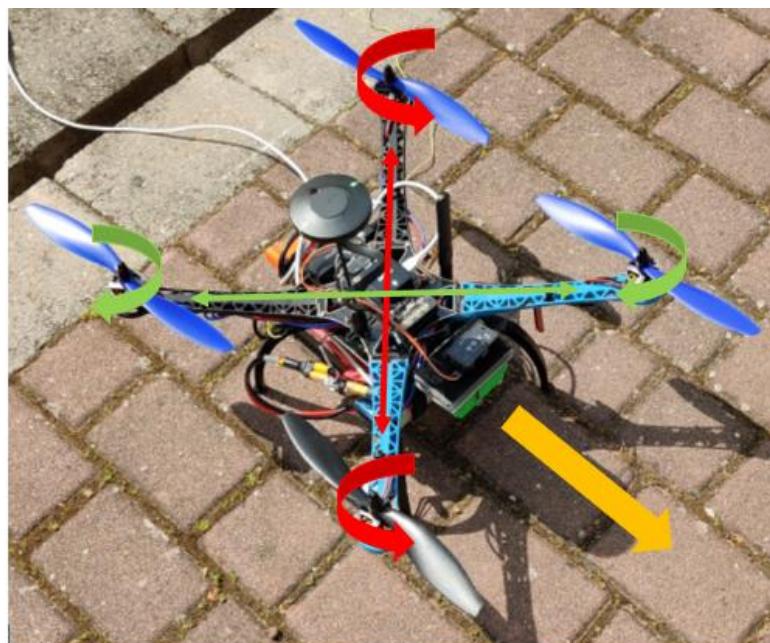


Figura 4.33: Sentido de giro de las hélices.

- **Pixhawk:** Para que el dron reciba la información de vuelo y la posición GPS de forma más precisa, es recomendable que la Pixhawk esté actualizada con el ultimo firmware de PX4 (v1.12.3); para ello, simplemente hay que conectar la Pixhawk directamente al ordenador, sin alimentación, mediante un cable micro-usb y abrir la pestaña de Firmware en QGroundControl y actualizar. Otra opción viable seria instalar ArduPilot en vez de PX4; sin embargo, los métodos de calibración resultan más tediosos.

Otro tema importante a tener en cuenta es la velocidad de transmisión del puerto serie que conecta la Pixhawk con la Jetson. Es necesario configurar la velocidad para solucionar problemas de comunicación que se han dado durante las pruebas. Para ello, dentro del programa QGroundControl, hay que ir a la pestaña de parámetros y ajustar la variable del parámetro SER_TEL1_BAUD a 921600 8N1. De este modo, se obtiene una comunicación más fluida y sin fallos en la conexión.

- **Batería:** Para configurar las opciones de la batería es necesario inicializar QGroundControl y seleccionar el icono de ajustes del vehículo >> ajustes de alimentación. En este caso, el número de celdas conectadas en serie en la batería es de 4 (suele aparecer escrito en la batería con la terminología 4S). El voltaje máximo por celda suele venir predeterminado y el valor asignado para las baterías de tipo LiPo es de 4,05 V. El voltaje seguro mínimo de cada celda en las baterías LiPo oscila entre 3,7 V ~ 3,0 V en función de cuan conservador se quiere ser con la batería. El divisor de voltaje y los amperios por voltio deben de ser establecidos después de realizar una medición con un polímetro. En la Fig. 4.34 se puede apreciar los parámetros de la batería en el dron utilizado.

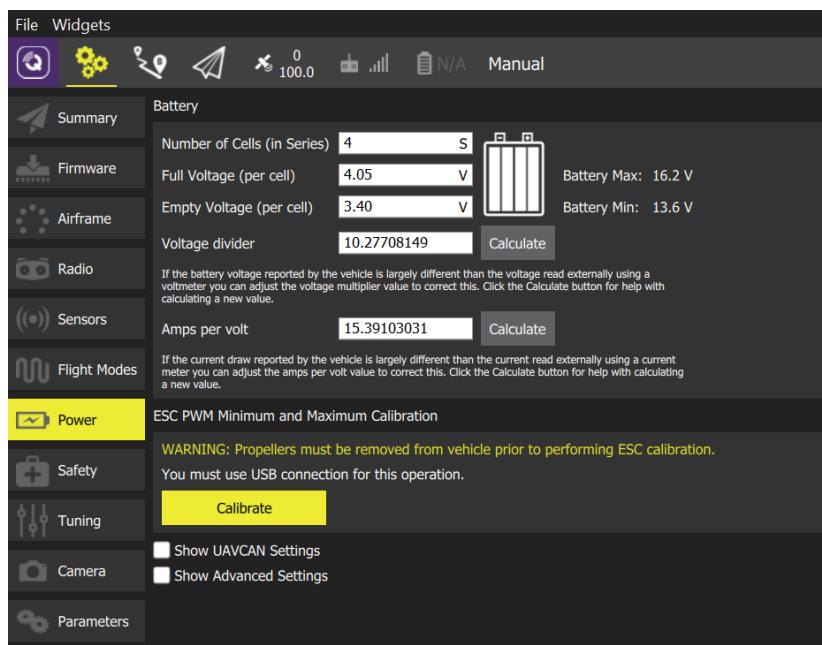


Figura 4.34: Configuración de la batería.

- **GPS:** El módulo GPS que se utiliza incluye una serie de sensores para mejorar la estabilidad de los drones, entre ellos compás, giroscopio y acelerómetro. Es necesario configurar bien su orientación en QGroundControl siendo una parte imprescindible de la calibración la posición de la flecha que aparece señalizada en el GPS como se puede observar en la Fig. 4.35. Esta flecha debe de apuntar siempre hacia delante, en sintonía con la rotación respecto al eje z (yaw) de la Pixhawk, que en este caso tiene un ángulo de 0° y se debe definir en QGroundControl como ROTATION_NONE.

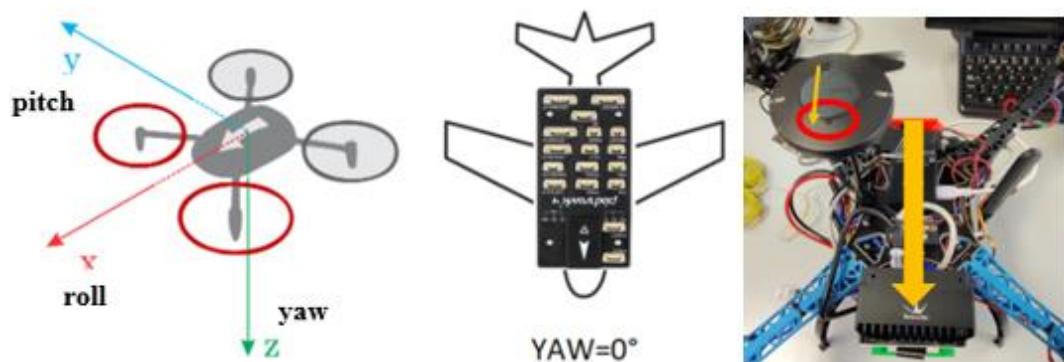
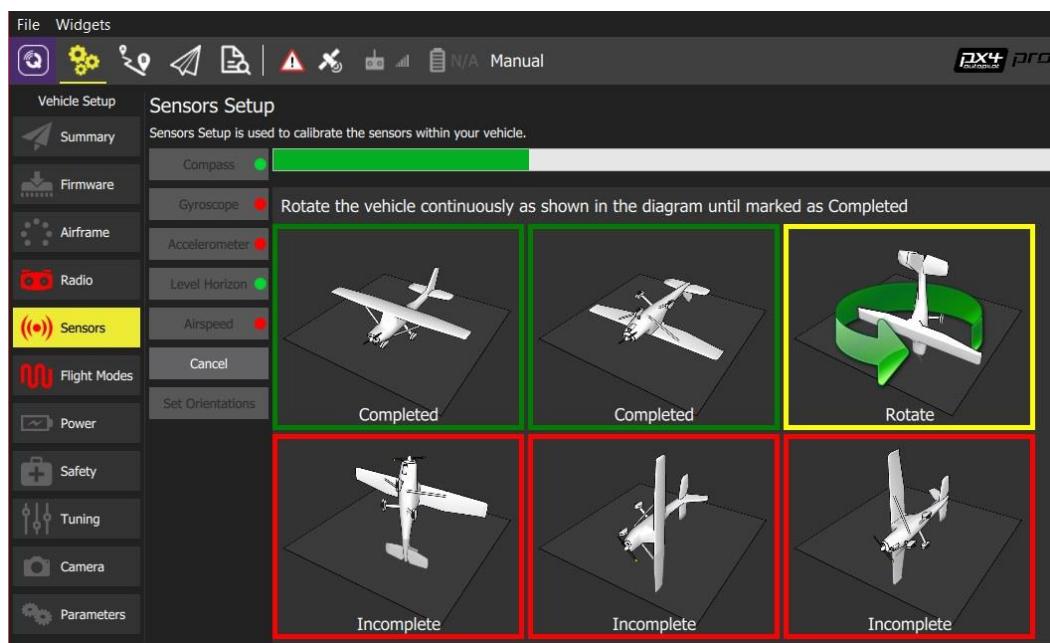


Figura 4.35: Calibración Pixhawk.

