



GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela de Ingeniería de Fuenlabrada

Curso académico 2023-2024

Trabajo Fin de Grado

Navegación Autónoma de drones basado en
inteligencia artificial y aprendizaje por refuerzo

Autor: Bárbara Villalba Herreros

Tutor: Dr. Roberto Calvo Palomino



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-NC-SA International License (Creative Commons AttributionNonCommercial-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato; y *(b) adaptar*: remezclar, transformar y crear a partir del material. El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatante.
- *No comercial.* Usted no puede hacer uso del material con propósitos comerciales.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

Agradecimientos

En primer lugar quería agradecer a todas las personas que han sido parte de este camino y trayectoria, agradezco a mis tres familias por apoyarme y no dejarme rendirme en ningún momento. Agradeciendo así a la madre de mi pareja por estar escuchandome tendida y aconsejando me.

Mención especial a mi pareja Renato Luigi por estar a pie de cañón en todo momento ayudando, apoyando, y escuchando, no me olvidaré de las charlas que teníamos en el coche mientras cenabamos. También quería agradecer a mi padre por sus charlas telefonicas de vuelta a casa, en donde me intentaba ayudar con sus ideas. Además de mi madre, mi hermana y abuelos por estar siempre a mi lado.

En segundo lugar, quiero mencionar a mi cuñado Angelo Vincenzo por ser compañero de carrera y poder haber compartido una variedad de recuerdos que nunca olvidaré.

Además de agradecer a mi tutor Roberto por la paciencia que ha tenido durante el desarrollo de este trabajo y brindarme ánimos en el camino.

Finalmente, dar las gracias a las personas que no pueden estar en estos momentos.

*A alguien especial,
que esta en el cielo, Vincenzo Barra.*

Bárbara Villalba

Resumen

La revolución tecnológica ha tenido un impacto sin precedentes en el mundo de la robótica, desde los robots industriales en las fábricas hasta tener robots capaces de entablar conversaciones con personas. Dentro del mundo de la robótica, la navegación autónoma emerge una de las áreas más emocionantes como la navegación autónoma de drones con sistemas de inteligencia artificial, permitiendo que los vehículos aéreos no tripulados no solo ejecuten tareas preprogramadas, sino que también sean capaces de aprender la adaptación de entornos dinámicos y cambiantes. Convirtiendo a estos pequeños vehículos en un gran desafío en el mundo de la robótica aérea. Los drones pueden ser programados para realizar tareas específicas por ejemplo mapear terrenos en aplicaciones cartográficas o entregar suministros médicos en zonas de difícil acceso.

A parte de la navegación autónoma, la inteligencia artificial permite a los drones por ejemplo ser entrenados para reconocer patrones y objetos en su entorno lo que les puede permitir realizar tareas como la identificación de personas en situaciones de búsqueda y rescate o la detección de anomalías en infraestructuras con algoritmos de aprendizaje automático. Sin embargo, a pesar de estos avances, la navegación autónoma e inteligencia artificial en drones sigue siendo un área de investigación debido a la necesidad de algoritmos de aprendizaje más robustos que en un futuro próximo se podrá llegar a cumplir con éxito.

Con este Trabajo de Fin de Grado demostraremos que la navegación autónoma de drones es capaz de tener un comportamiento autómato en entornos realistas y complejos de carreteras tomando decisiones en tiempo real para alcanzar sus objetivos de manera eficiente y segura. Para poder lograr esto, se explorarán y se implementarán técnicas de algoritmos de aprendizaje autómato y de inteligencia artificial con un enfoque en particular en el aprendizaje por refuerzo utilizando entornos de simulación de Airsim.

Acrónimos

UAV *Unmanned Air Vehicle*

UAS *Unmanned Air System*

IA *Inteligencia Artificial*

Airsim *Aerial Informatics and Robotics Simulation*

TFG *Trabajo de fin de Grado*

Índice general

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1. La robótica | 1 |
| 1.1.1. Enfoques en la robótica | 2 |
| 1.2. Robótica aérea | 5 |
| 1.3. La inteligencia artificial en la navegación autónoma de drones | 12 |
| 1.4. Navegación autónoma en Airsim basada en inteligencia artificial y aprendizaje por refuerzo | 14 |
| 2. Objetivos | 15 |
| 2.1. Descripción del problema | 15 |
| 2.2. Requisitos | 16 |
| 2.3. Metodología | 16 |
| 2.4. Plan de trabajo | 17 |
| 3. Plataforma de desarrollo | 19 |
| 3.1. Lenguaje de programación | 19 |
| 3.1.1. Python | 19 |
| 3.2. Entornos de Programación para la Visión por Computadora | 20 |
| 3.2.1. OpenCV | 20 |
| 3.2.2. Numpy | 21 |
| 3.2.3. Pytorch y ONNX Runtime | 21 |
| 3.2.4. YOLOP | 22 |
| 3.3. ROS | 25 |
| 3.3.1. Distribuciones | 27 |
| 3.3.2. Mavros | 27 |
| 3.4. Entornos de Simulación | 28 |
| 3.4.1. Configuración de Integración Distribuida en equipos separados . | 29 |
| 3.4.2. Airsim | 31 |
| 3.4.3. Airsim ROS Wrapper | 34 |

| | |
|---|----------|
| 3.4.4. Client Airsim | 36 |
| 3.5. PX4 AutoPilot | 36 |
| 3.5.1. Software in The Loop(SITL) | 36 |
| 3.5.2. Modos de vuelo | 37 |
| 3.6. QGroundControl | 39 |
| 4. Anexo | 1 |
| A. Bibliografía | 5 |
| Bibliografía | 5 |

Índice de figuras

| | | |
|-------|--|----|
| 1.1. | Definición de robot | 2 |
| 1.2. | Sojourner Rover | 3 |
| 1.3. | Spot de Boston Dynamics | 4 |
| 1.4. | Historia de los drones | 6 |
| 1.5. | El dron Ingenuity | 7 |
| 1.6. | Drones en inspección electrica en Galicia | 8 |
| 1.7. | El primer prototipo de dron de Prime Air | 9 |
| 1.8. | El dron MK27-2 | 10 |
| 1.9. | El dron MK30 | 10 |
| 1.10. | Ilustración de como serán los drones en un futuro creado con Copilot . | 11 |
| 1.11. | Esquema de Reinforcement Learning | 13 |
| 2.1. | Seguimiento de trabajo en GitHub | 17 |
| 3.1. | Arquitectura de YOLOP | 23 |
| 3.2. | Definición de ROS | 25 |
| 3.3. | Arquitectura de ROS | 26 |
| 3.4. | ROS noetic y ROS melodic | 27 |
| 3.5. | Infraestructura de Mavros | 28 |
| 3.6. | Esquema de comunicaciones entre las plataformas de desarollo Mavros, PX4 y QGC junto con el entorno de simulación | 30 |
| 3.7. | Esquema de comunicaciones entre las plataformas de desarollo Airsim ROS Wrapper junto con el entorno de simulación | 30 |
| 3.8. | Ejemplos de escenarios en Airsim | 33 |
| 3.9. | Ilustración de los ejes de referencia del dron | 35 |
| 3.10. | Diagrama del comportamiento del modo de vuelo Position | 38 |
| 3.11. | QGroundControl | 39 |

Listado de códigos

| | |
|---|----|
| 3.1. Ejemplo de código en Python de una función para calcular el factorial de un número | 20 |
| 3.2. Ejemplo de código en Python de operaciones básicas utilizando la libreria OpenCv | 21 |
| 3.3. Ejemplo de código en Python de operaciones básicas utilizando la libreria Numpy | 21 |
| 3.4. Cargar modelo YOLOP con pesos preentrenados End-to-end.pth | 24 |

Listado de ecuaciones

Índice de cuadros

Capítulo 1

Introducción

1.1. La robótica

En la última década, la evolución tecnológica ha provocado una transformación radical en nuestra forma de vivir, trabajar y relacionarnos desempeñando la tecnología un papel fundamental en el avance de la sociedad e impulsando una serie de innovaciones que se extienden desde la invención de la rueda hasta la era digital contemporánea. Por ejemplo, los ordenadores empezaron siendo grandes máquinas que ocupaban habitaciones enteras que requerían una gran cantidad de energía y mantenimiento. Hoy en día, los ordenadores son dispositivos ligeros y eficientes que pueden realizar múltiples cálculos por segundo. Respecto a la telefonía móvil también ha experimentado una evolución destacada, los primeros móviles se caracterizaban de ser dispositivos pesados y grandes que solo podían realizar llamadas, pero hoy en día no solamente podemos realizar llamadas telefónicas con un teléfono móvil, sino que también podemos enviar mensajes, navegar por internet, tomar fotos, escuchar música, ver videos y mucho más.

Entre las diversas ramas de la tecnología, la robótica se destaca como una de las más prometedoras. Apareciendo como disciplina durante la década de los años 60, la robótica ha tenido un cambio asombroso pasando de ser simples máquinas programables a sistemas inteligentes capaces de aprender y adaptarse a su entorno teniendo avances en diversas disciplinas de la ingeniería, como la informática, la inteligencia artificial, la ingeniería de control, la mecánica y otras más. Los robots de hoy en día no solo tienen la capacidad de realizar tareas programadas y repetitivas, sino que también tienen la capacidad de interactuar con su entorno , tomar decisiones basadas en la información sensorial y aprender de sus experiencias. Este avance en la robótica nos ha permitido tener una definición más precisa de lo que es la robótica moderna, definiendo la robótica como ciencia interdisciplinaria encargada de la creación, funcionamiento, estructuración, fabricación y uso de los robots. Esta definición mencionada incluye no

solo los componentes mecánicos y eléctricos, sino que también los algoritmos que los controlan, los sensores que les permiten recopilar datos de su entorno y los sistemas que procesan esta información y toman decisiones.

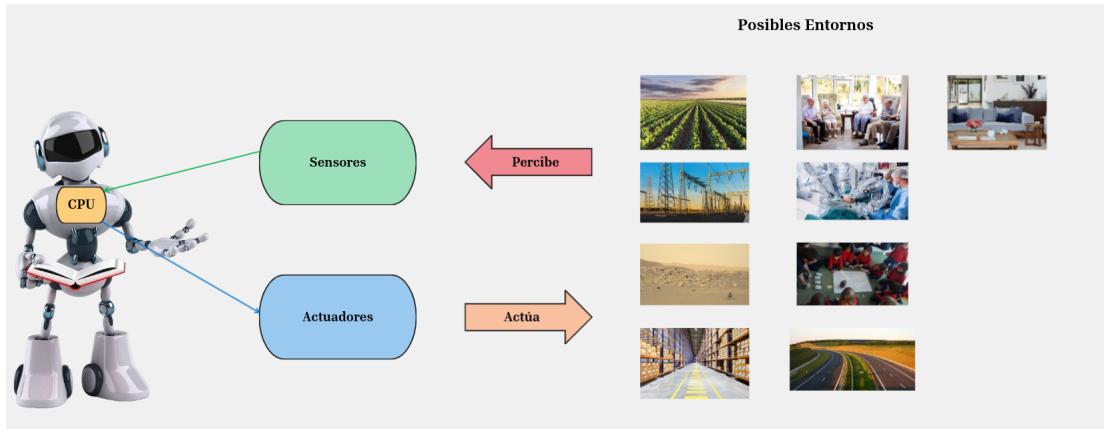


Figura 1.1: Definición de robot.

