
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Grado en Ingeniería en Tecnologías Industriales

# Diseño Integral de Hardware para un Cuadricóptero Autónomo: Sensores, Actuadores y Comunicaciones

Autor
Clara Lucena Vicente

Dirigido por
Juan Luis Zamora Macho

Madrid
Junio 2025

**Clara Lucena Vicente**, declara bajo su responsabilidad, que el Proyecto con título **Diseño Integral de Hardware para un Cuadricóptero Autónomo: Sensores, Actuadores y Comunicaciones** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: ..... Fecha: 30 / 06 / 2025

Autoriza la entrega:

El Director del Proyecto

**Juan Luis Zamora Macho**

Fdo.: ..... Fecha: 01 / 07 / 2025


ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Grado en Ingeniería en Tecnologías Industriales

# Diseño Integral de Hardware para un Cuadricóptero Autónomo: Sensores, Actuadores y Comunicaciones

Autor
Clara Lucena Vicente

Dirigido por
Juan Luis Zamora Macho

Madrid
Junio 2025

# Diseño Integral de Hardware para un Cuadricóptero Autónomo: Sensores, Actuadores y Comunicaciones

**Autor:** Clara Lucena Vicente
**Director:** Zamora Macho,Juan Luis
**Entidad colaboradora:** Universidad Pontificia Comillas, ICAI

## Resumen

Este Trabajo de Fin de Grado es una continuación de trabajos anteriores ([1],[2], [3]). A diferencia de proyectos de otros años, donde se utilizó una Raspberry para el desarrollo del UAV [4], este año el proyecto se ha apoyado en el Cube Orange+ de la familia de CubePilot (anteriormente conocida como Pixhawk). Se han utilizado programas como Matlab/Simulink (2024b) y QGroundControl. Entre los elementos del hardware utilizados, se encuentran el sensor de altura TFmini-S y el ESP32-C3, utilizado para la comunicación Wi-Fi entre el UAV y la estación de control.

## Introducción

Los UAVs han causado una revolución en los últimos años, debido a las múltiples aplicaciones que tienen y pueden llegar a tener, y gracias a los avances tecnológicos. Las mejoras en los sensores, algoritmos e incluso en el desarrollo de la inteligencia artificial, están incrementando el abanico de posibilidades que pueden llegar a tener estos dispositivos, empleándose en sectores centrados en la industria, la cinematografía o incluso teniendo aplicaciones militares y científicas [5]. La respuesta a los vuelos autónomos de interiores se obtiene con las aplicaciones técnicas como es la SLAM, la cual requiere de sensores integrados en el UAV, o con sistemas como el MCS, el cual utiliza sensores externos para la monitorización del dron.

## Objetivos

El objetivo de este proyecto está separado en 2 partes: en primer lugar, conseguir que el dron realice exitosamente vuelos manuales, para posteriormente lograr que vuele de manera autónoma en interiores. Para ello, se ha dividido el proyecto en varias fases:

- Calibrado e integración de los sensores y actuadores necesarios ([6], [7]).
- Implementación de las comunicaciones necesarias para el vuelo manual y autónomo ([8], [9]).
- Colaborar en el desarrollo e implementación del control para el vuelo manual.
- Integrar el hardware adicional necesario para el vuelo autónomo ([10]).
- Colaborar en el desarrollo del control para el vuelo autónomo.

# Conclusiones

En este proyecto se han alcanzado varios objetivos:

- Se ha realizado la calibración e integración de todos los sensores, actuadores necesarios para el vuelo manual y autónomo.
- Se han incorporado las comunicaciones necesarias para poder recibir mensajes de forma inalámbrica sobre el estado del dron.
- Se han completado las configuraciones necesarias para el correcto funcionamiento del Cube Orange+.
- Se han realizado grandes avances en el apartado del control del UAV.

# Referencias más relevantes


1 J. G. Aguilar, «Desarrollo de sistemas de navegación autónoma en interiores para un UAV,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, Madrid, 2018. dirección: `https://repositorio.comillas.edu/xmlui/handle/11531/14540`.

2 N. G. García, «Control de un cuadricóptero para navegación en interiores usando un sensor de flujo óptico,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, Madrid, 2016. dirección: `https://repositorio.comillas.edu/xmlui/handle/11531/22629`.

3 J. L. Castillo, «Sistema de navegación autónoma para un cuadricóptero en exteriores,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia de Comillas, 2022. dirección: `https://repositorio.comillas.edu/jspui/bitstream/11531/22632/1/TFG-LoringCastilloJaime.pdf`.

4 J. J. B. Vázquez, «Control de navegación autónoma de un cuadricóptero en interiores,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, 2019. dirección: `https://repositorio.comillas.edu/xmlui/handle/11531/32609`.

5 FAS, *UAVs in agriculture*, Última consulta el 1 de junio de 2025. dirección: `https://www.fas.scot/environment/uavs-in-agriculture/#:~:text=Current%20uses%20for%20drones%20in,health%20monitoring%20and%20disease%20detection`.

6 CubePilot, *The Cube Module Overview*, Última consulta el 7 de julio de 2025. dirección: https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview.

7 Benewake, *TFmini-S Datasheet*, Última consulta el 3 de marzo de 2025. dirección: `https://en.benewake.com/DataDownload/index_pid_20_lcid_22.html`.

8 *Emisora FS-i6S*, Última consulta el 31 de enero de 2025. dirección: `https://rcinnovations.es/shop/fls00003129-flysky-fs-i6s-10ch-rx-ia6b-15631#attr=10534`.

9 *Espressif*, ESP32-C3esp-dev-kits Documentation, Última consulta el 16 de julio de 2025. dirección: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c3/esp32-c3-devkitm-1/user_guide.html.

10 *Cámara Optitrack Flex13*, Última consulta el 31 de enero de 2025. dirección: https://www.optitrack-shop.de/flex-serie/kameras/125/flex-13.

# Integrated Hardware Design for an autonomous quadcopter: Sensors, Actuators and Communications

**Author:** Clara Lucena Vicente
**Supervisor:** Zamora Macho,Juan Luis
**Collaborating Entity:** Universidad Pontificia Comillas, ICAI

## Abstract

This project continues previous works ([1],[2], [3]). Unlike earlier projects that relied on a Raspberry Pi for UAV development [4], this year's project is based on a Cube Orange+ flight controller, from the CubePilots family (formerly called Pixhawk). Development tools such as Matlab/Simulink (R2024b) and QGroundControl have been used. Among the hardware components used, we can find the TFmini-S height sensor and the ESP32-C3, which provides Wi-Fi communication between the UAV and the ground control station.

## Introduction

Unmanned Aerial Vehicles (UAVs) have undergone a revolution in recent years thanks to rapid technological progress and their versatility. The improvement in sensors, algorithms, and AI techniques is increasing the opportunities that these devices can have. They can be used in industrial sectors, in cinematography, or they can even be used for military and scientific purposes [5]. The challenge of autonomous indoor flight is answered with both technical applications, such as SLAM, which relies on integrated sensors in the UAV, and with systems such as MCS, which use external sensors to monitor the drone.

## Objectives

The objective of this project is divided into two parts: complete a successful manual flight with the drone and be able to develop a successful autonomous flight. To reach these goals, the project has been divided into the following tasks:

- Calibrate and integrate all required sensors and actuators ([6], [7]).
- Implement the communication links required for both manual and autonomous flight ([8], [9]).
- Contribute to the development and implementation of the manual flight controller.
- Integrate the additional hardware needed for autonomous flight [10].
- Contribute to the development of the autonomous flight controller.

## Conclusions

In this project, several objectives have been met:

- The calibration and implementation of all the sensors and actuators needed for the manual and autonomous flight has been successfully completed.
- Successful implementation of wireless telemetry, enabling real-time status reporting from the drone.
- Configuration of the Cube Orange+ for correct operation.
- Significant progress has been made in developing and validating the UAV control algorithms.

# Most relevant references


1 J. G. Aguilar, «Development of Indoor Autonomous Navigation Systems for a UAV,» Bachelor's Thesis, ICAI School of Engineering, Comillas Pontifical University, Madrid, 2018. Available at: https://repositorio.comillas.edu/xmlui/handle/11531/14540.

2 N. G. García, «Control of a Quadcopter for Indoor Navigation Using an Optical Flow Sensor,» Bachelor's Thesis, ICAI School of Engineering, Comillas Pontifical University, Madrid, 2016. Available at: https://repositorio.comillas.edu/xmlui/handle/11531/22629.

3 J. L. Castillo, «Autonomous Navigation System for a Quadcopter in Outdoor Environments,» Bachelor's Thesis, ICAI School of Engineering, Comillas Pontifical University, Madrid, 2022. Available at: https://repositorio.comillas.edu/jspui/bitstream/11531/22632/1/TFG-LoringCastilloJaime.pdf.

4 J. J. B. Vázquez, «Autonomous Navigation Control of a Quadcopter in Indoor Environments,» Bachelor's Thesis, ICAI School of Engineering, Comillas Pontifical University, Madrid, 2019. Available at: https://repositorio.comillas.edu/xmlui/handle/11531/32609.

5 FAS, *UAVs in agriculture*, Accessed 1 June 2025. Available at: https://www.fas.scot/environment/uavs-in-agriculture/#:~:text=Current%20uses%20for%20drones%20in,health%20monitoring%20and%20disease%20detection.

6 CubePilot, *The Cube Module Overview*, Accessed 7 July 2025. Available at: https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview.

7 Benewake, *TFmini-S Datasheet*, Accessed 3 March 2025. Available at: https://en.benewake.com/DataDownload/index_pid_20_lcid_22.html.

8 *FS-i6S Controller*, Accessed 31 January 2025. Available at: https://rcinnovations.es/shop/fls00003129-flysky-fs-i6s-10ch-rx-ia6b-15631#attr=10534.

9 *Espressif*, ESP32-C3esp-dev-kits Documentation, Accessed 16 July 2025. Available at: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c3/esp32-c3-devkitm-1/user_guide.html.

10 *Optitrack Flex13 Camera*, Accessed 31 January 2025. Available at: https://www.optitrack-shop.de/flex-serie/kameras/125/flex-13.

# Agradecimientos

A mi director del TFG, Juan Luis Zamora Macho, por haberme ofrecido esta increíble oportunidad, y por haberme ayudado a lo largo del proyecto.

A mi compañero Álvaro Maestroarena Navamuel, con quien he tenido la suerte de trabajar conjuntamente en este proyecto. Gracias por tu esfuerzo, dedicación y paciencia.

A mi familia, por su apoyo continuado durante el transcurso de este proyecto y también a lo largo de la carrera. Gracias por haberme dado esta oportunidad.

# Índice general

# Índice de figuras

# Índice de tablas

# Códigos

# Acrónimos

| | |
|---|---|
| *BLDC* | Brushless Direct Current |
| *ESC* | Electronic Speed Controller |
| *GCS* | Ground Control Station |
| *I2C* | Inter-Integrated Circuit |
| *IMU* | Inertial Measurement Unit |
| *LiDAR* | Light Detection and Ranging |
| *MAVLink* | Micro Air Vehicle Link |
| *uORB* | Micro Object Request Broker |
| *MCS* | Motion Capture System |
| *PDB* | Power Distribution Board |
| *PWM* | Pulse Width Modulation |
| *QGC* | QGroundControl |
| *RC* | Remote Control |
| *TFG* | Trabajo Fin de Grado |
| *UAV* | Unmanned Aerial Vehicle |
| *UDP* | User Datagram Protocol |
| *VSCode* | Visual Studio Code |
| *SLAM* | Simultaneous Localization and Mapping |

# Capítulo 1

# Introducción

Los vehículos aéreos no tripulados (UAVs) han causado una revolución en los últimos años, debido a las múltiples aplicaciones que tienen y pueden llegar a tener, y gracias a los avances tecnológicos. El progreso reciente en la miniaturización de los distintos componentes, como es el caso de los microcontroladores, las mejoras en los sensores, algoritmos, e incluso el desarrollo de la inteligencia artificial, están incrementando el abanico de posibilidades que pueden llegar a tener estos dispositivos. Los UAVs tienen aplicaciones tanto militares como civiles y comerciales, siendo aprovechados tanto en el sector cinematográfico, cartográfico y agrario [11]. Actualmente, han llegado a utilizarse en la vigilancia, la entrega o transporte de paquetes, inspecciones e incluso salvamento. Los UAVs de interiores se utilizan principalmente en entornos industriales, en el inventariado en almacenes, o para investigación, aunque también podrían servir para asistencia médica [12].

Su versatilidad no hace más que aumentar, si además se contempla la integración de un sistema autónomo para su vuelo, mediante la incorporación de sistemas como Simultaneous Localization and Mapping (SLAM) o Motion Capture System (MCS) para lograrlo.

Este proyecto ha consistido en el diseño y desarrollo de un cuadricóptero de interiores, contando con la implementación de un control tanto para vuelo manual como autónomo. El proyecto es continuación de otros trabajos realizados anteriormente, tomando principalmente como base el trabajo realizado por Jorge Bennasar Vázquez (véanse [4], [1], [2], [3]). Además, se ha trabajado conjuntamente con otros estudiantes, uno de los cuales ha trabajado en el desarrollo del control del UAV, mientras que otro ha diseñado la estructura del dron.

Para el proyecto se ha utilizado como microcontrolador el Cube Orange +, y el software principal ha sido Matlab/Simulink (R2024b). Se ha empleado un sistema MCS (Motion Capture System) para el vuelo autónomo, empleando cámaras Flex 13 de OptiTrack, y también se integró un sensor de altura, el sensor LiDAR TFmini-S, para recoger medidas adicionales.

## 1.1. Motivación

Como se ha explicado anteriormente, los UAVs presentan múltiples aplicaciones posibles, un abanico que no hace más que ampliarse con los desarrollos tecnológicos de los últimos años. El potencial de los UAVs radica en que son la solución más versátil para realizar tareas complejas e incluso peligrosas para los seres humanos. Los drones adaptados para entornos de interiores, donde el GPS no es necesario, pueden realizar tareas

de todo tipo, desde el transporte, la inspección o incluso operaciones de búsqueda. En este caso, el desarrollo de un cuadricóptero autónomo de interiores permite juntar varias especialidades, como son las de la electrónica, robótica y mecánica, incluyendo un proceso completo de diseño, integración y ensayos. Este Trabajo Fin de Grado se centra en la integración de los distintos sensores y actuadores que constituirán el cuadricóptero (o dron), además del apartado de comunicaciones. Por otro lado, en el TFG de Álvaro Maestroarena Navamuel (titulado Diseño e implementación de un sistema de control de un cuadricóptero autónomo) se desarrolla el diseño del control del dron, mientras que el estudiante Antonio Engelinus Dijkema de Lucas ha diseñado la estructura del aparato.

## 1.2. Objetivos

El objetivo final de este proyecto es lograr que un cuadricóptero sea capaz de volar tanto de forma autónoma como manual en interiores. Para ello hay que cumplir de forma exitosa las siguientes fases:

- Calibrado e integración de los distintos componentes necesarios para el montaje de un cuadricóptero funcional (actuadores y sensores)
- Implementación de las comunicaciones necesarias tanto para el vuelo manual como autónomo
- Incorporar el hardware y software adicional necesario para conseguir el vuelo autónomo
- Pruebas de control del vuelo manual y autónomo, realizando los ajustes necesarios para el vuelo óptimo en ambos casos.

## 1.3. Metodología

La metodología de trabajo seguida se ha coordinado con Álvaro Maestroarena Navamuel, responsable del diseño del control del cuadricóptero. Los pasos seguidos fueron los siguientes:

- Se escogieron los distintos componentes necesarios para el montaje del dron, se configuraron ciertos parámetros del Cube Orange+ y se instalaron los programas necesarios, incluyendo las librerías necesarias en Matlab/Simulink, y el programa utilizado como estación de control, QGroundControl (QGC).
- Se realizó la calibración y los ensayos necesarios para verificar el comportamiento adecuado de los sensores y actuadores, y se procedió con ensayos relacionados con la comunicación de la emisora.
- Se colaboró en el desarrollo del control necesario para el vuelo manual (si bien se escapa del objetivo de este TFG), adaptando el control realizado en años anteriores, el cual se realizó para una Raspberry Pi, para poder utilizarlo con un CubePilot [4].

## 1.4. Recursos

Los recursos empleados incluyen componentes que han tenido que ser comprados, pero también componentes disponibles en la universidad. En el listado se incluyen todos los dispositivos que se han empleado.

### 1.4.1. Hardware

**CubePilot Cube Orange+:** Es un microcontrolador de la familia CubePilot (anteriormente conocida como PixHawk), dispositivo en el que se apoya todo el proyecto [13].


Figura 1.1: Microcontrolador Cube Orange+ (Fuente: RobotShop)

**TFmini-S:** Es un sensor de altura LiDAR. Se caracteriza por su alta precisión para medir altura en interiores, además de ser especialmente pequeño, ligero y de bajo consumo. Dado que el cuadricóptero será de interiores, este sensor es el que más ventajas y fiabilidad aporta frente a otras alternativas [14].

Figura 1.2: Sensor de altura TFmini-S (Fuente: Mouser Electronics)

**Batería Tattu 3S 1550mAh 45C:** Se compraron otras baterías de repuesto para tener siempre baterías cargadas disponibles durante los ensayos. Las baterías de repuesto fueron Zeee 3S 1500mAh 120C [15].


Figura 1.3: Batería Tattu 3S 1550mAh 45C (Fuente: Amazon)

**Motores CrazePony DX2205 2300KV y Racerstar BR2205 2300KV:** Motores BLDC de años anteriores. Se ha tenido en cuenta la estabilidad y eficiencia de estos modelos [16].


Figura 1.4: Motor CrazePony (izquierda) y motor Racerstar (derecha) (Fuente: Amazon)

**Hélices tripalas de 5 pulgadas (5x3.1x3.5):** Se consideran la mejor opción cuando lo que se busca es un mayor control y maniobrabilidad, teniendo en cuenta, por supuesto el tamaño del aparato [17].

Figura 1.5: Hélices tripalas (Fuente: APC Propellers)

**ESC BLHeli 20A:** Alimentados a través de la PDB y conectados a los puertos PWM del microcontrolador [18].


Figura 1.6: ESC BLHeli 20A (Fuente: Oscar Liang)

**PDB con conector XT60**, para baterías LiPo de 3S-4S: Permite alimentar los 4 ESCs, regulando la tensión a 5V [19].

Figura 1.7: PDB con conector XT60 (Fuente: Amazon)

**Emisora FS-i6S y receptor FS-A8S:** Para el manejo del dron durante los vuelos manuales ([8], [20]).

Figura 1.8: Emisora FS-i6S (Fuente: RC Innovations)

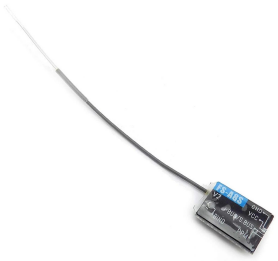

Figura 1.9: Receptor FS-A8S (Fuente: Banggood)

**ESP32-C3-DevKitM-1:** Placa con Wi-Fi y Bluetooth integrado, caracterizado por su bajo consumo. Necesario para enviar la información en tiempo real del dron a la estación de control, mediante comunicación inalámbrica. Se utiliza para la recepción y el envío de comandos durante el vuelo [21].

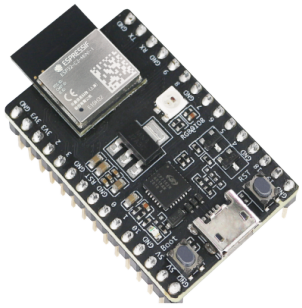

Figura 1.10: ESP32-C3-DevKitM-1 (Fuente: Amazon)

**Cámaras Flex13 de OptiTrack:** Cámaras utilizadas en años anteriores, por lo que ya se encuentran instaladas en la universidad. El sistema Motion Capture System (MCS) permite monitorizar el dron empleando cámaras infrarrojas repartidas por la zona de vuelo, permitiendo así el vuelo autónomo [10].


Figura 1.11: Cámara Flex13 de OptiTrack (Fuente: Optitrack)

**Estructura del UAV:** El cuadricóptero sigue una configuración X, y ha sido diseñada por Antonio Engelinus.

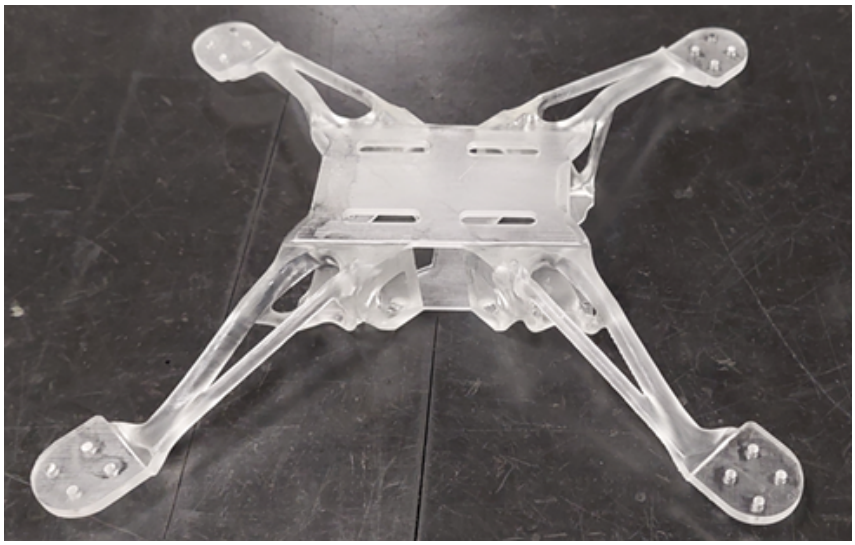

Figura 1.12: Prototipo de la estructura del dron

## 1.5. Software

**Matlab/Simulink (R2024b):** Se utilizó como programa principal del proyecto, con el que se realizaron las pruebas de los componentes y la implementación del control. Se utilizarán los paquetes UAV Toolbox y UAV Toolbox Support Package for PX4 Autopilots.

**QGroundControl (QGC):** Fue la estación de control. Con este programa se configuraron parámetros preliminares, se calibraron los ESC y permitía el acceso a la consola de MAVLink.

**Benewake_TF:** Plataforma oficial para la configuración del sensor TFmini-S.

# Capítulo 2

# Estado del Arte

## 2.1. Estado del Arte

### 2.1.1. Tipos de UAVs y sus características

Según la función que se quiera llevar a cabo, se suele usar un tipo de UAV u otro. Actualmente, se pueden clasificar dentro de 4 tipos, siendo los 2 primeros los más habituales: multirotor, ala fija, de un rotor e híbrido de ala fija. Según el modelo, tendrá una mayor duración de vuelo, mayor maniobrabilidad o eficiencia, al igual que la complejidad del vuelo será distinta [22].

Los UAVs multirotores son los más populares. Son VTOL (Vertical Take-Off and Landing), lo que les permite despegar y aterrizar en áreas pequeñas. Debido a que son mucho más agiles que los drones de ala fija, suelen emplearse en áreas exteriores pequeñas o zonas interiores. Ofrecen un mayor manejo y estabilidad, por lo que son los UAVs elegidos habitualmente para fotografía, vídeos e inspecciones. Su principal desventaja es la poca autonomía, otro motivo por el que no se usan en áreas de gran extensión.

Los UAVs de ala fija se utilizan para cubrir áreas de gran tamaño, al tener una mayor autonomía. Tienen una menor maniobrabilidad y requieren zonas amplias para el despegue y aterrizaje, pero cuentan con la ventaja de mayor tiempo de vuelo, mayor eficiencia, transportar mayor peso y, en caso de que un fallo de alimentación, no se pierde el control del aparato, sino que puede seguir planeando hasta el aterrizaje, mientras que, en el caso de un multirotor, un fallo de alimentación provocaría una pérdida de control del aparato. Suelen utilizarse para mapear áreas de gran tamaño o la inspección de líneas eléctricas.

Los drones de un solo rotor no son habituales, debido a su complejidad, inestabilidad y coste. Son más difíciles de pilotar, y tienen poca autonomía. Pueden mantenerse fijos en el aire durante más tiempo que los multirotores, por lo que suelen utilizarse para escaneo láser.

Finalmente, los drones híbridos de ala fija son una mezcla entre los multirotores y los de ala fija. Pueden despegar y aterrizar verticalmente y pueden mantenerse fijos en el aire como los multirotores, a la vez que mantienen la velocidad de los drones de ala fija y su autonomía. Sin embargo, estos drones aún se encuentran en desarrollo, y pocos están disponibles en el mercado.

## 2.2. Navegación autónoma en interiores

### 2.2.1. Simultaneous Localization and Mapping (SLAM)

La técnica SLAM consiste en el procesamiento de señales de sensores y optimización de grafos de pose. Es una técnica con la que el vehículo autónomo opera en un entorno inicialmente desconocido, utilizando los sensores que tiene a bordo para construir un mapa de dicho entorno, el cual utiliza para localizarse al mismo tiempo.

No necesita sensores externos, sino que se utilizan exclusivamente los sensores integrados en el dron, como cámaras o sensores LiDAR, lo que permite una autonomía total al UAV.

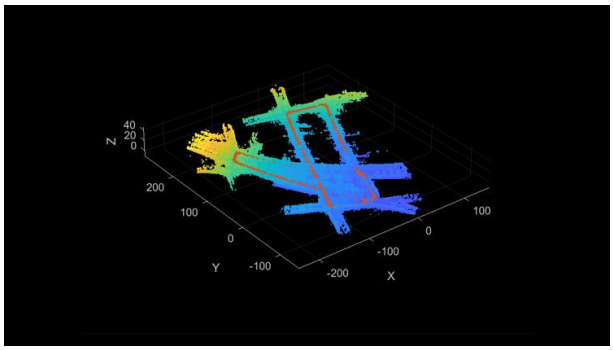


Figura 2.1: SLAM (Fuente: MathWorks)

### 2.2.2. Motion Capture System (MCS)

El sistema de captura en movimiento consiste en varias cámaras infrarrojas situadas en la zona donde va a operar el UAV para monitorearlo. Estas cámaras permiten determinar la posición del dron en el espacio, y la trayectoria que debe seguir para alcanzar su destino, teniendo en cuenta los obstáculos. Este sistema permite un seguimiento muy preciso del vehículo, tanto de su desplazamiento como de su orientación. Los principales sistemas MCS provienen de OptiTrack y VICON y, dado que la universidad ya las tenía instaladas, se escogieron las cámaras de la marca OptiTrack.

Evidentemente, al requerir un hardware externo instalado en una zona determinada, el UAV tiene una autonomía limitada, al no poder salir de dicha zona.

Por otro lado, cabe destacar que el sistema MCS no solo permite el funcionamiento autónomo de un dron, sino que podría utilizarse con varios drones a la vez, permitiendo que estos "enjambres" sean capaces de realizar tareas más complejas o de completarlas de forma más eficiente [23].

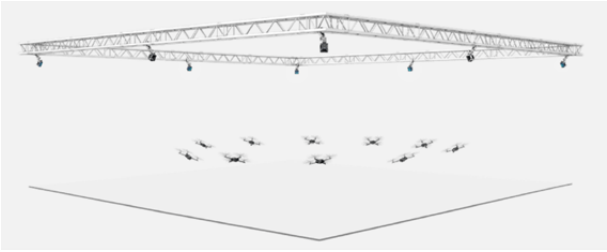

Figura 2.2: MCS (Fuente: Optitrack)

## 2.3. Aplicaciones

### 2.3.1. Agricultura

Actualmente, los drones en la agricultura se utilizan para la topografía, cartografía, el monitoreo de los cultivos (humedad, condiciones del terreno) y de enfermedades o plagas. El uso de UAVs en la agricultura permite mejorar la producción, ahorrar tiempo y reducir la incertidumbre [5].

### 2.3.2. Logística

Los drones pueden utilizarse para cumplir tareas como la gestión de inventarios, el reparto de productos o incluso la vigilancia. Actualmente, se está produciendo un incremento del uso de drones para controles de inventario, reduciendo significativamente el tiempo requerido para realizar la tarea, y evitando el posible peligro que podría suponer que lo hiciera una persona. Este es el caso del dron Kaires, de Airvant, capaz de realizar el inventario de manera autónoma ([24], [25]).

### 2.3.3. Operaciones de rescate

Los drones están cada vez más equipados para utilizarse en diversas operaciones de búsqueda y rescate (SAR), mejorando la eficiencia y seguridad durante el proceso. Las aplicaciones de los drones SAR difieren según el dron y el hardware que tenga integrado.

Hay drones equipados con cámaras de alta definición y gran autonomía, útiles en labores de rescate realizadas por bomberos, permitiendo que su labor sea lo más eficaz y segura para ellos [26]. También existen drones enfocados para el monitoreo de desastres naturales, que pueden estar equipados con sensores multiespectrales, lo que puede ayudar a localizar personas atrapadas [27], [28]. Por otro lado, existen drones enfocados en el salvamento marítimo, los cuales cuentan con cámaras de alta resolución, sensores térmicos y son capaces de transportar el material de asistencia necesario [29].

Finalmente, cabe destacar que, como los drones cuentan con la ventaja de ser mucho más ágiles que los vehículos aéreos tripulados, permitiendo que cubran más terreno en menos tiempo, son una herramienta muy recurrida por la policía en la búsqueda de personas desaparecidas [30].


Figura 2.3: Dron de salvamento, Comunidad Valenciana (Fuente: elDiario)

# Capítulo 3

# Hardware

## 3.1. Cube Orange Plus


Figura 3.1: Microcontrolador Cube Orange Plus (Fuente: RobotShop)

El microcontrolador utilizado para el cuadricóptero es el Cube Orange Plus, con la placa ADS-B integrada, y está especialmente diseñado para drones y otros vehículos autónomos. Forma parte de la familia CubePilot, también conocida como Pixhawk.

Las características principales de este dispositivo son:

- Procesador principal: STM32H757 Dual Core M7+M4
- Memoria Flash 2MB

- Memoria RAM 1MB
- 400 MHz CPU
- Coprocesador: STM32F100 (“failsafe”)
- Memoria Flash 64KB
- Memoria RAM 8KB
- 3 Sensores IMU redundantes
- 2 Barómetros
- 1 Magnetómetro
- La placa ADS-B cuenta con un receptor (1090MHz) integrado que recibe la actitud y localización de vuelos comerciales cercanos al dron, para garantizar el vuelo seguro.

El procesador principal es un chip con dos núcleos de procesamiento independientes. El Cortex-M7 es el encargado de realizar los cálculos más complejos, el filtrado y el control de vuelo. El Cortex-M4, por otro lado, se utiliza para realizar tareas menos pesadas en paralelo, como la lectura de los sensores y las comunicaciones. Contar con 2 núcleos permite obtener un mejor rendimiento y velocidad de respuesta durante el vuelo.

El coprocesador, por otro lado, es un procesador de seguridad. Como se puede observar en sus características, es mucho más sencillo que los anteriores, ya que su función es proteger el sistema, en caso de que el procesador principal no responda. Es el “failsafe” del aparato, capacitado para activar modos de emergencia en caso de que ocurra algún problema.