- **Lidar:** El sensor Lidar Benewake CE30-C requiere de una alimentación de 5V proporcionada por la batería y una conexión directa con la Jetson mediante ethernet; debido a un golpe del dron en una de las pruebas, la conexión ethernet de la Jetson quedó inutilizada por lo que se ha utilizado un adaptador tipo-C/ethernet como solución. Es necesario configurar la dirección IP para poder recibir la información del sensor.



Figura 4.36: Configuración IP lidar.

Una vez está conectado el lidar de forma correcta, se puede comprobar su correcto funcionamiento a través del programa RViz en el cual se puede apreciar la nube de puntos en tiempo real tal y como se muestra en la Fig. 4.37.

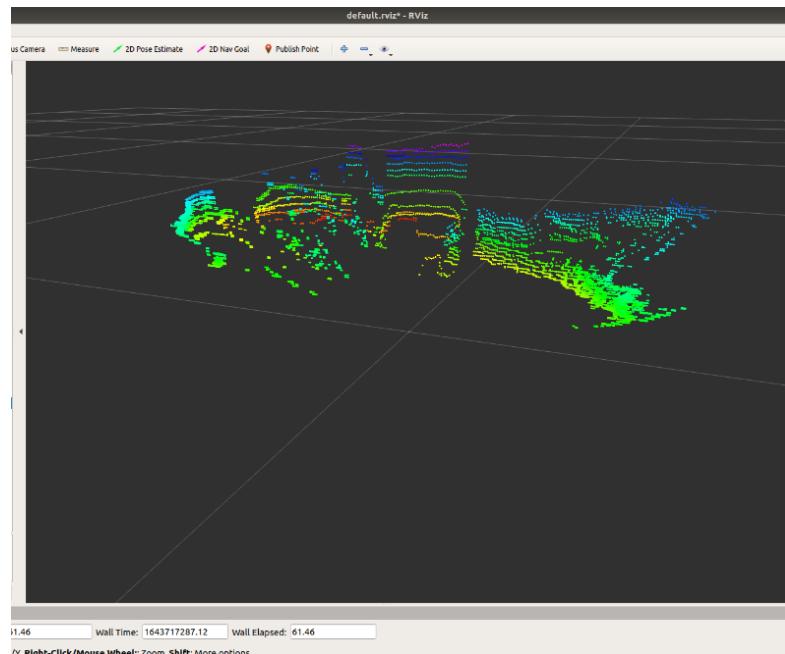


Figura 4.37: Nube de puntos del sensor lidar.

- **Antena Wifi:** Se ha añadido una antena WiFi al dron a mayores de la que viene incluida con la Jetson debido a la mala conexión que se obtuvo al realizar las pruebas reales. Esta antena está conectada a través de un hub usb al navegador de abordo. Para su instalación y correcto funcionamiento hay que seguir una serie de pasos:

Terminal

```
cd /usr/src/linux-headers-x.x  
sudo make scripts  
  
cd catkin_ws/src/tfm/driver/  
git clone https://github.com/brektrou/rtl8821CU.git  
cd rtl8821CU  
make  
sudo make install
```

Una vez que la antena Wifi está instalada correctamente es necesario configurar la misma, para que al encender la Jetson Nvidia, se conecte automáticamente con la red WiFi que genera el ordenador principal. En la Fig. 4.38 se puede apreciar la información correspondiente a dicha conexión.

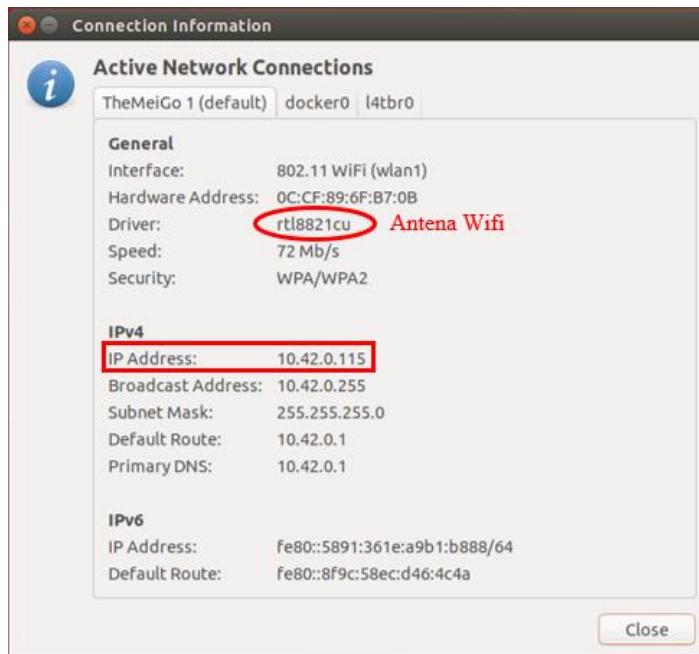


Figura 4.38: Conexión Wifi.

Una vez se tienen todos los componentes del dron revisados y listos para despegar, es necesario mandar de alguna forma la información a la Jetson para que ejecute el código deseado. En esta ocasión, el método utilizado para conectar el dron con el ordenador en tierra ha sido mediante Wifi. Para ello, el ordenador genera una red Wifi local a la que se

conecta el dron mediante un archivo .sh que permite controlar todo lo que está haciendo el dron en ese mismo momento desde el ordenador y sin necesidad de ningún cable que dificulte el vuelo; en la Fig. 4.38, se puede apreciar la IP que necesitamos para ese archivo .sh mencionado con anterioridad. Como punto en contra, quiero destacar que la distancia entre el dron y el ordenador no puede ser muy elevada debido a posibles fallos de conexión.

Una vez se tiene realizada la conexión, es necesario abrir tres terminales para ejecutar el código.

Terminal 1: Hay que dar permiso a la conexión de la Pixhawk con la Jetson por el puerto /dev/ttyUSB0 y lanzar el .launch que activa el código.

Terminal 1

```
sudo chmod 777 /dev/ttyUSB0  
~/catkin_ws/src/tfm/real/launch  
roslaunch real_planner.launch
```

Terminal 2: Para poder ejecutar los scripts en el dron, es necesario definir la dirección del puerto con relación a mavsdk_server. Debe estar definido en el script como:

```
drone = System(mavsdk_server_address="localhost", port=50051)
```

Terminal 2

```
cd ~/.local/lib/python3.6/site-packages/mavsdk/bin  
./mavsdk_server -p 50051 serial:///dev/ttyUSB0:921600
```

Terminal 3: Una vez realizado todos los pasos previos, solamente falta ejecutar la misión de vuelo deseada.

Terminal 3

```
~/catkin_ws/src/tfm/scripts  
python3 mision.py
```

5. PRUEBAS Y RESULTADOS.

En este apartado se discuten las pruebas y los resultados que se han obtenido a lo largo de las simulaciones de vuelo y las pruebas con el dron real en el complejo universitario.

5.1. Fase simulada.

En este proyecto se ha hecho uso principal de un mapa 3D de la UC3M, junto con otras ubicaciones emblemáticas de la ciudad, para crear las simulaciones de vuelo tal como se puede observar en el video² donde se recopilan algunas de dichas simulaciones. En este apartado se describe únicamente la simulación en el entorno de la universidad ya que es el lugar donde también se realizan los ensayos de la fase real. Cabe destacar que algunos entornos de simulación han sido modificados añadiendo objetos de diferentes dimensiones con el objetivo de forzar al dron a evitar obstáculos en la trayectoria.

El primer paso que llevar a cabo en la simulación es incluir en el archivo `~/.bashrc` las coordenadas (Latitud, Longitud, Elevación) de la posición de inicio de la trayectoria. Una forma de comprobar que las coordenadas han sido introducidas de forma correcta es abrir el programa QGroundControl, una vez lanzado el `.launch` de la simulación, y comprobar que la posición del dron en el mapa es la deseada (Fig. 5.1).

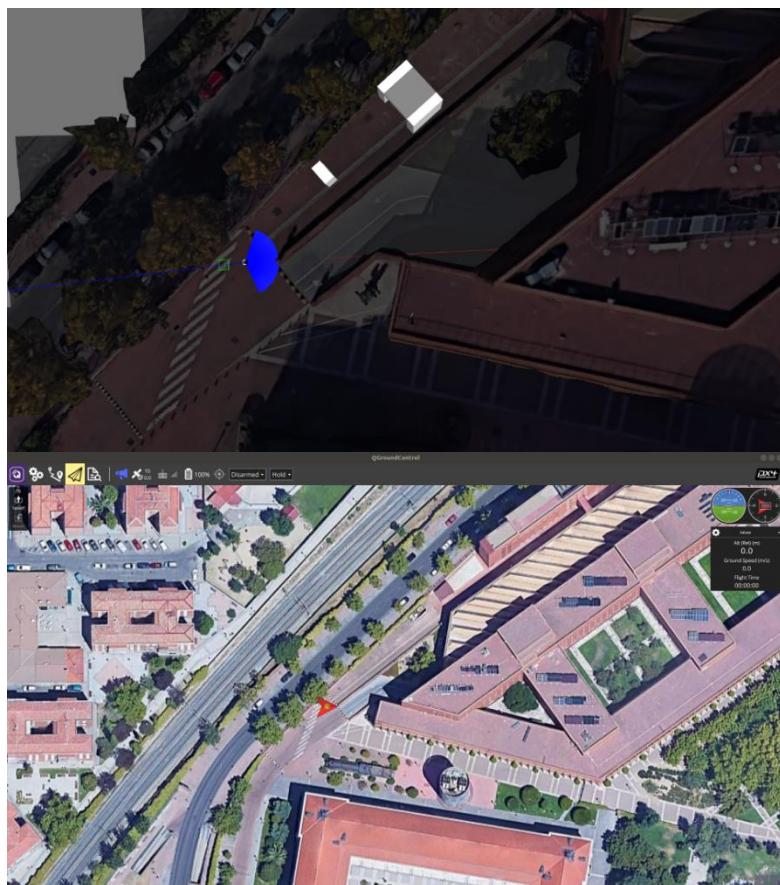


Figura 5.1: Ubicación del dron en la simulación.