Lo que hace que un robot tenga la capacidad de aprender y adaptarse a un entorno abierto de nuevas oportunidades para la robótica, como la medicina, la exploración lunar, la asistencia personal, la automatización industrial y más. Además de abrir nuevas aplicaciones y tareas como puede ser la navegación autónoma, la detención de objetos o la manipulación de objetos con sensores táctiles y de fuerza, dichas tareas pueden realizar pueden ser peligrosas, delicadas, sucias o monótonas (conocidas como las 4D's: dull,dirty, dangerous and dear) ¹

1.1.1. Enfoques en la robótica

A lo largo de la evolución de la robótica, han surgido dos enfoques fundamentales para el diseño y la operación de robots, cada uno de estos enfoques presentan diferentes formas de interactuar y operar robots, con sus propias características y aplicaciones únicas.

Teleoperación

La teleoperación surge de la necesidad de manipular objetos o realizar tareas en entornos complejos, peligrosos y distantes para el ser humano. Desde la historia, el ser humano ha utilizado una variedad de herramientas para ampliar su capacidad

¹:<https://www.forbes.com/sites/bernardmarr/2017/10/16/the-4-ds-of-robotization-dull-dirty-dangerous-and-dear/?sh=40bb6cec3e0d>

de manipulación como palos utilizados para caer la fruta madura de un arbol. Con el tiempo, se desarrollaron dispositivos más complejos, como pinzas que permitían manipular piezas o alcanzar objetos de difícil acceso facilitando el trabajo para el operario. En la era moderna, la teleoperación ha estado evolucionando hasta el punto de incluir sistemas robóticos robustos que pueden ser controlados a distancia, permitiendo al operario poder realizar tareas en entornos peligrosos e innaccesibles para el ser humano como puede ser la exploración espacial, la medicina o la inspección nuclear.

La intervención del operador humano en los sistemas de teleoperación de robots es imprescindible, debe ser capaz de poder intepretar los datos sensoriales que proporciona el robot, así como de tomar decisiones robustas y precisas dependiendo de la situación. Esto conlleva tener una capacidad de realizar múltiple tareas simultáneamente adaptándose a situaciones imprevistas.

Hoy en día, la teleoperación de robots tiene variedad de aplicaciones. Una de ellas puede ser la exploración espacial, en donde se utiliza la teleoperación como técnica de manipulación remota como el Sojourner Rover. El Sojourner Rover² es un pequeño robot móvil compuesto por 6 ruedas creado por los científicos de la NASA para estudiar la superficie de Marte con la capacidad de enviar imágenes en directo y realizar análisis del terreno del planeta. Gracias a sus ruedas podía moverse por terrenos rocosos y de difícil acceso ya que estaban equipadas materiales como de aluminio y acero inoxidable.



Figura 1.2: Sojourner Rover

²<https://www.astronomy.com/space-exploration/sojourner-nasas-first-mars-rover/>

Con esta misión espacial se pudo probar como era el entorno marciano con técnicas realizadas en los laboratorios de la NASA demostrando que se podía realizar una teleoperación en el espacio abriendo el camino a futuros rovers como el Spirit, Opportunity y más ³.

Autonomía

La robótica autónoma consiste en tener robots que sean capaces de operar y realizar tareas de forma independiente sin la intervención de un ser humano. En contraste con los robots teleoperados, este tipo de robots necesitan un comportamiento más robusto y preciso para realizar tareas independientes basándose en la percepción del entorno y en la toma de decisiones autónomas.

El concepto de automía en los sistemas robóticos se está convirtiendo en un área de investigación activa y en rápido desarrollo. Los avances en inteligencia artificial (IA), visión artificial, aprendizaje automático han facilitado la creación de robots autónomos capaces de llevar a cabo amplias variedades de tareas en entornos no estructurados y cambiantes [8].



Figura 1.3: Spot de Boston Dynamics

³<https://spaceplace.nasa.gov/mars-spirit-opportunity/sp/>

1.2. Robótica aérea

Dentro del campo de la robótica aérea tenemos los drones. Podemos definir un dron, como vehículo aéreo no tripulado (UAV), es un tipo de aeronave que puede operar sin la necesidad de un piloto humano a bordo. Estos dispositivos pueden ser controlados remotamente por un operador humano o navegar autonomamente incoporando software en su sistema.

El origen de los drones se remonta a la Primera Guerra Mundial con el biplano Kettering Bug. Este era un torpedo no tripulado de 240 kg (con una envergadura de 4,5 m, una longitud de 3,8 m y una altura de 2,3 m)⁴ era propulsado por un motor alternativo. Podía volar de forma autónoma hasta un punto específico, donde soltaba sus alas y caía en “caída libre” ⁵.

Avanzando en la historia, en 1935 se desarrolló el DH.82 Queen Bee⁶. Este era un blanco aéreo sin piloto que era controlado por radio. De hecho, parece que el término “dron” se originó a partir del nombre, que se refiere a la abeja macho que realiza un vuelo en busca de la abeja reina y luego fallece.

Durante la Segunda Guerra Mundial, quizás el más conocido fue el V-1 ”Flying Bomb”⁷, el primer misil de crucero operativo del mundo, en donde su sistema de guía pre establecido incluía una brújula magnética que monitoreaba un autopiloto con giroscopios. También en este periodo, destacaremos el *Proyect Aphrodite* [3], fue un programa que tenía como objetivo convertir bombarderos en bombas voladoras no tripuladas que eran controladas por radio. Más adelante estos bombarderos no tripulados se utilizaron para volar a través de nubes de hongo después de las pruebas nucleares.

Destacando más UAVs, tenemos la familia Teledyne Ryan Firebee/Firefly ⁸, estos sistemas generalmente se lanzaban desde el aire y se recuperaban mediante una combinación de paracaídas y helicópteros. El Lockheed D-21 fue uno de los sistemas más impresionantes durante la Guerra Fría. Este UAV fue impulsado por estatorreactor con

⁴<https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/198095/kettering-aerial-torpedo-bug/>

⁵<https://daytonunknown.com/2023/06/30/the-kettering-bug-the-worlds-first-drone/>

⁶<https://dronewars.net/2014/10/06/rise-of-the-reapers-a-brief-history-of-drones/>

⁷<https://migflug.com/jetflights/the-v1-flying-bomb/>

⁸<https://www.designation-systems.net/dusrm/m-34.html>

velocidades mayores que Mach 3⁹. En la Edad Moderna, destacamos El Condor [1], fue el primer UAS en utilizar navegación GPS y tecnología de aterrizaje automático y el Predactor¹⁰. En la época dorada, gracias a los avances anteriores se pudo desarrollar sistemas militares esenciales que han demostrado su valor y el desarrollo de vehículos aéreos no tripulados pequeños (small UAV). Este ultimo ha despertado un gran interés significativo resaltando como puntos de entrega al mercado civil ya que con sus cargas útiles reducidas pueden ser portátiles y tener un coste menor .



Figura 1.4: Historia de los drones

Cada vez es más común que los drones sean más sofisticados y accesibles. Por ejemplo, el dron Ingenuity de la NASA se ha convertido en el primer vehículo aéreo en poder volar sobre la superficie de otro planeta. Fue transportado a Marte mediante el rover Perseverance de la NASA, una vez fue posicionado el dron se elevó

⁹<https://www.marchfield.org/aircraft/unmanned/d-21-drone-lockheed/>

¹⁰<https://www.airforce-technology.com/projects/predator-uav/?cf-view>

cerca de 3 metros realizando diferentes giros y desplazamientos tomando fotos a la superficie, teniendo la capacidad de escoger de forma autónoma los sitios de aterrizaje en el terreno marciano ¹¹.

Uno de los gran retos de este proyecto fue poder volar sobre la superficie de Marte ya que su atmósfera esta compuesta por el 1% de la densidad terrestre dificultando el vuelo del dron. Sin embargo, gracias a su diseño ligero y a sus hélices especialmente diseñadas para crear suficiente sustentación en la atmósfera del planeta, el Ingenuity fue capaz de superar este desafío ¹².



Figura 1.5: El dron Ingenuity

Otro ejemplo de uso podemos tener control y mantenimiento de redes eléctricas y otras infraestructuras. Algunas construcciones constan de grandes alturas y tamaños, lo que puede dificultar el trabajo y su correcto mantenimiento. No obstante, estas tareas con los drones se agilizan y se vuelven más eficientes y robustas, porque permiten poder inspeccionar dichas infraestructuras desde cerca sin poner en peligro a la seguridad de los operarios. Hay drones que se encargan en la monitorización de infraestructuras eléctricas.

Unión Fenosa, la distribución eléctrica en España de Naturgy, en 2018 incorporó

¹¹<https://ciencia.nasa.gov/sistema-solar/finaliza-la-mision-del-helicóptero-ingenuity-en-marte/>

¹²<https://www.bbc.com/mundo/noticias-56738201>

drones a sus instalaciones eléctricas para realizar labores de supervisión. Estos drones aportan soluciones optimizadas y eficientes en costes. Si tenemos en cuenta la longitud que puede tener las redes eléctricas, el uso de estos vehículos autónomos facilita las tareas de supervisión equipados de cámaras de última generación permitiendo al operario observar en tiempo real el estado de las infraestructuras. Además de que los drones podrían acceder a zonas de difícil acceso para comprobar daños y poder repararlos ¹³.

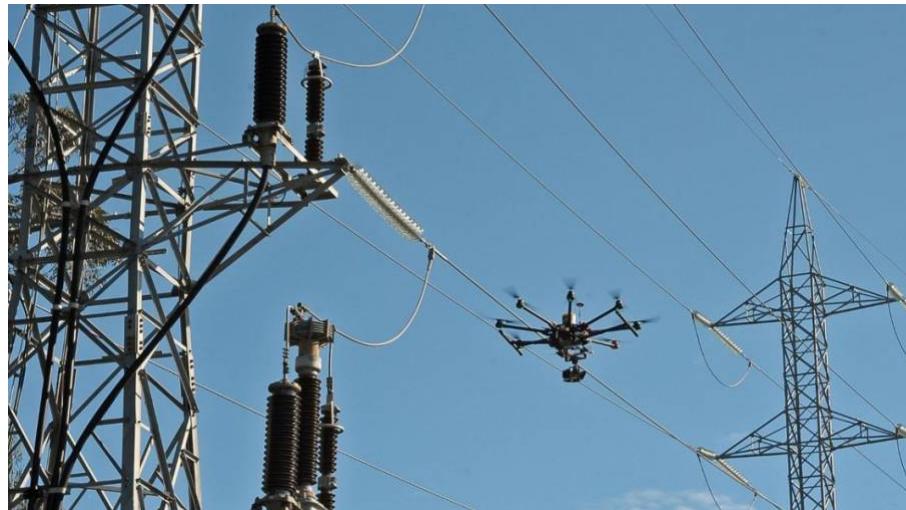


Figura 1.6: Drones en inspección electrica en Galicia

Asimismo, Amazon ha estado trabajando en el desarrollo de drones para la entrega de paquetes durante varios años denominado así Prime Air^[5], que consiste en un sistema de entrega de paquetes utilizando estos vehículos. Durante este programa, han realizado diferentes pruebas de reparto de paquetes a clientes en 60 minutos o menos. Estas entregas se realizan desde sitios preparados con esta modalidad, el proceso de entrega de los drones empezaría en los centros de preparación de pedidos en donde los empleados seleccionarían el artículo, lo llevarían a la estación de embalaje y cuando este preparado su embalaje se deslizaría por una rampa hacia la fase de entrega. En la fase de entrega los empleados preparan el paquete y la batería del dron para su destino. Cuando el dron haya realizado el despegue seguirá una ruta preestablecida con supervisión de un operador para garantizar su entrega.