Si bien es cierto que este microcontrolador no cuenta con memoria FRAM, como lo hacía su antecesor el Cube Black, el Cube Orange Plus combina la memoria Flash y RAM para el procesamiento de datos. Además, cuenta con una tarjeta microSD, utilizada para almacenar datos de vuelo sobre los distintos sensores.

Los sensores redundantes, como es el caso de las IMU y los barómetros, permiten comparar las lecturas, ofreciendo resultados más precisos y un control más robusto, y garantizan una mayor fiabilidad, ya que el sistema no se verá gravemente afectado en caso de que un sensor falle. Además, cuenta tanto con IMUs estabilizadas térmicamente, como con protección contra las vibraciones excesivas, garantizando medidas con el menor ruido posible.

**Interfaces de comunicación:**

- UART (4 puertos)
- I2C (2 puertos)
- CAN (2 puertos)

Una de las principales ventajas que tiene I2C es que permite conectar múltiples dispositivos, tanto maestros como esclavos, en un mismo bus. Aunque en este caso solo se utilizó un sensor que requería de la comunicación I2C, si en el futuro fuera necesario implementar un dispositivo (adicional a los utilizados) que admitiera comunicación UART, no se tendría que modificar ningún componente ya integrado.

Además, interesaba que el sensor de altura estuviera configurado con la comunicación I2C al ser más rápida que la UART, algo importante durante los vuelos autónomos, para que el dron reciba datos y tome decisiones en tiempo real en el menor tiempo posible.

**Puertos empleados:**

- **POWER1:** Se utilizó uno de los puertos de alimentación para la conexión de la batería.
- **RC IN:** Se utilizó para la conexión del receptor, pudiendo utilizar la emisora para controlar el dron durante los vuelos manuales.
- **MAIN OUT:** Se utilizaron los pines del 1 al 4 para la conexión de los ESC. Transmiten la señal PWM necesaria para el movimiento de los motores.
- **GPS1:** Este puerto está disponible tanto para el Safety Switch, como para el GPS (el cual cuenta con su propio Safety Switch). Dado que el dron se utilizará en interiores, no tenía sentido integrar un GPS, por lo que se conectó únicamente el Safety Switch.
- **I2C 2:** Es el segundo puerto que permite comunicación I2C, y se utilizó para el sensor de altura, el TFmini-S.
- **TELEM1**: Este puerto se utilizó para la comunicación inalámbrica, implementando un ESP32-C3.

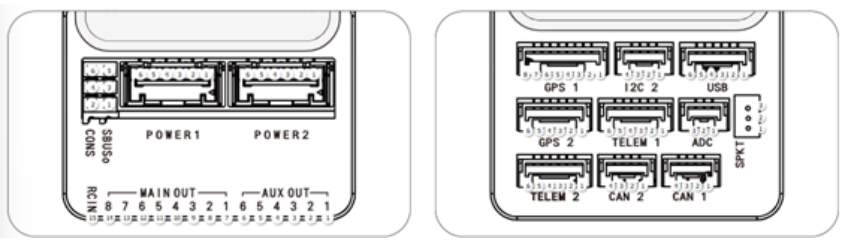


Figura 3.2: Puertos Cube Orange Plus (Fuente: CubePilot)

Finalmente, se utilizarán, de entre todos los sensores integrados del microcontrolador, los acelerómetros y giróscopos disponibles.

En la Figura 3.3 se puede observar la estructura global del Cube Orange+. La FMU se encarga del filtrado y control, además de la lectura de los sensores y el manejo de los puertos auxiliares (AUX OUT). La IO, el coprocesador, se encarga de las comunicaciones RC, las cuales serán SBUS, para controlar los motores PWM conectados a los puertos principales (MAIN OUT). El coprocesador, como se ha explicado anteriormente, recibe información relacionada con el Failsafe.

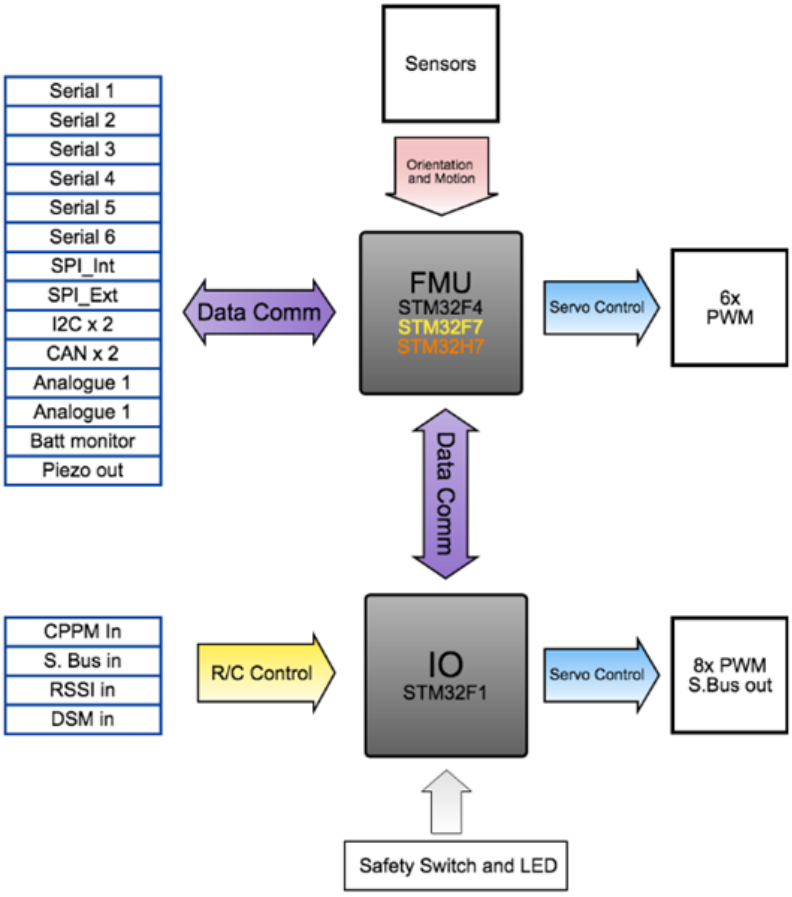


Figura 3.3: Arquitectura del sistema (Fuente: CubePilot)

## 3.2. TFmini-S

Se utilizó el Benewake TFmini-S, para medir la altitud. El propósito de este sensor es utilizarlo durante el vuelo autónomo, para tener información adicional a las cámaras OptiTrack y los propios sensores del microcontrolador. Es un sensor LiDAR (Light Detection and Ranging), es decir, un sensor láser, un tipo de sensor caracterizado por ser mucho más preciso y con un mayor alcance que los más comúnmente conocidos, los ultrasónicos.

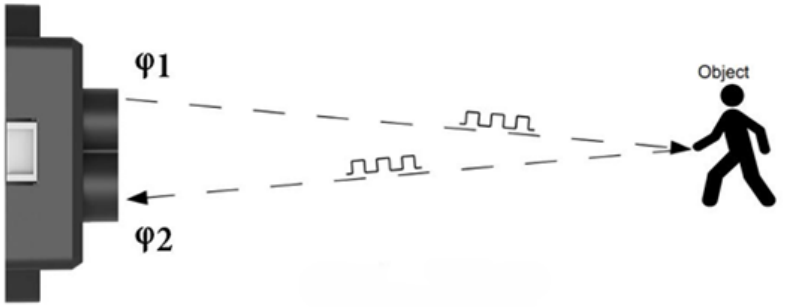


Figura 3.4: Funcionamiento ToF (Fuente: Benewake)

Los sensores "Time of Flight" (ToF), cuentan el tiempo que tarda en ir y volver la señal que envían periódicamente. Es el caso de los sensores ultrasónicos y LiDAR.

Concretamente, el TFmini-S se caracteriza por su tamaño reducido y su peso despreciable, con bajo consumo energético y alta precisión. Esto lo hace el sensor idóneo para un cuadricóptero de interiores como es el caso, donde priman el tamaño reducido y el bajo consumo.

En cuanto a los principales parámetros de este sensor, cabe destacar:

- El sensor tiene un alcance de hasta 12m, algo más que suficiente para medidas en interiores.
- Tiene una precisión de ±6cm hasta los 6m. Superada esta distancia, la precisión oscila en ±1 %. La precisión en el Datasheet fue calculada bajo las siguientes condiciones: una pared blanca, con reflectividad del 90 %, a 25ºC (aunque es cierto que un sensor ultrasónico es más sensible a los cambios térmicos, los LiDAR también pueden verse afectados). [7]
- Tiene una tasa de actualización de 100Hz, es decir, toma una nueva medida cada 10ms.

Se realizó un ensayo para comprobar la precisión del sensor en las condiciones del laboratorio, con el objetivo de determinar la medida mínima que puede reconocer y el error medio que comete para un rango inferior a 6m.

Siguiendo la información del fabricante, el sensor es operativo a partir de los 100mm, aunque en la Figura 3.5 el error se mantiene en un porcentaje significativo hasta los 500mm. Esto no supondrá un problema, ya que no se ha requerido de una alta precisión en distancias tan pequeñas. Además, a medida que los valores se acercan a los 3m, una altura razonable de vuelo para el cuadricóptero, el error se hace prácticamente despreciable.

| Medida real (mm) | Medida sensor (mm) |
|---|---|
| 75 | 1 |
| 100 | 27 |
| 500 | 450 |
| 1000 | 963 |
| 1500 | 1472 |
| 2000 | 1978 |
| 2500 | 2478 |
| 3000 | 2979 |

Tabla 3.1: Ensayos TFmini-S

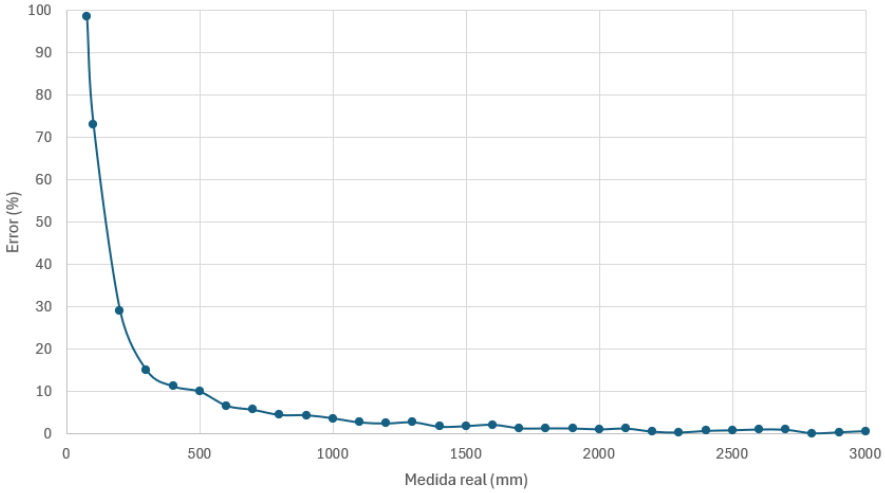


Figura 3.5: Error de medida TFmini-S

Aunque el rango del sensor alcanza hasta los 12m, no se comprobó el extremo superior porque el dron no volará a alturas tan elevadas.

El sensor TFmini-S suele tener que demandar una corriente media de 50mA, aunque varía según la distancia a medir [31]. Tan poco consumo beneficiará el uso de una batería más pequeña y ligera, algo que se ha tenido en cuenta durante la elección de todos los componentes.

Por otro lado, es compatible con interfaces UART e I2C. La configuración original era UART, pero se prefirió optar por I2C, como se explicó anteriormente.

## 3.3. Motores Brushless DC

Los motores brushless DC (BLDC) son motores síncronos que utilizan controladores electrónicos para alternar la corriente continua por los devanados, generando campos magnéticos que los imanes permanentes del rotor siguen.

Cuando circula corriente por un hilo conductor, se forma un campo magnético. Según el sentido de la corriente, este campo magnético tendrá la polaridad en un sentido o en otro. En este caso, los motores constan de 14 imanes en el rotor y 12 bobinas. El cable correspondiente a cada fase está enrollado en 4 núcleos en total. Los núcleos contiguos se enrollan en sentido contrario (uno horario, otro antihorario), generando una polaridad opuesta en sus extremos cuando circula corriente por su cable, permitiendo así el empuje

de los imanes de la campana del motor.

Los motores BLDC son trifásicos, y su disposición habitual es en triángulo. La interacción de los imanes permanentes del rotor con las bobinas por las que circula la corriente es lo que provoca el movimiento del motor. La corriente alterna se controla con los ESC, los cuales utilizan la fase por la que no circula corriente en cada momento para determinar cuándo debe producirse la conmutación.

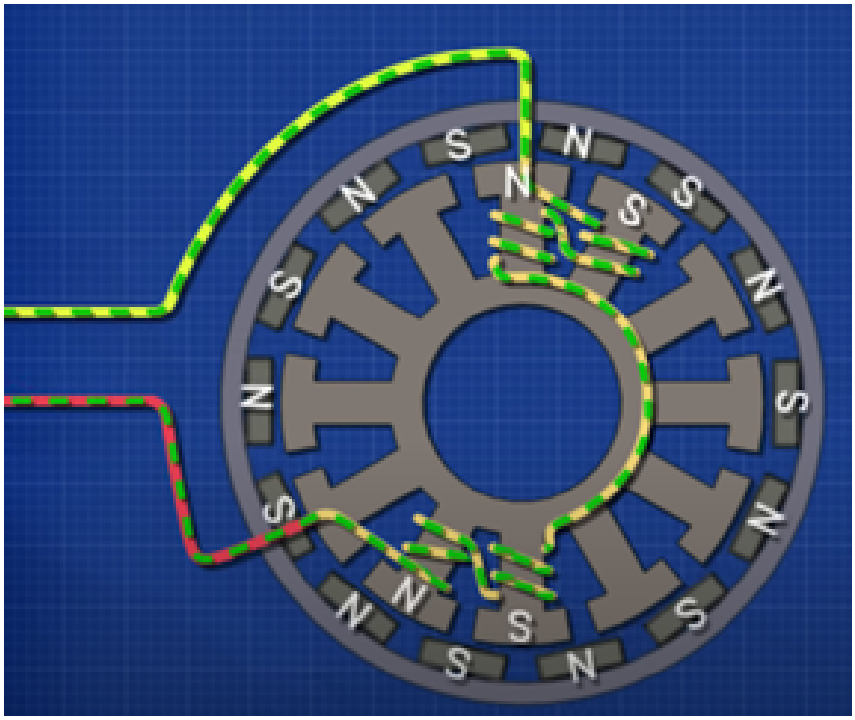

Figura 3.6: Bobinado motores BLDC (Fuente: The Engineering Mindset)

Se escogieron motores de 2300KV (siendo KV las revoluciones por minuto (RPM) que puede entregar el motor por voltio), al ser los que mejor se ajustan a los vuelos estables, eficientes y con un mayor tiempo de vuelo. Cabe destacar que, al igual que con otros componentes del cuadricóptero, se reutilizaron motores que se encontraban en el laboratorio y cumplían las especificaciones. Había 7 motores disponibles, de 2 marcas diferentes. Inicialmente, se contempló la posibilidad de usar únicamente la marca Racerstar, al tener 4 motores de esta. Sin embargo, se dicidió hacer ensayos de empuje de los 7 motores, con el objetivo de escoger los 4 más parejos y evitar problemas futuros con el diseño del control. Los ensayos se realizaron dentro del rango de velocidad de los motores, es decir, entre 1000$\mu$s (parado) y 2000$\mu$s (máxima velocidad).

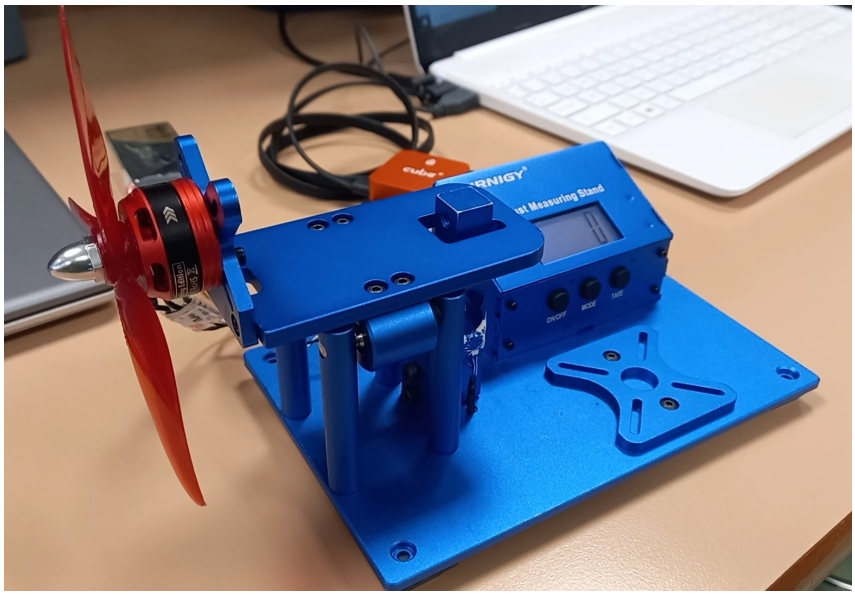

Figura 3.7: Banco de ensayos

En la Tabla 3.3, se encuentran los motores que a nivel global dieron los resultados más similares. 3 de estos motores eran de la marca Racerstar, pero el cuarto correspondía a la marca Crazepony. Por este motivo, se utilizaron motores de marcas diferentes, pero todos siendo de 2300KV y de igual tamaño.

Es importante destacar que no todos los motores podían combinarse entre sí, ya que unos estaban preparados para girar en un sentido, y otros en el sentido opuesto. Teniendo esto en consideración, los motores escogidos fueron los numerados como 1, 3, 4 y 7.

| | **Empuje (g)** | | | | | | |
|---|---|---|---|---|---|---|---|
| **PWM (µs)** | **MOTOR 1** | **MOTOR 2** | **MOTOR 3** | **MOTOR 4** | **MOTOR 5** | **MOTOR 6** | **MOTOR 7** |
| 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1100 | 14,0 | 11,0 | 5,0 | 24,0 | 30,0 | 27,0 | 10,0 |
| 1200 | 22,0 | 24,0 | 28,0 | 48,0 | 56,0 | 51,0 | 35,0 |
| 1300 | 55,0 | 42,0 | 60,0 | 76,0 | 85,0 | 81,0 | 67,0 |
| 1400 | 96,0 | 66,0 | 104,0 | 117,0 | 127,0 | 120,0 | 109,0 |
| 1500 | 142,0 | 85,0 | 144,0 | 164,0 | 164,0 | 150,0 | 145,0 |
| 1600 | 185,0 | 110,0 | 181,0 | 196,0 | 202,0 | 182,0 | 183,0 |
| 1700 | 225,0 | 135,0 | 208,0 | 243,0 | 243,0 | 219,0 | 222,0 |
| 1800 | 252,0 | 162,0 | 242,0 | 290,0 | 290,0 | 251,0 | 261,0 |
| 1900 | 317,0 | 190,0 | 285,0 | 342,0 | 344,0 | 282,0 | 305,0 |
| 2000 | 370,0 | 217,0 | 335,0 | 408,0 | 401,0 | 345,0 | 350,0 |

Tabla 3.2: Empuje en gramos de cada motor según PWM aplicado

| Empuje (g) | | | | | | |
|---|---|---|---|---|---|---|
| **PWM (µs)** | **MOTOR 1** | **MOTOR 3** | **MOTOR 6** | **MOTOR 7** | **Desviación típica** | **Media** |
| 1000 | 0 | 0 | 0 | 0 | 0,000 | 0,000 |
| 1100 | 14,0 | 5,0 | 24,0 | 10,0 | 8,057 | 11,385 |
| 1200 | 22,0 | 28,0 | 48,0 | 35,0 | 11,177 | 33,985 |
| 1300 | 55,0 | 60,0 | 76,0 | 67,0 | 9,110 | 64,025 |
| 1400 | 96,0 | 104,0 | 117,0 | 109,0 | 8,813 | 106,226 |
| 1500 | 142,0 | 144,0 | 164,0 | 145,0 | 10,243 | 148,496 |
| 1600 | 185,0 | 181,0 | 196,0 | 183,0 | 6,702 | 186,141 |
| 1700 | 225,0 | 208,0 | 243,0 | 222,0 | 14,387 | 224,156 |
| 1800 | 252,0 | 242,0 | 290,0 | 261,0 | 20,678 | 260,654 |
| 1900 | 317,0 | 285,0 | 342,0 | 305,0 | 23,824 | 311,571 |
| 2000 | 370,0 | 335,0 | 408,0 | 350,0 | 31,606 | 364,749 |

Tabla 3.3: Empuje en gramos de los motores utilizados según PWM aplicado

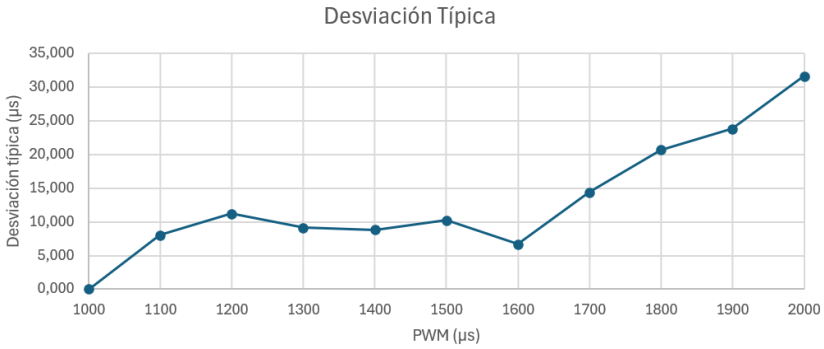


Figura 3.8: Desviación típica en función del PWM

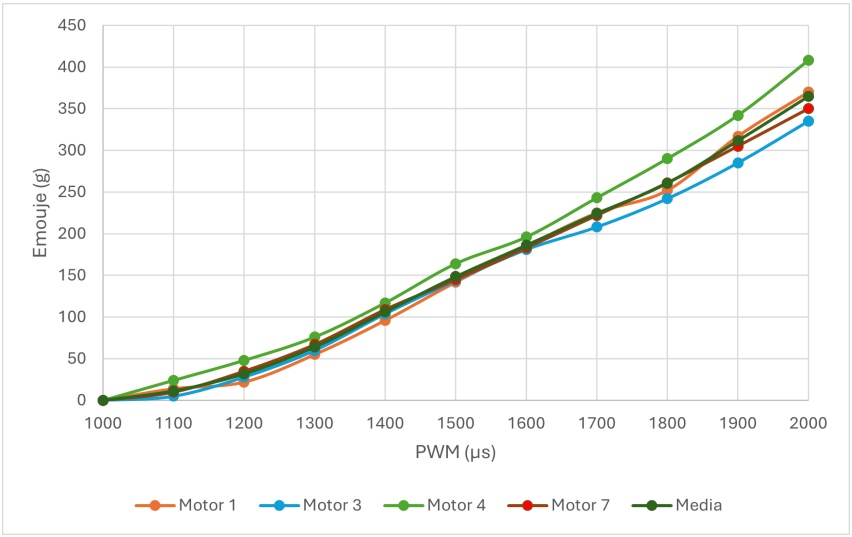


Figura 3.9: Empuje de los motores en función del PWM

### 3.3.1. ESC

Un Electronic Speed Controller (ESC) es un circuito electrónico encargado de controlar la velocidad de los motores al recibir pulsos PWM. Algunos ESC pueden controlar el sentido de giro de los motores, pero este no es el caso. Los ESC utilizados son los BLHeli 20A, los cuales soportan hasta 20A continuos, un amperaje muy por encima de la demanda. Un ESC consta de 3 partes principales:

- Puente trifásico DC-AC compuesto por 6 MOSFETs con diodos de libre circulación
- Procesador: encargado de determinar por qué fases circulará corriente, además de ser el encargado de enviar los pulsos PWM a los MOSFETs correspondientes.
- "Gate driver": Evita que, durante las conmutaciones de corriente (las cuales son a gran velocidad), la corriente entre en el Gate. También es el encargado de amplificar los pulsos PWM recibidos del procesador.

Los MOSFETs actúan de interruptores, conmutando rápidamente entre ON y OFF según el gate driver y el procesador. Cada pareja de MOSFETs (de fases distintas, lógicamente) actuará como interruptor, provocando el movimiento del motor. El giro que realizan cada vez es controlado por el procesador. Los motores BLDC utilizados constan de 14 imanes permanentes en el rotor, es decir, hay 7 pares polares. Dado que cada conmutación abarca $60^{o}$ eléctricos, $60/7 = 8.57^{o}$ mecánicos. Por tanto, son necesarias 42 conmutaciones para que el motor dé una vuelta completa.

La pareja de MOSFETs que actúe como interruptor recibirá una señal PWM por su Gate y, si es una tensión superior a la tensión umbral, entonces conducirán (ON). En caso contrario, permanecerán en OFF. Cuanto mayor sea el ancho de pulso de la señal PWM, más rápido girará el motor.

El procesador puede determinar cuándo toca que circule otra pareja de fases de 2 formas distintas: Efecto Hall si el motor dispone del sensor necesario (no es este caso) o mediante la fuerza contraelectromotriz (mide la tensión de la fase que no se esté usando en ese momento respecto al neutro virtual).

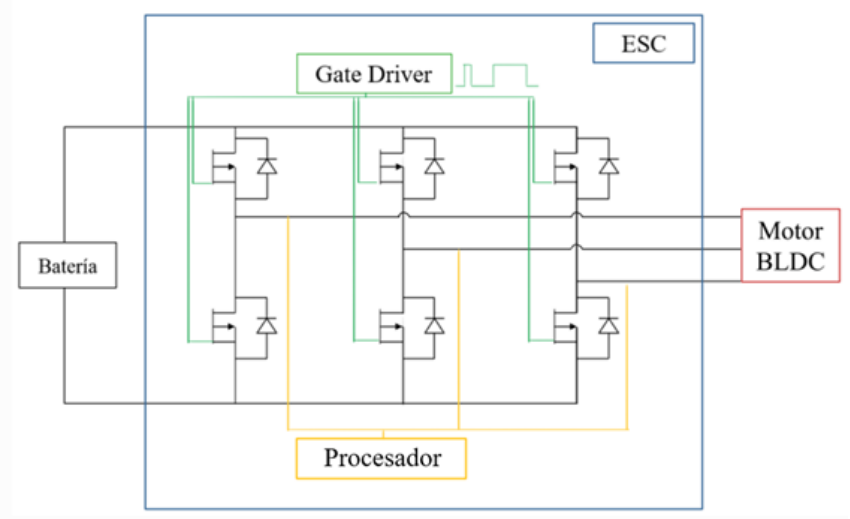


Figura 3.10: Esquema de un ESC

Los ESC se conectan a los puertos PWM del microcontrolador y se alimentan a través de una PDB (Power Distribution Board), la cual está conectada a la batería. Sus salidas

se conectan directamente a los motores. La PDB es necesaria ya que necesitamos 4 ESC, uno para cada motor, y utilizarla permite distribuir la potencia suministrada por una batería.

La PDB recibe la tensión de la batería y permite repartirla a los ESC. Ahorra espacio, algo fundamental en este caso, y minimiza el riesgo de un posible cortocircuito en los ESC.

La PDB utilizada tiene conector XT60, diseñada para baterías LiPo de 3S-4S (entre unos 11-15V). Dado que la batería usada es de 3 celdas y 11.1V, cumple con las especificaciones. Además, admite la conexión de hasta 4 ESC, perfecto para un cuadricóptero. Consta de salidas reguladas que transforman la tensión de la batería a 5V, las cuales no fueron necesarias para este proyecto.

### 3.3.2. Hélices

Para la elección de las hélices, hubo que tener en cuenta 3 conceptos clave: el diámetro, el paso o "pitch" y el número de palas:

- Un mayor diámetro implica un mayor empuje, pero también, el mayor tamaño implicará que serán más pesadas.
- Por otro lado, un pitch mayor o menor afectará al empuje del aire y la velocidad de avance con cada revolución. Se suele buscar un pitch grande en drones de competición, donde se buscan maniobras rápidas y un vuelo por lo general más agresivo.
- Cuantas más palas haya, más estable y mayor empuje tendrá la hélice, aunque también tendrá mayor resistencia con el aire y será más pesada.

Se escogieron unas hélices 5x3x3. Un diámetro de 5 pulgadas es una medida estándar y habitual para drones, al igual que las hélices tripalas. Para hélices de 5" un pitch de aproximadamente 3" también suele ser un rango estándar, al permitir un compromiso óptimo entre empuje, eficiencia y velocidad. En la elección de las hélices se priorizó el tamaño y la maniobrabilidad y estabilidad del dron.

## 3.4. Baterías

Para el funcionamiento del dron, solo es necesario una batería. Se reutilizó una batería Tattu 3S 11.1V 1550mAh 45C, pero también se compraron baterías Zeee LiPo 3S 11.1V 1500mAh 120C. Estas baterías se pueden conectar directamente al Cube Orange+, y además cuentan con el conector XT60, permitiendo alimentar de forma sencilla los actuadores.

Teóricamente, la batería de la marca Tattu permite una corriente continua máxima:

$$Imax = 1{,}5 * 120 = 180A \tag{3.1}$$

Evidentemente, nunca se llegará a valores extremos en el proyecto. 120 C es el "C-rating", es decir, la medida que indica la corriente y la rapidez con la que se carga o descarga una batería.

Una batería de 3 C y 100Ah significa que puede proporcionar 300A durante 1/3 h (es decir, 20min). Si es 1 C serán 100A durante 1h y si es 0.5C, entonces 50A durante 2h.

En este caso, al ser 1.5Ah y 120C, significa que podrá suministrar 180A durante el transcurso de 1/120 h, es decir, 30s.

Las baterías con mayor C tienen una resistencia interna menor, lo que implica principalmente 2 cosas: por un lado, sufre una caída de tensión menor, por lo que el voltaje que entrega es prácticamente el mismo al que recibe el dispositivo y, por efecto Joule, presenta una menor generación de calor, lo que prolonga la vida de la batería.

En este caso se tuvo en cuenta que el peso de la batería sería el más significativo. De hecho, que la batería fuera algo más pesada era beneficioso para el control del UAV, ya que la potencia de los motores era superior a lo que se tenía previsto inicialmente.

En principio, un C elevado es más común para drones de competición, mientras que un C más bajo es habitual en vuelos suaves como sería este caso. Por este motivo, se utilizó la batería Tattu para los ensayos definitivos, mientras que, para las pruebas, donde el tiempo de funcionamiento era breve, se recurrió a las baterías Zeee. La duración de las baterías seguía siendo alta en los 2 casos, pero la batería Tattu presentaba mejores especificaciones para el desarrollo de este proyecto, ya que contaba con una menor C y un mayor amperaje.

