ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
Máster en Ingeniería Industrial (MII)

PROYECTO FIN DE MÁSTER

DESARROLLO DE SISTEMAS DE NAVEGACIÓN AUTÓNOMA EN INTERIORES PARA UN UAV

Autor: Javier García Aguilar
Director: Prof. Dr. Juan Luis Zamora Macho

MADRID
julio 2018

This dissertation was typeset with LaTeX and compiled in TeXstudio.

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

"Desarrollo de sistemas de navegación autónoma

en interiores para un UAV"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017 / 2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: GARCÍA AGUILAR, JAVIER Fecha: 18/ 07/ 2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Dr. Juan Luis Zamora Macho Fecha: 18/ 07/ 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Javier García Aguilar

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:

Desarrollo de Sistemas de Navegación Autónoma para un UAV,

que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 18 de Julio de 2018

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Abstract

This project aims to develop and implement a Matlab/Simulink-programmed autonomous indoors flying system. It uses both inertial sensors and a Motion Capture System (MCS) developed by Optitrack. The controller used is a Raspberry Pi Zero W and an Extended Kalman Filter based in an inertial model has been used to fuse all the measurements of the sensors. The PXFmini module from Erle Robotics has been used in order to provide the Raspberry with the Inertial Measurement Unit (IMU) and a servo driver to comunicate with the electronic speed controllers.

Introduction

In the last years, drones have experienced a huge growth in popularity. But in practice, their use is limited to outdoors. That is because the limitation in space in indoors environments makes it difficult to non-expert pilots to fly the drone. Moreover, not being able to access GPS makes it difficult to implement an autonomous control unless there are several sensors in the actual drone or it flies in a space dedicated to that, with a Motion Capture System (MCS) in it.

Using a MCS like Motive-Optitrack or VICON allows the possibility to non-expert pilots to implement autonomous drones in indoors environments. This is ideal both for investigation and education purposes.

Objectives

The mean objective of this project is to implement a, autonomous indoors navigation system of a drone using a MCS. With that in mind, the following milestones:

- Implementation of a versatile tool for drone programming with *Matlab/Simulink*.
- Elaboration of an intuitive and easy to use drone control structure.
- Construction and programming of a drone with a *Raspberry Pi Zero W*.
- Executing autonomous flights in a MCS located space.

Solution

In Figure 1 it is shown a simplified diagram of the project. Basically, the drone may function in both manually and autonomous modes. From the RC emitter the pilot may sen the most critical orders to the drone, such as enabling the autonomous flight or taking manual control of it again. There is an Extended Kalman Filter (EKF) to estimate the position and orientation of the drone, in order to be able to incorporate redundant sensors. Lastly, a groud control station (GCS) is used for monitoring and sending to the drone the MCS data.

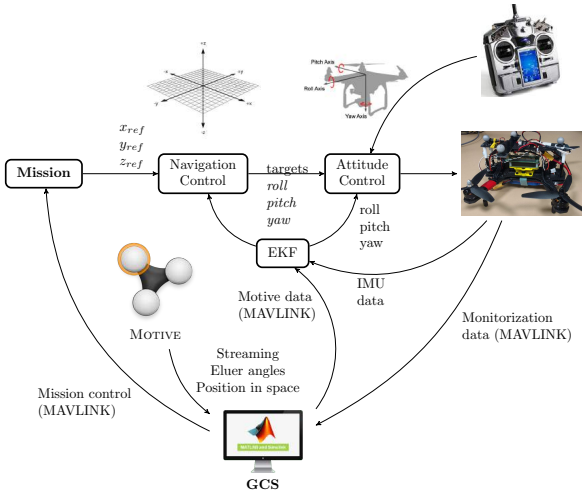


Figure 1. Project schema

In the solution implemented, both simulation and implementation environments are integrated in the same Simulink diagram. This allows a faster design and implementation of controls.

As the drone processor, a *Raspberry Pi Zero W* has been used. This controller allows to implement different sampling times in the model. Using the *Simulink Support Package for Raspberry Pi* has simplified the implementation of the communication protocols. With the Raspberry Pi, a PXFmini module by Erle Robotics has been used. This module comes with several sensors which are critical for a drone, as well as a servo driver to communicate with the motors.

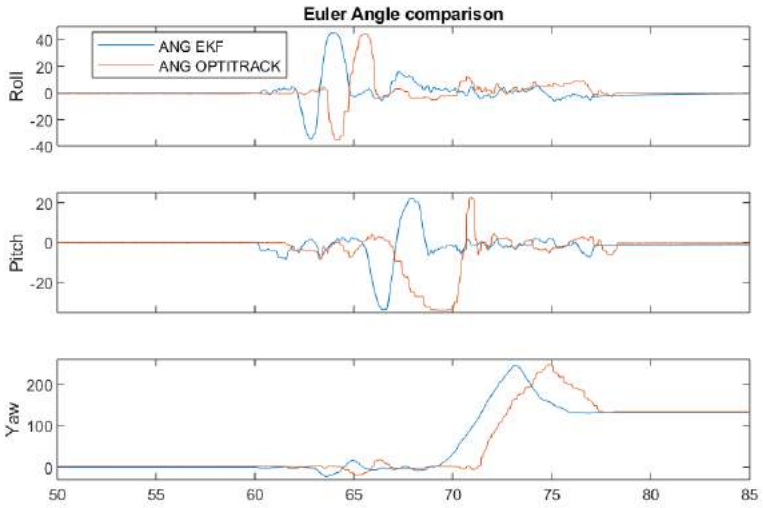
Figure 2. Drone Structure

For MAVLink communication, three protocols have been implemented: TCP/IP, UDP and Bluetooth. And for communication the RC receiver, the protocols studied were PWM, PPM, SBUS and IBUS.

Results

The stabilization control designed works perfectly, as there have been successful manual flight tests.

In Figure 3 there can be observed a comparative between the Euler angles provided by the IMU measurements with the EKF and the MCS data. As it can be seen, they are nearly the same, but with a little delay in the MCS.



[h]

Figure 3. Euler angles. IMU vs MCS

A black box has been implemented for depuration purposes. This has allowed to analyze data after flights.

After analizing the pros and cons of the communication protocols, the conclusion wass that the most stable was the TCP/IP protocol. But it was too slow for closing the loop with the MCS data, so the final communication implemented between GCS and the drone was Bluetooth.

Conclusions

Programming the Raspberry Pi with Simulink has helped to the communication management, but it makes it more difficult to implement other characteristics such as interrupts.

On the other hand, design and implementation of control is made easily and fast, it is quite fast to change sensor configuration and to implement new flight modes the only way to make changes is the state machine

Resumen

Este proyecto trata de desarrollar e implementar un sistema de control de vuelo autónomo en interiores para un cuadricóptero dentro de un entorno de programación específico de UAVs desarrollado con Matlab/Simulink. Para ello, se he hecho uso tanto de sensores inerciales a bordo del UAV como del sistema de captura de movimiento de Optitrack. El control será implantado en una Raspberry Pi Zero W junto con el módulo PXFmini de Erle Robotics. Con el objetivo de aúnar todos los sensores, se ha implantado un filtro extendido de Kalman basado en un modelo inercial.

Introducción

En los últimos años, los drones han experimentado un crecimiento en popularidad enorme. Sin embargo, su uso está prácticamente limitado a exteriores. Ésto es así debido a que el espacio reducido en interiores dificulta su manejo a pilotos no expertos, además que, al no contar con un sistema de localización tan accesible como el GPS, es complicado de implementar un control autónomo sin tener que recurrir a numerosos sensores integrados en el dron.

Sin embargo, si se emplea un sistema de localización en un recinto preparado para ello, como podrían ser los sistemas de VICON y Optitrack, se abre la posibilidad de trabajar cómodamente con los drones sin necesidad de ser piloto experto. Ésto es útil tanto para labores de investigación como lectivas.

Objetivos

El objetivo de este proyecto es implementar un sistema de navegación autónoma en interiores en un recinto equipado con el MCS (Motion Control System) de Optitrack. Para ello, se persiguen los siguientes hitos:

- Implementación de una herramienta versátil para la programación de drones con *Matlab/Simulink*.
- Elaboración de una estructura de control propia que sea intuitiva y fácil de implementar y modificar.
- Construcción y programación de un dron basado en una *Raspberry Pi Zero W*.
- Ejecución de vuelos autónomos en un espacio geolocalizado por un el MCS de Optitrack.

Solución

En la Figura 1 se muestra un diagrama simplificado de la estructura general del proyecto. Resumidamente, el dron puede funcionar tanto en vuelo manual como en vuelo autónomo. Dependiendo del modo de funcionamiento se activa sólo el control de estabilización o éste y el de navegación. Desde la emisora de radio se le mandan las órdenes más críticas, como el cambio de modo de vuelo y las referencias en vuelo manual. El EKF recibe datos tanto de la IMU como del MCS. Por último, la GCS se encarga de monitorizar señales y enviar al dron los datos obtenidos por el MCS.

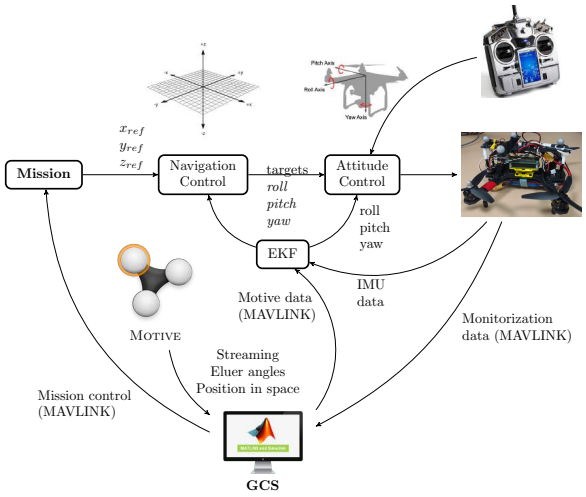


Figura 1. Esquema conceptual

La solución propuesta en este proyecto pasa por integrar las herramientas de simulación e implementación en un solo diagrama de *Simulink*. Con ello se consigue mayor rapidez en el diseño e implementación de controles, al no tener que estar cambiando constantemente de herramienta.

Como controladora de vuelo se ha empleado una *Raspberry Pi Zero W*, lo que ha permitido ejecutar partes del código a distintos tiempos de muestreo. Además, al emplear el paquete de soporte de *Simulink* para la *Raspberry Pi*, tras configurar ésta la gestión de los periféricos se facilitó enormemente. Junto con ella, se ha utilizado el módulo PXFmini, de Erle-Robotics, especialmente diseñado para trabajar con la Raspberry Pi Zero.

Figura 2. Estructura del dron

Para las comunicaciones MAVLink entre la GCS y el dron, se han implementado tres protocolos: TCP/IP, UDP y Bluetooth/Serial. Para comunicar la emisora con el dron, se han estudiado y analizado los protocolos PWM, PPM, SBUS e IBUS.

Resultados

El control de estabilización diseñado funciona correctamente, puesto que se han efectuado vuelos en modo manual.

En la Figura 3 Puede observarse una comparativa entre los ángulos de Euler obtenidos del EKF a partir de los datos de la IMU y los captados por las cámaras.

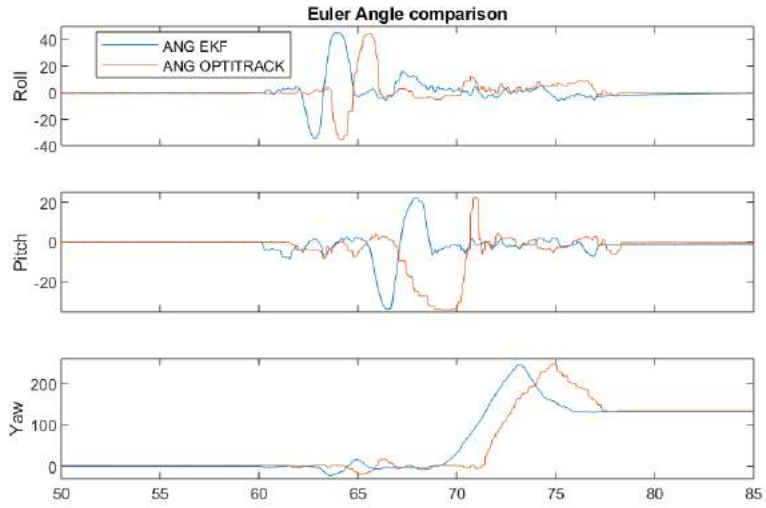


Figura 3. Ángulos de Euler. IMU vs MCS

Se ha implementado una caja negra para labores de depuración , lo que ha permitido analizar varias numerosas tras los vuelos y mejorar el control del dron. Además, mediante MAVLink pueden analizarse alguna variable directamente durante las pruebas de vuelo.

Tras analizar los pros y contras de los protocolos de comunicación, se ha concluido que el más estable para labores de monitorización es el TCP/IP, pero éste induce un retardo significativo en las comunicaciones. Ésto es inaceptable a la hora de cerrar el lazo con las medidas de las cámaras. Por ello, se ha empleado comunicación vía Bluetooth para comunicar la estación base y el dron.

Conclusiones

El hecho de programar la Raspberry Pi con Simulink sí que ayuda enormemente en la gestión de las comunicaciones, pero, por otro lado, complica el proceso de añadir nuevas características. Por ejemplo, no se encontró la manera de implementar interrupciones para emplear el protocolo PPM y liberar así la UART para las comunicaciones con MAVLink.

Por otro lado , el diseño e implementación de los controles se efectúa de manera muy sencilla y rápida, es muy sencillo conmutar entre los sensores a utilizar e implementar nuevos modos de vuelo es fácil e intuitivo, al tener únicamente que modificar la máquina de estados.

Índice general

Índice de figuras

Índice de tablas

Siglas

BLDC motores de continua sin escobillas

EKF Filtro Extendido de Kalman

ESC Controlador Electrónico de Velocidad

GCS estación base

IMU Unidad de Medida Inercial

IR Infrarrojos

LEC Laboratorio de Control

LEP Laboratorio de Electrónica de Potencia

PID Proporcional-Integral-Derivativo

PWM modulación por ancho de pulso

US Ultrasonidos

XYZ ejes de coordenadas

PARTE I

MEMORIA DESCRIPTIVA

1 Introducción

En este capítulo se hace una introducción al proyecto. En la Sección 1.1 y la Sección 1.2 se expone la motivación que ha llevado a realizar este proyecto y los antecedentes que ha habido en la universidad. En la Sección 1.3 se especifican los objetivos del proyecto. Por último, en la Sección 1.5 se presentan los recursos de hardware y software que se han empleado.

1.1. Motivación

La reciente masificación de los UAVs (Unmaned Aerial Vehicle) ha sido posible gracias a la inmensa evolución que han experimentado los microprocesadores en los últimos años. La aparición de controladores cada vez más potentes y rápidos han permitido que se consigan implementar los algoritmos de control necesarios en un espacio lo suficientemente reducido como para tener drones de tamaños propicios para vuelo en interiores. De hecho, cada vez más, están surgiendo microcontroladores diseñados especialmente para su uso en UAVs, como puede ser la tarjeta *Openpilot REVO*. Estos microcontroladores se caracterizan por que llevan un sensor de medida inercial (IMU), acelerómetros y giroscopios integrados en la propia tarjeta.

Una de las características más interesantes de los UAVs es la posibilidad de efectuar vuelos de forma completamente autónoma. Esto presenta una serie de ventajas e inconvenientes, donde el inconveniente principal es el posicionamiento del dron. En espacios exteriores, lo más normal es emplear GPS para ello, pero en interiores no se tiene esa ventaja, por lo que se ha de recurrir a otro tipo de sensores, ya sean externos al dron (cámaras de localización, en un espacio dedicado al vuelo de los drones) o incorporados en éste (sensores de flujo óptico, de US, IR, LIDAR, sonar...). Establecer un sistema de posicionamiento fiable habilita la incorporación del dron en ambientes industriales, como podría ser efectuar un inventariado automático en grandes almacenes.

La motivación principal de este proyecto es facilitar un sistema fiable de navegación de cuadricópteros a futuros proyectos. Con ello se pretende que dichos proyectos no hayan de enfocarse en el control de vuelo como tal, si no en darle utilidad a la aeronave.

1.2. Antecedentes

Este proyecto se sustenta en otros realizados en años anteriores en ICAI, debido a que ha habido una gran variedad de proyectos dedicados al vuelo de cuadricópteros en esta escuela, tanto orientados a la navegación en exteriores [SQ17] como en interiores [Men17], [Gon16]. Dichos proyectos estaban basados en la controladora de vuelo *Openpilot REVO*, que emplea un *STM34F4*. Además, el control implementado era una traducción a *Matlab* del *ArduPilot*, un programa de código abierto empleado en la programación de drones no comerciales.

El *STM32F4*, sin embargo, no tiene potencia suficiente para efectuar todos los cálculos que son necesarios, debido a la gran carga de computación que requiere el Filtro Extendido de Kalman (EKF). Por ello, se efectuaba un control con un periodo de muestreo de $20\,\mathrm{ms}$, que está al límite a la hora de implementar un control robusto. Además, es un microcontrolador mono tarea, por lo que no permite ejecutar partes del código a distintos periodos de muestreo.