Estos drones están automatizados para que logren volar a velocidades de 65 kilómetros por hora, al llegar el dron a su destino, se detendrá y descenderá lentamente para dejar el paquete en un área designada. Una vez el paquete haya sido entregado, el

¹³<https://www.ufd.es/blog/primer-vuelo-de-un-dron-mas-allá-de-la-línea-visual/>

vehículo regresará al centro de preparación de pedidos para realizar la siguiente entrega¹⁴.



Figura 1.7: El primer prototipo de dron de Prime Air

A lo largo de los años, Amazon ha seguido investigando y diseñando nuevos modelos de drones como el dron MK27-2¹⁵. Fue el primer dron que utilizó Amazon para las primeras entregas dentro del programa Prime Air durante el año 2023, se basaba en un dron eléctrico capaz de entregar paquetes con un peso máximo de 1,5 kilogramos a los clientes en menos de una hora y capaz de realizar vuelos evitando obstáculos como puede ser las chimeneas o las torres de telefonía aunque no puede realizar entregas durante tormentas, vientos fuertes, temperaturas extremas o cualquier situación climatológica desfavorable.

Este servicio solamente está disponible para domicilios que tengan patios traseros que dispongan de espacio suficiente para que el dron pueda realizar el aterrizaje y la entrega del pedido.

¹⁴<https://logistica.cdecomunicacion.es/e-commerce/140453/prime-air-amazon-entregas-drones>

¹⁵<https://www.europapress.es/portaltic/gadgets/noticia-amazon-prime-air-comienza-entregar-pedidos-drones-estados-unidos-20221229115034.html>



Figura 1.8: El dron MK27-2

Sin embargo, Amazon Prime Air será más eficiente gracias al dron MK30 creado y diseñado por Amazon. Este pequeño dron será capaz de volar en diferentes condiciones climatológicas y constará de un sistema capaz de identificar y evitar obstáculos en el área de entrega. Será capaz de volar en situaciones de lluvia ligera y poder realizar vuelos generando un 25 % menos de ruido que el anterior modelo. Una novedad de este dron en comparación con los anteriores modelos es que será capaz de aterrizar en espacios más reducidos lo que conlleva a que este tipo de servicio pueda llegar a más vencidarios.

Se tiene previsto que se llegue a probar en el año 2024 empezando por ciudades como Texas y California en Estados Unidos ¹⁶.



Figura 1.9: El dron MK30

¹⁶<https://www.forbesargentina.com/innovacion/asi-nuevo-asombroso-dron-amazon-mk30-n42612>

En un futuro cercano, puede que los drones sean más eficientes para las aplicaciones civiles y científicas incluyendo protección contra incendios forestales, misiones agrícolas, ayuda en catástrofes y más. Las demostraciones actuales del uso de los drones han revelado el potencial que pueden tener pero aun así el acceso al espacio aéreo sigue siendo un factor limitante. Con el paso del tiempo, se irán desarrollando nuevas tecnologías prácticas para poder permitir una integración segura en el espacio aéreo [7].

En resumen, los drones son una tecnología emergente con un potencial significativo para transformar una variedad de industrias. Sin embargo, también plantean desafíos únicos que deben ser abordados a medida que se integran más plenamente en nuestra sociedad. Con el desarrollo continuo de la tecnología de los drones y la evolución de las regulaciones, es probable que veamos un aumento en la variedad de las aplicaciones de los drones en el futuro.

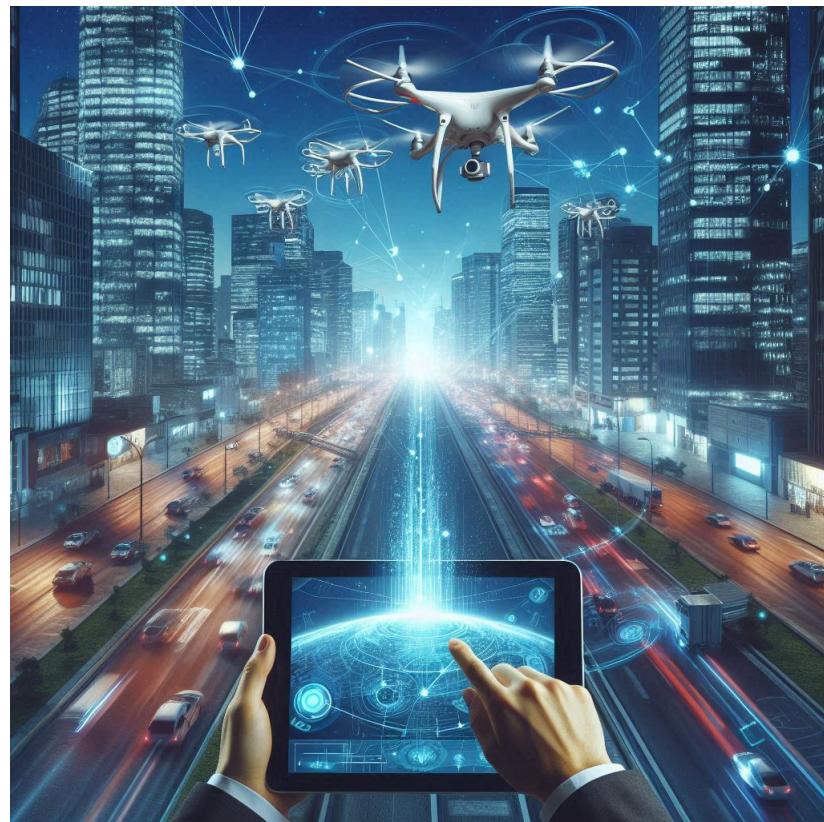


Figura 1.10: Ilustración de como serán los drones en un futuro creado con Copilot

1.3. La inteligencia artificial en la navegación autónoma de drones

La incorporación de inteligencia artificial en el mundo de los drones desempeña un papel crucial en la navegación autonómica, permitiéndoles tomar decisiones en tiempo real y adaptarse a entornos cambiantes de manera eficiente. Permitiendo a los drones poder aprender de sus experiencias y entender e interactuar con el entorno en el que se encuentran de una manera más natural.

Los drones que están equipados con IA pueden realizar vuelos de precisión, mantener la estabilidad incluso en condiciones adversas como fuertes vientos, y evitar obstáculos de forma dinámica. Esto es posible gracias a la combinación de datos sensoriales junto con los algoritmos de IA, lo que permite al dron interpretar su entorno y tomar decisiones en tiempo real.

Uno de los enfoques más destacados en la navegación autonómica de drones es el aprendizaje automático. Este enfoque permite a los drones mejorar su objetivo a través de la experiencia y los datos recopilados durante el vuelo.

Los algoritmos de aprendizaje automático, como las redes neuronales convolucionales (CNN), son utilizados para la detección y clasificación de objetos. Por ejemplo, las CNN son capaces de analizar imágenes capturadas por las cámaras a bordo del dron para identificar obstáculos, peatones o vehículos. Un tipo de aplicación de uso de redes neuronales puede estar en la detección y clasificación de malas hierbas [4]. Mediante el sensor de la cámara, el dron es capaz de capturar imágenes en tiempo real para más adelante usar la red neuronal CNN YOLOv8¹⁷ para detectar y clasificar las diferentes hierbas que puede haber en un campo de cultivo. Este tipo de aplicación es bastante útil para la inspección agrícola ya que los drones pueden crear mapas detallados que permiten a los agricultores aplicar herbicidas de manera más eficiente y precisa, también este tipo de aplicación puede ser útil para tener una monitorización general sobre la salud del cultivo.

Por otro lado, el reinforcement learning (RL) [9] es una técnica dentro del aprendizaje automático que promete bastante en la navegación autonómica de drones. Esta metodología permite a los drones aprender a realizar trayectorias y planificar rutas de forma autonómica, mejorando su desempeño a mediante un esquema de penalizaciones y recompensas permitiendo así al dron poder tomar decisiones decisivas en situaciones

¹⁷<https://github.com/ultralytics/ultralytics>

puntuales. En este artículo [10] precisamente se utiliza un algoritmo de RL para la evitación de obstáculos en un espacio continuo y se llega a conseguir que con estos tipos de algoritmos que un dron pueda llegar aprender comportamientos y tomar decisiones por él mismo.

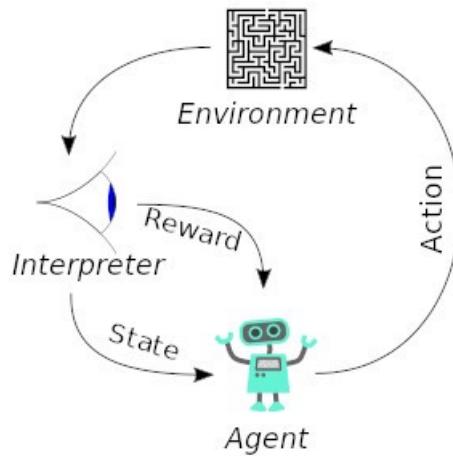


Figura 1.11: Esquema de Reinforcement Learning

En conclusión, la inteligencia artificial puede ser fundamental en la navegación autónoma de drones al permitirles percibir su entorno, podemos tomar decisiones y planificar acciones de manera anticipada y autónoma. A medida que vayamos avanzando, se espera que los drones tengan más sistemas de inteligencia artificial abordo para cubrir una amplia gama de tareas de manera autónoma, lo que abriría nuevas fronteras en campos como el rescate, la vigilancia, la logística y la exploración, y que promete seguir transformando la forma en que interactuamos con el espacio aéreo en un futuro.

1.4. Navegación autónoma en Airsim basada en inteligencia artificial y aprendizaje por refuerzo

En este trabajo realizaremos una navegación autonómica en entornos de simulación por Airsim mediante redes neuronales, algoritmos de aprendizaje automático y aprendizaje por refuerzo. Demostrando así que un dron es capaz de tomar decisiones por si mismo dependiente de la situación en la que se pueda encontrar en un entorno de carreteras.

