Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Desarrollo de un dron con capacidad de transmisión de vídeo, control y procesamiento desde un ordenador

Autor:

Diez Pajón, Álex

Tutor(es):

Duque Domingo, Jaime
Departamento de ingeniería de Sistemas y Automática

Valladolid, marzo 2025.

Resumen

Este Trabajo de Fin de Grado desarrolla un dron capaz de ser controlado desde un ordenador, y transmitir y procesar vídeo en tiempo real. Partiendo de un kit de montaje y un controlador Pixhawk, el software se basa en ArduPilot, MAVLink y DroneKit, permitiendo la comunicación bidireccional para recibir telemetría y enviar comandos de vuelo. También se añade un sistema FPV compuesto por cámara RunCam y transmisor TBS Unify para enviar la señal de vídeo a tierra. Adicionalmente, se integra OpenCV y YOLOv8 para analizar imágenes en tiempo real y actuar ante detecciones cercanas. El proyecto abarca el montaje, la calibración y el desarrollo de software de control, tanto en simulación SITL en Ubuntu, como en un entorno real, sirviendo como guía práctica para quienes deseen replicar o ampliar estas capacidades.

Palabras clave

Dron, Control Remoto, Control por ordenador, Visión artificial, YOLO

Abstract

This Final Degree Project develops a drone capable of being controlled from a computer, while transmitting and processing real-time vídeo. Built from a DIY kit and a Pixhawk flight controller, the software is based on ArduPilot, MAVLink, and DroneKit, enabling bidirectional communication to receive telemetry and send flight commands. An FPV system composed of a RunCam camera and a TBS Unify transmitter is also added to stream vídeo to the ground. Additionally, OpenCV and YOLOv8 are integrated to analyze images in real time and respond to nearby detections. The project covers assembly, calibration, and control software development in both SITL simulation and real-world environments, serving as a practical guide for those wishing to replicate or expand these capabilities.

Keywords

Drone, Remote Control, Computer-Based Control, Computer Vision, YOLO

Tabla de contenido

Índice de Ilustraciones

Índice de Tablas

CAPÍTULO 1: INTRODUCCIÓN

1.1 Contexto del proyecto

Este proyecto representa un desafío técnico, tanto practico como teórico, sobre la integración de hardware y software, aunando diferentes aspectos de disciplinas como electrónica, automática y programación.

En la parte de hardware tenemos la construcción de un dron basado en un kit de iniciación, incorporando dispositivos extra como el sistema de transmisión de vídeo y el sistema de comunicaciones, enfocado a la recepción de datos de telemetría y envío de órdenes desde un ordenador.

En la parte del software se realizará el diseño, desarrollo e implementación de dos procesos, que implican disciplinas como telecomunicaciones, control, tratamiento de datos, visión e inteligencia artificial.

El primero nos permitirá la interacción entre el dron y el ordenador, dándonos la capacidad de enviar ordenes en forma de comandos, ya sean de arranque o movimientos. Automatizando así las acciones del dron y convirtiendo nuestro ordenador en un control remoto.

El segundo proceso consiste en la recepción y tratamiento en tiempo real de la transmisión de vídeo recibida desde el dron. Usando librerías de visión artificial y redes neuronales convolucionales de detección de objetos, para así automatizar ordenes en base a la situación del entorno alrededor del dron.

1.2 Motivación

Este proyecto despertó mi interés debido a las múltiples posibilidades que ofrece en términos de integración de tecnologías. En particular, el control remoto del dron desde un ordenador representa un aspecto de gran relevancia, el cual combinado con técnicas de visión e inteligencia artificial amplía significativamente su potencial, permitiendo no solo la interacción remota, sino también la capacidad de procesar y responder al entorno de manera autónoma.

El hecho de trabajar con un vehículo volador añade un nivel adicional de complejidad y desafío, haciendo que este Trabajo de Fin de Grado sea una oportunidad ideal para aplicar y consolidar habilidades adquiridas en mi titulación.

Más allá del atractivo tecnológico, al abarcar un enfoque multidisciplinar se convierte en un excelente ejercicio práctico para integrar diversas áreas del conocimiento dentro de un mismo desarrollo. No se trata únicamente de trabajar con diferentes tecnologías de forma individual, sino de generar una

plataforma que las combine, lo que resulta en una formación completa y alineada con mi desarrollo profesional en el ámbito de la ingeniería.

1.3 Objetivos

Hemos denominado objetivos a las etapas clave que iremos completando a lo largo de la realización de este proyecto, cada una de las cuales constituye una tarea a completar y un aspecto a aprender, son estas:

- **Estudio de los componentes**, garantizando su compatibilidad, ya sea directa entre dispositivos o indirecta mediante soluciones alternativas, como el uso de adaptadores eléctricos o conversores de señal. Esta etapa permitirá profundizar en la selección de hardware, el análisis de especificaciones técnicas y la identificación de posibles limitaciones o conflictos en la integración de los sistemas.

- **Montaje y configuración del dron**, asegurando la correcta integración de los componentes electrónicos y de propulsión. Durante esta fase, se adquirirán conocimientos sobre ensamblaje, distribución del peso, conexiones eléctricas y posibles interferencias entre dispositivos, además de la configuración inicial del controlador de vuelo y emisor/receptor para control remoto manual, con su correspondiente comprensión de los parámetros de vuelo y control.

- **Implementación del sistema de transmisión de vídeo**, estableciendo la conexión entre la cámara FPV y el ordenador receptor. Se aprenderán conceptos sobre señales de vídeo analógicas, frecuencias de transmisión y métodos para minimizar interferencias y latencias en la transmisión.

- **Desarrollo del sistema de control remoto**, permitiendo el envío de comandos desde el ordenador al dron mediante protocolos de comunicación específicos. Esto implicará el aprendizaje del protocolo MAVLink, la estructura de los comandos de control y su implementación mediante software.

- **Recepción y procesamiento de la señal de vídeo**, aplicando librerías de visión artificial para la detección y análisis del entorno. En esta etapa se trabajará con herramientas como OpenCV y modelos de procesamiento de imágenes, desarrollando habilidades en manipulación y tratamiento de datos visuales en tiempo real.

- **Automatización de respuestas en base a la detección de objetos**, integrando modelos de inteligencia artificial para la toma de decisiones autónoma. Se explorarán redes neuronales convolucionales y su

aplicación en el reconocimiento y categorización de elementos dentro del entorno del dron.

- **Documentación del proceso y análisis de resultados**, recopilando los aprendizajes adquiridos y evaluando mejoras futuras en el diseño del sistema. Se desarrollarán competencias en la redacción técnica y en la interpretación de datos experimentales para la mejora continua del proyecto.

1.4 Elementos del proyecto

El desarrollo de este proyecto requiere la integración de diversos componentes de hardware y software, cada uno con un papel específico en la construcción, control y transmisión de datos del dron. A continuación, una descripción general de los principales elementos utilizados:

Hardware

El dron se basa en un **kit de iniciación F450**, que incluye un chasis de cuadricóptero que proporciona la estructura base para la instalación de los componentes electrónicos y de propulsión, además de un tren de aterrizaje. También entre otros dispositivos que veremos después, el kit también nos proporciona los **motores sin escobillas**, los **controladores electrónicos de velocidad (ESCs)** y el sistema de alimentación mediante una **batería LiPo**,

El **controlador de vuelo Pixhawk 2.4.8** es el núcleo del sistema, encargado de gestionar la estabilización y el comportamiento en vuelo mediante sensores inerciales y GPS. Este dispositivo es compatible con sistemas avanzados de navegación y permite la interacción con software de control desde un ordenador.

Para la transmisión de vídeo, el dron cuenta con una **cámara FPV RunCam Nano 2**, la cual envía imágenes en tiempo real a través del **transmisor de vídeo TBS Unify Pro 5.8GHz**. La señal es recibida en tierra mediante un **receptor de vídeo Eachine ROTG02**, que permite visualizar el flujo de vídeo en un ordenador para su posterior procesamiento.

El sistema de comunicación y telemetría se gestiona mediante un **módulo Holybro SiK Telemetry Radio**, que facilita el intercambio de datos entre el dron y el ordenador, permitiendo el control remoto y la monitorización de parámetros en tiempo real.

Adicionalmente, se han incorporado elementos auxiliares como el **adaptador XT60 a JST** para la correcta alimentación de los dispositivos, un **zumbador de alerta**, un **GPS M8N** para mejorar la precisión en la navegación y una **emisora**

FS-i6X con su receptor FS-iA6B, que permite el control manual del dron si fuese necesario.

Para puntualizar, el kit de iniciación proporciona lo indispensable para el montaje de un dron completamente operativo y controlado de forma remota y manual con el mando (Emisora FS-i6x), es decir, tanto el sistema de vídeo (Transmisor/Runcam) como el de comunicaciones (Holybro SiK Telemetry Radio) son componentes y funcionalidades adicionales, externas al kit. Al igual que el adaptador XT60-JST.

Software

El control del dron y la transmisión de datos requieren la implementación de software especializado, que permite la configuración y comunicación con los sistemas a bordo:

- **Mission Planner**: Herramienta utilizada para la configuración del Pixhawk, permitiendo calibrar sensores, definir parámetros de vuelo y monitorizar datos en tiempo real.
- **MAVLink**: Protocolo de comunicación empleado para el intercambio de datos entre el ordenador y el dron, facilitando el envío de comandos y la recepción de telemetría.
- **DroneKit**: API en Python que permite programar y automatizar la interacción con el dron, posibilitando la ejecución de trayectorias, movimientos específicos y respuestas basadas en eventos detectados en el entorno.
- **DroneKit-SITL**: Software de simulación de drones basados en los modelos provistos en el repositorio de GitHub de ArduPilot, permitiendo realizar pruebas de código seguras.
- **Librerías de visión artificial**: Se emplearán herramientas como OpenCV y modelos de redes neuronales convolucionales (YOLOv8) para el procesamiento y análisis de la señal de vídeo en tiempo real, permitiendo la detección de objetos y la toma de decisiones autónoma.

1.5 Estructura de la memoria

Este documento se divide en capítulos, llegados a este punto ya se ha visto la introducción, que nos pone en contexto sobre qué vamos a realizar en este Trabajo de Fin de Grado, cómo y por qué. A continuación, se va a describir en que van a consistir el resto de los capítulos y el orden que va a tener la redacción de este proyecto.

- **Capítulo 2 - Estado del Arte de los Drones**: Se podría decir que consiste en una introducción o repaso general sobre los drones, su definición, tipos y principales aplicaciones. Además, se analiza su evolución histórica y se profundiza en las tecnologías de transmisión de vídeo y los principales softwares de control, como ArduPilot y MAVLink. Finalmente, se incluyen ejemplos de proyectos similares y sus diferencias con el presente trabajo.

- **Capítulo 3 - Tecnologías del Proyecto:** para empezar este capítulo se plasma el primer objetivo de este proyecto, el estudio de componentes. Detalla el hardware utilizado en el dron, como el chasis F450, el controlador Pixhawk, el sistema de transmisión de vídeo y la telemetría. Después de esto, también se describen las herramientas de software empleadas, incluyendo Mission Planner, MAVLink, DroneKit y lo correspondiente a Visión Artificial.

- **Capítulo 4 - Montaje, Integración y Calibración:** Se explica el proceso de ensamblaje del dron, la instalación de los distintos dispositivos y la distribución de estos. Asimismo, se detallan las configuraciones necesarias, como la calibración del controlador de vuelo Pixhawk y la emisora, así como pruebas iniciales de vuelo.

- **Capítulo 5 - Desarrollo de Software:** Se describe la implementación del control del dron desde el ordenador mediante MAVLink y DroneKit, así como la programación de comandos automatizados. Además, se aborda la recepción y visualización del vídeo en tiempo real.

- **Capítulo 6 - Gestión de Trabajo y Costes:** Se documentan las distintas fases del proyecto, desde la investigación y elección de componentes hasta la calibración y pruebas finales. También se incluye un análisis económico con los recursos empleados.

- **Capítulo 7 – Conclusiones y Futuros Desarrollos:** Además de las conclusiones del desarrollo de este TFG, se plantean posibles mejoras futuras, como la optimización de la autonomía del dron o el refinamiento de los algoritmos de procesamiento de imágenes.

Esta estructura permite abordar el desarrollo del proyecto de manera organizada y sistemática, asegurando la correcta documentación de cada fase y proporcionando una visión completa del trabajo realizado.

CAPÍTULO 2: ESTADO DEL ARTE DE LOS DRONES

2.1 Introducción a los drones: Definición, tipos y usos principales

Definición de un dron

Un dron, también conocido como vehículo aéreo no tripulado, es una aeronave que puede operar sin la necesidad de un piloto a bordo. Su control puede ser remoto (RPA, Remoted Piloted Aircaft), semiautónomo o completamente autónomo (UAV, Unmanned Aerial Vehicle) mediante sistemas de navegación y control computarizados. Estos dispositivos han experimentado un crecimiento significativo en los últimos años, impulsados por avances en tecnologías de comunicación, sensores y procesamiento de datos.

Funcionamiento

Los drones funcionan mediante un conjunto de motores eléctricos que impulsan el dispositivo a través de un **sistema multi-hélice** que le proporciona gran estabilidad en el momento de despegar, volar y aterrizar.

Suelen contar con baterías de gran autonomía, por lo que son capaces de mantenerse en el aire por bastante tiempo, antes de requerir un cambio o recarga de las baterías. De modo que son de gran utilidad para diversos usos.

En el caso de los **RPAs**, son **operados desde una estación remota** en tierra a través de un enlace de comunicaciones, que envía señales al dron para que realice las maniobras que el piloto desea que ejecute.

Por otra parte, aquellos que no son controlados de manera remota, **UAVs**, se enlazan con un software que **usa un GPS para programar vuelos controlados** y éstos operan de forma autónoma. En esta variedad se encuentran los drones usados para la agricultura, que normalmente se programan para que realicen diversas tareas en los cultivos.

Algunos de ellos cuentan con múltiples sensores que les permiten detectar obstáculos en la ruta, capturar imágenes, hacer filmaciones con drones o accesorios adicionales, como por ejemplo luces que permiten volar dron de noche, de modo que su funcionamiento específico varía según sean las características del dron.

Tipos de drones

Los drones se pueden clasificar de diversas formas en función de su estructura, propulsión y aplicación. Algunas de las categorías más comunes incluyen:

- **Según su número de rotores:**

  - Multirrotores: Son los más populares en aplicaciones comerciales y recreativas. Incluyen cuadricópteros (4 hélices), hexacópteros (6 hélices) y octocópteros (8 hélices). Son fáciles de maniobrar y estabilizar, aunque su autonomía es limitada debido al alto consumo energético.
  - Helicópteros: Equipados con un solo rotor principal y uno de compensación en la cola, ofrecen mayor autonomía que los multirrotores, pero requieren mayor complejidad en el control.
  - Aviones de ala fija: Se asemejan a los aviones convencionales y pueden cubrir grandes distancias con menor consumo de energía. Son ideales para aplicaciones de cartografía y vigilancia.
  - Drones híbridos: Combinan características de multirrotores y aviones de ala fija, permitiendo despegues y aterrizajes verticales junto con vuelos eficientes a largas distancias.

- **Según su uso:**
  - Recreativos: Destinados a aficionados y competiciones de carreras FPV (First Person View).
  - Industriales y comerciales: Empleados en topografía, agricultura de precisión, inspección de infraestructuras y logística.
  - Militares y de seguridad: Utilizados en misiones de reconocimiento, vigilancia, ataques estratégicos y rescates.
  - Investigación y exploración: Aplicados en estudios científicos, arqueología y exploración de entornos hostiles como volcanes o el espacio.

Usos principales de los drones

Los drones han transformado múltiples sectores gracias a su versatilidad. Algunos de sus principales usos incluyen:

- **Fotografía y vídeo**: Producción de contenido audiovisual en cine, televisión y publicidad.
- **Agricultura de precisión**: Monitorización de cultivos mediante sensores multiespectrales y fumigación de cultivos con mayor eficiencia.
- **Seguridad y vigilancia**: Uso en fuerzas de seguridad, supervisión de eventos y control de fronteras.

- **Inspección industrial**: Evaluación de infraestructuras como líneas eléctricas, oleoductos y parques eólicos.
- **Entrega de paquetes**: Empresas como Amazon y UPS han experimentado con el uso de drones para la entrega rápida de productos.
- **Búsqueda y rescate**: Localización de personas en desastres naturales o accidentes mediante cámaras térmicas y sensores.
- **Exploración y cartografía**: Creación de mapas en 3D de terrenos inaccesibles o peligrosos.
- **Ciencia y medio ambiente**: Monitoreo de especies en peligro, estudio de la calidad del aire y seguimiento de fenómenos meteorológicos.
- **Entretenimiento**: Los drones tienen una amplia variedad de aplicaciones en este ámbito, desde el uso recreativo por parte de aficionados y competiciones de carreras FPV, hasta espectáculos de luces a gran escala, donde cientos de drones sincronizados crean impresionantes coreografías en el cielo.

Documentación de Interés

IDC Drones. (2022). *Que es un dron y para qué sirve*. Recuperado de https://umilesgroup.com/que-es-un-dron-y-para-que-sirve/

Ferrovial. (s.f). *Drones: Qué son, cómo funcionan, normativa española*. Recuperado de https://www.ferrovial.com/es/recursos/drones/

Ferrovial. (s.f). *Drones: qué son, tipos de drones y para qué sirven*. Recuperado de https://www.ferrovial.com/es-es/innovacion/tecnologias/drones/

2.2 Historia y evolución de los drones

Introducción

Los drones, o vehículos aéreos no tripulados (UAV, por sus siglas en inglés), han experimentado un desarrollo significativo desde sus inicios hasta convertirse en herramientas esenciales en diversos sectores. Su evolución ha estado marcada por avances tecnológicos y cambios en sus aplicaciones, desde usos militares hasta comerciales y recreativos.

Orígenes de los drones

Aunque los primeros conceptos de vuelo no tripulado se remontan a la antigüedad, fue en el siglo XX cuando comenzaron a desarrollarse los primeros prototipos de drones modernos. Durante la Primera Guerra Mundial, se

realizaron intentos de crear aeronaves no tripuladas con fines militares. Por ejemplo, en 1916, Archibald Low en Gran Bretaña desarrolló prototipos de aviones no tripulados, y en 1917, el francés Max Boucher logró volar un avión "Voisin" sin piloto a más de 500 metros de distancia y 50 metros de altura.

Avances tecnológicos y aplicaciones militares

En la década de 1950, se realizaron avances significativos en el campo de los drones. El desarrollo de tecnologías como los sistemas de control remoto y la miniaturización de componentes electrónicos permitió crear los primeros prototipos de aviones no tripulados tal como los conocemos hoy en día. Estos primeros intentos de vuelo no tripulado sentaron las bases para futuras investigaciones y desarrollo de la tecnología de drones.

Durante la Segunda Guerra Mundial, se utilizaron drones para misiones de reconocimiento y entrenamiento de artilleros antiaéreos. En las décadas siguientes, especialmente durante la Guerra Fría, los drones se convirtieron en herramientas esenciales para misiones de vigilancia y reconocimiento, impulsando el desarrollo de tecnologías avanzadas en navegación y control.

Expansión a aplicaciones civiles y comerciales

A medida que la tecnología avanzaba, los drones comenzaron a encontrar aplicaciones más allá del ámbito militar. En las últimas décadas, se han utilizado en sectores como la agricultura de precisión, la inspección de infraestructuras, la entrega de paquetes y la fotografía aérea. Su capacidad para acceder a áreas de difícil acceso y recopilar datos de manera eficiente ha transformado numerosas industrias.

Drones en la sociedad actual

Además de sus aplicaciones industriales, los drones han ganado popularidad en el ámbito recreativo y deportivo. Muchos entusiastas los utilizan para capturar fotografías y vídeos aéreos, participar en carreras de drones y explorar nuevas formas de creatividad. Esta accesibilidad ha democratizado el uso de la tecnología de drones, haciéndola más cercana al público general.

Futuro de los drones

La industria de los drones continúa evolucionando rápidamente. Las tendencias actuales incluyen avances en autonomía de vuelo, integración de inteligencia artificial y expansión en aplicaciones como el transporte de pasajeros y la vigilancia ambiental. Estas innovaciones prometen ampliar aún más el impacto de los drones en la sociedad y la economía global.

Documentación de Interés

IDC Drones. (s.f). *La historia de los drones: Desde los inicios hasta la actualidad*. Recuperado de https://idc.apddrones.com/educacion/la-historia-de-los-drones-desde-los-inicios-hasta-la-actualidad/

UMILES Group. (2024). *La historia y el futuro de los drones*. Recuperado de https://umilesgroup.com/la-historia-y-el-futuro-de-los-drones/

Aviation Group. (s.f). *Drones: ¿Quién los inventó?*. Recuperado de https://www.aviationgroup.es/actualidad/drones-quien-invento/

National Geographic. (2024). *Drones: Un arma con mucha historia*. Recuperado de https://historia.nationalgeographic.com.es/a/drones-arma-mucha-historia_21118/

2.3 Sistemas de transmisión de vídeo en drones

La transmisión de vídeo en tiempo real es esencial para muchas aplicaciones de drones, especialmente en operaciones que requieren una vista en primera persona (FPV, por sus siglas en inglés). Existen diversas tecnologías y protocolos que permiten esta transmisión, cada una con características específicas en cuanto a calidad de imagen, latencia, alcance y frecuencia de operación.

Tecnologías de transmisión de vídeo

Los sistemas de transmisión de vídeo para drones se dividen principalmente en dos categorías: analógicos y digitales.

Sistemas analógicos: Tradicionalmente, la transmisión de vídeo en drones se ha realizado mediante sistemas analógicos que operan en la banda de 5.8 GHz. Estos sistemas ofrecen una latencia muy baja, lo que es crucial para aplicaciones como las carreras de drones. Sin embargo, la calidad de imagen es limitada, generalmente no alcanzando resolución HD. Es común que los vídeos de alta calidad que se ven en plataformas como YouTube sean grabados con cámaras HD adicionales, diferentes de las utilizadas para la transmisión FPV en tiempo real.

Sistemas digitales: Con los avances tecnológicos, han surgido sistemas de transmisión de vídeo digitales que ofrecen mejoras significativas en calidad de imagen y alcance. Por ejemplo, el sistema DJI FPV proporciona una experiencia de vuelo inmersiva con transmisión de vídeo en alta definición y baja latencia. Estos sistemas permiten una visualización más clara y detallada, lo que es beneficioso para aplicaciones que requieren una mayor precisión visual.

Frecuencias de transmisión

La elección de la frecuencia de transmisión es crucial, ya que afecta el alcance, la penetración de la señal y la calidad de la transmisión. Las frecuencias más comunes utilizadas en sistemas FPV son:

- **900 MHz:** Ofrece un buen alcance y mejor penetración a través de obstáculos como árboles y edificios. Sin embargo, requiere antenas más grandes y puede estar sujeta a regulaciones específicas según la región.
- **1.2 GHz:** Proporciona un equilibrio entre alcance y penetración, pero su uso puede estar restringido en algunas áreas debido a interferencias con otras comunicaciones.
- **2.4 GHz:** Es una frecuencia común que ofrece un buen equilibrio entre alcance y penetración de señal. Sin embargo, puede estar sujeta a interferencias debido a su uso en otras aplicaciones, como Wi-Fi y Bluetooth.
- **5.8 GHz:** Es la frecuencia más utilizada para sistemas FPV debido a su menor interferencia con sistemas de control de drones que operan en 2.4 GHz. Ofrece una buena calidad de señal en entornos despejados, pero su capacidad de penetración a través de obstáculos es limitada.

Receptores y antenas

Los receptores de vídeo en tierra son componentes esenciales del sistema FPV, ya que reciben la señal transmitida desde el dron. La elección del receptor debe ser compatible con la frecuencia y el tipo de señal (analógica o digital) utilizada por el transmisor del dron.

Las antenas también juegan un papel crucial en la calidad de la transmisión. Las antenas omnidireccionales ofrecen una cobertura amplia y son ideales para vuelos a corta distancia, mientras que las antenas direccionales, como las de tipo patch o helicoidales, proporcionan un mayor alcance y mejoran la calidad de la señal en vuelos de larga distancia, aunque requieren una orientación más precisa hacia el dron.

Protocolos de transmisión

En sistemas digitales, se emplean protocolos avanzados para mejorar la eficiencia y la calidad de la transmisión de vídeo. Por ejemplo, la tecnología de multiplexación por división de frecuencia ortogonal (OFDM) es común en sistemas de transmisión de datos de alta velocidad, ya que permite transmitir grandes cantidades de datos en un ancho de banda estrecho y es resistente a interferencias.

Además, algunos sistemas digitales incorporan protocolos de corrección de errores y técnicas de modulación avanzadas para garantizar una transmisión de vídeo estable y de alta calidad, incluso en entornos con interferencias o a largas distancias.

Consideraciones finales

La selección de la tecnología de transmisión de vídeo, la frecuencia de operación, los receptores y los protocolos adecuados depende de las necesidades específicas de la aplicación del dron. Es fundamental considerar factores como la latencia, la calidad de imagen, el alcance y las condiciones del entorno de operación para garantizar una experiencia FPV óptima y segura.

Documentación de Interés

Prometec. (s.f). *Cómo elegir el mejor VTX para tu dron FPV*. Recuperado de https://www.prometec.net/elegir-vtx-para-dron/

DJI. (s.f). *DJI FPV: Sistema de vuelo inmersivo*. Recuperado de https://www.dji.com/es/fpv

ADV Dron. (2023). *Frecuencias de transmisión en drones FPV*. Recuperado de https://advdron.com/emisoras-para-drones-fpv/

iWave Comms. (s.f). *Tecnologías de transmisión inalámbrica en UAVs*. Recuperado de https://www.iwavecomms.com/es/news/what-are-the-common-unmanned-aerial-vehicle-uav-wireless-transmission-technologies/

2.4 Software de control de drones

El desarrollo y operación de drones requieren de software especializado que permita el control autónomo, la planificación de misiones y la comunicación entre los distintos componentes del sistema. A continuación, se describen algunas de las plataformas y protocolos más destacados en este ámbito.

ArduPilot

ArduPilot es una plataforma de piloto automático de código abierto que ofrece soluciones para una amplia variedad de vehículos no tripulados, incluyendo multirrotores, aviones de ala fija, helicópteros, rovers y submarinos. Su flexibilidad y robustez la han convertido en una opción popular tanto para aficionados como para aplicaciones profesionales. ArduPilot es compatible con diversos controladores de vuelo y se integra con múltiples sensores y sistemas de navegación.

PX4

PX4 es otra plataforma de piloto automático de código abierto que se destaca por su arquitectura modular y su enfoque en la seguridad y la eficiencia. Soporta una amplia gama de tipos de vehículos y ofrece modos de vuelo que van desde el control manual hasta la autonomía completa. PX4 se integra con el software de estación terrestre QGroundControl y utiliza el protocolo MAVLink para la comunicación.

MAVLink

MAVLink (Micro Air Vehicle Link) es un protocolo de comunicación liviano diseñado para permitir la comunicación entre vehículos aéreos no tripulados y estaciones terrestres. Es ampliamente utilizado en la comunidad de drones debido a su eficiencia y flexibilidad. MAVLink soporta una variedad de autopilotos, incluyendo ArduPilot y PX4, y es compatible con múltiples aplicaciones de control en tierra.

Mission Planner

Mission Planner es una aplicación de estación terrestre diseñada para interactuar con vehículos que utilizan el firmware ArduPilot. Ofrece herramientas para la planificación de misiones, monitoreo en tiempo real y análisis de datos posteriores al vuelo. Aunque está optimizada para sistemas Windows, también puede ejecutarse en Linux y Android con ciertas consideraciones.

QGroundControl

QGroundControl es una aplicación de estación terrestre que proporciona control de vuelo completo y planificación de misiones para cualquier dron habilitado con MAVLink. Es compatible con múltiples autopilotos, incluyendo PX4 y ArduPilot, y soporta una variedad de tipos de vehículos. QGroundControl está disponible para Windows, macOS, Linux, iOS y Android, y se destaca por su interfaz intuitiva y su enfoque en la facilidad de uso tanto para profesionales como para desarrolladores.

En resumen, la combinación de plataformas de piloto automático como ArduPilot y PX4, protocolos de comunicación como MAVLink y aplicaciones de estación terrestre como Mission Planner y QGroundControl, proporciona un ecosistema robusto y flexible para el control y operación de drones en diversas aplicaciones.

Documentación de Interés

ArduPilot. (s.f). *Documentación oficial de ArduPilot*. Recuperado de https://ardupilot.org/planner/docs/mission-planner-installation.html

PX4 Autopilot. (2024). *Documentación oficial de PX4*. Recuperado de https://en.wikipedia.org/wiki/PX4_autopilot

MAVLink. (s.f). *Especificaciones y documentación técnica de MAVLink*. Recuperado de https://mavlink.io/en/about/implementations.html

QGroundControl. (s.f). *Guía de usuario de QGroundControl*. Recuperado de https://qgroundcontrol.com/

2.5 Ejemplos de proyectos similares y diferencias con el propuesto

Los proyectos expuestos en este apartado están referenciados en la documentación de interés de este apartado.

En el ámbito de los drones con capacidad de transmisión de vídeo y control desde un ordenador, existen diversos proyectos que abordan esta temática desde diferentes perspectivas. A continuación, se describen algunos ejemplos relevantes y se analizan las diferencias con respecto al proyecto propuesto. Todos los enlaces a estos proyectos se encuentran en el apartado correspondiente al 2.5 de la bibliografía

Proyecto 1: Sistema de transmisión de contenidos entre drones y vehículos mediante difusión

Este proyecto, desarrollado en la Universitat Politècnica de València, se centra en el diseño de un sistema para la transmisión de contenidos entre drones y vehículos utilizando técnicas de difusión. El objetivo principal es establecer una comunicación eficiente para compartir información en tiempo real entre estos dispositivos.

