
Middleware y aplicaciones con robots aéreos autónomos

Máster Universitario en Robótica y Automatización
Campus de Leganés
Trabajo de Investigación Tutelado

Pedro Arias Pérez NIA 100421902
100421902@alumnos.uc3m.es

uc3m

Universidad
Carlos III
de Madrid

Índice

1. Introducción	3
1.1. Robótica Aérea	3
1.2. Motivación	5
1.3. Problema	5
1.4. Objetivos	6
2. Estado del Arte	7
2.1. Segmento Tierra	7
2.1.1. Estación de Control Terrestre	8
2.1.2. ToolKits de Vuelo	8
2.1.3. Otro software	8
2.2. Segmento Aire	9
2.2.1. Aeronaves Reales	9
2.2.2. Aeronaves Simuladas	11
2.2.3. Dispositivos Embebidos	12
2.3. Protocolo de Comunicaciones	12

Índice de figuras

1.	Clasificación de los UAV [1].	3
2.	Tipos de UAV.	4
3.	Drone de vigilancia en carretera de la Dirección General de Tráfico [2].	4
4.	QGroundControl.	7
5.	Ejemplos de drones de fabricantes propietarios.	9
6.	Drone PX4 real.	10
7.	Diferentes modelos de controladoras PixHawk.	11
8.	Drone Iris simulado mediante <i>Gazebo</i>	11
9.	NVIDIA Jetson Xabier.	12

Introducción

Este trabajo propone un estudio previo y diseño de un sistema para la programación y navegación de drones. Se persigue una estructura modular donde los diferentes bloques que constituyen el sistema puedan sustituirse para adaptarlos al problema, y que a su vez, permita la reutilización del código en diversas circunstancias.

Este primer capítulo recoge una introducción a la materia de estudio, la robótica aérea. Además, se expone la motivación cuyo resultado ha derivado en este estudio, junto al problema concreto al cual se le quiere dar solución.

Robótica Aérea

El ámbito de la robótica y de los vehículos aéreos no tripulados es un sector caracterizado por una fuerte expansión en los últimos tiempos, con unas expectativas de crecimiento y de demanda de personal cualificado tanto a nivel Europeo como nacional muy relevantes para los próximos años. En el año 2035 el volumen de negocio anual estimado será 10.000M€ y 90.000 puestos de trabajo (1.220M€ y 11.000 ud solo para España), pasando en el año 2050 a un volumen de 14.600M€ y 110.000 puestos de trabajo (1.520M€ y 11.500 ud caso español) [3].

La robótica aérea es la rama de la robótica que se encarga del estudio del comportamiento autónomo de aeronaves no tripuladas. Se entiende como una aeronave no tripulada (UAV, *Unmanned Aerial Vehicle*, o más recientemente UAS, *Unmanned Aircraft System*) a aquella que es capaz de realizar una misión sin necesidad de tener una tripulación embarcada [1]. Otro término que también se utiliza con frecuencia es VANT, Vehículo Aéreo No Tripulado.