Capítulo 2

Objetivos

En esta sección se describirá el problema a resolver junto con los objetivos pautados en el desarrollo del TFG

2.1. Descripción del problema

Como anteriormente hemos relatado, los drones tienen un uso actualmente elevado para solventar tareas de alta complejidad adoptando de sensores para ello.

El objetivo principal de este TFG, es desarrollar un comportamiento de navegación autónoma basado en aprendizaje por refuerzo e inteligencia artificial, en el que el dron sea capaz de desenvolverse por escenarios urbanos.

A continuación, se definen los siguientes subobjetivos:

1. Análisis de la viabilidad de desarrollar aplicaciones de navegación autónoma en entornos fotorrealistas para drones.
2. Análisis y desarrollo de una aplicación de navegación autónoma de drones basandonos en el seguimiento de un carril.
3. Creación de un comportamiento autónomo sigue-carril basado en inteligencia artificial y aprendizaje por refuerzo.
4. Análisis y comparativas de los diferentes comportamientos desarrollados en este trabajo con el fin de encontrar interesantes acerca de la utilización de redes neuronales y aprendizaje por refuerzo en la navegación autónoma de drones.

2.2. Requisitos

Los requisitos que han de cumplirse en este trabajo son:

1. Uso del vehículo UAV en el entorno de simulación fotorrealista Airsim junto a UnRealEngine.
2. El comportamiento robusto y en tiempo real para garantizar la navegación segura del vehículo dentro de su entorno mediante Airsim client y ROS Wrapper Airsim.
3. Los sistemas desarrollados deben ser reactivos para poder reaccionar a su entorno de manera concisa y eficiente.

2.3. Metodología

Este trabajo, comenzó oficialmente en Septiembre del 2023 aunque en Diciembre del 2022 se plantearon varias ideas a desarrollar, y se finalizó en Mayo del 2024.

La metodología que se llevo a cabo fue:

1. Reuniones semanales o cada dos semanales mediante Teams¹ con una duración de media o una hora, con el fin de tener un control semanal y pactar los objetivos semanales a seguir. Gracias a estas reuniones, se tenia una organización global del proyecto.
2. Contacto via email de la universidad con el fin de solventar problemas urgentes.
3. Utilización de la metodología Scrum², definiendo así los sprints de forma variable normalmente de una semana: Esta metodología consiste marco ágil de gestión y desarrollo de proyectos, especialmente diseñado para entornos complejos y cambiantes, basadas en sesiones de trabajo de dos semanas con reuniones, con un enfoque particular en el desarrollo de software. Se fundamenta en la colaboración, la flexibilidad y la entrega iterativa e incremental de productos de alta calidad.
4. Tener un control de versiones mediante la plataforma GitHub³, con el objetivo de tener un almacenamiento de código y respaldos de ello.

¹<https://www.microsoft.com/es-es/microsoft-teams/group-chat-software>

²<https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/>

³<https://github.com/RoboticsLabURJC/2022-tfg-barbara-villalba>

5. El uso de un blog ⁴, en el cual se describió brevemente los pasos que se siguieron para el desarrollo del TFG.



Figura 2.1: Seguimiento de trabajo en GitHub

2.4. Plan de trabajo

Finalmente, los pasos a seguir de este trabajo han sido:

1. Comienzo del trabajo.
 - Búsqueda del problema a desarrollar y análisis del estado del arte del uso de los drones en aplicaciones robóticas.
 - Instalación de las diferentes librerías y aplicaciones de software.
 - Preparación de configuración de toda la infraestructura, teniendo un análisis y estudio de comunicaciones para poder comenzar con el desarrollo.
2. Desarrollo: Una vez se tuvo listo toda la infraestructura tanto de comunicaciones como de librerías de software, se dio a pie el comienzo del desarrollo del código
 - En primer lugar, se desarrolló un teleoperador sencillo del drone para ver el funcionamiento del vehículo y dicho comportamiento.
 - Una vez finalizada la tarea del teleoperador, se comenzó con los algoritmos de percepción de detención de carril mediante redes neuronales.
 - Análisis y comparación de los resultados de los diferentes modelos que ofrece la red neuronal escogida con el propósito de tener la mejor solución.
 - El siguiente paso fue estudiar la posibilidad de tener un algoritmo de aprendizaje no supervisado llamado clustering para clasificar las diferentes líneas que aparezcan en el escenario de la carretera.

⁴<https://roboticslaburjc.github.io/2022-tfg-barbara-villalba/>

- Desarrollo del algoritmo de percepción junto con los dos puntos anteriormente mencionados.
 - Con el fin del algoritmo de percepción, se comenzó el desarrollo de un controlador sencillo PID para ver el funcionamiento de la percepción y de la navegación en el vehículo.
 - A continuación, fue la programación del algoritmo de aprendizaje por refuerzo para el seguimiento del carril.
3. Evaluación: Se realizó la comparativa de los resultados obtenidos en el aprendizaje por refuerzo.
 4. Redacción de la memoria del trabajo para la documentación de todo el proceso de investigación realizado.

Capítulo 3

Plataforma de desarrollo

En este capítulo hablaremos sobre que tecnologías hemos utilizado durante el desarrollo de este trabajo junto con el lenguaje de programación y las librerías utilizadas.

3.1. Lenguaje de programación

3.1.1. Python

Para el desarrollo de este TFG, hemos utilizado como lenguaje de programación Python. Python¹ es un lenguaje de programación interpretado, de tipado dinámico y orientado a objetos, utilizado para el desarrollo de software, aplicaciones web, data science y machine learning (ML). Fue creado por Guido van Rossum² en 1989, el nombre de "Python" se inspiró en el programa de televisión británico "Monty Python's Flying Circus".

Este lenguaje con los tiempos se ha convertido en uno de los lenguajes más populares del mundo, esto se puede deber a su sintaxis sencilla, clara y legible, aparte de estos beneficios también presenta una amplia gama de bibliotecas y marcos de trabajo. Además, se trata de un lenguaje de programación 'open source' (código abierto) y está disponible bajo una licencia de código abierto aprobada por la Iniciativa de Código Abierto (OSI), esto significa que Python es libre de usar, distribuir y modificar, incluso para uso comercial.

También a destacar, la facilidad que puede ser la instalación de dicho lenguaje en sistemas operativos como Linux o Windows.

En el caso de este TFG, se utilizará Python3 para todo el desarrollo del código

¹<https://www.python.org/>

²<https://gvanrossum.github.io/>

junto con el middleware robotico ROS (veáse la sección 3.3) y para el desarrollo de los diferentes algoritmos.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(5)) # Output: 120
```

Código 3.1: Ejemplo de código en Python de una función para calcular el factorial de un número

3.2. Entornos de Programación para la Visión por Computadora

Para el desarrollo del sistema de percepción, hemos utilizado varias librerías mediante el lenguaje de programación Python para poder percibir el entorno en el que vamos a estar trabajando.

3.2.1. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de open source dedicada para el tratamiento de imágenes y aprendizaje automático. Fue desarrollada por Intel y lanzada en el año 2000 y esta disponible para todos los públicos tanto para uso comercial como para uso personal³.

Ofrece diferentes tareas como procesamiento de imágenes, detención de objetos, extracción de características, reconocimiento facial, estimación de movimiento entre otras. Además de ser compatible para múltiples sistemas operativos como Windows, Linux, MacOS, Android e iOS, lo que hace que sea una librería muy versátil para el desarrollo de aplicaciones en diferentes dispositivos y entornos.

En nuestro caso utilizaremos esta biblioteca para poder obtener la imagen mediante la cámara que va abordo del vehículo.

³<https://opencv.org/>

```

import cv2

imagen = cv2.imread('ruta/a/tu/imagen.jpg')

cv2.imshow('Imagen', imagen)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

Código 3.2: Ejemplo de código en Python de operaciones básicas utilizando la libreria OpenCv

3.2.2. Numpy

Numpy⁴ (Numerical Python) es una librería dedicada para el cálculo científico en Python como arrays multidimensionales y matrices, junto con una amplia colección de funciones matemáticas. Esta biblioteca es bastante utilizada en la comunidad debido a su eficiencia y facilidad de uso.

A continuación, se muestra un simple ejemplo de como podemos crear un array en NumPy e realizar operaciones básicas como la suma y calcular la matriz transpuesta

```

import numpy as np

array1d = np.array([1, 2, 3, 4, 5])

array2d = np.array([[1, 2, 3], [4, 5, 6]])

suma = np.sum(array1d)
transpuesta = np.transpose(array2d)

```

Código 3.3: Ejemplo de código en Python de operaciones básicas utilizando la libreria Numpy

Utilizaremos esta libreria para el desarrollo del sistema de percepción que más adelante se explicará con más detalle.

3.2.3. Pytorch y ONNX Runtime

Pytorch⁵ es una biblioteca de Python para el aprendizaje automático que permite a los desarrolladores crear y entrenar modelos de aprendizaje profundo. Su capacidad de poder construir gráficos computacionales dinámicos simplifica la depuración permitiendo realizar modificaciones en tiempo real durante el desarrollo de modelos.

⁴<https://numpy.org/>

⁵[Pytorch: https://pytorch.org/](https://pytorch.org/)

Además de proporcionar una integración con otras bibliotecas de Python, como puede ser NumPy, lo que facilita la manipulación y el análisis de datos. Una de sus características es que puede soportar cálculos en GPU, lo que acelera considerablemente el entrenamiento y la inferencia de modelos.

Onnx⁶(Open Neural Network Exchange) es un framework de aprendizaje automático intermedio, permite que los modelos de aprendizaje automático creados en un framework (como Pytorch o TensorFlow) se conviertan a un formato estandar (ONNX), y luego se puedan ejecutar en otros frameworks sin necesidad de reconstruir o volver a entrenar el modelo. Esto facilita el uso del mejor framework para cada tarea específica, ya sea para entrenamiento, optimización o inferencia.

Utilizaremos ambas librerías para cargar y utilizar los modelos que ofrece la red neuronal YOLOP.

3.2.4. YOLOP

YOLOP[2] (You Only Look Once for Panoptic Driving Perception) es una red de percepción de conducción panóptica que ofrece múltiples tareas que puede manejar simultáneamente tres tareas cruciales en la conducción autónoma:

1. **Detención de objetos de tráfico:** Identifica los objetos presentes en la carretera, como otros vehículos, peatones, señales de tráfico, etc.
2. **Segmentación del área transitable:** Determina las áreas de la carretera por las que un vehículo puede conducir de manera segura.
3. **Detección de carriles:** Identifica los carriles de la carretera.