Diferencias con el proyecto propuesto:

- **Enfoque de comunicación:** Mientras que este proyecto se enfoca en la transmisión de datos entre drones y vehículos, el proyecto propuesto se centra en la transmisión de vídeo desde el dron hacia un ordenador para su control y procesamiento.
- **Tecnologías utilizadas:** El proyecto de la UPV se basa en técnicas de difusión para la comunicación, mientras que el proyecto propuesto considera el uso de tecnologías como ArduPilot, MAVLink y DroneKit para el control y transmisión de vídeo.

Proyecto 2: Transmisión de vídeo en tiempo real y control de un sistema UAV conectado a redes celulares

Este proyecto desarrolla un prototipo de un sistema UAV que utiliza redes celulares para la transmisión de vídeo en tiempo real y el control del dron. El estudio evalúa el rendimiento del sistema en términos de latencia, fiabilidad y calidad de la transmisión bajo diferentes condiciones de vuelo y configuraciones de red.

Diferencias con el proyecto propuesto:

- **Medio de transmisión:** Este proyecto emplea redes celulares para la transmisión de vídeo y control, mientras que el proyecto propuesto considera el uso de transmisores de vídeo específicos y comunicación mediante protocolos como MAVLink.
- **Objetivo del estudio:** El proyecto mencionado se centra en evaluar el rendimiento de las redes celulares para aplicaciones UAV, mientras que el proyecto propuesto busca desarrollar un sistema completo de control y transmisión de vídeo desde un dron utilizando componentes específicos como el kit HAWK´S WORK F450 y el controlador Pixhawk.

Proyecto 3: Robot de vigilancia con transmisión de vídeo en tiempo real basado en Raspberry Pi

Este proyecto propone el diseño de un robot de vigilancia que utiliza una Raspberry Pi para realizar tareas de detección de movimiento, transmisión de vídeo en tiempo real y control remoto a través de Internet. Incluye funcionalidades como detección de incendios y una aplicación cliente para monitorear y controlar el robot desde cualquier dispositivo con conexión a Internet.

Diferencias con el proyecto propuesto:

- **Tipo de vehículo:** Mientras que este proyecto se centra en un robot terrestre, el proyecto propuesto se enfoca en un vehículo aéreo no tripulado (dron).
- **Plataforma de hardware:** El proyecto del robot de vigilancia utiliza una Raspberry Pi como unidad central de procesamiento, mientras que el proyecto propuesto considera el uso de un controlador de vuelo Pixhawk y otros componentes específicos para drones.

Proyecto 4: Transmisión de vídeo en directo desde drones para aplicaciones comerciales

Este proyecto aborda la transmisión de vídeo en directo desde drones para aplicaciones comerciales, proporcionando información situacional en tiempo real a las partes interesadas. Se enfoca en casos de uso como la seguridad pública, inspecciones industriales y respuesta a emergencias, destacando la importancia de la transmisión de vídeo en tiempo real para mejorar la toma de decisiones.

Diferencias con el proyecto propuesto:

- **Aplicaciones específicas:** Aunque ambos proyectos comparten el interés en la transmisión de vídeo en tiempo real, este proyecto se centra en aplicaciones comerciales específicas, mientras que el proyecto propuesto tiene un enfoque más general en el desarrollo de un dron con capacidades de transmisión de vídeo y control desde un ordenador.
- **Tecnologías y protocolos:** El proyecto mencionado puede utilizar diferentes tecnologías y protocolos para la transmisión de vídeo, mientras que el proyecto propuesto considera el uso de componentes y software específicos como ArduPilot, MAVLink y DroneKit.

En resumen, aunque existen proyectos que abordan la transmisión de vídeo y el control de drones, el proyecto propuesto se distingue por su enfoque en la construcción de un dron desde un kit de iniciación, la integración de una cámara FPV y sistemas de transmisión de vídeo, y el desarrollo de software de control y procesamiento desde un ordenador utilizando tecnologías y componentes específicos.

Documentación de Interés

Universitat Politècnica de València. (2016/2017). *Desarrollo de un sistema de transmisión de contenidos entre drones y vehículos mediante difusión*. Recuperado de https://m.riunet.upv.es/bitstream/handle/10251/86057/Ortiz%20-%20Desarrollo%20de%20un%20sistema%20de%20transmisi%C3%B3n%20de%20contenidos%20entre%20drones%20y%20veh%C3%ADculos%20mediante%20d....pdf?isAllowed=y&sequence=1

Arxiv.org. (2021). *Transmisión de vídeo en tiempo real y control de un sistema UAV conectado a redes celulares*. Recuperado de https://arxiv.org/abs/2101.10736

Arxiv.org. (2022). *Robot de vigilancia con transmisión de vídeo en tiempo real basado en Raspberry Pi*. Recuperado de https://arxiv.org/abs/2209.14130

FlytBase. (2019). *Transmisión de vídeo en directo desde drones para aplicaciones comerciales*. Recuperado de https://www.flytbase.com/es/blog/live-video-feed-from-drones

CAPÍTULO 3: TECNOLOGÍAS DEL PROYECTO

En este capítulo se detallan las características principales de los componentes que se usarán en la construcción de nuestro dron y los softwares para el desarrollo del control por ordenador y recepción y tratamiento de vídeo.

Al final de la descripción de cada elemento se incluirá el enlace de compra en el caso del Hardware y la documentación de Interés en caso del Software.

3.1 Hardware

Los componentes para describir a continuación están distribuidos en apartados según la funcionalidad del dron a la que pertenecen, por eso antes de empezar se expone como se han adquirido estos componentes.

El **kit DIY HAWK'S WORK F450** incluye la mayoría de los componentes esenciales para el ensamblaje del dron, como el chasis **F450**, los motores, los controladores de velocidad (**ESC**), y el controlador de vuelo **Pixhawk 2.4.8 con GPS**, garantizando estabilidad y precisión en el vuelo. Además, este kit incorpora un sistema **RC** que permite el control manual, así como accesorios clave como el **cargador de baterías, módulo de potencia y zumbador**, asegurando un montaje seguro y eficiente.

Su versatilidad lo convierte en una base sólida para la integración de otros sistemas, como la transmisión de vídeo y telemetría, permitiendo dotar al dron de capacidades avanzadas de procesamiento y comunicación en tiempo real.

Ilustración 1: Kit DIY completo

Además de los elementos incluidos en el kit, se han adquirido por separado los siguientes sistemas complementarios:

- **Sistema de transmisión de vídeo**: Compuesto por el **transmisor TBS Unify Pro** y la **cámara FPV RunCam Nano 2**.

- **Sistema de telemetría**: Incorporación del módulo **Holybro SiK Telemetry Radio** para la comunicación en tiempo real con el ordenador.
- **Adaptador XT60-JST**: Utilizado para facilitar la conexión y distribución de la alimentación en el sistema.

Enlace del Kit: HAWK'S WORK. (2025). *Kit de iniciación HAWK´S WORK F450*. Amazon. Recuperado de https://www.amazon.es/dp/B09SZ7LNXB

3.1.1 Chasis y Propulsión

El chasis y el sistema de propulsión son elementos fundamentales en la estabilidad y maniobrabilidad del dron.

Chasis F450

El **chasis F450** es una estructura de cuadricóptero pre-soldado con un diseño en forma de "X", ideal para la estabilidad y distribución de carga en el vuelo. También viene con un tren de aterrizaje que protege los componentes del dron al aterrizar y proporciona espacio para montar cámaras y otros sensores. Sus características principales son:

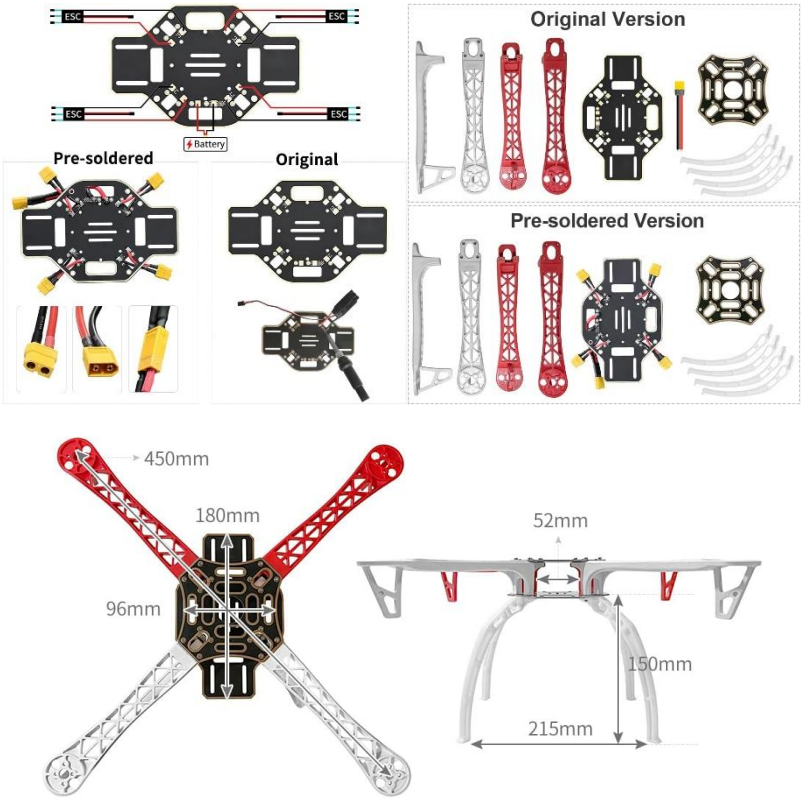


Ilustración 2: Chasis F450

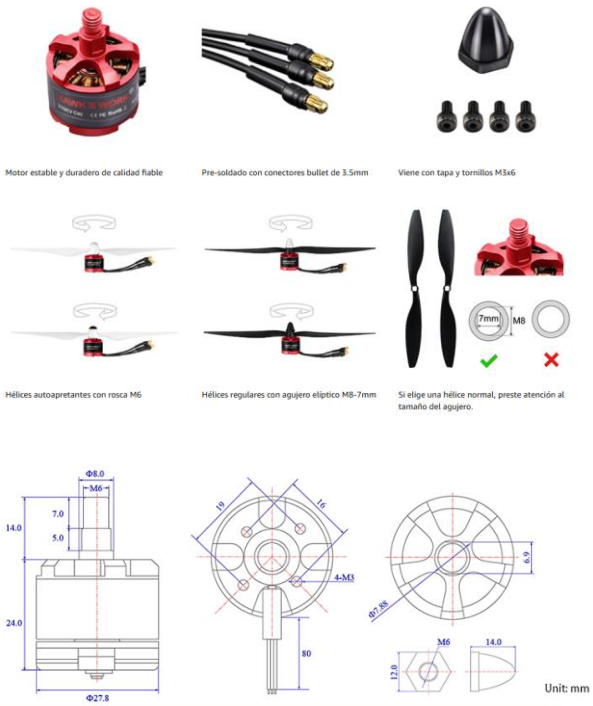
Tabla 1: Características Chasis F450 y Tren de aterrizaje

Chasis F450	
Material	Plástico reforzado con fibra de vidrio.
Dimensiones	450 mm de diámetro (motor a motor).
Peso	Aproximadamente **280 g** sin componentes adicionales.
Montaje	Incluye una placa de distribución de energía con conexiones **XT60** para la alimentación de los componentes.
Tren de aterrizaje	
Altura	Aproximadamente **20 cm**.
Material	Plástico ABS reforzado.
Montaje	Se fija en la parte inferior del chasis con tornillos.

HAWK'S WORK. (2025). Chasis y tren de aterrizaje. Amazon. Recuperado de https://www.amazon.com/HAWKS-WORK-Quadcopter-Soldered-version/dp/B09YQ4G4ZZ?th=1

Motores sin escobillas 2212 (4 unidades)

Los motores sin escobillas (**brushless**) son un tipo de motor eléctrico que utiliza imanes permanentes en lugar de escobillas para crear un campo magnético que gira el rotor. Esto significa que no hay contacto físico entre las escobillas y el rotor, lo que reduce el desgaste y aumenta la eficiencia energética. El modelo **2212** de HAWK´S WORK proporcionan la potencia necesaria para el vuelo del dron. Sus características son:



Moto	KV (PRM/V)	Voltage (V)	Prop (Inch)	Currenc (A)	Thrust (G)	Power (W)	Efficiency (G/W)	Speed (RPM)
A2212	920	11.1 (3S)	9450	1	110	11.1	9.91	3160
				3	270	33.3	8.11	4830
				5	410	55.5	7.39	5810
				7	550	77.7	7.08	6460
		14.8 (4S)	9450	4	370	59.20	6.25	5690
				6	550	88.80	6.19	6580
				8	670	118.40	5.66	7260
				10	810	148.00	5.47	7860

Ilustración 3: Motores Brushless 2212

Tabla 2: Características Motores Brushless

Voltaje nominal	7.4V - 14.8V (2-4SLiPo).
KV (velocidad por voltio)	920 KV.
Corriente máxima	Aproximadamente **12A**.
Peso neto del motor	52g
Empuje máximo	**~850 g** por motor con hélices de 9 pulgadas.
Eficiencia	Diseñados para ofrecer buena relación empuje-consumo.

HAWK'S WORK. (2025). *Motores sin escobillas 2212*. Amazon. Recuperado de https://www.amazon.es/HAWKS-WORK-Escobillas-Conectores-Actualización/dp/B0CDGNGLHM?th=1

Controladores de velocidad electrónica (ESC) 20A (4 unidades)

Los **ESC 20A** son un circuito electrónico encargado de conseguir que el motor gire a la velocidad deseada de forma correcta y eficiente. Se encargan de regular la potencia suministrada a los motores desde la batería incluyendo protección de arranque, protección de sobrecarga, protección de batería baja, protección de pérdida de señal del acelerador, protección de rotación bloqueada, protección de sobrecalentamiento.

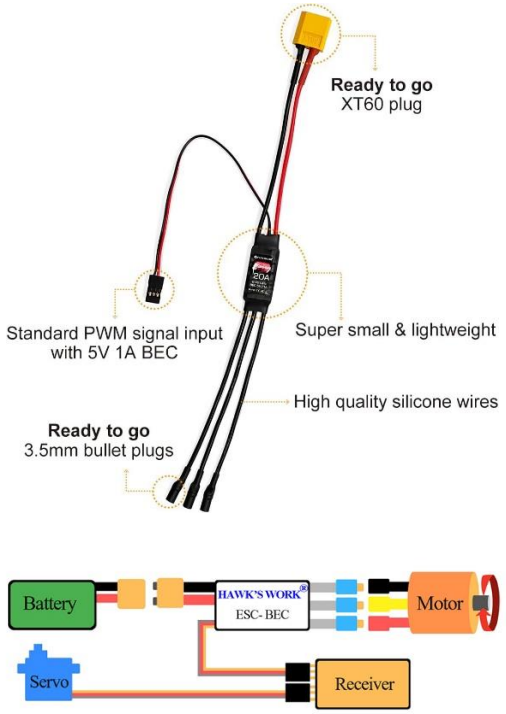


Ilustración 4: ESC

Tabla 3: Características ESC

Peso del producto	17.9g
Corriente continua	20A.
Máximos instantáneos corriente	30A 10 segundos.
Voltaje de operación	2S - 4S LiPo (7.4V - 14.8V).
BEC	5V 1A
Frecuencia de operación	8 kHz – 20 kHz.
Firmware	Compatible con **BLHeli / SimonK**.

Conexión	Entrada **PWM** para recibir señales de control desde el controlador de vuelo, conector **XT60** para la batería y conexiones **bullet** para los motores

HAWK'S WORK. (2025). *ESC 20A - Controlador de velocidad electrónica*. Amazon. Recuperado de https://www.amazon.es/ESC-escobillas-Brushless-Controlador-Multi-Rotor/dp/B0B25DLFZ2

Hélices 9450 (4 unidades)

Las hélices **9450** están diseñadas para optimizar la eficiencia aerodinámica del dron.

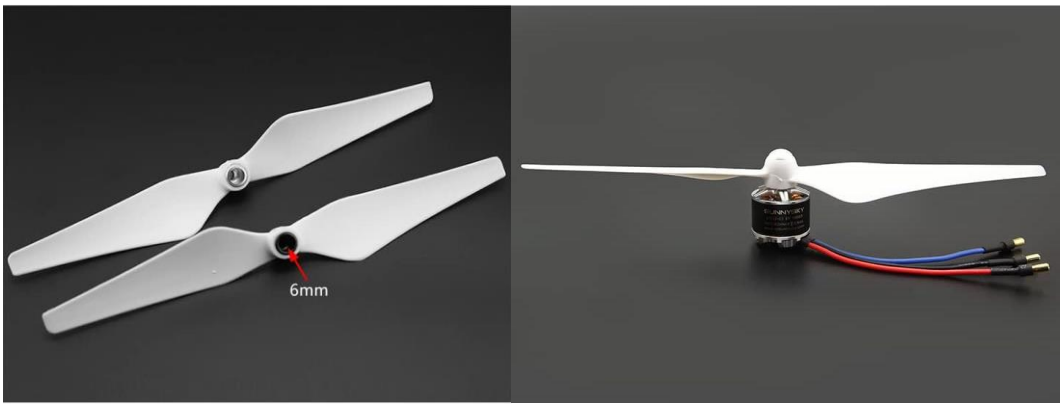


Ilustración 5: Hélices 9450

Tabla 4: Características Hélices 9450

Dimensiones	9.4 x 5.0 pulgadas.
Material	Plástico reforzado con fibra de carbono.
Montaje	Sistema de fijación rápida CW y CCW.
Beneficios	Aportan mayor estabilidad y eficiencia en el consumo energético.

HAWK'S WORK. (2025). *Hélices 9450*. Amazon. Recuperado de https://www.amazon.com/-/es/dp/B0B12W1SWM?ref=emc_s_m_5_i_atc&th=1

3.1.2 Control y Sensores

El sistema de control y sensores permite la estabilidad y navegación del dron.

Pixhawk 2.4.8 - Controlador de vuelo

El **Pixhawk 2.4.8** es el cerebro del dron, controlador de vuelo avanzado y ampliamente utilizado en la comunidad DIY. Es compatible con software como ArduPilot, lo que te permitirá controlar el dron desde un ordenador mediante el protocolo MAVLink. Tiene sensores como giroscopios, acelerómetros y un barómetro integrado.

Ilustración 6. Pixhawk, conexiones frontales

Ilustración 7: Pixhawk conexiones laterales

Tabla 5: Características Pixhawk 2.4.8

Procesador	ARM Cortex-M4 32 bits.

IMU	Giroscopio, acelerómetro y barómetro.
Puertos de comunicación	I2C, SPI, UART, CAN.
Modos de vuelo soportados	Stabilize, AltHold, Loiter, Auto, RTL.
Software compatible	ArduPilot y PX4.
Alimentación	**5V – 5.5V** (vía módulo de potencia).
Dimensiones	81,5 mm x 50 mm x 15,5 mm.
Peso	38 g.

HAWK'S WORK. (2025). *Controlador de vuelo Pixhawk 2.4.8*. Amazon. Recuperado de https://www.amazon.es/HAWKS-WORK-Pixhawk-Control-Adhesives/dp/B09XXJM2DN

GPS M8N

El **GPS M8N** permite el navegación y posicionamiento global del dron precisos, fundamental si se planea utilizar funciones autónomas o realizar trayectorias predefinidas.

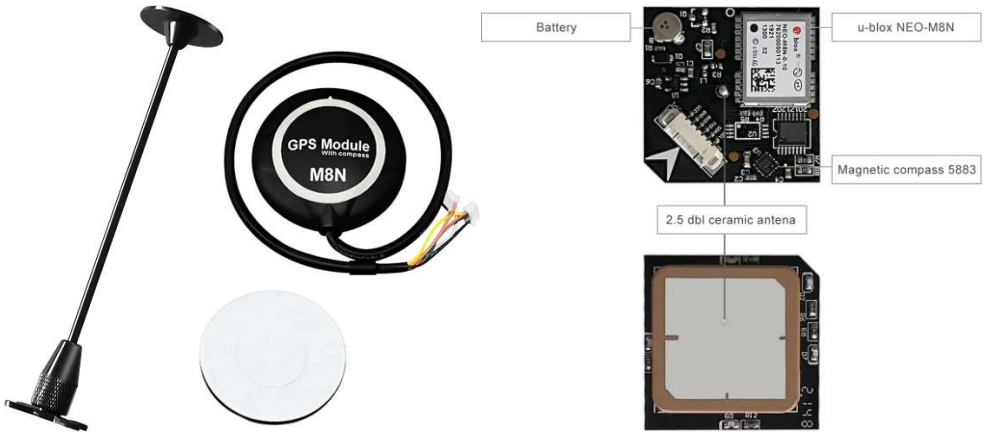


Ilustración 8: GPS M8N

Tabla 6: Características GPS M8N

Frecuencia	1 Hz - 10 Hz (configurable).
Precisión de velocidad	0.1 m/s
Precisión horizontal	~2.5 metros sin correcciones.

Voltaje de operación	voltaje CC 5 V ± 5 %
Interfaz de comunicación	UART.
Brújula incorporada	QMC5883L (puerto I2C).

HAWK'S WORK. (2025). *GPS M8N*. Amazon. Recuperado de https://www.amazon.com/-/es/M%C3%B3dulo-soporte-precisi%C3%B3n-posicionamiento-Multirotor/dp/B09YC5X8SF

Sistema RC

El transmisor FS-i6X y el receptor FS-iA6B constituyen un sistema RC informatizado proporcional digital AFHDS 2A de 6 canales de 2,4 GHz. Este sistema ofrece una protección superior contra interferencias mientras mantiene un menor consumo de energía y una alta sensibilidad fiable del receptor. Está especialmente desarrollado para todos los modelos de control de radio.

Ilustración 9: Emisora y receptor, sistema RC

Emisora FS-i6X

Una emisora versátil de 6 a 10 canales que permite controlar el dron de forma manual. Puede asignar funciones específicas a distintos interruptores o mandos, como modos de vuelo.

Ilustración 10: Emisora FS-i6X

Tabla 7: Características Emisora FS_i6X

Canales	6-10 (por defecto 6)
Tipo de modelo	Ala fija/Planeador/Helicóptero
Rango RF	2.408-2.475 GHz.
Potencia RF	<20dBm
Canal RF	135
Ancho de banda	500 KHz.
Sistema de 2,4 GHz	AFHDS 2A / AFHDS.
Tipo de modulación	GFSK
Resolución de la barra	4096
Advertencia de baja tensión	<4.2V
Puerto DSC	Puerto PS/2
Cargable	no.
Longitud de la antena	26 mm (antena dual).
Peso	aproximadamente 400 g.
Alimentación	Funciona con **4 pilas AA** (no incluidas).
Pantalla	STN pantalla transflectiva, LCD 128* 64 celosía, VA 73* 39mm, LCD con retroiluminación blanca
Tamaño	190 x 174 x 89 mm.
Actualización en línea	Sí
Color	negro.
Certificado	CE0678, FCC ID

Alcance	500m – 1km en condiciones óptimas.

FlySky. (2025). *Emisora FS-i6X 2.4GHz*. Amazon. Recuperado de https://www.amazon.es/Flysky-2-4-GHz-Transmisor-fs-ia6b-receptor/dp/B0744DPPL8?th=1

Receptor FS-iA6B

Compatible con la emisora, asegura una conexión confiable con el dron y permite recibir las órdenes de control desde la emisora.

Ilustración 11: Receptor FS-i6B

Tabla 8: Características Receptor FS-i6B

Alcance	500m-1500m
Canales	6
Tipo de modelo	Avión/Planeador/Helicóptero
Rango RF	2.4055-2.475GHz
Canal RF	140
Sensibilidad del receptor RF	-105dBm
Ancho de banda	500 KHz.
Sistema de 2,4 GHz	AFHDS 2ª
Tipo de modulación	GFSK
Potencia	4,0-6,5 V CC.

Transmisor compatible	FS-i6X, FS-i4, FS-i6, FS-i10, FS-GT2E, FS-GT2G.
Longitud de la antena	26 mm (antena dual).
Peso	16,3 g.
Tamaño	47 x 26,2 x 15 mm.
Puerto i-BUS	Sí
Puerto de adquisición de datos	Sí

HAWK'S WORK. (2025). *Receptor FS-iA6B*. Amazon. Recuperado de https://www.amazon.es/HAWKS-WORK-FS-iA6B-Receptor-transmisor/dp/B0B1MJ8CYQ?th=1

Zumbador

El zumbador es un sistema de alerta acústica que notifica errores en la conexión o batería baja. Estos errores son transmitidos por el controlador. El enlace de compra es el del kit completo.

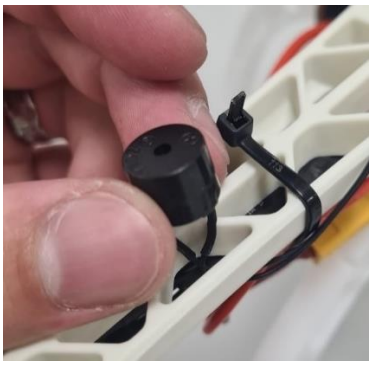


Ilustración 12: Buzzer

Tabla 9: Características Buzzer

Voltaje	**5V.**
Tono	**85 - 95 dB.**
Montaje	**Se fija al chasis o a uno de los brazos.**

3.1.3 Alimentación

El sistema de alimentación está compuesto por:

Batería LiPo 4200 mAh (35C)

Una batería de capacidad suficiente para vuelos de entre 10-15 minutos (dependiendo del peso total y las condiciones de vuelo). Su conector XT60 es estándar para integración con la placa de distribución del dron.

Ilustración 13: Batería LiPo

Tabla 10: Características Batería LiPo

Material	polímero de litio
Voltaje	11.1V
Capacidad	4200mAh
Tasa de descarga	25C
Conector	XT60

Dimensiones	25*43*138 mm
Peso	330 g
Tasa de carga	5C
Máxima Corriente de carga	21A

HAWK'S WORK. (2025). *Batería LiPo 4200mAh 35C*. Amazon. Recuperado de https://www.amazon.es/HawkS-Battery-4200mAh-Rechargeable-Multirotor/dp/B09ZYP2XQQ

Módulo de potencia PM02 V3.2

Este módulo de alimentación PM02 proporciona una forma sencilla de suministrar 5,2 V regulados al controlador de vuelo desde la batería, al tiempo que proporciona mediciones de consumo de corriente y voltaje de la batería a través de señal analógica de un cable de 6 pines para el control de vuelo. Viene con cables soldados previamente, condensador y conector XT60 en ambos extremos, junto con tubería retráctil en el PCBA para protección física.

Ilustración 14: Power Module PM02 V3.2

Tabla 11: Especif. Módulo de potencia

Voltaje de entrada	2S-12S
PCB Cont. Clasificación actual	60A
Clasificación de corriente de explosión de PCB	100A (<60 segundos)
Detección de corriente máxima	120A
Salida	DC 5,2 V y 3A Max

Divisor de voltaje	18.182
Amperios por voltios	36.364
Peso	20g

(2025). *Módulo de Potencia para Pixhawk*. AliExpress. Recuperado de https://es.aliexpress.com/item/1005006272457374.html

Adaptador XT60-JST

Este adaptador consiste en un cable que por un extremo tiene un conector **XT60** macho o hembra (dependiendo de la versión adquirida), mientras que en el otro extremo presenta un conector **JST** estándar (generalmente JST-XH o JST-BEC).

Ilustración 15: Adaptador XT60-JST

Su función principal es permitir la obtención de alimentación directa desde la batería LiPo (que emplea conector XT60) a través de una toma JST más pequeña, sin necesidad de soldar o modificar el cableado original. En el caso del dron, resulta útil para suministrar tensión al transmisor de vídeo (VTx) u otros dispositivos que requieran alimentación independiente o no proveída por el controlador de vuelo, manteniendo la polaridad y el voltaje adecuados.

Tabla 12: Especif. Adaptador XT60-JST

Material de la chaqueta	Silicona
Color	Como se muestra
Conectadores	XT60 hembra a XT60 macho y amp; JST Hombre
XT60 AWG	14
JST AWG	22

Cantidad	2Pcs / Set
Peso	32g
Tipo	Accesorio de RC

HAWK'S WORK. (2025). *Adaptador XT60-JST*. Amazon. Recuperado de https://www.amazon.es/Dilwe-Hembra-Conector-Adaptador-Accesorio/dp/B07DHCYDFL/

3.1.4 Sistema de Transmisión de Vídeo

El sistema de transmisión de vídeo permite la captura de vídeo en primera persona del dron y transmitirla vía radio al ordenador para su posterior tratamiento

Cámara FPV RunCam Nano 2

La **RunCam Nano 2** es una cámara FPV analógica de alta calidad, diseñada específicamente para drones de tamaño reducido, como micro drones y Tiny Whoops. Su tamaño compacto y su rendimiento excepcional la convierten en una opción popular entre los entusiastas de las carreras de drones y pilotos FPV que buscan una cámara ligera sin comprometer la calidad de imagen.

Esta cámara puede montarse fácilmente en el chasis F450 del dron, pero necesita estar firmemente sujeta para evitar vibraciones. Emite señales de vídeo analógicas por lo que se debe conectar al transmisor de vídeo compatible que opere en la banda de 5.8 GHz, comúnmente utilizada para FPV.

Ilustración 16: Runcam Nano 2

Tabla 13: Características Runcam Nano 2

Sensor de imagen	Equipada con un sensor CMOS de 1/3", la RunCam Nano 2 ofrece una resolución horizontal de 700TVL, proporcionando imágenes nítidas y detalladas durante el vuelo.
Lente 2.1 mm (M8)	Ofrece un campo de visión (FOV) de 155°, proporcionando una visión amplia del entorno.
Lente 1.8 mm (M8)	Proporciona un FOV de 170°, ideal para una visión aún más amplia, útil en entornos de vuelo cerrados.
Sistema de señal	Compatible con los sistemas PAL y NTSC (no conmutables), asegurando versatilidad en diversas configuraciones de vuelo.
Rango dinámico amplio (WDR)	La función D-WDR automática permite manejar condiciones de iluminación variables, garantizando una calidad de imagen consistente en entornos con alto contraste de luz y sombra.
Iluminación mínima	Con una sensibilidad de 0.01Lux@1.2F, la cámara es capaz de operar en condiciones de baja iluminación, manteniendo una imagen clara y funcional.
Consumo de energía	Funciona con un voltaje de entrada de DC 3-5.5V y consume aproximadamente 110mA a 5V, lo que la hace eficiente y adecuada para drones con limitaciones de energía.
Dimensiones y peso	Con medidas de 14 mm x 14 mm x 16 mm y un peso neto de solo 3.2 gramos, la RunCam Nano 2 es extremadamente compacta, facilitando su instalación en drones de pequeño tamaño.