²<https://www.youtube.com/watch?v=fwy4yhDFsR0>

Una vez se tiene el mapa cargado en el entorno de simulación Gazebo y el dron está situado en la posición del mapa deseada, es hora de lanzar el script de la misión. Es un código sencillo que basa su funcionamiento en crear una trayectoria de un punto inicial a un punto final con la particularidad de poder ir añadiendo puntos en el camino, definidos mediante coordenadas. A medida que el dron va avanzando en su trayectoria cabe la posibilidad de que se encuentre con algún obstáculo en el camino y tengo que recalcular la trayectoria desde su posición actual al siguiente punto de paso. Dentro de este código también se tiene en cuenta la altura máxima a la que se desea que vuele el dron, la velocidad máxima y el coste en el riesgo que puede tomar en las decisiones de evitar obstáculos.

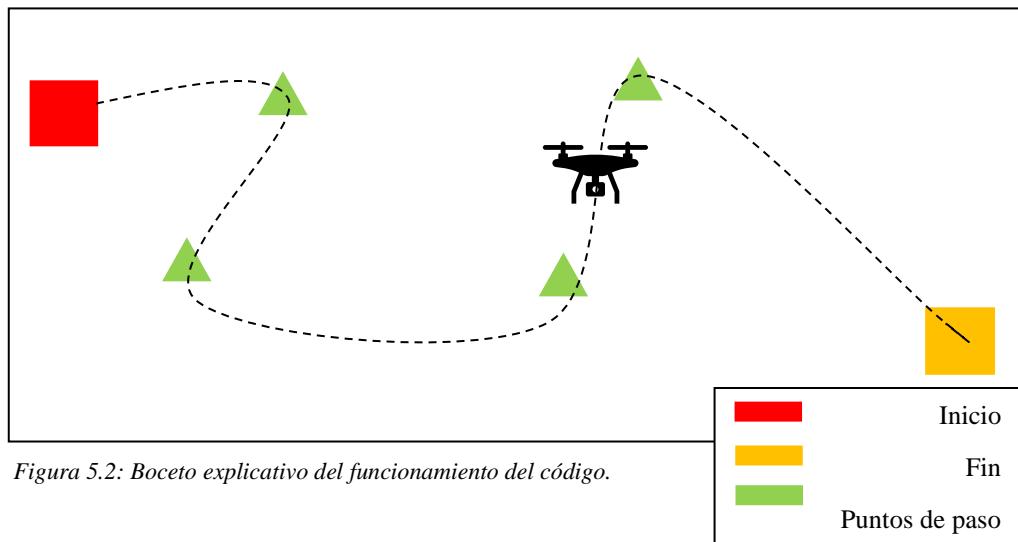


Figura 5.2: Boceto explicativo del funcionamiento del código.

En las imágenes de la Fig. 5.3, se puede observar el resultado obtenido en la simulación del dron con el sensor lidar. Como resultado, y debido al poco rango del sensor (explicado en el punto 4.1.2), el mapa local de ocupación no aporta una información muy detallada del entorno; sin embargo, el dron es capaz de llegar del punto inicial al punto final sin problemas después de tener que desviarse de la trayectoria debido a varios obstáculos que han ido apareciendo en el camino. El dron siempre busca moverse en un plano horizontal evitando un gasto mayor de energía. La velocidad en la trayectoria tiene que reducirse, en comparación con otras cámaras y sensores que se pueden encontrar en el mercado, para que el dron tenga el tiempo suficiente de recalcular su senda en el caso de que se encontrarse con un objeto por el camino. Por otra parte, es posible reducir el tiempo de computación a costa de aumentar el riesgo de choque si se aumenta el tamaño de los voxels del OctoMap; se debe de testear una medida de voxel que proporcione la trayectoria más eficiente.

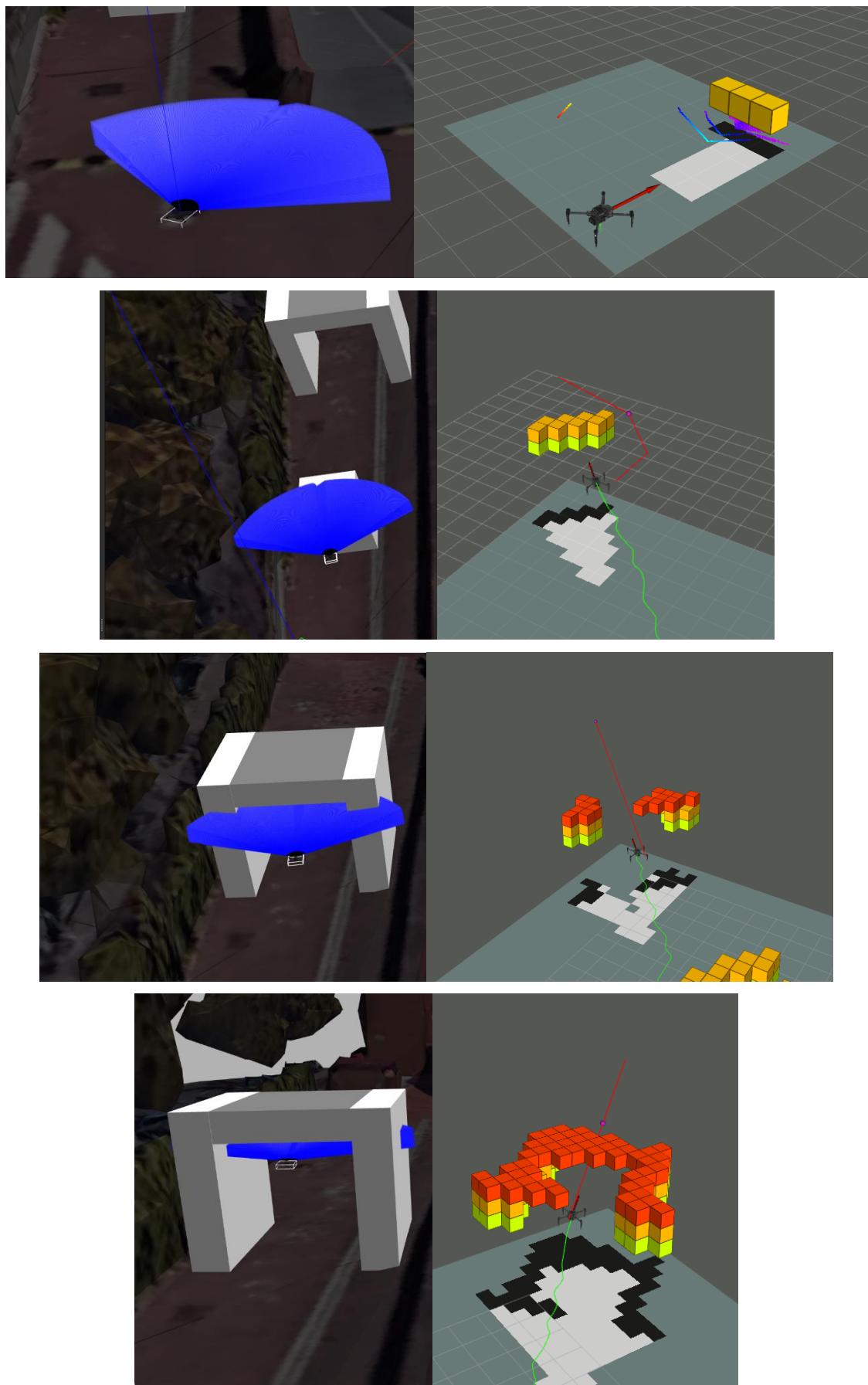


Figura 5.3: Simulación dron con sensor lidar.

Recopilando toda la información obtenida en la trayectoria de la misión del dron, se obtiene el mapa local de ocupación mostrado en la Fig. 5.4. Se puede observar como el dron detecta correctamente los obstáculos (circulo morado y azul), generando una trayectoria alternativa, en función de su posición actual y la posición final, a medida que va detectando los obstáculos.