Igualmente, si bien la elección de las baterías podría haber sido más adaptada a las necesidades de este proyecto, en todos los casos permitían la autonomía del cuadricóptero, debido al bajo consumo de algunos componentes ([32], [33]).

## 3.5. Emisora

El vuelo manual se realizó utilizando la emisora FS-i6S, que también estaba disponible en el laboratorio. Se utilizó un receptor FS-A8S el cual se conectaba directamente al puerto RC del Cube Orange+ (véase Figura 3.2).


Figura 3.11: Emisora FS-i6S

La emisora cuenta con 10 canales configurables, de los cuales se utilizarán 3 para la selección del modo de vuelo: detener los motores, vuelo manual y vuelo autónomo. Concretamente, se utilizará la palanca SwB para ello, dejando el resto en su posición estándar. Funciona a 2.4GHz y sigue el protocolo AFHDS 2A. Tiene un alcance de hasta 1500m, más que suficiente para cubrir las necesidades del UAV.

2.4GHz es la banda ISM (Industrial, Scientific, Medical). Se usa en la mayoría de las emisoras RC, al ofrecer una conexión robusta frente a interferencias, con salto de

frecuencia (FHSS- Frequency-hopping spread spectrum) (la banda se divide en distintos canales, cambiando entre ellos para evitar las interferencias), y permitiendo que múltiples transmisores operen al mismo tiempo.

| SwB Position | SwC Position | Mode |
|---|---|---|
| Up | Up | 1 |
| Up | Middle | 2 |
| Up | Down | 3 |
| Middle | Down | 4 |
| Middle | Middle | 5 |
| Middle | Up | 6 |
| Down | Up | 7 |
| Down | Middle | 8 |
| Down | Down | 9 |

Figura 3.12: Canales disponibles

El protocolo AFHDS 2A es la segunda generación del "Automatic Frequency Hopping Digital System" de FlySky. Este protocolo permite recibir información del receptor, como por ejemplo, el voltaje del UAV, y también cuenta con un sistema de seguridad, donde los canales adquieren unos valores preconfigurados en caso de que se pierda la señal con el aparato.

Típicamente, las emisoras RC emiten con menos de 20dBm, es decir, emiten señales a potencias inferiores de 100mW. Según con lo establecido por la Unión Europea, para las condiciones de uso en banda ISM (2.4GHz), los equipos de baja potencia, como es el caso de un dron, solo pueden emitir hasta 20dBm sin licencia [34].

La emisora permite la configuración completa de los joysticks, con los que se controla el empuje, el alabeo, el cabeceo y la guiñada (throttle, roll, pitch y yaw, en inglés).

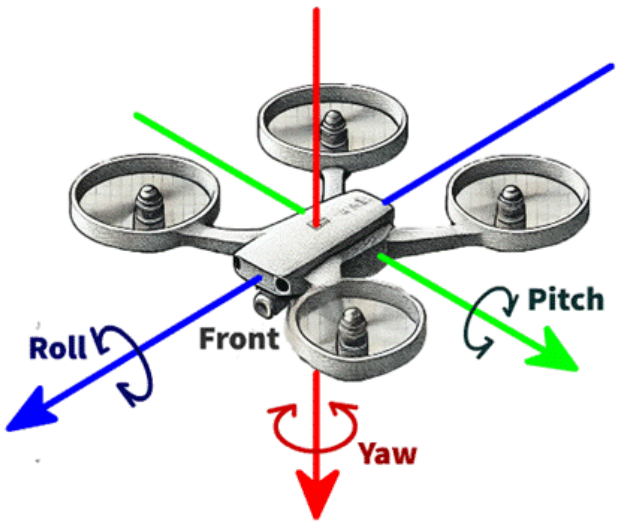


Figura 3.13: Sistema de ejes de un UAV (Fuente: StickMan Physics)

El modo de vuelo escogido es el 2, el más popular para el manejo de drones:

Como es habitual en las emisoras de drones, si bien los joysticks en el manejo de Roll, Pitch y Yaw, regresan al punto central si se sueltan, en el eje del Throttle eso no sucede.El joystick que maneja el Throttle se mantendrá en la posición que se haya dejado

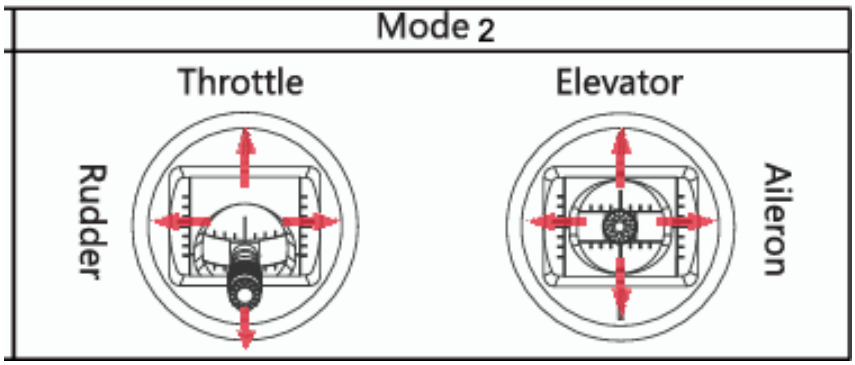


Figura 3.14: Modo de vuelo. Configuración joysticks (Pitch= Elevator, Roll= Aileron, Yaw = Rudder) (Fuente: FlySky)

y no volverá a su posición central. Es un motivo de seguridad, y todas las emisoras para drones son de esta manera, para evitar que se produzca accidentalmente un cambio brusco en la altitud del dron.

## 3.6. ESP32-C3-DevKitM-1

La placa ESP32-C3-DevKitM-1 es una placa que integra tanto funciones WiFi como Bluetooth. En este proyecto se utilizó la placa para la comunicación inalámbrica entre el UAV y la estación de control (QGroundControl), con el propósito de recibir mensajes MAVLink en tiempo real durante el vuelo y así realizar un estudio del control de vuelo de forma inmediata.

Este dispositivo permite una alimentación tanto de 5V como de 3.3V, al tener integrado un regulador de tensión que convierte la alimentación de 5V a 3.3V. Aunque los puertos del Cube Orange+ se mantuvieron a 3.3V, se prefirió alimentar la ESP32C3 por el pin de 5V, ya que este es el único que cuenta con protección de tensión.

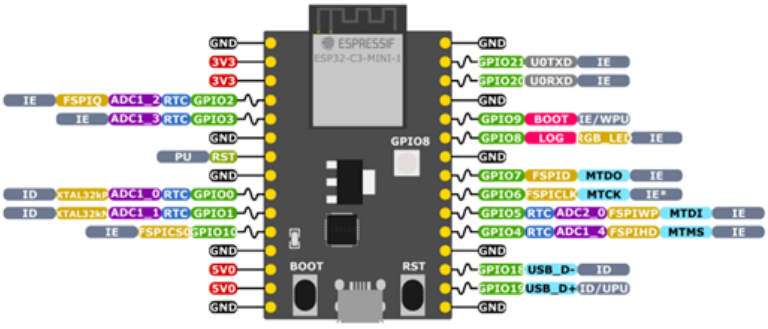


Figura 3.15: Disposición pines ESP32-C3 (Fuente: Espressif)

Por otro lado, la placa también cuenta con 2 botones, el BOOT y RESET (RST). Estos botones fueron necesarios únicamente para la descarga del firmware.

Otros pines utilizados (además de fuente y tierra) fueron los correspondientes a GPIO 4 y GPIO 5, es decir, los pines RX y TX respectivamente. Si bien es cierto que en la Figura 3.15 los pines correspondientes serían GPIO 20 y GPIO 21 [9], estos pines suelen ser el puerto UART predeterminado de depuración y programación del ESP32-C3.

Dentro de la placa, existen varios controladores UART. UART-0 (GPIO 20 y GPIO 21), es utilizado para la depuración y los mensajes de diagnóstico. La mayoría de los firmwares, como DroneBridge, mantienen UART-0 para la depuración, y no la reconfiguran. UART-1, en cambio, es la más habitual para telemetría, sensores, Bluetooth... y sus pines más comunes son GPIO 5 para TX y GPIO 4 para RX.

En este caso, el firmware utilizado ha sido el de DroneBridge, el cual no reasigna el UART después de la instalación. Por este motivo, los pines GPIO 20 y GPIO 21 posteriormente son "inservibles", y deben utilizarse para la comunicación inalámbrica los GPIO estándar de este dispositivo, los cuales corresponden con el 4 y el 5 [35].

En total, se necesitaron 4 pines de los 6 disponibles en el puerto TELEM1 del cubo, ya que los pines restantes, CTS y RTS, no son necesarios en este caso, como se explicará más adelante en el apartado de comunicaciones.

## 3.7. Cámaras OptiTrack Flex13

Si bien no se llegaron a utilizar las cámaras debido a que no se realizó el vuelo autónomo, es conveniente explicar el motivo por el que estas cámaras son una buena opción para realizar vuelos autónomos de interiores.

Estas cámaras se caracterizan principalmente por su baja latencia y su alta frecuencia, de tal manera que cada cámara es capaz de entregar hasta 120 FPS con tan solo unos 8ms de retardo. Además, cuenta con una precisión milimétrica (de unos 0.20mm). Todo esto asegura un vuelo estable en interiores y sin GPS.

Para trabajar con estas cámaras, es necesario incorporar marcadores en el dron. Los marcadores son puntos de referencia óptimos utilizados para el cálculo de las coordenadas 3D. Los marcadores que se habría utilizado son los conocidos como marcadores pasivos, unas esferas reflectantes que se colocan de forma asimétrica (para poder determinar la orientación del dron). De las cámaras sale luz infrarroja que rebota en las esferas y regresa. Esta información la transforma en una nube de píxeles, con la que es capaz de determinar las coordenadas 3D con alta precisión.

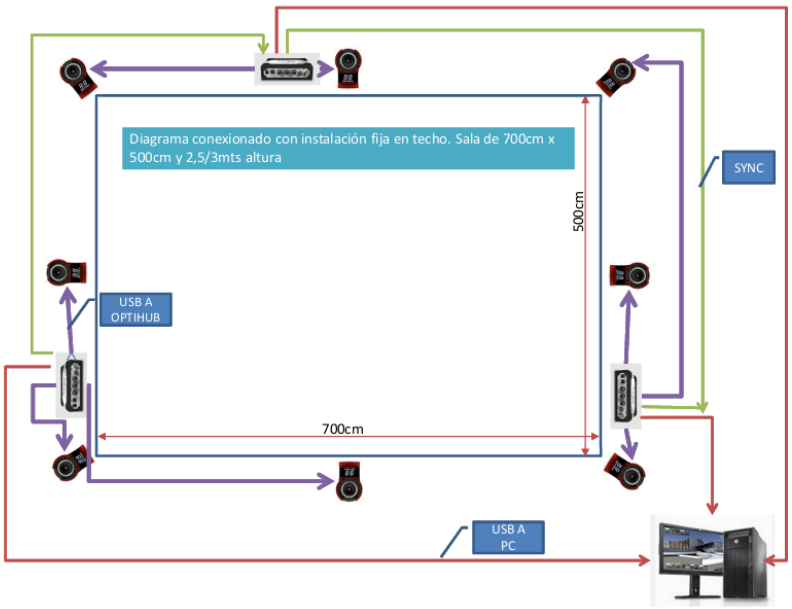


Figura 3.16: Escenario de las cámaras Flex 13 en la universidad

Una ventaja particular de estas cámaras es el hecho de que ya estén integradas en el laboratorio de la universidad, por lo que no sería necesaria la configuración inicial ni la preparación del escenario para poder utilizarlas (Figura 3.16).

## 3.8. Conjunto

Como se ha explicado, para la elección del hardware se han tenido en cuenta varios factores, como el tamaño, el consumo y el peso.

| **Componentes** | **Peso (g)** | **% Peso total** |
|---|---|---|
| Cube Orange + | 73 | 11.3 |
| TFmini-S | 5 | 0.77 |
| ESCs | 36 | 5.57 |
| PDB | 56.7 | 8.78 |
| Motores BLDC CrazePony (x1) | 56 | 8.67 |
| Motores BLDC Racerstar (x3) | 56 | 8.67 |
| ESP32-C3-DevKitM-1 | 10 | 1.55 |
| Hélices | 13 | 2.01 |
| Batería | 140 | 21.68 |
| Estructura | 200 | 30.97 |
| **Peso total:** | **645.7** | **100** |

Tabla 3.4: Distribución del peso total del dron por componente

Si se compara el peso total con el empuje de los motores para cada velocidad, para una velocidad de 1500$\mu$s, se proporcionaría un empuje total de 600g (el empuje de cada motor

es de aproximadamente 150g). Es decir, como para la velocidad media de los motores el empuje es aproximadamente similar al peso total del UAV, el dron se mantendría suspendido en el aire. El peso total es el idóneo, ya que se tiene el mismo margen de maniobra tanto para suministrar más como menos potencia.

| **Componentes** | **Peso (g)** | **Voltaje (V)** | **Corriente (A)** |
|---|---|---|---|
| Cube Orange + | 73 | 4–5.7 | 2.5 (nominal) |
| TFmini-S | 5 | 5±0.1 | ≤0.140 |
| ESCs | 5 | 7.4–14.8 (2S–4S) | 20 (nominal), 25 (máx) |
| PDB Maket Systems | 56.7 | 9–18 (3S–4S) | 2 (nominal), 2.5 (máx) |
| Motores BLDC CrazePony (x1) | 56 | 7.4–16.8 (2S–4S) | 10 (máx) |
| Motores BLDC Racerstar (x3) | 56 | 7.4–16.8 (2S–4S) | 11.8 (máx) |
| ESP32-C3-DevKitM-1 | 10 | 3.3-5 | 1(máx) |
| Batería Tattu | 140 | 11.1 | 45C |
| Batería Zeee | 140 | 11.1 | 120C |

Tabla 3.5: Peso, voltaje y corriente nominal de los principales componentes

# Capítulo 4

# Software

Los programas utilizados a lo largo del trabajo fueron los siguientes:

- QGroundControl (QGC): La estación de control.
- Matlab y Simulink, empleando los paquetes destinados a los UAV, en especial los PX4.
- Benewake_TF: requerido para la configuración del sensor de altura, el LiDAR TFmini-S.

## 4.1. QGroundControl

Como estación de control de un UAV, los programas más utilizados son QGroundControl (QGC) (en el caso de PX4 Autopilots) y MissionPlanner (en el caso de los ArduPilots). Dado que el Cube Orange+ es compatible tanto con PX4 como con ArduPilot, se podrían haber utilizado cualquiera de los 2 programas, pero finalmente se optó por QGC. QGC es la estación de control, un programa que permite monitorizar el estado de vuelo del UAV. Este programa permite configurar los distintos parámetros del PX4 Autopilot según las necesidades del dron, al igual que permite cargar un firmware completamente personalizado o subir uno autogenerado adaptado a los nuevos parámetros. QGC se utilizó para configurar distintos parámetros del microcontrolador, así como calibrar sus sensores y otros componentes del hardware, como los actuadores y la batería. Además, este programa permitió realizar los ensayos preliminares de sensores, como los integrados en el Cube Orange+, la velocidad de los motores y la comunicación por radio control con la emisora.

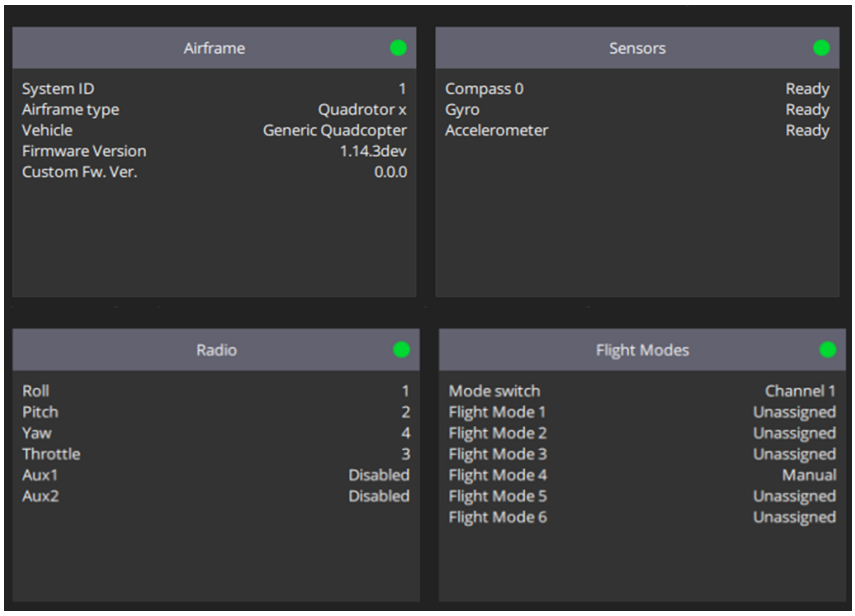


Figura 4.1: Calibración de sensores, actuadores y comunicaciones. QGroundControl.

Por ejemplo, se pueden ajustar parámetros del control que vienen por defecto en el firmware de PX4, ajustando las ganancias con las gráficas que se obtienen en tiempo real, tanto para medidas de velocidad, aceleración y de posición (véase la Figura 4.2).

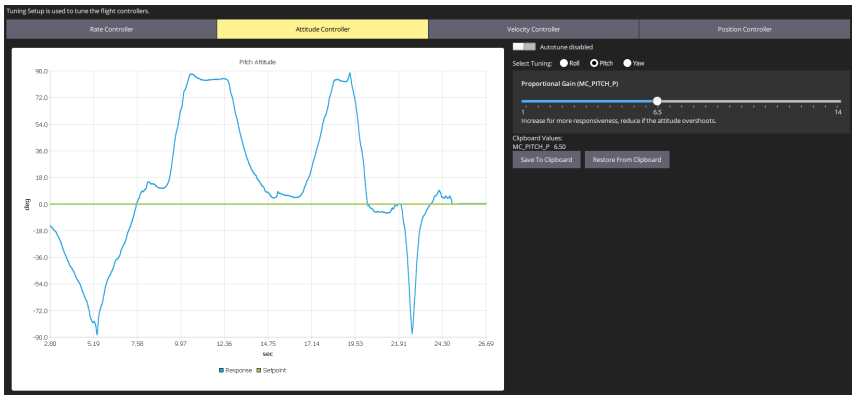


Figura 4.2: Pruebas sensores integrados. QGroundControl.

Desde QGC se puede limitar la velocidad de los actuadores, la frecuencia, y los puertos que se utilizarán. Se utilizó al inicio del proyecto, para verificar que tanto los motores como los ESC funcionaran correctamente.

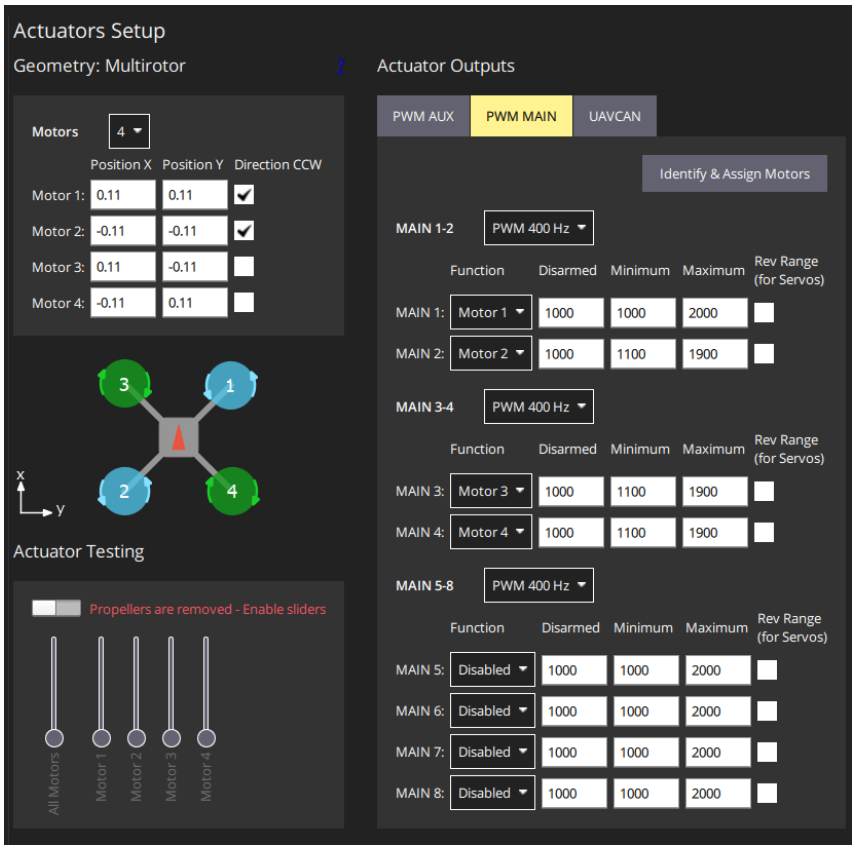


Figura 4.3: Actuadores. QGroundControl

Además, como se puede observar en la parte superior izquierda de la Figura 4.3, se debe indicar la distancia de cada motor al centro de masas del dron (donde se ubica el microcontrolador), además del sentido de giro de cada uno de los motores.

En proyectos anteriores, se utilizó BLHeliSuite para la calibración de los ESC. En este caso, al tratarse de un CubePilot, los ESC se pueden calibrar directamente desde QGC. La calibración ajusta los ESC para que su respuesta dentro del rango 1000-2000$\mu$s sea idéntica.

Desde la pestaña de parámetros también se pudo ajustar la velocidad máxima y mínima de los motores, además del "trim" de cada canal. En este caso, los canales siguen el siguiente orden: roll, pitch, throttle y yaw. El valor "trim" corresponde con el valor de reposo o neutro de cada canal. Lo habitual es configurar los canales correspondientes a la actitud (roll, pitch y yaw) con un valor de reposo de 1500$\mu$s (el joystick centrado). Por otro lado, el canal que corresponde al throttle tiene un valor de reposo de 1000$\mu$s, cuando el joystick está situado abajo del todo.

| | | |
|---|---|---|
| RC3_MAX | 2000.000 us | RC channel 3 maximum |
| RC3_MIN | 1000.000 us | RC channel 3 minimum |
| RC3_REV | Normal | RC channel 3 reverse |
| RC3_TRIM | 1000.000 us | RC channel 3 trim |
| RC4_MAX | 2000.000 us | RC channel 4 maximum |
| RC4_MIN | 1000.000 us | RC channel 4 minimum |
| RC4_REV | Normal | RC channel 4 reverse |
| RC4_TRIM | 1500.000 us | RC channel 4 trim |

Figura 4.4: Configuración de los canales de la emisora. QGroundControl

La emisora también se puede preparar en QGC. Al igual que puedes realizar los ajustes desde la emisora directamente, determinando el modo de vuelo (como se explicó anteriormente, el modo de vuelo habitual es el 2), la estación de control también permite calibrar los joysticks y los distintos canales disponibles, para verificar que su funcionamiento es el esperado (véase Figura 4.5).

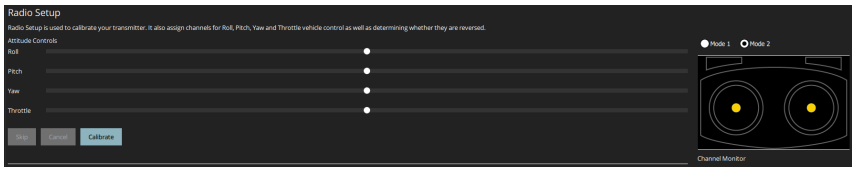


Figura 4.5: Configuración de la emisora. QGroundControl

Finalmente, QGroundControl permite el acceso a los mensajes MAVLink, pudiendo ver tanto mensajes predeterminados como personalizados, como se verá más adelante en el apartado de comunicaciones. Esto fue útil a la hora de depurar el control, ver el proceso de la máquina de estados y comprobar variables adicionales de interés.

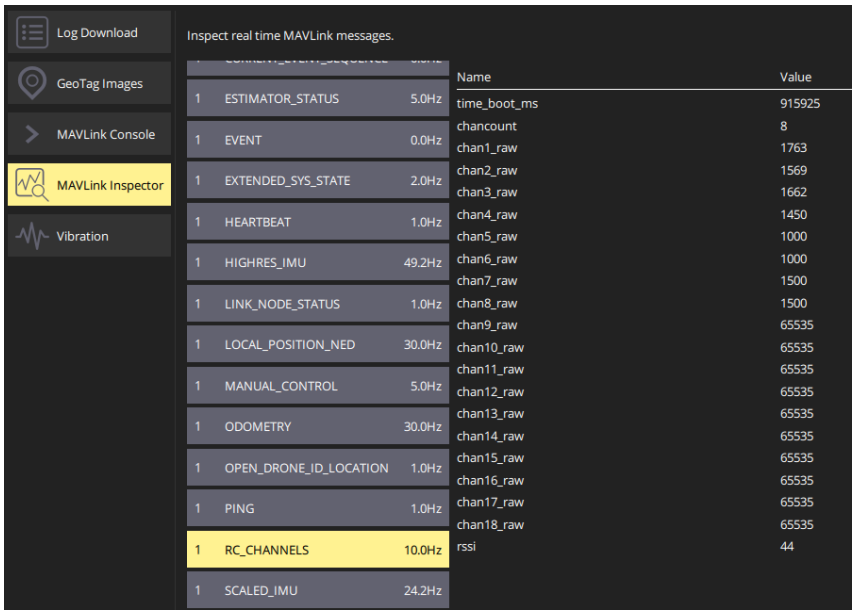


Figura 4.6: Mensajes MAVLink predeterminados. QGroundControl

## 4.2. Benewake_TF

Como se explicó anteriormente, el sensor TFmini-S funciona tanto con interfaz UART como I2C. La interfaz escogida fue I2C ya que este era el único dispositivo que se utilizaría en todo el proyecto con posibilidad de usar esta interfaz. Por este motivo, interesaba dejar los 2 puertos UART disponibles en el microcontrolador.

Para cambiar la configuración, se requirió del programa que ofrece Benewake, BW_TF. Este programa permite modificar la configuración del sensor, incluyendo la frecuencia de actualización, y la velocidad de transmisión en el caso del protocolo UART.

| Parameters | Command | Response | Remark | Default setting |
|---|---|---|---|---|
| Communication interface setup | 5A 05 0A MODE SU | 5A 05 0A 00 69 | 0 (UART)<br>1 (I2C) | UART |

Figura 4.7: Comando cambio de la interfaz TFmini-S

El programa sigue la interfaz mostrada en la Figura 4.8:

1. Selecciona el tipo de sensor Benewake. En este caso, el TFmini-S.
2. Selecciona el puerto serial del ordenador, aunque lo identifica automáticamente.
3. "Pix mode" modifica las unidades de distancia a metros, ya que inicialmente no está configurado de esta manera. "Device command" es donde se escriben los comandos para la configuración. En este caso, solo fue necesario un comando, para cambiar la configuración del interfaz de UART a I2C (véase la Figura 4.7).
4. Muestra las medidas en tiempo real, en las unidades especificadas. Una vez se cambie al protocolo I2C, no mostrará ninguna medida. Esto se debe a que I2C es un protocolo pasivo (esclavo), requiere de un maestro que le envíe una petición con una dirección determinada. En cambio, UART es "plug and play", muestra los datos sin necesidad de una solicitud o configuración adicional.

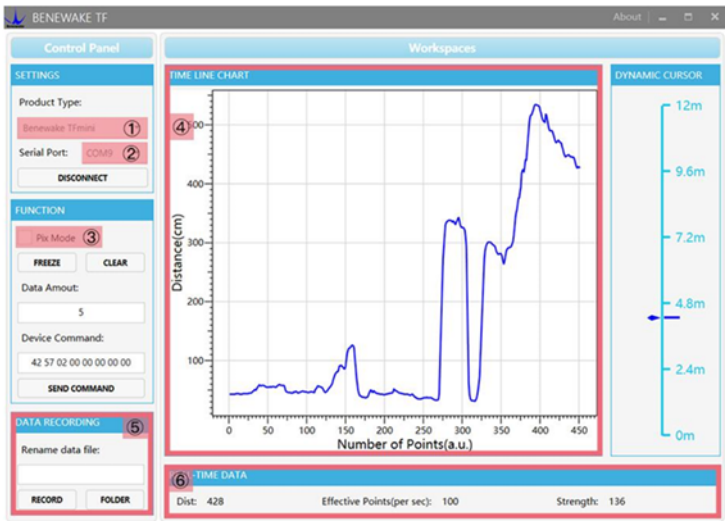


Figura 4.8: Programa Benewake_TF

## 4.3. Matlab y Simulink

Se ha utilizado Matlab y Simulink (versión R2024b) como programa principal para la programación del sistema de control. Para el proyecto fue necesario instalar 2 paquetes: UAV Toolbox y UAV Toolbox Support Package for PX4 Autopilots. Estos paquetes cuentan con las herramientas necesarias para acceder a la información de la emisora y la comunicación con los temas uORB internos del microcontrolador (uORB es un sistema de mensajería interno del PX4, el cual comunica procesos como el de los sensores y controladores).

### 4.3.1. Sensores integrados

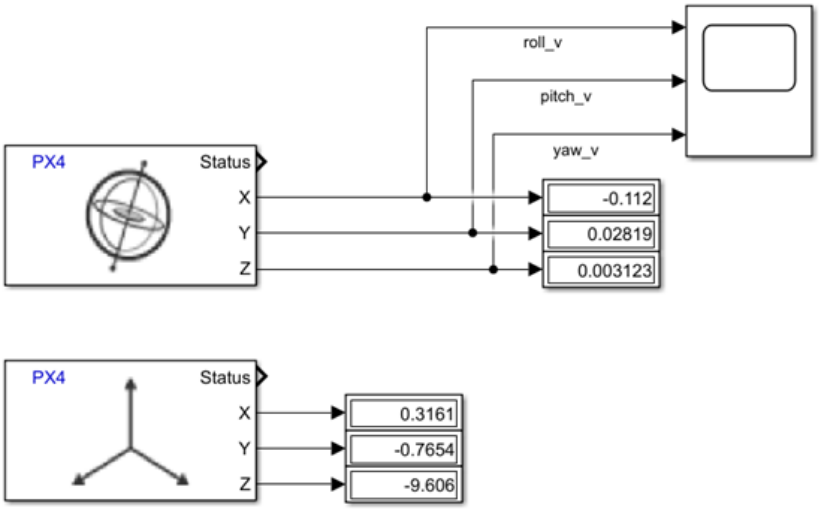


Figura 4.9: Mediciones sensores integrados microcontrolador. Simulink

Se utilizó el diagrama de bloques de la Figura 4.9 para comprobar los sensores integrados del microcontrolador. Los bloques a la izquierda de la figura son bloques uORB, y ya están configurados para que puedan acceder directamente a los tópicos sensor_accel y sensor_gyro.

El bloque superior es el giroscopio, el cual muestra la velocidad de giro en rad/s. El bloque inferior es el acelerómetro, el cual toma las medidas en m/s.