RunCam. (2025). *RunCam Nano 2 - Cámara FPV 700TVL*. Amazon. Recuperado de https://www.amazon.es/RunCam-Nano2-700TVL-C%C3%A1mara-Negro/dp/B07ZNRWTBK

TBS Unify Pro 5.8GHz

El **TBS Unify Pro 5G8** es un transmisor de vídeo (VTX) de alta calidad diseñado para aplicaciones FPV (First Person View) en drones. Este dispositivo permite transmitir señales de vídeo en la banda de 5.8 GHz, ofreciendo una transmisión clara y estable, esencial para pilotos que requieren una conexión confiable durante el vuelo.

Ilustración 17: TBS Unify Pro 5.8GHz

Tabla 14: Características TBS Unify Pro 5.8GHz

Potencia de salida ajustable	Ofrece niveles de potencia seleccionables, típicamente entre 25 mW y 800 mW, lo que permite adaptarse a diferentes necesidades, desde vuelos de corto alcance hasta largas distancias.
Conectividad	Dispone de un conector RP-SMA para la antena, asegurando una conexión segura y eficiente.
Compatibilidad	• Es compatible con una amplia gama de controladores de vuelo y cámaras FPV, facilitando su integración en diversos sistemas.
Funciones avanzadas	Incluye características como PitMode, que permite realizar ajustes sin interferir con otros pilotos, y CleanSwitch, que garantiza cambios de canal sin generar interferencias

Team Black Sheep. (2025). *TBS Unify Pro 5G8 HV (RP-SMA)*. Recuperado de https://www.team-blacksheep.com/products/prod:unify_pro_hv

Antena SMA 5.8GHz

La **Bingfu Antena WiFi Doble Banda 2.4GHz 5GHz 5.8GHz 8dBi** es una antena omnidireccional diseñada para mejorar la recepción y transmisión de señales en dispositivos que operan en las bandas de frecuencia mencionadas.

Ilustración 18: Antena SMA 5.8Ghz

Tabla 15: Características Antena SMA 5.8Ghz

Frecuencia	Compatible con bandas de 2.4 GHz (2400 - 2485 MHz) y 5 GHz / 5.8 GHz (5150 - 5850 MHz), lo que la hace versátil para diversas aplicaciones inalámbricas.
Ganancia	Ofrece una ganancia de 8dBi, mejorando la intensidad y el alcance de la señal en comparación con antenas estándar.
Direccionalidad	Es una antena omnidireccional, lo que significa que emite y recibe señales en todas las direcciones horizontales, proporcionando una cobertura amplia.
Conector	Dispone de un conector macho SMA, asegurando compatibilidad con dispositivos que utilizan este tipo de conector.

Receptor de vídeo Skydroid Receptor

El **Skydroid Receptor de Control UVC de Antena Dual 5.8G 150CH** es un receptor FPV (First Person View) diseñado para recibir señales de vídeo en tiempo real desde drones que operan en la banda de 5.8 GHz. Este receptor destaca por su diseño portátil y capacidades avanzadas, lo que lo hace ideal para aplicaciones FPV con dispositivos Android.

Ilustración 19: Skydroid-receptor

Tabla 16: Características Skydroid-receptor

Compatibilidad	Compatible con dispositivos Android que soporten **UVC (USB Vídeo Class)** y **OTG (On-The-Go)**. Permite la conexión a smartphones y tablets sin necesidad de alimentación externa adicional.
Frecuencia y canales	Opera en la banda de **5.8 GHz**, una frecuencia estándar para transmisores de vídeo FPV. Ofrece **150 canales**, lo que permite flexibilidad para seleccionar el canal con la mejor recepción, evitando interferencias con otros transmisores.
Diseño de antena dual	Incluye dos antenas que trabajan en **modo diversidad**, seleccionando automáticamente la mejor señal para asegurar una transmisión estable y de alta calidad. Mejora la recepción en entornos complejos o con obstáculos, como áreas urbanas densas.
Conexión UVC OTG	Utiliza un puerto USB para conectarse directamente al dispositivo móvil. Compatible con aplicaciones FPV en Android que soporten señales UVC, permitiendo la visualización en tiempo real y grabación de vídeo.
Latencia baja	Ofrece una transmisión en tiempo real con una latencia mínima, lo que es esencial para control preciso del dron y monitoreo en tiempo real.
Audio	Soporta transmisión de audio además del vídeo, lo que puede ser útil para aplicaciones específicas que requieran monitoreo sonoro.
Compacto y portátil	Diseñado para ser ligero y fácil de transportar, ideal para pilotos que necesiten una solución de recepción práctica.

Skydroid. (2025). *Receptor de* vídeo *5.8GHz para FPV*. AliExpress. Recuperado de https://es.aliexpress.com/item/1005007422411267.html

3.1.5 Sistema de Telemetría: Holybro SiK Telemetry radio

El kit de telemetría SiK de Holybro se compone de dos módulos (uno para el dron y otro para la estación en tierra) que establecen una comunicación bidireccional entre el controlador de vuelo (como Pixhawk) y el ordenador.

Ilustración 20: SiK Telemetry Radio, Holybro. Kit

Utiliza el firmware SiK, ampliamente compatible con el ecosistema ArduPilot/MAVLink, y se conecta fácilmente vía USB en tierra y mediante puertos TELEM en el dron. Proporciona un enlace fiable para el envío y recepción de telemetría y órdenes de vuelo, permitiendo monitorear parámetros (GPS, altitud, batería) y controlar el dron (carga de misiones, cambio de modos de vuelo) en tiempo real.

Tabla 17: Datos Holybro SiK Telemetry Radio

Características	Firmware SIK de código abierto. Conexión plug-and-play para controladores de vuelo estándar Pixhawk. La forma más sencilla de conectar el Autopiloto y la Estación en Tierra. Radio intercambiable entre aire y tierra. Puerto Micro-USB (incluye adaptador a USB Tipo-C). Conector JST-GH de 6 pines.
Especificaciones	Potencia de salida máxima: 100mW / 500mW (ajustable).

	Sensibilidad de recepción: -117 dBm. Conector: RP-SMA. Comunicación bidireccional en full-dúplex a través de una interfaz UART con TDM adaptativo. Enlace serie transparente. Compatible con protocolo MAVLink. Tecnología de espectro ensanchado por salto de frecuencia (FHSS) con ciclo de trabajo configurable. Corrección de errores capaz de corregir hasta el 25% de errores de bits. Firmware SIK de código abierto. Configuración mediante Mission Planner & APM Planner. Chip FT230X/FT231: IC USB a UART básico.
Dimensiones	Tamaño: 28 x 53 x 10.7 mm (sin antena).
Datos Eléctricos	Voltaje de alimentación: 5V DC (desde USB o JST-GH). Corriente de transmisión: 100 mA a 20 dBm. Corriente de recepción: 25 mA. Interfaz serie: UART 3.3V.
Peso	13.6 g (sin antena). 7.4 g (antena de 433 MHz). 9.5 g (antena de 915 MHz).
Identificador FCC	2AP22-HT03-V3
Contenido del paquete	2 módulos de radio con antenas. 1 cable Micro-USB a USB-A. 1 cable adaptador Micro-USB a Micro-USB OTG.

	1 adaptador Micro-USB a Tipo-C. 1 cable JST-GH-6P a JST-GH-6P (para controladores estándar Pixhawk). 1 cable JST-GH-6P a Molex DF13 6P (para Pix32, Pixhawk 2.4.6, etc.).

Holybro. (2025). *SiK Telemetry Radio V3*. Holybro. Recuperado de https://holybro.com/products/sik-telemetry-radio-v3

3.2 Software

Para la configuración, control y transmisión de datos del dron, se emplearán diversas herramientas de software especializadas en la interacción con el hardware del dron, el procesamiento de datos de telemetría y la gestión de la transmisión de vídeo.

Además, se ha usado un simulador de drones en una máquina virtual, como zona de pruebas previas.

A continuación, se describen los principales programas y APIs utilizadas.

3.2.1 Mission Planner

Introducción

Mission Planner es una aplicación gratuita, de código abierto y con el apoyo de la comunidad, desarrollada por Michael Oborne para el proyecto de piloto automático de código abierto APM, en colaboración con la comunidad de **ArduPilot**, que funciona como una estación de control terrestre para aviones, helicópteros y vehículos todoterreno.

Ilustración 21: Mission Planner

Es compatible únicamente con Windows. Se puede utilizar como una utilidad de configuración o como un complemento de control dinámico para el vehículo autónomo. Estas son algunas de las cosas que se puede hacer con Mission Planner:

- Cargar el firmware (el software) en la placa del piloto automático (es decir, la serie Pixhawk) que controla su vehículo.
- Configurar el vehículo para un rendimiento óptimo, calibrando todos los dispositivos y ajustando entradas y canales.
- Planificar, guardar y cargar misiones autónomas en su piloto automático con una simple entrada de puntos de referencia con solo apuntar y hacer clic en Google u otros mapas.
- Descargar y analizar los registros de misión creados por su piloto automático.
- Interfaz con un simulador de vuelo de PC para crear un simulador de UAV completo con hardware en el circuito.
- Con el hardware de telemetría adecuado podrá:
    - Supervisar el estado de su vehículo mientras está en funcionamiento.
    - Registrar registros de telemetría que contienen mucha más información sobre los registros del piloto automático a bordo.
    - Ver y analizar los registros de telemetría.
    - Operar su vehículo en FPV (vista en primera persona).

Después de esta introducción, procederemos a analizar alguna de las ventanas de Mission Planner, explicando sus funcionalidades y opciones disponibles. El objetivo es proporcionar una guía detallada que sirva de referencia y facilite la navegación por el software. En esta sección no profundizaremos en las configuraciones específicas, ya que estas se abordarán en el Capítulo 4. Nuestro enfoque será describir las distintas funciones y ventanas para que, al mencionarlas más adelante, quede claro a qué nos referimos y cómo se estructuran dentro del programa.

Interfaz y Pantallas principales

Descripción general

La siguiente captura de pantalla muestra la vista principal de la pantalla de visualización frontal (HUD) de la estación terrestre Mission Planner. Una vez

que se haya conectado a un vehículo, esta pantalla mostrará la telemetría enviada por ArduPilot.

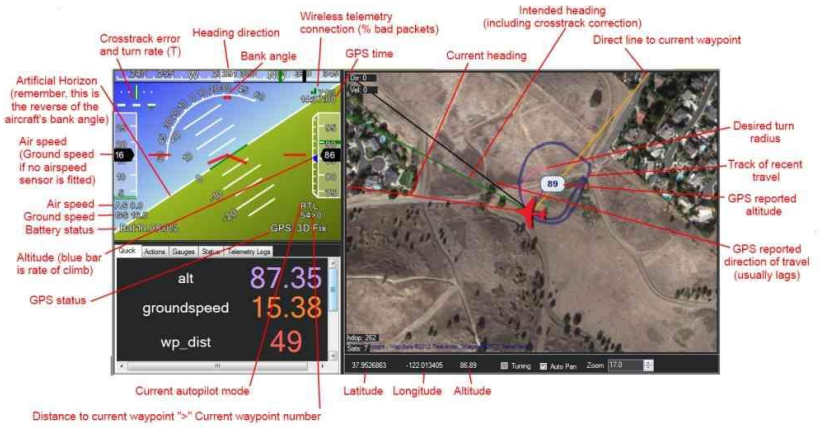


Ilustración 22: Mission Planner, pantalla de datos y HUD

A continuación, se muestra una vista más detallada del HUD (con leyenda).

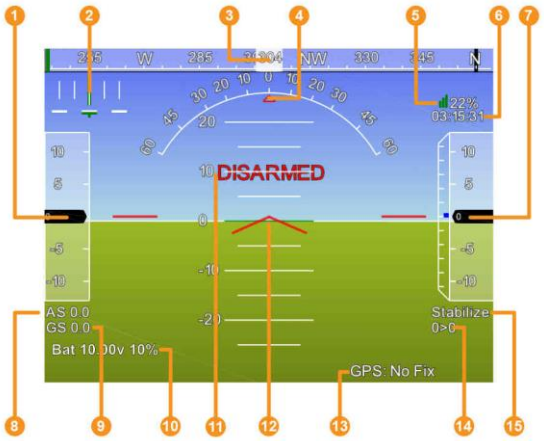


Ilustración 23: Mission Planner, HUD con leyenda

1. Velocidad aerodinámica (velocidad terrestre si no hay ningún sensor de velocidad aerodinámica instalado)
2. Error de cruce y velocidad de giro (T)
3. Dirección del rumbo
4. Ángulo de inclinación
5. Calidad del enlace de conexión de telemetría (porcentaje promedio de paquetes buenos)
6. Tiempo GPS

7. Altitud (la barra azul representa la velocidad de ascenso)
8. Velocidad aerodinámica
9. Velocidad terrestre
10. Estado de la batería
11. Horizonte artificial
12. Actitud de la aeronave
13. Estado del GPS
14. Distancia al punto de referencia > Número de punto de referencia actual
15. Modo de vuelo actual

Consejos para utilizar la pantalla de datos de vuelo

- El mapa solo mostrará la posición actual cuando tengas el GPS bloqueado o estés usando un simulador de vuelo.
- Horizontes artificiales: cuando el avión se inclina hacia la derecha, el horizonte se inclina hacia la izquierda. Solo se tiene que inclinar la cabeza y se entenderá.
- Se puede realizar cambios de modo y otros comandos de acción en el aire con el planificador de misiones y otros GCS, pero se tiene en cuenta que se debe estar bajo el control del piloto automático para que surtan efecto. Cuando el interruptor de palanca del control remoto está en la posición manual, ya no está bajo el control del piloto automático y ningún comando surtirá efecto. Debe estar en una de las otras posiciones (estabilización, vuelo por cable, automático o cualquier otro modo controlado por el piloto automático) para que los comandos MAVLink surtan efecto.
- Si se hace doble clic en el HUD, este aparecerá y permitirá ejecutar el HUD en pantalla completa en una segunda pantalla.
- Debajo del HUD hay varios botones para acciones, estado, etc.
- Si se activa la casilla de verificación Ajuste en la parte inferior del mapa y luego se hace doble clic en la ventana de ajuste que aparece, se puede graficar cualquier dato disponible en la pestaña Estado debajo del HUD. Esto significa que se puede mostrar en tiempo real la altitud, la actitud y muchas otras opciones.

- Se puede utilizar imágenes personalizadas en lugar de Google Maps. Pulsar Control-F. Esto permitirá cargar sus propias ortofotografías. Para ello, necesitará Globalmapper, ya que actualmente es uno de los pasos clave para exportar en el formato requerido para su uso en el planificador.

(Descripción general de la pantalla de datos de vuelo, 2024)

Conexión con el Autopiloto

Este apartado explica cómo conectar *Mission Planner* a un piloto automático para recibir telemetría y controlar el vehículo.

Configurar la conexión

Para establecer una conexión, primero se debe elegir el método/canal de comunicación que se desea utilizar y luego configurar el hardware físico y los controladores de dispositivos de Windows. Se puede conectar la PC y el piloto automático mediante cables USB, radios de telemetría , Bluetooth ,conexiones IP, etc.

El controlador del hardware de conexión debe estar presente en Windows ya que esto hace que el puerto COM de la conexión y la velocidad de datos predeterminada estén disponibles para *Mission Planner*.

Ilustración 24: Mission Planner, Conexión USB Pixhawk

Ilustración 25: Mission Planner, Conexión mediante SiK radio

En *Mission Planner*, la conexión y la velocidad de datos se configuran mediante los cuadros desplegables en la parte superior derecha de la pantalla.

Ilustración 26: Mission Planner, Botón de conexión

Una vez que se haya conectado el USB o la radio de telemetría, Windows asignará automáticamente al piloto automático un número de puerto COM, que aparecerá en el menú desplegable (el número real no importa). También se establece la velocidad de datos adecuada para la conexión (normalmente, la velocidad de datos de la conexión USB es 115200 y la velocidad de conexión de la radio es 57600).

Seleccionar el puerto y la velocidad de datos deseados y luego presionar el botón **CONECTAR** para conectarse al piloto automático. Después de conectarse, **Mission Planner** descargará los parámetros del piloto automático y el botón cambiará a **DESCONECTAR** como se muestra:

Ilustración 27: Mission Planner, Botón de desconexión

Consejo

El menú desplegable “seleccionar puerto” también contiene opciones de puerto TCP o UDP que se pueden usar para conectarse a un piloto automático a través de una red.

Si se hace clic en el enlace directo “Estadísticas...” que se encuentra debajo del cuadro de selección de puerto, se brindará información sobre la conexión, como si la seguridad de firma está activa, las estadísticas del enlace, etc. A veces, esta ventana aparece debajo de la pantalla actual y deberá llevarse al frente para poder verla.

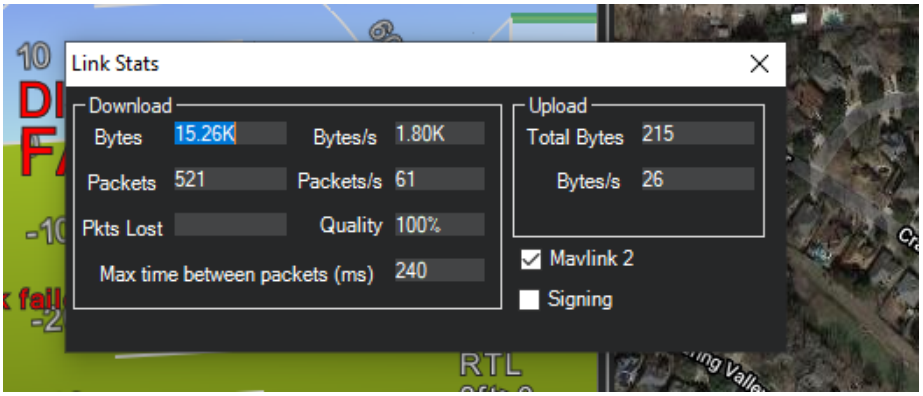


Ilustración 28: Mission Planner, Estadísticas Controlador

(Conecte Mission Planner al piloto automático, 2024)

Cargando Firmware

Estas instrucciones muestran cómo descargar el firmware más reciente en el hardware del piloto automático que ya tiene instalado el firmware de ArduPilot. Para poder cargar el firmware es necesaria la conexión directa del autopiloto al PC mediante USB como se ha visto antes. Windows debería detectar e instalar automáticamente el software del controlador correcto. Conectar desde misión planner y ya estará preparado para comenzar a cargar el firmware.

En la pantalla CONFIGURACIÓN | Instalar firmware de Mission Planner, seleccionar el ícono apropiado que coincida con el vehículo o tipo de chasis (es decir, Quad, Hexa). Responder **Sí** cuando se pregunte "¿Está seguro?".

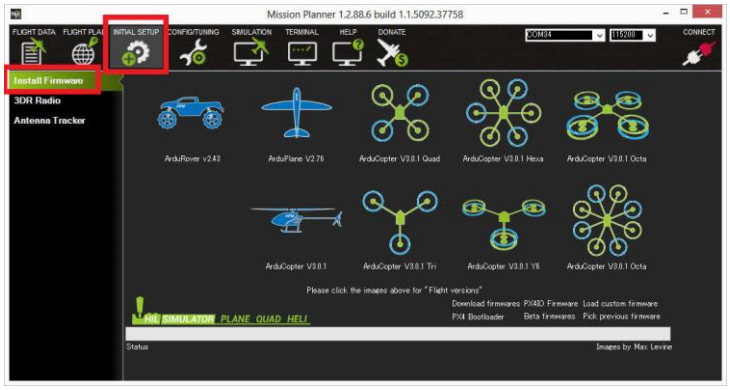


Ilustración 29: Mission Planner, Instalar Firmware (1)

Mission Planner intentará detectar qué placa se está usando. Es posible que se pida desconectar la placa, presionar OK y se vuelve a conectar para detectar el tipo de placa.

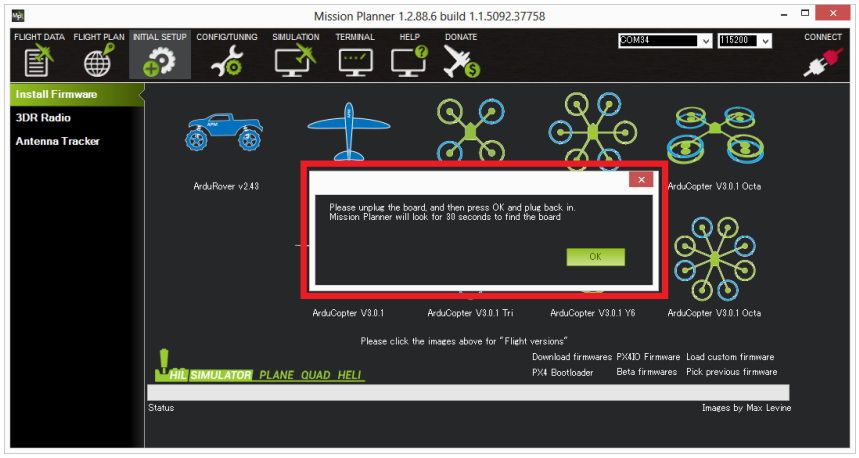


Ilustración 30: Mission Planner, Instalar Firmware (2)

A menudo, se presentará un cuadro desplegable con variantes de firmware para la placa, entre las que puede seleccionar (como variantes DShot bidireccionales, si están disponibles). Para las placas que comparten el ID de placa Pixhawk, la lista será extensa, como se muestra a continuación:

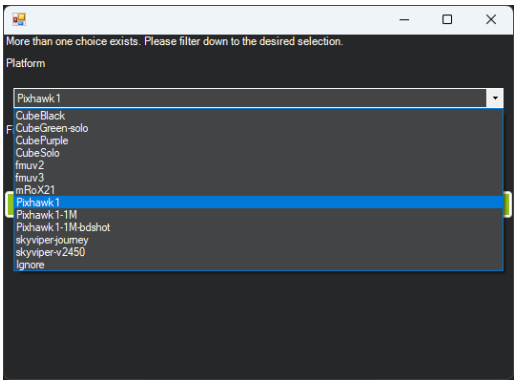


Ilustración 31: Mission Planner, Instalar Firmware (4)

Seleccione el firmware adecuado para la placa. Para las placas marcadas como “Pixhawk”, el firmware Pixhawk1 suele ser la mejor opción.

Si todo va bien, aparecerá un mensaje de estado en la parte inferior derecha que incluye las palabras: “borrar...”, “programar...”, “verificar...” y “Carga realizada”. El firmware se ha cargado correctamente en la placa.

Por lo general, el gestor de arranque tarda unos segundos en salir e ingresar el código principal después de la programación o el encendido. Esperar a presionar CONECTAR hasta que esto ocurra.

(Cargando Firmware, 2024)

Planificador de misión PLAN de vuelo

Esta sección cubre lo que se ve y lo que se puede hacer en la pantalla PLAN DE VUELO del Planificador de misiones, seleccionada en el menú en la parte superior de la aplicación Planificador de misiones. Esta sección permite planificar y ejecutar planes de vuelo, denominados misiones.

La pantalla PLAN de vuelo

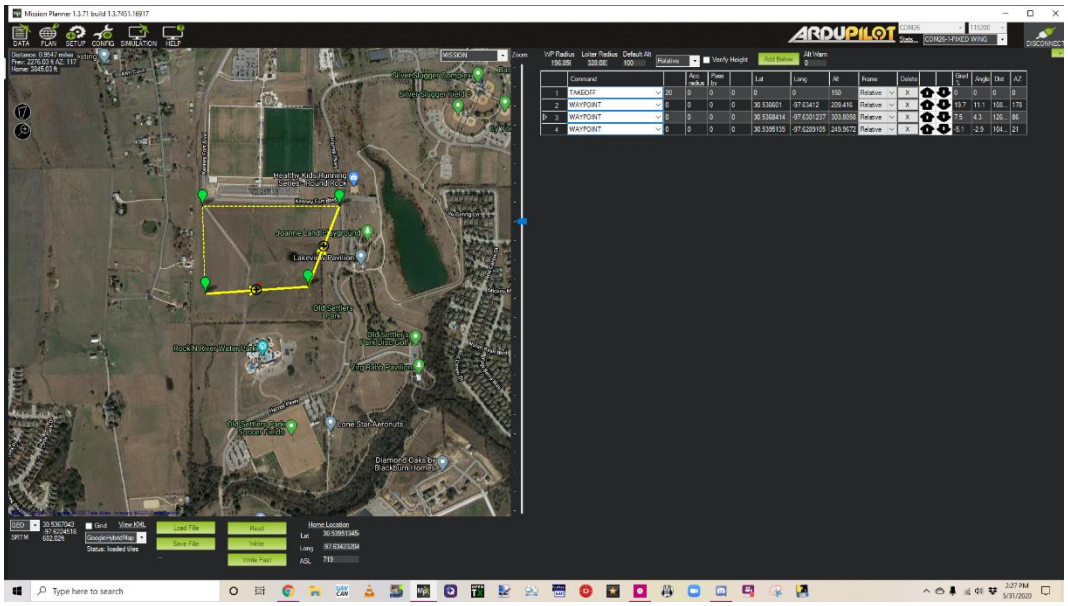


Ilustración 32: Mission Planner, Pantalla Plan de vuelo

- **Esquina superior izquierda**: muestra la distancia desde INICIO, el rumbo y la distancia desde el último punto de referencia ingresado (INICIO si no existe) y la distancia total del viaje de la misión.

- **Lado derecho**: muestra la lista de puntos de referencia para la misión. Encima de esto, se puede establecer el radio de navegación, el radio de vuelo y la altitud predeterminada para los puntos de referencia. Las altitudes se pueden ingresar en relativa (por encima de la altitud inicial), absoluta (ASL) y por encima del terreno. Las unidades se configuran en la página CONFIGURACIÓN/Planificador.

- **Parte inferior**: En la parte inferior de la vista del planificador se verá la latitud y longitud del cursor del mouse, el ASL (de SRTM, es decir, datos del terreno). Se puede seleccionar diferentes proveedores de mapas, cargar o guardar archivos de puntos de referencia, leer o escribir la misión en el piloto automático. Además, se muestran los detalles de la posición de INICIO en el mapa.

Cualquier clic izquierdo del mouse agrega un punto de referencia a la misión.

Herramientas avanzadas de creación de misiones

Mission Planner tiene la capacidad de generar automáticamente numerosas misiones de tipo topográfico y cuadrícula. La mayoría se basan en un polígono dibujado en el mapa.

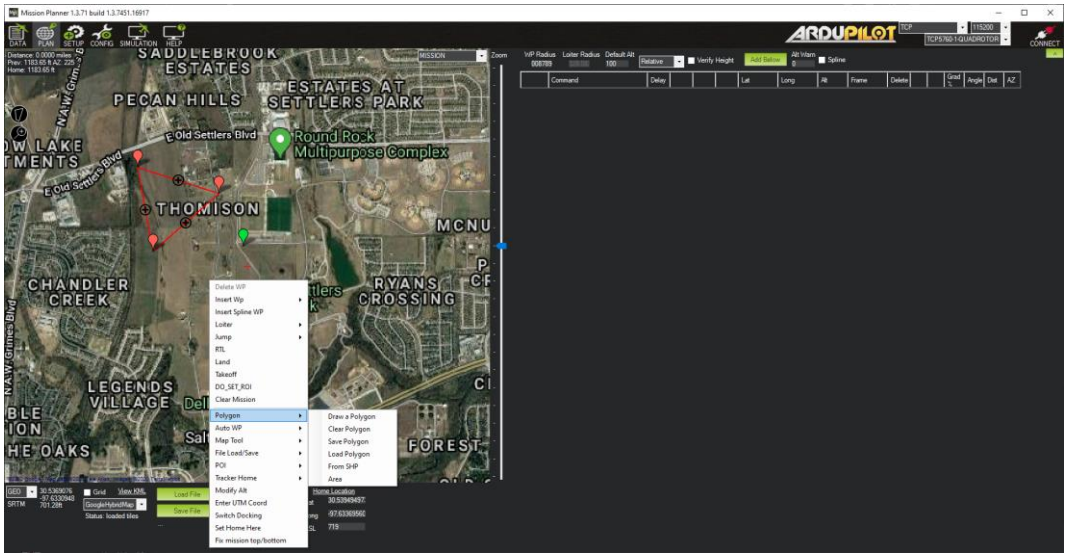


Ilustración 33: Mission planner, creación de misiones avanzada

Una vez dibujados, se pueden generar muchos levantamientos topográficos, cercas o cuadrículas de puntos de referencia diferentes.

(Planificador de misión PLAN de vuelo, 2024)

Configuración Inicial

Esta sección del Planificador de misiones, a la que se accede mediante el elemento de menú `SETUP` en la parte superior del Planificador de misiones, tiene varias subsecciones. Las subsecciones son donde se configura el piloto automático para prepararlo para el vehículo en particular. Por lo general, estas secciones son acciones "imprescindibles" que se requieren antes del primer vuelo.

Lo que se verá al ingresar a esta sección depende de si se está conectado o no. Cada elemento del menú mostrará una nueva pantalla; a continuación, se detalla cada uno de ellos con enlaces a más detalles.

La primera parte sería de conexión e instalación del firmware, pero ya se ha visto en el apartado Conexión con el Autopiloto, así que no se incluirá aquí, se parte desde ese punto.