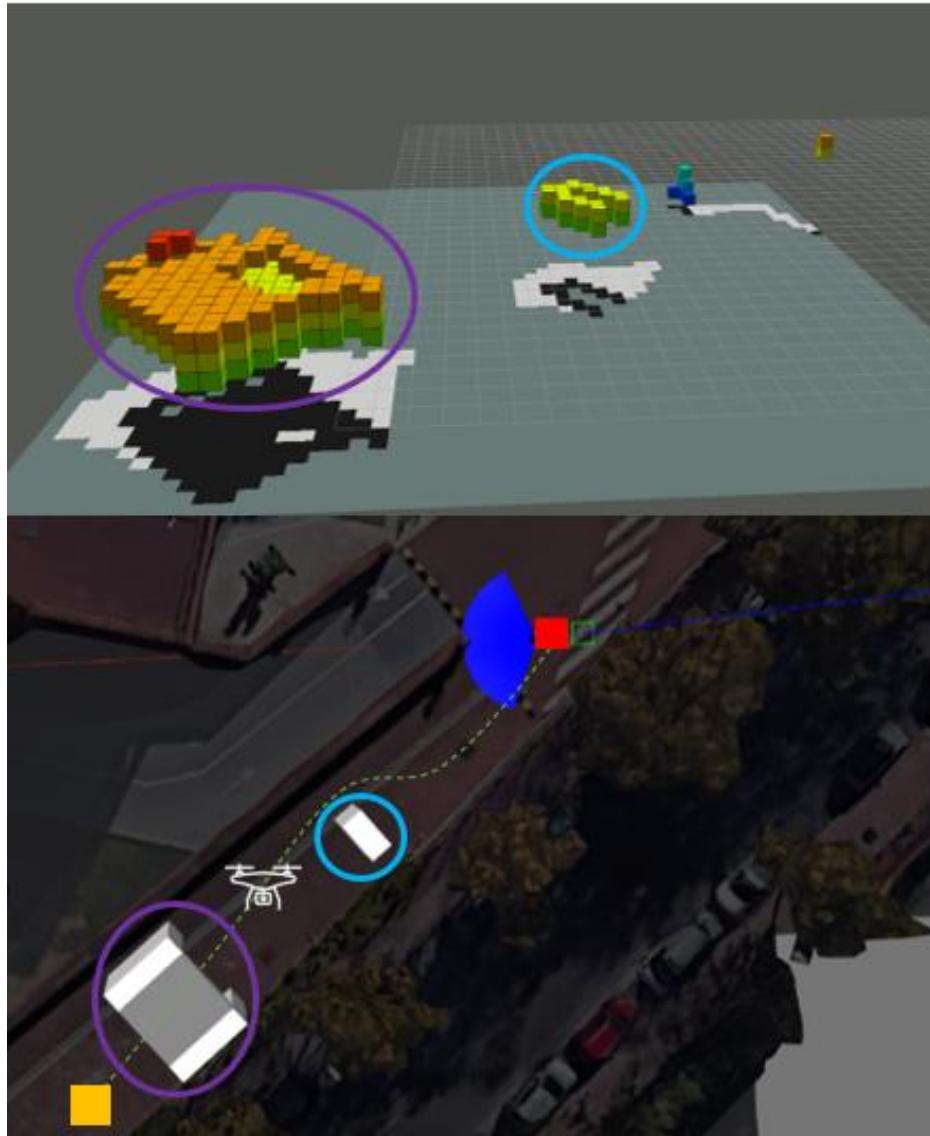


Figura 5.4: Mapa local del dron con sensor lidar incorporado.

A modo de comparación, en la Fig. 5.5, se puede observar el mapa local de ocupación obtenido con el dron que lleva a bordo una cámara de profundidad. La conclusión que se obtiene en esta comparación es que el vuelo autónomo mejora en función del alcance del sensor/cámara, ayudando a prevenir con antelación la posición de los objetos que obstruyen la trayectoria y reducir el tiempo de vuelo, acortando distancias y ahorrando en energía.

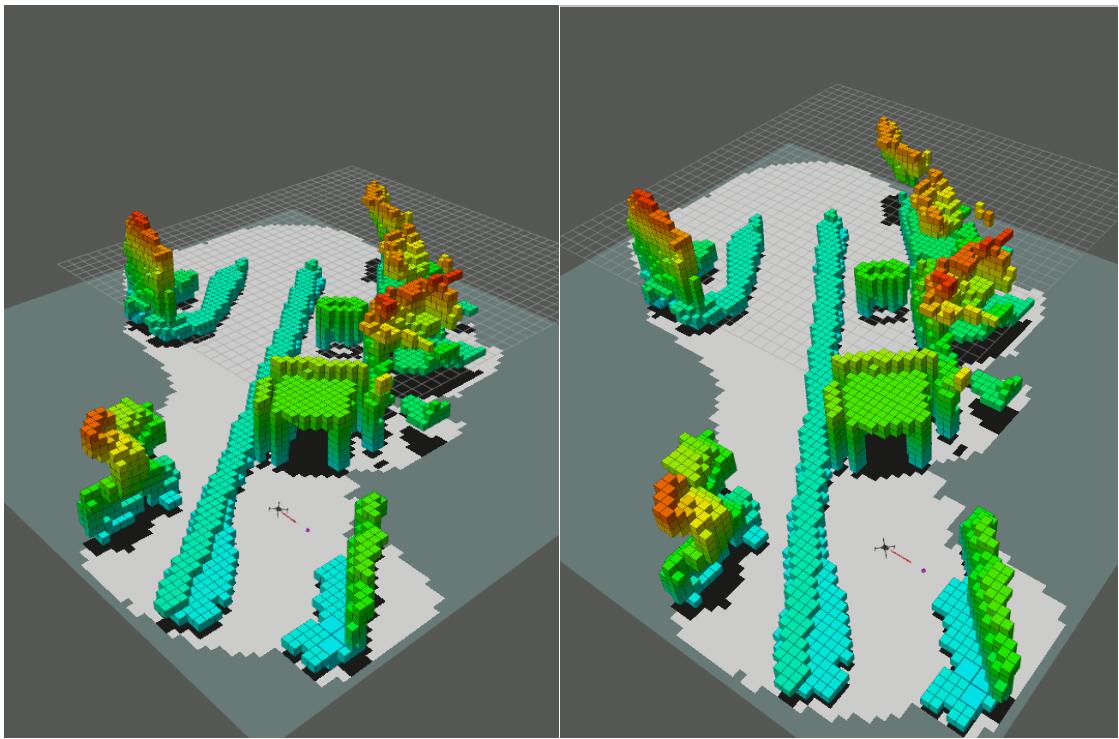


Figura 5.5: Mapa de ocupación del dron con cámara de profundidad.

5.2. Fase real.

A la hora de realizar las pruebas con el dron real han surgido varios problemas con relación a diversos componentes del vehículo.

El primer paso por llevar a cabo para comprobar el correcto funcionamiento del vehículo es conectar el dron mediante usb al ordenador de tierra y lanzar el programa QGroundControl. Una vez abierto el programa, se puede observar la disponibilidad de satélites que hay en ese preciso momento; en el caso de que el número de satélites sea menor de diez, la controladora de vuelo no permite ejecutar ninguna misión de vuelo a modo de seguridad y con el fin de tener información precisa de la localización del dron.

En este video³ se puede observar como el dron es capaz de realizar un armado y un despegue de forma satisfactoria, lanzando las instrucciones desde QGroundControl. Cabe destacar que, para poder volar el dron, las condiciones meteorológicas deben de ser favorables y con cielos despejados para poder obtener el mayor número de satélites y, por ende, una mayor precisión en la ubicación del vehículo.

³<https://www.youtube.com/watch?v=YduHo5d-01w>

Una vez comprobado el correcto funcionamiento del dron con QGroundControl, se procede a conectar el dron con el ordenador terrestre mediante una red wifi generada por el propio ordenador, tal y como se ha explicado en el apartado 4.3. Es en este punto donde empiezan a aparecer los problemas de comunicación que impiden realizar las pruebas con el dron. Antes de lanzar el script de la misión, se ha decidido lanzar instrucciones más básicas, tales como el armado y el despegue, para comprobar el funcionamiento de estas.

Sin embargo, cada vez que se ha lanzado el script que sigue las instrucciones del modelo mostrado en la Fig. 5.6, han surgido errores, en relación con la conexión, en diferentes líneas del código. Cada vez que se lanza el script el error aparece en una línea diferente y no se sabe en qué momento puede surgir el error.

Este es un gran problema ya que si el código se para mientras el dron esté despegando, la única vía posible para aterrizar el dron es desconectar manualmente la batería. Una tarea que pone en peligro la seguridad de la persona que esté realizando las pruebas y que se debe evitar a toda costa. En la Fig. 5.7 se pueden observar los fallos en la conexión que han ido surgiendo al realizar las pruebas.