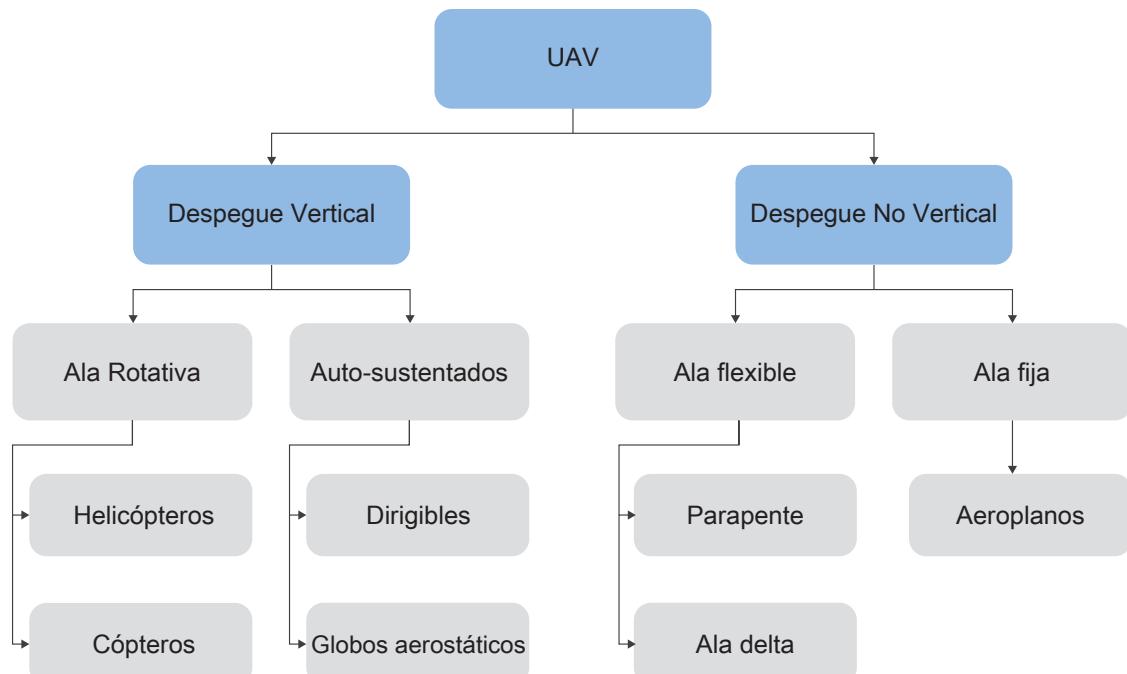


Figura 1: Clasificación de los UAV [1].

A la hora de establecer una clasificación de los UAV es posible atender a diferentes criterios. Siguiendo la clasificación propuesta por *Barrientos et al.* [1] se distinguen aeronaves en función del tipo de despegue, que puede ser vertical o no. A su vez, podemos subdividir las aeronaves en función del origen de su sustentación o del tipo de ala que poseen. Esta clasificación se representa en la Figura 1, mientras que en la Figura 2 se muestran ejemplos de los diferentes tipos de UAV.



(a) Aeroplano.

(b) Multicóptero.

Figura 2: Tipos de UAV.

Otros criterios de clasificación pueden responder a las capacidades de vuelo como el alcance, la altitud, la autonomía o la carga máxima. A su vez, también se pueden clasificar las aeronaves en función de la actividad que realizan. Algunas de estas clasificaciones distinguen entre uso civil y militar o aplicaciones en explotación (producto) o en desarrollo (prototipo).



Figura 3: Drone de vigilancia en carretera de la Dirección General de Tráfico [2].

En la actualidad tiende a utilizarse el concepto de UAS frente al de UAV. La extensión del concepto de vehículo a sistema refleja que el vehículo aéreo autónomo precisa para su funcionamiento de todo un sistema y no solo de la aeronave instrumentada. Típicamente, un sistema UAS se compone de el segmento aire, compuesto principalmente por la aeronave, y por el segmento tierra, compuesto por un computador donde se ejecuta algún software de control, habitualmente una estación terrestre. Entre ambos segmentos debe existir en todo momento una comunicación, que se realiza a través de un protocolo de comunicaciones. Sin embargo, embarcar en la aeronave el segmento tierra también es posible, y de hecho, muy común. La inmensa mayoría de los sistemas que hoy en día entendemos como *inteligentes* o *autónomos* llevan embarcado un ordenador con un software que les permite cerrar el bucle de control. Este software no deja de ser un tipo de estación terrestre que también puede ejecutarse en un segmento tierra no embarcado. Las características de la plataforma aérea y el problema a resolver definirán si es necesario embarcar o no el software de control sobre la aeronave.

Motivación

Este trabajo busca realizar un estudio previo y el diseño de un software de vuelo para la navegación autónoma de aeronaves, principalmente multicópteros.

La importancia del vuelo autónomo se ha visto reflejada en el alto incremento del uso de UAVs. Recientemente, los UAV se utilizan ampliamente tanto en aplicaciones militares como civiles debido a su pequeño tamaño y gran capacidad de maniobra [4]. Especialmente en aplicaciones civiles, los usos de los UAV se han expandido a casi todas las áreas. En el área de la agricultura, la rápida evolución de los UAV puede conducir a aplicaciones de agricultura de precisión, como la monitorización aérea de cultivos y las tareas de fumigación inteligente [5]. En el campo industrial, los desarrollos de los UAV mejoran la eficiencia de misiones como inspección industrial (e.g. plantas fotovoltaicas) [6], identificación de carga y entrega o logística, fuertemente ligadas a técnicas de *slam* visual (VSLAM). Además, los UAV también se pueden ver en tareas de búsqueda y rescate, de topografía o de vigilancia entre otras aplicaciones.

Dentro del ámbito nacional, el Ministerio de Ciencia e Innovación recoge en el documento Estrategia Española de Ciencia, Tecnología e Innovación 2021-2027 [7] las Líneas Estratégicas de I+D+I nacional. Entre ellas se encuentra Inteligencia Artificial y Robótica, en el ámbito de intervención de Mundo Digital, Industria, Espacio y Defensa, ratificando la relevancia que están cobrando los UAVs en los últimos años.

Problema

El problema a resolver es realmente complejo, alcanzar un software seguro y robusto que permita el vuelo autónomo de aeronaves. Es inabordable en tiempo y esfuerzo para este estudio. Por ello, este trabajo se centra en un subproblema del software. En concreto, se busca un software intermedio, a modo de *middleware*, que permita conectar distintas plataformas de vuelo con diferentes aplicaciones finales, entre las que se encuentran algoritmos de navegación autónoma.