Hardware obligatorio

Solo se verá este elemento del menú si el piloto automático está conectado. Hacer clic en este elemento del menú para ver los elementos que debe

configurar antes de intentar operar su vehículo. Los detalles se encuentran en los documentos de ArduPilot.org que cubren el vehículo específico (helicóptero, avión, vehículo explorador, dirigible).

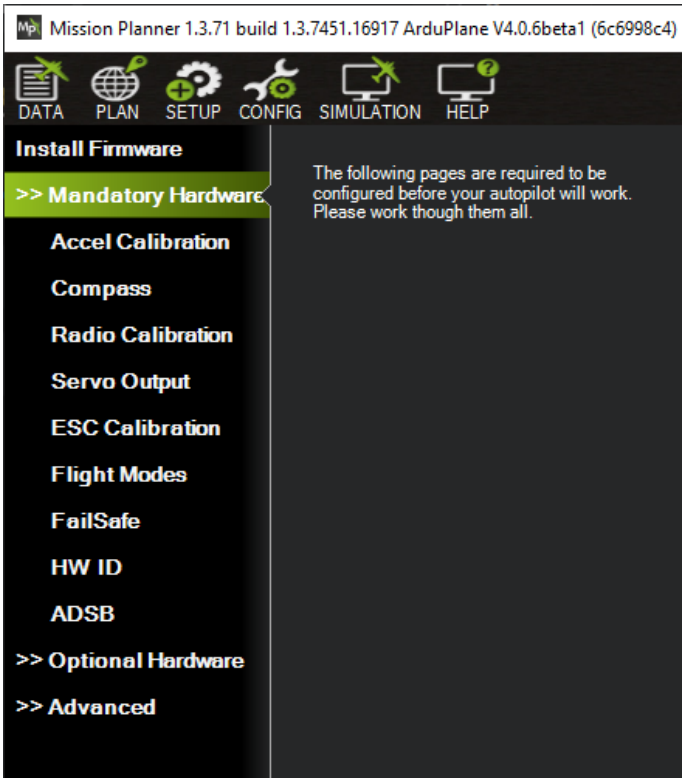


Ilustración 34: Mission planner, Mandatory Hardware

Antes de operar el vehículo, se debe configurar:

- Calibración de aceleración
- Brújula (opcional para avión)
- Calibración de radio
- **Salida del servo:** configura la función de cada salida. Los valores predeterminados se cargan durante la instalación inicial del firmware, pero hay que asegurarse de verificarlos aquí.
- **Calibración ESC solo para helicóptero** (no es necesario para los ESC que ejecutan el protocolo DShot, pero debe configurarse en ArduPilot). El avión y el rover utilizan su propiatécnica de calibración ESC, pero también es un elemento de configuración obligatorio.
- **Modos de vuelo:** El helicóptero tiene 25 modos de vuelo integrados, 10 de los cuales se utilizan con regularidad. Hay modos que admiten diferentes niveles/tipos de estabilización de vuelo, un piloto automático sofisticado, un sistema de seguimiento, etc.

Los modos de vuelo se controlan a través de la radio (mediante un interruptor de transmisor), mediante comandos de misión o usando comandos desde una estación terrestre (GCS) o una computadora compañera. Vamos a explicar los que se consideran principales.

Tabla 18: Modos de vuelo

Modos	Control de altura	Control de posición	Sensor de posición	Resumen
Alt Hold	s	+		Mantiene la altitud y nivela automáticamente el balanceo y el cabeceo.
Loiter	s	s	Sí	Mantiene la altitud y la posición, utiliza GPS para los movimientos.
RTL	A	A	Sí	Regresa por encima del lugar de despegue, también puede incluir aterrizaje.
Stabilize	-	+		Nivela automáticamente el eje de balanceo y cabeceo.
Simple/Super Simple			Sí	Un complemento a los modos de vuelo para utilizar la vista del piloto en lugar de la orientación de yaw.
Guided	A	A	Sí	Navega a puntos individuales

				ordenados por el sistema de control terrestre.

Tabla 19: Leyenda símbolos, modos de vuelo

Símbolo	Definición
-	Control manual
+	Control manual con límites y nivelación automática
s	El piloto controla la velocidad de ascenso
A	Control automático

Se verá cómo se hace cada configuración en el capítulo 4.

Hardware opcional

Este submenú permite la configuración de dispositivos opcionales, muchos de los cuales se pueden configurar mientras Mission Planner no está conectado. Aquí se puede programar la radio telemétrica Sik, la configuración de UAVCAN, el sensor de flujo óptico PX4 y el rastreador de antena, así como la configuración de un joystick para utilizarlo junto con Mission Planner.

Cuando se conectan, se pueden configurar periféricos como monitores de batería, OSD integrado, sensores de velocidad del aire y telémetros. Además, este submenú tiene una función de prueba de motor que le permite probar la dirección y el orden de los motores del helicóptero y el cuadriplano.

(Planificador de misiones CONFIGURACIÓN inicial, 2024)

3.2.2 Ubuntu con Oracle VirtualBox

¿Qué es?

Ubuntu es un sistema operativo basado en Linux, ampliamente utilizado en entornos de desarrollo.

Oracle VirtualBox es un software de virtualización que permite ejecutar sistemas operativos como Ubuntu dentro de otro sistema operativo anfitrión, como Windows.

¿Cómo funciona?

VirtualBox crea una máquina virtual que emula un ordenador, sobre el que se instala Ubuntu como sistema operativo invitado. Esto permite trabajar con software específico de Linux sin necesidad de modificar el sistema operativo principal del ordenador.

¿Por qué se usa?

El entorno de desarrollo de ArduPilot y DroneKit está más optimizado y documentado para Linux, lo que facilita la instalación y ejecución de las herramientas necesarias para el control y simulación del dron. Además, el uso de VirtualBox permite mantener la estabilidad y funcionalidad del sistema principal mientras se desarrolla en un entorno controlado.

¿Cómo se usa en este proyecto?

Ubuntu en VirtualBox es el entorno donde se desarrollan y prueban los scripts de control del dron, se ejecuta el simulador SITL, y se configuran las librerías MAVLink y DroneKit, garantizando así que todo el software funcione correctamente antes de realizar pruebas con el dron físico.

3.2.3 DroneKit SITL

¿Qué es?

DroneKit SITL es un simulador de vuelo que forma parte del ecosistema de ArduPilot. SITL (Software In The Loop) permite simular el comportamiento de un dron en un entorno virtual, replicando las señales y respuestas que se obtendrían con un dron real.

¿Cómo funciona?

El simulador SITL emula el firmware de ArduPilot ejecutándose en un ordenador. Genera datos de telemetría y permite recibir comandos del mismo modo que lo haría un dron físico. Esto permite comprobar el comportamiento de las instrucciones de vuelo sin necesidad de un dron real.

¿Por qué se usa?

Permite desarrollar y probar el software de control del dron en un entorno seguro, sin riesgo de daños al dron físico. Además, posibilita realizar múltiples pruebas y depurar los programas antes de implementarlos en el dron real.

¿Cómo se usa en este proyecto?

SITL se utiliza como plataforma de simulación para comprobar que los comandos enviados desde el ordenador funcionan correctamente, que el dron responde según lo esperado y que se reciben los datos de telemetría sin errores. Esto garantiza que el software esté optimizado antes de realizar vuelos reales.

3.2.4 MAVLink

¿Qué es?

MAVLink (Micro Air Vehicle Link) es un protocolo de comunicación de código abierto diseñado para el intercambio de datos entre vehículos no tripulados (drones) y estaciones de control en tierra.

¿Cómo funciona?

MAVLink permite el envío de paquetes de datos que contienen comandos e información sobre el estado del dron. Estos paquetes son transmitidos mediante radioenlaces, USB o redes Wi-Fi, facilitando la comunicación bidireccional entre el dron y el ordenador.

¿Por qué se usa?

Es el estándar de comunicación utilizado por ArduPilot y Pixhawk, garantizando la compatibilidad con el resto de herramientas del proyecto. Además, es eficiente y ligero, lo que lo hace adecuado para la transmisión de telemetría y el envío de comandos en tiempo real.

¿Cómo se usa en este proyecto?

MAVLink es el protocolo mediante el cual el ordenador envía instrucciones de vuelo al dron y recibe en tiempo real los datos de telemetría, como posición, altitud, estado de la batería, entre otros. Es la base de comunicación sobre la que operan SITL y DroneKit.

3.2.5 DroneKit

¿Qué es?

DroneKit es una API de código abierto escrita en Python, que permite programar y automatizar el control de drones compatibles con el protocolo MAVLink.

¿Cómo funciona?

DroneKit se conecta al dron (real o simulado) mediante MAVLink, permitiendo enviar comandos para realizar acciones como despegar, aterrizar, moverse en el espacio, o ejecutar misiones predefinidas. También permite recibir en tiempo real datos de telemetría.

¿Por qué se usa?

Proporciona una interfaz sencilla y flexible en Python para programar el comportamiento del dron. Es compatible con ArduPilot y SITL, y permite desarrollar soluciones personalizadas de control autónomo y recepción de datos.

¿Cómo se usa en este proyecto?

DroneKit es la herramienta principal con la que se desarrollan los scripts que controlan el dron desde el ordenador. Con ella se envían comandos como despegar, moverse en distintas direcciones y aterrizar, además de recibir información sobre el estado del dron.

3.2.6 OpenCV

¿Qué es?

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto enfocada en el procesamiento de imágenes y visión artificial. Desarrollada inicialmente por Intel, se ha convertido en una de las herramientas más utilizadas en aplicaciones de análisis de imágenes, reconocimiento de objetos y visión por computadora.

¿Cómo funciona?

Proporciona una serie de funciones y algoritmos optimizados para el tratamiento de imágenes y vídeos en tiempo real.

¿Por qué se usa?

Es una herramienta clave en el proyecto porque permite la captura y visualización el vídeo en tiempo real desde la cámara FPV del dron y su integración con YOLOv8, un modelo de detección de objetos basado en redes neuronales convolucionales.

¿Cómo se usa en este proyecto?

En este TFG, OpenCV será utilizado para capturar el vídeo recibido desde el dron mediante el receptor de vídeo. Para así mostrar y analizar la imagen procesada en tiempo real, permitiendo generar respuestas basadas en la detección de objetos.

3.2.7 YOLOv8

¿Qué es?

YOLOv8 (You Only Look Once, versión 8) es la última versión del popular modelo de detección de objetos en tiempo real basado en redes neuronales convolucionales (CNN). Desarrollado por Ultralytics, es una evolución de versiones anteriores de YOLO, con mejoras en precisión, velocidad y facilidad de uso.

¿Cómo funciona?

YOLOv8 es un modelo de detección y segmentación de objetos que divide una imagen en una cuadrícula y predice simultáneamente múltiples bounding boxes y etiquetas de clase para los objetos detectados. Sus principales características incluyen:

- Inferencia en tiempo real: Optimizado para ejecutarse de manera eficiente en GPU y CPU.
- Mayor precisión: Utiliza técnicas avanzadas como anchor-free detection y optimización de pérdidas.
- Entrenamiento y fine-tuning: Puede ser entrenado con datasets personalizados para mejorar su rendimiento en tareas específicas.
- Compatibilidad con OpenCV y PyTorch: Facilita su integración en aplicaciones de visión artificial.

¿Por qué se usa?

YOLOv8 es la opción ideal para el proyecto porque:

- Permite la detección de objetos en tiempo real, crucial para la autonomía del dron.
- Es altamente optimizado, lo que permite su ejecución en hardware limitado sin perder precisión.
- Es compatible con OpenCV, lo que simplifica su integración con el procesamiento de vídeo.

¿Cómo se usa en este proyecto?

En el TFG, YOLOv8 será utilizado para detectar de objetos en la transmisión de vídeo del dron. En este proyecto se usará un modelo de solo detección, y se centrará en la visualización de personas, y generar respuestas automatizadas, como cambios en la trayectoria del dron.

Se usará el modelo más pequeño para una realización de las pruebas más fluida. El tamaño de estos modelos influye en la detección de tal forma que el modelo más pequeño tiene más rapidez, pero menor precisión, y el más grande es muy preciso, pero bastante más lento. En una implementación en el mundo real sería interesante el uso de un modelo medio en estas características.

Documentación de Interés del Software

Mission Planner

ArduPilot. (2022). *Conecte Mission Planner al piloto automático*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/common-connect-mission-planner-autopilot.html

ArduPilot. (2024). *Mission Planner: Estación de Control Terrestre*. ArduPilot Documentation. Recuperado de

https://ardupilot.org/planner/docs/mission-planner-ground-control-station.html

ArduPilot. (2022). *Planificación de vuelo en Mission Planner*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/mission-planner-flight-plan.html

ArduPilot. (2024). *Configuración inicial en Mission Planner*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/mission-planner-initial-setup.html

ArduPilot. (2022). *Configuración y ajustes en Mission Planner*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/mission-planner-configuration-and-tuning.html

ArduPilot. (2023). *Simulación en Mission Planner*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/mission-planner-simulation.html

ArduPilot. (2020). *Uso de scripts en Python en Mission Planner*. ArduPilot Documentation. Recuperado de https://ardupilot.org/planner/docs/using-python-scripts-in-mission-planner.html

Oborne, M. (s.f.). *Mission Planner: Ground Control Software for ArduPilot*. Mission Planner. Recuperado de https://ardupilot.org/planner/

MAVLink

Meier, L. (s.f.). *MAVLink: Micro Air Vehicle Communication Protocol*. MAVLink Developer Guide. Recuperado de https://mavlink.io/en/

ArduPilot. (2025). *MAVLink and ArduPilot*. ArduPilot Documentation. Recuperado de https://ardupilot.org/dev/docs/mavlink-basics.html

PX4. (s.f.). *MAVLink Protocol Overview*. PX4 Developer Guide. Recuperado de https://docs.px4.io/main/en/middleware/mavlink.html

QGroundControl. (s.f.). *MAVLink Protocol*. QGroundControl Documentation. Recuperado de https://docs.qgroundcontrol.com/en/mavlink/

MAVLink Community. (s.f.). *MAVLink Protocol Reference*. GitHub. Recuperado de https://github.com/mavlink/mavlink

DroneKit

ArduPilot. (s.f.). *Using DroneKit with ArduPilot*. ArduPilot Documentation. Recuperado de https://ardupilot.org/dev/docs/dronekit.html

DroneKit Developers. (2025). *Getting Started with DroneKit-Python*. GitHub. Recuperado de https://github.com/dronekit/dronekit-python

PX4. (s.f.). *Using MAVSDK and DroneKit with PX4*. PX4 Developer Guide. Recuperado de https://docs.px4.io/main/en/robotic_vehicle/mavsdk_dronekit.html

3D Robotics. (s.f.). *DroneKit API Reference*. DroneKit Documentation. Recuperado de https://dronekit.io/

OpenCV

OpenCV. (s.f.). *Open-Source Computer Vision Library* [Software]. Recuperado de https://opencv.org/

YOLOv8

Ultralytics. (s.f.). *YOLOv8: Real-Time Object Detection and Segmentation* [Software]. Recuperado de https://www.ultralytics.com/es

CAPÍTULO 4: MONTAJE Y CONFIGURACIÓN

En este capítulo se tratará el proceso completo de ensamblaje y configuración del dron, desde la instalación de sus componentes físicos hasta la calibración de sus sistemas de control y comunicación. Se detallará el montaje del chasis y de los motores, la conexión de los distintos módulos electrónicos y la integración del sistema de transmisión de vídeo.

Además, se explicarán los pasos necesarios para configurar la emisora FS-i6X, el controlador de vuelo Pixhawk y la calibración de sensores y motores mediante el software **Mission Planner**. Esta configuración es esencial para garantizar la estabilidad, precisión y seguridad en el vuelo del dron.

El objetivo de este capítulo es proporcionar una guía clara y estructurada que permita replicar el proceso sin dificultades, asegurando un montaje correcto y una configuración óptima para su posterior uso en pruebas y desarrollo de software.

4.1 Montaje del Dron

En este apartado se describe todo el montaje de los componentes descritos anteriormente.

4.1.1 Chasis y motores

Para empezar, se monta lo que viene a ser el chasis principal de dron con los motores. Al abrir el paquete del armazón del dron están los 4 brazos del dron con las dos placas para atornillar componentes y distribuir la batería a los motores, después, en el paquete del sistema de alimentación se encuentran los 4 motores, cada uno con 4 tornillos y un cable ESC, que se usarán después.

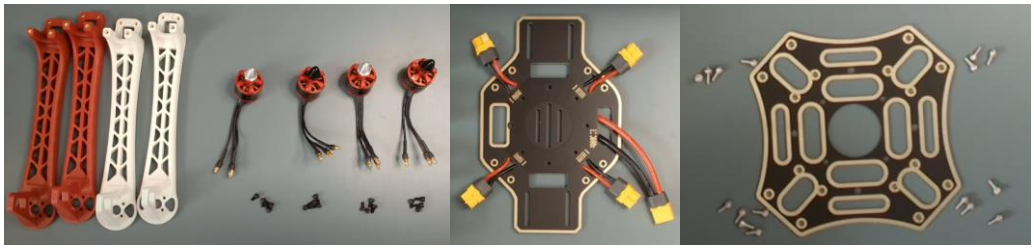


Ilustración 35: Montaje, piezas chasis y motores

Antes de atornillar los motores hay comprobar de que tipo es cada motor, para ello hay que mirar en la pegatina de cada motor si es CW o CCW, y a parte desenroscar el capuchón que llevan (negros y plateados) para poder ver la forma en la que acaba el tornillo donde van estos capuchones.

Hay dos tipos de motores, los CW cuyo tornillo va a terminar en un agujero o hendidura, y los CCW cuyo tornillo va a tener una terminación plana. Una vez comprobado cual es cual, atornillar cada motor a la parte redonda de los brazos

con los 4 tornillos que da el kit, alternando los motores en los brazos. En este caso los que tienen el capuchón negro son los CW (flat).

Ilustración 36: Montaje, motor atornillado y forma de alternarles

Cuando los motores estén ya listos se procede a montar estos brazos en las dos placas donde van a ir conectados los demás componentes.

En este caso la placa de los cables de la batería venía distinta a las de los vídeos para el montaje, en los cuales los cables de alimentación que salen hacia los brazos están metidos a través de los agujeros cuadrados (como se puede ver en la Ilustración 35 que está sacada de uno de los vídeos), es decir, van desde la cara de la soldadura hacia la otra. Lo que permite después instalar el tren de aterrizaje.

Al no ser así en esta placa (Ilustración 50 se ve como es) se ha dado la vuelta para poder instalar el tren de aterrizaje. Se verá el cambio en las últimas fotos, ya que el tren es de las ultimas cosas que se han montado.

Ilustración 37: Montaje, placa inferior (colocar al revés)

Y después simplemente atornillar la placa superior con los tornillos dados por el kit.

Ilustración 38: Montaje, placa superior

4.1.2 Controlador, dispositivos y conexiones

A continuación, se va a ir ensamblando y acoplando al chasis los diferentes dispositivos de los que dispone, haciendo las conexiones correspondientes de cada uno al controlador. La siguiente imagen es el esquema de las conexiones.

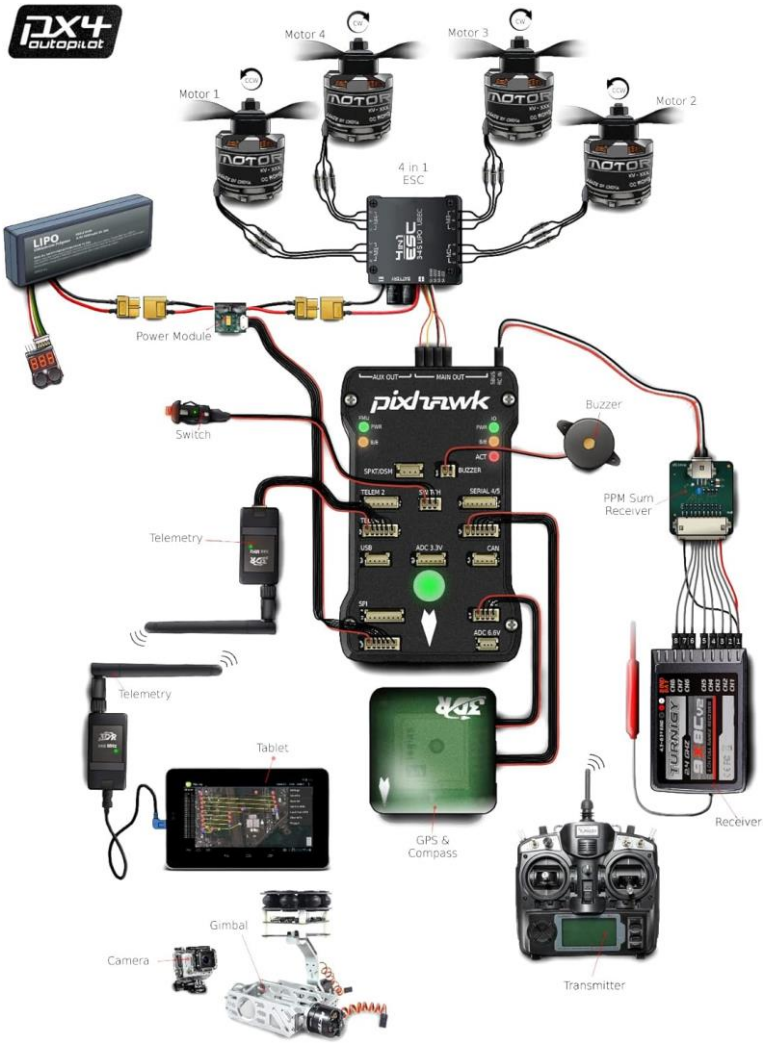


Ilustración 39: Montaje, esquema de conexiones del controlador

Controlador de vuelo Pixhawk

Primero de todo hay que incorporar el dispositivo más importante del dron, el controlador de vuelo Pixhawk 2.4.8. El paquete del Pixhawk viene con una serie de accesorios ya descritos anteriormente.

Ilustración 40: Montaje, Pixhawk y complementos

De momento usaremos solamente el soporte amortiguador de vibraciones, las tiras adhesivas y el propio controlador. Esto va situado en la placa superior del dron, en la que se colocarán 2 tiras adhesivas para situar el soporte, en este caso de forma horizontal, se coloca el soporte y se repite lo mismo para pegar el controlador al soporte.

Ilustración 41: Montaje, tiras adhesivas controlador

Con lo cual, el controlador debería de quedar acoplado en el chasis de esta forma:

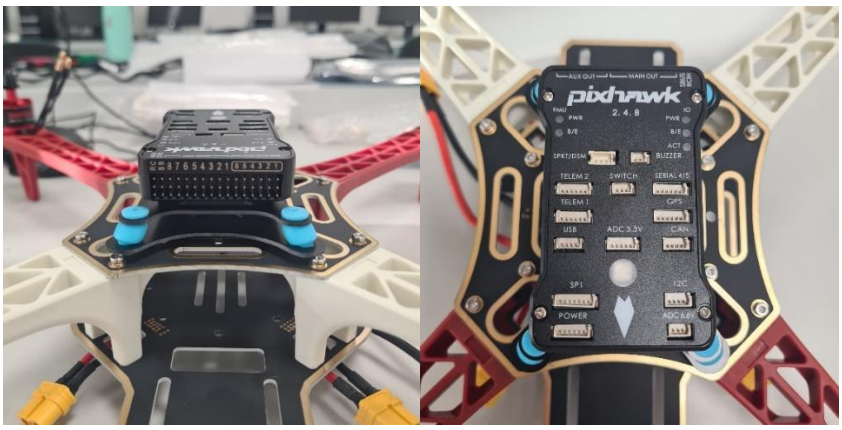


Ilustración 42: Montaje Controlador integrado en chasis

Cables ESC

Una vez con el controlador integrado se completan las conexiones de los motores mediante los cables ESC.

Ilustración 43: Montaje, cables ESC

Los motores tienen 3 pines que se conectan a uno de los extremos del cable (sin seguir específicamente ningún orden), en el otro extremo salen 2 conectores, uno de ellos es de alimentación que se conecta a los que salen de la placa (son exactamente iguales), y el otro va conectado a los pines de la parte de arriba del controlador.

Ilustración 44: Montaje, Pines motores

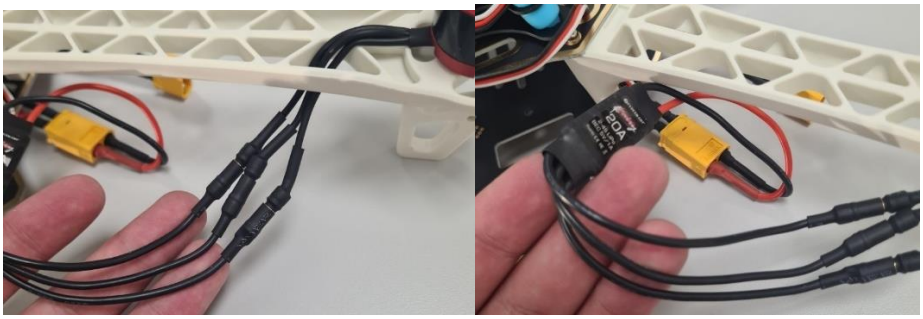


Ilustración 45: Montaje, conexión Motor-ESC-Batería

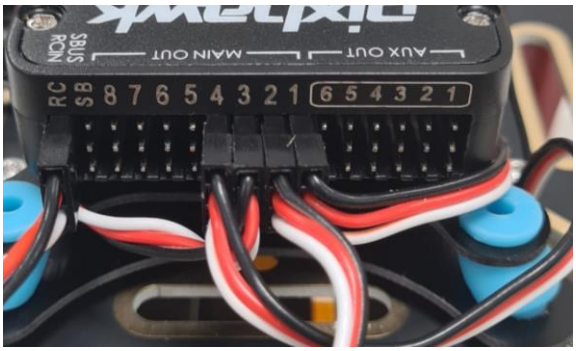


Ilustración 46: Montaje, conexiones Motor-ESC-Pixhawk

En este caso en la Ilustración 46 se han conectado al controlador según los números que venían en el manual en el punto Chasis y motores a la hora de poner cada motor en un brazo según si era CW o CCW. Los pines de arriba son tierra, así que el cable negro irá arriba. AL final de esta parte debería quedar así:

Ilustración 47: Montaje, Pixhawk-ESC-Motores completo

Al quedar los cables tan sueltos hay que ajustarlos a los brazos del dron con las bridas que venían con el kit, de tal forma que queden pegados a los brazos:

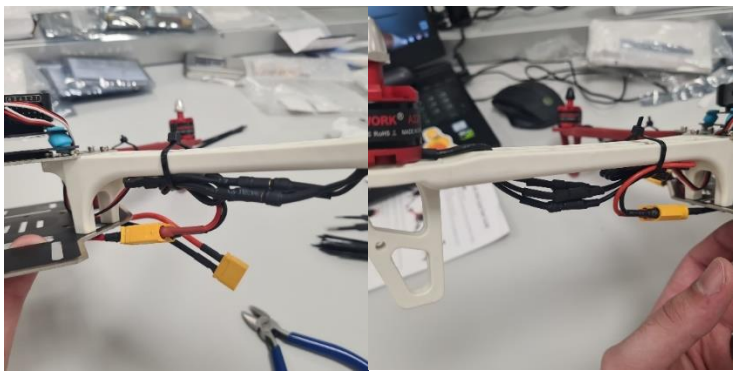


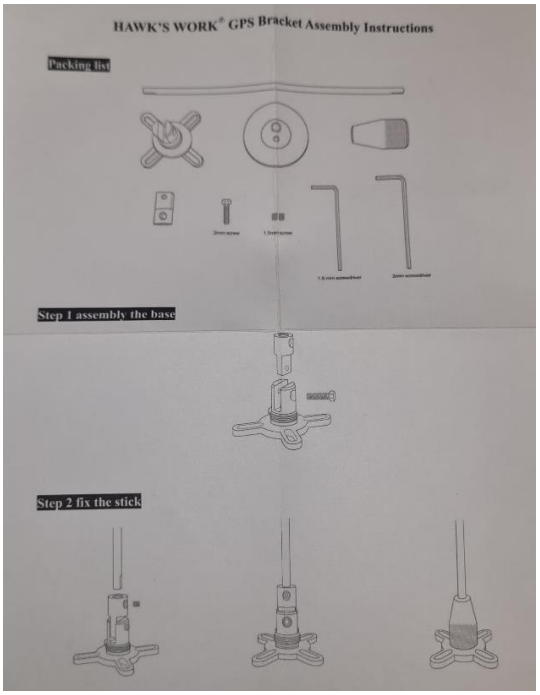
Ilustración 48: Montaje, ajustar cables

GPS

A continuación, se ensamblarán las partes del GPS, en las que vienen el soporte y el propio GPS.

Ilustración 49: Montaje, partes GPS

El GPS viene con unas instrucciones, es bastante sencillo así que las siguientes ilustraciones son las instrucciones de ensamblado.