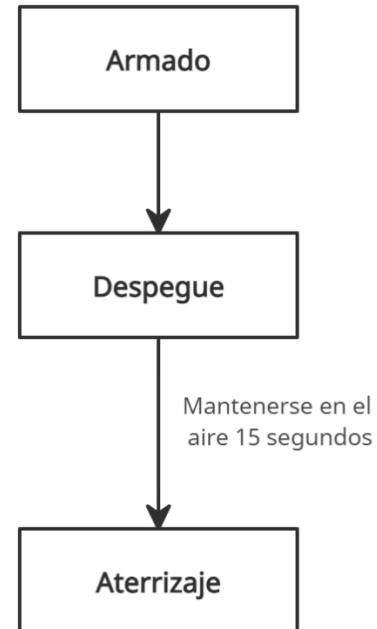


Figura 5.6: Grafo órdenes del código.

```

nvidia@jetson-0421419020930:~/catkin_ws/src/tfm/scripts$ python3 takeOff.py
Waiting for mavsdk_server to be ready...
Connected to mavsdk_server!
Esperando la conexion del dron...
Dron conectado!
Esperando la posicion global del dron...
Posicion_global estimada OK
-- Arming
-- Taking Off
Traceback (most recent call last):
  File "takeOff.py", line 38, in <module>
    loop.run_until_complete(run())
  File "/usr/lib/python3.6/asyncio/base_events.py", line 484, in run_until_complete
    return future.result()
  File "takeOff.py", line 28, in run
    await drone.action.takeoff()
  File "/home/nvidia/.local/lib/python3.6/site-packages/mavsdk/action.py", line 300, in takeoff
    raise ActionError(result, "takeoff()")
mavsdk.action.ActionError: CONNECTION ERROR: 'Connection Error'; origin: takeoff(); params: ()
nvidia@jetson-0421419020930:~/catkin_ws/src/tfm/scripts$ python3 takeOff.py
Waiting for mavsdk_server to be ready...
Connected to mavsdk_server!
Waiting for the conexion
Dron connected
Waiting for the global position
Global_position estimate
Arming
Taking Off
Landing
Traceback (most recent call last):
  File "takeOff.py", line 38, in <module>
    loop.run_until_complete(run())
  File "/usr/lib/python3.6/asyncio/base_events.py", line 484, in run_until_complete
    return future.result()
  File "takeOff.py", line 33, in run
    await drone.action.land()
  File "/home/nvidia/.local/lib/python3.6/site-packages/mavsdk/action.py", line 322, in land
    raise ActionError(result, "land()")
mavsdk.action.ActionError: CONNECTION ERROR: 'Connection Error'; origin: land(); params: ()
  
```

Figura 5.7: Errores de conexión.

A modo de solución, se ha abordado este problema con diferentes métodos que se explican a continuación.

- Dado al número elevado de usos que ha sido sometido el dron y, por ende, a los diversos accidentes que han podido surgir en las pruebas, se ha decidido cambiar el chasis del dron junto con la controladora de vuelo y los motores; dejando únicamente la Jetson como elemento del vehículo anterior.
- Se ha incorporado una antena wifi a mayores de la que incorpora el dron abordo, tal y como se muestra en el apartado 4.3., con el fin de obtener una mejor conexión entre el ordenador de tierra y la Jetson. Se ha realizado un ping entre la estación de tierra y el vehículo y se ha observado que no hay perdida de paquetes, por lo que se descarta que este sea la raíz del problema.

```
... 10.42.0.115 ping statistics ...
402 packets transmitted, 402 received, 0% packet loss, time 401781ms
rtt min/avg/max/mdev = 1.033/1.730/19.541/1.288 ms
```

Figura 5.8: Ping entre GCS y el dron.

- Se ha reseteado la Jetson y se han vuelto a descargar únicamente los paquetes necesarios para el correcto funcionamiento.
- Se ha cambiado la velocidad del puerto /dev/ttyUSB0 a 921600 con el propósito de tener una mayor velocidad a la hora de recibir la información de la controladora PX4 en la Jetson.
- Se ha hecho uso de diferentes GPS del mismo modelo que incorpora la Pixhawk con el fin de comprobar si el error reside en los datos proporcionados de la ubicación. Se han calibrado en repetidas ocasiones con el fin de mejorar la precisión en la localización del vehículo y no se han obtenido los resultados deseados.

En las últimas semanas, debido a sucesos que se desconocen, el número de satélites no ha llegado a alcanzar el mínimo para poder volar el dron de forma segura, tal y como se puede apreciar en el siguiente video⁴ por lo que no se han podido realizar las pruebas pertinentes para asegurar el correcto funcionamiento del vuelo autónomo del dron.

Parece ser que el error reside en las comunicaciones entre Mavros y PX4. Mientras que en la simulación Mavros envía constantemente la posición global del dron, en las pruebas reales se aprecian constantes perdidas de este tipo de mensajes. El código, al depender de la posición global del dron, requiere de una comunicación sin perdidas. En el caso de este proyecto las pérdidas pueden surgir entre la comunicación WiFi del dron y el GCS, debido a que cuanto mayor es la distancia que separa a estos dos componentes menor es la señal y, por ende, empiezan a surgir pérdidas en la comunicación.

Sin embargo, en contra de todo pronóstico, la última semana antes de entregar este proyecto, surgió una nueva posibilidad de llevar a cabo las pruebas. Se trata de la incorporación de un GPS adicional (Fig. 5.8), el cual formaba parte de un dron DJI que se estaba utilizando en el laboratorio LSI de la UC3M. Gracias a este nuevo componente se ha conseguido obtener una mejor señal en la posición del vehículo, llegando a alcanzar la cifra de 20 satélites y evitando en gran medida las pérdidas en las comunicaciones. .

⁴<https://www.youtube.com/watch?v=WBslr6P2txU>



Figura 5.9: GPS U-Blox.

Los resultados de las pruebas después de acoplar este nuevo GPS se pueden ver recogidas en el video⁵. En un primer vistazo se puede apreciar que el dron es capaz de realizar el armado de forma satisfactoria por medio del código aplicado y mencionado con anterioridad. A la hora de realizar el despegue también ha respondido de una buena forma llegando a aterrizar sin ningún problema. Estas dos pruebas implican que el código utilizado en la simulación es aplicable al dron con el que se han realizado las pruebas. Aunque sean menos frecuentes, todavía siguen apareciendo los fallos de comunicación entre el dron y el GCS; sin embargo, se ha podido apreciar que el código responde bien a una puesta en marcha real y en base a las simulaciones realizadas previamente se puede considerar que el dron cumple con el objetivo de realizar una navegación autónoma y es capaz de evitar obstáculos.

A la hora de lanzar la misión que define la trayectoria del vehículo aéreo se ha optado por planificarla en QGroundControl y evitar que el dron se quede suspendido en el aire debido a fallos en la comunicación. Lo bueno de este método es que se pueden controlar directamente las acciones desde el programa y marcar los puntos de paso directamente sobre el mapa. Lo malo es que el dron necesita estar conectado mediante un cable USB al GCS y pasa a ser un dron cautivo creando una dependencia al cable.

A la par que se lanza la misión se lanza el .launch que permite al sensor lidar crear un mapa de ocupación y detectar los obstáculos que se encuentra al seguir la trayectoria establecida.

A continuación, se muestra una secuencia de imágenes (Fig. 5.10) que reflejan el vuelo del dron en una misión. Esta misión está basada en despegar el dron tres metros del suelo, avanzar hasta un punto fijado previamente por coordenadas, retorno a la posición inicial y aterrizaje del vehículo.

⁵<https://www.youtube.com/watch?v=AmqmRQt43Ws>

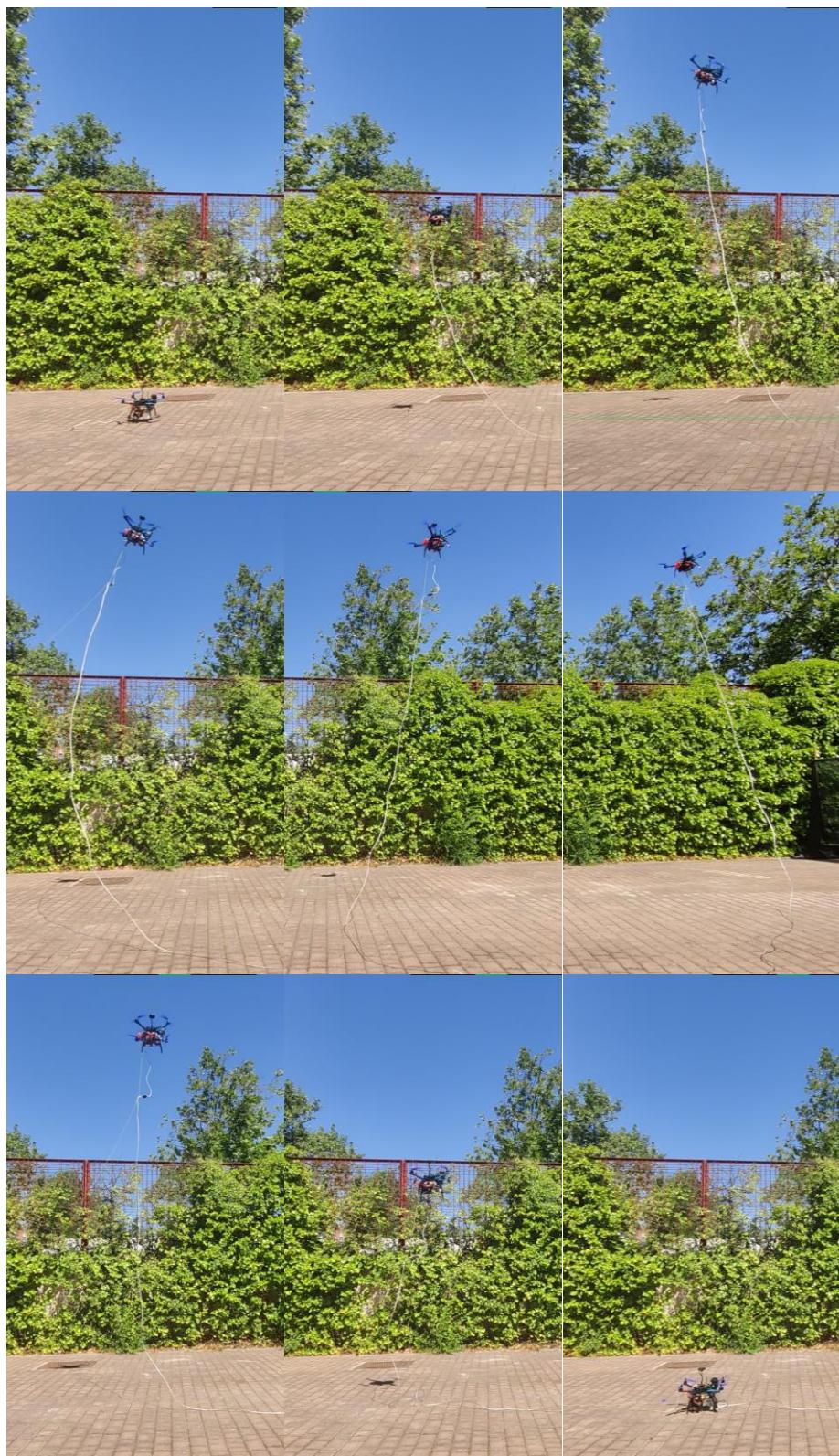


Figura 5.10: Secuencia de vuelo en una misión.