Más allá del típico control por posición en exteriores, basado en GPS, el *middleware* que queremos investigar deberá permitir aplicaciones con control basado en posición en interiores (sin GPS, por ejemplo con algoritmos de auto-localización visual final, visual SLAM), como aplicaciones de control visual, no basado en posición. Estas aplicaciones pueden ser desarrolladas por usuarios ajenos al resto del software e incluso de la aeronave utilizadas. Además, el estudio se centrará solamente en el uso de un tipo de aeronaves, los multicópteros.

Objetivos

El objetivo del trabajo es el desarrollo de herramientas software que permitan resolver el problema anteriormente descrito. Además, se adhieren como objetivos secundarios una serie de comprobaciones que permitan comprobar el correcto funcionamiento del software. Entre estas, se encuentran por un lado, pruebas sobre diferentes aeronaves y, por otro lado, pruebas con diferentes aplicaciones ilustrativas de ejemplo que serán desarrolladas a propósito para las pruebas.

Así pues, los objetivos del proyecto son los siguientes:

- Desarrollo de herramienta software tipo *middleware*.
- Pruebas de la herramienta sobre distintas plataformas aéreas.
- Desarrollo de diferentes aplicaciones de control.

Para tratar de entender mejor los objetivos anteriores, en el siguiente capítulo se describirá el contexto asociado a este tipo de software.

Estado del Arte

A lo largo de este segundo capítulo se analizan las diferentes herramientas *software* y *hardware* utilizadas tanto en la comunidad científica como en la industria. Dado que la clasificación entre *software* y *hardware* puede ser algo confusa, se presentarán indistintamente distinguiendo entre herramientas del segmento tierra, del segmento aire (principalmente compuesto por la plataforma aérea) y el protocolo de comunicaciones.

Segmento Tierra

El lado tierra del sistema se compone principalmente por la emisora de vuelo y/o un ordenador. La presencia de la emisora depende en gran medida de la aeronave en cuestión, mientras que la presencia de un ordenador en el lado tierra depende de la aplicación. Ambos elementos son compatibles y el uso de uno no significa la no necesidad del otro.

Una emisora radiocontrol es un control remoto que permite controlar la aeronave. Habitualmente, la comunicación es mediante una antena, aunque existen modelos que se comunicaban con un cable (aeronaves en vuelo cautivo).

Un ordenador ofrece una amplia variedad de posibilidades de software, desde aplicaciones comerciales a aplicaciones libres o incluso propias. Debido a la alta cantidad de software disponible, esta sección se centrará en una serie de herramientas que resultan relevantes para los objetivos del proyecto.

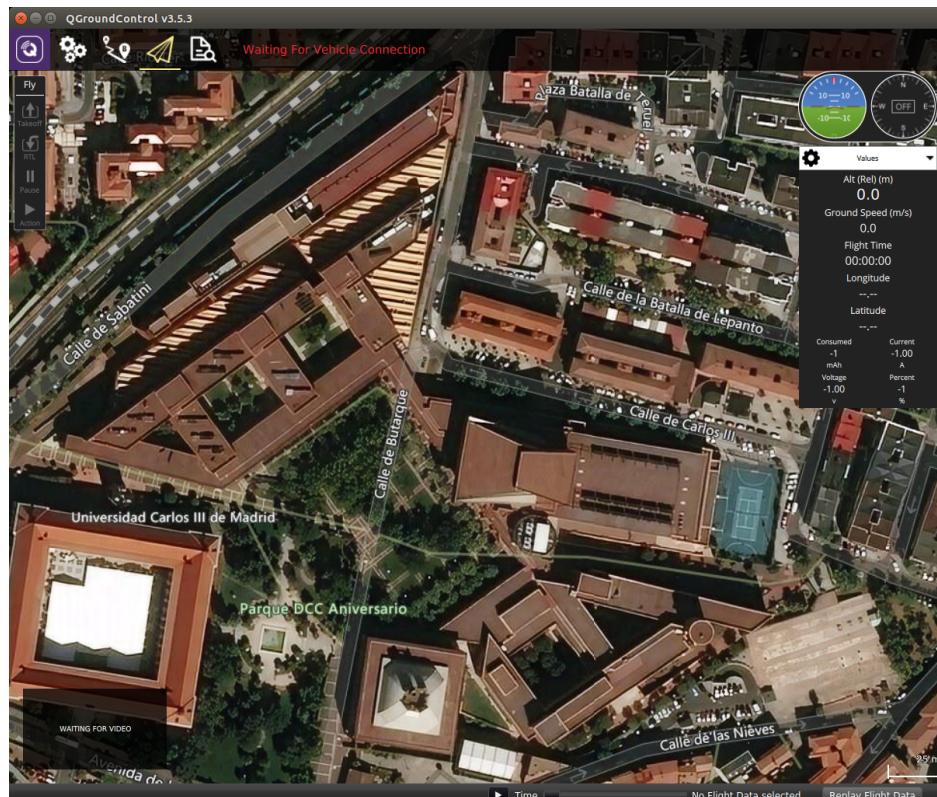


Figura 4: QGroundControl.