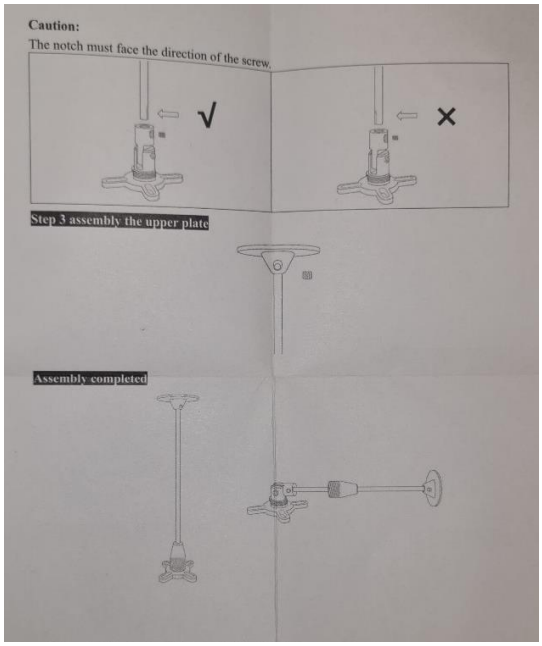


Ilustración 50: Montaje, instrucciones GPS

Después esto se pega el GPS a la parte de arriba del soporte con los adhesivos, se atornilla a la placa superior del chasis y se conecta al controlador de esta forma:

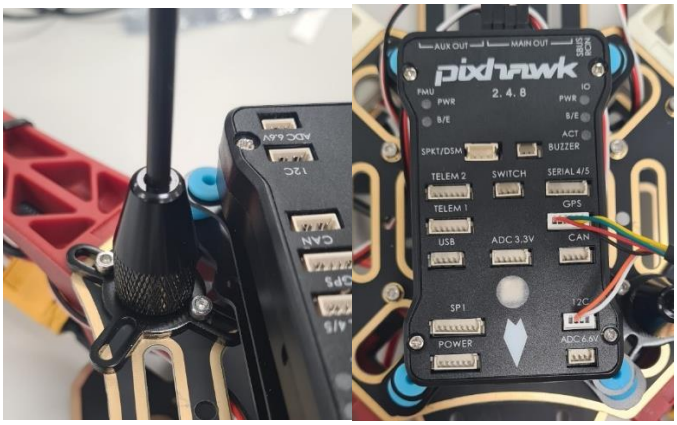


Ilustración 51: Montaje, acoplo y conexión del GPS

Ilustración 52: Montaje, GPS completo

Interruptor de seguridad

Después del GPS se instala el interruptor de seguridad, que simplemente irá enganchado en un lateral del dron y conectado al “switch” del controlador.

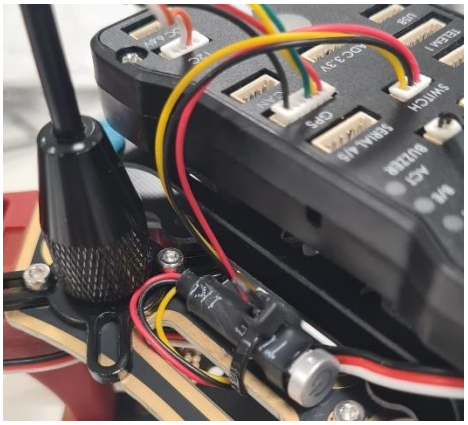


Ilustración 53: Montaje, interruptor de seguridad

Zumbador

El zumbador irá situado en uno de los brazos por falta de espacio en la placa superior, sujeto con las bridas que estaban ya puestas o un adhesivo, y va conectado al conector “buzzer” del controlador.

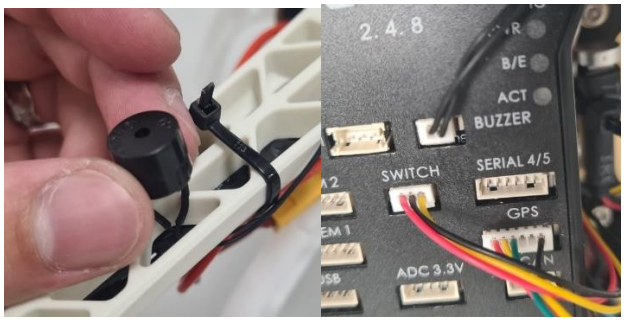


Ilustración 54: Montaje, zumbador

Tren de aterrizaje

El tren de aterrizaje se ha montado de las últimas cosas porque así lo hacían en los tutoriales que se han visto, y consiste en sustituir los tornillos de la placa inferior del chasis por unos más largos que vienen en el pack del tren de aterrizaje. Para así poder atornillar en esos agujeros las patas del dron, la placa inferior y los brazos donde se encuentran los motores.

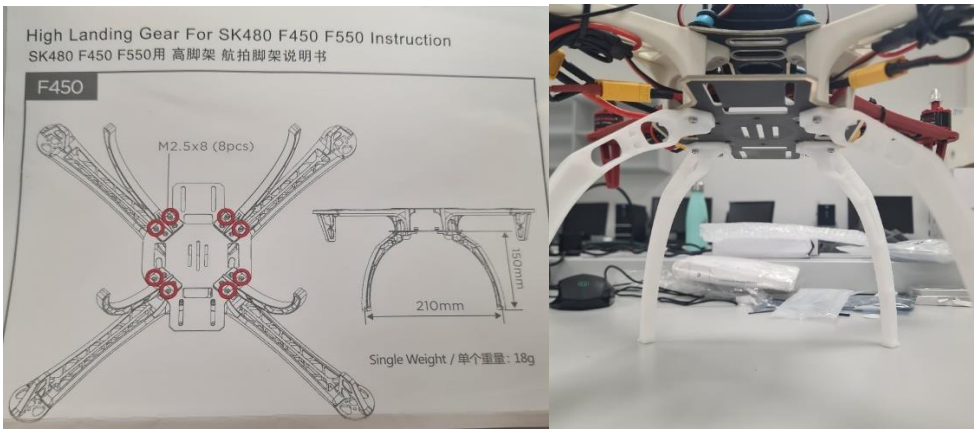


Ilustración 55: Montaje, tren de aterrizaje

Receptor FS-iA6B

El ultimo dispositivo a conectar que nos venía con el kit es el receptor de las órdenes del mando. Este se situará sobre uno de los brazos y va conectado a los pines RC del conjunto de pines de la parte de arriba del controlador.

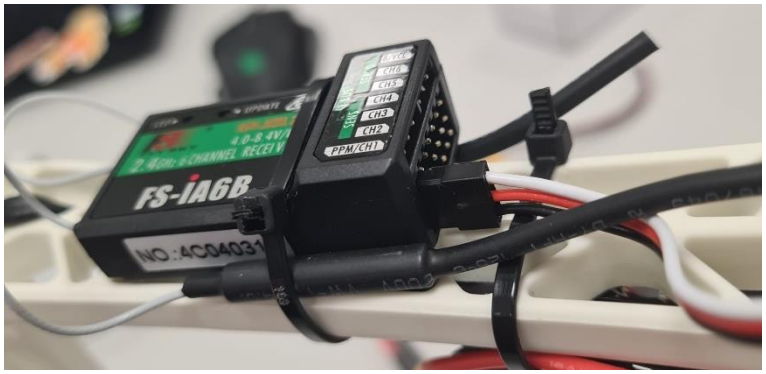


Ilustración 56: Montaje, receptor de control manual

SiK Telemetry Radio

El kit de telemetría SiK de Holybro se compone de dos módulos (uno para el dron y otro para la estación en tierra) que establecen una comunicación bidireccional entre el controlador de vuelo y el ordenador. Se conecta fácilmente vía USB en tierra y mediante puertos TELEM en el dron. En este caso y en la mayoría el recomendable, se conecta al puerto TELEM 1.

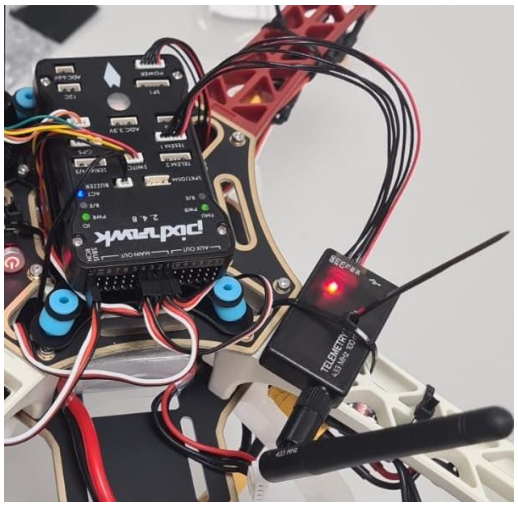


Ilustración 57: Montaje, SiK Telemetry Radio

Transmisor de vídeo + Cámara

Finalmente se va a conectar los dispositivos para realizar la funcionalidad extra que le hemos dado a e este dron, la capacidad de realizar vídeo y transmitirlo. Para ello se tiene el transmisor de vídeo TBS Unify Pro Race y la cámara Runcam Nano 2.

Estos dos dispositivos van conectados entre sí mediante 3 cables, el naranja que es una salida de 5V del transmisor para alimentar a la cámara, el negro que es la tierra y el amarillo que es el que envía el vídeo de la cámara al transmisor. Después, el transmisor va a alimentado de 7 a 23 voltios.

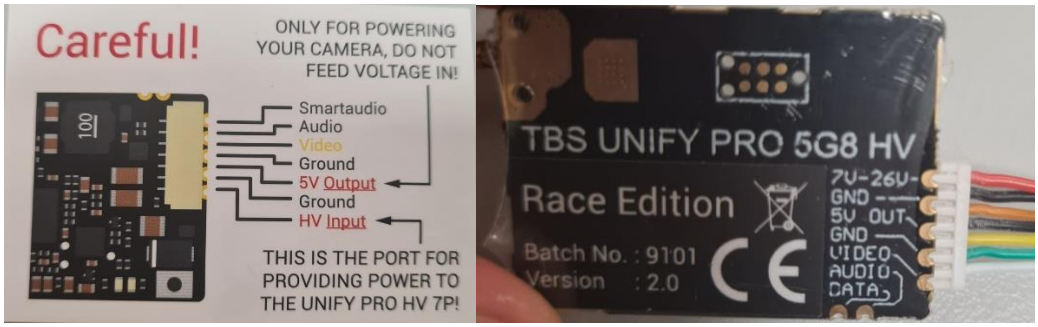


Ilustración 58: Montaje, distribución de cables TBS Unify

Al empezar a montarlo se ha visto que había un problema con los conectores. Aunque son los mismos 3 cables en los dos dispositivos, el conector del transmisor está hecho para que se conecten 4 pines aunque uno de ellos no vaya a ningun cable, en la ilustracion 59 se ve mejor (verde camara, rojo transmisor).

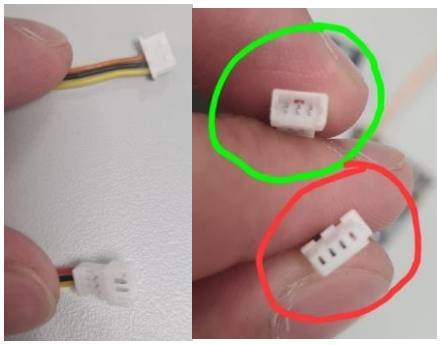


Ilustración 59:Montaje, Problema conectores TBS+Runcam

También ha aparecido ese mismo problema con la alimentación del transmisor, ya que teniendo una conexión mediante 2 cables (2 pines), no se puede conectar a ningún puerto de voltaje de salida del Pixhawk (como se ve en la Ilustración 60, lo que se sostiene es la alimentación del transmisor). Además de que todas las salidas proporcionan menos voltaje del indicado (7-23V).

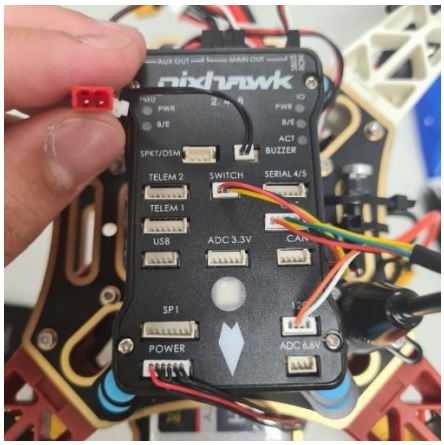


Ilustración 60: Montaje, Problema alimentación transmisor

Como solución para la cámara se ha optado por unos empalmes mediante regletas (ilustración 61), en el que se une cada cable con su gemelo.

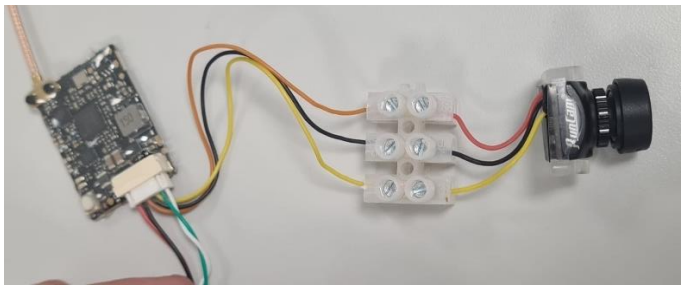


Ilustración 61: Montaje, solución conexión TBS+Runcam

Pero en el caso de la alimentación no hay posibilidad de realizar ningún empalme ni conexión, así que se ha optado por usar un cable adaptador de XT60 (cables de salida de la batería) a JST macho (conexión de pines). Este adaptador va situado entre la batería y el módulo de potencia. Por lo que, aunque se conecte el transmisor, seguirá alimentando a los motores y al Pixhawk + dispositivos.

Ilustración 62: Conexión batería-transmisor-Cámara

4.1.3 Baterías

La ubicación que se le dará a la batería va a ser en el hueco entre placas sujetas por una correa como se ve en la Ilustración 63.

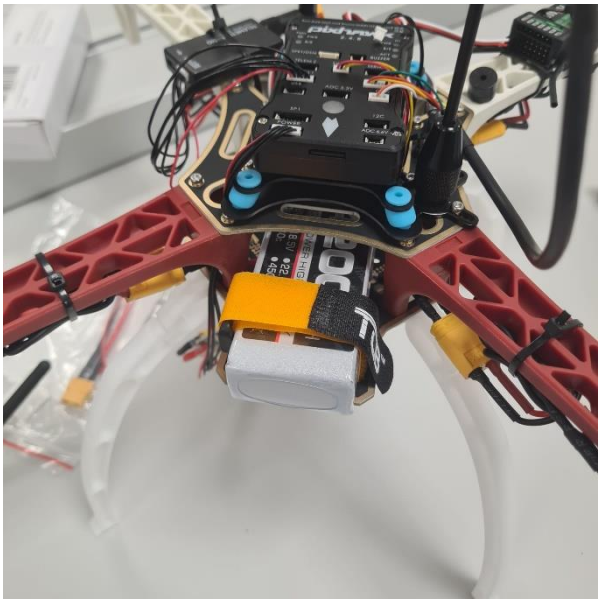


Ilustración 63: Montaje, ubicación batería

Con la batería se podrá alimentar a todos los dispositivos con los accesorios de los siguientes apartados.

Adaptador XT60-JST

Como se ha visto antes este añadido nos permite alimentar el sistema de vídeo.

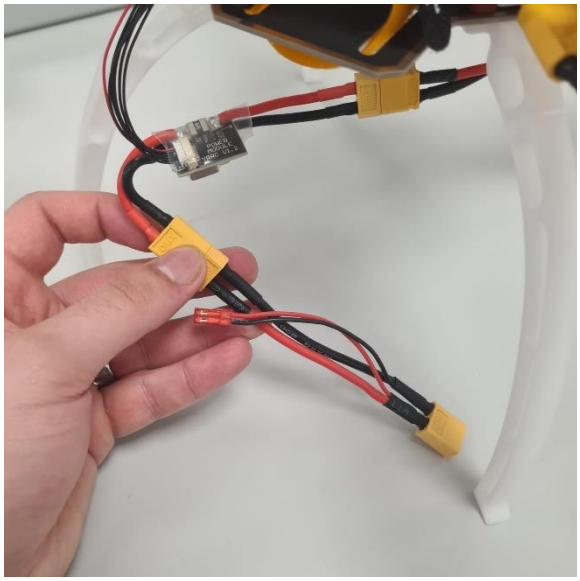


Ilustración 64: Conexión Batería a XT60-JST

Módulo de potencia

El módulo de potencia aparte de distribuir la batería al Pixhawk y a los motores, va a servir también como sensor para el monitor de la batería para Mission Planner.

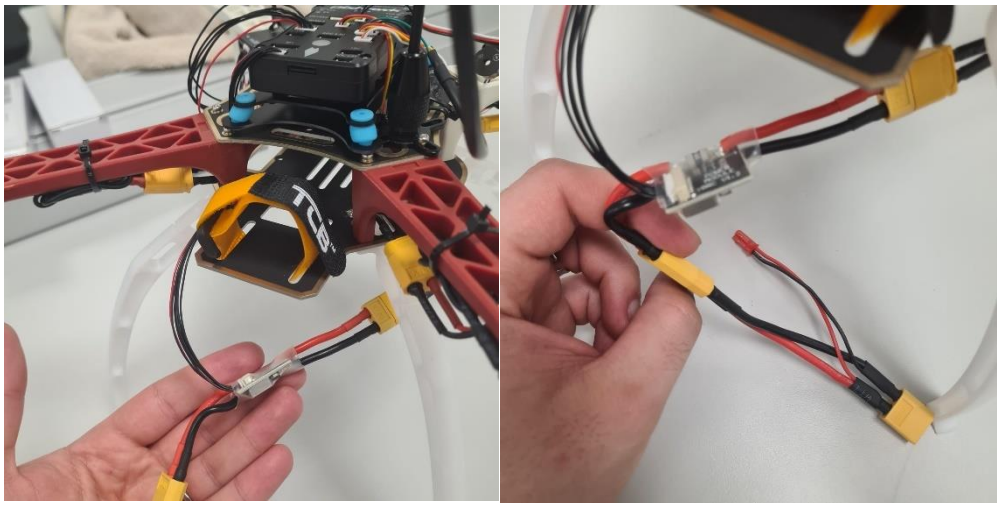


Ilustración 65: Montaje, conexión XT60-JST con módulo de potencia

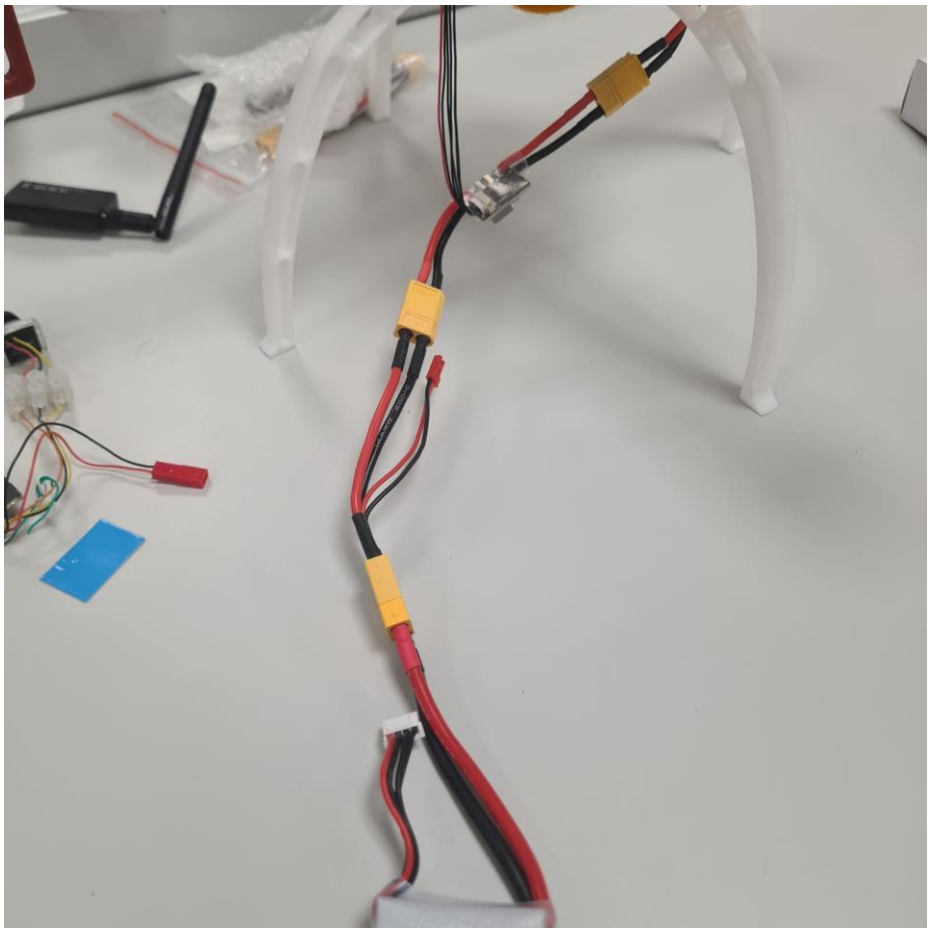


Ilustración 66: Montaje, conexión batería-controlador con módulo de potencia

4.2 Configuraciones

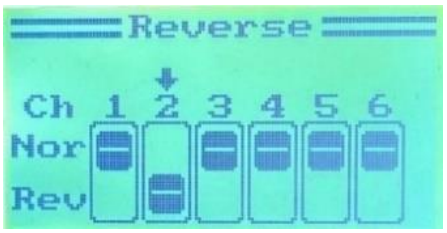
Una vez se tengan todos los componentes montados comenzarán las configuraciones del mando o emisora, y las calibraciones del controlador Pixhawk usando el Mission Planner. Las cuales vienen especificadas en el manual que nos viene con el kit del dron, y también en su página web.

4.2.1 Emisora FS-i6X

Para el control del dron, se ha utilizado la emisora FS-i6X, configurada específicamente para este proyecto siguiendo estos pasos:

1. Inversión del Canal 2

La inversión del Canal 2 permite adaptar el control de dirección del dron para un comportamiento correcto durante el vuelo.

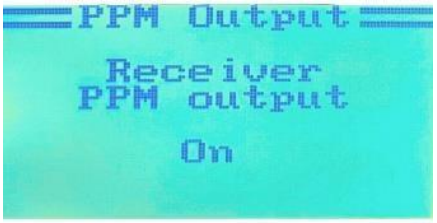


Pasos: Mantener presionado OK -> Functions Setup -> Reverse -> Seleccionar Ch 2 y presionar OK -> Mantener presionado CANCEL para guardar y salir.

Ilustración 67: Configuración FS-i6X (1)

2. Activación del modo PPM

El modo PPM (Pulse Position Modulation) permite al receptor comunicarse con el controlador de vuelo Pixhawk utilizando una única conexión de señal.

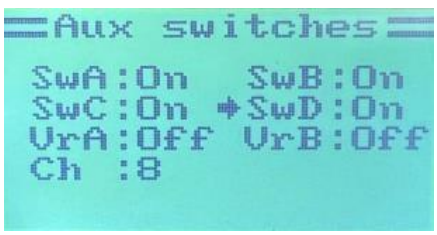


Pasos: Mantener presionado OK -> System Setup -> RX Setup -> Output Mode -> Seleccionar Output PPM -> Mantener presionado CANCEL para guardar y salir.

Ilustración 68: Configuración FS-i6X (2)

3. Activación de los interruptores auxiliares (Aux Switches)

Los interruptores auxiliares se utilizan para asignar funciones específicas al dron, como cambiar modos de vuelo o activar la cámara.



Pasos: Mantener presionado OK -> System Setup -> Aux Switches -> Activa SwA, SwB, SwC, SwD -> Asignar canal 8 -> Mantener presionado CANCEL para guardar y salir.

Ilustración 69: Configuración FS-i6X (3)

4. Configuración de los canales auxiliares

Los canales auxiliares permiten asignar funciones específicas a los interruptores auxiliares.

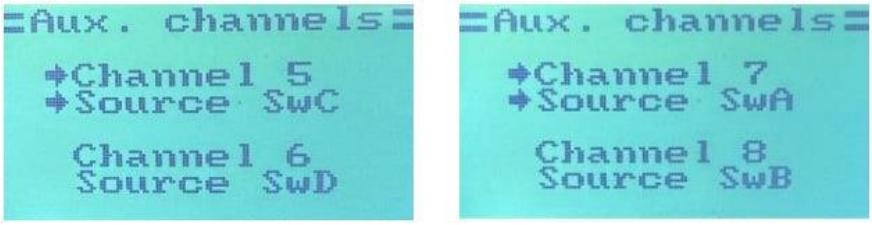
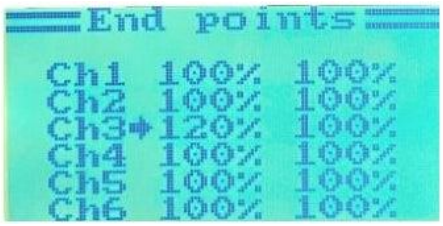


Ilustración 70: Configuración FS-i6X (4)

Pasos: Mantener presionado OK -> Functions Setup -> Aux Channels -> Configura: CH5: SwC, CH7: SwA, otros CH: None -> Mantener presionado CANCEL para guardar y salir.

5. Ajuste del extremo del Canal 3 a 120%

Este ajuste permite establecer el rango completo de movimiento del acelerador para una calibración correcta del sistema de control.



Pasos: Mantener presionado OK -> Functions Setup -> End Point -> Selecciona Channel 3 -> Colocar el acelerador en su posición mínima -> Cambiar 100% a 120% -> Mantener presionado CANCEL para guardar y salir.

Ilustración 71: Configuración FS-i6X (5)

6. Configuración del Failsafe

El Failsafe asegura que el dron tome medidas de seguridad en caso de pérdida de señal, como colocar el acelerador en su posición mínima.

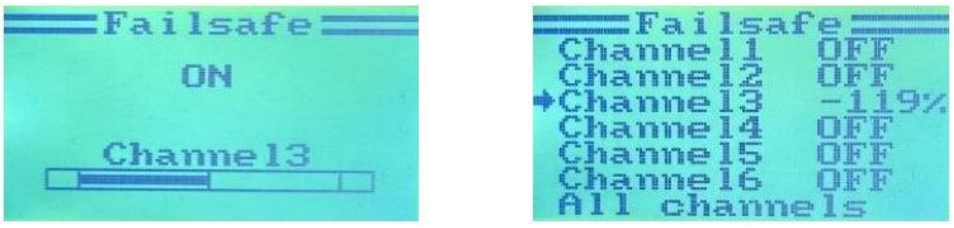


Ilustración 72: Configuración FS-i6X (6)

Pasos: Mantener presionado OK -> System Setup -> RX Setup -> Failsafe -> Activa Channel 3 -> Colocar el acelerador en su posición mínima -> Mantener presionado CANCEL para guardar y salir.

7. Restauración del extremo del Canal 3 a 100%

Este paso es necesario para restablecer el rango del acelerador tras configurar el Failsafe.

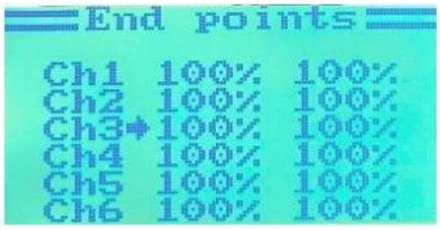


Ilustración 73: Configuración FS-i6X (7)

Pasos: Mantener presionado OK -> Functions Setup -> End Point -> Seleccionar Channel 3 -> Colocar el acelerador en su posición mínima -> Cambiar 120% a 100% -> Mantener presionado CANCEL para guardar y salir.

4.2.2 Controlador Pixhawk

Los siguientes pasos de este apartado se ven más claros en el vídeo tutorial, pero se explicarán a continuación. En todo momento las hélices tienen que estar desmontadas. Partir desde el punto en el que está el Pixhawk conectado al PC, y cargado el firmware, como hemos visto en Conexión con el Autopiloto.

1. Calibrar el acelerómetro

Colocar el dron en diferentes posiciones según las indicaciones de Mission Planner para calibrar los sensores inerciales.

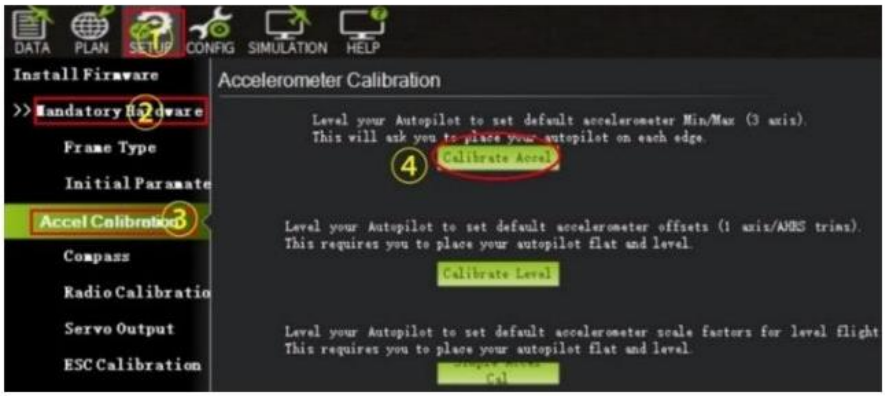


Ilustración 74: Calibración, acelerómetro

El acelerómetro detecta las inclinaciones del dron. La calibración asegura que el Pixhawk interprete correctamente las inclinaciones y movimientos.

2. Calibrar la brújula (compass)

Una vez empezada la calibración se debe realizar rotaciones del dron en todos los ejes hasta que las barras verdes se completen.

Ilustración 75: Calibración, calibrar brújula

La brújula ayuda al dron a orientarse respecto a los puntos cardinales. Una brújula mal calibrada puede causar desviaciones de rumbo.

3. Calibrar el transmisor de radio

Configurar los canales del mando RC en Mission Planner. Asegúrate de invertir el canal 2 (Pitch), según las indicaciones. Esta parte es para conocer los límites de los diferentes botones del mando, y simplemente hay que usar todo lo que se ve en el mando de la ilustración 76, llevando hasta sus extremos.

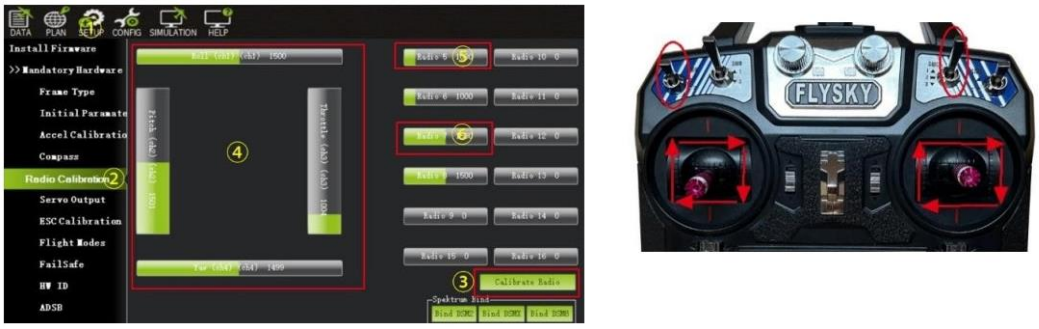


Ilustración 76: Calibración, limites transmisor

La calibración de la radio permite que el Pixhawk interprete correctamente las señales del mando para controlar el dron.