Este proyecto trata de solventar los problemas derivados del uso de la tarjeta *OpenPilot Revo* mediante el uso de una *Raspberry Pi Zero W*. La *Raspberry* presenta una potencia de cálculo bastante superior al *STM32F4* y es capaz de gestionar tareas a distintos tiempos de muestreo, por lo que se ha considerado la alternativa perfecta para sustituir a la *OpenPilot Revo*. Además, aprovechando el cambio de hardware, se va a cambiar la estructura de control por otra má intuitiva y robusta.

1.3. Objetivos

El objetivo principal de este proyecto es conseguir vuelos autónomos estables de un cuadricóptero en un espacio interior.

Para ello, se persigue la consecución de los siguientes objetivos:

1. Implementación de una herramienta versátil para la programación de drones con *Matlab/Simulink*.
2. Desarrollo de una estructura de control propia con el fin de que sea más intuitiva y permita rediseñar los controles de forma sencilla.
3. Implementar la estructura de control en un cuadricóptero basado en una *Raspberry Pi Zero W*. Se ha decidido emplear esta controladora por su mayor potencia frente al *STM32*, además de que, al disponer de un sistema operativo basado en *Linux*, permite ejecutar partes distintas del código a distintos tiempos de muestreo.
4. Efectuar vuelos autónomos en un recinto geolocalizado mediante cámaras para verificar el correcto funcionamiento del sistema.

1.4. Metodología

En la Figura 1.1 se muestra un esquema conceptual del sistema. Todos los pasos llevado a cabo durante el proyecto han servido para enfocar o solventar alguno de los conceptos representados en ella.

Primero, se diseñó y construyó un entorno de simulación para efectuar el diseño y validación de los controles y el EKF. Para ello se cuenta con un modelo matemático del dron y una simulación de las medidas de los sensores.

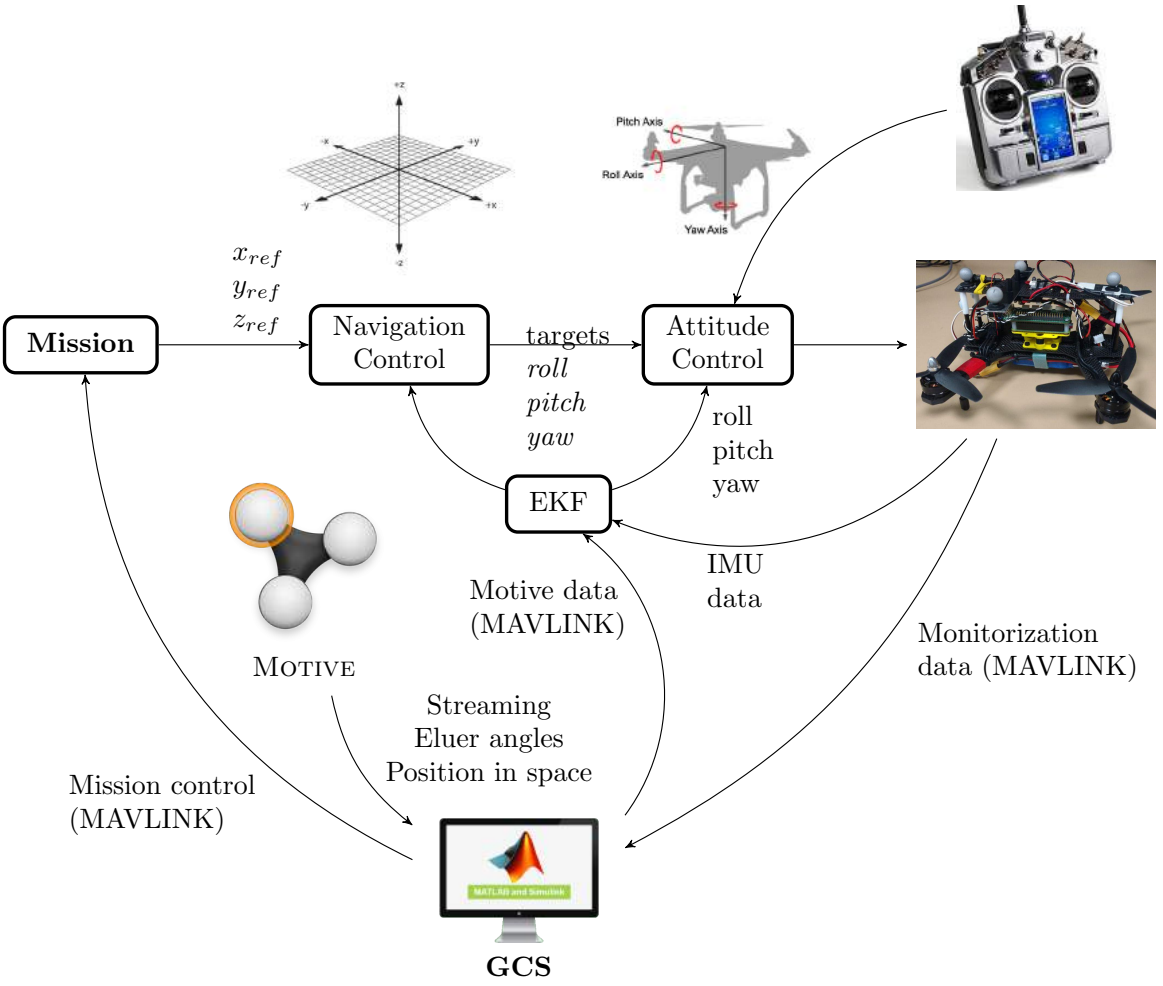


Figura 1.1. Estructura general del proyecto.

De forma simultánea, se configuró la *Raspberry* y se establecieron y comprobaron las comunicaciones con la estación base (GCS) y la emisora. También se comprobó el correcto funcionamiento de la Unidad de Medida Inercial (IMU) y se construyó otro sistema de *Simulink* para emplearlo como GCS.

A continuación, se procedió a montar el dron e implementar el sistema de control en la *Raspberry*. Para ello se tuvo que caracterizar la fuerza de empuje de los motores en un banco de ensayos. El modelo de *Simulink* empleado para programar la *Raspberry* fue el mismo que se empleó para la simulación, comentando simplemente la parte de código exclusivamente dedicada a simular el proceso. Ésto permite conmutar entre el entorno de simulación y programación de forma sencilla, además de agilizar el proceso de diseño y prueba de nuevos controles.

Una vez construido el cuadricóptero, comprobado las comunicaciones y el correcto funcionamiento de la máquina de estados y el EKF, se procedió a efectuar vuelos en manual y depurar el control de estabilización. Después de ello, se configuró *Motive* para obtener obtener los datos de las cámaras externas y enviarlos a la GCS.

Por último, se procedió a implementar y depurar los controles de navegación diseñados previamente.

1.5. Recursos

1.5.1. Hardware

Raspberry Pi Zero W

Como controladora de vuelo se empleó una *Raspberry Pi zero w*, debido a su mayor potencia de cálculo frente a otras alternativas, y al hecho de que permite ejecutar partes de código a distintos tiempos de muestreo.

Módulo PXFmini

Junto con la *Raspberry*, se empleó el módulo *PXFmini* de *Erle Robotics*. Éste módulo incorpora una IMU y un controlador de servos, cosa que facilita enormemente la implantación del sistema de control al tener ya las conexiones de los sensores y actuadores.

Baterías, regulador y distribuidor de potencia

Para alimentar al cuadricóptero se emplearon baterías LiPo de tres celdas con una capacidad de $1000mAh$. Esta batería alimenta simultáneamente a los Controlador Electrónico de Velocidad (ESC), a través de un distribuidor de potencia; y a la *Raspberry Pi*, a través de un regulador de tensión.

Estructura de un cuadricóptero

En este proyecto se ha empleado una estructura de un mini-dron de $250\,\mathrm{mm}$[1].

Emisora y receptor

Para controlar el cuadricóptero, se empleó una *Turnigy i10* enlazada a un receptor *FS-A8S*. Se ha optado por este par debido a que el receptor emplea el protocolo IBUS para enviar la información, lo que permite comunicarlo fácilmente con la *Raspberry*.

Cámaras de Optitrack

Con el objetivo de geolocalizar el cuadricóptero durante el vuelo, se ha instalado un sistema de ocho cámaras *Flex 13* de *Optitrack*.

Banco de ensayos y servo-tester

Se empleó un banco de ensayos y un servo-tester para caracterizar las curvas de empuje que proporciona el conjunto motor-esc-hélice.

1.5.2. Software

Matlab/Simulink

Se empleó el software *Matlab/Simulink* para programar el cuadricóptero, diseñar los controles y ejecutar la GCS. Desde la versión *R2018a* existe soporte para la *Pi zero* en *Simulink*, por lo que se ha empleado esta versión. Ésto se traduce en una gestión bastante sencilla de las comunicaciones, permitiendo enfocar la mayoría de los esfuerzos en la gestión de los datos y el desarrollo del control.

[1] Esta medida corresponde a la distancia entre dos motores opuestos

Putty

Se utilizó esta aplicación como cliente de *SSH* para conectarse remotamente a la *Raspberry* y poderla configurar sin necesidad de conectar una pantalla o un teclado externo.

Motive

Esta es la aplicación de *Optitrack*. Se encarga de recoger y analizar la información de las cámaras *Flex 13*, proporcionando así una medida de la posición y orientación del cuadricóptero en vuelo.

Blheli Suite

Se empleó esta aplicación para configurar los ESC. Al cargar el firmware de *Blheli*, los ESC detectan automáticamente si les llega un protocolo *PWM* estándar o un *OneShot*.

SolidWorks

Se empleó la aplicación de diseño 3D *SolidWorks* para adaptar un diseño de amortiguador mecánico a la *Raspberry Pi Zero*.

2

Estado del Arte

Uno de los principales problemas en el uso de drones autónomos en interiores es el relacionado con el posicionamiento del mismo. Al no poder emplear GPS, se necesita recurrir a otras tecnologías para controlar la trayectoria del dron.

Algunas de las soluciones que se han empleado implican la utilización de sensores adicionales, tanto integrados en el dron, como pueden ser sensores de flujo óptico; o externos al dron, mediante uso de cámaras externas. En éste capítulo se van a analizar las soluciones que se han empleado hasta la fecha, además de alguna aplicación que se les ha dado a los UAV en interiores.

2.1. Motion Capture System (MCS)

Los sistemas de captura en movimiento (MCS por sus siglas en inglés) basan su funcionamiento en un conjunto de cámaras que captan elementos característicos de un sólido rígido. Normalmente se emplean cámaras y reflectores de IR para ser capaz de obtener la posición y la orientación del dron en el espacio de trabajo. En la Figura 2.1 se muestra un diagrama de funcionamiento de un MCS.

Las cámaras de IR envían vectores dinámicos a un ordenador. Éste emplea la información para detectar y posicionar los marcadores mediante tiangulación, identificar los grupos que corresponden a sólidos rígidos distintos (previamente definidos) y calcular las posiciones y orientaciones de dichos sólidos.

Los sistemas de captura de movimiento más empleados son los de *Optitrack* y *VICON*. Estos sistemas se emplean tanto en capturas de cuerpos para animación 3D como de robots, tanto terrestres como aéreos [YCF+17]. Una de la principales ventajas que presenta este tipo de localización es que no requiere instalar un sensores muy voluminosos en el dron, basta con unos pequeños reflectores. Por ello, se puede emplear de forma bastante simple no solo en drones construidos de cero, si no también en drones comerciales, como puede ser el *Parrot AR Drone* [HSA+15], que, además, puede reprogramarse con ayuda de *Matlab/Simulink* gracias al paquete de soporte proporcionado por *Parrot*.

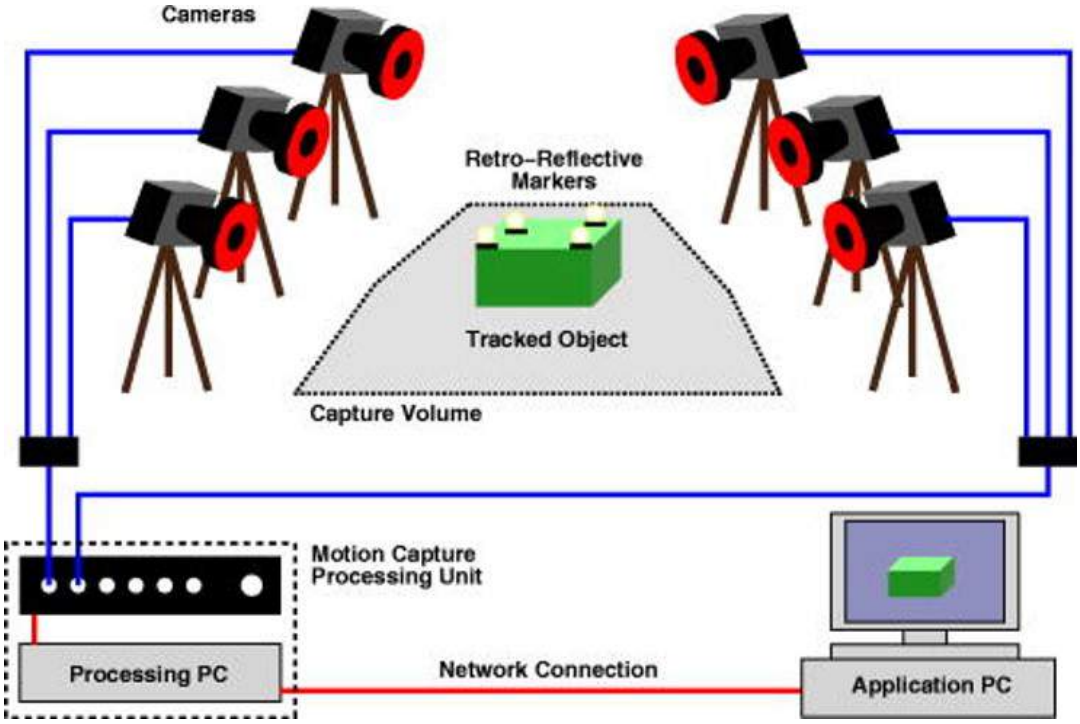


Figura 2.1. Diagrama de funcionamiento de un MCS.

Otra ventaja de estos sistemas es que pueden captar múltiples drones de forma simultánea. Esto es especialmente útil para coordinar varios drones y que se posicionen entre ellos. Este es el caso de *Crazyswarm* [PHSA17], un sistema comercial basado en VICON para posicionamiento de diversos drones de pequeño tamaño.

2.2. Ultra Wide Band (UWB)

La localización por *UWB* se basa en la emisión de señales de radio de amplia longitud de onda. La ventaja principal de éste método es la cantidad de datos que es capaz de comunicar en comparación con la energía que consume. Este sistema sólo es efectivo en cortas distancias, como puede ser en interiores.

Para posicionar un sólido rígido mediante esta tecnología, son necesarias varias balizas fijas situadas en el entorno de trabajo, al igual que una baliza adicional en el sólido. Un aspecto negativo de este sistema es que no identifica la orientación del sólido. Por ello, no son efectivos a la hora de corregir el ángulo de guiñada de los drones.

La empresa Bitcraze implementa dos modos de funcionamiento para su sistema *Loco Positioning*, el TWR (Two Way Ranging) y TDoA (Time Difference of Arrival).

En el TDoA, las balizas fijas emiten paquetes de sincronización de forma continua, por lo que una etiqueta receptora puede triangular su posición con la diferencia de tiempos entre las señales de las distintas balizas [TW17]. Al tratarse de un sistema unidireccional (las etiquetas receptoras no emiten información), el añadir distintos receptores no disminuye la efectividad del sistema. Debido a que la localización se efectúa mediante triangulación, el aumentar el número de balizas mejora sustancialmente la precisión global del sistema.

Por otro lado, en modo TWR la etiqueta móvil se comunica de forma secuencial y cíclica con cada una de las balizas. Midiendo la distancia a cada baliza por el tiempo que tarda en

recibir la respuesta, la etiqueta es capaz de calcular su posición respecto a éstas. Teóricamente, son necesarios sólo 4 balizas para posicionar el dron en 3D. Sin embargo, es recomendable instalar balizas adicionales para tener información redundante y reducir la precisión en la localización.

Pozyx ha desarrollado un sistema de posicionamiento basado en UWB con vistas a emplearse con un Arduino. El sistema es ligeramente más barato que el proporcionado por *Bitcraze*, y presenta unas características similares.

Otra posible aplicación de esta tecnología pasa por prescindir de las balizas fijas en el terreno y emplear un solo dispositivo como radar de corta distancia. Esto permite, por ejemplo, emplear este sistema para la localización de personas en espacios semi abiertos [KK18].