6. CONCLUSIONES.

En base a las pruebas realizadas, el mapeado y evitación de obstáculos para el vuelo autónomo de un dron es una opción viable y a tener en cuenta de cara a su implementación en los diferentes sectores. Sin embargo, en este proyecto han surgido varios problemas que pueden dar a una nueva línea de investigación con el fin de perfeccionar este método.

Uno de los mayores problemas que ha surgido en este trabajo han sido los continuos fallos de comunicación a la hora de querer volar el dron real en el complejo de la Universidad, como ya se ha mencionado en el apartado 5.2.

Otro tema a tener en cuenta es si el uso del sensor lidar Benewake CE30-C es el indicado para este tipo de misiones. Si bien es cierto que cumple con los objetivos básicos, genera dudas en cuanto al rango del sensor y su capacidad para crear un mapeado local del entorno. Mientras que en las simulaciones se puede apreciar como el dron con la cámara de profundidad es capaz de crear un mapa local más complejo y recorrer la trayectoria en menos tiempo gracias a su mayor rango de distancia, el sensor lidar requiere que el dron viaje a menor velocidad ya que detecta los obstáculos más tarde y en menor medida. En este punto surge un dilema ya que la cámara de profundidad ha sido testada únicamente en un entorno simulado, sin tener en cuenta su mayor desventaja que es operar en sitios abiertos con luz natural; otro punto que considerar son las dimensiones y pesos de este tipo de cámaras dificultando su montaje a bordo del vehículo aéreo.

A modo de conclusión, quiero destacar que la generación de trayectorias y la evitación de obstáculos en este proyecto funcionan de forma satisfactoria. El dron usado para los experimentos quizás no ha sido el indicado debido a los continuos fallos de comunicación que han surgido en las pruebas. Por otro lado, el sensor a bordo del dron ha cumplido el objetivo de la misión, pero tiene margen de mejora en cuanto al rango de captura. Una propuesta de cara a mejorar el rendimiento del vehículo sería cambiar la controladora de vuelo del dron y disponer de un nuevo sensor con mayor rango.

6.1. Trabajos futuros.

De cara a implementar lo puesto en práctica en este proyecto, una futura línea de investigación podría basarse en mejorar los problemas de comunicación que se han tenido a la hora de conectar el dron. Un método interesante podría ser prescindir del GPS y basar la localización mediante la fusión de mapas locales y globales. También se podría probar a implementar diferentes sensores y cámaras para hacer una comparación entre el peso-precio-precisión.

Por otro lado, como consecuencia del aumento de la demanda en la industria de los drones, ha surgido el concepto de los enjambres de drones. Una futura línea de investigación podría estar centrada en la comunicación entre drones. Se discutiría cual es el método de comunicación más seguro y la formación que adoptaría dicho enjambre. Estos enjambres pueden facilitar todavía más las cosas, ya que permiten completar tareas complejas de una forma más eficiente y con un menor coste, especialmente en entornos difíciles.

A la hora de hablar sobre los enjambres de drones, cabe destacar la amplia variedad de métodos de sincronización que se pueden utilizar entre ellos y el gran número de aplicaciones a las que pueden brindar un servicio.

A continuación, se muestran una serie de métodos de sincronización que se han llevado a cabo en diferentes estudios.

- **Redes ad hoc:** El funcionamiento de las redes ad hoc tiene como característica que una persona se encarga de pilotar un dron (líder) mientras que el resto de la flota (seguidores) sigue al líder a través de señales Wi-Fi y de forma completamente autónoma. El estudio mostrado en [37], concluye que el uso de las redes ad hoc permite sincronizar drones sin la necesidad de incluir sensores específicos; sin embargo, los datos obtenidos no son lo suficientemente precisos ya que los drones están a merced del piloto y las formaciones que han implementado en el software no son suficientes para prevenir situaciones complicadas en el exterior.
- **FSO:** El método FSO (Free Space Optical) se usa para mejorar la detección y la toma de decisiones en profundidad de problemas topológicos (por ejemplo, fuga de petróleo en un barco) donde es necesario una gran calidad de imagen. En los drones industriales el paso de la información se realiza a través de señales de radio frecuencia; sin embargo, para obtener esa resolución de imagen que se busca, es necesario utilizar un mayor ancho de banda. En el estudio mostrado en [38], se hace uso del método FSO en un enjambre en forma de V y se optimiza el número de drones necesarios en función de la calidad de imagen requerida y el área topológica.
- **GPS:** Este método ha sido desarrollado y probado en el artículo [39], en el cual se ha hecho uso del dron Parrot AR junto con el software de código abierto Paparazzi que permite el auto pilotaje de drones. El Parrot AR cuenta con la ventaja de llevar incorporados una gran cantidad de sensores, tales como dos cámaras, barómetro, acelerómetro y el más destacado en este caso, el módulo GPS. En este artículo han conseguido que los drones sean capaces de volar de forma autónoma a base de simples cambios en el software.
- **Fast Marching Square (FM2):** Este método permite planificar trayectorias suaves y seguras sin la aparición de mínimos locales, creando una función temporal dependiendo de las condiciones del entorno y la propagación de las ondas. En el artículo [40], se presenta una solución para gestionar las formaciones de los UAV utilizando el método de Fast Marching Square. Se basa en una configuración líder-seguidor en el que se busca que la trayectoria mantenga una geometría definida y pueda adoptar otras formas para superar diversos obstáculos.
- **Redes inalámbricas:** En el artículo [41], se hace uso de las redes inalámbricas 3G/4G convencionales para abarcar el problema de la comunicación en tiempo real entre los enjambres de drones. De este modo, encuentran una solución adaptativa y económica para cubrir las grandes áreas de vuelo.

- **MAVLink (Micro Air Vehicle Link):** Mencionado con anterioridad, es un servicio de código abierto que permite la comunicación entre drones (y sus componentes) mediante el paso de mensajes muy ligeros.

Un buen ejemplo es el que se muestra en [42], en el cual se hace uso de MAVLink y el simulador QGroundControl para controlar a un enjambre de drones destinados a monitorizar la contaminación en el aire que genera la industria sin necesidad de poner en riesgo la vida de los trabajadores y alcanzando zonas inaccesibles para el ser humano. Por otro lado, en [43] se puede observar cómo se hace uso de MAVLink para comunicar a los drones como dispersar los pesticidas y fertilizantes necesarios para controlar la agricultura y así evitar problemas de salud a los trabajadores.

7. PRESUPUESTO Y MARCO REGULADOR.

Este apartado está dedicado a mostrar el presupuesto de este proyecto, al estudio del alcance social y económico que puede alcanzar y el marco regulador correspondiente.

7.1. Presupuesto.

El presupuesto de este proyecto se realiza en base a la mano de obra que realiza el estudiante y el coste, en el marco actual, del material necesario para llevar a cabo las pruebas.

7.1.1. Mano de obra.

A la hora de calcular el coste de la mano de hora, es necesario hacer un desglose de las horas dedicadas en cada parte del proyecto como se muestra en la tabla 7.1.

	Cantidad [días]	Cantidad [horas/día]	Total [horas]
Estudio	20	6	120
Análisis	8	6	48
Diseño	12	6	72
Desarrollo	60	6	360
Pruebas	30	6	180
Memoria	20	6	120
			Total: 900

Tabla 7.1: Cantidad de horas totales del proyecto.

Para calcular el coste total de la mano de obra, se ha realizado una búsqueda del sueldo medio de un ingeniero junior en España (25.358 €) y el número de horas laborales (1725 horas/año); obteniendo una media aproximada de 15 €/hora.

	Cantidad [horas]	Sueldo [€/h]	Total [€]
Trabajo Final de Máster (TFM)	900	15,00	13.500,00

Tabla 7.2: Coste de mano de obra.

7.1.2. Material.

El inventario de componentes se ha realizado en base a todos los elementos que componen el dron y los útiles necesarios para comprobar su funcionamiento y crear la simulación.