4. Configurar los modos de vuelo

Asignar diferentes modos de vuelo (por ejemplo, estabilizado, althold o RTL) a los interruptores de la emisora.

Ilustración 77: Calibración, modos de vuelo

Esto permite al piloto cambiar entre diferentes configuraciones durante el vuelo, adaptándose a distintas necesidades (manual, autónomo, etc.).

5. Configurar el sistema de seguridad (Fail Safe)

Establecer condiciones para que el dron regrese automáticamente al punto de origen (Return to Launch - RTL) si pierde la señal de la emisora.

Ilustración 78: Calibración, Failsafe

Evita perder el dron en caso de desconexión. Comprobar que el valor de radio 3 sea < 975 garantiza que el RTL se active correctamente.

6. Asignar la función RTL al canal 7

Configurar el canal 7 para activar la función de regreso al punto de despegue (RTL). Proporciona una forma manual de iniciar el RTL desde el mando en caso de emergencia.

Ilustración 79: Calibración, RTL

7. Prueba de motores

Antes de probar, asegurarse de que todos los cables de señal de los ESC estén conectados correctamente al Pixhawk. La batería alimenta el dron, permitiendo la prueba de los motores.

Verificar que los motores giren en las direcciones correctas: motores 1 y 2 en sentido antihorario (CCW) y motores 3 y 4 en sentido horario (CW).

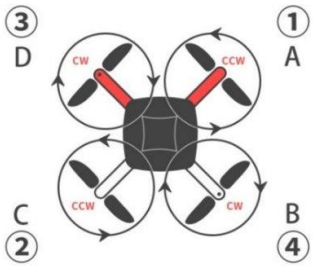


Ilustración 80: Calibración, Prueba de motores

Un giro incorrecto de los motores puede desestabilizar el dron y hacerlo incontrolable durante el vuelo.

4.3 Documentación del Montaje y Calibraciones

1.Assemble the F450 drone kit. (2023). Obtenido de @hawkswork1911 YouTube:https://www.youtube.com/watch?v=WjM5gVHy3Pw&list=PLuo-fIple9k90donChk-0l52ud7gy6K4t

3.Bind the transmitter and receiver. (2023). Obtenido de @hawkswork1911 YouTube: https://youtu.be/R21PWt_QTQ8?si=MsDAD__cEb_SkHRV

4.Calibrate Pixhawk flight control for F450. (2023). Obtenido de @hawkswork1911 YouTube: https://youtu.be/nGI5PJAdTvw?si=bZBGYTPu_91ncKDi

5.Calibrate ESCs. (2023). Obtenido de @hawkswork1911 YouTube: https://youtu.be/qdwG6ZUqDdI?si=6_X8ihy9s7Ye95MJ

F450 Guide Book. (s.f.). Obtenido de HAWK´S WORK: https://www.hawks-work.com/pages/f450_guide_book

Hawk's Work F450 Drone Kit Assembly. (2024). Obtenido de @OmniRobotix YouTube: https://www.youtube.com/watch?v=b1R6LzGlxo8&t=822s

4.4 Dificultades en modo manual (STABILIZE) al volar en interiores sin GPS

Cuando se vuela en interiores, el dron no puede adquirir señal GPS suficiente. Esto imposibilita el uso de modos de vuelo que dependen de la posición absoluta (Loiter, PosHold, Auto, etc.). Como alternativa, el modo STABILIZE es puramente manual y no requiere GPS. Sin embargo, esto acarrea las siguientes complicaciones:

1. **No retención automática de posición ni altura**
    - **Posición horizontal**: En STABILIZE, el piloto debe corregir constantemente con los joysticks para contrarrestar cualquier deriva. El FC (controlador de vuelo) no “frena” el dron, por lo que tiende a moverse en alguna dirección si no se corrige manualmente.
    - **Altitud**: Aunque el barómetro mide la altura, no interviene automáticamente para mantenerla (como sí haría AltHold). Por tanto, es necesario controlar el acelerador (throttle) de forma activa para que el dron no ascienda o descienda.
2. **Ajustes y calibraciones críticas**

- **Acelerómetro y PID**: Un desajuste en la calibración del acelerómetro puede hacer que el dron se incline levemente sin el stick centrado. Además, unos PID por defecto pueden no adaptarse bien al peso o a la configuración específica, lo que deriva en inestabilidad o excesiva deriva lateral.
- **Brújula/GPS**: Dentro de casa, el GPS no obtiene "fix" y la brújula puede presentar interferencias, pero en STABILIZE esto no afecta directamente a la estabilidad (ya que no la gestiona el FC por GPS). Aun así, una brújula mal calibrada puede generar problemas en transiciones a otros modos.

3. **Dependencia total del piloto**
   - Al ser un modo de vuelo manual, si el piloto no compensa adecuadamente, el dron se irá a un lado, puede acercarse a paredes u otros obstáculos. La concentración debe ser máxima para no perder el control en espacios reducidos.
4. **Alternativas parciales sin GPS**
   - **AltHold**: Aunque no fija la posición horizontal, sí mantiene automáticamente la altitud mediante el barómetro. Reduce la carga de trabajo del piloto en el eje vertical, pero sigue siendo necesario corregir la deriva en horizontal.
   - **Sensores de flujo óptico**: Algunas configuraciones añaden un sensor de flujo óptico o un telémetro (lidar/sonar) que permiten modos de "loiter indoor" sin GPS, basados en la lectura del suelo. Esto puede mantener la posición horizontal con más facilidad.

En conclusión, al volar en STABILIZE dentro de un recinto sin GPS, es normal que el dron no se quede fijo en el aire y exija mando continuo. Ello no implica un mal funcionamiento, sino que es la naturaleza de un modo totalmente manual.

4.5 Observaciones del montaje

Durante el montaje y pruebas posteriores en manual se han observado diferentes problemas o inconvenientes que se ha considerado que es preferible dejar por escrito en este trabajo:

Motores

A la hora del montaje de los motores en el chasis, los tornillos que provee el kit son demasiado pequeños, siendo bastante difícil llegar a atornillarlo, dando por consiguiente poca fiabilidad frente a la sujeción de estos. En uno de los

motores fue imposible su atornillado y se usaron en sustitución unos tornillos de otra parte del montaje, un poco más largos. En el caso de que se quieran sustituir, es importante comprobar su largura, ya que unos demasiado largos podrían dañar el motor, al atravesar demasiado el taladro.

Placa de distribución de voltaje

Ya se ha hablado del problema con la placa de distribución de energía a los motores, en la que los cables venían mal colocados. En un principio no ha habido problema con la colocación que se le ha dado, pero es conveniente investigar si pudiera dar problemas a la larga.

Sistema de vídeo

Respecto al sistema de vídeo no se va a montar en el dron, ya que no se dispone de un soporte para la colocación de la cámara. Existe la opción de pegar todo el sistema al chasis con las tiras adhesivas que vienen con el kit y la transmisión de vídeo, pero se ha descartado debido a la fuerza del pegamento. Lo óptimo sería un soporte para amortiguar las vibraciones del dron, pero en este caso no disponemos de ello, y como no se van a realizar vuelos de prueba con el dron, actualmente no es imperante la integración de este sistema en el conjunto.

En caso de ser necesario se podría fabricar con impresión 3D, pero se debería comprobar el payload del dron con el soporte, las dimensiones de la cámara, las opciones de ubicación del sistema, y las conexiones con el sistema de transmisión.

CAPÍTULO 5: DESARROLLO DE SOFTWARE

En este capítulo se va a describir el proceso completo de diseño, programación y pruebas del software que permitirá el control del dron desde un ordenador, así como la recepción y visualización del vídeo transmitido por la cámara FPV.

Dado que el desarrollo de software para un dron implica riesgos asociados a posibles fallos en el código que pueden comprometer la estabilidad y seguridad del vuelo, se ha optado por dividir este proceso en dos fases diferenciadas:

- **Fase 1: Desarrollo mediante simulación.**
- **Fase 2: Implementación en el dron real.**

Esta división permite minimizar riesgos y garantizar que el código desarrollado es funcional antes de aplicarlo sobre el dron físico.

5.1 Simulación: Máquina virtual con Ubuntu y SITL

Para el desarrollo y validación del software de control de drones, se realizaron pruebas en un modelo de dron antes de que se implemente en un dispositivo físico real. Se prefiere un entorno de simulación basado en **SITL (Software In The Loop)** para su uso. Este entorno permite simular el comportamiento del dron, por lo que puede volar, recibir comandos y enviar datos de telemetría sin los riesgos involucrados en probar código experimental en un dron real.

Considerando que las herramientas de simulación y las interfaces de programación de aplicaciones (API) utilizadas son más estables y compatibles con sistemas operativos Linux, la plataforma base utilizada es **Ubuntu 20.04 LTS**. Esto se ejecuta en un entorno virtual utilizando **Oracle VirtualBox**.

Este entorno es la base para toda la ingeniería y depuración del software del proyecto. En la Fase 1, todas las pruebas y ajustes se llevarán a cabo en el dron simulado. Solo después de que el resultado sea estable y satisfactorio se trasladará el código al dron real en la Fase 2.

Instrucciones de creación del entorno de trabajo

Instalación de Ubuntu 20.04 en VirtualBox

1. **Descarga Ubuntu 20.04 LTS:**

   Se descarga la imagen ISO de Ubuntu 20.04 LTS desde la página oficial

2. **Creación de la Máquina Virtual en VirtualBox**

   - Abrir VirtualBox y hacer clic en "Nuevo".
   - Nombre: El que se prefiera.

- Tipo: Linux.
- Versión: Ubuntu (64-bit).
- Memoria RAM: 4GB mínimo (8GB recomendado si es posible).
- Procesadores: Al menos 2 (4 recomendados).
- Disco Duro: 40GB (VDI, reservado dinámicamente).
- Tarjeta de vídeo: 128MB de VRAM.

3. **Instalación de Ubuntu 20.04 en la máquina virtual**
   - Iniciar la máquina virtual recién creada e insertar la ISO descargada.
   - Seleccionar "Instalar Ubuntu".
   - Conectarse a Internet.
   - Seleccionar "Instalación normal".
   - Marcar la casilla "Descargar actualizaciones mientras se instala".
   - Seleccionar la opción "Borrar disco e instalar Ubuntu".
   - Esperar a que finalice el proceso e iniciar sesión tras el reinicio.

Configuración inicial de Ubuntu 20.04

A partir de aquí se trabajará introduciendo estos comandos en la terminal del sistema.

1. **Actualizar el sistema operativo**

   ```
   sudo apt update && sudo apt upgrade -y
   ```

2. **Instalar herramientas básicas**

   ```
   sudo apt install -y python3-pip git curl nano wget net-tools
   ```

3. **Comprobar la versión de Python:** Debe mostrar Python 3.8.x. Esta versión es compatible con DroneKit, evitando problemas con versiones más recientes como Python 3.12

   ```
   python3 --version
   ```

Instalación de SITL y ArduPilot

1. **Descargar el repositorio de ArduPilot**

```
cd ~

git clone --recurse-submodules https://github.com/ArduPilot/ardupilot.git

cd ardupilot
```

2. **Instalar las dependencias necesarias**

```
Tools/environment_install/install-prereqs-ubuntu.sh -y

. ~/.profile
```

3. **Comprobar la instalación de SITL:** Si muestra las opciones de ayuda, el simulador SITL está correctamente instalado

```
sim_vehicle.py --help
```

Ejecución del Simulador SITL

1. **Acceder al directorio de ArduCopter**

```
cd ~/ardupilot/ArduCopter
```

2. **Iniciar la simulación con SITL**

```
sim_vehicle.py -v ArduCopter --console --map
```

Se abrirán dos ventanas:

- **Consola de telemetría**: muestra datos en tiempo real del estado del dron simulado.
- **Mapa**: visualiza la posición y el movimiento del dron en el entorno virtual.

Instalación de DroneKit y pymavlink

1. **Instalar las bibliotecas necesarias**

```
pip3 install dronekit pymavlink
```

2. **Verificar la instalación:** Si aparece dronekit en la lista, la instalación se ha completado con éxito

```
python3 -m pip list | grep dronekit
```

Simulación SITL y MAVProxy

Una vez acabado estos pasos aparecerá en nuestra máquina virtual 3 ventanas a parte de la consola de comandos que ya se estaba usando.

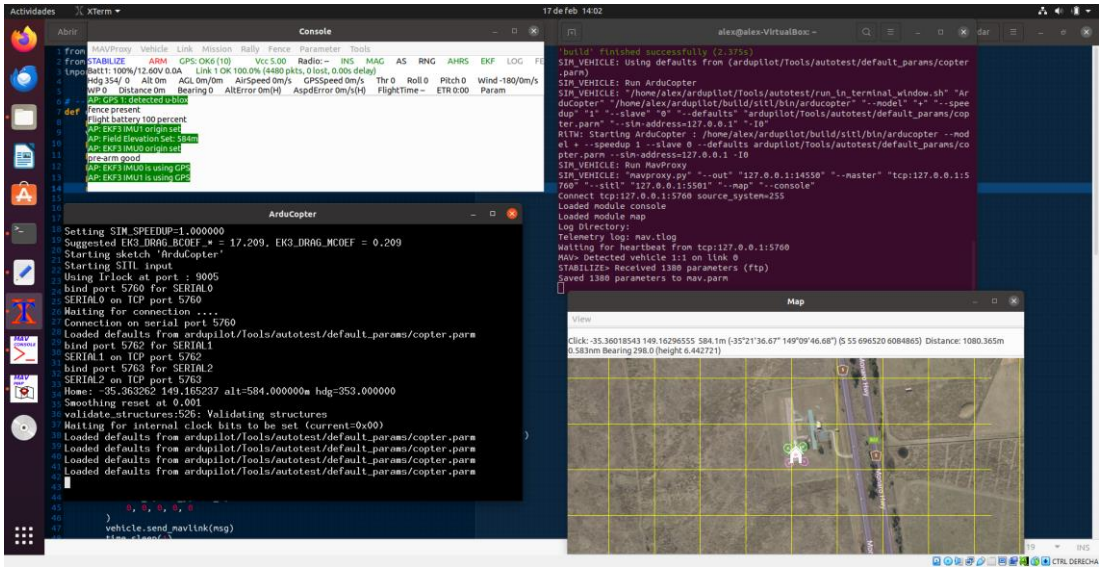


Ilustración 81: Activación de simulador SITL

Arducopter

Es una ventana no interactiva que funciona como el corazón de SITL, donde se ejecuta el firmware de Arducopter, como si existiera el dron real. Simula sensores, motores y físicas del dron.

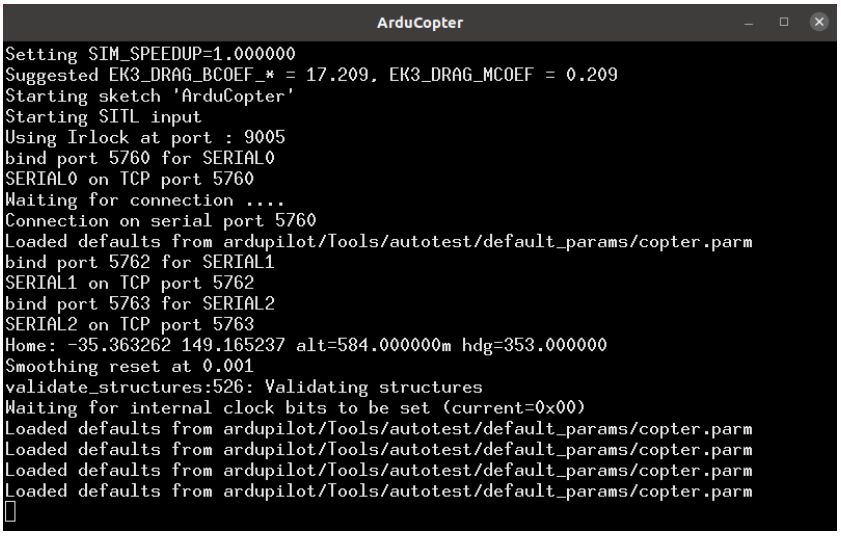


Ilustración 82: SITL ArduCopter

En la ilustración se nos muestra los mensajes del sistema, como configuración de puertos (a través de los cuales nos conectaremos con nuestros scripts de DroneKit), parámetros cargados por defecto, validación de estructuras, etc.

Además, nos mostrará que hemos conectado con el dron desde la API, en nuestro caso en el puerto 5762, SERIAL1.

Console

Es otra ventana no interactiva generada por MAVProxy que en este caso nos muestra todos los parámetros de interés del dron simulado en tiempo real. Se podría decir que es la versión visual de la comunicación por MAVProxy. En ella se nos irán mostrando actualizaciones del estado del dron, como cambios del modo de vuelo, armado, batería,...

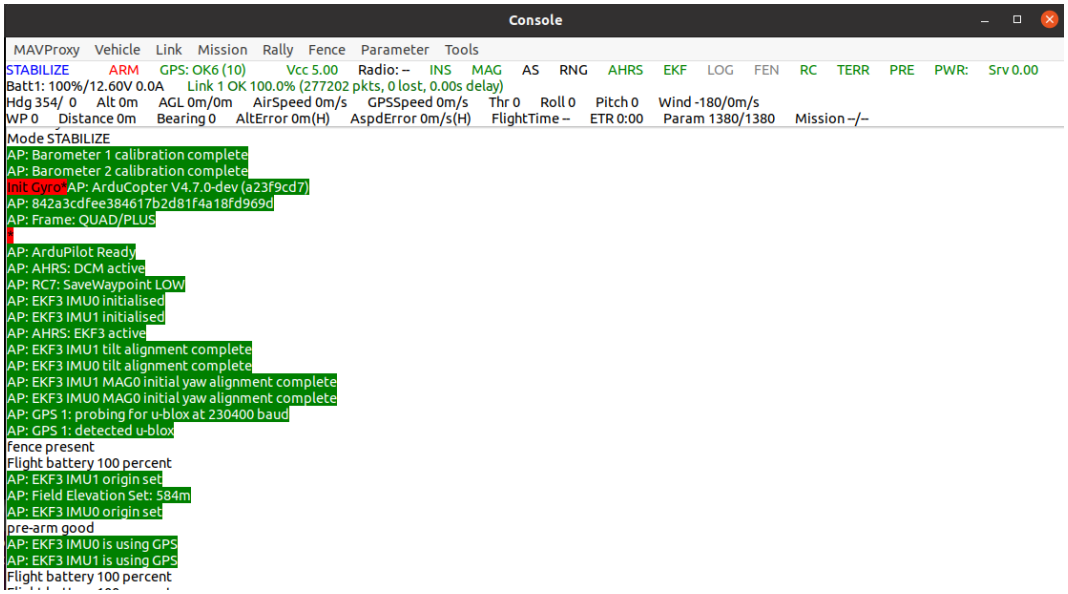


Ilustración 83: SITL MAVProxy Console

Por ejemplo, los mensajes en verde son eventos del firmware de Arducopter que informan de cosas como:

- EKF IMU0 is using GPS → La unidad de medición inercial ha cogido GPS.
- Flight battery 100 percent → Batería simulada al 100%.
- Pre-arm good → Todo OK para armar motores.

Mapa

Esta ventana se comporta igual que la ventana del mapa de Mission Planner, mostrará todos los movimientos del dron en tiempo real, además de todas las opciones del clic derecho, como volar a un punto, dibujar misiones, geocercas, etc.

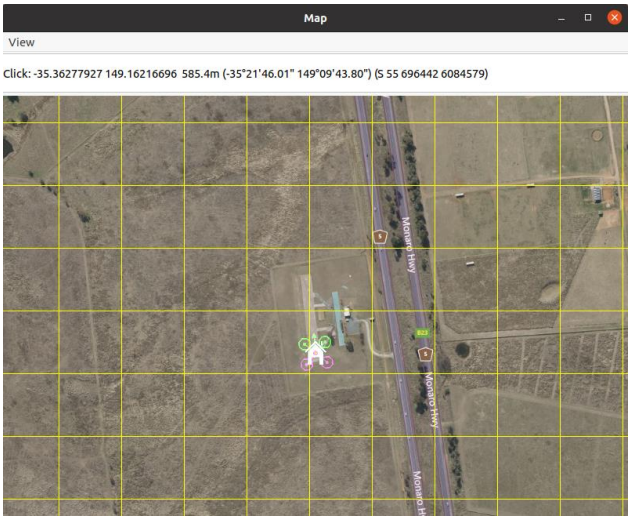


Ilustración 84: Mapa de SITL y MAVProxy

Consola de comandos

Una vez iniciada la simulación, la consola de comandos se convierte en la zona de lanzamiento de órdenes. Cualquier acción se puede realizar desde aquí. Escribiendo "help" en la consola, se muestra todos los comandos disponibles. Por ejemplo, "batreset" restaura la batería al 100%, lo cual será muy útil, ya que la batería en la simulación se gasta bastante rápido.

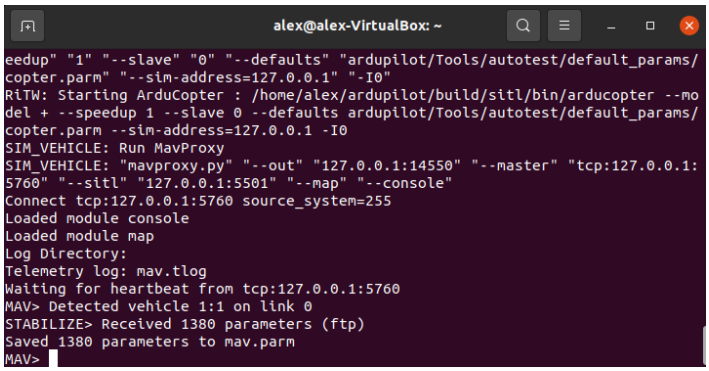


Ilustración 85: Consola control del dron

Primeros scripts de Dronekit

Dronekit es una API que dispone de bastantes ejemplos y programas ya desarrollados en su repositorio de GitHub, así que para familiarizarse con esta librería se probarán acciones simples como despegue, movimiento relativo, cambio de modo de vuelo y aterrizaje.

Estas serán la base para el desarrollo de una aplicación de simulación más funcional y estética. De la cual se reutilizará el código de control para la fase 2 del proyecto de software.

Para que la explicación de los primeros pasos sea más fluida y completa se ha creado un menú interactivo con las principales funcionalidades antes ya descritas. Además, el producto final tendrá esta estructura o partirá de ella, añadiendo una pequeña interfaz para hacer más visual este proceso de control.

No hace falta decir que en los scripts se deberá de importar principalmente las librerías de **dronekit** y **mavutil.** Para las primeras pruebas se ha importado eso:

```
from dronekit import connect, VehicleMode, LocationGlobalRelative
from pymavlink import mavutil
```

Ilustración 86: Librerías principales, primeros pasos

Código

Se comienza la explicación con la conexión con el vehículo, esta se realiza mediante IP:puerto, en este caso como es una simulación dentro del ordenador se usa la dirección de localhost y el puerto se puede elegir entre los que se muestran en la ventana Ardupilot al iniciar la simulación.

```
def conectar(puerto):
    print(f"Conectando al dron en el puerto {puerto}...")
    vehicle = connect(f"tcp:127.0.0.1:{puerto}", wait_ready=True)
    print("Dron conectado exitosamente")
    print("Modo de vuelo:", vehicle.mode.name)
    print("Batería:", vehicle.battery)
    print("Coordenadas:", vehicle.location.global_relative_frame)
    return vehicle
```

Ilustración 87: Script conexión SITL

Normalmente se ha usado el puerto 5762. Además de la conexión, imprimimos diferentes parámetros del dron. Al final se devuelve el objeto vehículo para su posterior utilización.

A continuación, todas las acciones se toman como comandos, por lo que el **modo de vuelo** necesario es el guiado **“GUIDED”.** Que se activará en la función de despegue, que permite posteriormente comenzar a mover el dron.

```
def despegar(vehicle, altura):
    print(f"Despegando a {altura}m...")
    vehicle.mode = VehicleMode("GUIDED")
    vehicle.armed = True

    while not vehicle.armed:
        print("Esperando armado...")
        time.sleep(1)

    vehicle.simple_takeoff(altura)

    while True:
        altitud_actual = vehicle.location.global_relative_frame.alt
        print(f"Altitud: {altitud_actual:.2f}m")
        if altitud_actual >= altura * 0.95:
            print("Altitud objetivo alcanzada")
            break
        time.sleep(1)
```

Ilustración 88: Script despegue

Solo necesita especificarle la altura a la que tiene que despegar, y después de armar los motores, simple_takeoff() lanza la orden de despegar hasta la altura especificada. Irá mostrando la altura a la que se encuentra el dron en cada segundo hasta llegar a la deseada.

Una vez hecho esto se podrá mover el dron. Para ello se va a usar un movimiento basado en velocidades, es decir, hay que introducir la velocidad en cada eje X Y Z y el tiempo que va a estar moviéndose.

```
36
37 def mover_ned(vehicle, vel_x, vel_y, vel_z, duracion):
38     print(f"Moviendo: X={vel_x} m/s, Y={vel_y} m/s, Z={vel_z} m/s durante {duracion}s")
39     for _ in range(duracion):
40         msg = vehicle.message_factory.set_position_target_local_ned_encode(
41             0, 0, 0, mavutil.mavlink.MAV_FRAME_LOCAL_NED,
42             0b0000111111000111,
43             0, 0, 0,
44             vel_x, vel_y, vel_z,
45             0, 0, 0, 0, 0
46         )
47         vehicle.send_mavlink(msg)
48         time.sleep(1)
49
```

Ilustración 89: Script movimiento relativo

Es necesario especificar el sentido del movimiento por ejes del dron, ya que por ejemplo el sentido positivo del eje Z sería mirando hacia el suelo. Por lo que si queremos que el dron suba tendremos que proporcionarle una velocidad negativa.

Los movimientos en el eje X son hacia delante, velocidad positiva y hacia atrás negativa. En el Y, izquierda será velocidad negativa y derecha positiva.

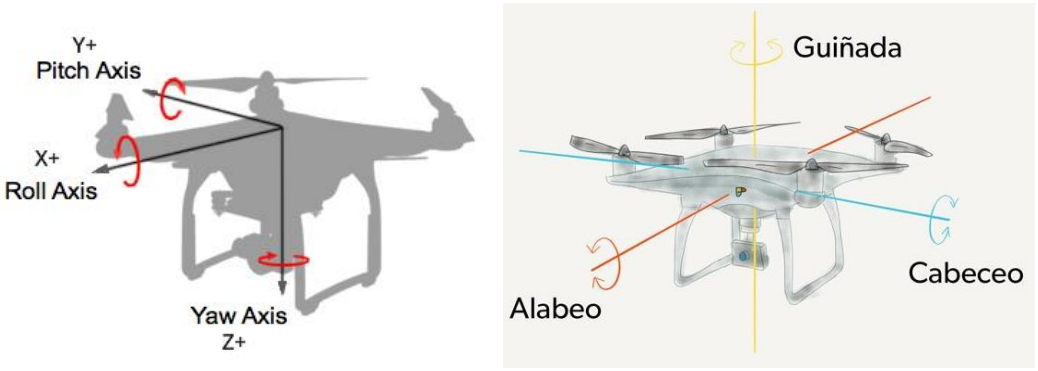


Ilustración 90: Ejes y rotaciones del dron

Otras funciones son cambiar el modo de vuelo entre ellos RTL(vuelta a la posición Home), STABILIZE, LOITER y aterrizaje.

Por último, la función cerrar_conexión() simplemente usa un vehicle.close().

```python

def cambiar_modo(vehicle, modo):
    print(f"Cambiando modo a {modo}...")
    vehicle.mode = VehicleMode(modo)
    time.sleep(2)
    print(f"Modo actual: {vehicle.mode.name}")


def aterrizar(vehicle):
    print("Aterrizando...")
    vehicle.mode = VehicleMode("LAND")
    time.sleep(5)


def cerrar_conexion(vehicle):
    print("Cerrando conexión con el dron...")
    vehicle.close()
    print("Conexión cerrada")

```

Ilustración 91: Script cambio de modos de vuelo y cerrar conexión

Ahora se implementa todo en un menú muy simple, con los diferentes inputs necesarios para las funciones.

```python
# ------------------------- MENÚ INTERACTIVO ------------------------- #
def menu_interactivo():
    puerto = input("Introduce el puerto (5760 o 5762): ")
    vehicle = conectar(puerto)

    while True:
        print("\n--- MENÚ DE CONTROL DEL DRON ---")
        print("1. Despegar")
        print("2. Mover (NED)")
        print("3. Cambiar modo de vuelo")
        print("4. Aterrizar")
        print("5. Cerrar conexión y salir")
        opcion = input("Selecciona una opción: ")

        if opcion == "1":
            altitud = float(input("Introduce la altitud de despegue (m): "))
            despegar(vehicle, altitud)

        elif opcion == "2":
            vel_x = float(input("Introduce velocidad en X (adelante + / atrás -): "))
            vel_y = float(input("Introduce velocidad en Y (derecha + / izquierda -): "))
            vel_z = float(input("Introduce velocidad en Z (bajar + / subir -): "))
            duracion = int(input("Introduce la duración del movimiento (s): "))
            mover_ned(vehicle, vel_x, vel_y, vel_z, duracion)

        elif opcion == "3":
            modo = input("Introduce el modo de vuelo (GUIDED, LOITER, LAND, RTL, STABILIZE, etc.): ")
            cambiar_modo(vehicle, modo)

        elif opcion == "4":
            aterrizar(vehicle)

        elif opcion == "5":
            cerrar_conexion(vehicle)
            break

        else:
            print("Opción no válida. Inténtalo de nuevo.")


if __name__ == "__main__":
    menu_interactivo()
```

Ilustración 92: Script primera versión menú

Pruebas

- **Conexión**: se introduce el puerto y se muestran los parámetros obtenidos.

```
alex@alex-VirtualBox:~$ python3 P1.py
Introduce el puerto (5760 o 5762): 5762
Conectando al dron en el puerto 5762...
Dron conectado exitosamente
Modo de vuelo: STABILIZE
Batería: Battery:voltage=12.6,current=0.0,level=100
Coordenadas: LocationGlobalRelative:lat=-35.3632621,lon=149.1652374,alt=-0.003
```

Ilustración 93: Vista de la conexión

- **Despegue:** se introduce los datos en la consola de comandos y se verá como el valor Alt en la consola de MAVProxy irá cambiando. En el mapa no se ve nada ya que es una vista de satélite, y el eje Z no le percibimos.