2.3. OTUS Tracker

El *OTUS Tracker* es un sistema similar a los MCS de Optitrack y VICON, donde la principal diferencia reside en que emplea únicamente dos cámaras fijas situadas en esquinas opuestas de un recinto de 5x5 metros. Una estructura con múltiples sensores (Figura 2.2) instalada en el dron recibe y analiza los rayos emitidos por cámaras VIVE (como las empleadar en VR), y es capaz de calcular su posición y orientación respecto a ellas.

Figura 2.2. OTUS Tracker.

Debido a que sólo requiere instalar dos cámaras, este sistema es sustancialmente más barato ofreciendo unas características similares. Las únicas limitaciones que ofrece respecto a los sistemas de Optitrack y VICON, es el espacio de trabajo máximo de 5x5m, y la necesidad de instalar una estructura en el dron que, aunque ligera (60g), es algo aparatosa.

2.4. Marvelmind Robotics IPS

La empresa *Marvelmind Robotics* ha creado un sistema de posicionamiento en interiores basado en la triangulación de las señales emitidas desde diversas balizas de ultrasonidos.

Respecto a los sistemas de balizas por UWB, el IPS de *Marvelmind Robotics* es sustancialmente más barato y otorga una mejor precisión (2cm).

2.5. Navegación autónoma

Por último, se disponen sistemas completamente autónomos, capaces tanto de localizarse dentro de un entorno conocido como de efectuar una identificación del entorno mediante técnicas de SLAM (del inglés Simultaneous Localization and Mapping) [Dij12] y Machine Learning. Para ello, se requiere de múltiples sensores a bordo del robot. A continuación, se enumeran los sensores más típicos empleados en UAVs para éste propósito. La mayoría de estos sensores no sirven por si mismos para posicionar completamente el dron, por lo que es necesario integrar varios de ellos para conseguir un UAV completamente controlado.

Cámaras a bordo

Uno de los sensores más empleados en los drones son las cámaras. Éstas permiten, mediante técnicas de *Computer Vision*, obtener información de puntos significativos del entorno para evitar colisiones y reconocer espacios ya visitados.

Sensores de flujo óptico

Un sensor de flujo óptico se basa en una cámara que detecta las singularidades del terreno y las analiza para obtener la velocidad del dron con respecto al suelo. Para poder posicionar un dron correctamente [ZWC+16], es necesario incluir un sonar para conocer la distancia a la superficie captada.

Sensores de distancia

Los sensores de distancia más empleados son los basados en ultrasonidos y los láser. Ambos permiten detectar la distancia a los objetos colindantes más cercanos para evitar colisiones. O, empleando dos sensores paralelos separados una cierta distancia en la estructura del dron, permiten, además, posicionarlo de forma paralela a una pared cercana.

LIDAR

Los sensores LIDAR (Light Detection And Ranging) pueden emplearse en un UAV para obtener información y posicionar el dron en un entorno conocido. O, por otro lado, se pueden implementar técnicas SLAM [LLZH14].

Los sensores LIDAR se basan en calcular la diferencia de tiempo que tarda un haz de luz desde que se emite desde el sensor hasta que vuelve a éste. Se trata, básicamente, de un sensor de distancia muy preciso. La mayoría de sensores LIDAR son capaces de orientar la emisión del haz de luz para obtener información de todas las direcciones.

2.6. VPS de DJI

DJI vende un sistema de posicionamiento visual pensado para desarrolladores. Viene integrado con una IMU, un sonar y múltiples cámaras para detectar obstáculos en las inmediaciones y poder posicionarse.

En la Figura 2.3 puede observarse el módulo Guidance de DJI.

Figura 2.3. DJI Guidance VPS.

2.7. Comparativa entre los sistemas de posicionamiento

En la Tabla 2.1 se muestra una comparativa entre los distintos sistemas de posicionamiento comerciales que se emplean en interiores. En cuanto a las dimensiones de trabajo y al coste, exceptuando los sistemas de VICON y Optitrack, se han empleado datos de productos listos para instalarse y ser usados. En el caso de los sistemas de VICON y Optitrack, el número de cámaras a instalar y, por tanto, el precio del sistema, depende de las dimensiones deseadas del entorno de trabajo. Por ejemplo, el sistema de Optitrack empleado en este proyecto consta de 8 cámaras *Flex13*, la licencia de Software (tracking), la vara de calibración y el recuadro de origen, en total suma $\$9430$.

	MCS			UWB			US IPS
	VICON	Optitrack	OTUS	Pozyx	Bitcraze	DJI VPS	Marvelmind
Dimensiones	Escalable	Escalable	$5x5x3m$	$30x30m$	$6x3x2{,}7m$	$---$	$30x30m$
Coste	Variable	Variable	$\$2280$	$> \$999$	$> \$1000$	$e1099$	$\$399$
Precisión	$< 1mm$	$< 1mm$	$< 1mm$	$10\,\mathrm{cm}$	$10\,\mathrm{cm}$	cm	$2\,\mathrm{cm}$
Orientación	✓	✓	✓	✗	✗	✓	✗

Tabla 2.1. Comparativa entre los sistemas de posicionamiento comerciales

Puede observarse que el mejor sistema en relación calidad/precio es el *OTUS* de *RC Benchmark*, siendo su único inconveniente la restricción de espacio. Por otro lado, fuera de los sistemas de *MCS*, prácticamente ningún sistema dispone de precisión suficiente para realizar controles de posición finos en un recinto pequeño, aunque puedan resultar útiles en espacios en interiores relativamente grandes con ayuda de sensores a bordo para evitar colisiones.

2.8. Aplicaciones del vuelo en interiores

Blue Jay

Blue Jay, junto con Philips Lighting, tiene en desarrollo un dron autónomo para labores de apoyo en interiores. Dicho dron emplea tecnología VLC (*Visible Light Communication*) para posicionarse y navegar en interiores. La comunicación por VLC se basa en emitir pulsos de luz visible desde distintas fuentes (balizas) a distintas frecuencias. Dichos pulsos son detectados por fotodiodos integrados en el dron, de forma que éste es capaz de determinar su posición y orientación en un espacio conocido.

Este cuadricóptero se está implantado en hospitales holandeses como apoyo médico, siendo capaz tanto de acercar objetos a personas con movilidad reducida como de jugar al 3-en-raya con los niños (Figura 2.4).

Figura 2.4. Dron de Blue Jay, jugando al 3 en raya

Inventariado de almacenes

Otra posible aplicación para los drones es el inventariado automático de almacenes. La propia identificación del producto puede hacerse mediante RFID [BHCL16a] o por análisis de imágenes de códigos de barras [XKM18]. A este respecto, en [HGG+16] se solventa el problema del posicionamiento del dron empleando un vehículo terrestre que sirve tanto como referencia de posición del UAV como para transportarlo entre estanterías.

Seguimiento de objetos en interiores

Otra aplicación sería la de identificar y seguir cierto tipos de objetos en espacios cerrados. En un artículo de la ICUAS [CMBH16] se presenta una posible solución, con dos modos de funcionamiento. Cuando se está detectando y siguiendo un objeto en concreto, se emplea una cámara controlada por un servo. En el caso de no detectar ningún objeto interesante, se emplea un sensor de flujo óptico para la estimación del estado.

Ayuda en labores de rescate

En operaciones de rescate, obtener información de forma rápida y eficiente es de gran importancia. Por ello, emplear drones de reconocimiento puede ser de gran ayuda para identificar dónde deben de dirigirse los esfuerzos de forma más inmediata. Dichos drones deben de ser capaces de volar de forma autónoma tanto en interiores como en exteriores.

Por ejemplo, en [ATD14] emplean, a parte de posicionamiento por GPS, una cámara de $720p$ para detectar obstáculos cercanos y posibles personas.

3
Hardware

En este capítulo se muestran los elementos de hardware que se han empleado en el proyecto. En la Sección 3.1 se analiza la estructura que se ha escogido para el cuadricóptero. En la Sección 3.2 se comenta por qué se ha escogido una *Raspberry Pi Zero W* comocontroladora de vuelo, y cómo ha sido configurada para trabajar con *Matlab/Simulink*, acceder a la UART por el GPIO y conectarse a una red WLAN. En la Sección 3.3 se mencionan las características más importantes del módulo *PXFmini* de cara a éste proyecto. En la Sección 3.4 se muestra qué esructura se ha ampleado en los actuadores, y los resultados de la calibración de éstos. Por último, en la Sección 3.5 se muestran las características de las cámaras empleadas para localizar el dron.

3.1. Estructura del dron

Los únicos drones considerados en este proyecto han sido los multirotores, debido a que son capaces de mantener una posición concreta en el espacio y, debido a su configuración, son capaces de moverse en cualquier dirección independientemente de su orientación. En la Figura 3.1 se muestran las configuraciones más típicas de este tipo de aeronaves.

El número de motores a emplear depende, principalmente, del tamaño del dron que se quiera volar, y, más específicamente, del peso de éste. Debido a que se quiere volar en un espacio reducido, en interiores, se ha optado por un cuadricóptero, al ser la configuración simétrica con menos motores que permite moverse en cualquier dirección.

Las configuraciones más características de este tipo de dron son en **+**, **H** y **X**. En la Figura 3.2 pueden apreciarse las configuraciones **+** y **X** y, en la Figura 3.3, la **H**.

La diferencia fundamental entre las configuraciones **H** y **X** es el sentido de giro de los motores. Dos cuadricópteros podrían tener la misma estructura mecánica y distinta configuración de rotores y serían clasificados según ésta última característica.

Si bien con la estructura de control que se va a implementar (Sección 4.1.2) el diseño se efectuaría de la misma forma independientemente de la estructura elegida, por lo que sería

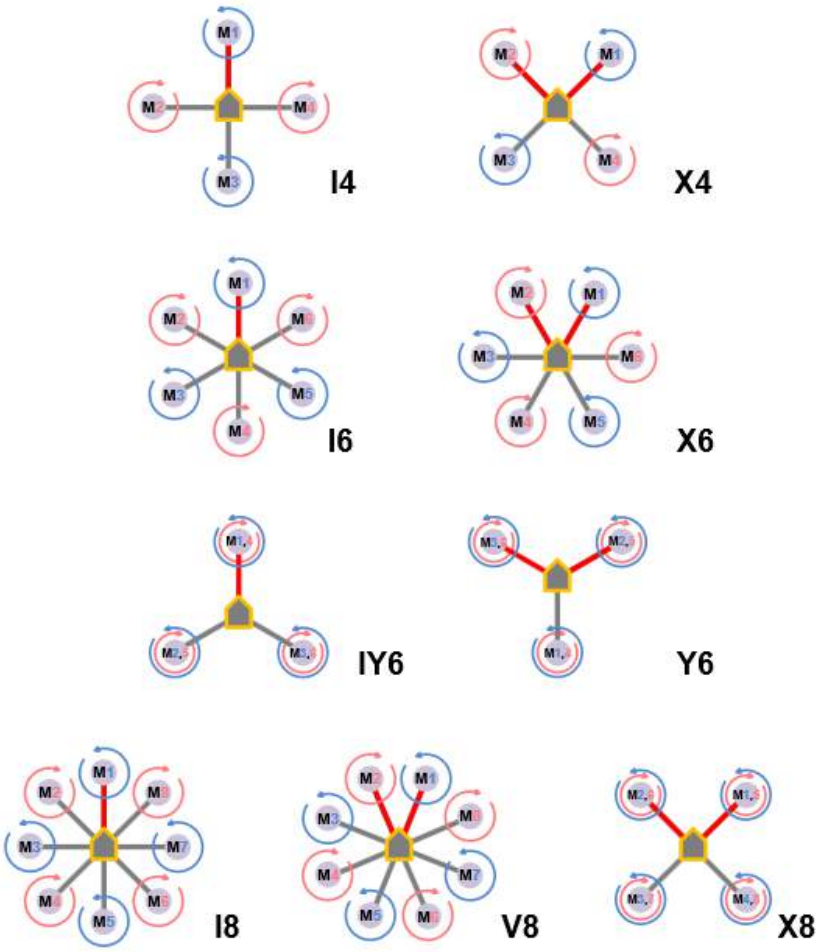


Figura 3.1. Configuraciones típicas de multirotores.

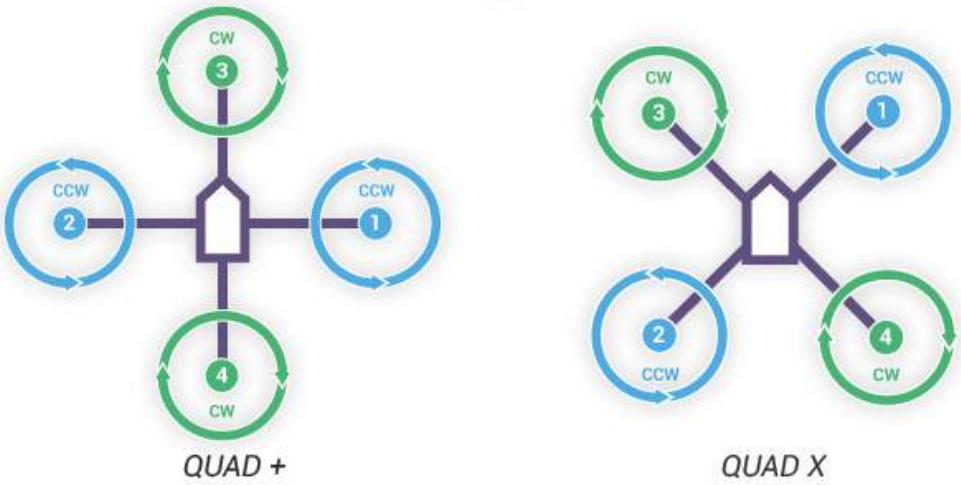


Figura 3.2. Configuraciones + y X.

prácticamente indiferente cuál elegir, las tres presentan sutiles diferencias que se pueden tener en cuenta a la hora de elegir una configuración determinada:

Por un lado, en las configuraciones **X** y **H** es más fácil encontrar una buena posición para incorporar una cámara frontal. Esto es debido a que ninguna hélice se encuentra en esa dirección, como sí ocurre con la hélice de proa en la configuración **+**.

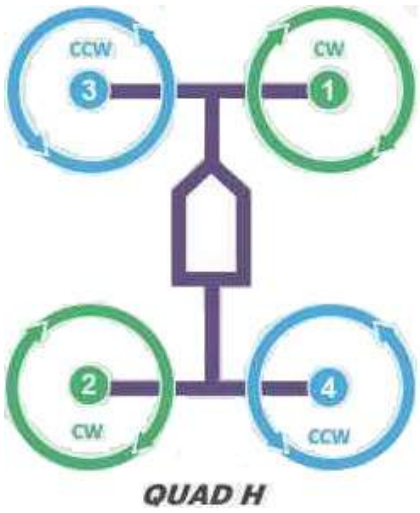


Figura 3.3. Configuración H.

Efectuando vuelos a baja velocidad, como es en este proyecto, las configuraciones **X** y **H**, salvando un signo en el control de la guiñada, se comportan de forma idéntica. Sin embargo, en vuelos a alta velocidad en los que la dirección de vuelo sea predominantemente hacia delante[1], una configuración en **H** es ligeramente más eficiente y presenta una estabilidad mayor frente a los giros. Esto es debido a que las hélices de popa trabajan mejor frente al aire turbulento creado por las de proa cuando la pala más alejada del centro del cuadricóptero avanza, ya que provoca un momento mayor en el dron. Por otro lado, al acercarse a obstáculos frontales, los cuadricópteros en configuración **H** tienden a repelerlos y no quedarse enganchados en ellos, al contrario que los **X**.

En base a lo que se ha comentado en la esta sección, se ha decidido montar un cuadricóptero en configuración **H**. La estructura del dron puede verse en la Figura 3.4.

3.2. Raspberry Pi Zero W

La *Raspberry Pi Zero W* es un pequeño ordenador con sistema operativo *Linux*. Debido a ello es capaz de ejecutar el código como si fuera multitarea. Es decir, permite ejecutar partes del programa a distintos tiempos de muestreo, ya que los considera tareas distintas. En la Figura 3.5 se muestra un resumen de las características de esta tarjeta.

