# Borrador Memoria TFG 2.0

***Entorno de simulación para el entrenamiento mediante Reinforcement Learning del vuelo autónomo de un cuadricóptero***

**Índice**

1. *Introducción*
2. *Objetivos*
3. *Resultados*
4. *Conclusiones*
5. *Bibliografía*

*Anexos:*

* *Estado del Arte*

## INTRODUCCION

Presentar los drones. Contextualizar su uso hoy en dia (y el que se dará mañana), e introducir la necesidad de tener sistemas para entrenar drones inteligentes.

Introducir el RL y explicar la necesidad de hacer pruebas “una y otra vez”

Una de las tecnologías que más destacan y llaman la atención en estos últimos años son los llamados drones, definidos formalmente como *aeronaves no tripuladas* por la RAE [http://dle.rae.es/?id=ED2QqnQ]. Estos aparatos se han convertido en una atracción para casi cualquier ámbito, comenzando en usos militares y llegando a ser juguetes de ocio, no sin pasar por usos comerciales como el transporte o relacionados con seguridad y defensa.

Su bajo coste, reducido tamaño y la posibilidad de manejarlos remotamente les da un potencial inmenso para una infinidad de usos, entre otras cosas, en tareas que debería hacer una persona pudiendo tener que asumir riesgos o peligros, como puede ser la vigilancia desde el aire en zonas de conflicto, la búsqueda de supervivientes tras una catástrofe natural o la actuación en incendios forestales.

Sin embargo, para algunas tareas los recursos humanos son limitados y no es viable ocupar a un operario pilotando un dron, apareciendo de esta forma la necesidad de diseñar aeronaves que puedan llevar a cabo su cometido de forma autónoma.

Volar de forma autónoma es una tarea muy compleja en la que pueden suceder innumerables situaciones diferentes e imprevisibles, a las que se debe dar respuesta de la forma más rápida y acertada posible. Es necesario dotar a los drones de inteligencia para que puedan desenvolverse por sí mismos y aprender de su propia experiencia.

Esta necesidad se puede satisfacer con técnicas de Reinforcement Learning (o Aprendizaje por Refuerzo), un área del Machine Learning que agrupa técnicas de aprendizaje automático basadas en la experimentación y la experiencia de la propia máquina. Estas técnicas comparten una etapa de entrenamiento basada en sucesivos episodios de prueba y error en los que la máquina debe procurar llegar a un objetivo. Durante estos episodios el aparato aprende de sus propias interacciones con el entorno que lo rodea.

En este trabajo fin de grado se pretende crear un freamwork de simulación, desarrollo y entrenamiento de sistemas RL orientado a su uso en drones.

## OBJETIVOS

Objetivo principal: crear un framework de simulación, desarrollo y testeo de sistemas RL realista, versátil y plug&play.

* El objetivo de este tfg es crear un framework de simulación y entrenamiento para drones. Es decir, un conjunto de herramientas que nos ayuden a diseñar y probar algoritmos, así como a entrenar el dron con esos algoritmos. De forma que una vez el dron esté preparado, solo haya que mover la lógica (incluyendo ese “conocimiento generado”) al dron real, y a volar.
* Ventajas de esto? Que no hay que modificar nada, lo que se entrena es lo mismo que va a llevar el dron real (filosofia plug&play)

REESCRIBIIIIIR

El objetivo final de este trabajo es la obtención de un freamwork de simulación, desarrollo y entrenamiento de sistemas RL, es decir, un conjunto de herramientas que faciliten el desarrollo y posterior entrenamiento (en simulación) de sistemas inteligentes (basados en RL) para drones.

Para ello se empleará AirSim como entorno de simulación por su gran realismo, el PX4 como controlador, se utilizará ROS por su versatilidad para programar todo tipo de algoritmos y para controlar el PX4 y se integrará la plataforma GYM para facilitar el desarrollo de los sistemas de entrenamiento.

## RESULTADOS

* Explicar los bloques que componen el sistema completo (airsim, px4, ros, gym) y cómo se integran.
* A continuación, explicar las implementaciones llevadas a cabo en cada uno de los bloques. (airsim con integración de ROS, px4 modificado para poder hacer el reset, ROS + GYM para unir todo y hacer el sistema de entrenamiento)
* Por último, mencionar las pruebas y el desarrollo en general que se llevó a cabo durante la implementación de esos módulos.