⁶Onnx: <https://onnxruntime.ai/>

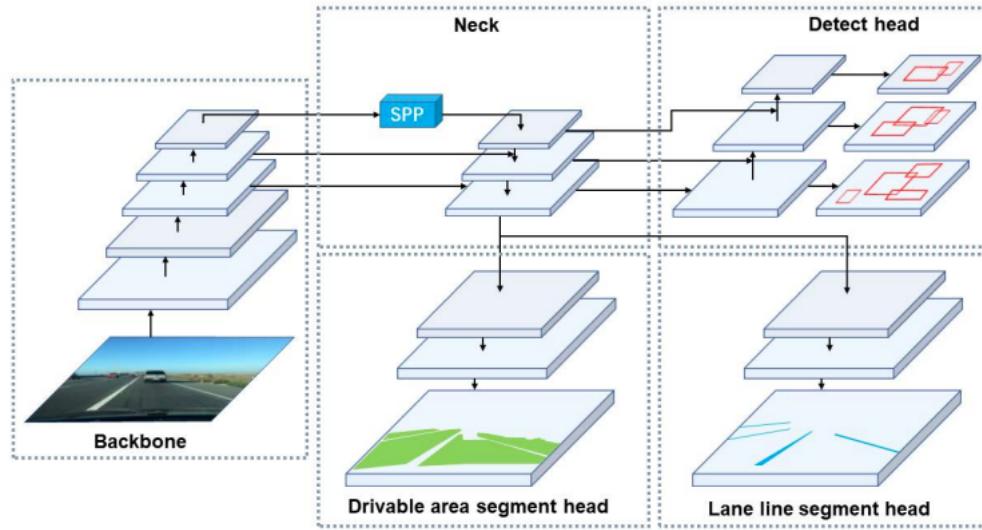


Figura 3.1: Arquitectura de YOLOP

YOLOP está compuesto por un codificador para la extracción de características y tres decodificadores para manejar las tareas específicas. Este modelo ha demostrado un rendimiento extremadamente bueno en el desafiante conjunto de datos BDD100K, logrando el estado del arte en las tres tareas en términos de precisión y velocidad.

Nosotros utilizaremos los decodificadores de segmentación del área transitable y detección de carriles. El decodificador de segmentación del área transitable utiliza los mapas de características extraídos por el codificador para realizar una predicción semántica a nivel de pixeles. Esto significa que para cada pixel de la imagen, el decodificador de segmentación del área transitable predice si el pixel pertenece a un área transitable o no.

El decodificador de detección de carriles, también utiliza los mapas de características extraídos por el codificador. Sin embargo, en lugar de predecir si un pixel es transitabile o no, este decodificador predice si un pixel pertenece a un carril de la carretera.

Es importante destacar que estos decodificadores no funcionan de manera aislada, sino que forman parte de un sistema de aprendizaje multitarea, es decir, se entranaran conjuntamente para realizar sus tareas respectivas, lo que puede mejorar el rendimiento general del sistema.

Ambos decodificadores, deben de recibir una entrada (W/8,H/8,256):

1. **W/8 y H/8**: Representa la anchura y la altura de la imagen de entrada, respectivamente, divididas por 8.
2. **256**: es el número de canales en el tensor de entrada. En el contexto de las redes neuronales convolucionales, un canal puede ser una característica aprendida (como bordes, texturas, colores, etc.) o una capa de color en una imagen (como rojo, verde, azul en imágenes RGB)

Devuelven una salida de tipo (W,H,2), siendo W la anchura y H la altura de la imagen, solo hay dos canales en cada mapa de características, ya que cada píxel representa si pertenece a una clase de objeto o al fondo.

Modelo de YOLOP

YOLOP sigue un modelo de red neuronal CNN(Convolutional neuronal network). Es un tipo de red neuronal artificial diseñada para procesar datos con una estructura de cuadrícula, como una imagen. Dicho modelo presenta unos pesos preentrenados:

1. **End-to-end.pth**: Este archivo se ha construido a partir de la biblioteca Pytorch.
2. **Yolop-320-320.onnx, yolop-640-640.onnx y yolop-1020-1020.onnx**: Estos tres archivos se han construido a partir de Onnx.

Dependiendo de que pesos preentrenados queramos utilizar tendremos resultados diferentes en el ámbito de cómputo y rápidos a la hora de realizar la inferencia del modelo con los pesos.

```
import torch

# load model
model = torch.hub.load('hustvl/yolop', 'yolop', pretrained=True)

#inference
img = torch.randn(1,3,640,640)
det_out, da_seg_out, ll_seg_out = model(img)
```

Código 3.4: Ejemplo básico de cómo poder utilizar YOLOP

Este ejemplo se puede encontrar en la página Pytorch dedicada a la red neuronal YOLOP⁷.

Por lo que, utilizaremos esta red neuronal para poder detectar los carriles que puede tener las áreas transitables en Airsim.

3.3. ROS

ROS (Robot Operating System)⁸ es un sistema operativo de código abierto utilizado principalmente para aplicaciones robóticas. Podemos definir este middleware de la siguiente forma:



Figura 3.2: Definición de ROS

1. **Plumbing:** ROS proporciona una infraestructura de mensajería publicador-subscriptor diseñada para facilitar la construcción sencilla y rápida de sistemas informáticos distribuidos.
2. **Tools:** ROS proporciona introspección, lanzamiento, depuración, visualización, trazado, registro, reproducción y detener sistemas informáticos distribuidos.
3. **Capabilities:** ROS proporciona una amplia colección de bibliotecas que implementan funciones útiles para los robots, por ejemplo, movilidad, manipulación y percepción.
4. **Community:** ROS cuenta con el apoyo y la mejora de una gran comunidad, con un fuerte enfoque en la integración y la documentación, gracias a ello, es una ventaja poder aprender a cerca de los miles de paquetes que ofrece ROS que están disponibles de desarrolladores de todo el mundo.

Este middleware sigue un modelo parcialmente centralizado de publicación y suscripción, el cual el publicador genera mensajes y eventos asociados a un topic y

⁷https://pytorch.org/hub/hustvl_yolop/

⁸<https://www.ros.org/>

el subscriptor es quien se subscribe al topic correspondiente y recibe la información que ha generado el publicador.

Este tipo sistema es bastante útil ya que permite a los desarrolladores cambiar, añadir o eliminar nodos (programas en ejecución) sin afectar al resto del sistema, facilitando de forma asíncrona el desarrollo iterativo, permitiendo construir sistemas robóticos complejos, escalables y robustos mejorando la eficiencia y permitiendo el desarrollo y mantenimiento de aplicaciones robóticas.

Utilizaremos ROS en el desarrollo del TFG para realizar la conexión con el simulador Airsim y el desarrollo del sistema de percepción del seguimiento del carril a través del controlador PID y aprendizaje por refuerzo.

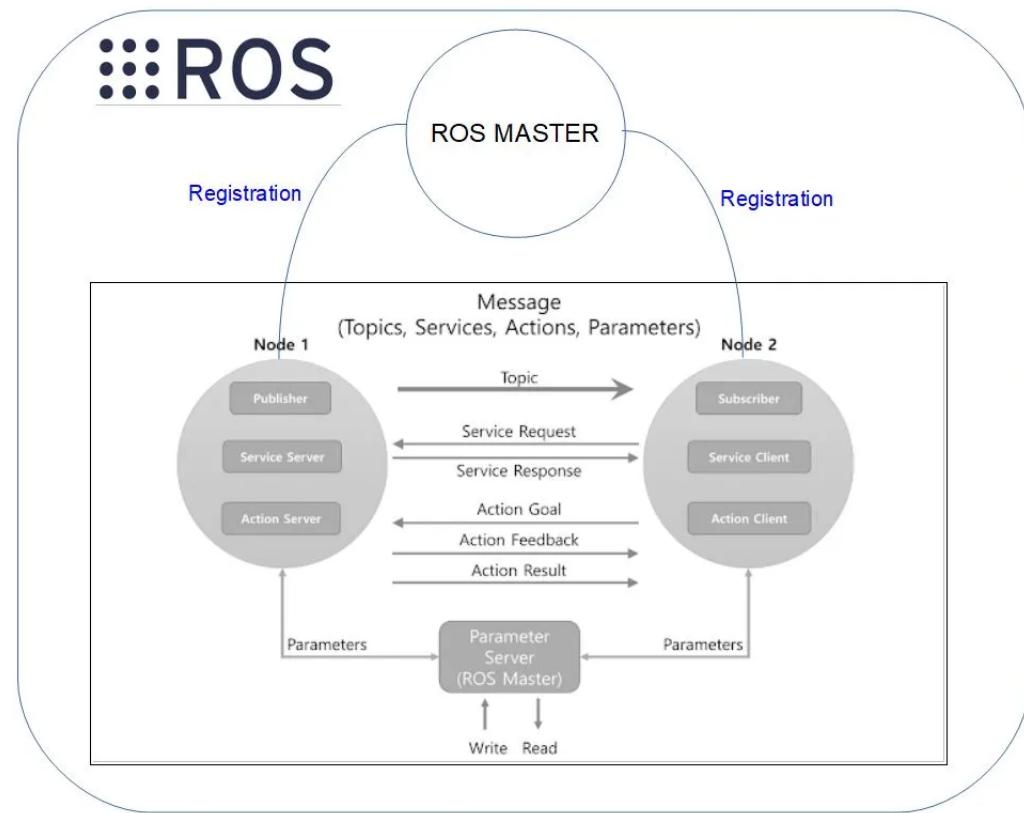


Figura 3.3: Arquitectura de ROS

3.3.1. Distribuciones

¿Qué es una distribución?

Una distribución ROS es un conjunto versionado de paquetes ROS, similar a las distribuciones de Linux (por ejemplo, Ubuntu). El principal objetivo que tiene ROS con las distribuciones es permitir a los desarrolladores trabajar con un código relativamente estable hasta que estén preparados y puedan publicar versiones robustas y estables.

En este TFG se ha utilizado ROS con la distribución Noetic⁹ debido al uso de Airsim, ya que dentro de Airsim las distribuciones más estables con ROS están entre las distribuciones melodic y noetic¹⁰.



Figura 3.4: ROS noetic y ROS melodic

3.3.2. Mavros

Mavros¹¹ es un paquete formado por **ROS** y el protocolo de comunicaciones ligero **MAVLink** (Micro Air Vehicle Link) diseñado por Lorenz Meier¹² bajo el LGPL licencia. Este protocolo es utilizado para enviar información de estado, para controlar el vehículo y recibir datos de telemetría. Fácil de implementar en sistemas con recursos limitados, lo que lo hace ideal para su uso en drones y otros vehículos aéreos no tripulados.

⁹<http://wiki.ros.org/noetic>

¹⁰https://microsoft.github.io/AirSim/airsim_ros_pkgs/

¹¹<http://wiki.ros.org/mavros>

¹²<https://www.technologyreview.es/listas/35-innovadores-con-menos-de-35/2017/inventores/lorenz-meier>

Ademas, **Mavros** traduce los mensajes **ROS** a mensajes **MAVLink** y viceversa por lo que permite que los datos y comandos fluyan entre **ROS** y el dron, permitiendo un control más sofisticado y una mayor funcionalidad.

Por lo que, en este TFG estudiaremos y analizaremos si **Mavros** se podría utilizar para el control del dron junto con **PX4 ArduPilot** (3.5) a través de Airsim.