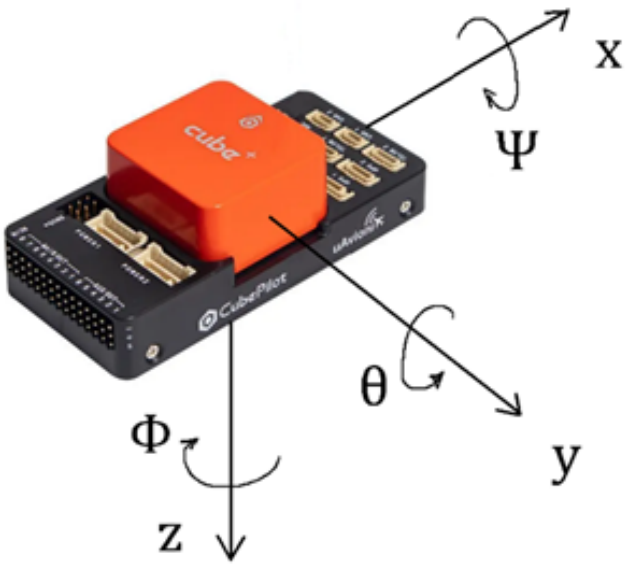


Figura 4.10: Sistema de ejes del Cube Orange+

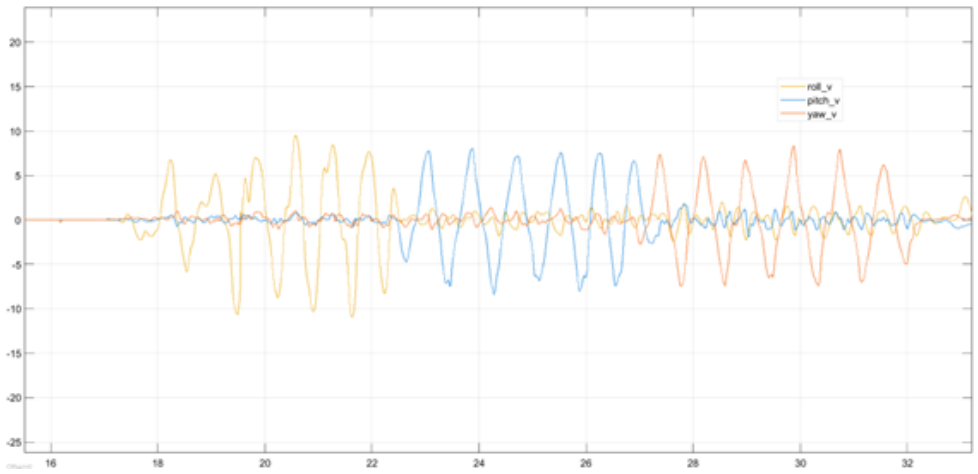


Figura 4.11: Gráfica con las medidas del giroscopio en tiempo real. Simulink

### 4.3.2. Tfmini-S

Para el sensor de altura, se tuvo especial cuidado en la configuración del bloque de lectura I2C y la configuración de Simulink. Siguiendo las indicaciones del datasheet [7], se ajustó la velocidad del bus a la máxima posible, 400kHz, aunque también es posible ajustarla a 100kHz.

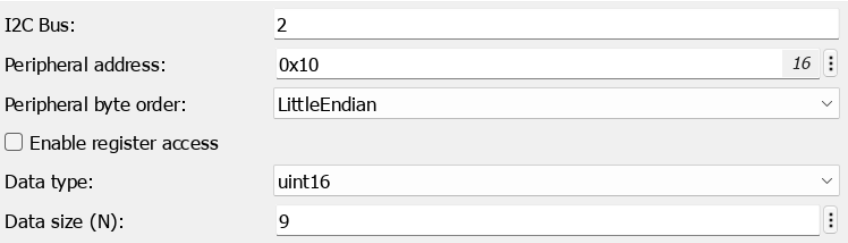


Figura 4.12: Configuración del bloque I2C Read. Simulink

Se configuraron el resto de los parámetros teniendo en cuenta que utiliza Little Endian, es decir, primero se guarda el byte menos significativo (LSB), su dirección es 0x10 y envía 9 bytes. Además, el bus correspondiente al sensor es el 2, ya que, como se explicó anteriormente, se conectó al puerto I2C 2 del cubo.

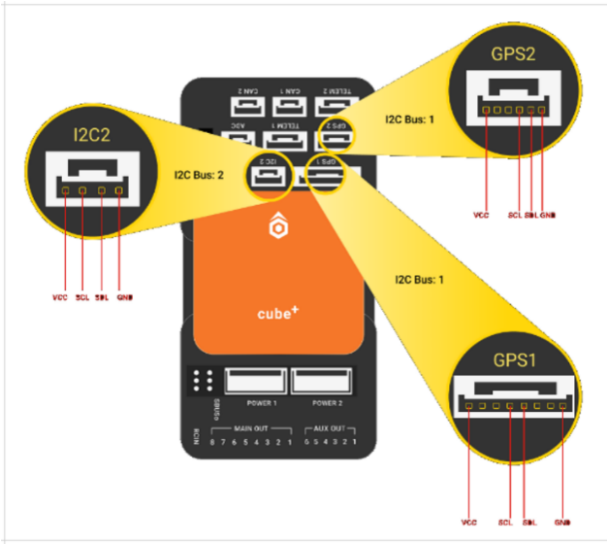


Figura 4.13: Buses I2C disponibles (Fuente: Matlab)

Las pruebas del sensor de altura se realizaron con un sencillo diagrama de bloques en Simulink. Se puede observar en la Figura 4.12 que se ajustó el Data Type como uint16. Esto se hizo porque, como los datos van en parejas (los 2 primeros son el encabezado, los 2 siguientes la distancia, etc) (Figura 4.14), se puede configurar directamente como uint16 en vez de uint8, de forma que cada par de bytes consecutivos se interprete directamente como un entero de 16 bits.

| Byte0 -1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 |
|---|---|---|---|---|---|---|---|
| 0x59 59 | Dist_L | Dist_H | Strength_L | Strength_H | Temp_L | Temp_H | Checksum |

Figura 4.14: Formato de los datos del TFmini-S. Simulink

En la Figura 4.15 se pueden observar 2 diagramas de bloques, el superior con la configuración uint8, y el inferior con la configuración uint16. Se puede observar con claridad que hacerlo de la segunda forma evita la necesidad que hacer la conversión para poder ver el valor de la distancia medida (unidades en mm).

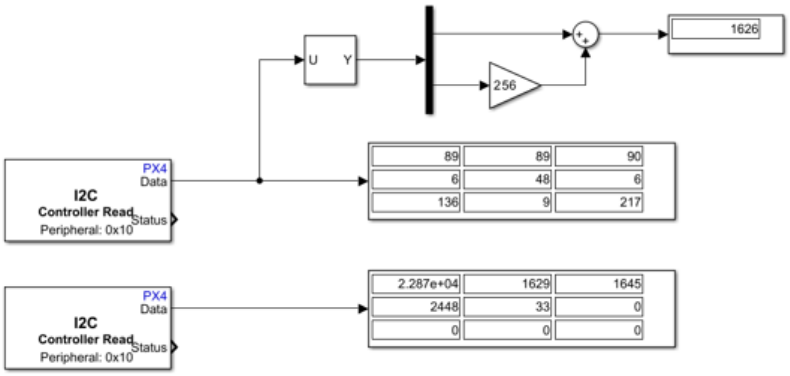


Figura 4.15: Mediciones TFmini-S. Simulink

### 4.3.3. Actuadores

Para realizar los ensayos de los motores, se utilizó un sencillo diagrama de bloques (véase Figura 4.16). Utilizando un "Slider Gain", se podía modificar el valor PWM del motor fácilmente y de forma precisa, permitiendo realizar las pruebas sin dificultades.

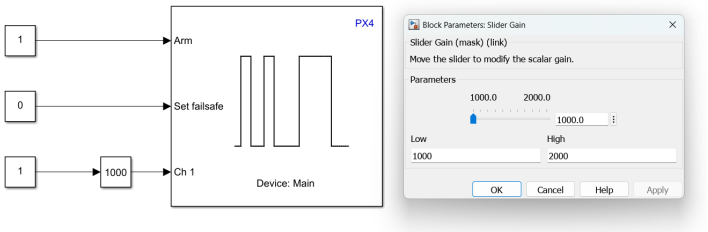


Figura 4.16: Diagrama de bloques de los actuadores. Simulink

El bloque PWM permite la conexión de hasta 8 motores, aunque en este caso, al tratarse de un cuadricóptero, solo activaremos 4. Las entradas arm y set failsafe son necesarias para mantener la seguridad. Para que los motores respondan a los comandos recibidos, deben estar armados, es decir, deben recibir un '1' lógico en la entrada. Failsafe es una medida adicional en caso de que se pierda la comunicación, para evitar que el vehículo se descontrole.

### 4.3.4. Emisora

Si bien se puede ajustar el efecto que tiene cada posición del joystick directamente desde la emisora, esta solución no es la más conveniente, ya que entonces se depende de una emisora en particular para que el control funcione como se espera. La mejor manera de ajustar la potencia de los motores según la posición de los joysticks es mediante los bloques "1-D Lookup Table".

Como se puede observar en la Figura 4.17, primero se extrajeron los valores de la emisora, a los cuales se pudo acceder directamente aprovechando un bloque "uORB Read", el cual permite acceder al mensaje `RcChannels` que lee los valores de la emisora en pu. Posteriormente, se ajustaron a la escala de $\mu$s (1000-2000). En el caso del Roll, Pitch y Yaw (sus canales correspondientes, el 1, 2 y 4), la conversión es la misma, ya que el rango

de valores del joystick se encuentra entre [-1, 1], mientras que en el caso del Throttle (su canal correspondiente, el 3), varía entre [0, 1]:

$$PWM[\mu s] = RcChannels(1, 2, 4) * 500 + 1500 \tag{4.1}$$

$$PWM[\mu s] = RcChannels(3) * 1000 + 1000 \tag{4.2}$$

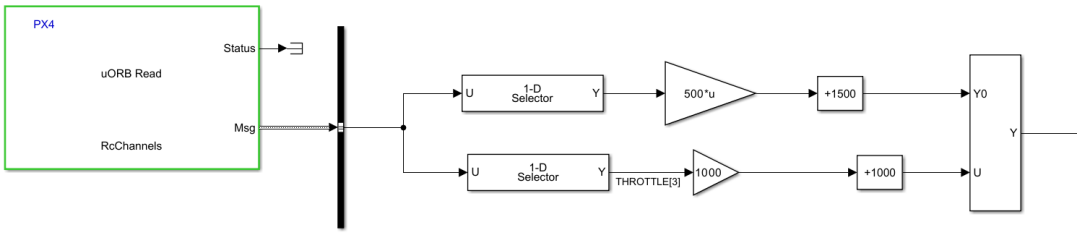


Figura 4.17: Conversión de los comandos de la emisora. Simulink

En la Figura 4.18 se pueden ver las 4 gráficas correspondientes a Roll, Pitch, Throttle y Yaw, respectivamente, donde Throttle es la única distinta a las demás.

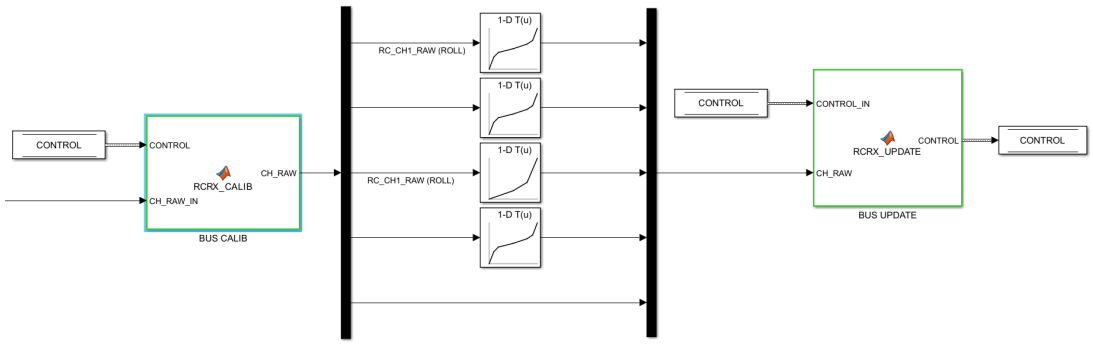


Figura 4.18: Ajuste del comportamiento de los motores según la posición de los joysticks. Simulink

Las gráficas se han ajustado de manera que el desplazamiento del joystick no provoque un cambio brusco en la velocidad de los motores, y solo se pueda alcanzar velocidades más altas (a partir de 1800$\mu$s) si se mueve el joystick a los extremos.

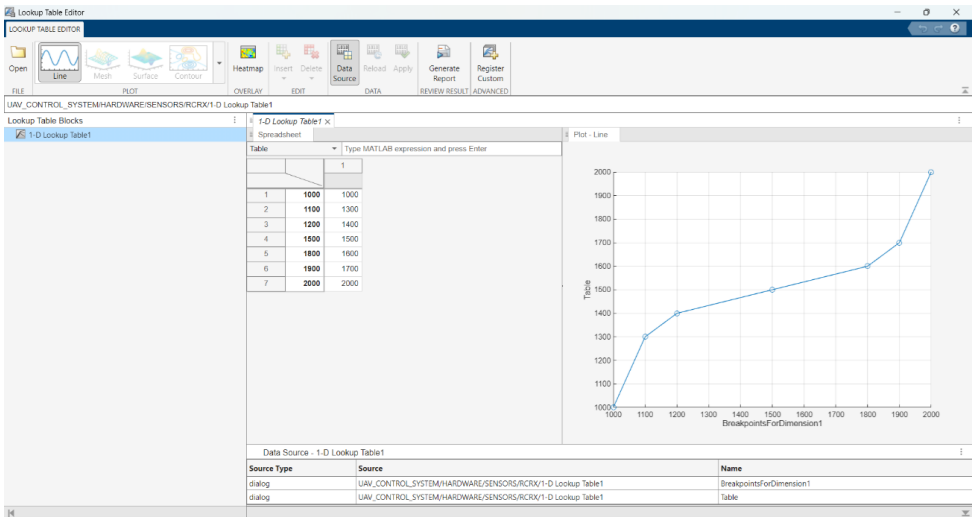


Figura 4.19: Ajuste del comportamiento de Roll, Pitch, Yaw. Simulink

El comportamiento del Throttle es distinto, como ya se explicó anteriormente, debido a que el rango de valores oscila entre 0 y 1. Como se puede observar en la figura, no es hasta pasada la mitad de dicho rango que la velocidad empieza a aumentar de forma significativa. De esta manera, el dron es mucho más fácil de controlar y se evitan incrementos bruscos de velocidad especialmente durante el despegue.

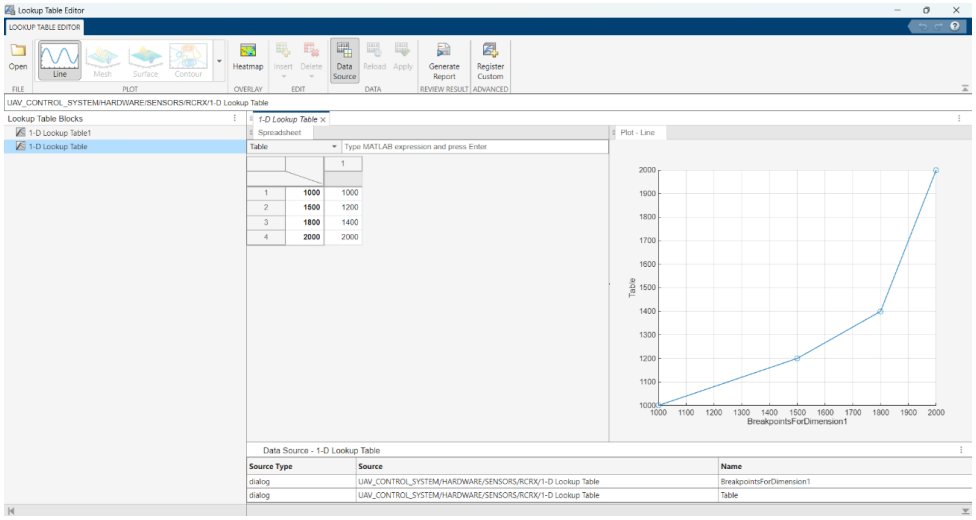


Figura 4.20: Ajuste del comportamiento de Throttle. Simulink

Las gráficas se fueron ajustando, utilizando un mensaje personalizado MAVLink, el cual mostraba los valores reales del joystick y los valores resultantes a la salida de las gráficas.

### 4.3.5. Configuración de Seguridad

Como medidas de seguridad, se utilizaron una palanca de la emisora y el safety switch. De esta manera, si el firmware estaba descargado en el Cube Orange+ y la batería estaba conectada, los motores no girarían por accidente.

El marco azul muestra la parte del diagrama de bloques que lee el canal 5 de la emisora. Solo si la palanca se encuentra en la posición precisa, se podrán armar los motores y desactivar el failsafe, siempre que se cumplan las otras condiciones.

El marco verde muestra la parte del diagrama que lee el botón de seguridad, el cual se encuentra conectado al puerto GPS1 del microcontrolador. En caso de que se mantenga pulsado durante unos instantes, enviará una señal de valor 1 a la entrada de armado de los motores y un 0 al failsafe.

El sensor de altura se planteó como una medida adicional, aunque no necesaria. Si la lectura del sensor es superior a 10cm, entonces los motores no podrán funcionar aunque las otras 2 situaciones se cumplan. Cada condición está incorporada a un puerto AND, de forma que solo cuando se cumplan las 3 condiciones, los motores puedan girar.

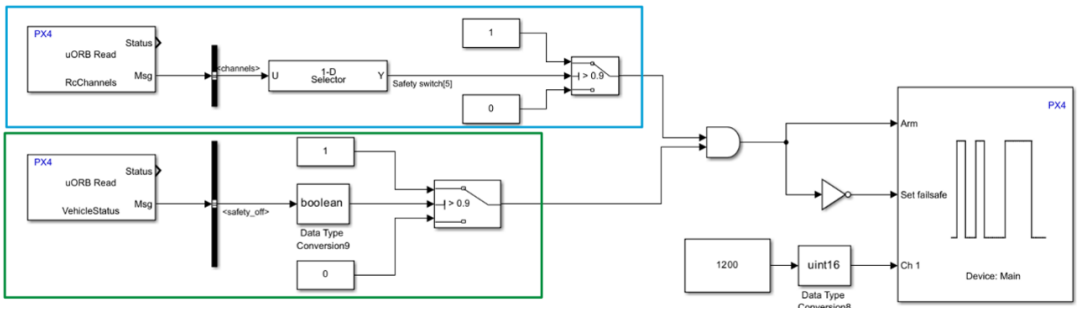


Figura 4.21: Diagrama de bloques simplificado sistema de seguridad. Simulink

## 4.4. RealTerm y VSCode

Para la configuración del ESP32-C3, se requirió de las herramientas RealTerm y Visual Studio Code, para verificar el correcto funcionamiento de la telemetría.

RealTerm se utilizó para verificar que, en efecto, el puerto TELEM1 del Cube Orange+ enviaba la información de forma continuada, y sin necesidad de una configuración adicional. Por otro lado, VSCode se utilizó para realizar un sencillo código y confirmar que el ordenador se conectaba correctamente mediante UDP al ESP32-C3. Aunque nada parecía indicar que no lo hiciera, se fueron depurando todos los posibles errores debido a que QGroundControl no conseguía recibir mensajes MAVLink por telemetría.

Este código lo que hace es esperar la llegada de paquetes UDP. Una vez recibido el mensaje MAVLink (cuyo tamaño máximo puede ser de 256 bytes si es MAVLink v1, o 279 bytes en caso de tratarse de la MAVLink v2), lo procesa, analizando que se trate de un mensaje MAVLink mediante el header, el cual puede ser 0xFE (MAVLink 1) o 0xFD (MAVLink 2) (véase 4.1).

```
import socket

# Configuracion del socket UDP
UDP_IP = "0.0.0.0"  # Escucha en todas las interfaces
UDP_PORT = 14550    # Puerto usado por DroneBridge/
   QGroundControl

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

print(f"Escuchando MAVLink en UDP puerto {UDP_PORT}...\n")

try:
    while True:
        data, addr = sock.recvfrom(280)
        print(f"Paquete recibido de {addr}: {len(data)} bytes")
        if data[0] == 0xfe or data[0] == 0xfd:
            print("Paquete MAVLink valido (inicio 0xfe o 0xfd)\
               n")
        else:
            print("No parece un paquete MAVLink\n")
except KeyboardInterrupt:
    print("Interrumpido por el usuario")
finally:
    sock.close()
```

Código 4.1: Código de prueba comunicación MavLink

# Capítulo 5

# Comunicaciones

**uORB:** es la comunicación interna del microcontrolador, permite la comunicación de los distintos módulos dentro de PX4.

**MAVlink:** Micro Air Vehicle Link: protocolo de comunicación para los UAVs y las estaciones de control. Permite la comunicación por telemetría entre tiempo real, para misiones y comandos.


Figura 5.1: Esquema de las comunicaciones

## 5.1. uORB

Micro Object Request Broker (uORB) es un middleware necesario en el sistema de control de vuelo de código abierto PX4. Es, en resumidas cuentas, la comunicación interna del microcontrolador, al ser responsable de la transmisión de información entre sus diferentes módulos. uORB implementa la comunicación entre procesos (IPC) y tareas mediante memoria compartida y logra un intercambio de datos de baja latencia, manteniendo un consumo de memoria óptimo.

Existen 2 roles dentro del bus uORB: suscriptores y publicadores (subscriber, publisher). Un tópico (topic) contiene un nombre e información sobre dicho nombre (metadatos). Cada tema admite varias instancias, donde la frecuencia del anunciante es la tasa de muestreo.

Los tópicos pueden ser de 2 tipos: de notificación o generales. En PX4 la mayoría se tratan como generales. Cuando una aplicación se suscribe a un tema de notificación, recibe la información más reciente del nodo, considerándola como estado actual. Si, por el contrario, una aplicación se suscribe a un tema general, solo recibirá la información publicada en la siguiente interrupción, sin tener en cuenta la información publicada anteriormente.

Cada tópico publicado o suscrito corresponde a un nodo, donde los suscriptores y anunciantes comparten datos a través de un búfer circular interno. En un nodo, el publicador escribe la información para publicarla, mientras que el suscriptor se suscribe para leerla.

uORB sigue un diseño sin bloqueos, es decir, ni el anunciante se detiene esperando a que un suscriptor consuma el dato, ni el suscriptor necesita reservar un espacio en la memoria hasta tener tiempo para leer la información. Esta información se va actualizando y sobrescribe la información anterior incluso si el suscriptor no llegó a leerla, siguiendo un protocolo FIFO (First In, First Out). De esta manera, cada rol es independiente y se evita un consumo de memoria innecesario.

El búfer es circular ya que, cuando éste se encuentra lleno pero se realiza una nueva escritura, los datos más antiguos se sobrescriben. La longitud de la cola puede ser igual o mayor que 1, y se fija en la definición del tópico. En caso de que la longitud de la cola sea mayor que 1, el suscriptor puede leer únicamente la información más reciente, o leer información anterior en caso de necesitarlo.

## 5.2. MAVLink

Micro Air Vehicle Link (MAVLink) es un protocolo de comunicación empleado en vehículos no tripulados, principalmente drones. Un mensaje de MAVLink ('msg') es un flujo de bytes que han sido codificados por la estación de control (GCS), en este caso QGroundControl. El transporte de mensajes MAVLink se puede hacer de varias formas distintas, como conexión USB y por telemetría (UDP, TCP), y permite la comunicación bidireccional entre la estación de control y el dron.

MAVLink es un protocolo de comunicación que puede seguir un modelo publicador/suscriptor con comunicación asíncrona o punto a punto con comunicación síncrona. Los tópicos de MAVLink se usan para el envío de información general, como los sensores del microcontrolador y el HEARTBEAT. Al ser información de la que no se espera respuesta, los suscriptores reciben los mensajes de manera asíncrona, lo que permite una actualización constante.

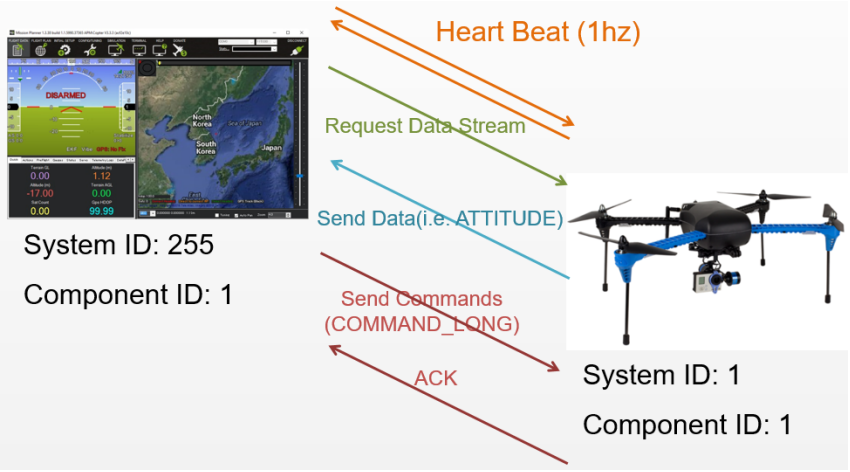


Figura 5.2: Esquema de la comunicación MAVLink (Fuente: Ardupilot)

Por otro lado, los subprotocolos se utilizan para la transmisión de información concreta de la que sí se espera una respuesta por parte del receptor, como es el caso de los protocolos de misión y de parámetros. Por este motivo, su comunicación es punto a punto.

Cada mensaje de MAVLink sigue la misma estructura, permitiendo enviar y analizar paquetes de información de manera consistente. Los mensajes se pueden modificar cambiando sus IDs. Un ID le da un significado específico al mensaje, es decir, es como si fueran

las palabras del sistema MAVLink. Cada mensaje contiene un número que representa qué mensaje de MAVLink fue enviado, y todos los mensajes tienen un propósito/significado objetivo.

Por ejemplo, el mensaje cuyo ID= 0 es el HEARTBEAT. Cuando QGC recibe un mensaje del dron cuyo ID es 0, sabe que el dron mandó un mensaje heartbeat, y que por tanto, el dron sigue activo.

Aunque todos los mensajes tienen la misma estructura, la información contenida dependerá de sus ID. MAVLink ha estandarizado el protocolo de comunicación para el envío de mensajes y los tipos de mensajes que pueden enviarse.

| Byte Index | Content | Value | Explanation |
|---|---|---|---|
| 0 | Packet start sign | v1.0: 0xFE (v0.9: | Indicates the start of a new packet. |
| 1 | Payload length | 0 - 255 | Indicates length of the following payload. |
| 2 | Packet sequence | 0 - 255 | Each component counts up his send sequence. Allows to detect packet loss |
| 3 | System ID | 1 - 255 | ID of the SENDING system. Allows to differentiate different MAVs on the same network. |
| 4 | Component ID | 0 - 255 | ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot. |
| 5 | Message ID | 0 - 255 | ID of the message - the id defines what the payload "means" and how it should be correctly decoded. |
| 6 to (n+6) | Data | (0 - 255) bytes | Data of the message, depends on the message id. |
| (n+7) to (n+8) | Checksum (low byte, high byte) | ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables). | |

Figura 5.3: Formato de los mensajes MAVLink (Fuente: Ardupilot)

Cada mensaje de MAVLink cuenta con un header de 6 bytes, y el tamaño del mensaje puede variar entre 8 a 263 bytes, según su ID.

**Byte 0:** El primer byte de un mensaje MAVLink es 0xFE (en caso de MAVLink 1) o 0xFD (en caso de MAVLink 2). El mensaje será ignorado si no es el caso correspondiente.

**Byte 1:** Indica la longitud de la información enviada, que como máximo puede ocupar 256 bytes.

**Byte 3:** El byte 3, o "System ID" es básicamente el IP del sistema que envía el mensaje. En el caso de la estación de control normalmente es 255, mientras que en el caso del dron suele ser 1.

**Byte 5:** El byte 5 es el "message ID", es decir, indica el tipo de mensaje que se recibe, permitiendo que el nodo receptor analice la información de la forma correspondiente. Este byte valdría 0 en el caso de un mensaje HEARTBEAT.

MAVLink 2 es la versión mejorada, aumentando el tamaño del ID (24 en vez de 16bits) y la posibilidad de firmar mensajes, aumentando así la seguridad de la transmisión de los datos y garantizando su autenticidad.

Creamos mensajes MAVLink (ficheros .msg) utilizando un comando disponible en Matlab, *createPX4uORBMessage* (Figura 5.1). Estos mensajes se pueden llamar en bloques llamados uLog, que son los ficheros donde se almacenan las variables que quieres guardar (Figura 5.4). A estas variables se puede acceder vía MAVLink, tanto en tiempo

real, de forma inalámbrica o por USB, como posteriormente, extrayendo la tarjeta SD donde se almacena dicha información (5.2).

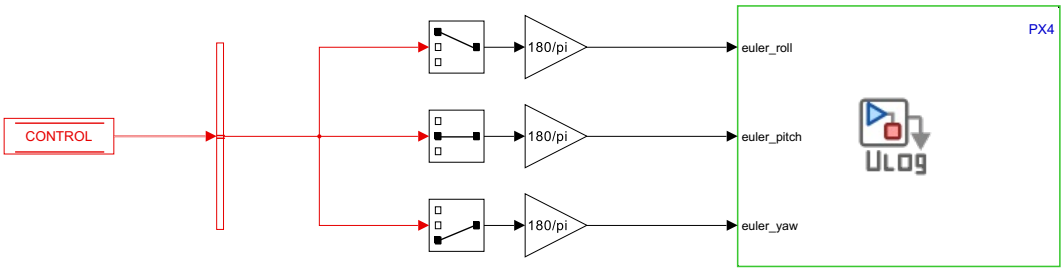


Figura 5.4: Ajuste del comportamiento de Roll, Pitch, Yaw. Simulink

Un principal problema que tuvimos durante el diseño y depuración del control en Matlab fue el hecho de que no podíamos ejecutar la función "Monitor & Tune", debido a todos aquellos bloques que requerían de MAVLink. Aunque la comunicación MAVLink se puede habilitar o no en Matlab, no seleccionar dicha opción implicaría que habría información que se perdería y partes del control no funcionarían correctamente. Para poder depurar el control, y verificar que las variables respondían como era de esperar, utilizamos la consola de MAVLink disponible en QGroundControl y los bloques uLog, para comprobar que las variables adquirían el valor esperado en tiempo real. Esto fue especialmente útil para 2 cosas: la máquina de estados y los ajustes de la emisora.

```
uint64 timestamp    #time since system start (microseconds)
float64 euler_roll
float64 euler_pitch
float64 euler_yaw
```

Código 5.1: Construcción de un mensaje MAVLink personalizado. Guarda la orientación del Cube Orange+

```
nsh> logger status
INFO  [logger] Running in mode: all
INFO  [logger] Number of subscriptions: 182 (5824 bytes)
INFO  [logger] Full File Logging Running:
INFO  [logger] Log file: /fs/microsd/log/sess228/log100.ulg
INFO  [logger] Wrote 3.40 MiB (avg 62.73 KiB/s)
INFO  [logger] Since last status: dropouts: 0 (max len:
   0.000 s), max used buffer: 42312 / 65536 B
nsh> listener euler

TOPIC: euler
 euler
    timestamp: 60624736 (0.006662 seconds ago)
    euler_roll: -0.425088
    euler_pitch: 5.212629
    euler_yaw: 2.042091
nsh> listener euler

TOPIC: euler
 euler
    timestamp: 67184614 (0.004369 seconds ago)
    euler_roll: -3.973997
    euler_pitch: 3.377839
    euler_yaw: -46.091161
```

Código 5.2: Mensaje MAVLink, muestra la orientación del Cube Orange+ en tiempo real. QGroundControl.