Estación de Control Terrestre

Las Estaciones de Control Terrestre (GCS) son herramientas software que se utilizan para planificar y volar una misión. Generalmente proporcionan una pantalla con un mapa donde el usuario puede definir puntos de referencia para el vuelo y ver el progreso de la misión. Además, suelen disponer de una "cabina virtual", mostrando muchos de los mismos instrumentos que los aviones tripulados.

Hoy en día, las principales GCS o bien son software propietario de los principales fabricantes de UAVs o bien son software libre. Debido a su naturaleza, resultan más interesantes estas segundas. Entre ellas destacan principalmente *QGroundControl* [8] y *MissionPlanner* [9]. La Figura 4 muestra este software en ejecución.

ToolKits de Vuelo

Los ToolKits de Vuelo son aplicaciones de control de vuelo programables. Ofrecen una interfaz de programación (API) que permite el desarrollo de aplicaciones a un nivel de abstracción muy elevado. Este tipo de software permite abstraerse de aspectos como el tipo concreto de aeronave o la comunicación con la misma. Cabe destacar que este tipo de vuelo puede ir a bordo de la aeronave y formar parte del segmento aire o pertenecer al segmento tierra y ejecutarse en un ordenador externo. Debido al alto número de software de este tipo, se han seleccionado dos que tienen gran interés para este estudio. Ambos han sido desarrolladas por universidades madrileñas o organismos con vinculación a las mismas.

La primera de estas, *Aerostack*, ofrece a desarrolladores herramientas para diseñar y construir la arquitectura de control de sistemas robóticos aéreos, integrando múltiples soluciones computacionales heterogéneas como algoritmos de visión por ordenador, métodos de mapeo y auto-localización, planificadores de movimiento, etc. [10]. Ha sido desarrollada por el grupo de Visión por Computador y Robótica Aérea de la Universidad Politécnica de Madrid y actualmente se encuentra en su versión 4.

En segundo lugar, *Drone Wrapper* [11], ha sido desarrollada por la asociación de robótica *JdeRobot* [12] ligada a la Universidad Rey Juan Carlos, y entre sus desarrolladores se encuentra el autor de este proyecto. Ofrece *drivers* para diferentes aeronaves, herramientas de visualización y teleoperación, y una API que permite abstraerse del sistema y desarrollar aplicaciones sobre la misma.

Otro software

En este apartado se quiere destacar herramientas muy utilizadas para el desarrollo de aplicaciones y ampliamente utilizadas en el ámbito de la robótica aérea. Al igual que sucedía con los ToolKits de Vuelo, el software descrito en esta sección puede pertenecer tanto al segmento aire como al segmento tierra en función de donde este ubicado.

En entornos donde se requiere de interacción software-hardware es necesaria el uso de un *middleware*. El más utilizado en robótica es ROS [13]. ROS (*Robot Operating System*) es "un meta-sistema operativo de código abierto para su robot", mantenido por *Open Source Robotics Foundation* (OSRF) [14]. ROS proporciona un entorno de nodos distribuido y fácilmente escalable. Estos nodos son programas que se ejecutan de forma independiente en la computadora (o se distribuyen en una red), por lo que pueden realizar tareas individuales. Sin embargo, pueden comunicarse entre ellos de forma síncrona o de forma asíncrona a través de mensajes.

Dentro de la robótica aérea, existe una colección de nodos de comunicación extendible MAVLink (ver Secc. 2.3) para ROS conocida como MAVROS (*Micro Air Vehicles ROS*) [15]. Este paquete de ROS proporciona un controlador de comunicación para varios autopilotos con protocolo de comunicación MAVLink.

Para el procesamiento de imágenes, OpenCV (Open Source Computer Vision) [16] es una biblioteca de código abierto C++/Python/Java (escrita de forma nativa en C++) utilizada en Visión por Computador. Entre los métodos clásicos y de última generación que incluye, se pueden encontrar varias funciones adecuadas para reconocimiento facial, seguimiento ocular, establecimiento de marcadores para realidad aumentada, etc. Debido al excelente rendimiento logrado por esta biblioteca se ha convertido en el estándar *de facto* para todo tipo de usuarios.

En el ámbito del aprendizaje profundo, existen diferentes marcos de programación que permiten el desarrollo de redes y algoritmos. Uno de ellos es TensorFlow [17]. Se trata de una biblioteca de cálculo numérico de alto rendimiento, muy centrada en la computación paralela, normalmente realizada por GPU o *clústeres* de procesamiento. Posee una alta eficiencia lo que permite entrenar y ejecutar un modelo neuronal (o cargar un modelo previamente entrenado) de forma rápida y sencilla.