## CONCLUSIONES

Finalmente que hemos conseguido? Hemos creado un framework con X ventajas y Y inconvenientes. Soluciona estos problemas del estado del arte.

Recalcar que se ha creado algo nuevo que aporta valor, y que no existía previamente.

------- Puede que se aproveche algo de esto ------

La ventaja principal que aporta este framework es la facilidad de migración del sistema entrenado al dron real. Esto se debe a que, gracias a la compatibilidad del simulador con un controlador tan ampliamente utilizado como es el px4, cualquier implementación de software en el framework funcionará exactamente de la misma manera en el vehículo físico.

La ventaja principal de este framework pues, reside en que todas las implementaciones son migradas de la simulación al equipo real de forma casi directa, sin necesidad de hacer ningún cambio en la implementación. Esto se consigue gracias a la integración en la simulación de un firmware de autopiloto creado para equipos reales, por lo que el sistema desarrollado se programará para la simualción exactamente igual que se haría para el dron físico.

En concreto, una característica que diferencia notablemente este trabajo de cualquier otro mencionado en el estado del arte, es su filosofía plug&play. El AirSim es compatible con un autopilot ampliamente utilizado en drones autónomos, el PX4. Esto da una ventaja enorme, ya que cualquier código que funcione para este autopilot, funcionará exactamente igual para un dron real con el mismo controlador.

---

## ANEXOS

### ESTADO DEL ARTE

Dado que el objetivo es tener un framework de simulación, desarrollo y testeo de sistemas RL, podemos hablar de los diferentes frameworks que hay para simulación (aptdo simuladores).

Dentro de los frameworks hay que mencionar al final de todo, cuando ya nos centramos en airsim, que existe un freamwork similar al que hemos creado pero que utiliza el simple\_flight. La ventaja del nuestro esta en que utiliza el px4, ya que en las ultimas versiones de airsim, integra ros tb, cosa queno hacia al principio.

Acabar hablando de por que escogemos airsim? Si no, se puede meter en SIMULADOR, pero pienso que quedará mejor aquí.

Desarrollar y experimentar con algoritmos para vehículos autónomos es un proceso muy complejo que requiere tiempo y muchas pruebas que no siempre terminan bien. Además, para desarrollar sistemas inteligentes a menudo es necesario recopilar una gran cantidad de datos de entrenamiento en diferentes condiciones y entornos. Realizar este proceso en el mundo real no es viable, por ello para llevarlo a cabo se utilizan entornos de simulación, sistemas informáticos que procuran imitar lo más fielmente posible el comportamiento que tendría el vehículo en un entorno real. Este tipo de software nos permite experimentar, analizar y entrenar a nuestros aparatos evitando los importantes inconvenientes que implican hacer lo propio en la realidad: accidentes, golpes, costosas roturas, desajustes, etc.

Cabe destacar que no solo el entorno de simulación es importante, sino también su compatibilidad e integración con otras tecnologías tales como los **controladores de vuelo**, que son dispositivos encargados de controlar los diferentes componentes del vehículo; los **protocolos de comunicación** para interconectar el simulador y el controlador, así como también otros componentes; o los **frameworks de programación** para desarrollar el software que definirá el comportamiento del dron.

Debido a que el interés de este trabajo reside en encontrar un sistema que agrupe todas estas tecnologías para facilitar la implementación de vehículos autónomos, es esencial hacer un análisis del estado del arte al respecto:

*Datos comparativos entre simuladores sacados del paper de AirSim* ***[*** <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/07/1705.05065.pdf> ***]***

A día de hoy existen variedad de entornos de simulación para drones, y entre ellos, uno de los más utilizados es Gazebo. [WEB <http://gazebosim.org/tutorials?tut=guided_b1&cat=#WhatisGazebo?> ] Un simulador de código abierto para todo tipo de robots, capaz de simular ambientes interiores y exteriores complejos con alto grado de fidelidad. Provee una amplia biblioteca de robots y entornos, gran cantidad de sensores y plugins para integrar muchas otras tecnologías, tales como controladores de vuelo, frameworks de programación, sistemas de entrenamiento, etc. Algunas de las grandes ventajas que tiene utilizar Gazebo son la amplia comunidad de usuarios que participan activamente generando conocimiento y la gran cantidad de documentos académicos basados en la experimentación con este simulador.