## 5.3. Radio Control: Protocolo SBUS

Serial Bus (SBUS) y Modulación por Posición de Pulso (PPM) son protocolos usados en sistemas de Radio Control (RC), para transmitir información del transmisor (la emisora) al receptor. El protocolo PPM es analógico, es decir, la información se envía mediante varios pulsos, uno detrás de otro, empezando por el canal 1 hasta llegar al último de todos. Esto hace que no sea un protocolo rápido, ya que, hasta que no haya llegado al último canal, no puede empezar el recorrido de nuevo y enviar nueva información. Esto se traduce en que es un protocolo con mayor latencia y menor resolución.

El protocolo SBUS, en cambio, es digital, y permite la transmisión de información de varios canales con un único cable, el intercambio de información en los 2 sentidos, y una mayor resolución. Al ser digital, este protocolo es más preciso y rápido que PPM, siendo el protocolo más utilizado en el control de vuelo.

En este proyecto, para el vuelo manual se utilizó el protocolo SBUS. Ambos protocolos necesitan 3 cables para conectar el receptor al microcontrolador: fuente (5V), tierra (GND) y señal.

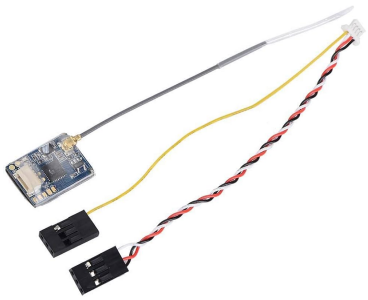

Figura 5.5: Receptor FS-A8S con el cableado necesario (Fuente: Amazon)

## 5.4. ESP32-C3

La placa ESP32-C3-DevKitM-1 es una placa que integra funciones WiFi y Bluetooth. En este proyecto se utilizó la placa para la comunicación inalámbrica entre el CubePilot y la estación de control, con el propósito de recibir mensajes MAVLink en tiempo real durante el vuelo y poder realizar un análisis del control de manera inmediata.

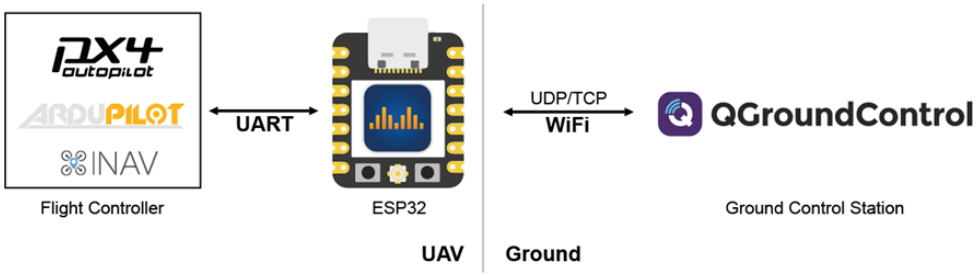


Figura 5.6: Esquema de la comunicación inalámbrica entre el dron y la estación de control (Fuente: PX4 Autopilot)

Transmission Control Protocol (TCP) y User Datagram Protocol (UDP) son protocolos para el envío de datos a través de internet. TCP se caracteriza por ser más segura y fiable que UDP, al cerciorarse de que la información ha llegado a su destino correctamente y reenviándola en caso contrario. Si bien UDP es más propenso a pérdidas de información, este protocolo es el más habitual para situaciones donde la velocidad es crucial. Aunque es menos fiable, es más rápido y directo, por lo que se utiliza en casos donde hay un flujo de datos más continuo. Por este motivo, UDP es el protocolo ideal para telemetría.

Además, por defecto PX4 utiliza el protocolo UDP para la comunicación MAVLink con la estación de control (QGroundControl en este caso). El puerto correspondiente para la comunicación con la estación de control es el 14550. Que los datos se envíen sin necesidad de "handsake" entre el emisor y el receptor reduce la sobrecarga.

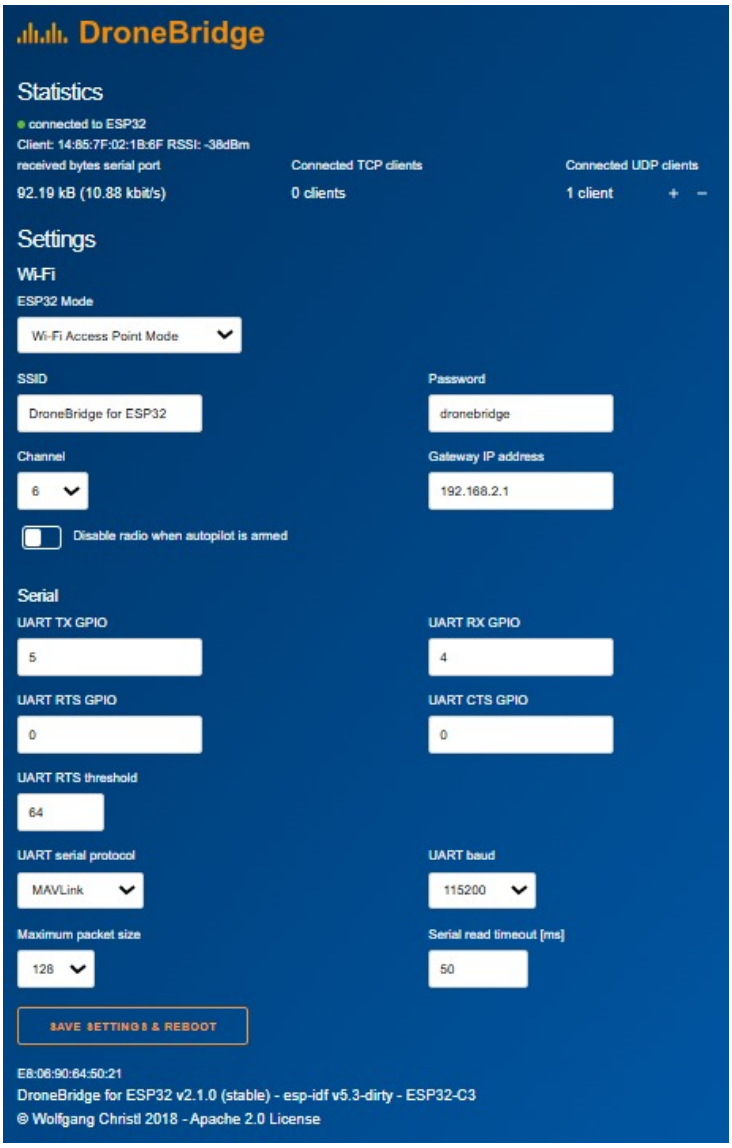


Figura 5.7: Configuración del firmware DroneBridge

La placa ESP32-C3-DevKitM-1 integra funciones WiFi y Bluetooth, con poca demanda de energía.

La descarga del firmware necesario para las funciones WiFi se realizó de manera sencilla mediante la plataforma oficial DroneBridge. Después, bastaba con conectar el PC a la red WiFi de este dispositivo. Una vez hecho esto, se podía configurar la placa, como se puede observar en la Figura 5.7.

El modo de comunicación fue AP (Access Point): el ESP32-C3 emite su propia red, lo que permite que el PC se conecte directamente. QGroundControl puede conectarse al dron directamente, sin necesidad de routers externos o hardware adicional.

Otro modo disponible es el STA (Station), donde es el ESP32-C3 el que actúa como cliente, conectándose a una red Wi-Fi existente.

También existe el modo STA/AP, donde puede funcionar en los 2 modos citados de forma simultánea, lo que le permite conexión a Internet mientras actúa como punto de acceso para otros dispositivos.

Se escogió el puerto TELEM1 ya que tiene habilitado MAVLink por defecto. Por otro lado, se ajustó la configuración del baudrate a 115200 bps, tanto en el ESP32-C3 como

en QGroundControl. Se cambió de 57600 a 115200 para duplicar el ancho de banda y asegurar que la telemetría MAVLink no fuera recortada.

En caso de que en un futuro se utilice una radio telemetría SiK, se recomienda utilizar el puerto TELEM1, ya que es el puerto estándar, con su configuración original de 57600 bps. Por tanto, habría que mover el ESP32-C3 al puerto TELEM2, y configurarlo para comunicación MAVLink. La radio telemetría es utilizada para comunicaciones de larga distancia y, dado que el dron en el que se ha trabajado es para vuelos interiores, la comunicación Wi-Fi era suficiente.

Se configuró como punto de acceso fijo en el canal 6, para evitar solapamiento entre los espectros de otros canales. Los canales más habituales con este propósito son el 1, el 6 y el 11. Fijar un canal asegura que cada vez que se inicie, seguirá en esa frecuencia, evitando que la placa cambie de canal en caso de interferencias. Si el punto de acceso cambia de canal en pleno vuelo, la estabilidad y el envío de información o comandos pueden verse comprometidos.

No se utilizaron los pines de RTS y CTS, por lo que se inhabilitaron asignándoles el valor 0. Por otro lado, se mantuvo la asignación de los pines TX y RX de la placa predeterminada, tal y como se explicó en el apartado de Hardware. Los pines CTS y RTS sirven para implementar un control de flujo de hardware. Se suele utilizar en el caso de comunicación UART, cuando un dispositivo es más lento en el procesamiento de datos que otro. En este caso, al estar ajustado para una velocidad y telemetría estándar, no hay riesgo de pérdida de bytes.

# Capítulo 6

# Sistema de Control

En este apartado se explicará brevemente la estructura seguida en el sistema de control, centrado en las partes más relacionadas con este proyecto. El fichero de Simulink, llamado UAV_CONTROL_SYSTEM, consta de varios subsistemas (véase Figura 6.1), de entre los cuales se tratará Control en este capítulo.

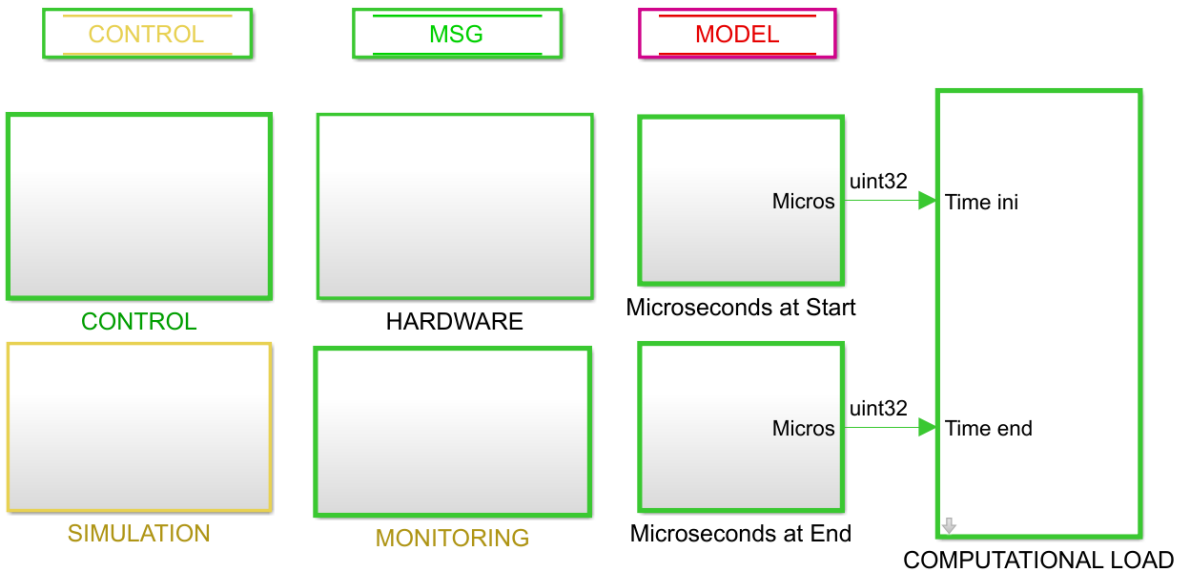


Figura 6.1: UAV_CONTROL_SYSTEM. Simulink

## 6.1. CONTROL

Dentro del subsistema de Control, se encuentran:

**Local Targets:** los controladores siguen unas referencias generadas en este bloque, como las referencias de actitud, de navegación y de misión, las cuales se generarán o no en función del modo de funcionamiento y el estado en el que se encuentre el UAV.

**State Estimator:** Estima el estado del UAV, mediante los distintos sensores del aparato. Extrae información bruta sobre la actitud, la posición y la aceleración del dron, para utilizarla después en los controladores.

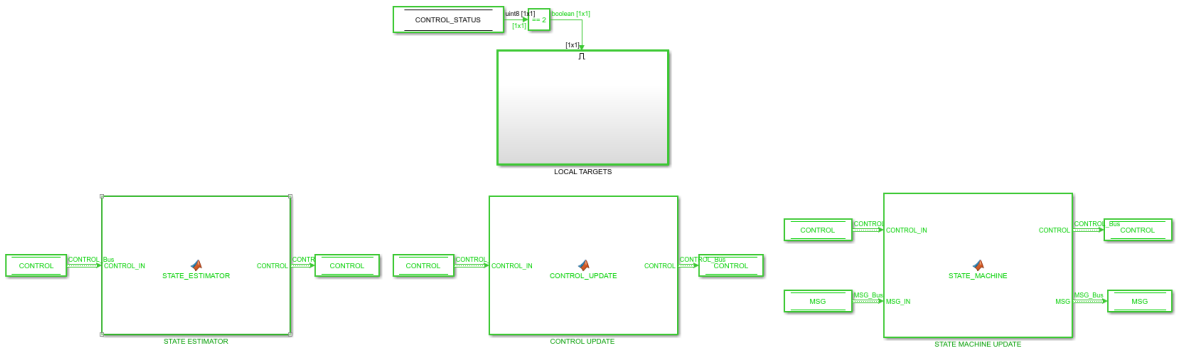


Figura 6.2: Subsistema de Control. Simulink

**Control Update:** Es el bloque principal del sistema de control. Consta de un controlador por realimentación de estados y 2 filtros EKF (Filtro Extendido de Kalman):

- El estimador de actitud calcula los ángulos de Euler a partir de las mediciones de los sensores del CubePilot (sus giroscopios, acelerómetros y su magnetómetro).
- El estimador de navegación es el encargado de, a partir de otros sensores como el LiDAR, estimar la velocidad y la posición del vehículo.

**State Machine Update:** Los estados por los que pasa el control son los siguientes:

0. BOOTING: El sistema se inicia, duración de 3s.
1. SENSOR CALIBRATION: Los sensores integrados se calibran, fijando los ángulos de Euler a 0 para la posición inicial y determinando los offset de los sensores.
2. READY: Una vez se hayan completado los 2 estados anteriores, el aparato está listo para ser armado.
3. DISARMED MOTORS: Se accede a este estado una vez se haya mantenido pulsado el botón de seguridad. Este estado se alcanza desde cualquier modo de vuelo (una vez los motores hayan sido armados con éxito, evidentemente), utilizando un canal de la emisora que se programó exclusivamente para ello (la palanca C, véase Figura 6.3).
4. ARMING MOTORS: Comienza el proceso de armado de los motores. La condición establecida para alcanzar este estado es que el Thrust $<0.05$ y Yaw $>0.9$, es decir, que el joystick izquierdo apunte abajo a la derecha.
5. ARMED MOTORS: Una vez los motores estén armados, el UAV está listo para volar en el modo de vuelo establecido.
6. RC Flight: El dron se controla mediante la emisora.
7. Altitude Control: El dron se mantiene estable en el eje vertical haciendo uso de sensores como el barómetro y el sensor de altura.
8. Navigation Control: En este estado, el dron controla tanto su posición en el eje vertical como en el resto de ejes, es decir, es capaz de ajustar tanto su posición como su orientación.

El objetivo inicial era ser capaces de alternar entre los distintos modos de vuelo utilizando los canales de la emisora. Por cuestión de tiempo, no se pudo implementar el vuelo autónomo.

Una vez el sistema se encuentra en el estado Ready, se debe pulsar el Safety Switch conectado al dron para pasar al estado 3, Disarmed motors. Moviendo el joystick izquierdo

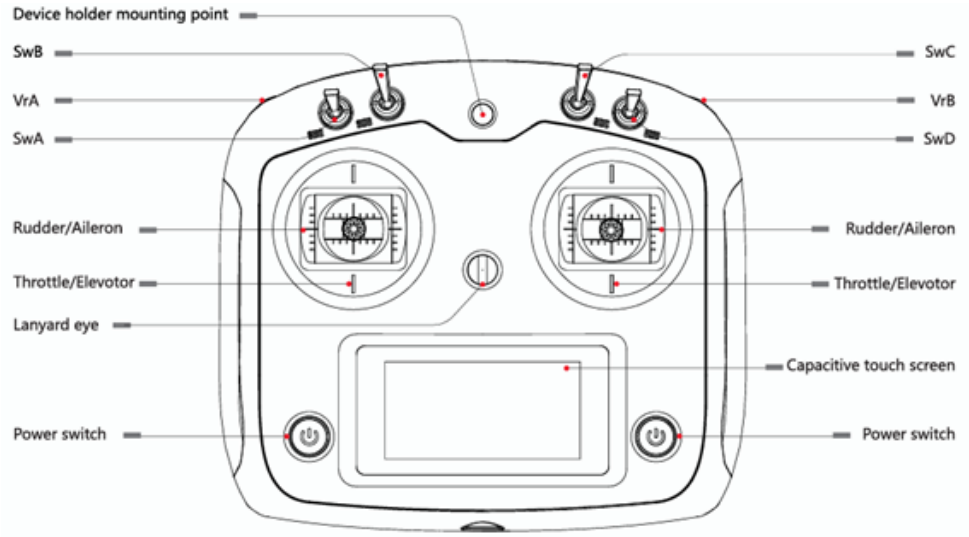


Figura 6.3: Emisora (Fuente: FlySky)

abajo a la derecha, y manteniéndolo unos instantes, el dron pasará al estado 4, y avanzará automáticamente después al estado 5, Armed motors. A partir de este momento, se podrá controlar el dron con la emisora en el caso de que se haya configurado el modo RC Flight.

# Bibliografía


[1] J. G. Aguilar, «Desarrollo de sistemas de navegación autónoma en interiores para un UAV,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, Madrid, 2018. dirección: https://repositorio.comillas.edu/xmlui/handle/11531/14540.

[2] N. G. García, «Control de un cuadricóptero para navegación en interiores usando un sensor de flujo óptico,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, Madrid, 2016. dirección: https://repositorio.comillas.edu/xmlui/handle/11531/22629.

[3] J. L. Castillo, «Sistema de navegación autónoma para un cuadricóptero en exteriores,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia de Comillas, 2022. dirección: https://repositorio.comillas.edu/jspui/bitstream/11531/22632/1/TFG-LoringCastilloJaime.pdf.

[4] J. J. B. Vázquez, «Control de navegación autónoma de un cuadricóptero en interiores,» Trabajo Fin de Grado, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, 2019. dirección: https://repositorio.comillas.edu/xmlui/handle/11531/32609.

[5] FAS, *UAVs in agriculture*, Última consulta el 1 de junio de 2025. dirección: https://www.fas.scot/environment/uavs-in-agriculture/#:~:text=Current%20uses%20for%20drones%20in,health%20monitoring%20and%20disease%20detection.

[6] CubePilot, *The Cube Module Overview*, Última consulta el 7 de julio de 2025. dirección: https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview.

[7] Benewake, *TFmini-S Datasheet*, Última consulta el 3 de marzo de 2025. dirección: https://en.benewake.com/DataDownload/index_pid_20_lcid_22.html.

[8] *Emisora FS-i6S*, Última consulta el 31 de enero de 2025. dirección: https://rc-innovations.es/shop/fls00003129-flysky-fs-i6s-10ch-rx-ia6b-15631#attr=10534.

[9] Espressif, *ESP32-C3esp-dev-kits Documentation*, Última consulta el 16 de julio de 2025. dirección: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c3/esp32-c3-devkitm-1/user_guide.html.

[10] *Cámara Optitrack Flex13*, Última consulta el 31 de enero de 2025. dirección: https://www.optitrack-shop.de/flex-serie/kameras/125/flex-13.

[11] P. Velusamy, S. Rajendran, R. K. Mahendran, S. Naseer, M. Shafiq y J.-G. Choi, «Unmanned Aerial Vehicles (UAV) in Precision Agriculture: Applications and Challenges,» *Energies*, vol. 15, n.º 1, 2022, ISSN: 1996-1073. DOI: 10.3390/en15010217. dirección: https://www.mdpi.com/1996-1073/15/1/217.

[12] C. Todd, M. Watfa, Y. El Mouden et al., «A proposed UAV for indoor patient care,» *Technology and health care : official journal of the European Society for Engineering and Medicine*, sep. de 2015. DOI: 10.3233/THC1046. dirección: https://www.researchgate.net/publication/282248726_A_proposed_UAV_for_indoor_patient_care.

[13] *Cube Orange+*, Última consulta el 31 de enero de 2025. dirección: https://www.amazon.com/-/es/Set-est%C3%A1ndar-The-Cube-Orange/dp/B0C8Y1LMGZ.

[14] *Benewake TFmini-S LiDAR Module*, Última consulta el 31 de enero de 2025. dirección: https://www.benewake.com/en/tfmini-s.html.

[15] *Batería Lipo*, Última consulta el 31 de enero de 2025. dirección: https://www.amazon.es/Tattu-Bater%C3%ADa-Quadcopters-Nighthawk-Lumenier/dp/B016MM2TA8.

[16] *T-Motor Brushless*, Última consulta el 31 de enero de 2025. dirección: https://www.amazon.es/Racerstar-Racing-BR2205-2300KV-escobillas/dp/B0BPDGZ92R.

[17] *Hélices tripalas 5x3*, Última consulta el 31 de enero de 2025. dirección: https://www.apcprop.com/product/5x3e/.

[18] *ESC BLHeli 20A*, Última consulta el 31 de enero de 2025. dirección: https://oscarliang.com/dys-bl20a-sn20a-mini-opto-esc-alternative-to-k/.

[19] *PDB XT60*, Última consulta el 31 de enero de 2025. dirección: https://www.amazon.es/INLIMA-helic%C3%B3ptero-cuadric%C3%B3ptero-muliticopter-distribuci%C3%B3n/dp/B0C7VVHY5L?th=1.

[20] *Receptor FS-A8S*, Última consulta el 31 de enero de 2025. dirección: https://es.banggood.com/Flysky-FS-A8S-FS-A8S-V2-2_4G-8CH-Mini-RC-Receiver-with-PPM-i-BUS-SBUS-Output-p-1092861.html?cur_warehouse=HK.

[21] *ESP32-C3-DevKitM-1*, Última consulta el 14 de junio de 2025. dirección: https://www.amazon.es/dp/B0CB84WRZY/ref=sspa_dk_detail_1?psc=1&pd_rd_i=B0CB84WRZY&pd_rd_w=szp1N&content-id=amzn1.sym.0c640cbd-b6e0-461d-8cfc-a8934b5122df&pf_rd_p=0c640cbd-b6e0-461d-8cfc-a8934b5122df&pf_rd_r=14Y7S4GD5V7NNCEY4T1M&pd_rd_wg=lfm1p&pd_rd_r=938fecbb-db3b-43d5-b8dc-58b37dbcbb81&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWw.

[22] P. Garg, «Characterisation of Fixed-Wing Versus Multirotors UAVs/Drones,» *Journal of Geomatics*, vol. 16, n.º 2, págs. 152-159, oct. de 2022. DOI: 10.58825/jog.2022.16.2.44. dirección: https://onlinejog.org/index.php/journal_of_geomatics/article/view/44.

[23] TheDmel, *A swarm of nano quadrotors*, Última consulta el 28 de febrero de 2025. dirección: https://www.youtube.com/watch?v=YQIMGV5vtd4.

[24] R. Drones, *Drones, inventarios autónomos y 5G*, Última consulta el 1 de julio de 2025. dirección: https://rpas-drones.com/2023-drones-inventarios-autonomos-y-5g/.

[25] StockRC, *Drones para la logística*, Última consulta el 1 de julio de 2025. dirección: https://stockrc.com/es/blog/dji-news/drones-para-la-logistica.

[26] infodron, *Los bomberos de Alicante adquieren 3 drones SAR*, Última consulta el 1 de junio de 2025. dirección: https://www.infodron.es/texto-diario/mostrar/3530716/bomberos-alicante-adquieren-tres-drones-sar#:~:text=Los%20drones%20de%20los%20bomberos,Foto:%20Speis..

[27] A. Drone, *Cómo ayudan los drones en la gestión de emergencias y rescates*, Última consulta el 1 de junio de 2025. dirección: https://acgdrone.com/como-ayudan-los-drones-en-emergencias-y-rescates/#:~:text=Inundaciones:%20Su%20capacidad%20para%20comprobar%20el%20alcance,reducir%20el%20impacto%20de%20los%20desastres%20naturales..

[28] HobbyTuxtla, *Trabajos que se pueden hacer con drones topográficos*, Última consulta el 1 de junio de 2025. dirección: https://www.hobbytuxtla.com/trabajos-drones-topograficos/#:~:text=La%20alta%20resoluci%C3%B3n%20de%20las%20c%C3%A1maras%20permite,y%20eficacia%20en%20%C3%A1reas%20dif%C3%ADciles%20de%20alcanzar..

[29] Gridflight, *Servicios de salvamento marítimo con drones*, Última consulta el 1 de junio de 2025. dirección: https://www.gridflight.tech/emergencias/salvamento-maritimo/#:~:text=El%20dispositivo,%20adem%C3%A1s%20de%20contar%20con%20una,comunicarse%20con%20la%20v%C3%ADctima%20si%20fuese%20necesario..

[30] Telemadrid, *Los drones de la Policía Nacional ayudan a encontrar a una pareja de ancianos con alzheimer desaparecida en Leganés*, Última consulta el 11 de julio de 2025. dirección: https://www.telemadrid.es/programas/buenos-dias-madrid/Los-drones-de-la-Policia-Nacional-ayudan-a-encontrar-a-una-pareja-de-ancianos-con-alzheimer-desaparecida-en-Leganes-2-2787341245--20250610100853.html.

[31] Benewake, *TFmini-S User Manual*, 2024. dirección: https://en.benewake.com/DataDownload/index_pid_20_lcid_22.html.

[32] P. Sonic, *What is a battery C rating?* Dirección: https://www.power-sonic.com/wp-content/uploads/2021/02/What-is-a-battery-C-rating.pdf.

[33] H. Solar, *Clasificación de la batería*. dirección: https://es.higonsolar.com/what-is-battery-c-rating#:~:text=La%20clasificaci%C3%B3n%20C%20de%20la%20bater%C3%ADa%20se%20puede%20definir%20como,clasificaci%C3%B3n%20C%20de%20la%20bater%C3%ADa.

[34] HeliFreak, *A closer look at the strength and shape of 2.4 GHz radio signals*, Última consulta el 29 de mayo de 2025, 2011. dirección: https://www.helifreak.com/showthread.php?t=347351.

[35] DroneBridge, *DroneBridge for ESP32*, Última consulta el 16 de julio de 2025. dirección: https://dronebridge.gitbook.io/docs/dronebridge-for-esp32/hardware-and-wiring.

[36] M. Petrlík, T. Krajník y M. Saska, «LiDAR-based Stabilization, Navigation and Localization for UAVs Operating in Dark Indoor Environments,» en *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, Atenas, Grecia, jun. de 2021. dirección: https://arxiv.org/pdf/2302.01883.

[37] A. Agarwal, A. Pandey y M. Malik, «LiDAR based object detection system for drones,» en *Proceedings of the 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, Uttar Pradesh, India, dic. de 2023. DOI: 10.1109/icac3n60023.2023.10541512.

[38] A. T. Miller y A. V. Rao, *Nonsingular Euler Parameterizations for Motion of a Point Mass in Atmospheric Flight*, 2021. arXiv: 2104.08972 [math.DS]. dirección: https://arxiv.org/abs/2104.08972.

[39] D. Mayne, J. Rawlings, C. Rao y P. Scokaert, «Constrained model predictive control: Stability and optimality,» *Automatica*, vol. 36, n.º 6, págs. 789-814, 2000. DOI: https://doi.org/10.1016/S0005-1098(99)00214-9.

[40] P. Autopilot, *Quick Start Cube*, Última consulta el 23 de mayo de 2025. dirección: https://docs.px4.io/main/en/assembly/quick_start_cube.html.

[41] P. Autopilot, *PX4 Autopilot*, Última consulta el 7 de julio de 2025. dirección: https://docs.px4.io/main/en/.

[42] ArduPilot, *Guía ArduPilot (MissionPlaner)*, Última consulta el 1 de marzo de 2025. dirección: https://ardupilot.org/copter/docs/common-benewake-tfmini-lidar.html.

[43] Mathworks, *PX4 PWM Output*, Última consulta el 1 de marzo de 2025. dirección: https://es.mathworks.com/help/uav/px4/ref/px4pwmoutput.html.

[44] Mathworks, *PWM getting started PX4*, Última consulta el 1 de marzo de 2025. dirección: https://es.mathworks.com/help/uav/px4/ref/pwm-getting-started-px4.html.

[45] NXP, *I2C-bus specification and user manual*, Última consulta el 2 de marzo de 2025. dirección: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[46] *UAV Toolbox*, Última consulta el 31 de enero de 2025. dirección: https://www.mathworks.com/products/uav.html.

[47] T. E. Mindset, *Brushless Motor - How they work BLDC ESC PWM*, Última consulta el 28 de febrero de 2025. dirección: https://www.youtube.com/watch?v=yiD5nCfmbV0.

[48] P. Autopilot, *ESP32 WiFi Module*, Última consulta el 16 de julio de 2025. dirección: https://docs.px4.io/main/en/telemetry/esp32_wifi_module.html.