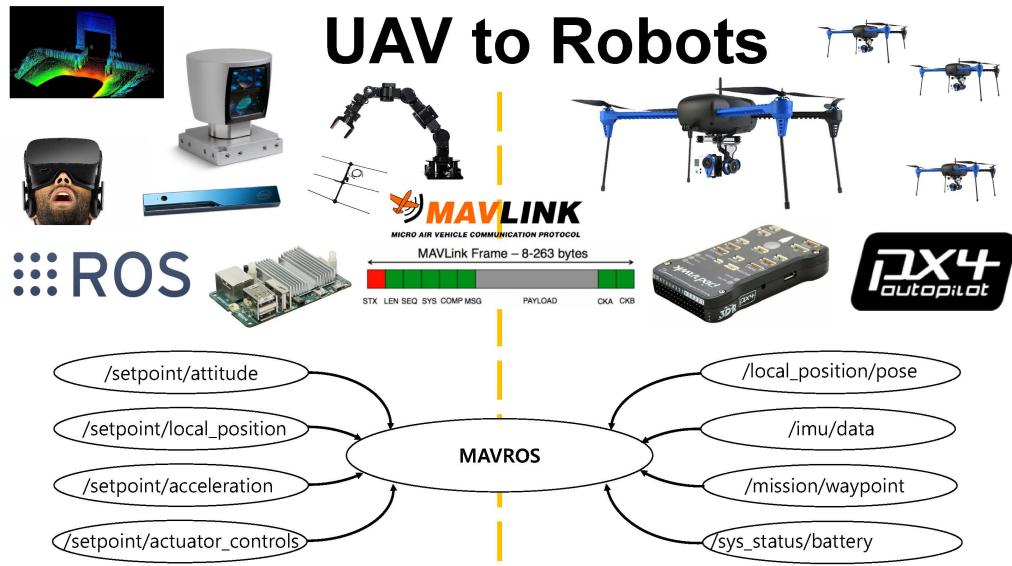


Figura 3.5: Infraestructura de Mavros

3.4. Entornos de Simulación

El entorno de simulación en el que vamos a estar trabajando será **Airsim** junto con **UnRealEngine** de Epic Games¹³. **Airsim**¹⁴ es un simulador de código abierto que se utiliza en aplicaciones robóticas y aprendizaje automático. Se construye sobre entornos 3D creados con **UnRealEngine**, estos entornos son utilizados para simular el mundo real y probar cómo los vehículos autónomos se comportarían en diferentes situaciones. **Airsim** es compatible en varias plataformas como Linux, Windows, macOS y también para Docker y WSL. En nuestro caso, se utilizará en Linux junto con **ROS**.

Por otro lado **UnRealEngine** es un motor de videojuegos que se utiliza para la

¹³<https://www.unrealengine.com/es-ES>

¹⁴<https://microsoft.github.io/AirSim/>

creación y simulación de entornos 3D realistas para videojuegos, películas animadas, experiencias interactivas y de realidad virtual. Es una propuesta innovadora utilizar este tipo de herramientas ya que puedes simular comportamientos físicos que se puedan producir en un entorno real.

3.4.1. Configuración de Integración Distribuida en equipos separados

En el desarrollo de este TFG, hemos tomado la decisión de tener dos enfoques:

1. El entorno de simulación (Airsim y UnRealEngine) estará en un ordenador con un sistema operativo Windows 10 con una gráfica Nvidia RTX 2070 Super con el plugin de **Airsim** al cargar el entorno mediante el motor UnRealEngine.
2. Las plataformas de desarrollo como ROS, Mavros, PX4 y QGC estarán en otro ordenador con un sistema operativo Ubuntu 18.04 con una gráfica Nvidia RTX 2070.

Esta propuesta fue tomada ya que al tener todo encapsulado en un único ordenador desembocaba a tener un bajo rendimiento del comportamiento que queremos desarrollar, por lo que finalmente al separar en dos partes, por una parte el simulador y por otra parte las plataformas de que utilizaremos a desarrollar los algoritmos.

De esta manera podemos tener un alto rendimiento por parte de ambos ordenadores y de las aplicaciones software que se utilizarán. Ambos ordenadores estarán comunicados a través de un router, cada equipo tendrá asignado una dirección IP y se comunicarán a través de protocolos de red TCP/UDP.

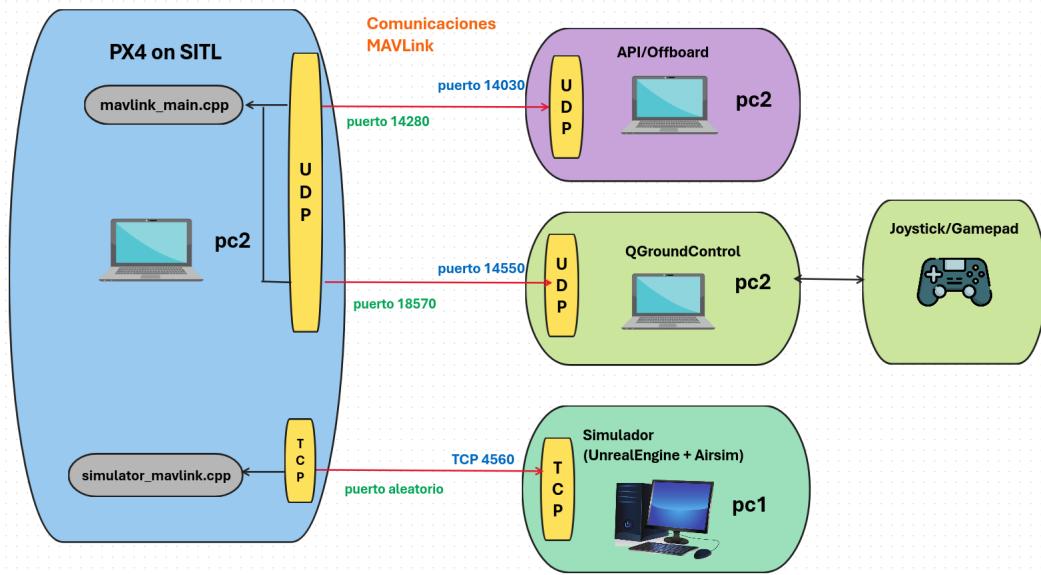


Figura 3.6: Esquema de comunicaciones entre las plataformas de desarrollo Mavros, PX4 y QGC junto con el entorno de simulación

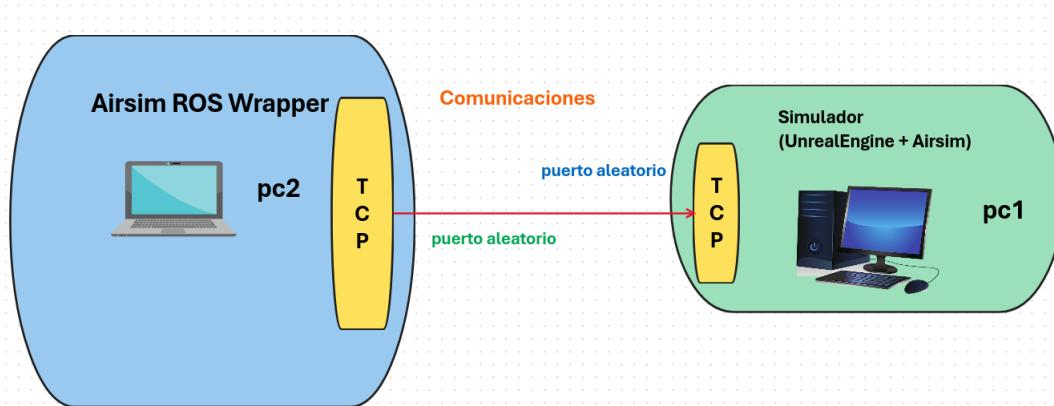


Figura 3.7: Esquema de comunicaciones entre las plataformas de desarrollo Airsim ROS Wrapper junto con el entorno de simulación

3.4.2. Airsim

Como hemos comentado anteriormente, **Airsim** es una buena opción de uso si queremos tener comportamientos similares a un entorno real. Ofrece una variedad de escenarios, tipos de vehículos, sensores y configuraciones del entorno según las necesidades u objetivos marcados de cada persona.

Para ello se debe todo configurar en un fichero de configuración con extensión json denominado settings.json ,lo cual para configurar el vehículo con nuestras necesidades necesitaremos definir diferentes variables.

Un archivo settings.json es un archivo de configuración específica de Airsim que define cómo se ejecutará la simulación en términos de propiedades del vehículo, configuración de sensores, condiciones climatológicas y más.

Un archivo settings.json consta de varias secciones:

1. **SimMode:** Este parámetro define el modo de simulación, se refiere si el modo de simulación es para coches, multirotores o vision de computador.
2. **ClockType:** Determina qué tipo de reloj se utiliza para medir el tiempo en la simulación.
3. **Vehicles:**Configuración de las propiedades de cada vehículo individualmente. Puedes especificar el tipo de vehículo, la posición inicial, la dinámica del vehículo, entre otros.
 - **VehicleType:** En este caso ese parámetro es el tipo de vehículo que utilizaremos en la simulación.
 - **UseSerial:** Es para saber si vamos a usar un puerto serial en fisico si utilizamos un vehículo en un entorno real.
 - **LockStep:** Es una característica importante cuando se comunica con el simulador AirSim a través de TCP.
 - **UseTcp:** Para poder comunicarnos a través de TCP necesitamos habilitar esta opción a true.
 - **TcpPort:** Especificamos el puerto TCP que vayamos a usar.
 - **ControlIp:** Esta opción es para especificar si el comportamiento se realizará simulado.

- **ControlPortLocal:** Se especificará el puerto Local.
- **ControlPortRemote:** Se especificará el puerto Remoto.
- **LocalHostIp:** La dirección IP del ordenador en donde llevaremos la simulación.
- **Parameters:** Estos parametros son de PX4 y permite la configuración del vehículo, como por ejemplo los modos de vuelo, sus configuraciones, configuraciones de velocidades y más.
- **Sensors:** Permite personalizar la configuración de los sensores simulados, como Lidar, IMU (Unidad de Medición Inercial), GPS y sensor de distancia.
- **Cameras:** Puedes configurar las cámaras utilizadas en la simulación, especificando sus propiedades como resolución, tipo de lente, posición y orientación relativas al vehículo.

Nosotros usaremos una cámara que proporcionará una imagen RGB de dimensiones 620x620 pixeles y activaremos el flag de PublishToRos a 1 para poder acceder a ella mediante el Airsim ROS Wrapper.

Para más detalles sobre el archivo de settings.json está la página oficial de Airsim¹⁵

Escenarios

Los escenarios que ofrece Airsim depende en que sistema operativo nos encontramos, en nuestro caso al utilizar el escenario en Windows tenemos más variedad que en comparación con Linux.