La principal desventaja de Gazebo es su falta de realismo en la simulación, debido a su orientación a robots genéricos y a la complejidad que implica crear entornos realistas a gran escala y con gran cantidad de detalles que simulen mejor el mundo real.

Una alternativa que pretende solventar la falta de realismo es Hector [PAPER <https://pdfs.semanticscholar.org/3ed3/948827e0949770e8583b51bd0fedf4fd73fe.pdf>], un trabajo orientado específicamente a cuadricópteros, y que integra el simulador Gazebo junto al middleware robótico ROS [ Referencia (anexo? Web?) ] . Lo que ofrece Hector [ fuente <http://wiki.ros.org/hector_quadrotor> ] es una simulación más realista de la física del vuelo y el comportamiento del aparato, pero por el contrario carece de soporte para la integración de controladores de vuelo como PX4 y protocolos de comunicación estándares como MavLink [ WEB <http://qgroundcontrol.org/mavlink/start> ].

De forma similar, RotorS [ PAPER <https://link.springer.com/chapter/10.1007/978-3-319-26054-9_23> ] es un framework modular de simulación que permite diseñar drones y desarrollar algoritmos para controlar su comportamiento y estimar su estado de forma más precisa. Al igual que en el caso anterior, RotorS [ fuente <http://wiki.ros.org/rotors_simulator> ] se basa en ROS y Gazebo, pero en este caso sí se soporta la integración de controladores de vuelo.

Garantizando la compatibilidad de controladores de vuelo y alejándose de la simulación de Gazebo, existe otro software llamado jMAVSim [ WEB <https://pixhawk.org/dev/hil/jmavsim> ], un extremadamente sencillo entorno de simulación diseñado en el contexto del proyecto Pixhawk [ <http://pixhawk.org/> ] con el objetivo de experimentar con el firmware de la plataforma del propio proyecto. Se trata de un entorno muy ligero y fácil de utilizar que consta de un motor de renderizado muy simple, imposibilitando la generación de objetos en la escena. Además no aporta ningún tipo de integración con otras tecnologías.

Otro grupo de entornos de simulación es aquel basado en el potente motor de videojuegos creado por Epic Games [ WEB <https://www.epicgames.com/site/es-ES/home> ], el Unreal Engine [ WEB <https://www.unrealengine.com/en-US/what-is-unreal-engine-4> ]. Este grupo de frameworks se caracteriza por el alto grado de realismo tanto en gráficos como en física de comportamiento de los vehículos, debido a las avanzadas tecnologías desarrolladas por su creador. Notables ejemplos de este grupo son Sim4CV, DroneSimLab y Microsoft AirSim.

El primero de ellos, Sim4CV [ WEB <https://sim4cv.org> ] [ <https://arxiv.org/pdf/1708.05869.pdf> ] [ <https://www.youtube.com/watch?v=SqAxzsQ7qUU> ], se presenta como un simulador fotorrealista orientado claramente a la experimentación en campos muy avanzados de la visión artificial. Gracias al potente motor gráfico, consigue alcanzar un alto nivel de realismo en la física de los vehículos. Como desventaja está su limitación para integrar tecnologías externas al propio freamwork.

En contraste con el anterior, DroneSimLab [ <https://arxiv.org/abs/1708.01938> ][ <https://github.com/orig74/DroneSimLab> ] es un freamwork con énfasis en la modularidad y la flexibilidad. Tanto es así, que no solo integra varios controladores de vuelo y motores de simulación de sensores, además del motor gráfico aportado por Unreal, sino que facilita la integración de otros muchos componentes mediante el uso de contenedores. Este conjunto de herramientas tiene una orientación genérica, permitiendo experimentar en visión artificial, vehículos autónomos y otros muchos campos para gran cantidad de vehículos. Claramente, este freamwork es la alternativa más cercana e interesante al objetivo del presente trabajo.