Segmento Aire

El lado aéreo se compone por la plataforma aérea, es decir el drone junto a todos los elementos a bordo que componen el sistema. En este estudio se presentan dos tipos de aeronaves, reales o simuladas. A mayores, la aeronave puede llevar embarcado sistemas embebidos que permitan la ejecución de algoritmos computacionalmente costosos en tiempo real.

Aeronaves Reales

Las aeronaves reales pueden poseer un firmware propietario o libre, en función de si la controladora y el software de vuelo es abierto al usuario o no.



(a) 3DR Solo Drone.



(b) DJI Tello.

Figura 5: Ejemplos de drones de fabricantes propietarios.

Por un lado, entre los principales fabricantes propietarios se encuentran *DJI* [18], *Parrot* [19] o *3DR Robotics* [20] (ver Fig. 5). Aeronaves como el Tello de DJI (Fig. 5b) son ampliamente utilizadas en investigación debido a su bajo coste y tamaño. Ejemplo de ello pueden ser el estudio realizado por A. Anwar y A. Raychowdhury donde proponen un nuevo algoritmo de navegación autónoma basado en técnicas de aprendizaje profundo [21] o A. Scannell et al. en su investigación sobre nuevos algoritmos de optimización de trayectorias en entornos complejos [22].

Por el otro lado, a la hora de hablar de plataformas libres es necesario distinguir entre los componentes hardware (principalmente la controladora) y el software de vuelo (firmware). Los dos principales firmware libre son *PX4* y *Ardupilot*.

PX4 es un software de control de vuelo de código abierto para drones y otros vehículos no tripulados. Proporciona un conjunto flexible de herramientas para que los desarrolladores compartan tecnologías que consigan crear soluciones personalizadas para aplicaciones de drones [23]. *PX4* pertenece Dronecode, una organización sin ánimo de lucro de la Fundación Linux.

ArduPilot es un sistema de piloto automático confiable, versátil y de código abierto que admite muchos tipos de vehículos: multicópteros, helicópteros, aviones de ala fija, botes, submarinos, rovers y más [24]. El código fuente está desarrollado por la comunidad abierta de desarrolladores *Ardupilot-Dev Team*.

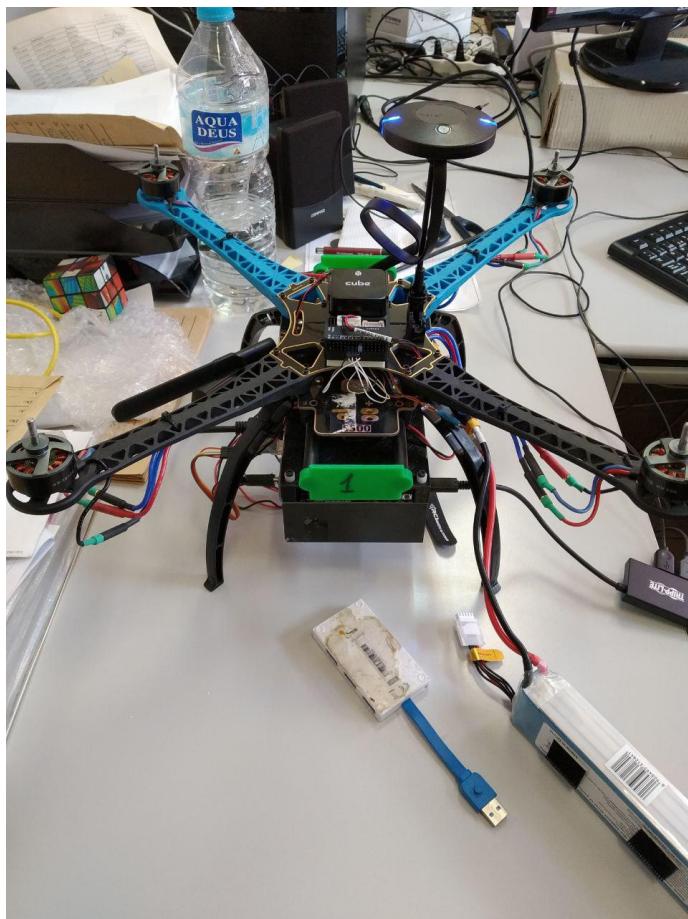


Figura 6: Drone PX4 real.

La principal controladora de vuelo utilizada con ambos firmware es *Pixhawk* [25]. Al igual que *PX4*, *Pixhawk* pertenece al grupo Dronecode. La Figura 6 muestra un multicóptero equipado con *PX4*, mientras que la Figura 7 contiene diferentes controladoras *Pixhawk*.