Componentes	Cantidad [ud.]	Coste total [€]
NVIDIA Jetson AGX Xavier	1	573,00
Pixhawk 2 The Cube Black	1	239,00
AX-2810Q-750KV	4	20,40
HERE 3 Módulo GPS GNSS M8P Pixhawk 2	1	168,00
Benewake-CE30-C	1	1.145,00
Approx APPC43 Adaptador USB-C a RJ45	1	21,87
Batería Lipo Gens ace 5000mAh batería de 11.1V	1	55,90
BrosTrend Linux Adaptador WiFi USB 650Mbps	1	22,99
Chasis Quadcopter S500	1	32,95
Hélice 1045	4	5,00
Matek System UBEC DUO	1	22,00
ESC T-Motor F55A	1	110,90
U-blox ANN-MB 00	1	60,00
Asus Vivobook S14 Intel Core i7, 16GB de RAM	1	999,00
Monitor Samsung LF27T350FHRXEN	1	139,00
Teclado Logitech K120	1	16,99
Ratón Logitech M90	1	6,99
		Total: 3.638,99

Tabla 7.3: Coste total de los componentes del proyecto.

7.1.2. Presupuesto final.

El presupuesto final es la suma del coste de la mano de obra junto con el coste de todos los componentes. La cifra obtenida al realizar este proyecto asciende a 17.138,99 €. Teniendo en cuenta la versatilidad de funciones que puede llegar a tener el dron en los diferentes sectores, se considera un coste asequible y amortizable a corto plazo.

7.2. Impacto socioeconómico.

Se espera que en Europa el uso de drones profesionales va a sufrir un gran ritmo de crecimiento, llegando a alcanzar los 400.000 vehículos aéreos no tripulados en el año 2035.

Si bien es cierto que en los próximos años la demanda de drones se centrará en los niveles más bajos del espacio aéreo (máx. 150 metros de altura), para el año 2035 la mayoría de UAV operarán lejos del alcance visual del piloto. Por otro lado, el uso recreativo de drones espera una mayor demanda que el ámbito profesional como consecuencia a una mayor flexibilidad en las leyes de la seguridad aérea.

Se estima que el impacto económico de este sector en 2035 será de 10.000 millones de euros al año y llegando a alcanzar los 14.600 millones de euros en 2050. Debido a esta situación, este sector sufrirá una gran demanda en la búsqueda de puestos de trabajo.

En territorio español se espera que para 2035 el número de aeronaves alcance las 51.400 unidades, lo que implica un impacto económico de 1.220 millones de euros al año, generando aproximadamente 11.000 puestos de trabajo.

Los sectores que sufrirán el mayor impacto se resumen en los comentados a continuación:

- **Agricultura:** España es uno de los países de la unión europea con mayor producción agrícola. Los drones se usarían principalmente en tareas como el control de crecimiento de los cultivos, el transporte y la pulverización.
- **Energía:** Teniendo en cuenta que en España hay 400.000 Km de tendidos eléctricos y 1.255 emplazamientos de generación de energía, se espera que los drones tengan un papel importante a la hora de inspeccionar estas redes de distribución eléctrica y los diferentes tipos de generadores de energía.
- **Seguridad y salvamento:** Los drones son y serán objeto de apoyo para tareas de vigilancia y salvamento en casos de prevención e incidentes. Se estima que, en el año 2035 los cuerpos de policía cuenten con un dron por cada cuatro patrullas y los cuerpos de bomberos con un dron cada dos vehículos.
- **Mensajería:** Actualmente, en España hay una media anual de 645 millones de pedidos y los tiempos de entrega son cada vez más exigentes. De cara a este problema, en parte debido a las pérdidas de tiempo en el uso de vehículos terrestres, se ha encontrado una solución en los vehículos aéreos no tripulados. En la actualidad, el uso de drones para mensajería se encuentra en desarrollo, pero no hay duda de que se convertirá en una realidad no muy lejana.
- **Construcción y minería:** Los drones tendrán un papel importante a la hora de ayudar a desarrollar estudios topográficos del terreno, al control de explotaciones mineras y al control en las obras de construcción.
- **Telecomunicaciones:** Al igual que en el apartado de la energía, los drones se enfocarán en la inspección de torres de comunicaciones y al cableado de las redes de comunicación

- **Movilidad:** De vistas al futuro, se está desarrollando en una fase inicial el uso de UAV para el transporte de personas. No cabe duda de que este es uno de los sectores con mayor capacidad de desarrollo y beneficio.

7.3. Marco regulador.

Actualmente en España el organismo encargado de regular el uso de vehículos aéreos no tripulados es la Agencia Estatal de Seguridad Aérea (AESPA). El uso de drones sigue la normativa europea que incluye el Reglamento de Ejecución (RE) (UE) 2019/947 [44] con los cambios pertinentes del RE (UE) 2020/639 [45], del RE (UE) 2020/746 [46] y del RE (UE) 2021/1166 [47] en relación con el uso de las aeronaves en cualquier ámbito y sin distinción entre usuarios. Por otro lado, incluye el Reglamento Delegado (RD) (UE) 2019/945 [48] con los cambios pertinentes del RD (UE) 2020/1058 [49] con relación al diseño y fabricación de UAV y su comercialización.

Si el dron no supera el peso de 250 gramos y no contiene una cámara, no es necesario que el piloto se registre en AESPA. Los requisitos necesarios para poder volar un dron de uso recreativo son los siguientes:

- El dron tiene que estar al alcance de la vista del piloto.
- El dron tiene que volar con una altura máxima de 120 metros.
- Se tiene que mantener una distancia mínima de 8 kilómetros de distancia con espacios aéreos controlados.
- El dron debe de llevar a bordo una placa identificativa con los datos correspondientes.
- Si el dron porta una cámara, las personas que puedan aparecer deben de quedar protegidas por el derecho a la intimidad.

En caso de que la zona de vuelo se encuentre dentro de una zona urbana, el dron no podrá pesar más de 10 Kg y deberá de contar con un sistema de amortiguación de caídas. Si el vuelo se realiza en horas nocturnas, el dron debe de disponer de elementos que faciliten su visibilidad.

Por otra parte, quiero destacar la obligatoriedad de que el piloto tenga en regla un seguro de responsabilidad civil frente a terceros de acuerdo con los artículos 11 y 127 de la Ley de Navegación Aérea [50]. En el caso de que el dron tenga un peso igual o inferior a 20 Kg deben de ajustarse al Real Decreto 37/2001 [51]. En el caso de que el dron supere los 20 Kg de peso, debe de ajustarse al reglamento (CE) 785/2004 [52].

BIBLIOGRAFÍA

- [1] Schroth, Lukas. (2020). The drone market size 2020-2025: 5 key takeaways. En: Drone Industry Insights [en línea] Disponible en: <https://droneii.com/the-drone-market-size-2020-2025-5-key-takeaways> [consulta: 01/06/2022].
- [2] Finance. Los Drones Comerciales están Revolucionando las Operaciones Comerciales. En: Finance [en línea] Disponible en: <https://www.toptal.com/finance/market-research-analysts/los-drones-comerciales-estan-revolucionando-las-operaciones-comerciales> [consulta: 4/02/2021].
- [3] BCG. (2017). Drones Go to Work. En: BCG [en línea] Disponible en: <https://www.bcg.com/publications/2017/engineered-products-infrastructure-machinery-components-drones-go-work> [consulta: 4/02/2021].
- [4] Gobierno de España. (2020). Plan de impulso de la cadena de valor de la industria de la automoción. [en línea] Disponible en: https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/transportes/Documents/2020/15062020_PlanAutomocion2.pdf [consulta: 1/06/2022].
- [5] Garcia-Aunon, P., Roldán, J. J., & Barrientos, A. (2019). Monitoring traffic in future cities with aerial swarms: Developing and optimizing a behavior-based surveillance algorithm. *Cognitive Systems Research*, 54, 273-286.
- [6] Roldán-Gómez, J. J., González-Gironda, E., & Barrientos, A. (2021). A Survey on Robotic Technologies for Forest Firefighting: Applying Drone Swarms to Improve Firefighters' Efficiency and Safety. *Applied Sciences*, 11(1), 363.
- [7] Aznar, F., Sempere, M., Pujol, M., Rizo, R., & Pujol, M. J. (2014, January). Modelling oil-spill detection with swarm drones. In Abstract and Applied Analysis (Vol. 2014). Hindawi.
- [8] Lomonaco, V., Trotta, A., Ziosi, M., Avila, J. D. D. Y., & Díaz-Rodríguez, N. (2018). Intelligent drone swarm for search and rescue operations at sea. arXiv preprint arXiv:1811.05291.
- [9] COCOMA-ORTEGA, Jos Arturo; MARTINEZ-CARRANZA, J. A cnn based drone localisation approach for autonomous drone racing. En Proceedings of the 11th international micro air vehicle competition and conference, Madrid, Spain. 2019.
- [10] AMER, Karim, et al. Deep convolutional neural network based autonomous drone navigation. En Thirteenth International Conference on Machine Vision. International Society for Optics and Photonics, 2021. p. 1160503.
- [11] ABDULMAJUID, Ahmed, et al. GPS-Denied Navigation Using Low-Cost Inertial Sensors and Recurrent Neural Networks. arXiv preprint arXiv:2109.04861, 2021.
- [12] AL-RADAIDEH, Amer; SUN, Liang. Self-Localization of Tethered Drones without a Cable Force Sensor in GPS-Denied Environments. *Drones*, 2021, vol. 5, no 4, p. 135.
- [13] TANAKA, Hideyuki; MATSUMOTO, Yoshio. Autonomous Drone Guidance and Landing System Using AR/high-accuracy Hybrid Markers. En 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE). IEEE, 2019. p. 598-599.
- [14] LEE, Seoungjun; HAR, Dongsoo; KUM, Dongsuk. Drone-assisted disaster management: Finding victims via infrared camera and lidar sensor fusion. En 2016 3rd

Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE). IEEE, 2016. p. 84-89.