Como controladora de vuelo se empleó una *Raspberry Pi Zero W*. Se hizo uso de ella por su mayor capacidad de cálculo frente a otras controladores, y por la posibilidad de ejecutar partes del programa a distintos tiempos de muestreo. Las características más relevantes se listan a continuación:

- Red WLAN 802 b/g/n
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- Procesador de un núcleo, 16 GHz

Para poder programarla desde *Matlab/Simulink* se puede seguir alguno de los procedimientos siguientes:

[1] Como podría ser en mini drones de competición o en drones dedicados a grabaciones a alta velocidad

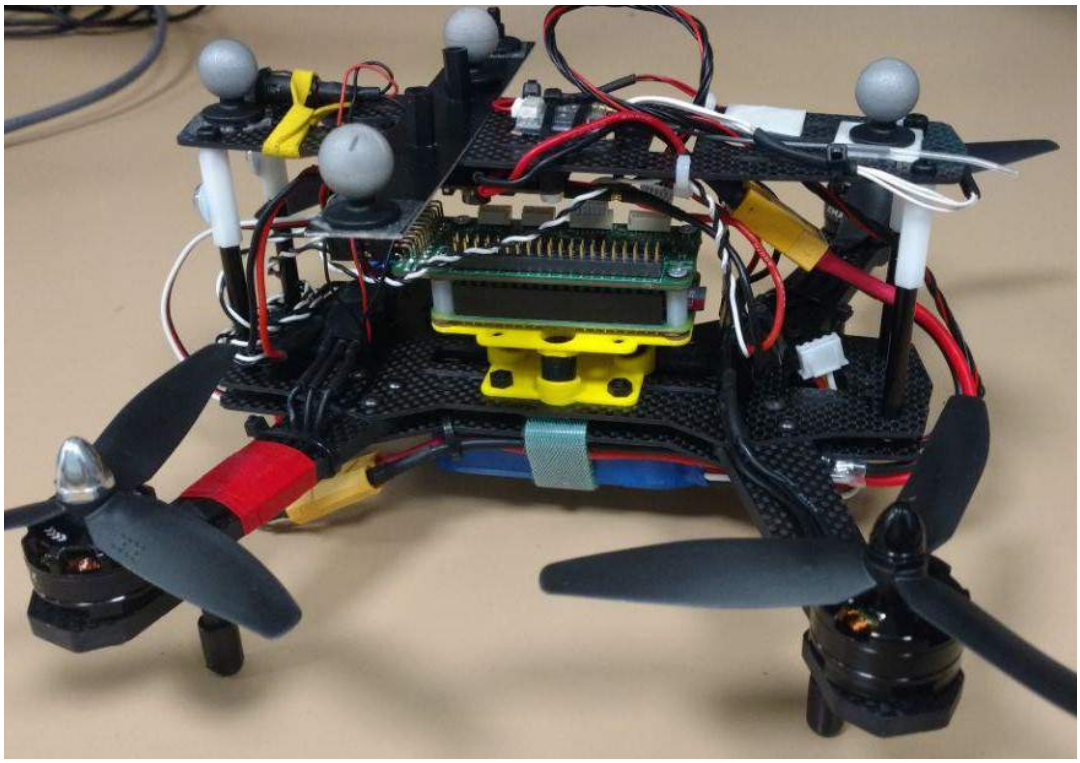


Figura 3.4. Cuadricóptero empleado en el proyecto..

- Cargar la imagen propia de *Mathworks*
- Actualizar desde *Matlab* un sistema operativo de *Raspbian* (*Stretch* o *Jessie*)
- Descargar e instalar por línea de comandos lo necesario para que funcione la programación y comunicación desde *Matlab/Simulink*.

En éste proyecto se empleó el segundo método. Hubo, además, que efectuar alguna configuración adicional para poder utilizar todas las características del hardware. Poara poder conectar la emisora con el protocolo IBUS, se tuvo que deshabilitar el *Bluetooth* y habilitar la UART po el GPIO de la *Raspberry*. Para ello, se modificaron un par de ficheros de configuración de la *Raspberry*:

- */boot/config.txt*

  Añadir '*enable_uart=1*' al final del fichero

  En el caso de querer deshabilitar el *Bluetooth* de la *Raspberry*, debe añadirse también la siguente línea:
  dtoverlay=pi3-disable-bt

  Si se quiere seguir empleando la comunicación *Bluetooth*, puede emplearse la mini-uart. Para ello, se debe añadir lo siguente:
  dtoverlay=pi3-miniuart-bt

- */boot/cmdline.txt*

  Eliminar '*console=serial0,115200*'. Ésto sirve para deshabilitar la consola. Es muy importante no modificar nada más de este fichero, pues cambiarle el formato puede evitar que la *Raspberry*

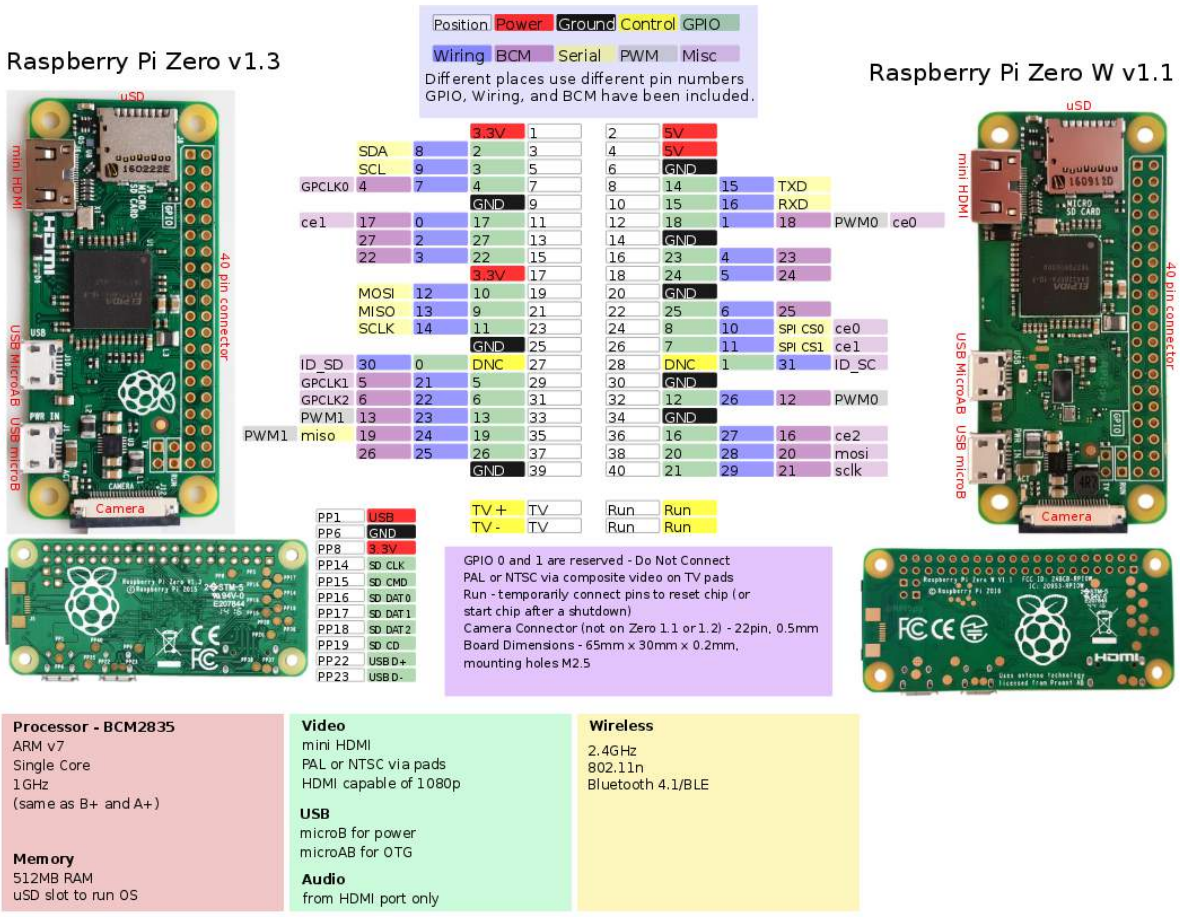


Figura 3.5. Resumen de las especificaciones de la Raspberry Pi Zero..

Por otro lado, se hubo de configurar la conexión a las 2 redes WiFi que se iban a emplear. Con ese propósito, se configuraron los siguientes ficheros:

- */etc/wpa_supplicant/wpa_supplicant.conf*

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network = {
        ssid="SSID_1"
        psk="PASSWORD_1"
        id_str = "home"
}