(a) PixHawk v1

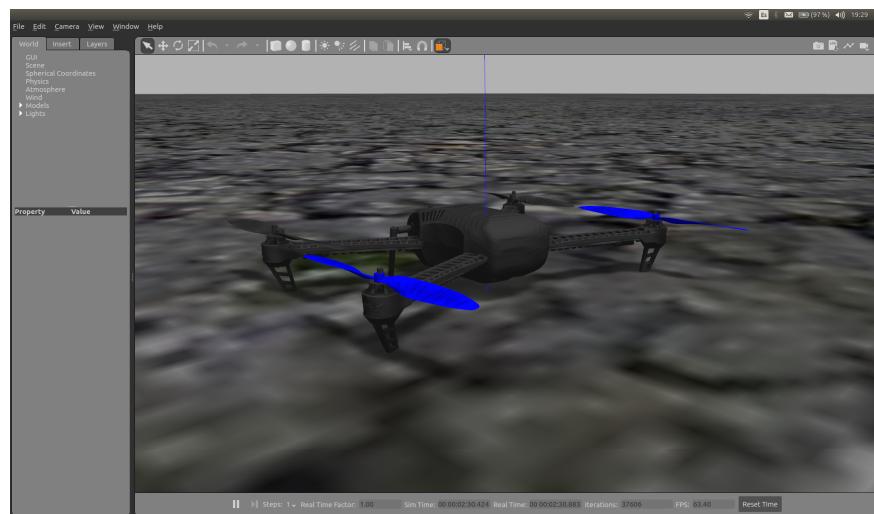


(b) PixHawk v2.1

Figura 7: Diferentes modelos de controladoras PixHawk.

Aeronaves Simuladas

Las aeronaves simuladas suelen utilizar software de vuelo libre, dada la naturaleza abierta del mismo. En una simulación, la parte física de la aeronave se ve sustituida por software (*SITL*, *Software-In-The-Loop*). El *SITL* contiene el software de vuelo y permite enviar y recibir comandos de vuelo sin necesidad de tener una controladora real, haciendo de intermediario entre el autopiloto y el modelo *SDL*, necesario en la visualización gráfica. Dichas simulaciones se suelen acompañar de una visualización. Para ello, es necesario el uso de un simulador como *Gazebo* [26], *AirSim* [27] o *jMAVSim* [28]. El más extendido, debido a sus grandes prestaciones y posibilidades de configuración es *Gazebo*. La Figura 8 muestra un drone Iris (*3DR Robotics*) simulado mediante *Gazebo* con *PX4* como autopiloto.

Figura 8: Drone Iris simulado mediante *Gazebo*.

Dispositivos Embebidos

En la Sección anterior (Sec. 2.1.3) se ha mencionado el interés de embarcar parte del software en la aeronave. Ejecutar ciertos procesos en la aeronave ofrece un salto en rendimiento en algoritmos en tiempo real, muy presentes en la visión por computador. Además, al tratarse de vehículos aéreos no es posible correr estos algoritmos en ordenadores portátiles como si se hace tradicionalmente en vehículos terrestres.

Por ello, debido al mencionado aumento del interés por aplicaciones de visión en tiempo real ha propiciado el desarrollo de dispositivos embebidos de baja potencia para su integración en sistemas robóticos móviles. Ejemplo de ello son los dispositivos como Arduino o Raspberry Pi, utilizados en robots como PiBot [29].

Sin embargo, para ejecutar algoritmos complejos, como redes neuronales, no es suficiente con los dispositivos mencionados y es necesario utilizar dispositivos específicos. Una alternativa viable son los dispositivos *Jetson* fabricados por *NVIDIA* [30]. Cada NVIDIA Jetson es un sistema en módulo (SOM) completo que incluye CPU, GPU, memoria, administración de energía, interfaces de alta velocidad y más, que permiten ejecutar CUDA [31], una biblioteca de computación paralela de bajo nivel, así como varios kits de herramientas (como JetPack SDK [32]) diseñados para optimizar los procesos que se ejecuten en el dispositivo.

El tamaño y consumo de estos dispositivos hacen que sea un sistema ideal para embarcar en vehículos aéreos.



Figura 9: NVIDIA Jetson Xabier.

Protocolo de Comunicaciones

El protocolo de comunicaciones puede ser privativo, utilizado por fabricantes de aeronaves y diferentes para cada una de ellas, o libre. El protocolo de comunicaciones libre por excelencia es MAVLink [33], que se ha convertido *de facto* en el estándar del sector.

MAVLink son las siglas de *Micro Air Vehicle Link*, un protocolo de comunicaciones muy ligero para el intercambio de mensajes con un dron y sus componentes a bordo. MAVLink se sitúa entre los extremos de la comunicación, siendo el puente de comunicación entre el lado tierra y el lado aire. Es considerado como el protocolo estándar de comunicaciones en robótica aérea y multitud soluciones comerciales lo utilizan, por ejemplo las ya mencionadas GCS, *QGroundControl* y *Mission Planner*.