- [15] AMER, Karim, et al. Convolutional neural network-based deep urban signatures with application to drone localization. En Proceedings of the IEEE International Conference on Computer Vision Workshops. 2017. p. 2138-2145.
- [16] Ministerio de transportes, movilidad y agenda urbana. Plan estratégico para el desarrollo del sector civil de los drones en España. En: Ministerio de transportes, movilidad y agenda urbana [en línea] Disponible en: <https://www.fomento.gob.es/NR/rdonlyres/7B974E30-2BD2-46E5-BEE5-26E00851A455/148411/PlanEstrategicoDrones.pdf> [consulta: 4/02/2021].
- [17] Nova. 1970s Firebee 1241 (Israel). En: Nova [en línea] Disponible en: https://www.pbs.org/wgbh/nova/spiesfly/uavs_11.html [consulta: 01/06/2022].
- [18] General Atomics. (2022). Predator C Avenger En: General Atomics [en línea] Disponible en: <https://www.ga-asi.com/remotely-piloted-aircraft/predator-c-avenger> [consulta: 01/06/2022].
- [19] Benavidez, Juan Carlos. (2020). El primer Heron TP para Alemania fabricado por IAI realiza su primer vuelo. En: Zona Militar [en línea] Disponible en: <https://www.zona-militar.com/2020/07/27/el-primer-heron-tp-para-alemania-fabricado-por-iai-realiza-su-primer-vuelo> [consulta: 01/06/2022].
- [20] General Atomics. (2022). MQ-9B SkyGuardian / SeaGuardian En: General Atomics [en línea] Disponible en: <https://www.ga-asi.com/remotely-piloted-aircraft/mq-9b> [consulta: 01/06/2022].
- [21] DJI. (2022). Matrice 300 RTK. En: DJI [en línea] Disponible en: <https://www.dji.com/es/matrice-300> [consulta: 01/06/2022].
- [22] Parrot. Professional drones built for work. En: Parrot [en línea] Disponible en: <https://www.parrot.com/en/drones> [consulta: 01/06/2022].
- [23] PX4. Cube Flight Controller. En: PX4 [en línea] Disponible en: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk-2.htm [consulta: 11/02/2021].
- [24] Airelectronics. U-Pilot. En: Airelectronics [en línea] Disponible en: https://www.airelectronics.es/products/air_segment/air_hw/ [consulta: 11/02/2021].
- [25] EMBENTION. Autopiloto Veronte 1X. En: EMBENTION [en línea] Disponible en: <https://www.embention.com/es/producto/autopiloto-simple/> [consulta: 11/02/2021].
- [26] Ramirez-Atencia, C., & Camacho, D. (2018). Extending qgroundcontrol for automated mission planning of uavs. Sensors, 18(7), 2339.
- [27] QGC. QGroundControl User Guide. En: QGC [en línea] Disponible en: <https://docs.qgroundcontrol.com/master/en/index.html> [consulta: 11/02/2021].
- [28] RCInnovations. S500 Quad PCB. En: RCInnovations [en línea] Disponible en: <https://rc-innovations.es/chasis-dron-economico-s500-quad-pcb> [consulta: 01/06/2022].
- [29] PX4. (2021). Hex Cube Black Flight Controller. En: PX4 User Guide [en línea] Disponible en: https://docs.px4.io/v1.12/en/flight_controller/pixhawk-2.html [consulta: 01/06/2022].

- [30] Benewake. CE30 3D obstacle avoidance Lidar. En: Benewake Guide [en línea] Disponible en: <http://en.benewake.com/product/detail/5c34571eadd0b639f4340ce5> [consulta: 01/06/2022].
- [31] Iha Race. ESC T-Motor F55A Pro. En: Iha Race [en línea] Disponible en: <https://iharace.com/producto/esc-t-motor-f55a-pro-ii-4-1-32bit-6s/> [consulta: 01/06/2022].
- [32] Dulal, Saurab. (2018). Octree Construction and Nearest Neighborhood Search (NNS). En: Saurab Dulal [en línea] Disponible en: <https://dulalsaurab.github.io/computerscience/octree-construction-and-nns/> [consulta: 02/06/2022].
- [33] Instituto geográfico nacional. Centro de descargas. En: Ministerio de transportes, movilidad y agenda urbana [en línea] Disponible en: <http://centrodedescargas.cnig.es/CentroDescargas/index.jsp> [consulta: 02/06/2022].
- [34] B&H. 3DR IRIS+ Quadcopter with GoPro Mount. En: B&H. [en línea] Disponible en: https://www.bhphotovideo.com/c/product/1101666-REG/3d_robots_3dr0171_iris_quadcopter_rtf.html [consulta: 02/06/2022].
- [35] Vilhjalmsson, V. (2016, May). Risk-based pathfinding for drones. Department of computer science, ETH Zürich.
- [36] DIJKSTRA, Edsger Wybe. A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271). 1959.
- [37] Shrit, O., Martin, S., Alagha, K., & Pujolle, G. (2017, June). A new approach to realize drone swarm using ad-hoc network. In *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)* (pp. 1-5). IEEE.
- [38] Mazher, W. J., Ibrahim, H. T., Ucan, O. N., & Bayat, O. (2018). Drone swarm with free-space optical communication to detect and make deep decisions about physical problems for area surveillance. *Optical Engineering*, 57(3), 036116.
- [39] Remes, B., Hensen, D., Van Tienen, F., De Wagter, C., Van der Horst, E., & De Croon, G. C. H. E. (2013, September). Paparazzi: how to make a swarm of parrot ar drones fly autonomously based on gps. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)* (pp. 17-20).
- [40] Monje, C. A., Garrido, S., Moreno, L., & Balaguer, C. (2020). UAVs Formation Approach Using Fast Marching Square Methods. *IEEE Aerospace and Electronic Systems Magazine*, 35(5), 36-46.
- [41] de Souza, B. J. O., & Endler, M. (2015, March). Coordinating movement within swarms of UAVs through mobile networks. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)* (pp. 154-159). IEEE.
- [42] Tosato, P., Facinelli, D., Prada, M., Gemma, L., Rossi, M., & Brunelli, D. (2019, June). An autonomous swarm of drones for industrial gas sensing applications. In *2019 IEEE 20th International Symposium on " A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)* (pp. 1-6). IEEE.
- [43] Mogili, U. R., & Deepak, B. B. V. L. (2018). Review on application of drone systems in precision agriculture. *Procedia computer science*, 133, 502-509.

[44] COMMISSION IMPLEMENTING REGULATION (EU) No 2019/947 of 24 May 2019 on the rules and procedures for the operation of unmanned aircraft, OJ, L 152 (11.6.2019).

[45] COMMISSION IMPLEMENTING REGULATION (EU) No 2020/639 of 12 May 2020 amending Implementing Regulation (EU) 2019/947 as regards standard scenarios for operations executed in or beyond the visual line of sight, OJ, L 150 (13.5.2020).

[46] COMMISSION IMPLEMENTING REGULATION (EU) No 2020/746 of 4 June 2020 amending Implementing Regulation (EU) 2019/947 as regards postponing dates of application of certain measures in the context of the COVID-19 pandemic, OJ, L 176 (5.6.2020).

[47] COMMISSION IMPLEMENTING REGULATION (EU) No 2021/1166 of 15 July 2021 amending Implementing Regulation (EU) 2019/947 as regards postponing the date of application for standard scenarios for operations executed in or beyond the visual line of sight, OJ, L 253 (16.7.2021).

[48] COMMISSION DELEGATED REGULATION (EU) No 2019/945 of 12 March 2019 on unmanned aircraft systems and on third-country operators of unmanned aircraft systems, OJ, L 152 (11.6.2019).

[49] COMMISSION DELEGATED REGULATION (EU) No 2020/1058 of 27 April 2020 amending Delegated Regulation (EU) 2019/945 as regards the introduction of two new unmanned aircraft systems classes, OJ, L 232 (20.7.2020).

[40] Ley 48/1960, de 21 de julio, sobre navegación aérea (1960). Boletín Oficial del Estado, 176, de 23 de julio de 1960, 10291 a 10299.
<https://www.boe.es/eli/es/l/1960/07/21/48/dof/spa/pdf>

[51] Real Decreto 37/2001, de 19 de enero, por el que se actualiza la cuantía de las indemnizaciones por daños previstas en la Ley 48/1960, de 21 de julio, de Navegación Aérea (2001). Boletín Oficial del Estado, 29, de 2 de febrero de 2001, 4100 a 4101.
<https://www.boe.es/boe/dias/2001/02/02/pdfs/A04100-04101.pdf>

[52] REGULATION (EC) No 785/2004 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 21 April 2004 on insurance requirements for air carriers and aircraft operators, OJ, L 138 (30.4.2004).