network = {
        ssid="SSID_2"
        psk="PASSWORD_2"
        id_str = "work"
}
```

- */etc/network/interfaces*

Añadir las siguientes líneas al final del documento:
iface home inet dhcp
iface work inet dhcp

Nótese que ésto es replicable. es decir, se pueden configurar el acceso de las redes WiFi que se desee siguiendo siempre el mismo esquema.

3.3. PXFmini

El *PXFmini* es un módulo de expansión diseñado para la *Raspberry Pi Zero* por *Erle Robotics*. En la Tabla 3.1 se muestra un resumen de los elementos que dispone este módulo.

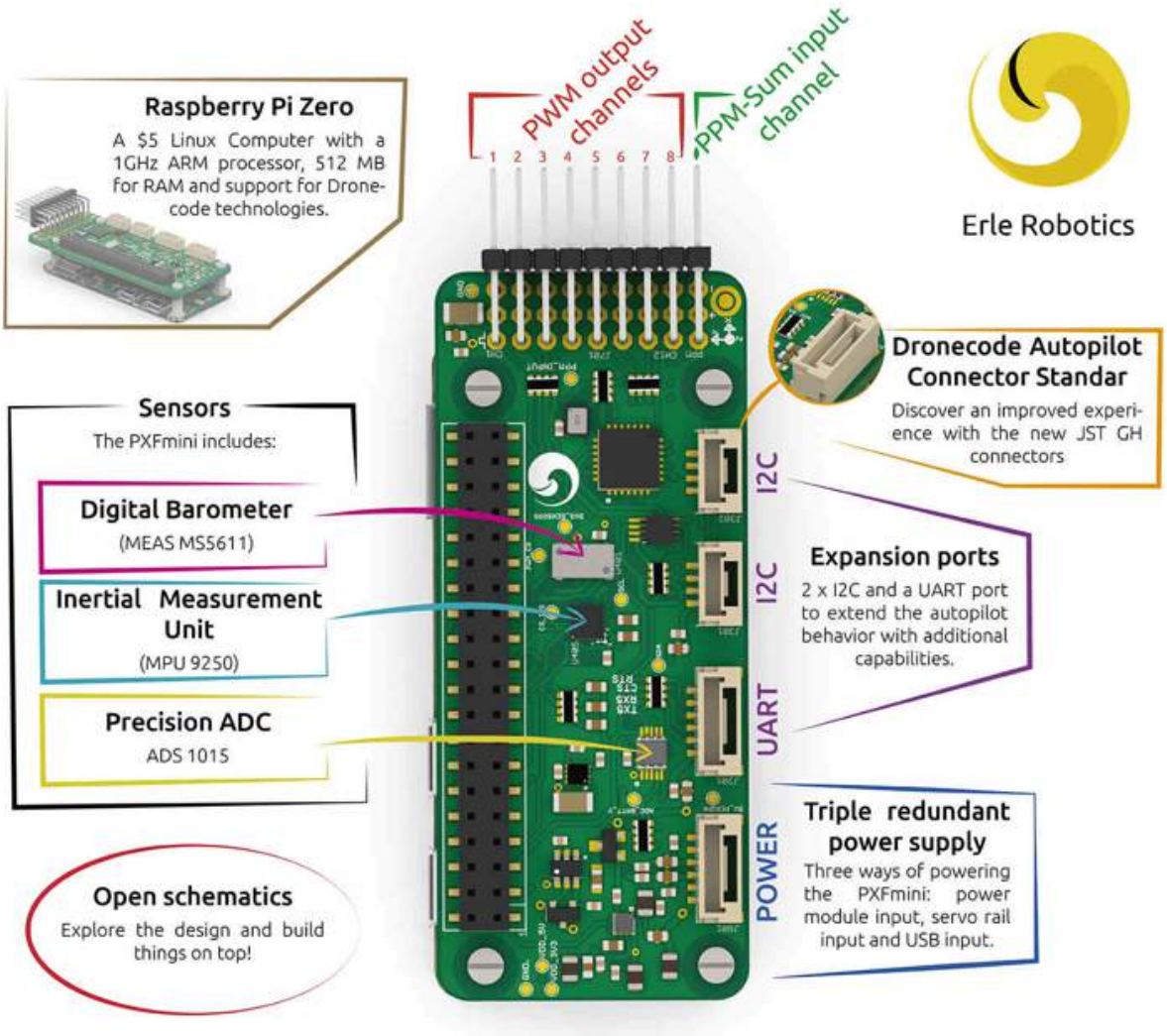


Figura 3.6. Resumen de las especificaciones del PXFmini..

En las siguientes secciones se explican los módulos que se han empleado en este proyecto.

3.3.1. MPU9250

Este sensor lleva incorporados giróscopo, acelerómetro, magnetómetro y sensor de temperatura. En este proyecto solo se ha hecho uso de los sensores inerciales (giróscopo y acelerómetro), de tres ejes cada uno.

PXFmini				
Sensores	MPU9350	Varios	SPI	$\overline{CE}$
	MS5611	Barómetro	SPI	$\overline{CE0}$
	ADS1015	ADC	I^2C	
Actuadores y LEDs	PCA9250		I^2C to PWM (8ch)	'$0x40$'
	LEDs	Azul	GPIO 25	
		Ámbar	GPIO 25	
		Verde	Alimentación	
Puertos de expansión	UART	x1		
	I^2C	x2		

Tabla 3.1. Elementos PXFmini

La MPU9250 trabaja con un reloj propio a $8\,\mathrm{MHz}$, pero la frecuencia mínima de renovación de las medidas es de $1\,\mathrm{kHz}$. Permite, además, la posibilidad de obtener las medidas ya filtradas digitalmente. En este proyecto, sin embargo, se optó por efectuar todas las lecturas posibles y tratarlas en la controladora, ya que se obtuvo mejores resultados en cuanto al ruido de la IMU se refiere.

Para caracterizar el ruido del acelerómetro y el giróscopo, se efectuó un ensayo en el dron. Éste consiste en montar las hélices al revés y hacerlas girar a una velocidad determinada (PWM constante de $1600\,\mu\mathrm{s}$) durante $15\,\mathrm{s}$.

Primero, se efectuó el ensayo simplemente con el filtrado de $20\,\mathrm{Hz}$ propio de la IMU, y efectuando la media de las últimas 10 medidas; con y sin amortiguador mecánico, para ver si de hecho éste ayuda a filtrar o más amplifica las vibraciones.

En la Figura 3.7 y en la Figura 3.8 se muestran los resultados del ensayo sin amortiguador (giróscopo y acelerómetro, respectivamente). Por otro lado, en la Figura 3.9 y la Figura 3.10 se muestran los datos obtenidos del giróscopo y el acelerómetro con amortiguador mecánico.

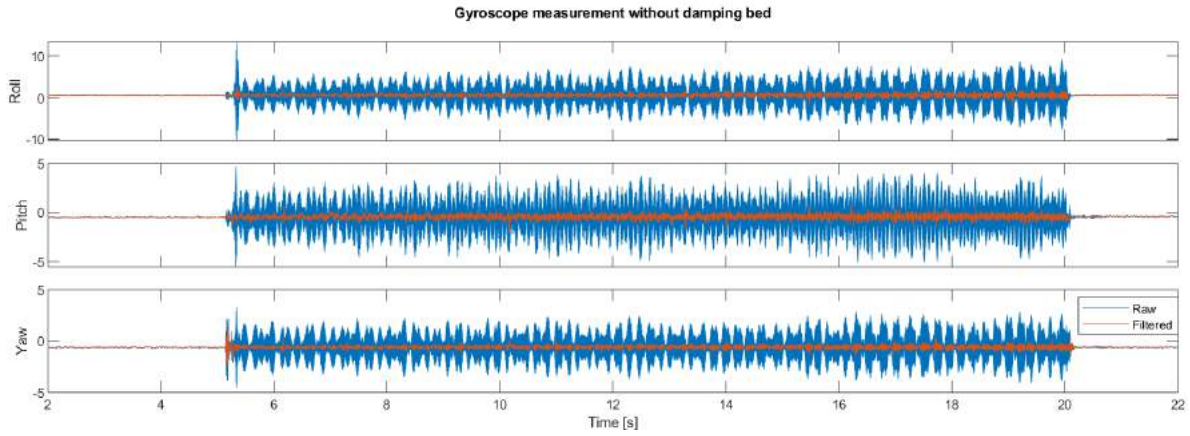


Figura 3.7. Medidas del giróscopo sin amortiguador mecánico.

En la Tabla 3.2 se muestra una comparativa entre las desviaciones típicas del giróscopo cuando se usa y cuando no se usa el amortiguador mecánico.

El amortiguador mecánico empleado ha sido una adaptación de un amortiguador para elementos *CC3D*. A partir de su modelo, se le ha incorporado mediante *Solidworks* unas alas para adaptarlo a la *Raspberry Pi Zero*, y se imprimió en *3D*

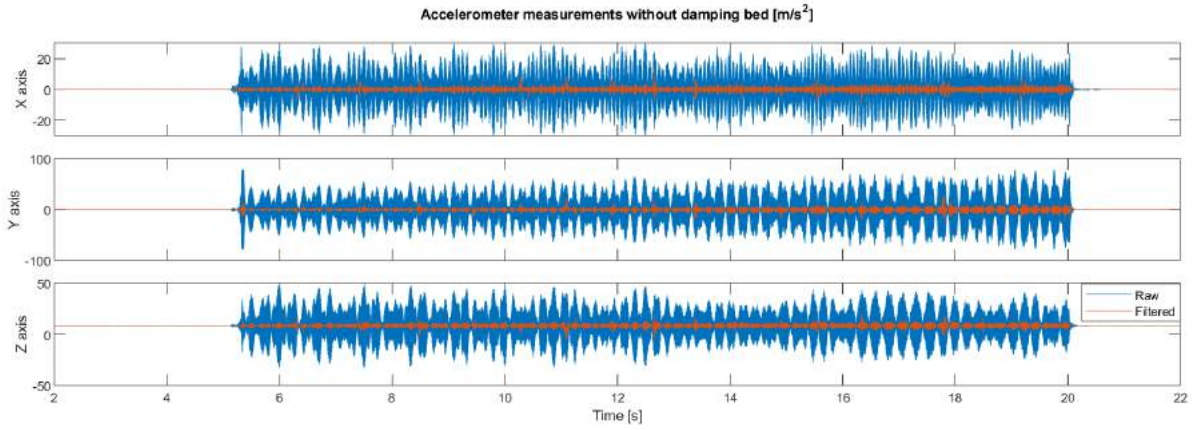


Figura 3.8. Medidas del acelerómetro sin amortiguador mecánico.

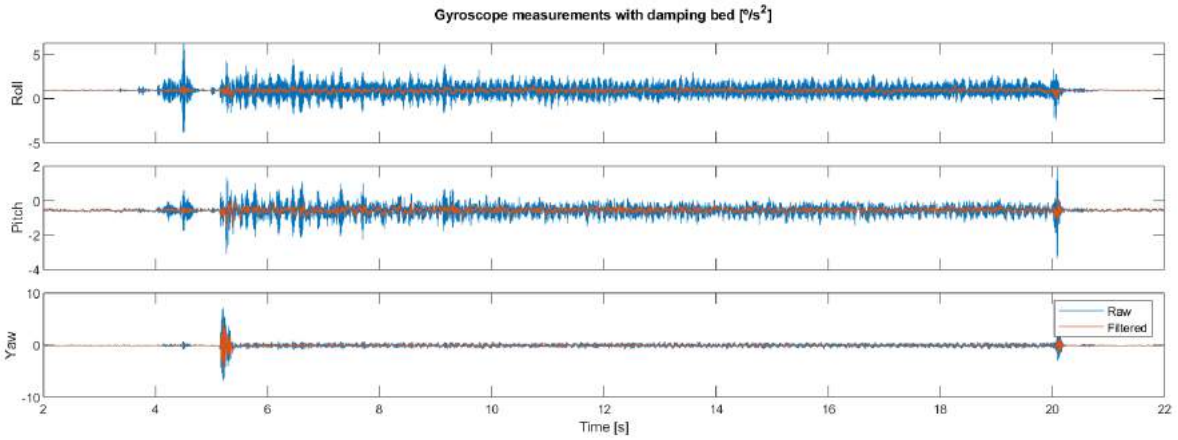


Figura 3.9. Medidas del giróscopo con amortiguador mecánico.

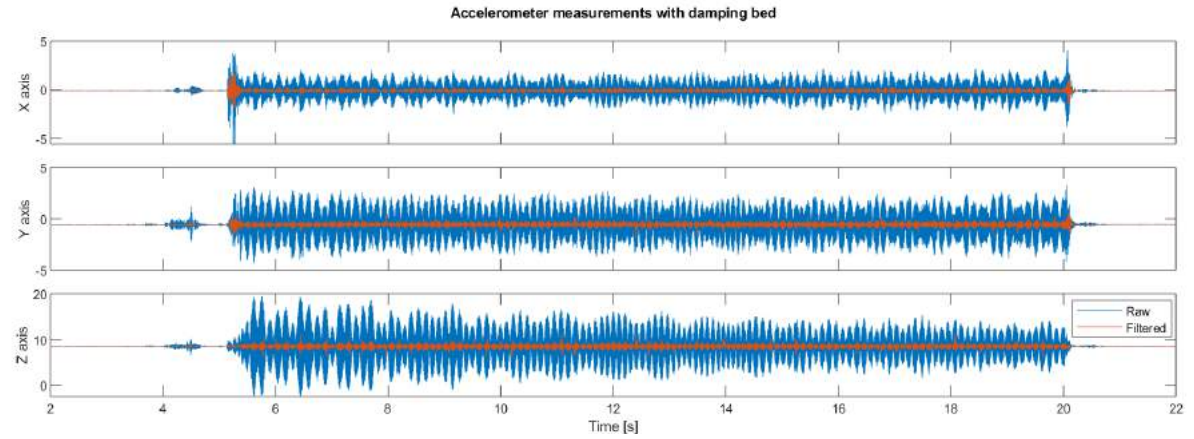


Figura 3.10. Medidas del acelerómetro con el amortiguador mecánico.

En la Figura 3.11 pueden apreciarse la lectura en crudo del giróscopo (azul) y la medida una vez filtrada (rojo) con un filtro digital de 10 Hz. Además, cada medida está compuesta por la media de las últimas 10 medidas para atenuar todavía más el ruido. En la Figura 3.12 se hace lo propio con las aceleraciones en cada eje. En la Tabla 3.3 se muestran las desviaciones estándar de las señales para cada eje, tanto en crudo como filtradas.

IMU with and without damping bed					
		w/o damping bed		w/ damping bed	
Sensor	Eje	STD cruda	STD filtrada	STD cruda	STD filtrada
Acelerómetro $[m/s^2]$	X	11,139	1,511	0,755	0,116
	Y	27,534	3,699	1,315	0,185
	Z	14,397	1,839	3,344	0,439
Giróscopo $[^\circ s^{-1}]$	X	2,597	0,371	0,615	0,185
	Y	1,607	0,255	0,282	0,140
	Z	1,164	0,161	0,234	0,150

Tabla 3.2. Comparación de las desviaciones típicas con y sin amortiguador

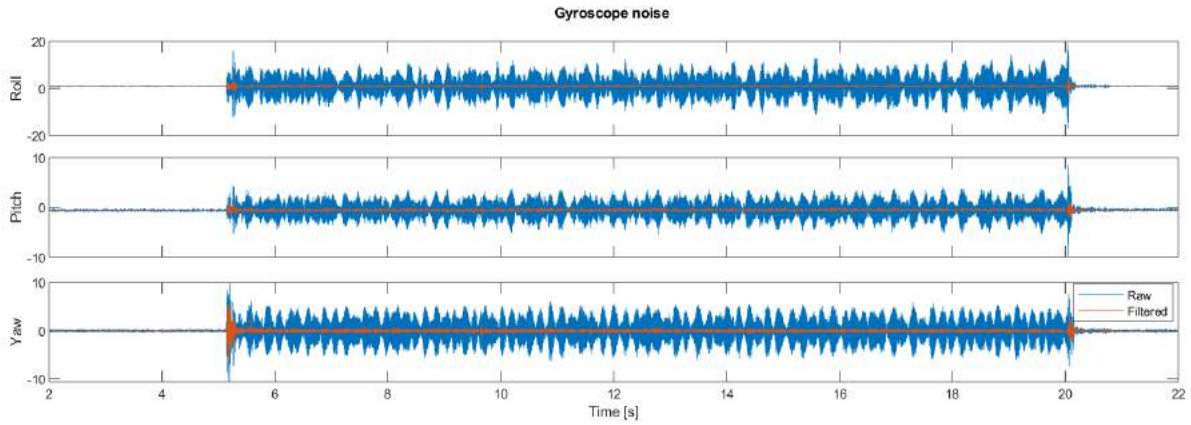


Figura 3.11. Filtrado del giróscopo.

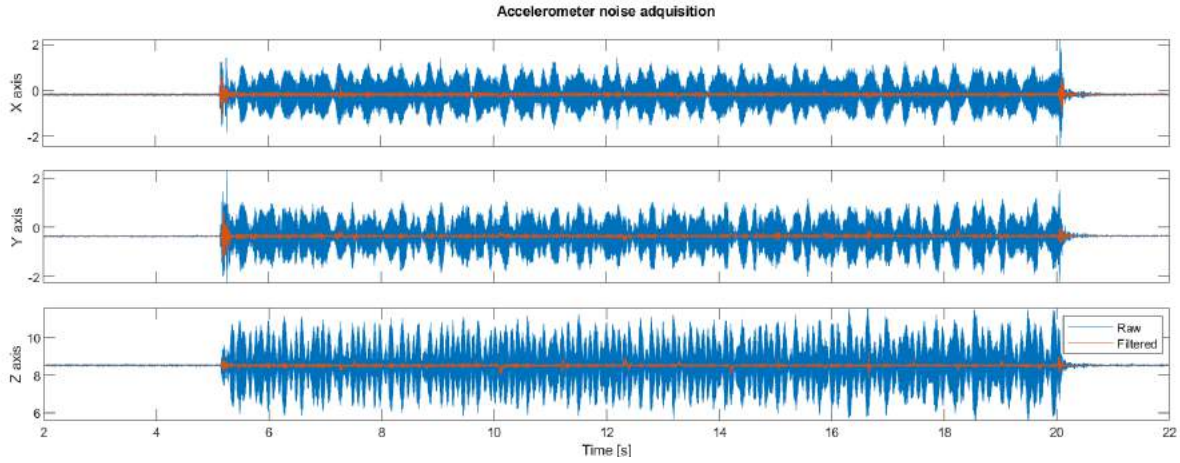


Figura 3.12. Filtraddo del acelerómetro.

IMU STD			
Sensor	Eje	STD cruda	STD filtrada
Acelerómetro $[m/s^2]$	X	0,561	0,048
	Y	0,556	0,048
	Z	1,032	0,066
Giróscopo $[^\circ s^{-1}]$	X	3,958	0,263
	Y	1,583	0,177
	Z	2,294	0,224

Tabla 3.3. Desviaciones estándar de la IMU

3.3.2. PCA9685

El módulo PCA9685 es un controlador de servos con interfaz I^{2}C. A pesar de que dispone de 16 canales, el módulo incorporado en la PCA9685 sólo expone 8 de ellos. Debido a que solo se disponen de cuatro motores, ésto no es un impedimento. Las características más importantes de este IC se listan a continuación:

- Resolución de 12 bits (4096 escalones)
- Frecuencia de salida entre 24 y $1526\,\mathrm{Hz}$
- Pin de 'enable'. Activo nivel bajo. Está conectado al GPIO27 de la *Raspberry*.
- 6 pines de programación de la dirección I^{2}C por hardware. (*0x40* en el caso del módulo *PXFmini*).
- Función de reset por software. Permite inicializar de nuevo el módulo a través de la comunicación I^{2}C.
- Reloj interno de $25\,\mathrm{MHz}$.
- Posibilidad de conectar un reloj externo de hasta $50\,\mathrm{MHz}$ para coordinar varios módulos. (No implementable en el *PXFmini*)
- Rango de temperatura de funcionamiento: $-40\,^{\circ}\mathrm{C}$ a $85\,^{\circ}\mathrm{C}$

El *PCA9685* funciona especificando los momentos en los que la salida tiene que ponerse a nivel alto y a nivel bajo. Por ello, todas salidas digitales disponen de cuatro registros, dos para especificar el tiempo de encendido dentro del periodo y dos para el de apagado. Debido a que van a emplearse para comunicarse con los motores, estos registros se escriben de la siguiente forma:

- Registros de encendido.
  Se ponen a 0 para encender la salida al inicio del periodo de la señal.
- Registros de apagado.
  Siguen la siguiente ecuación para generar la señal deseada:

$$Off = Pulse \cdot Freq \cdot 10^{-6} \cdot 4096 - 1$$

  Donde:

  Off : tiempo en el que se pasa a $0\,\mathrm{V}$, en número de 'muestras' (0-4095).

  $Pulse$: amplitud del pulso *PWM* [μs].

  $Freq$: Frecuencia de la señal *PWM* [Hz]. En nuestro caso, $432.5\,\mathrm{Hz}$.

3.4. ESC, motores y hélices

Como actuadores se emplearon unos motores de continua sin escobillas (BLDC) *MT2204-2300KV* de *EMAX*, controlados a través de unos ESC *dys SN 16A*. Las hélices empleadas fueron unas 5030[2] de tres palas. La señal de control de los ESC la genera el *PCA9685*.

[2]El número que acompaña a las hélices simboliza las dimensiones de ésta. Concretamente, especifica el diámetro y el paso. En este caso (5030), las hélices presentan un diámetro de 5 pulgadas y un paso de 3 pulgadas

El conjunto completo se ensayó en un banco de ensayos para caracterizar su empuje frente al pulso de PWM aplicado al ESC. Los cuatro motores dieron resultados muy parecidos (Tabla 3.4), y la media de todos ellos puede apreciarse en la Figura 3.13.

Empuje de los motores [g]					
PWM	PROA ESTRIBOR	POPA BABOR	PROA BABOR	POPA ESTRIBOR	Media
1000	0	0	0	0	0
1100	16	17	16	19	17
1200	39	45	36	38	40
1300	70	74	67	67	70
1400	98	106	96	97	99
1500	132	141	130	128	133
1600	161	173	163	166	166
1700	200	210	193	206	202
1800	230	246	230	242	237
1900	273	292	264	283	278
2000	322	335	308	329	324

Tabla 3.4. Resultados de la caracterización del empuje de los motores

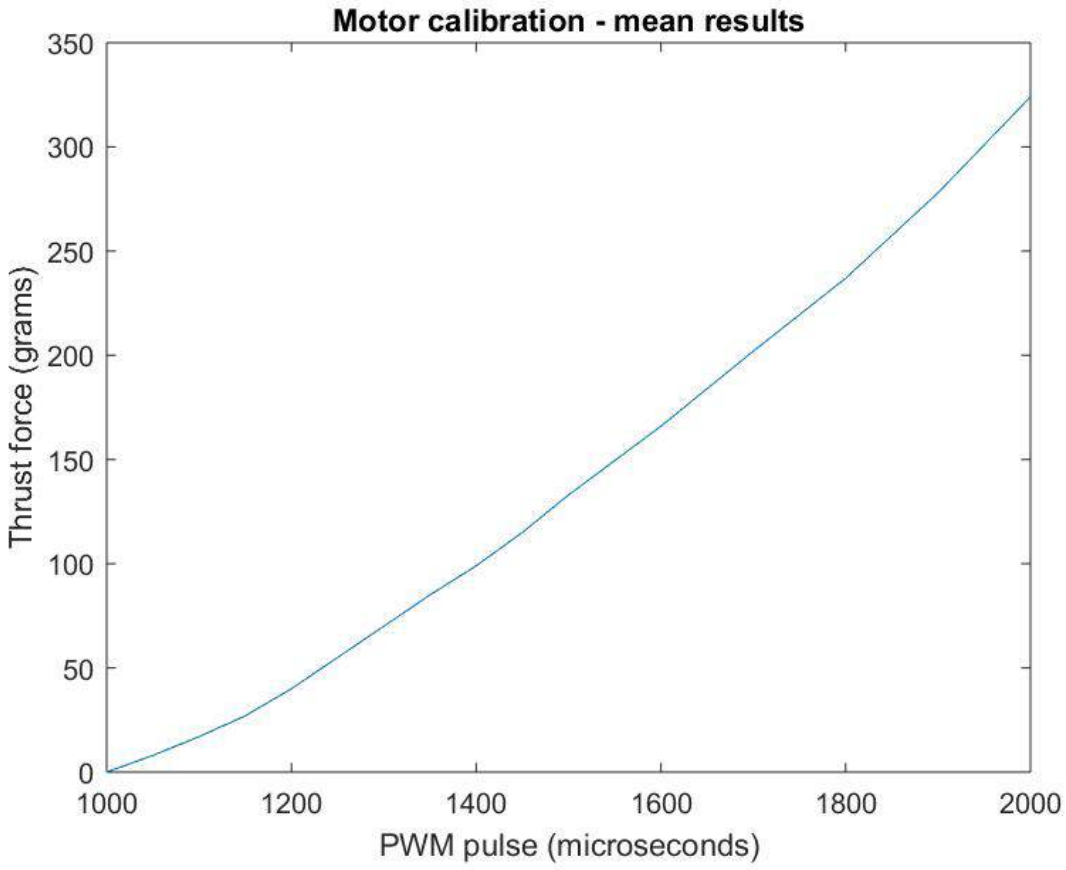


Figura 3.13. Calibrado medio del empuje de los motores.

3.5. Cámaras Optitrack Flex 13

Como parte del sistema de localización, se emplearon 8 cámaras *Flex 13* de *Optirack*. Éstas se conectan con un ordenador a través de 3 *OptiHub 2*.

Las cámaras *Flex13* tienen una corona de 28 LEDs que emiten luz infrarroja a $850\,\mathrm{nm}$ y brillo ajustable. Ésta es reflejada por cuatro elementos reflectantes situados en la estructura del dron, y posteriormente es captada por las cámaras. Éstas envían la información a *Motive*, que se encarga de analizarla y extraer la posición y la orientación del dron.

Las características técnicas de estas cámaras se listan a continuación:

- Corona de 28 LEDs infrarrojos $(850\,\mathrm{nm})$
- Lente de $5\,\mathrm{mm}$ F1.8, con un campo de visión de $46°$ verticales y $56°$ horizontales.
- Filtro paso alto de $800\,\mathrm{nm}$ para evitar interferencias con la luz ambiente.
- Sensor de Imagen:

  Resolución de 1.3 Megapíxeles $(1280$ x $1024)$.

  Tamaño de píxel: 4.8x$4.8\,\mu\mathrm{m}$

  Velocidad de adquisición de datos: 30 - 120 FPS

Figura 3.14. Dimensiones de la Flex13.

4

Software

En este capítulo se explica el software que se ha empleado en las labores más más importantes del proyecto: *Matlab/Simulink*v(Sección 4.1), *Motive* (Sección 4.2) y *BLHeli Suite* (Sección 4.3)

4.1. Matlab / Simulink

Para programar el cuadricóptero, se ha empleado *Matlab/Simulink* con el paquete de soporte de Raspberry Pi para *Simulink*. Ésto ha solventado de forma sencilla el controlar los periféricos, lo que ha permitido centrarse en el tratamiento de la información y la implantación del control.

En cuanto a los criterios de programación, se ha querido que el mismo diagrama de *Simulink* sirva tanto para programar la *Raspberry* como para simular el sistema. Por ello, se ha estructurado la programación en tres bloques, como puede apreciarse en el esquema siguiente:

- **AUTOPILOTO**
  - **Hardware**
    - Lectura y análisis de los sensores
    - Comunicación con la GCS
    - Gestión de actuadores
  - **Control**
    - Máquina de estados
    - EKF
    - Ejecución del control
  - **Simulación**
    - Simulación de los sensores
    - Simulación del modelo
    - Visualización de variables (Scopes)

El bloque de *Control* siempre está activo. Con ello se consigue que el mismo control que se emplea en la simulación sea el que se descargue al dron sin necesidad de copiarlo ni transferirlo a un fichero distinto.