Microsoft AirSim [ WEB <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/07/1705.05065.pdf> ] forma parte del proyecto *Aerial Informatics and Robotics Platform* [ WEB <https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/> ] desarrollado por Microsoft, y consiste en un entorno de simulación basado en el motor Unreal Engine y en un conjunto de herramientas para el desarrollo de sistemas autónomos. Se trata de una plataforma que cuenta con las tecnologías más avanzadas de simulación en cuanto a física de los vehículos y visualización del entorno, reduciendo al máximo la diferencia entre realidad y simulación, ayudando así a crear sistemas autónomos más avanzados, seguros y eficaces.

AirSim es un freamwork de código abierto en continuo desarrollo que pretende convertirse en una plataforma modular, altamente compatible y tecnológicamente avanzada para contribuír al desarrollo e investigación en vehículos autónomos, con la intención de crear “una comunidad de desarrolladores que impulse el estado del arte en este campo”. [Cita al paper de airsim, ultimas 2 lineas antes de Related Work]

La reciente novedad de AirSim y su apuesta por la muy alta fidelidad de la simulación, además de la creciente presencia de trabajos realizados con esta plataforma y el prestigio de la propia compañía, han sido los factores impulsores de la elección de este entorno de simulación para realizar este trabajo de fin de grado.

Existen muchos más entornos de simulación, pero la mayoría están orientados a robots de tipo genérico (como ARGoS [WEB <http://www.argos-sim.info/index.php> ] o V-REP [WEB <http://www.coppeliarobotics.com/index.html> ]), por lo que no incorporan una simulación realista del comportamiento de un dron ni soportan compatibilidad con la mayoría de tecnologías necesarias, o bien no son de código abierto (como XPLANE [WEB <http://www.x-plane.com/> ] o RealFlight [WEB <https://www.realflight.com/> ]).

### TECNOLOGÍAS EMPLEADAS

* SIMULADOR: centrarse en el airsim, hablar de sus ventajas y desventajas, compatibilidades e incompatibilidades.
* AUTOPILOTO: explicar qué es y para que sirve, cuál se ha utilizado y por qué.
  + HITL vs SITL: hablar también de la existencia de dos modos: HITL y SITL, y de cuál se ha utilizado y por qué.
* ROS: explicar qué es y por qué se ha utilizado.
* RL: explicar por encima lo que es y en qué se basa, por qué es útil en nuestro contexto, y cómo funciona el entrenamiento (hacer ver la necesidad de una función de reset). Además comentar que se usa el gym, decir que es y por que es útil.

En este proyecto se han empleado diversas tecnologías, de las cuales se explicará un poco más a continuación:

#### ENTORNO DE SIMULACIÓN: AIRSIM

Introducir el airsim rapidito (ya se ha hablado de el justo encima) a partir del paper

https://github.com/Microsoft/AirSim/blob/master/docs/paper/main.pdf

AirSim es una plataforma modular con énfasis en la escalabilidad, dando la posibilidad de integrar nuevos tipos de vehículos, plataformas hardware y protocolos software. Proporciona soporte para MavLink, simulación SITL y HITL, integración con ROS y es compatible con PX4.

AirSim:

Es un software de simulación que emula un entorno real y vehiculos.

Tiene buena documentación, esta en continuo desarrollo, es crossplatform

Es una plataforma crossplatform win y Linux

Simula coche, cuadcopter y en el futuro mas

RC : manual drive

, Computer Vision mode (no hay vehiculo, solo cámaras)

, Programatically (vuelo autónomo)

Tiene apis para imagen, para control del dron y para mas cosas

Trae un ap, el simple flight que tiene A ventajas y B desventajas <https://github.com/Microsoft/AirSim/blob/master/docs/simple_flight.md> (comentarlo bien, que es importante)

En cuanto a compatibilidad, el AS es compatible con autopilots PX4 (hitl y sitl) en el futuro tb rosflight y hackflight

Tb es compatible con ROS (en desarrollo), y utiliza mavlink como prot de com

#### CONTROLADOR DE VUELO: PX4

REESCRIBIR PARTES, PULIR Y HOMOGENEIZAR TODO UN POCO: PARECE QUE ESTÁ A CACHOS INDEPENDIENTES

Completar todo lo escrito con lo que hay aquí:

<https://github.com/Microsoft/AirSim/blob/master/docs/flight_controller.md>

El controlador de vuelo o “autopilot” es uno de los componentes base más esenciales de cualquier vehículo multi-rotor. Se trata de una placa electrónica a la que se conectan los sensores (GPS, barómetros, giroscopios, acelerómetros, etc) y los actuadores (motores) del vehículo. Esta placa lleva instalado un firmware que se encarga de controlar el comportamiento de los actuadores en función de las lecturas de los sensores, con la finalidad de mantener la aeronave estable así como de cambiar la velocidad o la dirección de movimiento de la misma, de forma controlada. [ <https://www.tomshardware.com/reviews/multi-rotor-quadcopter-fpv,3828-2.html> ]

\*\*\*\*Pulir este cambio: meter un párrafo pequeño que comente que cuando se desarrolla en drones a menudo se utilizan entornos de simulación, y en ocasiones estos no integran un controlador de vuelo, por lo que tenemso que usar el autopilot en simulación. Para ello, existen dos métodos bla bla bla\*\*\*\*

Para integrar un controlador de vuelo con un entorno de simulación, existen dos métodos diferentes: HITL y SITL.

[ <https://wiki.paparazziuav.org/wiki/HITL> ]

[ <https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1109&context=mae_facpub> ]

HITL (Hardware In The Loop) es una técnica que se basa en conectar el hardware real del autopilot al equipo de desarrollo, de forma que el controlador de vuelo interactúe directamente con los sensores y actuadores del entorno de simulación.

SITL (Software In The Loop), en cambio se basa en emular el hardware ejecutando directamente el firmware del autopilot sobre el equipo de desarrollo, de forma que también interactúe con los sensores y actuadores del entorno de simulación.

El método SITL aporta una ventaja muy significativa, ya que elimina por completo la necesidad de adquirir una plataforma hardware. Sin embargo, cuando se trabaja con equipos de desarrollo con recursos limitados, el método HITL evita la sobrecarga de trabajo, permitiendo emplear toda la potencia disponible en el entorno de simulación, que normalmente es un software exigente.

{

Se utilizará el firmware *PX4 autopilot* [ <http://px4.io/> ] en modo SITL

PQ? PX4 es un autopilot opensource ampliamente utilizado perteneciente al proyecto *Dronecode* [ <https://www.dronecode.org/> ]. Es compatible con AirSim, asi como con el protocolo de comunicación Mavlink para comunicar los sensores y actuadores con el autopilot, y con ROS [Ref al anexo siguiente].

Se integrará con el método SITL, ya que se pretende obtener un freamwork de simulación que no dependa de ningún elemento hardware más allá del propio equipo de desarrollo.

}

Ademas el hecho de que el PX4 se puede construir en diferentes plataformas (raspi, pixhawk, qualcomm snapdragon flight, etc) [ <https://dev.px4.io/en/setup/dev_env.html> ] meter en gran cantidad de dispositivos hardware hace que nuestro código sea migrable directamente a un aparato real

Por que? Porque es compatible con airsim en versión sitl y hitl, y porque es un bicho utilizado tb en la realidad. Asique lo que implementemos aquí nos sirve para un dron real con esta misma placa.

Meter ref al código: <https://github.com/PX4/Firmware>

https://github.com/Microsoft/AirSim/blob/master/docs/flight\_controller.md

#### FRAMEWORK DE PROGRAMACIÓN: ROS

Ros es un middleware que bla bla bla

Sirve para bla bla bla

Nosotros lo hemos utilizado porque es versátil, nos permite trabajar en Python o C++ y tiene una extensión llamada mavros que nos permite hablar mavlink y comunciarnos con el autopilot y con el simulador.

#### REINFORCEMENT LEARNING

El RL es una rama del ML que se basa en bla bla bla (el tocho de la versión anterior)

Es útil para los vehiculso autónomos porque aprenden solitos a base de probar y probar

Hay un freamwork que es el gym de openai (comentar quien es openai y por que es relevante) que es muy útil para esto y es sencillo de utilizar. Además aporta una capacidad comparativa, ya que se pueden implementar diferentes algoritmos y probarlos de la misma forma, comparando como se comporta cada uno.