# Apéndice A

# Alineación con los Objetivos de Desarrollo Sostenible (ODS)

Durante el desarrollo del TFG, se ha tenido en cuenta que cumplan y contribuyan con los Objetivos de Desarrollo Sostenible establecidos por la ONU. Con este TFG, se busca la innovación y la eficiencia, buscando consumir lo menos posible y optimizando los recursos empleados al máximo. Los recursos se han seleccionado teniendo en consideración diversos factores, como el peso, el consumo y su fiabilidad, ajustándolos a las necesidades del dron. Los principales ODS alineados con este proyecto son:

## A.1. ODS 4: Educación de Calidad

El desarrollo de este dron servirá para futuros proyectos y proporcionará herramientas didácticas en asignaturas relacionadas con control y robótica.

## A.2. ODS 9: Industria, innovación e infraestructura

A la hora de escoger los distintos recursos necesarios para el proyecto, se han tenido en cuenta diversos factores, como el peso, el consumo y su fiabilidad, ajustándolos a las necesidades del dron. Se ha buscado en todo momento un enfoque que permita el funcionamiento del cuadricóptero con una demanda de energía baja.

## A.3. ODS 12. Producción y consumo responsable

La selección de componentes, como se explicó anteriormente, se ha hecho teniendo en cuenta su bajo consumo, pero también se ha priorizado la reutilización de componentes que ya estaban presentes en la universidad. De esta forma, se ha evitado el gasto innecesario en nuevos recursos, garantizando la optimización energética en varios aspectos.

## A.4. ODS 17. Alianzas para lograr los objetivos

Este proyecto ha sido una colaboración entre distintas especialidades y departamentos, promoviendo la cooperación y coordinación para aplicar los distintos conocimientos adquiridos durante la carrera.

Por todos los objetivos citados, este proyecto se alinea con los estándares de sostenibilidad fijados por la Unión Europea, garantizando el consumo responsable, el ahorro energético, la innovación y la eficiencia.

# Apéndice B

# Información adicional del Cube Orange+

La alimentación tanto de la FMU como IO es de 3.3V, donde cada uno regula la tensión que recibe a la entrada. La alimentación de los puertos del soporte ADS-B es de 3.3V por defecto, pero se puede modificar la tensión a 5V si es necesario. El valor estándar es 3.3V, para proteger los componentes más sensibles.

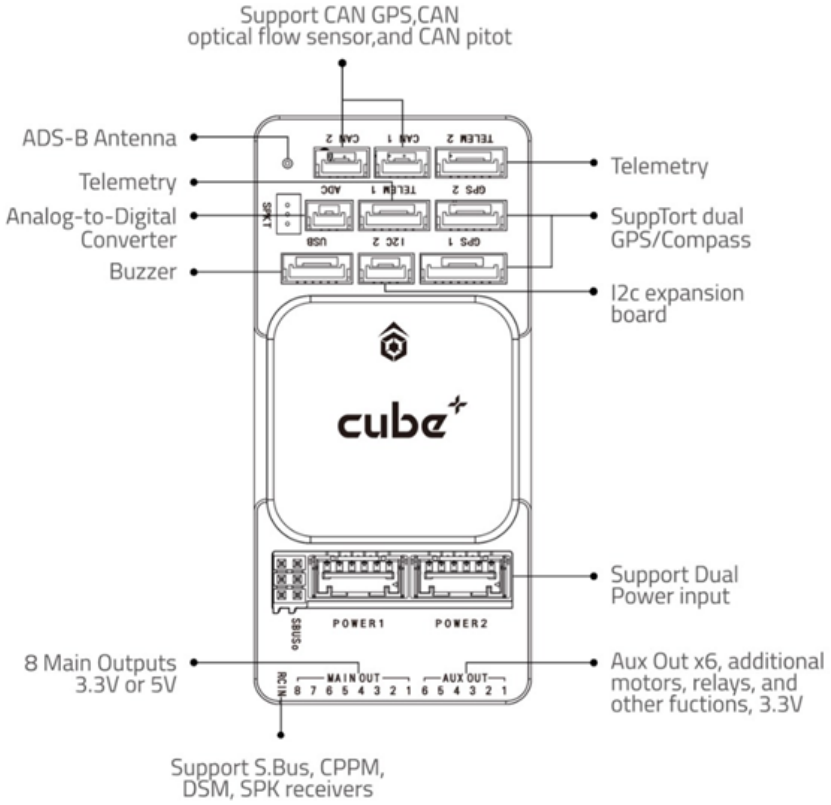


Figura B.1: Microcontrolador Cube Orange+

En la Figura B.2 se puede observar lo siguiente:

El puerto USB suministra 5V, y los puertos de las baterías entregan 5.3V. Las entradas de alimentación se encuentran limitadas a 5.7V. La prioridad la tiene la batería conectada al puerto principal (MAIN), seguido del puerto auxiliar. El puerto USB tiene prioridad baja.

La IO y la FMU reciben 5V (como máximo 5.7V) están alimentadas a 3.3V/300mA, y tienen sensores independientes para regular la tensión. Estos sensores son los LDO (*Low DropOut regulator*), unos reguladores lineales de voltaje que permiten mantener la tensión de salida constante para cualquier tensión de entrada. Evidentemente, la caída de tensión no interesa que sea muy grande, ya que esa caída se disipa en forma de calor.

RCT (*Real-Time Clock*) es un reloj que mantiene la fecha incluso aunque el dispositivo se apague. Tanto la FMU como la I/O lo requieren, para mantener configuraciones y parámetros. Por otro lado, la SRAM (*Static Random-Access Memory*) sirve para almacenar datos de manera temporal, es decir, es una memoria volátil.

Al igual que en el caso de los LDO, tienen RCT y SRAM independientes, para dar mayor robustez al sistema. Además, estos últimos están alimentados con una batería de respaldo (*Backup Battery* en la imagen), la cual está integrada en el propio microcontrolador.

Los filtros EMI (Interferencia Electromagnética) son necesarios en las rutas más sensibles, como es el caso de los sensores o la telemetría.

Los puertos de telemetría, GPS, y de comunicación I2C y CAN, están alimentados a 3.3V o 5V, según cómo esté configurado. En este caso, la alimentación se mantuvo en el valor estándar, que es 3.3V. La corriente está limitada a 1.5A, y la tensión no superará los 5.7V. Los servos se alimentan por una ruta independiente, con una tensión limitada a un rango entre 4.5 y 10V.

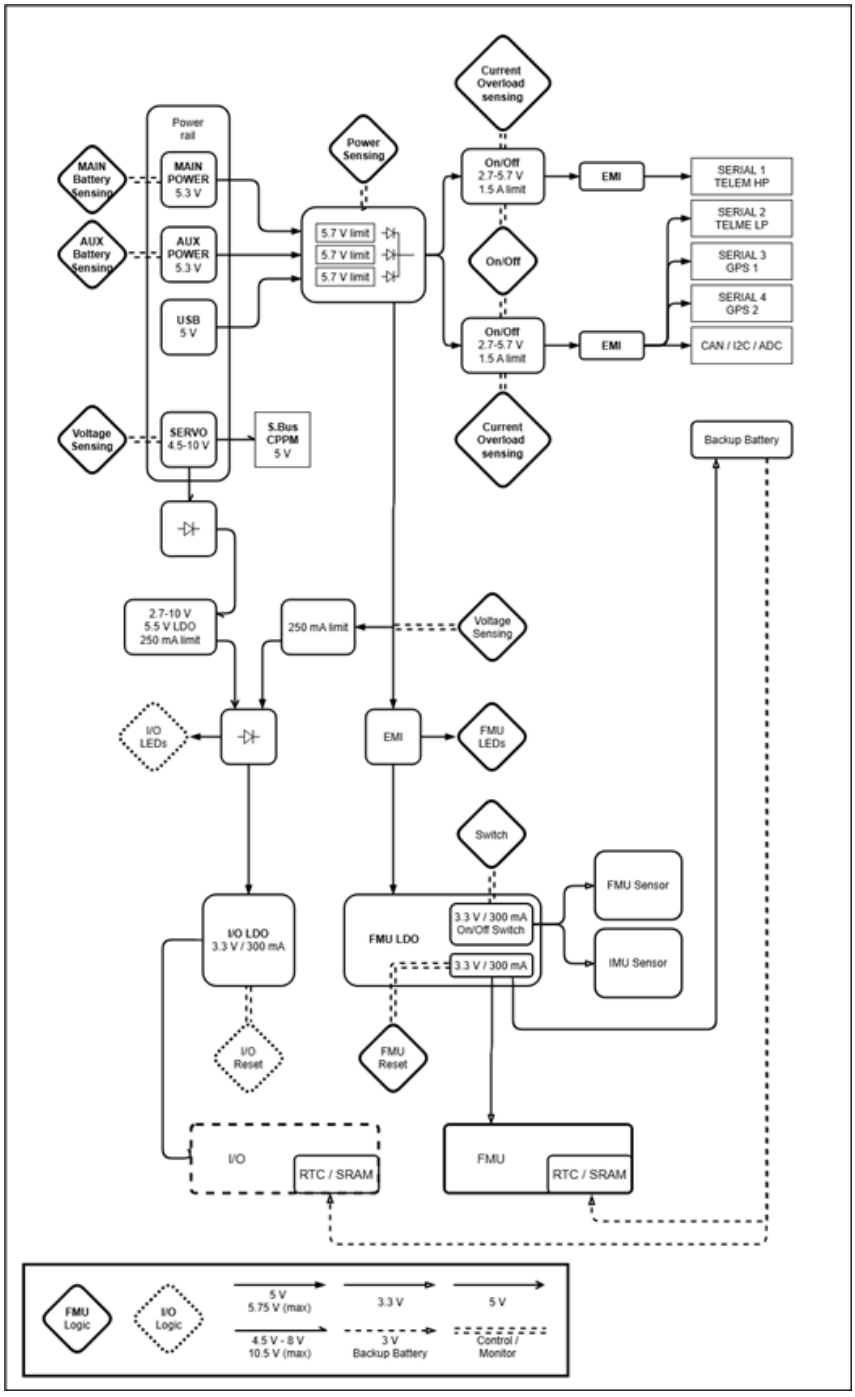


Figura B.2: Alimentación interna del microcontrolador

# Apéndice C

# Puesta en marcha del sistema y entorno de desarrollo

Este anexo documenta de forma detallada todos los procedimientos necesarios para replicar la instalación del entorno de desarrollo, la configuración del sistema de control, y la conexión con el hardware utilizado en el proyecto. El objetivo es facilitar que futuros estudiantes puedan continuar este trabajo sin partir desde cero.

## C.1. Requisitos previos

- Sistema operativo: **Windows 10 o Windows 11 / Ubuntu 22.04**
- Cuenta activa en MathWorks para acceder a MATLAB
- Acceso físico al hardware (Cube Orange+, emisora, sensores...)

## C.2. Instalación de MATLAB 2024b

1. Descargar MATLAB desde `https://www.mathworks.com/downloads/`
2. Instalar con licencia académica de la Universidad.
3. Durante la instalación, incluir Simulink, UAV Toolbox y UAV Toolbox Support Package for PX4 Autopilots.

De tener previamente Matlab 2024b instalado, solo se debe descargar el UAV Toolbox Support Package for PX4 Autopilots desde la pestaña "Get Add-Ons".

## C.3. Instalación del soporte para PX4

1. Abrir MATLAB y clickar en: Add-Ons ->Manage Add-Ons.

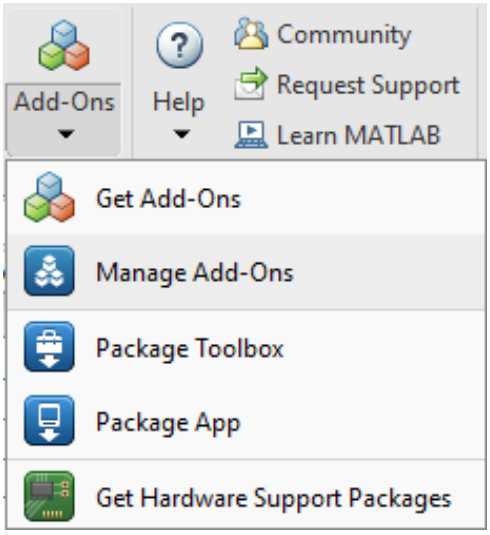


Figura C.1: Manage Add Ons

2. Abrir el setup (símbolo del engranaje) del UAV Toolbox Support Package for PX4 Autopilots

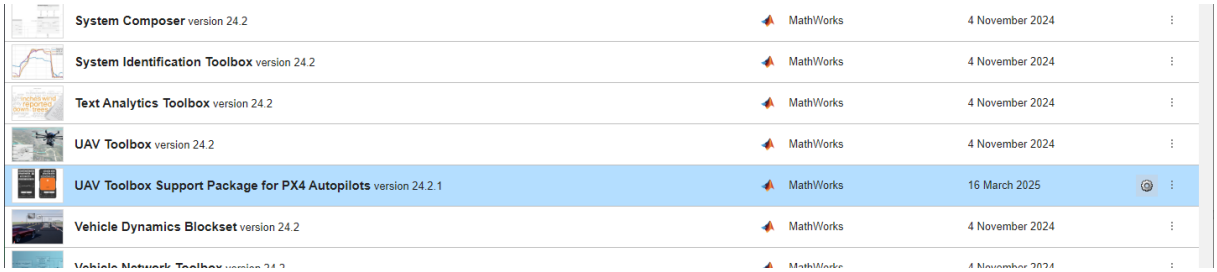


Figura C.2: Setup UAV Toolbox Support Package for PX4 Autopilots

3. Seguir el asistente para instalar todo.

**Paso 1: Instalar el subsistema Linux WSL2**
En primer lugar, si su sistema operativo es Windows, aparecerá esta pestaña:

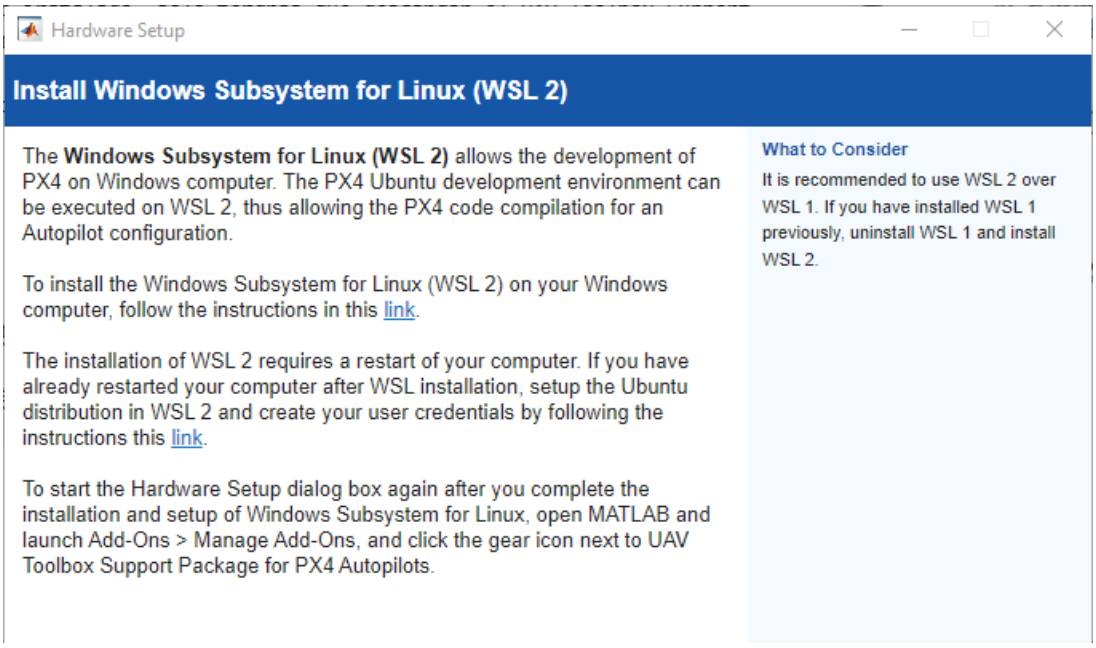


Figura C.3: Instalación de Linux WSL 2

Es necesario seguir el enlace indicado en la ventana, que redirige a la documentación oficial para instalar el Subsistema de Windows para Linux (WSL 2). Si su equipo ya tenía WSL 1 instalado, debe desinstalarlo antes de proceder.

**PRIMER LINK**

El primer link sirve para Instalar el subsistema Linux WSL2. Siga todos los pasos y una vez instalado, es necesario reiniciar el ordenador.

Para instalar correctamente la distribución de Ubuntu 22.04, es necesario abrir una ventana de terminal (PowerShell o CMD) con permisos de administrador y ejecutar el siguiente comando:

```
wsl --install -d Ubuntu-22.04
```

Una vez completada la instalación de Ubuntu 22.04, es necesario reiniciar el ordenador para que se apliquen correctamente los cambios del Subsistema de Windows para Linux (WSL 2).

La primera vez que se inicie Ubuntu 22.04 (ya sea automáticamente tras reiniciar o manualmente desde el menú de inicio), aparecerá una ventana como la mostrada en la Figura C.4, indicando que se está completando la instalación. Este proceso puede tardar unos minutos.

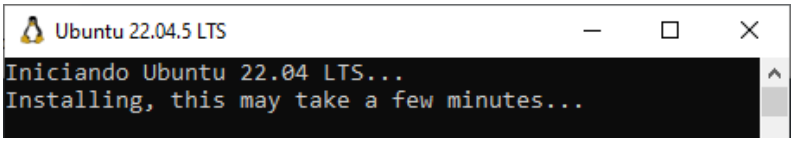


Figura C.4: Espera unos minutoss

Tras reiniciar, se debe abrir de nuevo una terminal (CMD o PowerShell) y establecer Ubuntu 22.04 como la distribución por defecto mediante el siguiente comando:

```
wsl --set-default Ubuntu-22.04
```

Este paso asegura que MATLAB utilice la versión de Ubuntu recién instalada como entorno principal para compilar el firmware PX4. Si no se realiza, pueden producirse errores durante la ejecución del asistente de configuración.

**SEGUNDO LINK**

Una vez instalado el Subsistema de Windows para Linux (WSL 2) y reiniciado el equipo, se debe instalar la distribución Ubuntu desde la Microsoft Store y crear un usuario, los pasos a seguir están en el segundo link. Esta distribución será utilizada por MATLAB para compilar el firmware PX4.

Durante la primera ejecución de Ubuntu, el sistema solicitará la creación de un nombre de usuario y una contraseña. Este usuario se utilizará como cuenta principal dentro del entorno WSL.

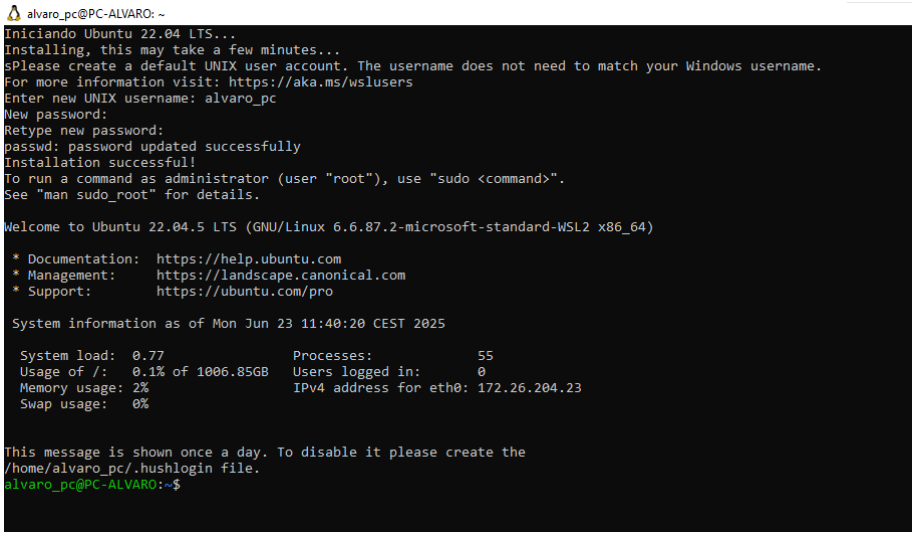


Figura C.5: Primera inicialización de Ubuntu 22.04 en WSL.

Una vez terminado, podrá escribir en la barra de tareas de búsqueda “Ubuntu” y le aparecerá la aplicación.

### Paso 2: instalación de Python 3.8.2

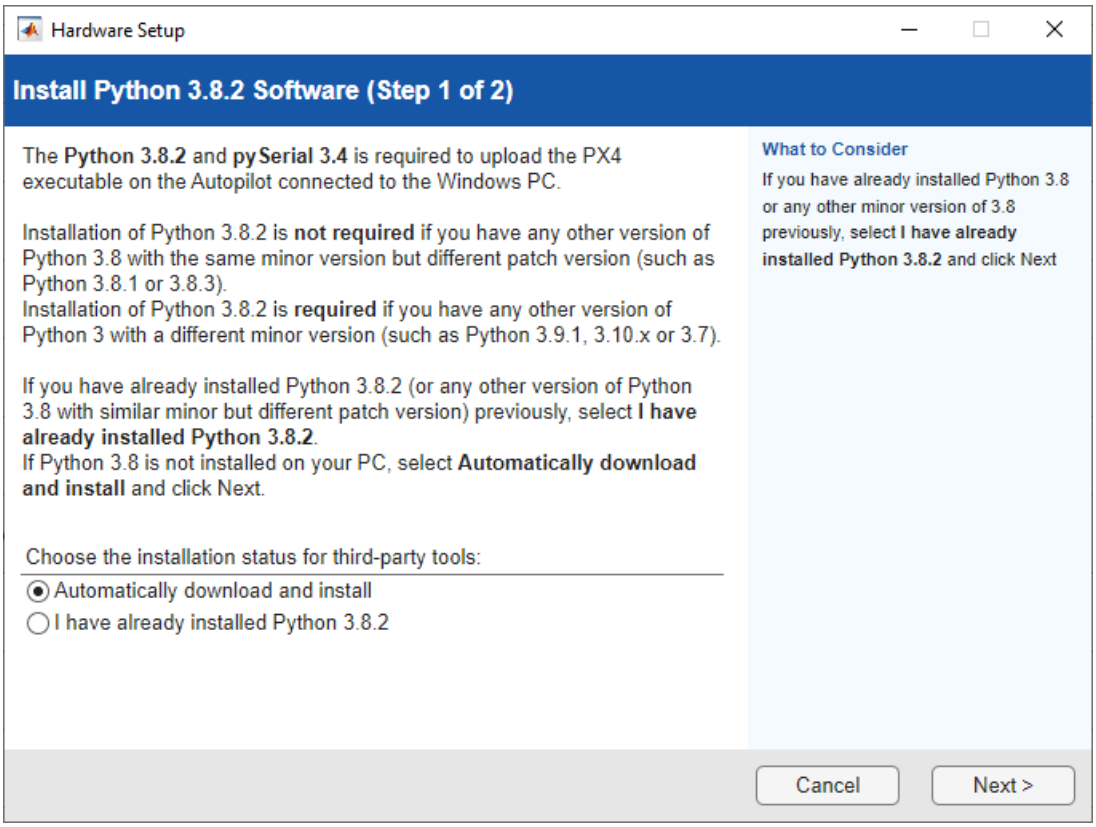


Figura C.6: Instalación de Python 3.8.2 y pySerial 3.4

Para posibilitar la carga del firmware PX4 en el autopiloto desde Windows, es necesario disponer de una versión compatible de Python 3.8.2 y del paquete `pySerial 3.4`.

- Si el sistema ya tiene instalada alguna versión de Python 3.8 con el mismo número de versión menor (por ejemplo, 3.8.1 o 3.8.3), no es necesario instalar Python 3.8.2 exactamente.
- Si el sistema tiene una versión distinta de Python (por ejemplo, 3.7, 3.9, 3.10...), sí será necesario instalar exactamente Python 3.8.2.

Se puede elegir una de las siguientes opciones:

- **Automatically download and install**: el asistente descargará automáticamente Python 3.8.2 y pySerial 3.4.
- **I have already installed Python 3.8.2**: si ya está instalado previamente.

**Paso 3: Confirmación de instalación de Python 3.8.2 y pySerial 3.4**

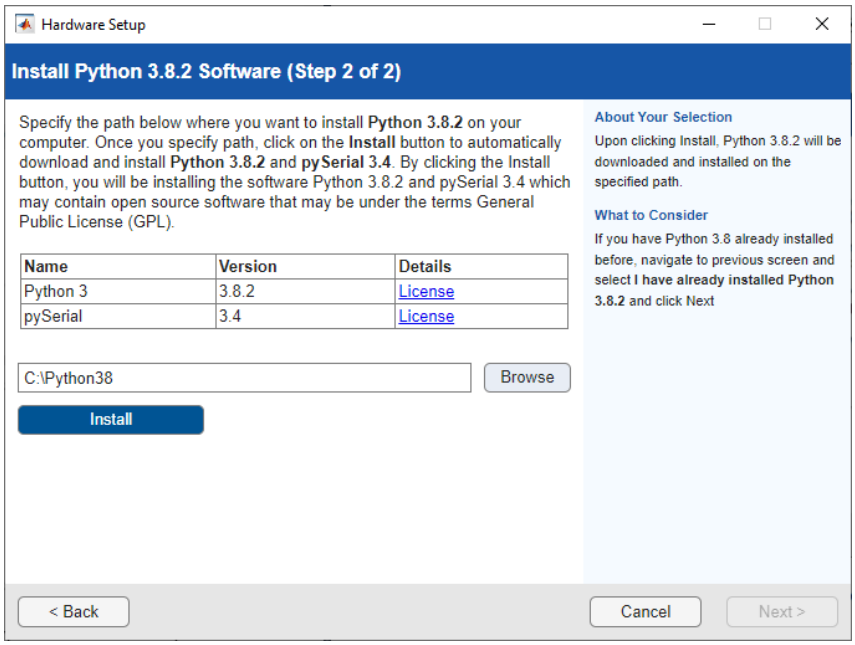


Figura C.7: Selección de ruta para instalar Python 3.8.2 y pySerial 3.4

En esta ventana se especifica la ruta donde se instalarán Python 3.8.2 y pySerial 3.4. Por defecto, la ruta sugerida es `C:\Python38`.

Para continuar, basta con hacer clic en el botón **Install**. El asistente descargará automáticamente las herramientas necesarias y las instalará en el sistema.

Se recomienda no modificar la ruta por defecto salvo que exista un motivo específico (por ejemplo, conflictos con otra instalación de Python).

Una vez completada la instalación, el asistente mostrará un mensaje de confirmación indicando que tanto **Python 3.8.2** como **pySerial 3.4** se han instalado correctamente, como se muestra en la Figura C.8.

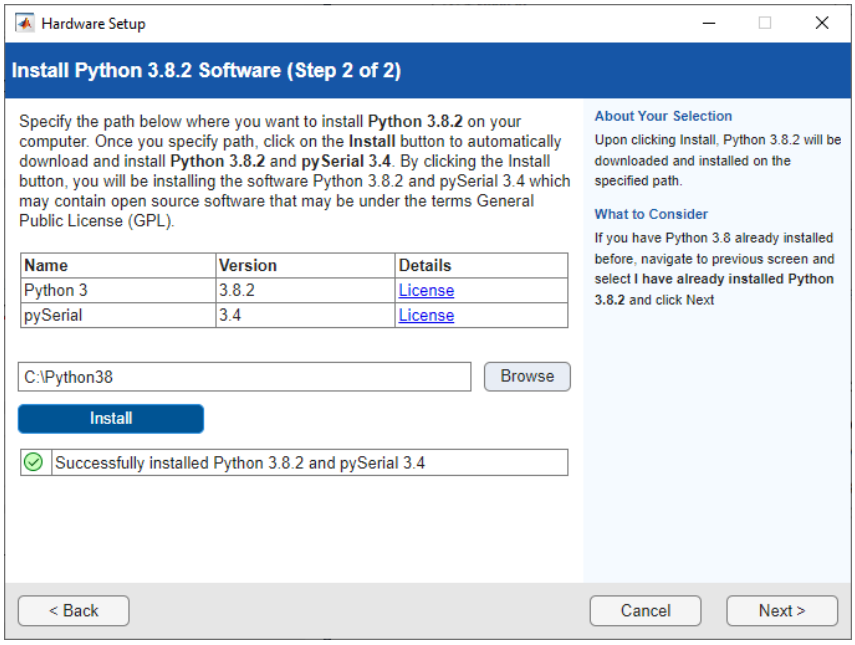


Figura C.8: Confirmación de instalación exitosa de Python 3.8.2 y pySerial 3.4

Para comprobar que la instalación se ha realizado correctamente, se recomienda abrir una ventana de terminal (CMD) y ejecutar los siguientes comandos:

```
python --version
```

Este comando debería devolver:

```
Python 3.8.2
```

Asimismo, es conveniente comprobar que el paquete `pySerial` está instalado correctamente con el siguiente comando:

```
pip list
```

En la lista de paquetes instalados, deberá aparecer una línea similar a:

```
pyserial 3.4
```


Figura C.9: Verificación de instalación correcta de Python 3.8 y de pyserial 3.4

Si al ejecutar
`python -version`
aparece un mensaje de error como el mostrado en la Figura C.10, es necesario añadir manualmente la ruta `C:\Python38` a las variables de entorno del sistema.

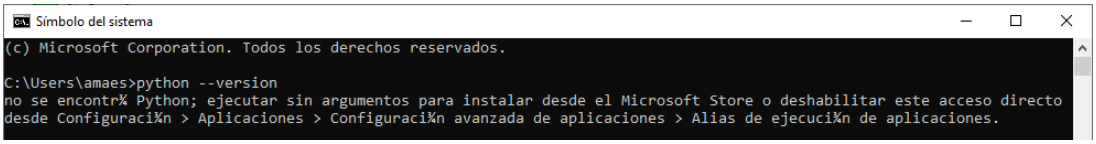


Figura C.10: Ejemplo de error al ejecutar `python -version`

Si alguno de estos comandos no devuelve el resultado esperado, asegúrese de que la ruta de instalación (por ejemplo, `C:\Python38`) se ha añadido correctamente a las variables de entorno del sistema. Para ello, abra el menú de inicio, escriba *"variables de entorno"*, y edite la variable **Path** del entorno de usuario o del sistema. Añada la carpeta donde se haya instalado Python, por ejemplo:

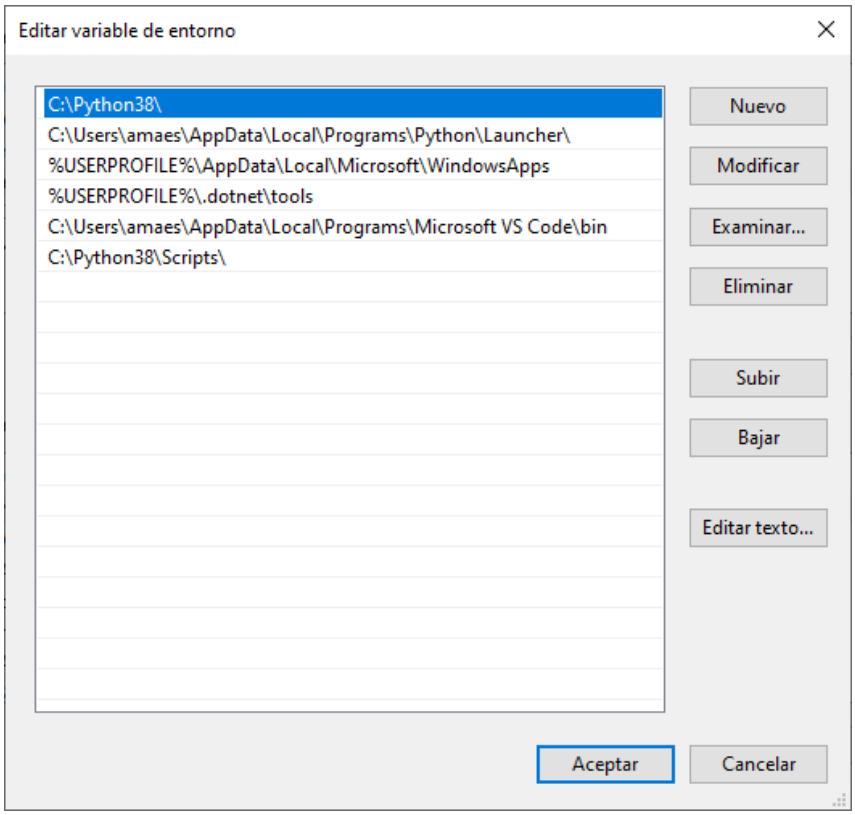


Figura C.11: Adición de la carpeta de instalación de Python al `Path` del sistema.