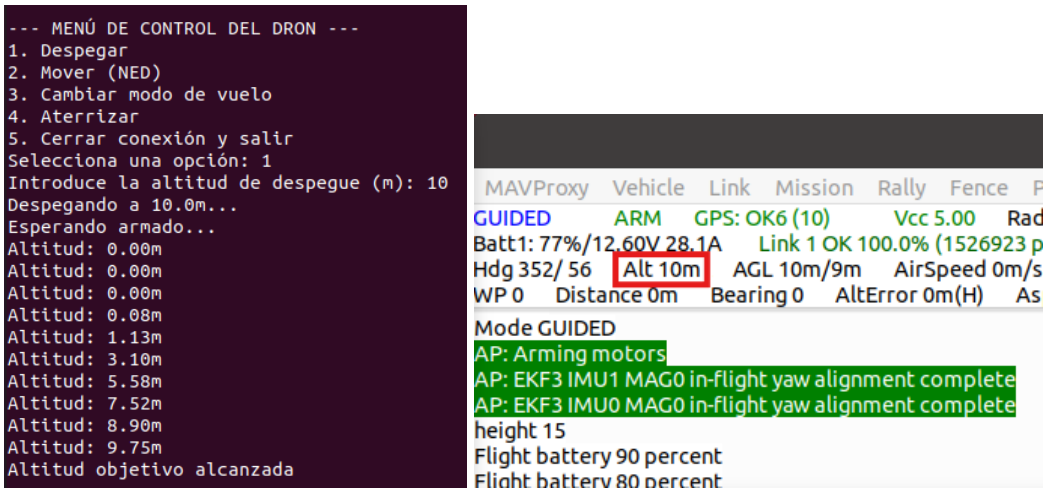


Ilustración 94: Vista del despegue

- **Movimiento NED:** después de introducir las velocidades y duración se ve como el dron el mapa se desplaza de la posición Home (icono de la casa). En este caso solo se ha movido a 1m/s en el eje X durante 5 segundos, es decir, se ha desplazado 5m hacia adelante.

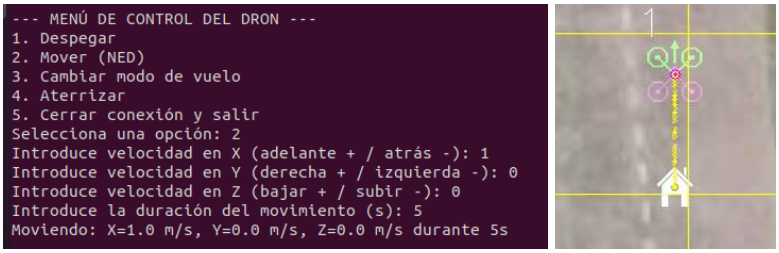


Ilustración 95: Vista movimiento NED1

- **Cambio de modo de vuelo a RTL:** con esta prueba se aprovechan dos funciones, el cambio de modo y el aterrizaje, en este caso el aterrizaje va a ser volviendo a la posición home, si fuese el modo LAND aterrizaría en las coordenadas en las que se encuentre en ese momento. En cuanto se cambie el modo se observa en el mapa la trayectoria de vuelta a casa, primero se desplazará en el plano XY y después en el Z.

```
--- MENÚ DE CONTROL DEL DRON ---
1. Despegar
2. Mover (NED)
3. Cambiar modo de vuelo
4. Aterrizar
5. Cerrar conexión y salir
Selecciona una opción: 3
Introduce el modo de vuelo (GUIDED, LOITER, LAND, RTL, STABILIZE, etc.): RTL
Cambiando modo a RTL...
Modo actual: RTL
```

Ilustración 96: Vista modo RTL

Con estas pruebas se ha comprobado un poco las funcionalidades básicas. Es importante que antes de realizar otra acción dejemos terminar completamente a la anterior, si no la API se quedará bloqueada y tendremos que reiniciar todo.

Evolución del desarrollo

El desarrollo de la aplicación ha seguido una evolución estructurada en distintas versiones. Vamos a explicarlas brevemente como introducción al definitivo.

Versión 1: Primer menú interactivo.

Aquí se implementaron las principales funcionalidades del dron, diferenciando el menú entre funciones de suelo y aire, añadiendo ya una detección de batería en estado crítico.

- **SUELO:** Despegar, consultar batería y GPS, y salir de Conexión.
- **AIRE:** ir a punto XYZ, cambiar altura, cambiar modo de vuelo, RTL, aterrizar, y consultar batería y GPS.
- **CRÍTICO:** RTL o Aterrizaje, y consultar batería y GPS.

El menú era por consola, imprimiendo las opciones y datos, sin interfaz.

Versión 2: Implementación de hilos

En la segunda versión, se mejoró la estabilidad y eficiencia del software mediante la implementación de hilos, usando **Threading**, lo que permitió ejecutar ciertas funciones en segundo plano sin bloquear la ejecución principal.

Las mejoras incluyeron:

- **Hilo de monitoreo de batería**: Cuando la batería baja del 25%, el dron activa automáticamente el modo RTL, se mostró una alerta visual en el menú y se activan eventos para cerrar los procesos y menú.
- **Separación del código en módulos**: Se modularizó el código separando la lógica de control (acciones.py) de la interacción del usuario (menu.py).
- **Ejecución fluida**: Se corrigieron bloqueos al ejecutar comandos, permitiendo la ejecución de múltiples tareas simultáneamente.

Version3: Integración de una interfaz gráfica con Tkinter

La tercera versión consistió principalmente en una mejora para el usuario, sustituyendo el menú en terminal por una interfaz gráfica desarrollada con Tkinter. En las que se mostraban botones asociados a funciones, y etiquetas para mostrar información en tiempo real como estado del dron, batería y velocidad y posición respecto al home.

Versión 4: mejora de interfaz con ttk y corrección de bloqueos

Esta es la versión final, que se explicará en profundidad en el siguiente apartado. Aquí se ha mejorado el aspecto de la interfaz, usando Custom Tkinter (Ttk). Distribuyendo los botones y los recuadros para introducir valores, y dividiendo la interfaz en zonas, parámetros, menú de suelo y menú de aire. Además de una zona de avisos/mensajes en el centro de la interfaz.

Algunas acciones bloqueaban la información del cuadro de parámetros hasta que acabasen así que se han creado funciones anidadas dentro de cada una de las acciones, las cuales ejecutamos mediante hilos, para que la información se actualice al momento y haya una mayor fluidez.

Se han reducido las funciones disponibles a las principales, prescindiendo de algunas redundantes.

Programa final

Introducción

El programa ControlDron.py es la versión final del software del control del dron, desarrollado en Python utilizando las bibliotecas:

- **DroneKit** → Para conectar y controlar el dron simulado en SITL.
- **Tkinter (ttk)** → Para la interfaz gráfica.
- **Threading** → Para manejar eventos en segundo plano sin bloquear la interfaz.
- **Pymavlink** → Para enviar comandos personalizados al dron.

Permite controlar el dron simulado en SITL a través de una interfaz gráfica, mostrando información en tiempo real , transmitiendo acciones al dron mediante botones.

Estructura

El programa está creado en forma de objeto, y consta de la siguiente estructura:

1. Inicialización de la interfaz gráfica, conexión con el dron y lanzamiento de procesos paralelos.
2. Creación de los botones y etiquetas para visualización de la telemetría.
3. Acciones de movimiento o control del dron.

Código

Llamada de librerías

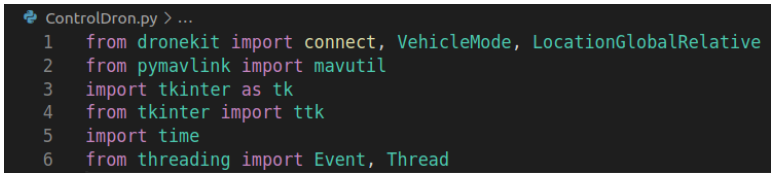


Ilustración 97: Scripts Control -> Librerías

- Se importan DroneKit y Pymavlink para comunicarse con el dron.
- Se usa Tkinter (ttk) para la interfaz gráfica.
- Se importan hilos (Threading) para ejecutar tareas en segundo plano.

Clase DroneControl: __init__

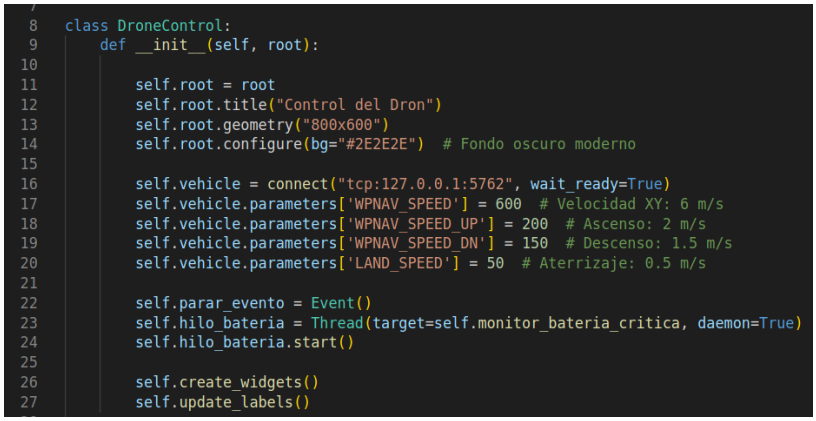


Ilustración 98: Scripts Control -> __init__

- **Líneas 11-12**: se crea la ventana principal con tkinter con un tamaño de ventana (800x600) y un fondo oscuro (#2E2E2E).
- **Líneas 16-20:** conexión con el vehículo simulado mediante IP (localhost) y puerto, con wait_ready=True que espera a que todos los parámetros del dron estes listos antes de continuar. Y después, configuración de velocidades.

- **Líneas 22-24:** lanzamiento de hilo monitor de la batería y creación del evento para terminar su ejecución.

Clase DroneControl: create_widgets

Esta es la función que se encarga de crear todos los botones y etiquetas de la interfaz.

```
28
29      def create_widgets(self):
30          style = ttk.Style()
31          style.configure("TLabel", foreground="black", background="light gray", font=("Arial", 12))
32          style.configure("TButton", padding=5, font=("Arial", 10))
33
34          main_frame = ttk.Frame(self.root, padding=10)
35          main_frame.pack(fill="both", expand=True)
36
37          # Mitad superior - Parámetros
38          param_frame = ttk.LabelFrame(main_frame, text="PARAMETROS DE INTERÉS", padding=10)
39          param_frame.pack(fill="both", expand=True, pady=10)
40
41          self.status_label = ttk.Label(param_frame, text=f"Estado: {self.vehicle.mode.name}")
42          self.status_label.grid(row=0, column=0, padx=20, pady=5)
43
44          self.battery_label = ttk.Label(param_frame, text=f"Batería: {self.vehicle.battery.level}%")
45          self.battery_label.grid(row=0, column=1, padx=20, pady=5)
46
47          self.battery_alert = ttk.Label(param_frame, text="Batería OK")
48          self.battery_alert.grid(row=0, column=2, padx=20, pady=5)
49
50          self.speed_xy_label = ttk.Label(param_frame, text=f"Velocidad XY: -- m/s")
51          self.speed_xy_label.grid(row=1, column=0, padx=20, pady=5)
52
53          self.speed_z_label = ttk.Label(param_frame, text=f"Velocidad Z: -- m/s")
54          self.speed_z_label.grid(row=1, column=1, padx=20, pady=5)
55
56          self.position_label = ttk.Label(param_frame, text="Posición: Cargando...")
57          self.position_label.grid(row=1, column=2, padx=20, pady=5)
58
59          self.message_label = ttk.Label(main_frame, text="", foreground="red")
60          self.message_label.pack(pady=5)
```

Ilustración 99: Scripts Control -> create_widgets (1)

Para empezar, establece un estilo de texto para los botones y etiquetas. Y se crea el frame sobre el que estarán los widgets.

Se divide el frame principal en parte superior e inferior. Primero se configura la parte superior de la interfaz, el bloque de parámetros, en el cual se localizan las etiquetas, el estilo de cada una y su posición. La última etiqueta (message_label) es en la que se imprimirán los diferentes avisos de la interfaz.

Se asigna un valor a determinadas etiquetas como la velocidad o modo, pero a otras en las que hacen falta realizar cálculos, se asignará su valor en la función que actualiza todo este bloque.

Una vez configurado la parte de arriba se procede a la parte inferior del frame, donde se encuentran las acciones del dron, mediante botones asociados a funciones y cuadros donde se introducirán los valores necesarios como altura de despegue o coordenadas XYZ a las que moverse.

```
61
62          # Mitad inferior - Botones
63          action_frame = ttk.Frame(main_frame)
64          action_frame.pack(fill="both", expand=True, pady=10)
65
66          suelo_frame = ttk.LabelFrame(action_frame, text="SUELO")
67          suelo_frame.pack(side="left", fill="both", expand=True, padx=10, pady=10)
68
69          ttk.Button(suelo_frame, text="Armar y Despegar", command=self.despegar).pack(pady=5)
70          ttk.Label(suelo_frame, text="Altitud de Despegue:").pack()
71          self.altitud_entry = ttk.Entry(suelo_frame)
72          self.altitud_entry.pack(pady=5)
73
74          ttk.Button(suelo_frame, text="Cerrar Programa", command=self.cerrar_programa).pack(pady=5)
75
76          aire_frame = ttk.LabelFrame(action_frame, text="AIRE")
77          aire_frame.pack(side="right", fill="both", expand=True, padx=10, pady=10)
78
79          ttk.Button(aire_frame, text="Ir a XYZ", command=self.ir_a_xyz).pack(pady=5)
80          ttk.Label(aire_frame, text="Coordenada X:").pack()
81          self.x_entry = ttk.Entry(aire_frame)
82          self.x_entry.pack(pady=5)
83
84          ttk.Label(aire_frame, text="Coordenada Y:").pack()
85          self.y_entry = ttk.Entry(aire_frame)
86          self.y_entry.pack(pady=5)
87
88          ttk.Label(aire_frame, text="Altitud (positiva hacia arriba):").pack()
89          self.z_entry = ttk.Entry(aire_frame)
90          self.z_entry.pack(pady=5)
91
92          ttk.Button(aire_frame, text="Aterrizar", command=self.aterrizar).pack(pady=5)
93          ttk.Button(aire_frame, text="RTL", command=self.rtl).pack(pady=5)
94
```

Ilustración 100: Scripts Control -> create_widgets (2)

Clase DroneControl: update_labels

Esta función es la que se encarga de actualizar los valores del bloque de parámetros de interés.

```
94
95      def update_labels(self):
96          self.status_label.config(text=f"Estado del Dron: {self.vehicle.mode.name}")
97          self.battery_label.config(text=f"Batería: {self.vehicle.battery.level}%")
98
99          if self.vehicle.battery.level <= 25:
100             self.battery_alert.config(text="BATERÍA CRÍTICA - RTL ACTIVADO", foreground="red")
101         else:
102             self.battery_alert.config(text="Batería OK", foreground="green")
103
104         velocity = self.vehicle.velocity  # [vx, vy, vz]
105         speed_xy = (velocity[0]**2 + velocity[1]**2) ** 0.5  # Magnitud en XY
106         speed_z = velocity[2]  # Velocidad en Z (positivo hacia abajo en MAVLink)
107
108         self.speed_xy_label.config(text=f"Velocidad XY: {speed_xy:.2f} m/s")
109         self.speed_z_label.config(text=f"Velocidad Z: {speed_z:.2f} m/s")
110
111         pos_ned = self.vehicle.location.local_frame
112         if pos_ned and pos_ned.north is not None:
113             self.position_label.config(text=f"Posición: X={pos_ned.north:.2f}, Y={pos_ned.east:.2f}, Z={-pos_ned.down:.2f}")
114         else:
115             self.position_label.config(text="Posición: No disponible")
116
117         self.root.after(1, self.update_labels)
```

Ilustración 101: Scripts Control -> update_labels

Usando .config en cada etiqueta se podrá actualizar su valor. Por ejemplo, con la etiqueta que muestra la alerta de la batería podemos cambiar tanto su texto como su color en función de su porcentaje. También calculará las velocidades del dron en XY y en Z, las cuales se ha decidido separar para comprobar los movimientos de despegue, aterrizaje y RTL. Se muestra la posición XYZ respecto a HOME debido a que es más intuitivo que las coordenadas de latitud y altitud que ofrece MAVProxy. La última línea permite actualizar las etiquetas a cada ciertos milisegundos, por lo que no hace falta lanzar un hilo con esta función.

Clase DroneControl: despegar

```
118
119     def despegar(self):
120
121         def arm_launch(altura):
122             if self.vehicle.mode != VehicleMode("GUIDED"):
123                 self.vehicle.mode = VehicleMode("GUIDED")
124                 if not self.vehicle.armed:
125                     self.root.update_idletasks()  # Fuerza la actualización gráfica
126                     self.vehicle.armed = True
127                     while not self.vehicle.armed:
128                         time.sleep(1)
129
130             if self.vehicle.location.global_relative_frame.alt< 0.01: self.vehicle.simple_takeoff(altura)
131             else:
132                 punto = LocationGlobalRelative(
133                     self.vehicle.location.global_relative_frame.lat,
134                     self.vehicle.location.global_relative_frame.lon,
135                     altura
136                 )
137                 self.vehicle.simple_goto(punto)
138
139             while self.vehicle.location.global_relative_frame.alt < altitud : time.sleep(1)
140
141             self.message_label.config(text="Altitud objetivo alcanzada")
142             self.root.update_idletasks()  # Fuerza la actualización gráfica
143
144         try:
145             altitud = float(self.altitud_entry.get())
146             self.message_label.config(text="Armando y en Despegue")
147
148             hilo_desp = Thread(target=arm_launch,args=(altitud,))
149             hilo_desp.start()
150
151             self.root.update_idletasks()  # Fuerza la actualización gráfica
152
153         except ValueError:
154             self.message_label.config(text="Error: Introduce una altitud válida")
155             self.root.update_idletasks()  # Fuerza la actualización gráfica
```

Ilustración 102: Scripts Control -> despegar

Se toma el valor de la altitud deseada desde la interfaz. Creando un hilo separado para que despegue sin bloquear la interfaz

En **arm_launch()** es donde se realiza el **armado de motores** y cambio de modo a **GUIDED**, modo necesario para el control por comandos. Siempre que se quiera comenzar a mover el dron después de un aterrizaje es necesario el despegue.

Si el vehículo no estaba ni armado ni en GUIDED, este se arma y se configura en guiado. Una vez hecho esto se comprueba si el vehículo está en el suelo, si es así se usa una función de despegue, y si no, se usa una función que realiza un envío al dron de un punto al que moverse.

De esta forma disponemos de una función que permite el armado, cambio a guiado, despegue y también cambio de altura.

Se va notificando con mensajes si la función se ha ejecutado y está en proceso, y si ha llegado a la altitud objetivo.

Clase DroneControl: aterrizar y rtl

Se explicarán juntas estas dos funciones debido a que tienen el mismo funcionamiento, se activa el modo de vuelo deseado, se actualizan mensajes y parámetros y el dron aterriza o vuelve a casa automáticamente.

```
156
157     def aterrizar(self):
158
159         def land():
160             while self.vehicle.location.global_relative_frame.alt > 0.1 : time.sleep(1)
161
162             self.message_label.config(text="Vehiculo aterrizado")
163             self.root.update_idletasks()  # Fuerza la actualización gráfica
164
165         self.message_label.config(text="Aterrizando...")
166         self.vehicle.mode = VehicleMode("LAND")
167         self.root.update_idletasks()  # Fuerza la actualización gráfica
168
169         hilo_aterr = Thread(target=land,args=())
170         hilo_aterr.start()
```

Ilustración 103: Scripts Control -> aterrizar

```
171
172     def rtl(self):
173         def home():
174             while self.vehicle.location.global_relative_frame.alt > 0.1 : time.sleep(1)
175
176             self.message_label.config(text="En Home")
177             self.root.update_idletasks()  # Fuerza la actualización gráfica
178
179         self.message_label.config(text="Volviendo a casa (RTL)...")
180         self.vehicle.mode = VehicleMode("RTL")
181         self.root.update_idletasks()  # Fuerza la actualización gráfica
182
183         hilo_rtl = Thread(target=home,args=())
184         hilo_rtl.start()
```

Ilustración 104: Scripts Control -> rtl

Clase DroneControl: ir_a_XYZ

A esta función se le especifica unas coordenadas XYZ, relativas a la posición HOME (0,0,0), y al igual que las demás se lanza un hilo con la función **cmd_xyz()** la cual utiliza mavutil para mandar un comando de movimiento al dron.

```
185
186     def ir_a_xyz(self):
187
188         def cmd_xyz( x, y, z):
189             msg = self.vehicle.message_factory.set_position_target_local_ned_encode(
190                 0, 0, 0, mavutil.mavlink.MAV_FRAME_LOCAL_NED,
191                 0b0000111111111000,
192                 x, y, z,
193                 0, 0, 0,
194                 0, 0, 0, 0, 0
195             )
196
197             self.vehicle.send_mavlink(msg)
198             self.vehicle.flush()
199
200         try:
201             if self.vehicle.armed:
202                 x = float(self.x_entry.get())
203                 y = float(self.y_entry.get())
204                 z = float(self.z_entry.get())
205
206                 self.message_label.config(text=f"movimiento a X={x}, Y={y}, Z={z} (respecto a HOME)")
207                 self.root.update_idletasks()  # Fuerza la actualización gráfica
208
209                 # Crear un hilo para ejecutar el movimiento sin bloquear la interfaz
210                 hilo_movimiento = Thread(target=cmd_xyz, args=(x, y, -z))
211                 hilo_movimiento.start()
212
213             else:
214                 self.message_label.config(text=f" Necesitas despegar primero ")
215                 self.root.update_idletasks()  # Fuerza la actualización gráfica
216
217
218         except ValueError:
219             self.message_label.config(text="Error: Introduce valores numéricos válidos")
220             self.root.update_idletasks()  # Fuerza la actualización gráfica
```

Ilustración 105: Scripts Control -> ir_a_xyz

Clase DroneControl: monitor_bateria_critica

```
221
222     #Crear el Hilo de Control de Batería (CRÍTICO)
223     def monitor_bateria_critica(self):
224         while not self.parar_evento.is_set():
225             if self.vehicle.battery.level is not None and self.vehicle.battery.level <= 25:
226                 self.message_label.config(text=f"\n \n \n WARNING: BATERIA CRITICA \n Activando RTL... ")
227                 self.vehicle.mode = VehicleMode("RTL")
228
229                 self.root.update_idletasks()  # Fuerza la actualización gráfica
230
231                 while  self.vehicle.location.global_relative_frame.alt > 0.1 or self.vehicle.armed:
232                     time.sleep(1)
233                 self.vehicle.close()
234                 self.parar_evento.set()
235                 self.root.quit()
236                 self.root.destroy()
237                 break
238
239             time.sleep(1)  # Comprobación cada 5 segundos
```

Ilustración 106: Scripts Control -> monitor_bateria_critica

Esta función se lanza en forma de hilo, actuando en segundo plano revisa constantemente la batería del dron, cuando esta baje del 25%, actualiza la zona de mensajes imprimiendo el aviso de la batería crítica. Para después activar el modo de vuelo RTL. Una vez el dron se haya posado en el suelo, la conexión se cierra, el hilo se para y el programa se cierra. Esto se hace como medida de seguridad, ya que, aunque quede un mínimo de batería no se debería de poder usar el dron.

Clase DroneControl: cerrar_programa

Cierra conexiones, para el monitor de batería, y elimina la interfaz, cerrando así todo el programa.

```
240
241     def cerrar_programa(self):
242         self.vehicle.close()
243         self.parar_evento.set()
244         self.root.quit()
245         self.root.destroy()
246
```

Ilustración 107: Scripts Control -> cerrar_programa

Clase DroneControl: main

Con root se crea la ventana principal de la interfaz gráfica, es la raíz de la interfaz. tk.TK() es el punto de inicio de cualquier aplicación en Tkinter. Después crea una instancia de DroneControl introduciéndola como parámetro root.

```
246
247 if __name__ == "__main__":
248     root = tk.Tk()
249     app = DroneControl(root)
250     root.mainloop()
251
```

Ilustración 108: Scripts Control -> main

root.mainloop() inicia el bule principal de los eventos de Tkinter, como clics de botones, entradas de texto, etc. Espera y responde a estos eventos, y mantiene la ventana abierta y funcional hasta que el usuario la cierre.

Manual de uso

Una vez se hayan instalado todas las dependencias explicadas y ordenadas en el apartado instrucciones de creación del entorno de trabajo, estará listo para comenzar a usar el control del dron.

Si se ha cerrado la simulación SITL simplemente en una terminal nueva se introducirá el comando: **sim_vehicle.py -v ArduCopter –console –map**

En otra terminal se iniciará el programa de control, usando el comando: **python3 ruta/nombre_programa.py** . Este se abrirá y se conectará automáticamente a la simulación. En el caso de que el puerto sea distinto al que se usa en el programa simplemente tienes que modificar el puerto en la línea connect de la clase DroneControl.

En la ventana de ArduCopter se pueden ver los puertos disponibles, y el mensaje de que hemos establecido conexión con un puerto.

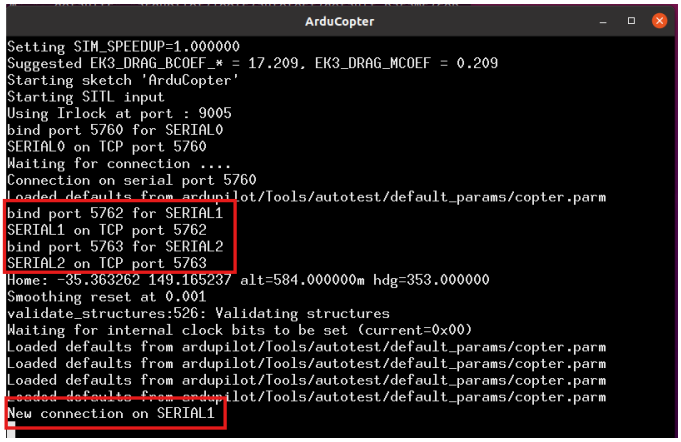


Ilustración 109: Visualizar puertos y conexión

Una vez esté todo arrancado, se debería ver 4 ventanas emergentes, a parte de las 2 terminales.

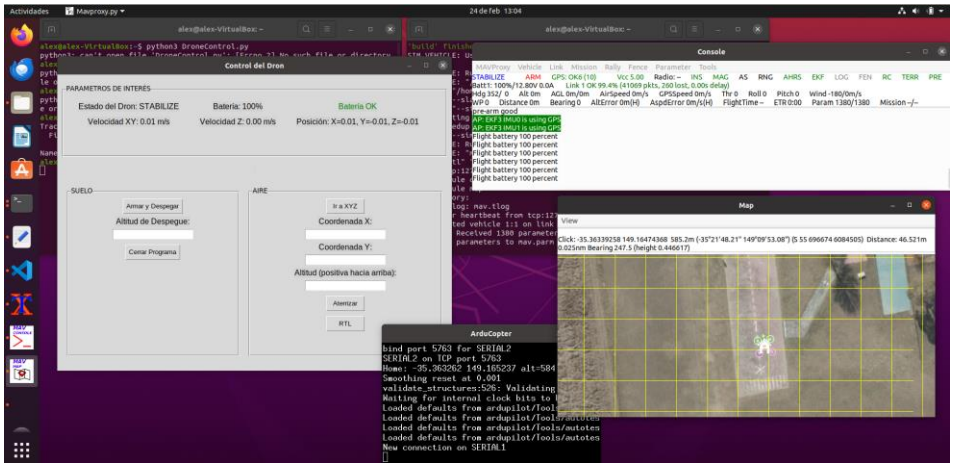


Ilustración 110: ControlDron y Simulación

Las ventanas creadas a partir de la simulación ya se han explicado anteriormente. Ahora vamos a usar la ventana de control.

Ilustración 111: Interfaz ControlDron.py

Antes de comenzar a mandar ordenes, en la ventana Console de la simulación es necesario ver el mensaje de *Flight battery 100 percent,* que aparecerá después del *pre-arm good.*

Una vez haya aparecido esto, el dron estará completamente operativo. Se comenzaría mandando la orden de despegue, especificándole una altura en el cuadro debajo del botón despegar. Al igual que las demás acciones, irá mostrando mensajes en el centro de la pantalla, en este caso serán armando y en despegue, después altitud objetivo alcanzada. Mientras tanto se observa como aumenta el parámetro Alt en la consola de MAVProxy.