Se compone por cuatro tipo de elementos: mensajes, comandos, enumerados y microservicios. Los mensajes sirven para transmitir el estado o información de la aeronave, principalmente datos de navegación o de posición. Ejemplo de ello son los mensajes *HEARTBEAT* (ver Cód. 1) o *ATTITUDE*.

```

1 <message id="0" name="HEARTBEAT">
2   <description>The heartbeat message shows that a system or component is present and responding.
3   The type and autopilot fields (along with the message component id), allow the receiving
4   system to treat further messages from this system appropriately.</description>
5   <field type="uint8_t" name="type" enum="MAV_TYPE">Vehicle or component type. For a flight
6   controller component the vehicle type (quadrotor, helicopter, etc.). For other
7   components the component type (e.g. camera, gimbal, etc.).</field>
8   <field type="uint8_t" name="autopilot" enum="MAV_AUTOPILOT">Autopilot type / class. Use
9     MAV_AUTOPILOT_INVALID for components that are not flight controllers.</field>
10  <field type="uint8_t" name="base_mode" enum="MAV_MODE_FLAG" display="bitmask">System mode
11    bitmap.</field>
12  <field type="uint32_t" name="custom_mode">A bitfield for autopilot-specific flags</field>
13  <field type="uint8_t" name="system_status" enum="MAV_STATE">System status flag.</field>
14  <field type="uint8_t_mavlink_version" name="mavlink_version">MAVLink version, not writable by
15    user, gets added by protocol because of magic data type: t_mavlink_version</field>
16 </message>
```

Código 1: Definición HEARTBEAT, mensaje de MAVLink.

Los comandos sirven para transmitir peticiones y acciones a la aeronave desde la GCS, o viceversa. El Código 2 muestra la estructura de un comando de navegación a un punto.

Los enumerados describen diferentes estados, como puede ser el modo de vuelo, el tipo de aeronave o el estado de la comunicación, y son usados como campos de los mensajes o de los comandos. En el mensaje HEARTBEAT (Cód. 1) se hace referencia a varios, como *MAV_TYPE* o *MAV_AUTOPILOT*.

Por último, los microservicios establecen cómo deben ser intercambiados los mensajes y comandos para un correcto funcionamiento de la comunicación. Los microservicios se utilizan para intercambiar muchos tipos de datos, incluidos: parámetros, misiones, trayectorias, imágenes y otros archivos.

Mensajes	
HEARTBEAT	VFR_HUD
SYS_STATUS	SET_MODE
HIGHRES_IMU	RADIO_STATUS
BATTERY_STATUS	ATTITUDE
	GLOBAL_POSITION_INT
	SET_POSITION_TARGET_LOCAL_NED

Tabla 1: Principales mensajes de MAVLink.

Comandos como el presentado (Cód. 2) son útiles a la hora navegar de forma tradicional mediante posición GPS. Para un control por posición visual o sin GPS (p.e. en interiores) es necesario utilizar otro tipo de mensajes. Una posible alternativa es el mensaje *SET_POSITION_TARGET_LOCAL_NED* que permite establecer la posición, velocidad, aceleración o fuerza deseada del vehículo en un marco de coordenadas local NED (*North East Down*).

Junto con este mensaje, otros mensajes y comandos son necesarios para un correcto funcionamiento del protocolo de comunicaciones. La tabla 1 recoge los más relevantes.

```

1 <enum name="MAV_CMD">
2   <description>Commands to be executed by the MAV. They can be executed on user request, or as
3       part of a mission script. This command is similar what ARINC424 is for commercial
4       aircraft: A data format how to interpret waypoint/missiondata.</description>
5   <entry value="16" name="MAV_CMD_NAV_WAYPOINT">
6     <description>Navigate to waypoint.</description>
7     <param index="1">Hold time in decimal seconds. (ignored by fixed wing, time to stay at
8         waypoint for rotary wing)</param>
9     <param index="2">Acceptance radius in meters (if the sphere with this radius is hit, the
10        waypoint counts as reached)</param>
11    <param index="3">0 to pass through the WP, if gt; 0 radius in meters to pass by WP. Positive
12        value for clockwise orbit, negative value for counter-clockwise orbit. Allows
13        trajectory control.</param>
14    <param index="4">Desired yaw angle at waypoint (rotary wing). NaN for unchanged.</param>
15    <param index="5">Latitude</param>
16    <param index="6">Longitude</param>
17    <param index="7">Altitude</param>
18  </entry>
19  ...
20 </enum>
```

Código 2: Definición CMD_NAV_WAYPOINT, comando de MAVLink.