Tiene escenarios desde carreteras y ciudades con coches simulados hasta entornos industriales como almacenes y entornos de montaña con carreteras. En nuestro caso hemos utilizado el escenario Coastline, consiste en un entorno fotorrealista de un recorrido amplio de 2 carriles con ambiente tropical. Con este entorno tendremos la información del entorno para realizar el comportamiento sigue carril con el dron.

Todos estos escenarios se pueden encontrar en las releases de Airsim¹⁶ tanto para Windows como para Linux. En nuestro caso hemos utilizado el escenario Coastline, ya

¹⁵<https://microsoft.github.io/AirSim/settings/>

¹⁶<https://github.com/Microsoft/AirSim/releases>

que queremos desarrollar un comportamiento de seguimiento de carril y este escenario ofrece un amplio recorrido de 2 carriles para poder llevarlo a cabo.



Figura 3.8: Ejemplos de escenarios en Airsim

Sensores

Ofrece sensores como cámaras, barómetros, Imus, GPS, Magnetómetros, sensores de distancia y Lidar. En nuestro caso utilizaremos como sensores una cámara para poder realizar la detención del carril que queremos seguir, el sensor Lidar para saber a qué altura se encuentra el dron respecto al suelo y el sensor GPS para poder obtener la localización del vehículo.

Tipos de vehículos

Airsim ofrece dos tipos principales de vehículos para la simulación: coches y drones. Dentro de estos tipos se encuentran los subtipos de coches y drones que se puede utilizar.

1. Coche

- **PhysXCar:** Representa un vehículo en tierra con física realista basado en el motor de física PhysX.
- **ArduRover:** Se utiliza para vehículos terrestres que sigan el estándar ArduRover. ArduRover¹⁷ se trata de un piloto automático de código abierto

¹⁷<https://ardupilot.org/rover/>

utilizado específicamente para vehículos terrestres.

2. Dron

- **SimpleFlight:** Representa un dron con un modelo de vuelo simplificado. Este tipo de opción puede ser útil si queremos simular comportamientos de movimiento básico para los drones.
- **PX4Multirotor:** Representa un dron mediante PX4 ArduPilot.
- **ArduCopter:** Representa un dron pero siguiendo el estándar ArduCopter. ArduCopter¹⁸ se trata de un piloto automático de código abierto utilizado para los drones

Como hemos numerado anteriormente, este entorno de simulación tiene un catálogo de vehículos, sensores y cambios climatológicos dentro del entorno. Nosotros utilizaremos un dron de tipo "SimpleFlight".

3.4.3. Airsim ROS Wrapper

Utilizaremos el paquete **Airsim ROS Wrapper**¹⁹ para poder acceder a ciertos sensores que serán necesarios como es la cámara, el Lidar y el GPS, pero antes de comentar lo que ofrece hablaremos sobre qué es un ROS Wrapper.

ROS Wrapper es un componente que facilita la integración entre dos sistemas o entornos diferentes. Si lo llevamos al contexto de **ROS**, un wrapper es un nodo o paquete que permite que los componentes de **ROS** se comuniquen con otros sistemas o bibliotecas que no fueron originalmente diseñadas para trabajar con **ROS**. Este paquete puede proporcionar publicación de datos desde el sistema externo a **ROS** a través de topics, suscripción a topics de **ROS** para recibir comando o datos, adaptación de interfaces de llamada (por ejemplo, entre C++ y Python).

Por lo tanto, **Airsim ROS Wrapper** es un paquete de **ROS** que comunicará **ROS** y **Airsim**. Este paquete contiene dos nodos principales:

1. **AirSim ROS Wrapper Node:** Este nodo proporciona una interfaz **ROS** para acceder a los datos del vehículo simulado, por ejemplo, sus sensores, proporcionar velocidades, acceder a su sistema de referencia, etc.

¹⁸<https://ardupilot.org/copter/>

¹⁹https://microsoft.github.io/AirSim/airsim_ros_pkgs/

2. **Simple PID Position Controller Node:** Este nodo es un controlador de posición simple basado en un controlador PID (proporcional-derivativo-integral). Ayuda controlar la posición del vehículo simulado en el entorno Airsim.

La primera aproximación fue utilizar **AirSim ROS Wrapper Node** para poder dar un comportamiento al vehículo mediante velocidades. La dificultad que nos podemos encontrar es al comandar las velocidades, la altura del vehículo no es constante en el aire, es decir, a medida que el vehículo avanza, el dron va perdiendo altura como resultado acabando en el suelo. Esto se debe a que **AirSim ROS Wrapper Node** no proporciona ningún controlador de posición para el eje z que correspondería con la altura y ni ningún controlador de velocidades en los ejes x e y.

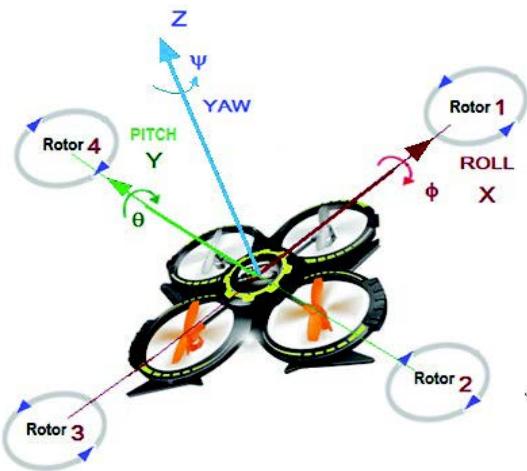


Figura 3.9: Ilustración de los ejes de referencia del dron

Es cierto que **Airsim** y **ROS** ofrecen **Simple PID Position Controller Node**, pero el controlador de este nodo es para todos los ejes (x,y e z) lo cual lo que necesitamos sería un controlador de posición para el eje z para cuando comandamos velocidades en los ejes x e y, y velocidades angulares en el eje z. Al obtener estos resultados, escogimos una de posibles soluciones de integrar un pequeño controlador PID junto con el sensor Lidar que ofrece el **AirSim ROS Wrapper Node** para poder controlar la altura del vehículo durante su navegación.

3.4.4. Client Airsim

Airsim ofrece una API implementada para Python denominada Client Airsim²⁰, en donde podemos conectarnos con el simulador y tener el control de los sensores y actuadores del vehículo. Esta interfaz es bastante útil ya que podemos tener el control del vehículo simulado sin necesidad de tener que utilizar los topics de ROS, es decir, existen métodos los cuales podemos comandar velocidades al vehículo, tener acceso a los sensores como las cámaras o cambiar configuraciones de la simulación como por ejemplo el clima, el viento, el tiempo del día o la densidad de tráfico en escenarios donde aparecen coches simulados.

Esta API la utilizaremos sobre todo para tener el control del vehículo para realizar su navegación con los controladores PID y en el desarrollo de aprendizaje por refuerzo.

3.5. PX4 AutoPilot

PX4²¹ es una plataforma de software de código abierto para desarrolladores de drones que les permite crear y controlar diversos tipos de drones, desde aplicaciones de consumo hasta aplicaciones industriales.

Unas de las principales características que ofrece esta plataforma es el soporte de múltiples tipos de vehículos, como aviones de ala fija, multirrotores, helicópteros, rovers y vehículos submarinos, también proporciona diferentes modos de vuelo, navegación por puntos de referencias predefinidos, estabilización del vehículo.

En este trabajo realizaremos un análisis respecto a la navegación del dron mediante **PX4** con el modo de simulación Software in The Loop(SITL) para tener el control el vehículo junto con **Mavros** y **Airsim**.

3.5.1. Software in The Loop(SITL)

Este modo de simulación permite a los desarrolladores probar y depurar códigos de control de drones sin necesidad de hardware físico, en lugar de ejecutar el código en un vehículo real, este modo simula el comportamiento del vehículo en una computadora. Es especialmente útil durante el desarrollo y la validación de control, navegación y

²⁰<https://microsoft.github.io/AirSim/apis/>

²¹<https://docs.px4.io/main/en/>

planificación de misiones.

Podemos tener diversos entornos de simulación con este modo como Gazebo, Airsim y jMAVSim. Dichos entornos de simulación permiten realizar simulaciones muy realistas y avanzadas de cualquier tipo de vehículo simulando una gran variedad de parámetros.

Para poder utilizar PX4 SITL, se debe configurar el entorno de desarrollo adecuado y seguir las instrucciones proporcionadas por la comunidad²². En nuestro caso realizaremos la configuración PX4 SITL junto con Airsim para estudiar si es posible tener un control en el comportamiento a querer desarrollar.

3.5.2. Modos de vuelo

PX4 ofrece varios modos de vuelo por ejemplo como Takeoff,Land,Hold, Position, Offboard, etc. Un modo de vuelo define como el usuario puede controlar el vehículo a través de comandos y ver que respuesta tiene.

1. **TAKEOFF:** Este modo de vuelo permite despegar el vehículo con una altitud y una velocidad de ascendente escogida por el usuario con los parámetros de PX4 MIS_TAKEOFF_ALT y MPC_TKO_SPEED, dichos parámetros tienen valores por defecto definidos por la plataforma (2.5 m y 1.5 m/s respectivamente). Antes de realizar el despegue el vehículo será armado para poder realizarlo.

Una vez se realice el despegue del vehículo se pasa al modo HOLD

2. **LAND:** Permite aterrizar el vehículo en donde nos encontremos en ese instante, una vez el vehículo sea aterrizado se desarmará por defecto. En este modo podemos cambiar por ejemplo la tasa de descenso durante se realiza el aterrizaje del vehículo con el parámetro MPC_LAND_SPEED, tiempo en segundos para que se realice el desamardo del vehículo si se establece dicho tiempo con un valor de -1 el vehículo no se desarmará cuando aterrice.

Cuando se realice este modo de vuelo por defecto se cambiará al modo de Position

3. **HOLD:** A partir de este modo de vuelo podemos parar el vehículo manteniéndolo en el aire con su actual posición GPS y altitud. Este modo puede ser bastante útil para cuando queremos pausar una misión o reiniciar el comportamiento que queramos realizar.

²²<https://docs.px4.io/v1.14/en/simulation/>

4. **POSITION:** Es un modo de vuelo manual el cual puedes controlar el vehículo mediante un joystick, dicho vuelo controla la posición del vehículo cuando comandemos velocidades mediante el joystick. Este modo de vuelo lo utilizaremos para teleoperar el dron para ver el funcionamiento de la percepción por parte de la red neuronal que utilizaremos. Dicho vuelo ofrece un gran catálogo de parametros que puede afectar al vuelo.