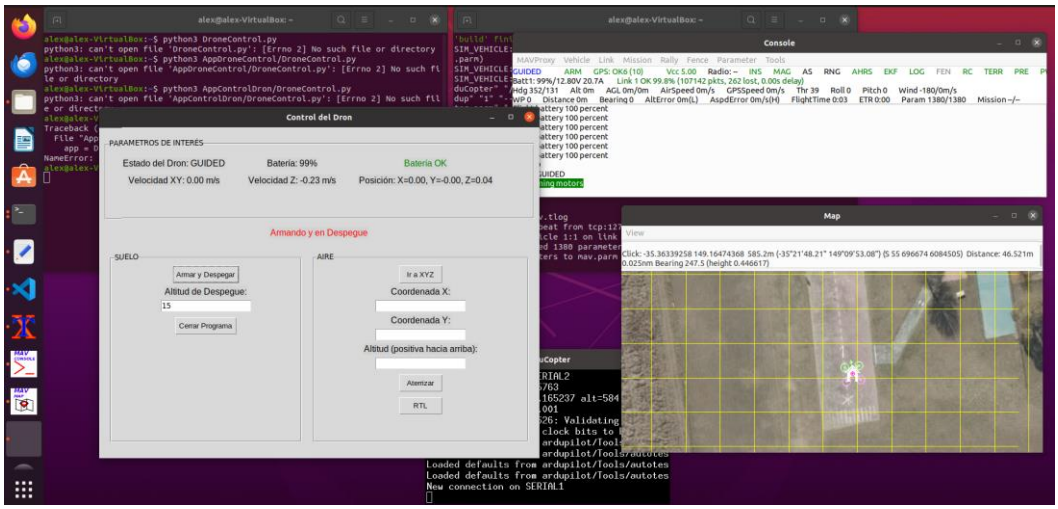


Ilustración 112: Visualización del despegue

La interfaz está hecha para que se pueda introducir las coordenadas de Z positivas, tomamos como si el eje Z de HOME fuera positivo hacia arriba. A continuación, se muestra lo que se vería con las diferentes funcionalidades de la interfaz.

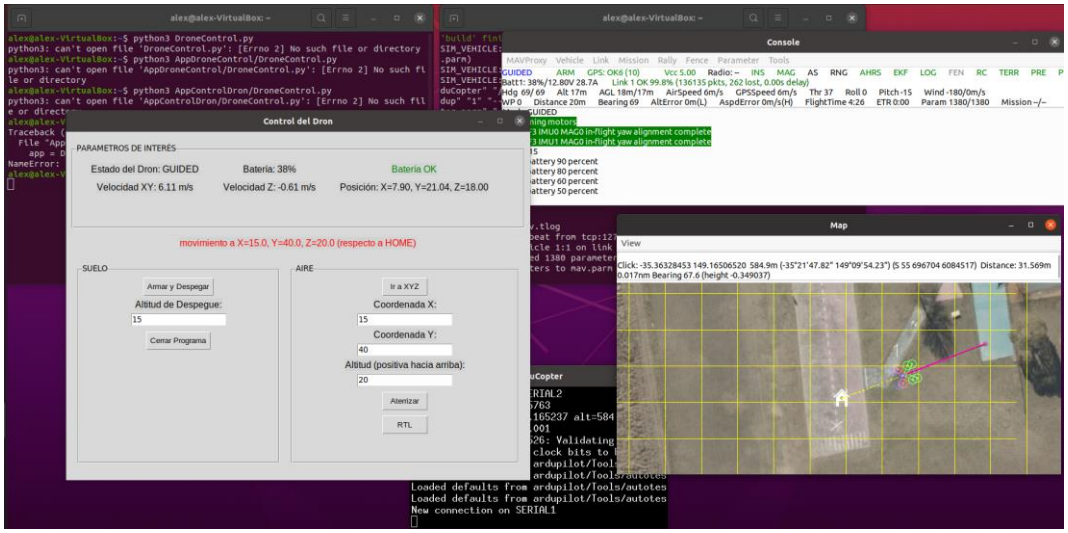


Ilustración 113: Visualización ir_a_XYZ

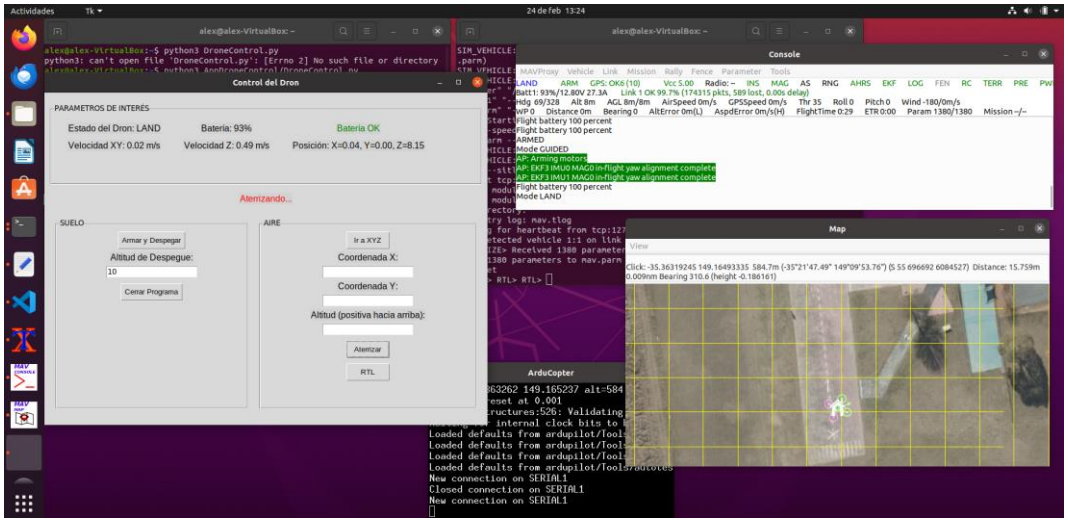


Ilustración 114: Visualización LAND

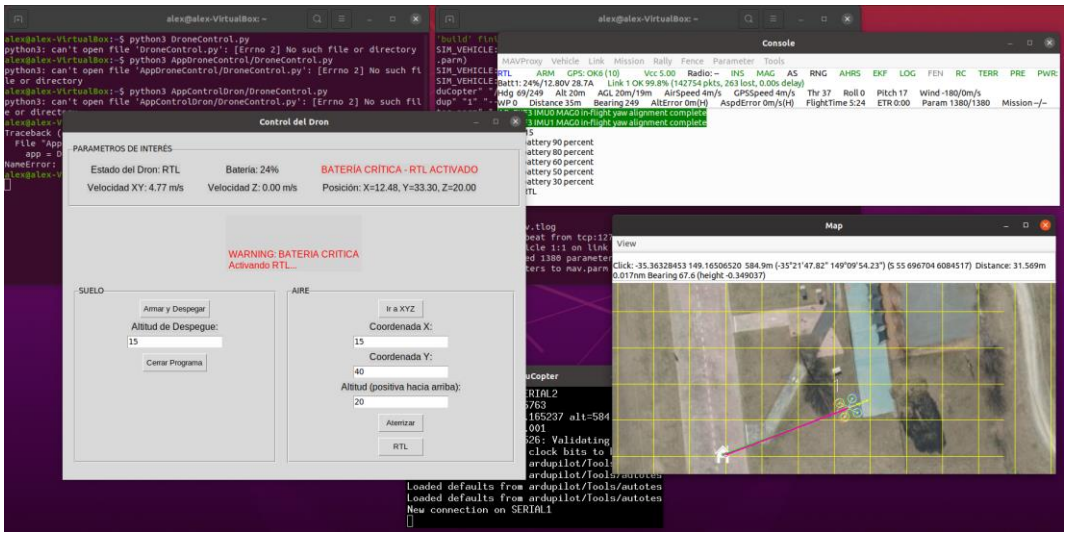


Ilustración 115: Visualización RTL por batería

5.2 Implementación en Dron real

En esta segunda fase, el objetivo principal es llevar el desarrollo realizado hasta el momento al dron físico para comprobar la compatibilidad real del software con el hardware. Sin embargo, se presentan diversas dificultades que condicionan la forma de realizar las pruebas:

Dificultades

1. **Falta de licencia de vuelo y pruebas en interior**
   Debido a la ausencia de la correspondiente licencia de vuelo, no es posible salir al exterior para probar todas las funcionalidades en condiciones reales, como vuelos de altura, desplazamientos horizontales amplios o maniobras automatizadas. Esta restricción de tipo legal fuerza a que las verificaciones iniciales se lleven a cabo en un entorno cerrado donde el espacio y las posibilidades de vuelo son muy limitados.

2. **Imposibilidad de utilizar modos basados en GPS**
   Al volar en interior, el GPS no adquiere señal suficiente o directamente no la recibe, imposibilitando la activación de modos de vuelo que requieran posicionamiento GNSS (como Guided, Loiter, PosHold o Auto). Esto limita cualquier funcionalidad que dependa de la localización por satélite.

3. **Limitación de desplazamientos y seguridad en interior**
   Intentar maniobras más complejas en un recinto cerrado implica un riesgo mayor de colisión y daños, tanto al dron como al entorno. Por ello, cualquier intento de elevación en interior se efectúa con precaución y manteniendo altitudes muy bajas, lo cual no permite apreciar con exactitud la estabilidad y rendimiento en vuelo libre. De igual forma, no es posible probar distancias ni comportamientos automáticos de regreso al punto de origen (RTL) o vuelos programados, dado que esos modos requieren del GPS.

Pruebas

A continuación, se describe el plan de pruebas que se llevará a cabo en el dron real durante la Fase 2. Debido a las limitaciones de vuelo en interior (sin GPS) y la falta de licencia de vuelo, nos centraremos en cuatro acciones básicas:

1. **Conexión**
2. **Armado**
3. **Desarmado**

4. Cambio de modo

Los objetivos por comprobar son:

- Establecer el enlace entre el ordenador y el dron (Pixhawk) a través del puerto serial o la radio de telemetría, de modo que el software sea capaz de acceder a los parámetros y estados del dron real.
- Comprobar que se puede enviar el comando de “armar” (arm) al dron y que éste pasa al estado “Armed” a través del software. No se realizará despegue real, solo se constatará que los motores comienzan a girar (aunque, por seguridad, las hélices deberían retirarse).
- Validar que es posible enviar el comando de “desarmar” (disarm) al dron remotamente. Esto es esencial al finalizar cualquier operación o en caso de emergencia.

Garantía de que el software será funcional en exteriores

Aun con estas limitaciones, el hecho de poder armar el dron, conmutar modos de vuelo que no requieran GPS y recibir telemetría es un indicador de que el núcleo de la API y la comunicación con el controlador de vuelo funcionan correctamente. Una vez superado este punto, se puede inferir que, al habilitar de nuevo los modos con GPS y realizar pruebas al aire libre (cuando sea posible contar con la licencia o un entorno de vuelo autorizado), el software responderá de forma muy similar, puesto que la estructura de comandos y la comunicación no cambian.

Código

```
1
2   from dronekit import connect, VehicleMode
3
4   # Puerto y velocidad de baudios pueden variar según la configuración real
5   # En Windows suele aparecer como COMx, en Linux como /dev/ttyUSB0 o /dev/ttyACM0
6
7   #vehicle = connect('com3', wait_ready=True, baud=57600)
8   # De esta forma, DroneKit solo esperará a cargar los parámetros y no requerirá actitudes, GPS y demás.
9   #vehicle = connect('COM3', wait_ready=['parameters'], baud=57600)
10  # De esta forma, DroneKit no esperará a cargar los parámetros y no requerirá actitudes, GPS y demás.
11  vehicle = connect('COM3', wait_ready=False, baud=57600)
12
13  print("Conectado al vehiculo")
14  print("Modo actual:", vehicle.mode.name)
15  print("Estado armado:", vehicle.armed)
16  print("bateria", vehicle.battery)
17
18
19  # Asumiendo que ya estamos conectados en 'vehicle'
20  # 1) Cambiar a un modo que permita armar sin GPS en interior (p.ej. STABILIZE)
21  vehicle.mode = VehicleMode("STABILIZE")
22  vehicle.flush()
23
24  # 2) Intentar armar
25  vehicle.armed = True
```

```python

# 3) Esperar hasta que se confirme el armado o tiempo máximo
import time
start_time = time.time()
while not vehicle.armed and time.time() - start_time < 10:
    time.sleep(1)

if vehicle.armed:
    print("¡Dron armado correctamente!")
else:
    print("Error: No se logró armar el dron")

time.sleep(5)  # Esperar un momento antes de desarmar
# 4) Desarmar
vehicle.armed = False

start_time = time.time()
while vehicle.armed and time.time() - start_time < 10:
    time.sleep(1)

if not vehicle.armed:
    print("Dron desarmado con éxito")
else:
    print("Error: No se logró desarmar el dron")

# 5) Cambiar a un modo seguro (p.ej. LOITER)

nuevo_modo = "ALT_HOLD"  # Ejemplo de modo que no exige GPS
vehicle.mode = VehicleMode(nuevo_modo)
vehicle.flush()

import time
time.sleep(2)  # Esperar un momento a que el FC procese el cambio

print("Modo actual:", vehicle.mode.name)

```

Ilustración 116: Scripts Pruebas (1)

Resultados de las Pruebas

Se ha realizado varias pruebas, pero en estas imágenes se ven dos casos, cada uno con un modo de vuelo inicial. El dron ha respondido correctamente a las órdenes dadas.

```
WARNING:autopilot:GCS Failsafe Cleared
Conectado al vehículo
Modo actual: STABILIZE
Estado armado: False
¡Dron armado correctamente!
Dron desarmado con éxito
Modo actual: ALT_HOLD
```

Ilustración 117: Prueba 1 scripts en mundo real

```
WARNING:autopilot:GCS Failsafe Cleared
Conectado al vehículo
Modo actual: BRAKE
Estado armado: False
¡Dron armado correctamente!
Dron desarmado con éxito
Modo actual: ALT_HOLD
```

Ilustración 118: Prueba 2 scripts en mundo real

- **Conexión exitosa** sin errores de timeout. El mensaje de "WARNING: autopilot: GSC Failsafe Cleared" se recibe debido a que el dron no puede obtener todos los parámetros debido a la falta de la señal del GPS. De ahí que la conexión se haya configurado con **wait_parameters=False.** Esto garantiza que el dron se reconoce en el software y se reciben datos de telemetría en tiempo real.
- **Armado y desarmado** validan el control sobre la seguridad básica y la posibilidad de movimiento del dron, ya que al conseguir encender los motores se debería de poder realizar movimientos.
- **Cambio de modo** demuestra la capacidad de enviar órdenes al FC y actualiza el estado en Mission Planner, lo que se reflejará igualmente en la interfaz personalizada.

Con esto se puede concluir que el control desarrollado en el apartado de simulación será completamente compatible con el dron físico.

5.3 Procesamiento de vídeo

El propósito general del programa es la **detección de personas con YOLOv8** en cada fotograma de la cámara y, si estima que una persona está "demasiado cerca" (porque ocupa gran parte de la imagen), activa un modo de freno en el dron (BRAKE) durante un corto intervalo. Este enfoque muestra cómo **integra** la visión por computador (YOLOv8, OpenCV) con la **lógica de control** del dron (DroneKit), permitiendo reacciones automáticas a los objetos detectados.

1. **Importaciones y configuración inicial**

```
1  from ultralytics import YOLO
2  import cv2
3  from dronekit import connect, VehicleMode
4  import time
5
6  # Conectar con el dron
7  vehicle = connect('/dev/ttyACM0', wait_ready=True, baud=57600)
```

Ilustración 119: Scripts vídeo (1)

   - Se importan las librerías necesarias:
     - **ultralytics.YOLO** para cargar y ejecutar el modelo YOLOv8.
     - **cv2 (OpenCV)** para capturar y procesar vídeo en tiempo real.
     - **dronekit** para conectarse y enviar comandos al dron.

- **time** para pausas y temporizaciones.

- A continuación, se establece la conexión con el dron mediante dronekit.connect(), especificando el puerto, velocidad de baudios y wait_ready=True para asegurarse de que el dron esté completamente inicializado antes de continuar.

2. **Carga del modelo YOLOv8**

```
9   # Cargar el modelo YOLO preentrenado
10  model = YOLO("yolov8n.pt")  # Usa tu modelo entrenado si tienes uno propio
```

Ilustración 120: Scripts vídeo (2)

- Se crea una instancia de la clase YOLO, indicando el archivo de pesos ("yolov8n.pt"). Este archivo corresponde a un modelo preentrenado en el conjunto de datos COCO.

- Si se dispone de un modelo propio entrenado para un conjunto de objetos específico, sería aquí donde se reemplaza la ruta del archivo.

3. **Captura de vídeo**

```
12  # Iniciar captura de video
13  cap = cv2.VideoCapture(0)  # Cambia a 1 o 2 si 0 no funciona
14
15  if not cap.isOpened():
16      print(" No se pudo abrir la cámara.")
17      exit()
18
19  # Definir umbral de tamaño para considerar una persona como "cercana"
20  FRAME_WIDTH = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
21  FRAME_HEIGHT = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

Ilustración 121: Scripts vídeo (3)

- A través de cv2.VideoCapture(0), se abre la cámara principal del equipo (usualmente la webcam integrada o la primera cámara detectada por el sistema). Si 0 no es la cámara deseada, se va cambiando el índice a 1, 2, … hasta encontrarla.

- Se verifica si la cámara se ha abierto correctamente; en caso contrario, se muestra un mensaje y se sale del programa.

- Se obtienen las dimensiones del vídeo (FRAME_WIDTH, FRAME_HEIGHT). Se usarán para calcular si una detección es “grande” en proporción a la imagen y, por tanto, está “cerca”.

4. **Función para “frenar” el dron**

```
23  # Función para detener el dron
24  def detener_dron():
25      print(" Persona demasiado cerca - Dron detenido")
26      vehicle.mode = VehicleMode("BRAKE")  # Cambia a modo de frenado
27      time.sleep(2)  # Mantiene el freno un momento
28      vehicle.mode = VehicleMode("GUIDED")  # Vuelve al modo controlado
29
```

Ilustración 122: Scripts vídeo (4)

- Se define detener_dron(), que imprime un aviso y luego cambia el modo de vuelo a "BRAKE". Este modo en ArduPilot hace que el dron se mantenga en el aire y detenga su movimiento de inmediato. No baja al suelo, sino que aplica frenado activo usando los motores para mantener su posición en el aire.
- Se deja ese modo activo durante 2 segundos (time.sleep(2)) y, pasado ese tiempo, se regresa a "GUIDED". De esta forma, el dron puede volver a ser controlado por la estación en tierra o por comandos automáticos.

5. Bucle principal de procesamiento

```
30  while True:
31      ret, frame = cap.read()
32      if not ret:
33          print("⚠ No se pudo capturar el frame.")
34          break
35
36      # Detectar objetos con YOLO
37      results = model(frame)
38      persona_cercana = False  # Bandera para detectar proximidad
```

Ilustración 123: Scripts vídeo (5)

- Se lee un frame de la cámara (cap.read()), comprobando que la lectura sea correcta (ret == True).
- El frame se pasa al modelo YOLO para detectar objetos (model(frame)). La variable results contendrá las coordenadas y la clase de cada objeto detectado.
- Se inicializa la bandera persona_cercana = False, que indicará si alguna detección cumple el criterio de “cercanía”.

6. Detección de personas y verificación de cercanía

```python
    for result in results:
        for box, cls in zip(result.boxes.xyxy, result.boxes.cls):
            if int(cls) == 0:  # Clase 0 en COCO es "person"
                x1, y1, x2, y2 = map(int, box[:4])
                width = x2 - x1
                height = y2 - y1

                # Determinar si la persona está cerca
                if width > FRAME_WIDTH * 0.4 or height > FRAME_HEIGHT * 0.6:
                    color = (0, 0, 255)  #  Rojo (persona cercana)
                    persona_cercana = True  # Activar el freno del dron
                else:
                    color = (0, 255, 0)  # Verde (persona lejos)

                # Dibujar el bounding box con el color correspondiente
                cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)

```

Ilustración 124: Scripts vídeo (6)

- Se recorren las detecciones dentro de results:
  - Si la clase (cls) corresponde a 0, que en COCO está asociada a “persona”.
  - Se extraen las coordenadas del bounding box (x1, y1, x2, y2) y se calcula su anchura y altura.
  - Se considera “persona cercana” si la caja ocupa más de cierto porcentaje del frame (por ejemplo, width > FRAME_WIDTH * 0.4 o height > FRAME_HEIGHT * 0.6).
    - De cumplirse, se dibuja un rectángulo **rojo** en la imagen y se marca la bandera persona_cercana = True.
    - Si no, se dibuja el rectángulo **verde**.

7. **Acción de frenado si hay persona cercana**

```python
    # Si una persona está cerca, detener el dron
    if persona_cercana:
        detener_dron()
    else:
        if vehicle.armed == False:
            vehicle.armed = True
```

Ilustración 125: Scripts vídeo (7)

- Tras revisar todas las detecciones, si persona_cercana está en True, se llama a detener_dron().
- Esto ordena inmediatamente al dron que frene y, unos segundos más tarde, recupera el modo de vuelo anterior.

8. **Visualización y salida**

```
61      # Mostrar el frame con detecciones
62      cv2.imshow("YOLOv8 - Detección de personas con freno automático", frame)
63
64      if cv2.waitKey(1) & 0xFF == ord('q'):  # Presiona 'q' para salir
65          break
66
67  cap.release()
68  cv2.destroyAllWindows()
69  vehicle.close()  # Cerrar conexión con el dron
```

Ilustración 126: Scripts vídeo (8)

- Se muestra el frame resultante en una ventana usando cv2.imshow(). El usuario puede comprobar en tiempo real los rectángulos dibujados alrededor de las personas y su color.
- Si se pulsa la tecla 'q', se interrumpe el bucle y se cierran los recursos:
  - cap.release() finaliza la captura de vídeo.
  - cv2.destroyAllWindows() cierra las ventanas de OpenCV.
  - vehicle.close() cierra la conexión con el dron para liberar el puerto y limpiar la sesión de MAVLink.

Resultados

Aunque es un código un poco rudimentario, en cuanto a la reactivación de las funciones del dron cuando la persona se haya alejado, el modo de vuelo cambia correctamente en cuanto una persona se acerca demasiado. No hay imágenes de resultados debido a que para el modo BRAKE es necesario el uso de GPS, y en muchos casos no se conseguía cambiar del todo a este modo de vuelo.

Por parte de la detección de Yolo, es un software bastante confiable. En el caso de que se necesite una mayor precisión, siempre se puede optar a modelos más grandes, que mejoran la precisión a costa de la velocidad de detección. En el caso de este TFG se ha usado un modelo pequeño para agilizar las pruebas.

CAPÍTULO 6: GESTIÓN DE TRABAJO Y COSTES

En este capítulo se presenta la planificación temporal de las tareas llevadas a cabo durante el desarrollo del proyecto (mediante un diagrama de Gantt) y el desglose de costes de los componentes empleados.

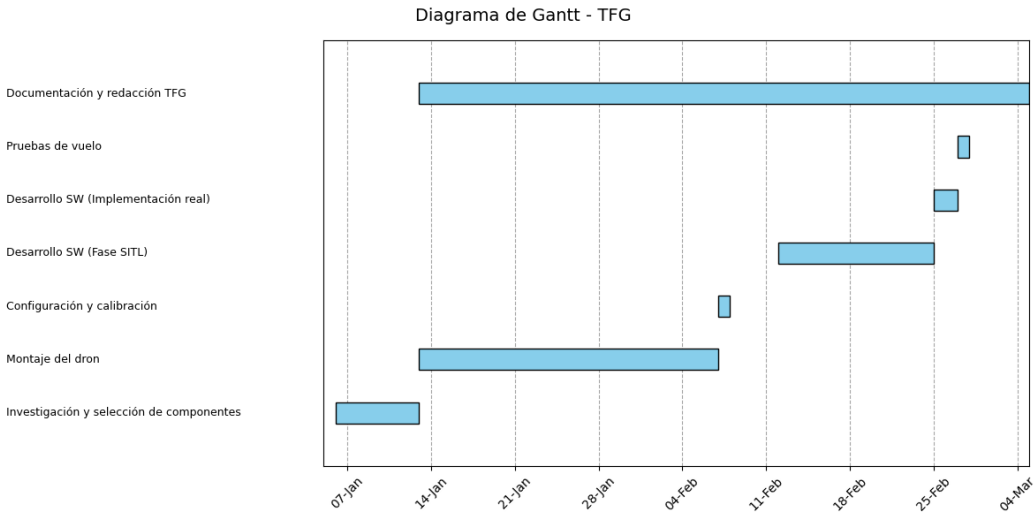


Ilustración 127: Diagrama de Gantt

6.1 Planificación Temporal

Para organizar las distintas fases del proyecto, se han definido las siguientes tareas principales:

1. **Investigación y selección de componentes**
    - Búsqueda de opciones de hardware, comparativas, elección final.
    - Duración: 1 semana (7 días), trabajando 6–7 horas al día.
2. **Montaje del dron**
    - Ensamblaje físico, cableado, instalación de componentes (chasis, motores, Pixhawk, cámara, etc.).
    - Del 13 de enero al 6 de febrero (solo de lunes a viernes), con una media de 4–5 horas/día.
3. **Configuración y calibración**
    - Mission Planner, emisora, ESC, GPS, modos de vuelo.
    - Duración total: 4 horas.
4. **Desarrollo de software**
    - Del 12 de febrero al 28 de febrero, con 6–8 horas/día, desglosado en:

- Fase de simulación (SITL): 12 al 24 de febrero
- Fase de implementación real (control, telemetría, recepción de vídeo): 25 y 26 de febrero

5. **Pruebas finales**
    - test de estabilidad, seguridad, vídeo en tiempo real.
    - Duración total estimada: de 3 a 6 horas en total.
6. **Documentación y redacción del TFG**
    - elaboración de memoria, anexos, bibliografía, etc.
    - Del 13 de enero al 4 de marzo (sin especificar horas exactas cada día; algunos días 4 h y otros hasta 12 h, en función de la disponibilidad).

6.2 Costes y Presupuesto

El proyecto ha requerido la adquisición de un kit de dron y diversos componentes adicionales para dotarlo de transmisión de vídeo y telemetría, así como otros accesorios. A continuación, se muestra el desglose de los costes más relevantes.

Tabla 20: Desglose de precios de componentes

Componente	Cantidad	Precio (€)	Subtotal (€)
Hawk's Work F450 DroneKit (kit completo)	1	469,99	469,99
Batería LiPo (4200 mAh)	1	36,00	36,00
Cámara RunCam Nano	1	41,15	41,15
TBS Unify Pro + Antena SMA 5.8GHz	1	26,23	26,23
Antenas (adicionales)	1	8,26	8,26
Eachine ROTG2 5.8Ghz	1	39,00	39,00
Adaptador XT60-KST	1	8,82	8,82
HolyBro SiK Telemetry Radio	1	71,19	71,19
Gestión (imprevistos, envíos, etc.)	-	10,00	10,00
Total Aproximado	-	-	**710,64**
Total + 21% IVA			**859,875**

CAPÍTULO 7: CONCLUSIÓN Y FUTUROS DESARROLLOS

La realización de este proyecto ha permitido demostrar la viabilidad de integrar un sistema de control externo en un dron y una cámara FPV para transmisión y procesado de vídeo desde un ordenador, utilizando para ello plataformas y librerías de software de código abierto (ArduPilot, MAVLink, DroneKit, etc.).

La experiencia de montar este dron ha sido instructiva en cuanto a la ampliación de conocimiento sobre APIs y comunicaciones, además de una introducción al mundo de los drones, aprendiendo un poco más sobre sus componentes e historia. Durante el montaje se presentaron algunas dificultades, lo que obligó a improvisar soluciones rápidamente.

A pesar de lo avanzado en el software, se ha echado bastante en falta la posibilidad de poder realizar vuelos reales en exterior, para validar por completo el funcionamiento, rendimiento del software desarrollado, y la estabilidad del dron en vuelos autónomos. Aun así, las pruebas en interior han demostrado la sensibilidad del dron a cualquier imprecisión de calibración y a la falta de señal GPS. En conclusión, se podría decir que se ha adquirido un conocimiento muy práctico sobre mecánica, electrónica y programación, lo que me deja satisfecho con el resultado alcanzado.

No obstante, además de lo más importante que sería la adquisición de una licencia de vuelo, existen numerosas líneas de mejora y ampliación que podrían abordarse en trabajos futuros para dotar al dron de mayores capacidades, optimizar su desempeño y ampliar su ámbito de aplicación. A continuación, se señalan algunas de ellas:

Añadir Sensores y Evitar Obstáculos

- Incluir sensores de ultrasonido, LiDAR o cámaras especiales para detectar y esquivar obstáculos.
- Usar cámaras térmicas o infrarrojas para búsquedas en condiciones de poca luz o inspecciones nocturnas.

Procesamiento de Vídeo Inteligente

- Implementar algoritmos de visión por computadora para reconocer objetos y seguirlos sin intervención humana.
- Añadir un miniordenador (por ejemplo, Raspberry Pi) para procesar el vídeo a bordo y disminuir la latencia.

Interfaz de Usuario

- Integrar la parte del software del sistema de vídeocon el software de control.
- Crear una aplicación web o móvil para controlar el dron sin instalar software extra.
- Añadir elementos de realidad aumentada (AR) o realidad virtual (VR) para pilotar con más información.

Chasis Más Amplio, Baterías y Motores de Mayor Capacidad

- Investigar un armazón de mayor tamaño que permita acomodar todos los componentes de manera ordenada.
- Incorporar baterías con más capacidad y motores más potentes para soportar el peso adicional y prolongar el tiempo de vuelo.

Gracias a estas posibles mejoras, el dron podría llevar a cabo misiones más complejas y exigentes, además de ofrecer un sistema más estable y versátil.

CAPÍTULO 8: REFERENCIAS

Cargando Firmware. (2024). Obtenido de ardupilot.org/planner/: https://ardupilot.org/planner/docs/common-loading-firmware-onto-pixhawk.html#common-loading-firmware-onto-pixhawk

Conecte Mission Planner al piloto automático. (2022). Obtenido de https://ardupilot.org/planner/: https://ardupilot.org/planner/docs/common-connect-mission-planner-autopilot.html#common-connect-mission-planner-autopilot

Descripción general de la pantalla de datos de vuelo. (2022). Obtenido de ardupilot.org/planner/: https://ardupilot.org/planner/docs/mission-planner-ground-control-station.html#mission-planner-ground-control-station

Planificador de misión PLAN de vuelo. (2022). Obtenido de ardupilot.org/planner/: https://ardupilot.org/planner/docs/mission-planner-flight-plan.html#mission-planner-flight-plan

Planificador de misiones CONFIGURACIÓN inicial. (2024). Obtenido de ardupilot.org/planner/: https://ardupilot.org/planner/docs/mission-planner-initial-setup.html#mission-planner-initial-setup