Referencias

- [1] A. Barrientos, J. Del Cerro, P. Gutiérrez, R. San Martín, A. Martínez, and C. Rossi, “Vehículos aéreos no tripulados para uso civil. tecnología y aplicaciones,” *Universidad politécnica de Madrid, Madrid*, 2007.
- [2] D. G. de Tráfico, “Tráfico distribuye los 39 drones que vigilarán las carreteras españolas este verano.” https://www.dgt.es/es/prensa/notas-de-prensa/2021/Trafico_distribuye_los_39_drones_que_vigilaran_las_carreteras_espanolas_este_verano.shtml. Último acceso: 12-07-2021.
- [3] M. de Fomento, “Plan estratégico para el desarrollo del sector civil de los drones en España 2018-2021.” <https://www.fomento.gob.es/NR/rdonlyres/7B974E30-2BD2-46E5-BEE5-26E00851A455/148411/PlanEstrategicoDrones.pdf>. Último acceso: 07-07-2021.
- [4] J. Kim, S. Kim, C. Ju, and H. I. Son, “Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications,” *IEEE Access*, vol. 7, pp. 105100–105115, 2019.
- [5] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, “A compilation of UAV applications for precision agriculture,” *Computer Networks*, vol. 172, p. 107148, 2020.
- [6] H. Yao, R. Qin, and X. Chen, “Unmanned aerial vehicle for remote sensing applications—a review,” *Remote Sensing*, vol. 11, no. 12, p. 1443, 2019.
- [7] M. de Ciencia e Innovación, “Estrategia española de ciencia, tecnología e innovación 2021-2027.” <https://www.ciencia.gob.es/stfls/MICINN/Ministerio/FICHEROS/EECTI-2021-2027.pdf>. Último acceso: 07-07-2021.
- [8] DronecodeProject, “Qgroundcontrol.” <http://qgroundcontrol.com/>. Último acceso: 12-07-2021.
- [9] ArduPilot-Dev-Team, “Mission planner.” <https://ardupilot.org/planner/>. Último acceso: 12-07-2021.
- [10] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, “Aerostack: An architecture and open-source software framework for aerial robotics,” in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 332–341, IEEE, 2016.
- [11] JdeRobot, “Drone wrapper.” <https://github.com/JdeRobot/drones/tree/noetic-devel>. Último acceso: 13-07-2021.
- [12] JdeRobot, “Jderobot.” <https://jderobot.github.io/>. Último acceso: 13-07-2021.
- [13] I. Open Source Robotics Foundation, “Ros.” <https://www.ros.org/>. Último acceso: 13-07-2021.
- [14] I. Open Source Robotics Foundation, “Osrf.” <https://www.openrobotics.org/>. Último acceso: 13-07-2021.
- [15] I. Open Source Robotics Foundation, “Mavros.” <http://wiki.ros.org/mavros>. Último acceso: 13-07-2021.
- [16] O. Team, “Opencv.” <https://opencv.org/>. Último acceso: 19-07-2021.
- [17] Google, “Tensorflow.” <https://www.tensorflow.org/>. Último acceso: 13-07-2021.
- [18] DJI, “Dji drones.” <https://www.dji.com/es>. Último acceso: 12-07-2021.
- [19] P. D. SAS, “Parrot drones.” <https://www.parrot.com/es/drones>. Último acceso: 12-07-2021.
- [20] D. Robotics, “3dr drone.” <https://www.3dr.com/>. Último acceso: 12-07-2021.
- [21] A. Anwar and A. Raychowdhury, “Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning,” *IEEE Access*, vol. 8, pp. 26549–26560, 2020.

- [22] A. Scannell, C. H. Ek, and A. Richards, “Trajectory optimisation in learned multimodal dynamical systems via latent-ode collocation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [23] I. Dronecode Project, “Px4.” <https://px4.io/>. Último acceso: 11-02-2020.
- [24] A. D. Team and Community, “Ardupilot.” <https://ardupilot.org/>. Último acceso: 11-02-2020.
- [25] A. Ltd., “Pixhawk.” <https://pixhawk.org/>. Último acceso: 07-03-2020.
- [26] O. S. R. Foundation, “Gazebo.” <http://gazebosim.org/>. Último acceso: 11-02-2020.
- [27] M. Research, “Airsim.” <https://microsoft.github.io/AirSim/>. Último acceso: 13-07-2021.
- [28] I. Dronecode Project, “jmavsim.” <https://github.com/PX4/jMAVSIM>. Último acceso: 13-07-2021.
- [29] J. Vega and J. M. Cañas, “Pibot: An open low-cost robotic platform with camera for stem education,” *Electronics*, vol. 7, no. 12, p. 430, 2018.
- [30] N. Corporation, “Nvidia jetson.” <https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/>. Último acceso: 20-07-2021.
- [31] N. Corporation, “Nvidia cuda.” <https://developer.nvidia.com/cuda-zone>. Último acceso: 20-07-2021.
- [32] N. Corporation, “Nvidia jetpack sdk.” <https://developer.nvidia.com/embedded/jetpack>. Último acceso: 20-07-2021.
- [33] DronecodeProject, “Mavlink.” <https://mavlink.io/en/>. Último acceso: 12-07-2021.