Los bloques de *Hardware* y *Simulación* nunca están habilitados de forma simultánea. El primero sólo se activa cuando se quiere implementar el modelo en la *Raspberry*, mientras que el segundo se activa para efectuar simulaciones y analizar datos.

4.1.1. Máquina de estados

La máquina de estados implementada sigue una estructura muy lineal. Tras tres segundos de inicialización, el dron entra en un estado de calibración de sensores, donde calcula el offset de medida que tienen éstos respecto a las condiciones que debería medir[1].

Al finalizar la calibración de los sensores, el dron espera a una señal de confirmación del usuario conforme todo está correcto para iniciar el armado de los motores. El operario puede, opcionalmente, efectuar un calibrado de los parámetros de la emisora, para centrar todos los canales y evitar posibles alteraciones respecto a los parámetros que tenga cargados el dron.

A continuación se entra en el procedimiento de armado de los motores, que consiste en enviar activamente una señal durante un intervalo de 2 segundos[2]. Tras armar los motores, se puede proceder a despegar el dron y activar los controles de vuelo pertinentes.

En la Figura 4.1 se muestra una máquina de estados simplificada respecto a la que se ha implementado en el proyecto.

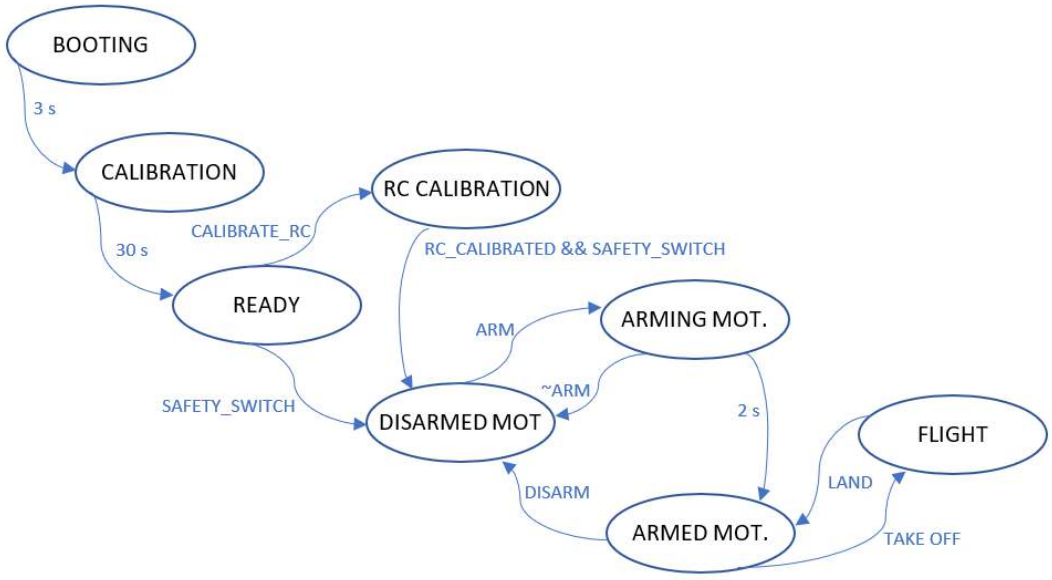


Figura 4.1. Máquina de estados simplificada.

4.1.2. Control

La estrategia implementada trata de abstraer el diseño del control de los parámetros físicos del dron. Para ello, se hace uso de las matrices obtenidas en el **??**. Esto permite tanto diseñar para un dron genérico como desacoplar los controles de cada ángulo. Una vez desacoplados los parámetros físicos, el modelo matemático del dron resultan ser cuatro integradores en cascada

[1]Es muy importante tener el dron sobre una superficie plana horizontal al iniciar el programa, y mantenerlo ahí al menos hasta que salga del estado de calibración.

[2]En el caso de este proyecto, por ejemplo, sería mantener un joystick de la emisora en una de sus posiciones extremas durante dichos dos segundos.

cuyas salidas son, respectivamente, las velocidades y orientaciones angulares, por un lado, y la velocidad y posición en el espacio. Nótese que, para transformar la señal de la orientación angular a la aceleración lineal en los ejes, se debe de aplicar una ganancia equivalente a la aceleración de la gravedad.

Por otro lado, se ha optado por cuatro controles Proporcional-Integral-Derivativo (PID) en cascada, cuya estructura puede observarse en la Figura 4.2. Dichos controles se encargan de regular, respectivamente, la posición en el espacio, la velocidad en éste, las orientaciones en ángulos de Euler y las velocidades angulares de Euler.

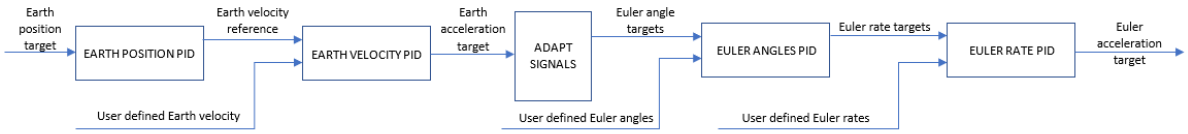


Figura 4.2. Estructura del control.

Modos de control

Los modos de control que se han implementado en este proyecto son los siguientes:

- Estabilización
  Este modo de control está pensado para funcionar en modo manual con emisora. El sistema de control sólo activa los controles de Alabeo, Cabeceo y las velocidades angulares en los tres ejes.

- Orientación
  Este modo también está pensado para funcionar con emisora. La diferencia con el anterior es que éste corrige las perturbaciones en la guiñada automáticamente, evitando así que el operario tenga que estar continuamente pendiente de corregirlo él.

- Altura constante
  En este modo de vuelo, el dron mantiene una altura constante de forma autónoma. Esto evita al operario tener que corregir continuamente el throttle de la emisora para mantener el dron a una altura aceptable.

- Altura y orientación
  Este modo aúna las ventajas de los dos anteriores al controlar automáticamente la guiñada y altura del UAV. Por ello, es el modo de vuelo ideal para familiarizarse con el comportamiento del dron ante los cambios de la emisora.

- Navegación 3D
  Éste es el modo de vuelo por excelencia del UAV. Aquí el dron vuela de forma autónoma respondiendo únicamente a las indicaciones especificadas desde la GCS.

- Velocidad 3D
  En este modo de vuelo, los ángulos van completamente controlados por el dron, y el usuario simplemente proporciona referencias de velocidad en los ejes de coordenadas.

- Velocidad 3D y altura
  Este modo de vuelo es idéntico al anterior, salvo por el hecho que el dron controla activamente la altura alcanzada en el momento en el que se anula la referencia de velocidad en el eje Z.

4.1.3. Filtro Extendido de Kalman

Debido a que se disponen de formas redundantes para calcular las variables de estado del sistema, se ha optado por emplear un EKF como estimador de estado. Esto permite trabajar con variables más precisas que las medidas directas de los sensores.

El EKF tiene un modelo inercial. Ésto implica que se basa en integrar las medidas del giróscopo y el acelerómetro para obtener, por un lado, los ángulos de *Euler*, y por otro, la velocidad y posición en el espacio. Para corregir dichas integraciones, hace falta sensores adicionales. Por ejemplo, el mismo acelerómetro se encarga de corregir de forma esporádica la medida del giróscopo. Ésto ocurre cuando el acelerómetro capta una aceleración equivalente a la gravedad en el eje Z, y los ángulos de cabeceo y alabeo son cercanos a cero.

El EKF implementado presenta varios modos de funcionamiento, a activar en función de los sensores disponibles y el modo de vuelo.

Estados del EKF:

- Deshabilitado
  El EKF no estima ninguna variable en este modo. Es el estado por defecto, y únicamente debería salir de él una vez haya pasado la etapa de calibración de los sensores.
- PITCH & ROLL
  En este modo, el EKF estima únicamente los ángulos de cabeceo y alabeo. Es el que debería estar activado cuando solo se dispone de la IMU como sensor, ya que en ese caso el vuelo es siempre manual y no se tiene ninguna medida directa de la guiñada para corregir el error acumulado de ésta.
- Ángulos de Euler
  El EKF estima la orientación completa del dron. Este estado es idóneo cuando se quiere efectuar un control manual y se cuenta con un sensor que corrija el ángulo de guiñada, como un magnetómetro; ya que permite controlarlo de forma automática. Esto ahorra al operario el tener que estas constantemente pendiente de corregirlo.
- PITCH, ROLL y altura
  En este estado, además del alabeo y el cabeceo, se estima también la altura a la que se encuentra el dron. Para que la estimación no acabe divergiendo, es necesario contar con un sensor que pueda medir la altura directamente, para corregir la estimación. Un sónar, un barómetro o las mismas cámaras de Optitrack servirían para este propósito.
- Ángulos de Euler y altura
  Este estado es igual que el anterior, con la salvedad que el EKF estima además la guiñada.
- Velocidad XY
  En este modo, se calculan todos los ángulos de Euler y las velocidades en los ejes horizontales.
- Velocidad XY e altura
- Posición en el espacio
  En este estado se estiman todas las variables de salida del EKF. A saber, ángulos de Euler y velocidad y posición en el espacio. Para el correcto funcionamiento de este modo, hace falta o bien un sistema de geolocalización[3] o múltiples sensores integrados en el dron que

[3] En exteriores, GPS. En interiores, se puede emplear un sistema de cámaras, beacons...

sean capaces de medir velocidad y corregir la posición[4]. En este proyecto, se ha empleado un sistema de cámaras de IR para captar la posición del dron.

4.1.4. Estación de Tierra (GCS)

La GCS se implantó también mediante *Simulink*. Se dividió en cuatro bloques principales:

- Comunicación *MAVLink*
  Éste bloque gestiona toda la comunicación por *MAVLink*.
- Gestión de las cámaras de *Optitrack*
  Aquí se efectúa la gestión de los datos que se reciben desde *Motive*. Básicamente, los cuaterniones son convertidos a ángulos de Euler del cuerpo, se escalan tanto los ángulos como la posición del dron en el espacio y se carga la información en un bus de datos para que sea enviada mediante el protocolo *MAVLink*.
- Monitorización En este bloque se representan gráficamente distintas señales que se reciben desde el dron por *MAVLink*. Dichas señales son escogidas a la hora de configurar el proyecto, y se emplean para ver el estado del dron tanto en vuelo como una vez ha acabado éste.
- Ajuste de parámetros
  Por último, se encuentra un bloque de ajuste de parámetros. Éste, junto a la comunicación *MAVLink*, permite modificar parámetros del control o la adquisición de datos sin necesidad de reprogramar el dron.

4.1.5. Black Box

Se implantó una caja negra para guardar datos en la tarjeta SD de la *Raspberry* durante el vuelo. Para ello, basta con añadir un bloque '*To File*' al diagrama de *Simulink* donde entren las variables que se quieran guardar, y especificar el nombre del fichero dónde quiera guardarse.

4.2. Motive

Motive es la aplicación que se encarga de recibir y analizar la información captada por las cámaras externas. Se comunica con la GCS mediante streaming. *Motive* envía la información a una red en concreto (en éste caso, al propio ordenador, pues se ha decidido trabajar con la GCS en el mismo ordenador que *Motive*, aunque se haya dejado preparado para trabajar en ordenadores distintos), y la GCS los lee mediante una función proporcionada por *Optitrack*. En la Figura 4.3 se muestra un detalle de la aplicación durante una prueba de vuelo.

Motive cuelga, para cada rígido correctamente definido e identificado, su posición en el espacio de trabajo y la orientación en forma de cuaterniones.

En la Figura 4.4 puede apreciarse una comparativa entre los ángulos de Euler obtenidos con el EKF usando sólo la IMU y la lectura que efectúa *Motive*. Se denota que la IMU es capaz de estimar correctamente los ángulos de cabeceo y alabeo, pero no la guiñada.

[4]Por ejemplo, podría emplearse un sensor de flujo óptico para medir la velocidad en los ejes horizontales, un sónar para la altura y sensores de Infrarrojos (IR) o Ultrasonidos (US) para medir la distancia a los obstáculos

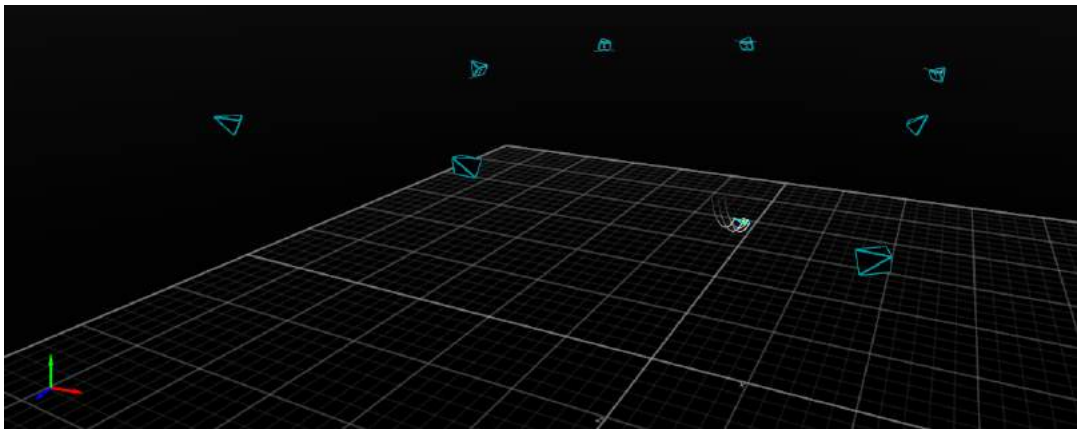


Figura 4.3. Comparación de ángulos de Euler EKF-Motive.

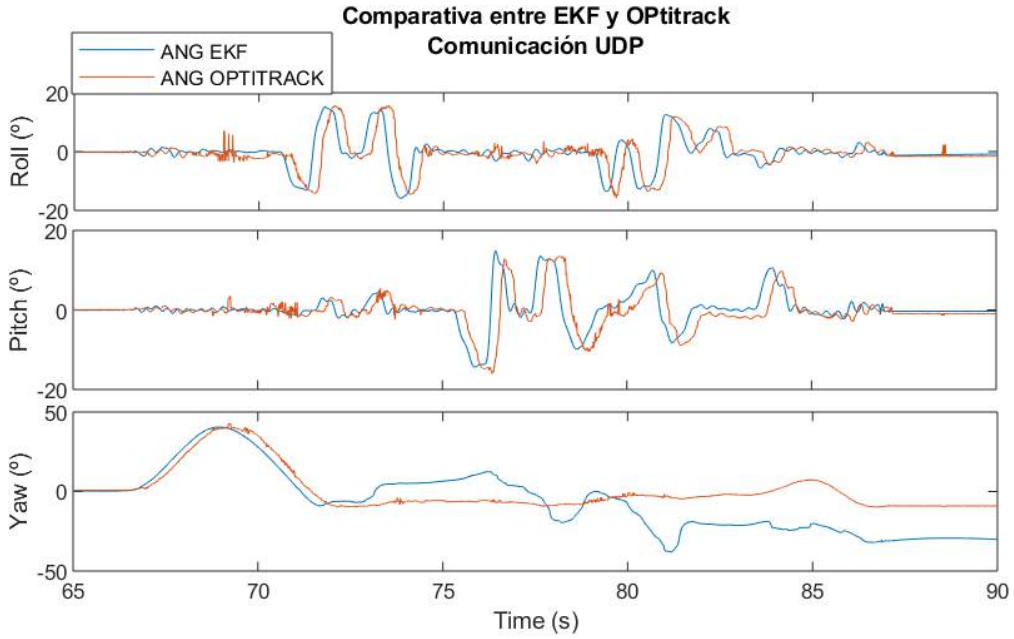


Figura 4.4. Comparación de ángulos de Euler EKF-Motive.