### Paso 4: Descarga del código fuente de PX4

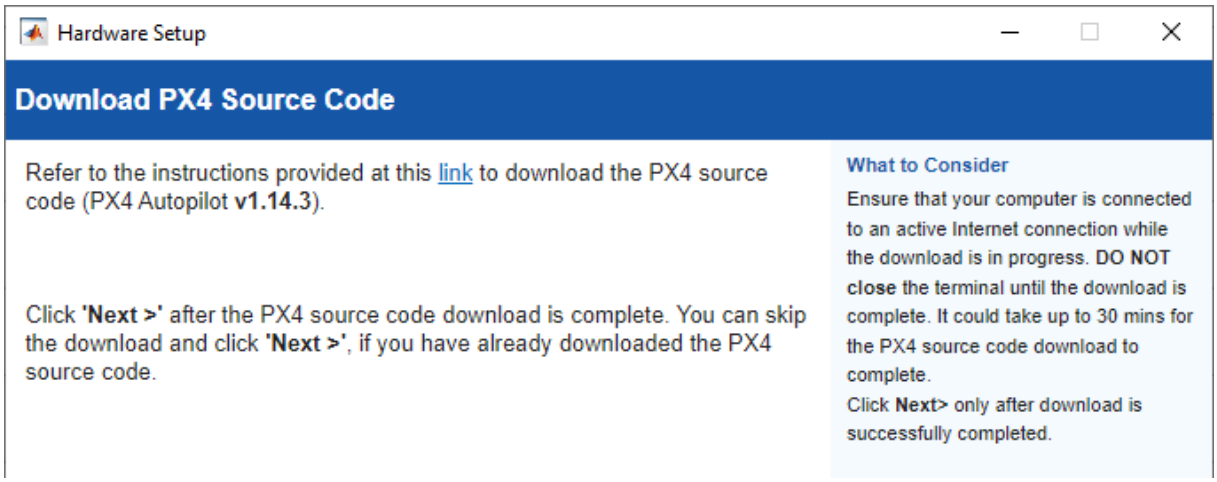


Figura C.12: Descarga del Código fuente de PX4

Para continuar con la instalación, es necesario descargar el código fuente del firmware PX4 (versión **v1.14.3**). Este proceso debe realizarse en el entorno WSL2 para evitar errores de permisos y problemas de rendimiento.

Los pasos a seguir son:

1. Abrir el terminal de Ubuntu (WSL2).
2. Navegar al directorio home con:

```
cd ~
```

3. Clonar el repositorio de PX4 desde GitHub:

   ```
   git clone https://github.com/PX4/PX4-Autopilot.git --recursive
   ```

4. Acceder al directorio descargado:

   ```
   cd PX4-Autopilot
   ```

5. Cambiar a la versión estable v1.14.3:

   ```
   git checkout v1.14.3 -f
   ```

6. Inicializar todos los submódulos del repositorio:

   ```
   git submodule update --init --recursive
   ```

Este proceso puede tardar varios minutos. Una vez completado, se puede continuar con los siguientes pasos del asistente de configuración.

### Paso 5: Validación del código fuente de PX4

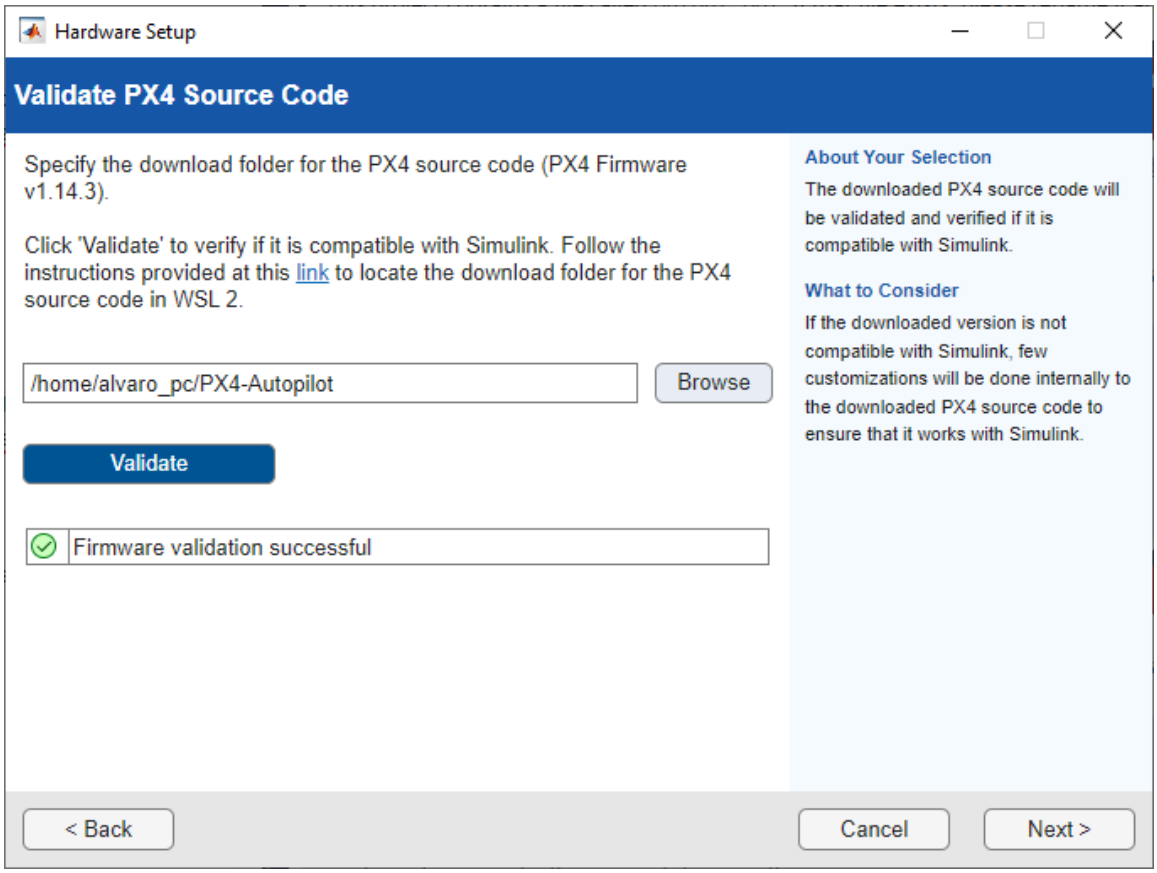


Figura C.13: Validación del código fuente de PX4 desde el asistente de instalación.

Una vez descargado el repositorio PX4 en el entorno WSL2, es necesario validar que la versión es compatible con Simulink. Para ello:

1. En la ventana del asistente, introducir la ruta completa al repositorio, por ejemplo:

   ```
   /home/alvaro_pc/PX4-Autopilot
   ```

2. Pulsar el botón **Validate**.

Si la validación es correcta, aparecerá el mensaje *Firmware validation successful*. Esto confirma que el repositorio contiene una versión adecuada del firmware (v1.14.3) y que puede ser utilizado con el entorno de desarrollo de Simulink.

### Paso 6: instalación del entorno de desarrollo (toolchain) de PX4 en WSL2

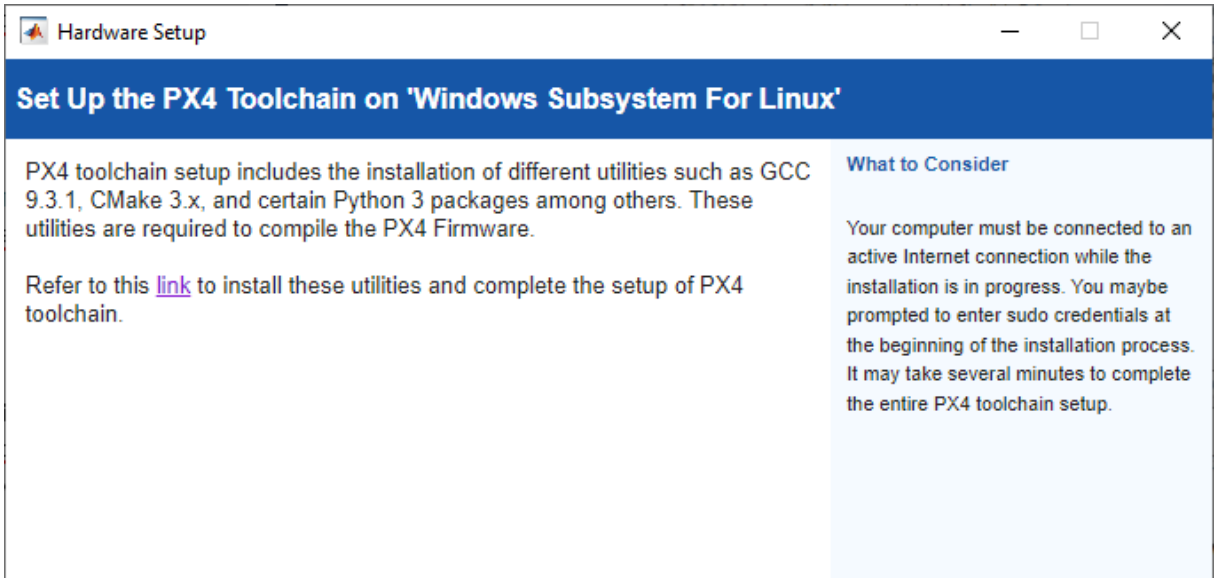


Figura C.14: Instalación del entorno de desarrollo (PX4 Toolchain)

Una vez validado el código fuente del firmware, el asistente solicita instalar el entorno de desarrollo necesario para compilar PX4. Este entorno incluye herramientas como `GCC 9.3.1`, `CMake 3.x`, `Ninja`, `Git` y varios paquetes de Python.

Para ello, es necesario seguir las instrucciones indicadas en el enlace proporcionado, que puede resumirse en los siguientes pasos:

1. Abrir una terminal WSL2.
2. Navegar a la carpeta donde se descargó el firmware:

   ```
   cd ~/PX4-Autopilot
   ```

3. Ir a la carpeta del script de instalación:

   ```
   cd Tools/setup
   ```

4. Ejecutar el script de instalación del toolchain:

   ```
   bash ./ubuntu.sh
   ```

5. Cuando el proceso haya terminado, reiniciar el sistema

### Paso 6: desactivar controladores por defecto de PX4

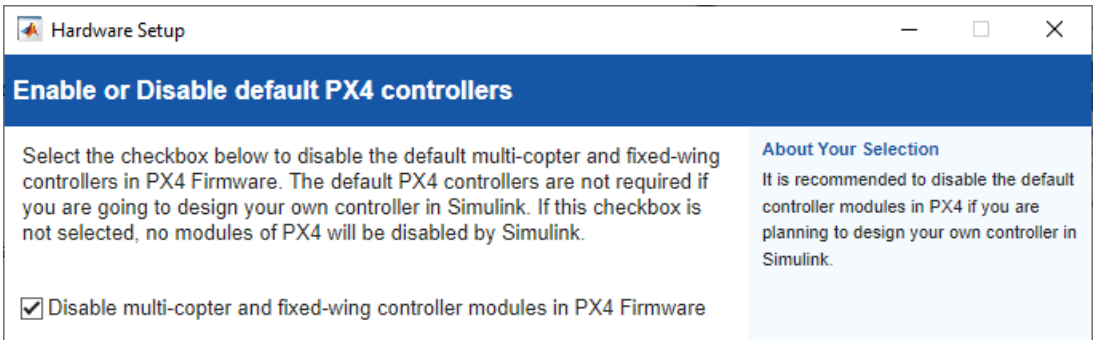


Figura C.15: Desactivación de controladores por defecto de PX4

Marca la casilla para desactivar los controladores por defecto de PX4. Es recomendable si vas a usar tu propio controlador en Simulink.

**Paso 7: seleccionar el autopiloto Cube Orange+**

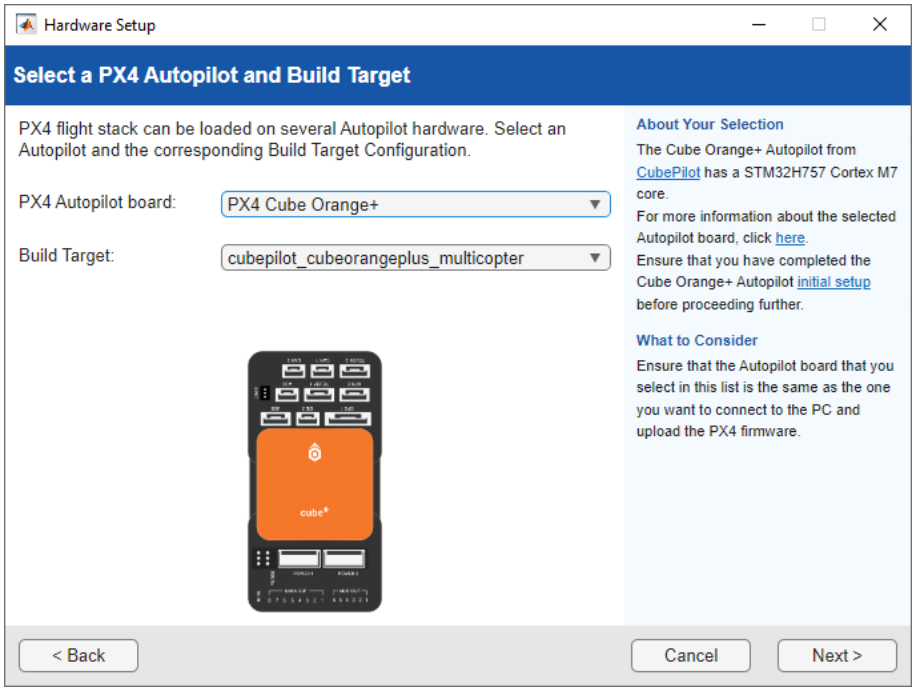


Figura C.16: Selección del autopiloto Cube Orange+ y el Build Target

Seleccionar:

- **PX4 Autopilot board:** `PX4 Cube Orange+`
- **Build Target:** `cubepilot_cubeorangeplus_multicopter`

**Paso 8: seleccionar script de inicio del sistema**

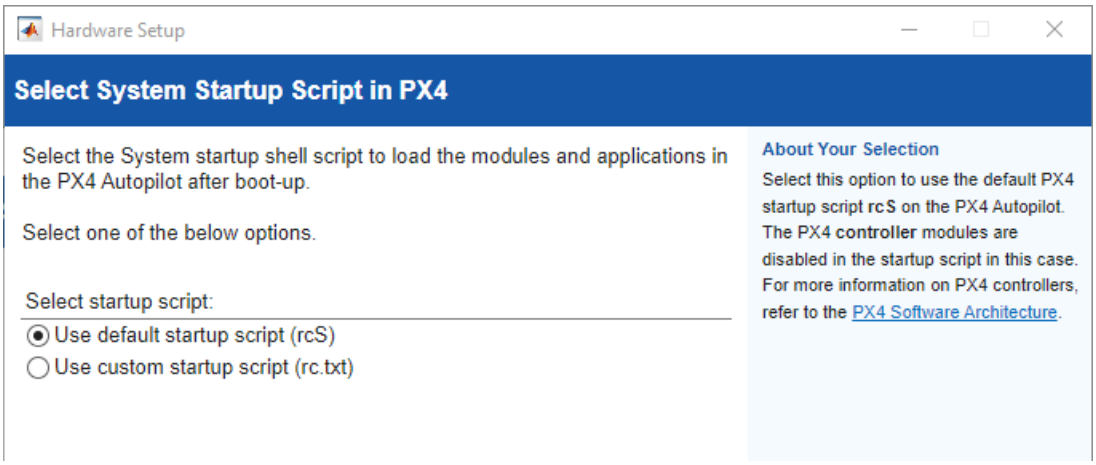


Figura C.17: Selección del script de arranque por defecto en PX4

Marcar la opción:

- `Use default startup script (rcS)`

### Paso 9: instalación de QGroundControl

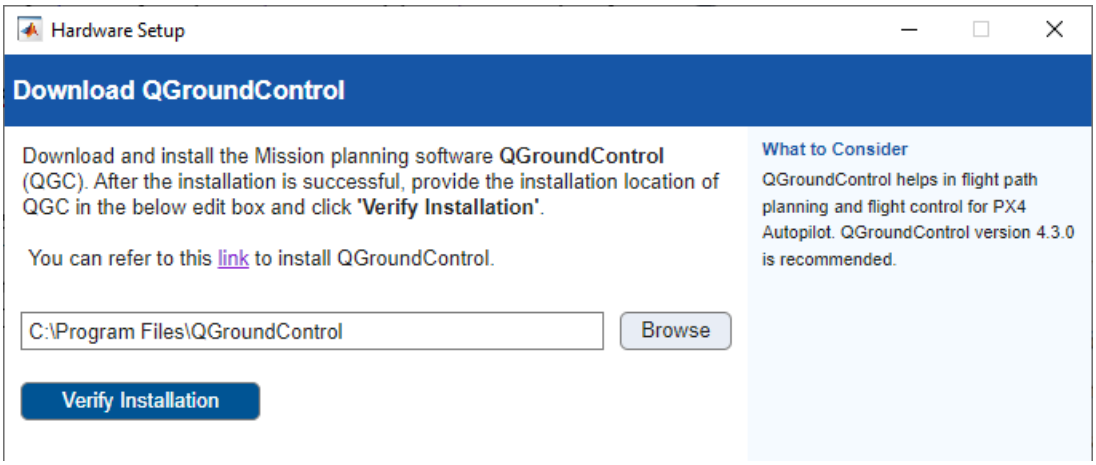


Figura C.18: Instalación de QGroundControl

QGroundControl (QGC) es el software de planificación de misiones y control de vuelo compatible con PX4. Para instalarlo:

1. Acceder al repositorio oficial (click en link)

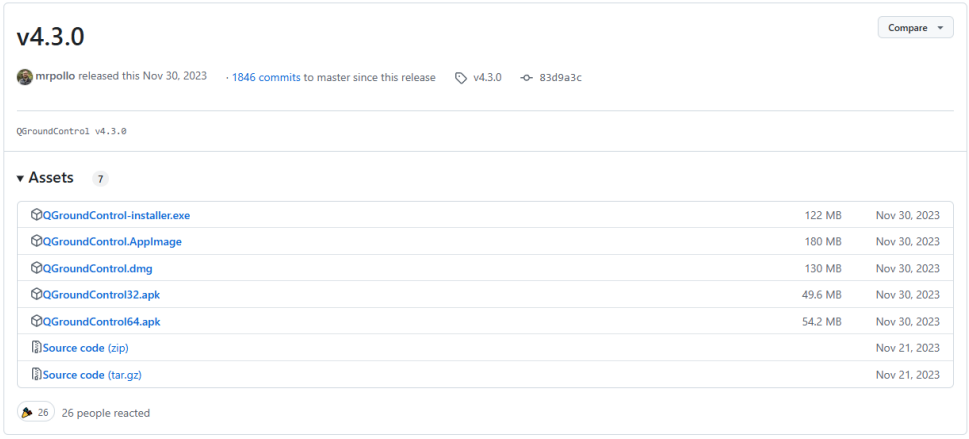


Figura C.19: Repositorio oficial de QGroundControl (v4.3.0) en GitHub

2. Descargar el archivo `QGroundControl-installer.exe`.
3. Ejecutar el instalador y seguir los pasos.
   Durante el proceso de instalación de QGroundControl, el sistema solicita instalar ciertos *drivers* de dispositivos USB, como los proporcionados por *Arduino LLC*. Estos controladores son necesarios para establecer la comunicación entre el ordenador y el autopiloto (por ejemplo, el Cube Orange+), ya que permiten reconocer correctamente el dispositivo cuando se conecta por puerto USB.

Tras la instalación, se debe verificar que el instalador se ha ubicado correctamente. Por defecto, QGroundControl se instala en:

```
C:\Program Files\QGroundControl
```

Una vez seleccionado el directorio de instalación (Browse), hacer clic en **Verify Installation**. Si la instalación es válida, se permitirá continuar con la configuración.

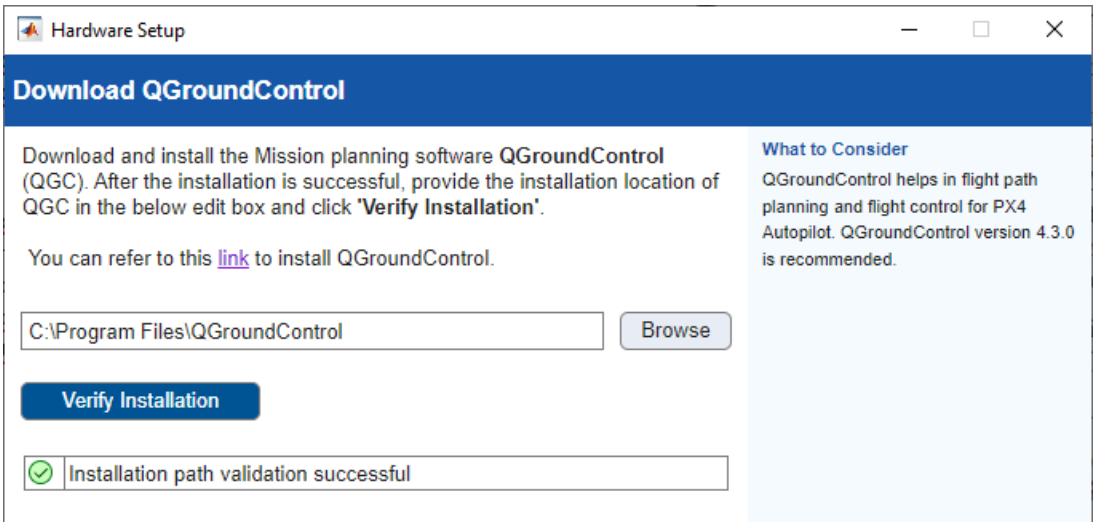


Figura C.20: Verificación de QGroundControl

### Paso 10: Seleccionar el *Airframe* en QGroundControl

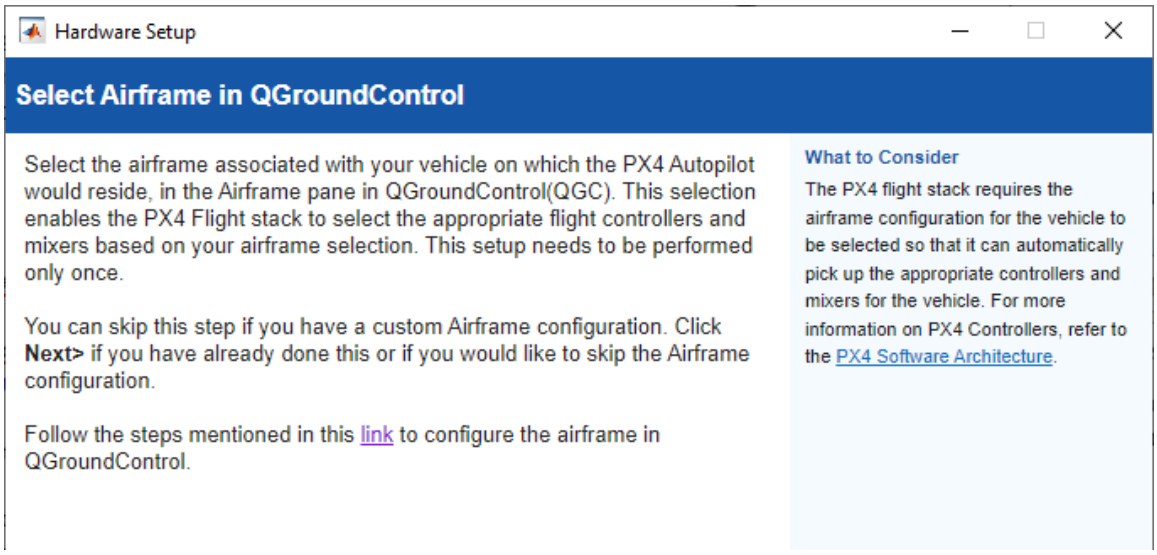


Figura C.21: Verificación de QGroundControl

Para completar la configuración del sistema PX4, es necesario especificar la geometría del vehículo que se está utilizando. Esta configuración permite que PX4 cargue los controladores y mezcladores adecuados para el tipo de dron. En este caso, se seleccionó un cuadricóptero en configuración X. Para ello, se siguieron los siguientes pasos en QGroundControl:

1. Iniciar **QGroundControl** y conectar el vehículo.
2. Desde QGroundControl, navegar a `Vehicle Setup >Airframe` (en la barra lateral) para abrir el panel de configuración de *Airframe*.
3. Dentro del grupo **Quadrotor X**, seleccionar la opción **Generic Quadrotor X geometry**, que corresponde a un cuadricóptero en configuración X.
4. Hacer clic en el botón **Apply and Restart**, situado en la parte superior derecha de la pantalla.
5. En el aviso emergente, confirmar pulsando **Apply** para guardar la configuración y reiniciar el vehículo.

En este proyecto se seleccionó la opción **Quadcopter en configuración "X"**, por ser la que mejor se adapta a la disposición física del dron.

**Paso 11: Compilación del firmware PX4**

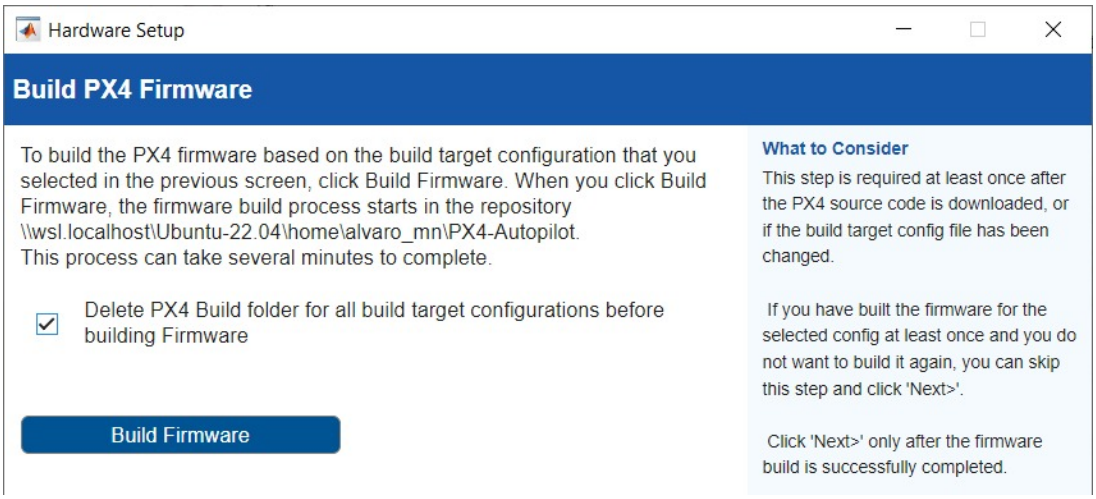


Figura C.22: Compilación exitosa del firmware PX4.

Una vez seleccionada la configuración del hardware y el airframe correspondiente, se procede a compilar el firmware de PX4 desde el asistente de configuración de hardware de Simulink.

Para ello, se marca la opción `Delete PX4 Build folder for all build target configurations before building Firmware` y se pulsa el botón **Build Firmware**. Este paso puede tardar varios minutos, ya que se compilan todos los módulos necesarios.

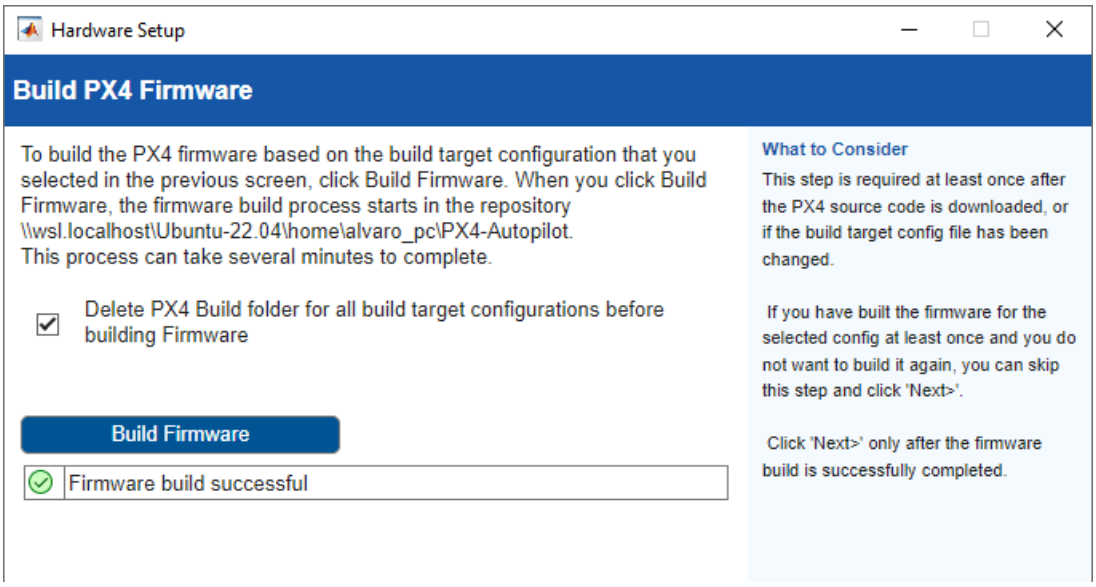


Figura C.23: Compilación exitosa del firmware PX4.