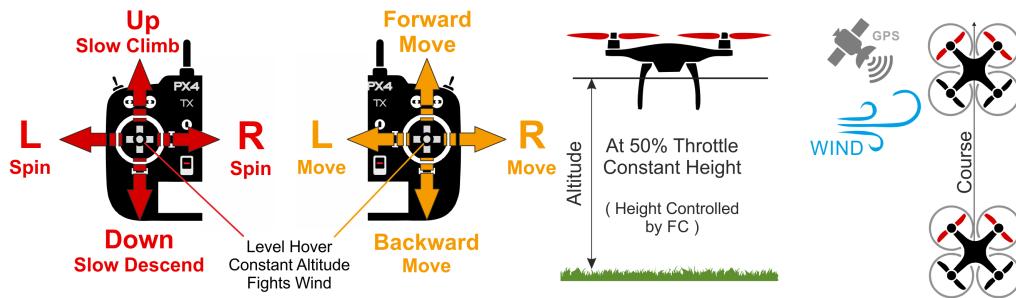


Figura 3.10: Diagrama del comportamiento del modo de vuelo Position

5. **OFFBOARD:** Con este modo de vuelo podemos controlar el movimiento y la altitud del vehículo a partir de comandos de posición, velocidad, aceleración, altitud, velocidades de altitud o puntos de ajuste de empuje/torque.

Dichos comandos debe ser una secuencia de mensajes de setpoint MavLink o a través de topics mediante ROS con Mavros.

En este modo, PX4 debe recibir una secuencia de mensajes continua. Si en algún momento dejamos de publicar mensajes, el control externo de PX4 dejará de estar en el modo Offboard después de pasar un tiempo de espera establecido por el parámetro COM_OF_LOOS_T (por defecto esta establecido a 1 s) e intentará aterrizar o realizar alguna acción de seguridad (dichas acciones de seguridad vienen definidas en la sección de Failsafes en PX4 Autopilot²³). La acción dependerá si el control RC está disponible, si este control esta disponible pasará a otro tipo de modo de vuelo definido en el parámetro COM_OBL_RC_ACT.

Para comandar las velocidades al vehículo mediante Mavros, se tendrá que utilizar el topic denominado /mavros/setpoint_velocity/cmd_vel_unstamped dicho topic utiliza el un marco de coordenadas por defecto definido en el archivo de

²³<https://docs.px4.io/v1.14/en/config/safety.html>

configuración de px4.config.yaml LOCAL_NED. Si queremos que el marco de coordenadas se mueva con el cuerpo del vehículo se tendrá que utilizar el marco de coordenadas En nuestro caso necesitamos un marco de coordenadas diferente para que el vehículo se mueva con el cuerpo del vehículo, por ello utilizaremos el marco de coordenadas BODY_NED.

Los marcos de coordenadas que ofrece Mavros se puede ver a través del servicio SetMavFrame.srv²⁴

3.6. QGroundControl

QGroundControl²⁵ es una plataforma de software que proporciona un control completo de vuelo y configuraciones de vehículos para drones por **PX4 ArduPilot**. Además, ofrece un control total durante el vuelo y permite la planificación de vuelos autónomos mediante la definición de puntos de referencia, se muestra la posición del vehículo junto con su trayectoria, los puntos de referencia y los instrumentos del vehículo. Es una opción cómoda para poder visualizar tu vehículo y querer cambiar parámetros del vehículo mediante esta aplicación y poder teleoperar el vehículo a través de un mando joystick.

Funciona en diferentes plataformas como Windows, macOS, Linux,iOS y dispositivos Android, en nuestro caso lo utilizaremos en Linux.

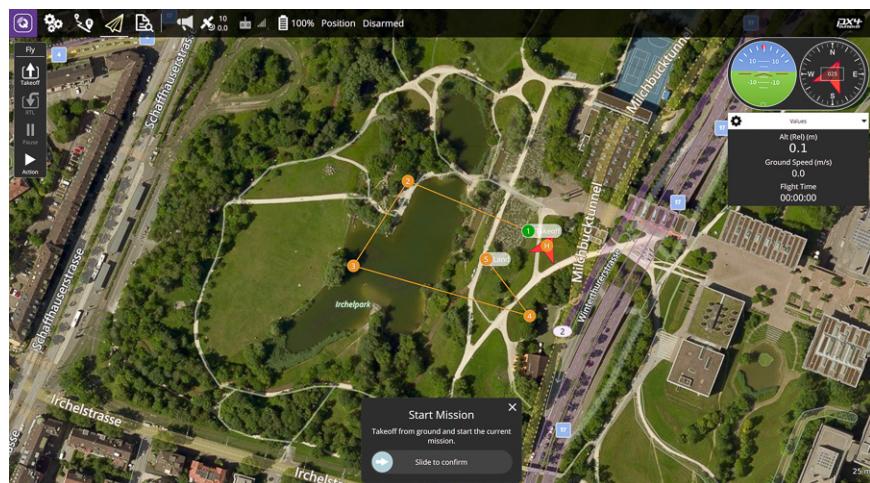


Figura 3.11: QGroundControl

²⁴https://github.com/mavlink/mavros/blob/master/mavros_msgs/srv/SetMavFrame.srv

²⁵<http://qgroundcontrol.com>

Capítulo 4

Anexo

A continuación se muestra las diferentes referencias a las figuras que hemos visto a lo largo de este trabajo junto con el enlace de donde ha sido obtenida. Las imágenes que no incluidas en este capítulo han sido formadas en el desarrollo de este trabajo provienen del mismo:

| Referencia de las imágenes | Enlaces de donde se ha obtenido |
|----------------------------|---|
| 1.2 | https://airandspace.si.edu/multimedia-gallery/web12070-2011640.jpg |
| 1.3 | https://www.iotworldtoday.com/robotics/boston-dynamics-spot-the-design-behind-the-robot-dog |
| 1.4 | https://www.smithsonianmag.com/arts-culture/unmanned-drones-have-been-around-since-world-war-i-16055939/ https://web.happystays.com/?m=file-winston-churchill-and-the-secretary-of-state-for-tt-YQ3jGVI4 https://en.wikipedia.org/wiki/V-1_flying_bomb https://www.google.com/url?sa=i&url=https%3A%2F%2Fthefrontlines.com%2Fstory%2Ffw2-project-aphrodite%2F&psig=A0vVaw20cBlgMDH1HVU5qsiJ9_Fg&ust=1714151925046000&source=images&cd=vfe&opi=89978449&ved=OCBQQjhxqFwoTCNjGs9bv3YUDFQAAAAAdAAAAABAE https://en.wikipedia.org/wiki/Ryan_Firebee https://en.wikipedia.org/wiki/Lockheed_D-21 https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FBoeing-Condor-UAV-23_fig10_261209014&psig=A0vVaw1q3J6eh2YCyEUzy1QM9z-K&ust=1714152091161000&source=images&cd=vfe&opi=89978449&ved=OCBQQjhxqFwoTCJDyqXw3YUDFQAAAAAdAAAAABAE https://www.timesofisrael.com/idf-launches-probe-after-two-more-mini-drones-crash/ https://www.google.com/url?sa=i&url=https%3A%2F%2Fnews.usni.org%2F2020%2F09%2F15%2Fmarines-placing-small-uavs-into-ground-combat-element-as-aviators-still-refining-large-uas-requirement&psig=A0vVaw2csv7vma6UxJokGv1G8j7h&ust=1714152048517000&source=images&cd=vfe&opi=89978449&ved=OCBQQjhxqFwoTCLiCtZhW3YUDFQAAAAAdAAAAABAE |
| 1.5 | https://www.xataka.com/espacio/helicoptero-ingenuidad-ha-terrizado-lugares-marte-que-nasa-se-esta-quedando-letras-para-nombrarlos |

| | |
|------|---|
| 1.6 | https://www.elcorreogallego.es/hemeroteca/union-fenosa-distribucion-implanta-uso-drones-supervisar-sus-lineas-alta-tension-galicia-MQCG1024080 |
| 1.7 | https://www.xataka.com/drones/asi-es-el-drone-repartidor-de-amazon-todavia-poco-mas-que-humo-que-promete-entregar-paquetes-en-media-hora |
| 1.8 | https://www.techtimes.com/articles/285562/20221228/amazon-begins-prime-air-drone-deliveries-california-texas.htm |
| 1.9 | https://emprendedores.es/marketing-y-ventas/ecommerce-marketing-y-ventas/drones-amazon-europa/ |
| 1.11 | https://www.rebellionresearch.com/what-is-the-disadvantage-of-deep-reinforcement-learning |
| 2.1 | https://github.com/RoboticsLabURJC/2022-tfg-barbara-villalba/graphs/contributors |

| | |
|------|---|
| 3.1 | https://pytorch.org/hub/hustvl_yolop/ |
| 3.2 | https://www.ros.org/imgs/ros-equation.png |
| 3.3 | https://medium.com/@robtech.impaciente/ros-robot-operating-system-fundamentos-e92478c26e02 |
| 3.4 | https://www.openrobotics.org/blog/2020/5/23/noetic-ninjemys-the-last-official-ros-1-release https://www.ros.org/news/2018/04/ros-melodic-morenia-logo-and-tshirt-campaign.html |
| 3.5 | https://404warehouse.net/2015/12/20/autopilot-offboard-control-using-mavros-package-on-ros/ |
| 3.8 | https://img-blog.csdnimg.cn/272026cef41047cdb7e523fb9a28e173.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBAamluYXV0bw==,size_20,color_FFFFFF,t_70,g_se,x_16 https://www.scrimmagesim.org/sphinx/html/_images/Asset_LandscapeMountains_1.png https://www.researchgate.net/figure/Appearance-of-the-maps-for-training-a-City-environment-b-Coastline-c_fig7_359436337 https://www.scrimmagesim.org/sphinx/html/_images/city_airsim_view.png https://github.com/Microsoft/AirSim/wiki/moveOnPath-demo |
| 3.9 | https://recimundo.com/index.php/es/article/view/814/1323 |
| 3.10 | https://docs.px4.io/v1.14/en/flight_modes_mc/position.html |
| 3.11 | https://flathub.org/es/apps/org.mavlink.qgroundcontrol |

Apéndice A

Bibliografía

- [1] Alavarez, D. A. R. (2016). The condor uav system.
- [2] Dong, W., Man-Wen, L., Wei-Tian, Z., Xing-Gang, W., Xiang, B., Wen-Qing, C., and Wen-Yu, L. (2012). Yolop: You only look once for panoptic driving perception. *Machine Intelligence Research*, 19:253–266.
- [3] Frisbee, J. L. (1997). Proyect aphrodite.
- [4] Gustavo Mesías-Ruiz, J. P., Ana de Castro, I. B.-S., and Dorado, J. (2024). Detección y clasificación de malas hierbas mediante drones y redes neuronales profundas: creación de mapas para tratamiento localizado. pages 1–5.
- [5] Jung, S. and Kim, H. (2017). Analysis of amazon prime air uav delivery service. *Journal of Knowledge Information Technology and Systems*, 12:253–266.
- [6] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Alexander Herzog, E. J., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. pages 1–23.
- [7] Krejci Garzon, E. (2014). Drones el futuro de hoy. *ashtag*, pages 96–103.
- [8] Loja Romero, J. D. (2022). Exploración autónoma en interiores para el robot spot basado en la red yolo. No Publicado.
- [9] Qiang, W. and Zhongli, Z. (2011). Reinforcement learning model, algorithms and its application. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 1143–1146.
- [10] Xue, Z. and Gonsalves, T. (2021). Vision based drone obstacle avoidance by deep reinforcement learning. 2:366–380.