Una vez finalizado, debe aparecer el mensaje `Firmware build successful`. En ese caso, se puede continuar al siguiente paso.

### Paso 12: Test de conexión

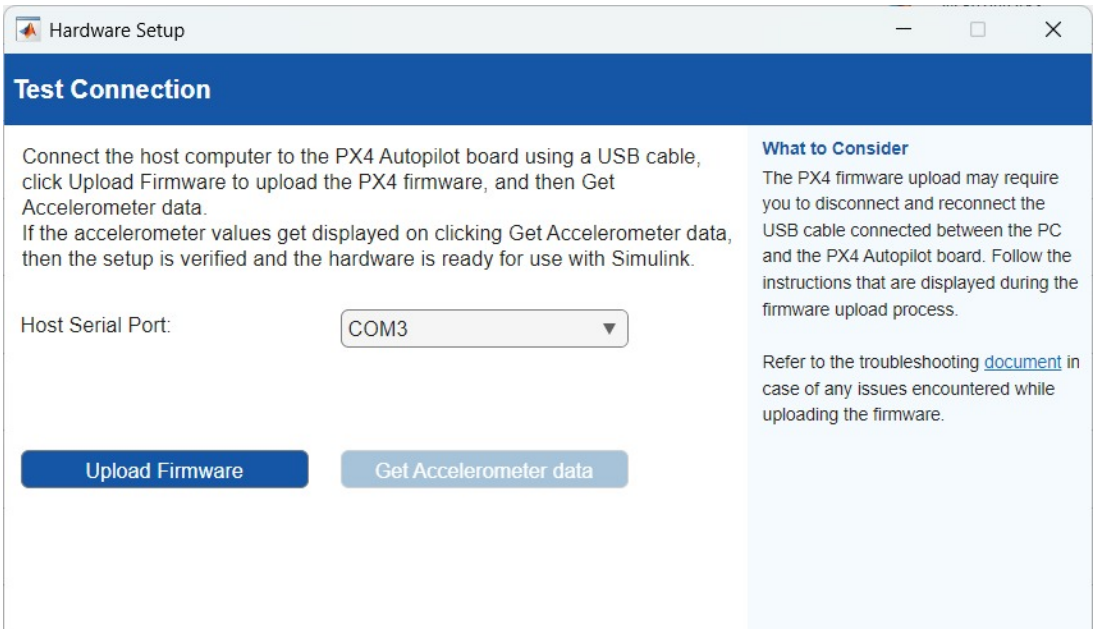


Figura C.24: Test de Conexión

Seleccione el puerto al que está conectado el Cube Orange+ (se puede confirmar en administrador de dispositivos) y pulse en "Upload Firmware". Le aparecerá la siguiente ventana:


Figura C.25: Desconectar y Conectar el cable USB

Desconecte el cable USB, pulse OK y vuelva a conectar. Tras breves minutos, si la conexión ha sido exitosa, le aparecerá el mensaje: "Firmware uploaded succesfully".

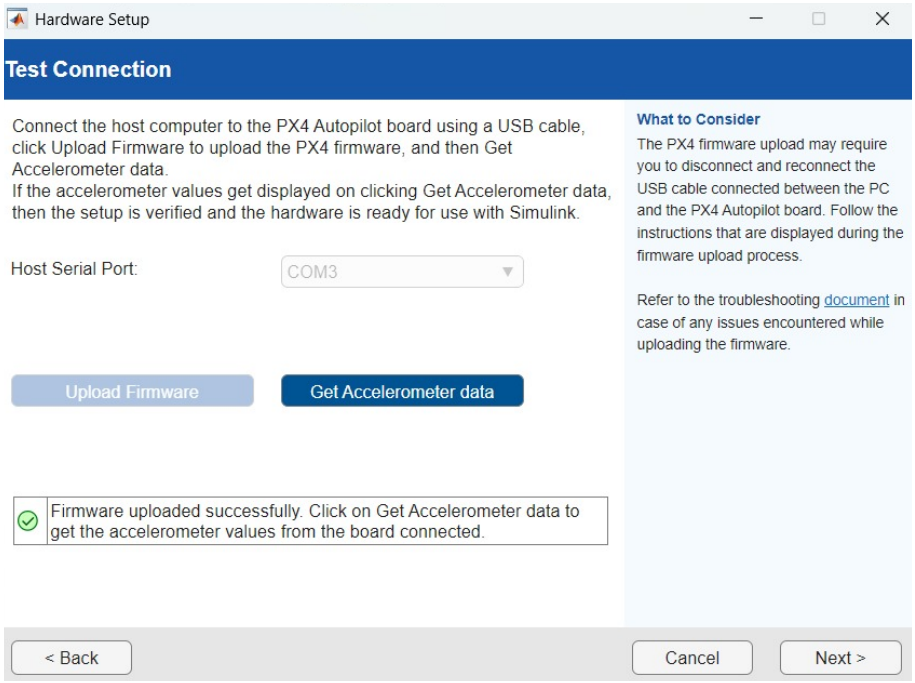


Figura C.26: validación Test de Conexión

### Paso 13: Hardware Setup Complete

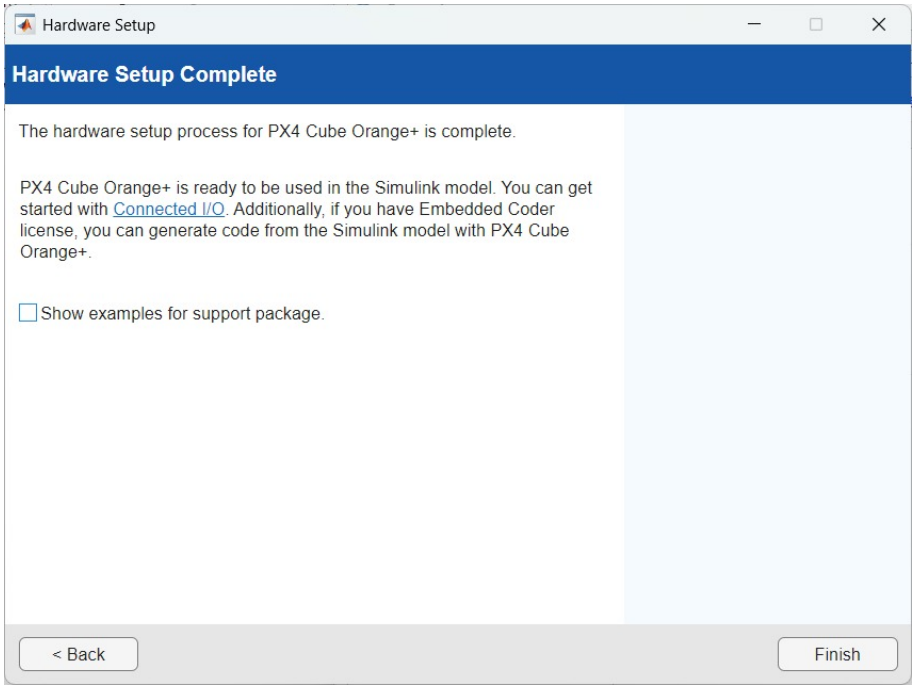


Figura C.27: Hardware setup completado

Haga click en Finish. Enhorabuena, has terminado de configurar el hardware.

## C.4. Configuración en Simulink

Una vez compilado correctamente el firmware, es necesario configurar el modelo `UAV_CONTROL_SYSTEM` en Simulink para establecer correctamente los parámetros de hardware y generación de código. A continuación se enumeran los pasos seguidos:

1. Abrir el modelo `UAV_CONTROL_SYSTEM` en Simulink.
2. Acceder a `Hardware Settings` mediante el menú: `Hardware >Hardware Settings`. En la sección `Hardware Implementation`, se seleccionan los siguientes parámetros:
   - **Hardware Board:** `PX4 Cube Orange+`
   - En `Target hardware resources`, dentro del grupo `Build options`, marcar la opción:
     - `Automatically determine serial port for firmware upload`
   - En el grupo `MAVLink`, habilitar:
     - `Enable MAVLink on /dev/ttyACM0`

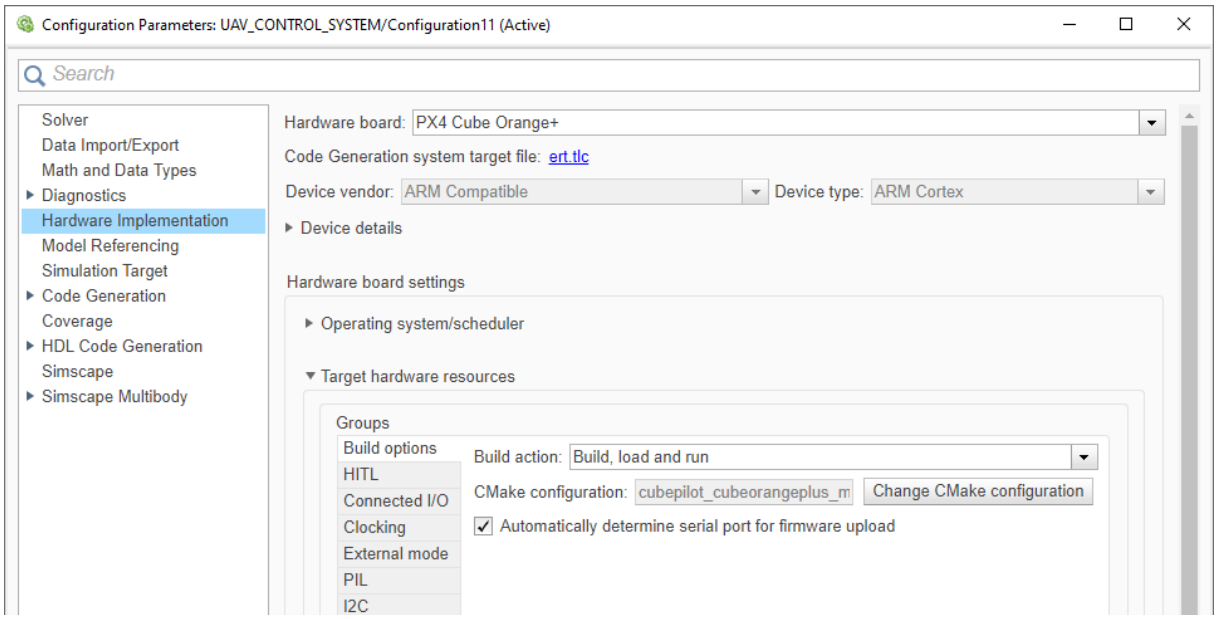


Figura C.28: Opciones de compilación seleccionadas

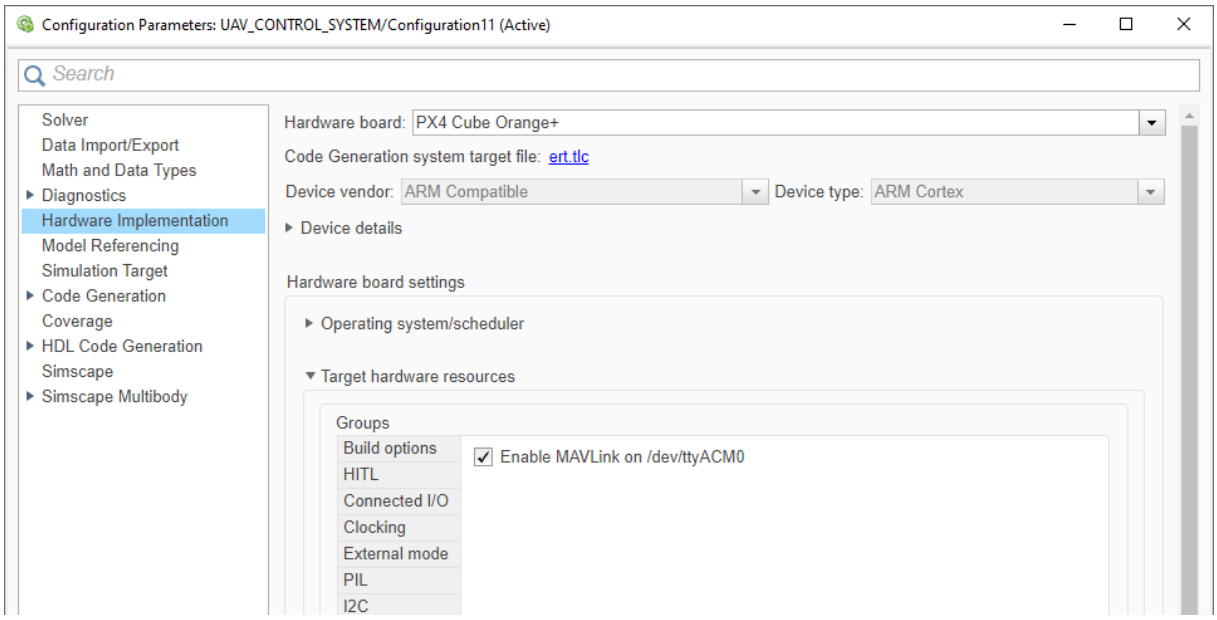


Figura C.29: Habilitación de MAVLink en `/dev/ttyACM0`

3. Finalmente, en la pestaña `Code Generation >Custom Code`, se deben realizar las siguientes modificaciones:
   - En `Code Information` añadir la cabecera:

```
#include <poll.h>
```

Código C.1: Incluir cabecera para compilación

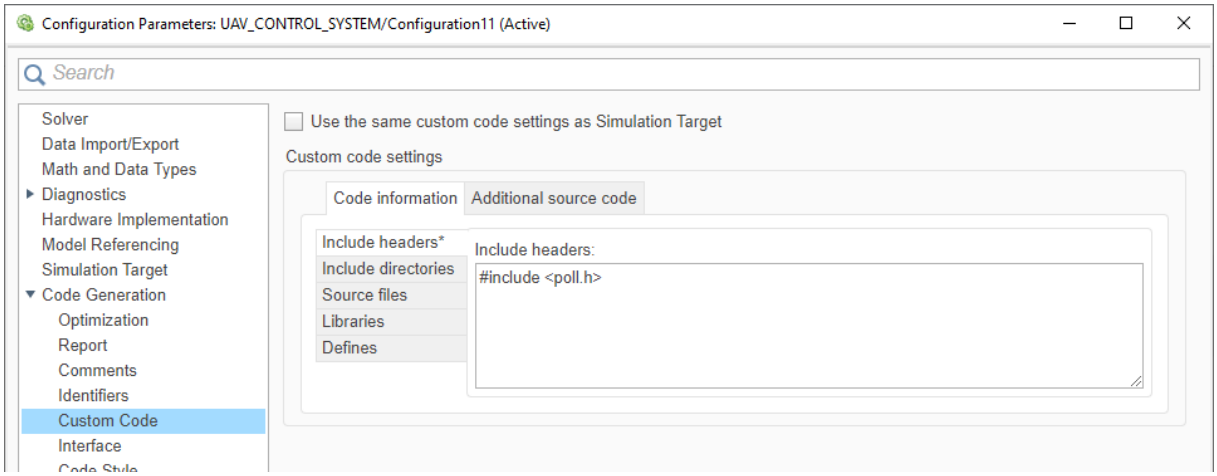


Figura C.30: Incluir cabecera `poll.h`

   - En `>Additional source code` incluir la definición del tipo `pollfd`:

```
typedef struct pollfd pollfd_t;
```

Código C.2: Inicialización del tipo pollfd

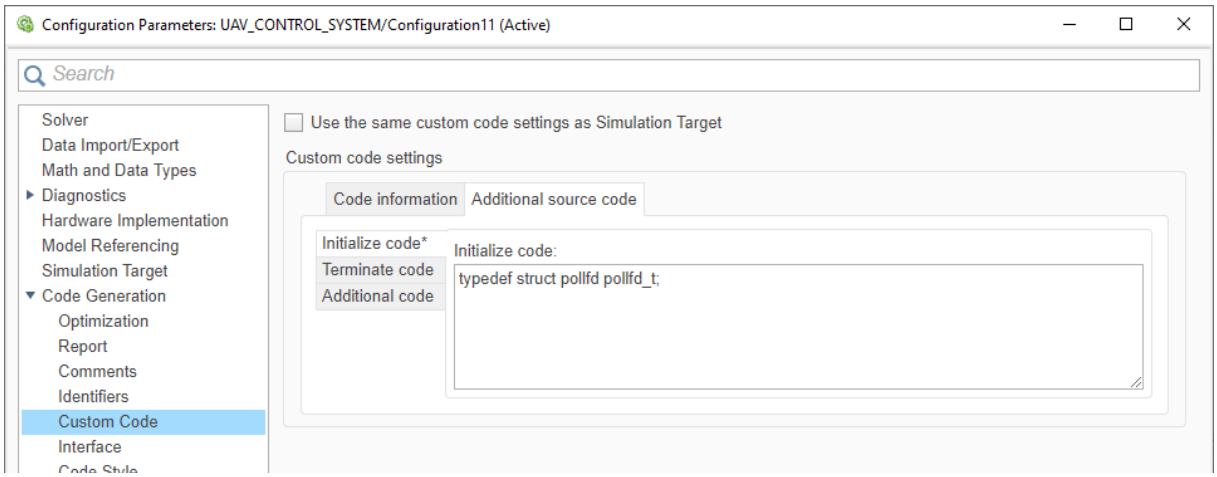


Figura C.31: Incluir la definición del tipo `pollfd`

4. Una vez configurado, para subir un nuevo firmware, hay que seleccionar dentro de Hardware: Run on board (External mode) y clickar en "Build Deploy and Start". Asegúrese previamente de haber ejecutado el script `CONFIG_UAV.m`)

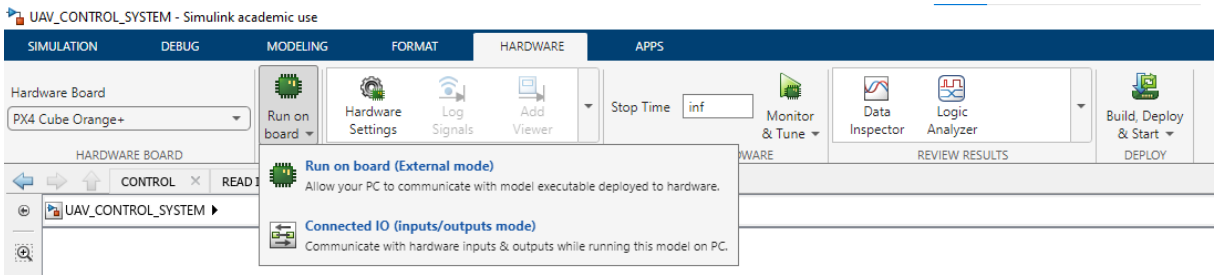


Figura C.32: External mode y Build Deploy & Start

Cuando haya terminado de construirse el firmware le aparecerá de nuevo la pestaña de conectar y desconectar. Tal y como indica dicha ventana, debe desconectar el cable, clickar en “OK” y reconectar el cable.

## C.5. Configuración en QGroundControl

Para monitorizar las variables de interés seleccionadas en el bloque PX4 uLOG, necesitamos comunicar el Cube Orange+ con la estación de tierra (QGroundControl). Para ello, debe dirigirse a:

`QGroundControl >Application Settings >Comm Links`

Allí debe añadir (Icono Add) una nueva configuración:

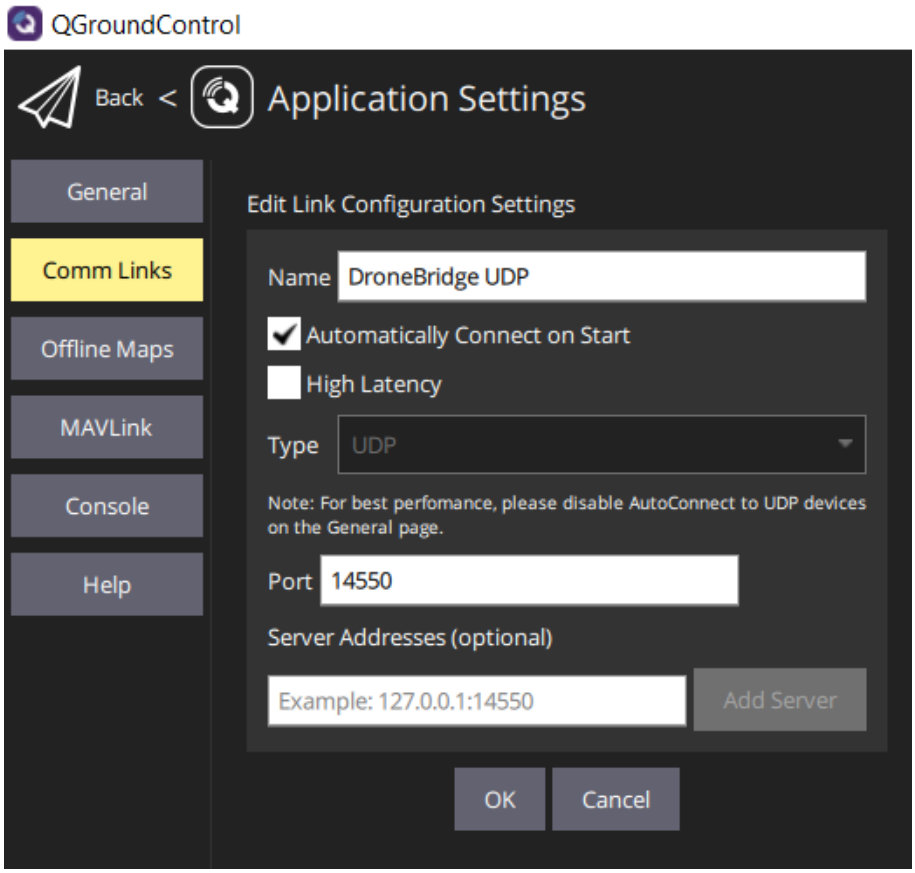


Figura C.33: Configuración conexión UDP en QGroundControl

Cuando haya terminado, pulse en OK y en Connect. Cuando se conecte la batería, se encenderá un led rojo en el ESP32 y se le habilitará un punto de acceso a internet llamado: DroneBridge for ESP32. Selecciónelo, y ya estará todo configurado.

De esta forma, en: `QGroundControl >Analize tools >MavLink console` podrá monitorizar las variables de interés.

# Apéndice D

# Construcción de Mensajes MAVLink

En este anexo se detalla el proceso a realizar para crear mensajes MAVLink, poder leerlos en tiempo real, y posteriormente realizar un análisis rápido de las medidas realizadas durante el vuelo.

## D.1. Creación de un mensaje

Para crear el mensaje, debe seguirse la siguiente estructura en la pestaña de Command Window de Matlab:

`createPX4uORBMessage(messageName, uORBField)`

Es importante que el nombre del mensaje empiece en mayúsculas, ya que dará error en caso contrario. Las variables pueden ser de cualquier tipo, sean enteros, flotantes, vectores... Véase Código D.1.

```
>> createPX4uORBMessage('Intentov', 'uint8 control_mode', 'uint8
   control_status', 'uint8 current_status_sys', 'uint8 motor_mode',
    'float64[4] ch_pu', 'float64[3] euler_ang_ref','double[4]
   emisora_in','double[4] emisora_out', 'uint16[4] pwm', 'double
   sensor_dist','boolean safety_sw', 'float64 aux1', 'float64 aux2'
   ,'float64 aux3')
```

Código D.1: Creación de un mensaje uORB

Al ejecutar el comando, saldrá el siguiente aviso (Código D.2):

```
The custom uORB message "Intentov.msg" is successfully included in
   the PX4 firmware.
To use the message, set up PX4 firmware using the Hardware Setup
   screens and build the PX4 firmware for the selected CMake.
To log the message in ulog format, update the logger_topics.txt in
   SD Card.
```

Código D.2: Confirmación en Command Window de MATLAB

Una vez realizados los pasos de instalación del firmware (véase Anexo C para más información), el fichero .msg debe aparecer en el directorio /PX4-Autopilots/msg y aparecerá en el listado de mensajes del archivo CMakeLists.txt en ese mismo directorio. De esta forma Simulink reconoce el mensaje.

En Simulink, dentro de la librería de UAV Toolbox Support Package for PX4 Autopilots, se encuentra el bloque utilizado para almacenar las variables dentro del mensaje. El bloque se llama uLog, y dentro del bloque se debe incluir el Topic que se acaba de crear. El nombre debe ser idéntico al que se puso, respetando las mayúsculas y minúsculas (Figura D.1).

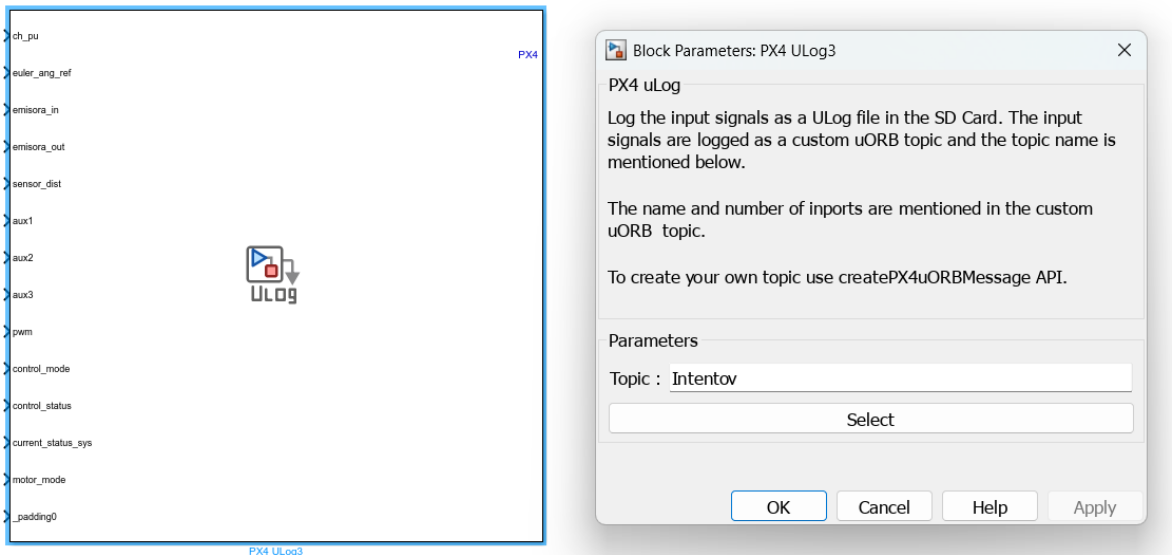


Figura D.1: Configuración del bloque uLog

Una vez conectadas las variables en su puerto correspondiente, y descargado el diagrama en el Cube Orange+, se puede proceder a la lectura de los mensajes.

## D.2. Lectura de los mensajes

Para la lectura de los mensajes debemos abrir la consola de MAVLink en QGroundControl.

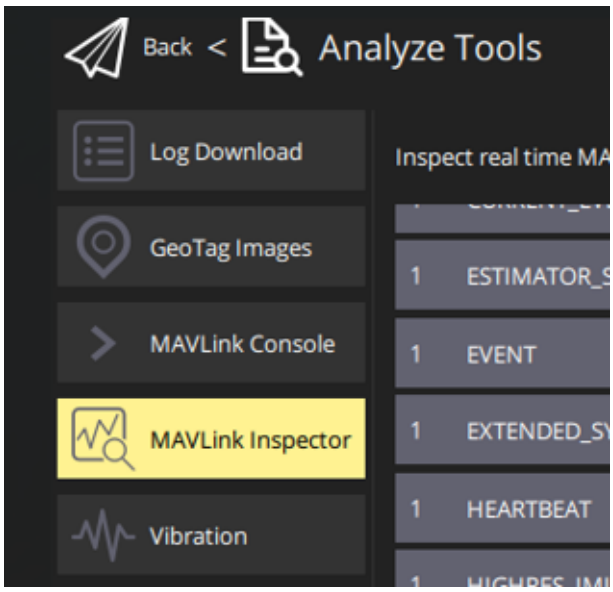


Figura D.2: MAVLink en QGroundControl

En MAVLink Inspector se pueden leer los mensajes predeterminados, como el HEARTBEAT, ATTITUDE y la lectura de los canales de la emisora. En MAVLink Console se pueden ver los mensajes personalizados, ejecutando un sencillo comando *listener messageName*. Importante que el nombre esté escrito en minúsculas, de lo contrario no lo reconocerá. Cada vez que se ejecute el comando, los valores se actualizarán según se haya programado.

```
nsh> listener intentov

TOPIC: intentov
 intentov
    timestamp: 59949425 (0.002207 seconds ago)
    ch_pu: [0.000000, 0.000000, 0.000000, 0.000000]
    euler_ang_ref: [0.000000, 0.000000, 0.000000]
    emisora_in: [1500.000000, 1500.000000, 1000.000000, 1500.000000]
    emisora_out: [1500.000000, 1500.000000, 1000.000000, 1500.000000]
    sensor_dist: 0.000000
    aux1: 0.000000
    aux2: 0.000000
    aux3: 0.000000
    pwm: [1000, 1000, 1000, 1000]
    control_mode: 0
    control_status: 1
    current_status_sys: 2
    motor_mode: 0
```

Figura D.3: Ejemplo de un mensaje MAVLink personalizado

Es aconsejable ejecutar en primer lugar *logger status*. Este comando indicará si la información se está guardando en la tarjeta SD (y dónde) o no.

```
nsh> logger status
INFO  [logger] Running in mode: all
INFO  [logger] Number of subscriptions: 182 (5824 bytes)
INFO  [logger] Full File Logging Running:
INFO  [logger] Log file: /fs/microsd/log/sess263/log100.ulg
INFO  [logger] Wrote 3.15 MiB (avg 62.56 KiB/s)
INFO  [logger] Since last status: dropouts: 2 (max len: 1.410 s), max used buffer: 36365 / 65536 B
```

Figura D.4: Comando logger status

Puede ser que en vez de devolver información como la de la Figura D.4, devuelva el aviso *not logging*. Este fallo es habitual, y es probable que sea debido a que la tarjeta SD esté llena. Puede comprobarse fácilmente ejecutando el comando *logger start*. Si la tarjeta está llena, saldrá una advertencia en la parte superior de la pantalla indicándolo. Para solucionarlo, basta con extraerla del Cube Orange+, borrar el contenido de la carpeta /log e insertarla de nuevo. El problema debería estar solucionado.

Es recomendable ejecutar *logger stop* si no se van a leer mensajes MAVLink pero sí se va a mantener el dron conectado a QGroundControl. Si luego se desea continuar almacenando información, solo es necesario ejecutar *logger start*.

Si solo se están realizando pruebas de los componentes o la respuesta de los motores, es posible que resulte más cómodo configurar el parámetro SD_LOG_MODE a 2, para que guarde información desde el momento que se conecta hasta que se apaga. Si, por el contrario, se van a realizar pruebas de vuelo, entonces sería más indicado configurarlo a 0 (el predeterminado), ya que solo guardará la información que haya tenido lugar desde que se armaron los motores hasta que se desarmaron.