El sistema de cámaras debe de calibrarse una vez montado. Para ello, se emplea un procedimiento de 'wanding', que consiste en recorrer toda la zona de trabajo moviendo la *CWM-250* formando ochos. El sistema recoge muestras de la posición y orientación de dicha vara para obtener la posición relativa entre las cámaras activas. En la Figura 4.5 puede verse una captura de *Motive* tras haber calibrado el sistema.

Una vez calibrado, se debe especificar a la aplicación dónde se encuentra el centro del recinto. Para ello, se dispone el *CS-200* en la posición y orientación deseados del origen de coordenadas.

El sistema sólo debería calibrarse de nuevo en el supuesto de que se modifique la posición y/o orientación de las cámaras. Sin embargo, se recomienda activar el calibrado continúo (*Application Settings* > *Live Reconstruction* > *Continuous Calibration*) para que *Motive* sea capaz de calibrar automáticamente pequeñas alteraciones. La calibración continúa puede especificarse a los siguientes parámetros:

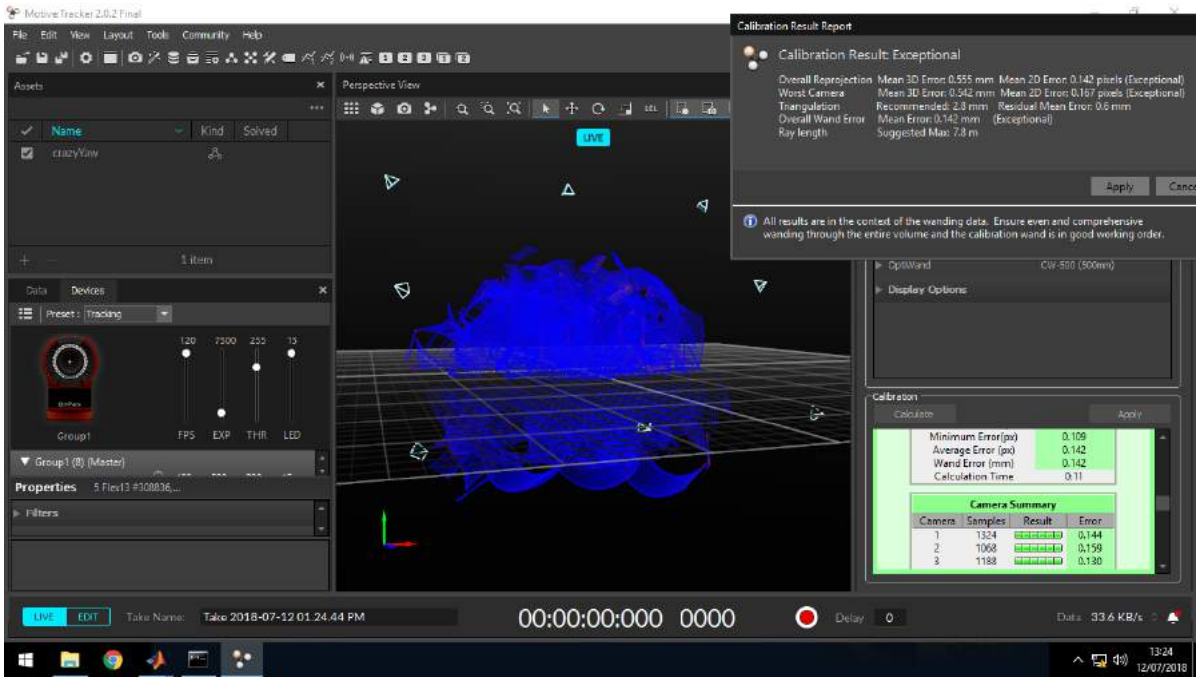


Figura 4.5. Comparación de ángulos de Euler EKF-Motive.

- *Disabled*
  La calibración continúa está deshabilitada.
- *Continuous*
  En éste modo, *Motive* optimiza constantemente el calibrado de las cámaras. Sólo es válido para pequeños cambios como vibraciones, expansiones térmicas y pequeñas variaciones en la posición y orientación de las cámaras.
- *Continuous + Bumped*
  Este modo sirve para cuando se conoce que una cámara en concreto ha sido movida drásticamente. En estos casos, en vez de calibrar de nuevo todo el sistema, se puede especificar qué cámara es, y la aplicación tratará de recalibrarla automáticamente.
- *Continuous Overhead Grid*
  Este modo es similar al de calibración continua. Sin embargo, no requiere que todas las cámaras detecten los marcadores a la hora de efectuar la corrección.

En el caso de que se haya movido drásticamente alguna cámara, también pueden calibrarse únicamente éstas. Sin embargo, el procedimiento es el mismo que el de calibrado completo, por lo que suele calibrarse siempre todo el conjunto de cámaras.

4.3. BLHeli Suite

Esta aplicación se empleó para descargar el firmware a los ESC y configurarlos. En la Figura 4.6 pueden observarse los parámetros empleados para uno de los ESC. Los ESC cargados con el firmware de *BLHeli* detectan de forma automática cuándo reciben una señal PWM estándar o cuando reciben un *Oneshot 125*, eliminando así la necesidad de reconfigurarlos se se decide cambiar de protocolo. Es recomendable activar la opción '*Damped Light*' en la configuración, ya que así el dron frena de forma activa los motores cuado tiene que bajar las revoluciones, en vez de esperar a que se frenen por la fricción de las palas con el aire.

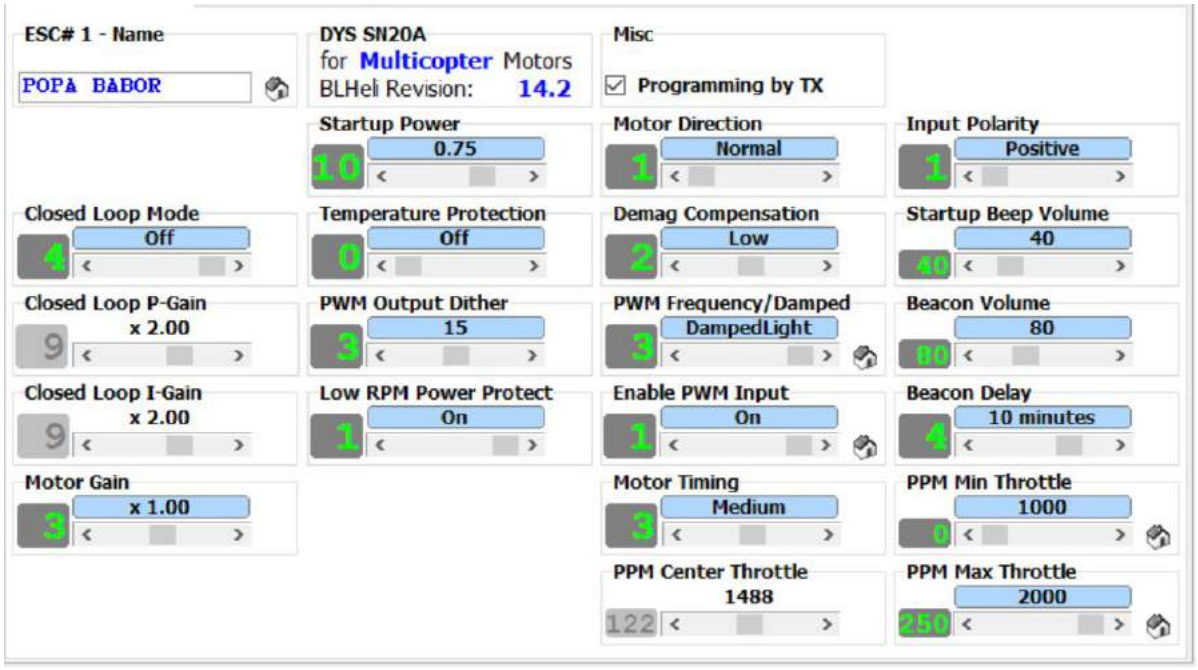


Figura 4.6. Configuración de los ESC en BLHeli.

5 Comunicaciones

En este capítulo se presentan las distintas comunicaciones que se efectúan en el sistema. En la Sección 5.1 se explica cómo funciona el protocolo *MAVLink* y cómo se ha implantado. Po otro lado, en la Sección 5.2 se muestran los distintos protocolos de comunicación entre la emisora y el dron que se han estudiado.

5.1. MAVLink

MAVLink es un protocolo de comunicaciones especialmente diseñado para la comunicación entre pequeñas aeronaves y una GCS. La estructura de este protocolo se muestra en la Figura 5.1. La descripción de cada campo de la trama se recoge en la Tabla 5.1.

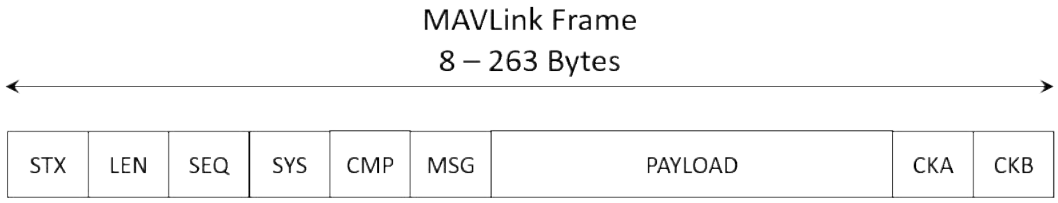


Figura 5.1. Estructura de un mensaje de MAVLink.

El protocolo MAVLink presenta varios mensajes típicos que han sido normalizados, como pueden ser los datos de la IMU o los mandos de actuación (PWM enviados a los ESC). Pero también permite crear mensajes propios. Por ejemplo, se ha implementado un mensaje de MAVLink para modificar los parámetros del control para ser capaces de efectuar un ajuste fin o de este sin necesidad de programar de nuevo la *Raspberry* cada vez.

En cuanto al medio de comunicación del protocolo, se han implementado tres formas distintas: por Bluetooth (serie), mediante TCP/IP y UDP. A este respecto, cabe destacar que *MAVLink* se encarga de tres comunicaciones de forma simultánea:

1. Visualización de variables.
   MAVLink permite ver y analizar parámetros y variables del dron durante el vuelo. No es una comunicación crítica en el tiempo, es cuanto a que se trata simplemente de una forma

Campo	Índice (bytes)	Descripción
STX	0	Indica el inicio de transmisión del mensaje
LEN	1	Especifica la longitud del PAYLOAD en bytes (n)
SEQ	2	Número de secuencia. Para dividir en múltiples tramas mensajes muy largos que no quepan en una sola.
SYS	3	Identificación del sistema emisor del mensaje.
CMP	4	Identificación del componente dentro del sistema emisor.
MSG	5	Identificador del mensaje enviado. Sirve para detectar cómo ha de descodificarse.
PAYLOAD	6:n+6	Información del mensaje.
CKA	n+7	Check-sum LSB
CKB	n+8	Check-sum MSB

Tabla 5.1. Descripción de los campos de una trama MAVLink.

MAVLink	Ventajas	Inconvenientes
UDP	No ocupa periféricos Más rápido que TCP/IP (no comprueba errores) Fácil configuración con Simulink	No es seguro recibir el paquete Requiere conexión WiFi
TCP/IP	No ocupa periféricos Comprueba la correcta transmisión Fácil configuración con Simulink	Puede ralentizarse (comprobación de errores) Requiere conexión WiFi
Bluetooth	No requiere conexión WiFi	Ocupa una UART Requiere de hardware adicional en la GCS Requiere configurar directamente la Raspberry

Tabla 5.2. Comparativa entre los tres protocolos implementados

de comprobar genéricamente si el sistema funciona correctamente. De hecho, al haber implementado una forma de guardar datos en la tarjeta de memoria durante el vuelo, se elimina la necesidad de enviar datos por *MAVLink* para su posterior análisis.

2. Envío de los datos de la misión.
   Mediante el protocolo *MAVLink* también pueden enviarse los parámetros del vuelo a efectuar. Tampoco es un proceso crítico, siempre y cuando no haya un retardo tan grande que sea apreciado a simple vista.

3. Envío de los datos de *Motive*.
   Se emplea *MAVLink* para cerrar el lazo con el sistema de cámaras de *Optitrack*. Debido a que ésto tiene impacto directo tanto en el control de estabilidad como en el de navegación, sí es una comunicación crítica.

En la Tabla 5.2 se muestra una comparativa entre las ventajas e inconvenientes de cada uno de los métodos implementados. Debido a que la UART se requiere para recibir datos de la emisora (Sección 5.2), se requieren velocidades de transmisión altas, no es preocupante que se

pierda algún paquete de datos puntual y se requiere de una conexión WiFi para programar la *Raspberry*, se ha optado por emplear el protocolo UDP para comunicar el cuadricóptero con la GCS.

5.2. Emisora

Para recibir los datos de la emisora, se requiere tener un receptor de radio enlazado con ésta en el dron. Además, es necesario comunicar este receptor con la controladora. Para ello se estudiaron cuatro protocolos de comunicación (PWM, PPM, SBUS e IBUS), explicados a continuación:

5.2.1. Protocolo PWM

Este protocolo consiste simplemente en generar una señal PWM a $50\,\mathrm{Hz}$ con un ancho de pulso entre 1000 y $2000\,\mu\mathrm{s}$ por cada uno de los canales. Esto implica que se necesita un GPIO de entrada de la *Raspberry* por cada canal que se desee emplear (mínimo 4 para controlar el dron de forma manual y 2 más para generar órdenes).

Si bien este protocolo es muy sencillo de implementar, el requerir tantas conexiones lo hace muy aparatoso e incómodo de instalar. Todo ello sin contar que el receptor de radio necesario para obtener todas las señales es mayor que los que emplean otros protocolos.

5.2.2. PPM

Una señal PPM (del inglés Pulse Position Modulation) consiste en un tren de pulsos de una misma amplitud separados por una distancia temporal variable. Esto permite modular varios canales en un mismo cable.

En la Figura 5.2 puede observarse cómo se modularían varias señales PWM en una misma señal PPM. Puede denotarse que el tiempo entre dos flancos de subida de la señal PPM coincide con el ancho de pulso de las señales PWM. Tras modular el último canal, debe efectuarse una pausa para indicar el final de la trama. Suele esperarse hasta que se acaba el periodo del PWM equivalente. En el caso de los receptores de radio, lo más normal es que los pulsos estén separados $0.3\,\mathrm{ms}$.

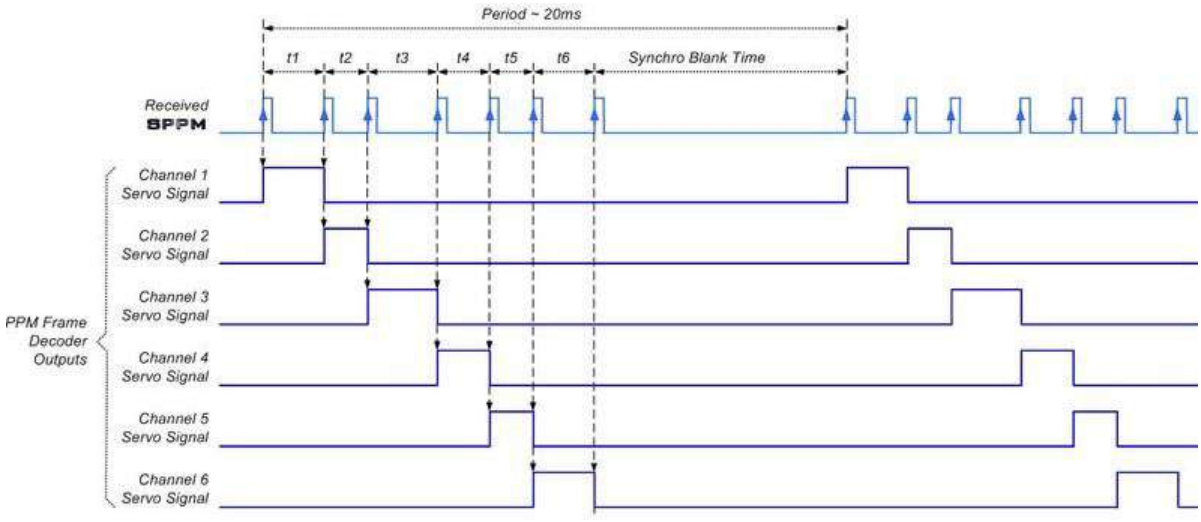


Figura 5.2. Comparación entre una señal PPM y varios PWM.

La empresa *Futaba R/C* emplea su propia forma de modular las señales PPM. El protocolo es en el fondo idéntico al ya explicado, pero envían la señal invertida, tal y como puede observarse en laFigura 5.3.

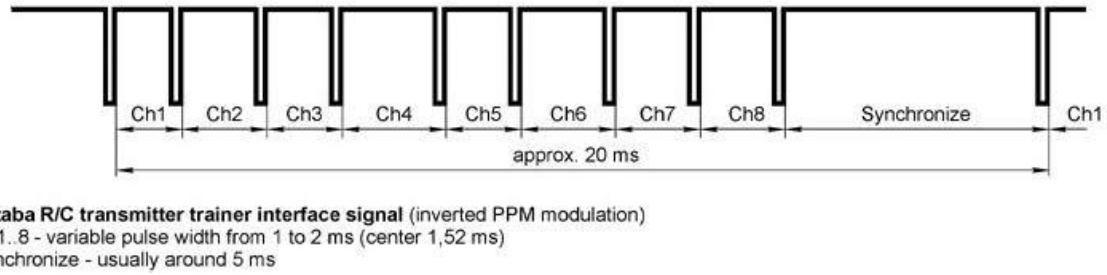


Figura 5.3. Modulación de una señal PPM por Futaba R/C (invertida).

Si bien este protocolo requiere sólo un canal para modular las mismas señales, es necesario o bien leer la entrada digital lo suficientemente rápido como para discernir los flancos de la señal con precisión (polling), o bien implementar interrupciones. Se intentó programar una sección de código para detectarlo por polling, pero ralentizaba demasiado el programa. Debido a que desde *Matlab/Simulink* no existe una forma sencilla de emplear las interrupciones, se descartó usar este protocolo.

5.2.3. SBUS

SBUS es un protocolo de comunicación digital para servos desarrollado por *Futaba R/C*. Está pensado para trabajar directamente con sus servos, pero venden también decodificadores de SBUS a PWM.

Si se investiga algo en los foros, se encuentra que SBUS es un protocolo serie que funciona a $100\,000$ baudios, y codifica 16 canales y 2 señales digitales en 25 bytes. Éstos incluyen un byte de cabecera, uno de 'flags' y canales digitales, y un último byte de cierre.

A partir de ahí, tuvo que efectuarse tanto ingeniería inversa como un estudio exhaustivo de librerías programadas por la comunidad para entender el funcionamiento de SBUS y poder interpretarlo en *Matlab*.

El protocolo presenta una cabecera de 15 (0x0F en hexadecimal). La organización de los datos puede observarse en la Tabla 5.3

A continuación se muestra el código empleado para extraer los datos de los primeros 4 canales:

```
function CH = fcn(u)
%Matlab function to extract Futaba SBUS data
CH = zeros(16,1,'uint16');
STREAM = zeros(22,8);
%Convert the data to binary and flip the matrix
for nn = 1:22
    STREAM(nn,:) = fliplr(dec2bin(u(nn),8));
end
%Select the channels data and convert them to uint16
CH(1) = uint16(bin2dec(fliplr([char(STREAM(1,1:8)) char(STREAM(2,1:3))])));
CH(2) = uint16(bin2dec(fliplr([char(STREAM(2,4:8)) char(STREAM(3,1:6))])));
CH(3) = uint16(bin2dec(fliplr([char(STREAM(3,7:8)) char(STREAM(4,1:8))
    char(STREAM(5,1))])));
CH(4) = uint16(bin2dec(fliplr([char(STREAM(5,2:8)) char(STREAM(6,1:4))])));
```

SBUS	Binary Data								data	
63	0	0	1	1	1	1	1	1		
250	1	1	1	1	1	0	1	0	CH1	010|00111111
164	1	0	1	0	0	1	0	0	CH2	100100|11111
60	0	0	1	1	1	1	0	0	CH3	1|00111100|10
49	0	0	1	1	0	0	0	1	CH4	0000|0011000
176	1	0	1	1	0	0	0	0	CH5	11111001011
252	1	1	1	1	1	1	0	0	CH6	10|11111110|1
254	1	1	1	1	1	1	1	0	CH7	11110|111111
254	1	1	1	1	1	1	1	0	CH8	11111110|111
254	1	1	1	1	1	1	1	0		
254	1	1	1	1	1	1	1	0		
254	1	1	1	1	1	1	1	0	CH9	110|11111110
254	1	1	1	1	1	1	1	0	CH10	111110|11111
254	1	1	1	1	1	1	1	0	CH11	0|11111110|11
254	1	1	1	1	1	1	1	0	CH12	1110|1111111
254	1	1	1	1	1	1	1	0	CH13	1101101|0011
62	0	0	1	1	1	1	1	0	CH14	10|11111100|0
218	1	1	0	1	1	0	1	0	CH15	01000|111111
252	1	1	1	1	1	1	0	0	CH16	01101000|101
254	1	1	1	1	1	1	1	0		
168	1	0	1	0	1	0	0	0		
104	0	1	1	0	1	0	0	0		

Tabla 5.3. Decodificación de la trama SBUS

Tras comprobar el correcto funcionamiento del protocolo con un diagrama de *Simulink*, se procedió a implementarlo en la *Raspberry*. No se consiguió configurar la UART para 100 000 baudios, por lo que este protocolo hubo de abandonarse.

5.2.4. IBUS

IBUS es otro protocolo de comunicación serie, esta vez de *FlySky*. Funciona a 115 200 baudios, con una cabecera de dos bytes (0x20, 0x40). Emplea dos bytes por cada canal de radio, enviando, para cada canal, primero el byte menos significativo y luego el más significativo.

El protocolo IBUS se hizo mucho más sencillo de implementar, en comparación con el SBUS; tanto en un diagrama de *Simulink* como en la *Raspberry*.

5.2.5. Comparativa entre los protocolos

En la Tabla 5.5 se muestra una comparación de los cuatro protocolos analizados. A pesar de la limitación del número de emisoras compatibles, se empleó el protocolo IBUS por su facilidad de implantación.

CHn	:	Canal 'n'
Bm(a:b)	:	bits del 'a' al 'b' del byte 'm' (SBUS)
CH0	=	[B1(2:0), B0(7:0)]
CH1	=	[B2(5:0), B1(7:3)]
CH2	=	[B4(0), B3(7:0), B2(7:6)]
CH3	=	[B5(3:0), B4(7:1)]
CH4	=	[B6(6:0), B5(7:4)]
CH5	=	[B8(1:0), B7(7:0), B6(7)]
CH6	=	[B9(4:0), B8(7:2)]
CH7	=	[B10(7:0), B9(7:5)]
CH8	=	[B12(2:0), B11(7:0)]
CH9	=	[B13(5:0), B12(7:3)]
CH10	=	[B15(0), B14(7:0), B13(7:6)]
CH11	=	[B16(3:0), B15(7:1)]
CH12	=	[B17(6:0), B16(7:4)]
CH13	=	[B19(1:0), B18(7:0), B17(7)]
CH14	=	[B20(4:0), B19(7:2)]
CH15	=	[B21(7:0), B20(7:5)]

Tabla 5.4. Organización de los datos en el protocolo SBUS

RC protocol	Ventajas	Inconvenientes
PWM	Protocolo muy simple Muy extendido	Requiere muchas entradas del GPIO
PPM	Una sola entrada Relativamente sencillo Muy extendido	Requiere interrupciones Difícil implantación desde Simulink
SBUS	Protocolo serie 16 canales Muy extendido	$100\,000$ baudios, la Pi no es capaz de leerlo
IBUS	Protocolo serie 10 canales $115\,200$ baudios, la Pi puede leerlo por la UART	Pocas emisoras soportan este protocolo

Tabla 5.5. Comparativa entre los protocolos de comunicación con el receptor de radio

6

Resultados

En la Sección 6.1 se muestran los resultados obtenidos en vuelo manual y autónomo. Por otro lado, en la Capítulo 7 se comentan los futuros desarrollos propuestos para la continuidad de este proyecto.

6.1. Resultados

6.1.1. Vuelo manual

En cuanto al vuelo con emisora, se han conseguido efectuar vuelos controlados. En las siguientes figuras se puede apreciar la evolución de los ángulos y velocidades de Euler frente a sus referencias durante varias pruebas de vuelo.

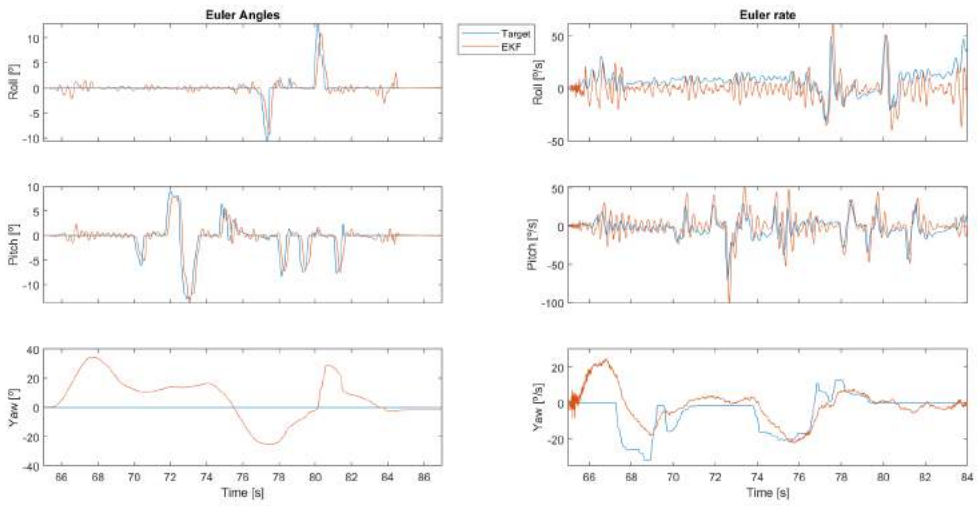


Figura 6.1. Control de estabilización. Vuelo 1.

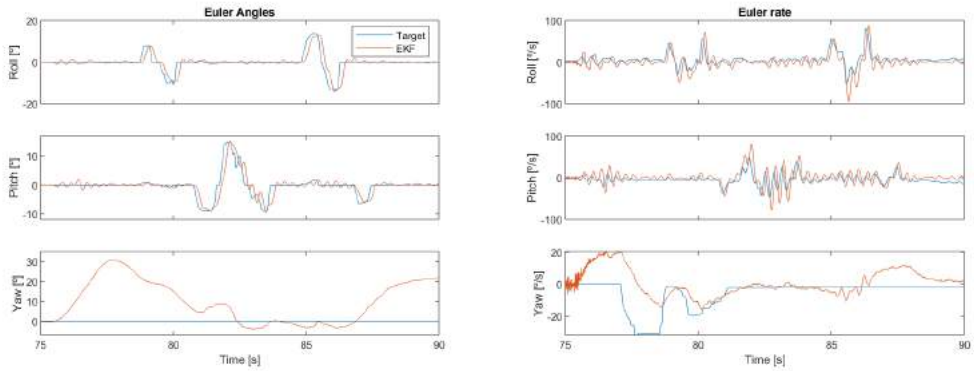


Figura 6.2. Control de estabilización. Vuelo 2.

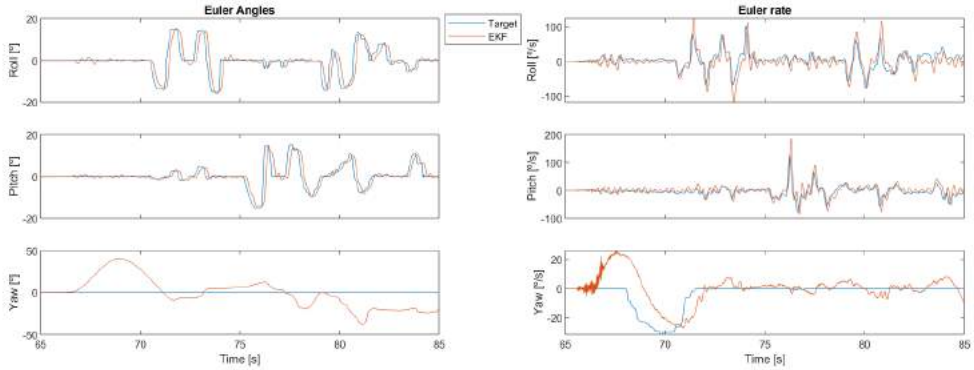


Figura 6.3. Control de estabilización. Vuelo 3.

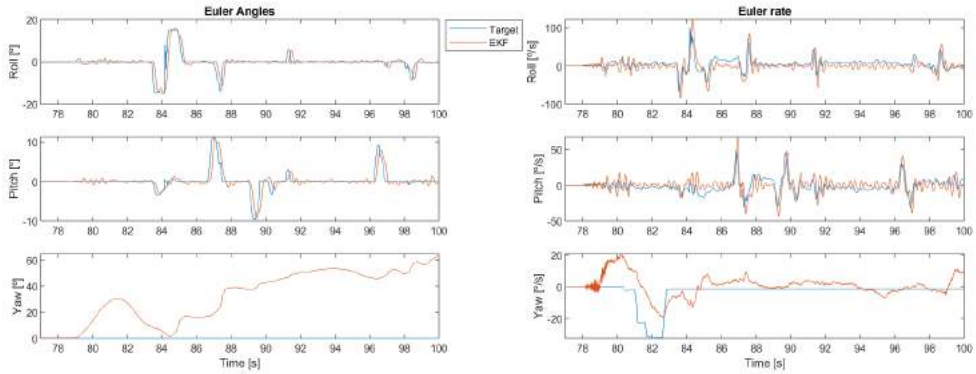


Figura 6.4. Control de estabilización. Vuelo 4.

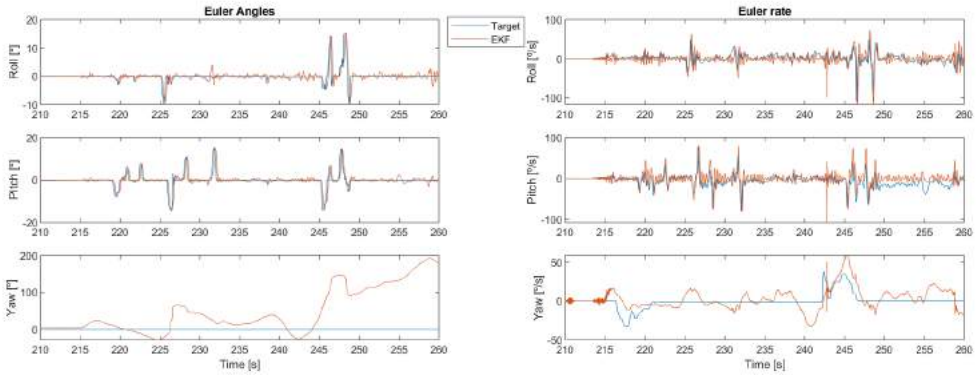


Figura 6.5. Control de estabilización. Vuelo 5.

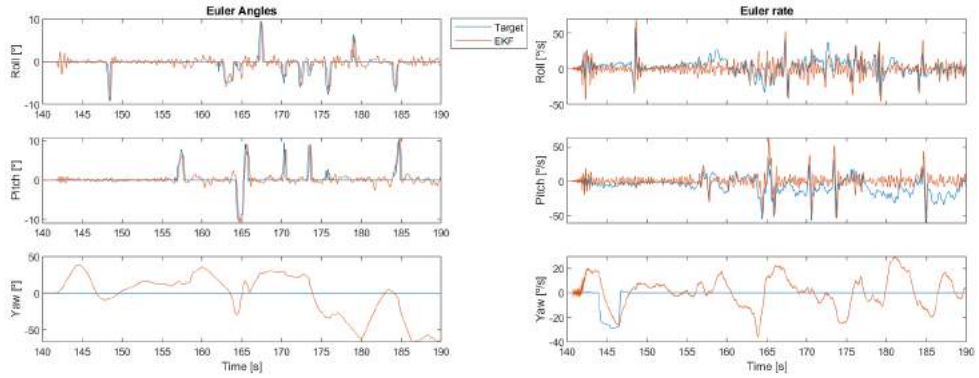


Figura 6.6. Control de estabilización. Vuelo 6.

Puede observarse que, si bien los controles de ángulo (salvo la guiñada, que no va controlada en vuelo manual) funcionan perfectamente, la velocidad de Euler es muy ruidosa. Esto es debido a la vibración natural del cuadricóptero, y haría falta un filtro digital de menor frecuencia (y ralentizar el control), un amortiguador mecánico con mejores propiedades para mitigarlo o, en su defecto, reducir todavía más el tiempo de muestreo del control y el de actuación de los motores para poder compensar dicha vibración.

Por otro lado, se habría de perfeccionar el control de guiñada ya que, si bien se es capaz de volar el dron corrigiéndola desde la emisora de una forma relativamente fácil, puede resultar incómodo debido a la lentitud del control.

6.1.2. Vuelo autónomo

En cuanto al vuelo autónomo, no se ha llegado a implementar en físico debido a los retardos en la transmisión de los datos del MCS (Figura 6.7). Este retardo es de unos $300\,\mathrm{ms}$, unos 30 periodos de muestreo, haciendo prácticamente inviable cerrar el lazo.

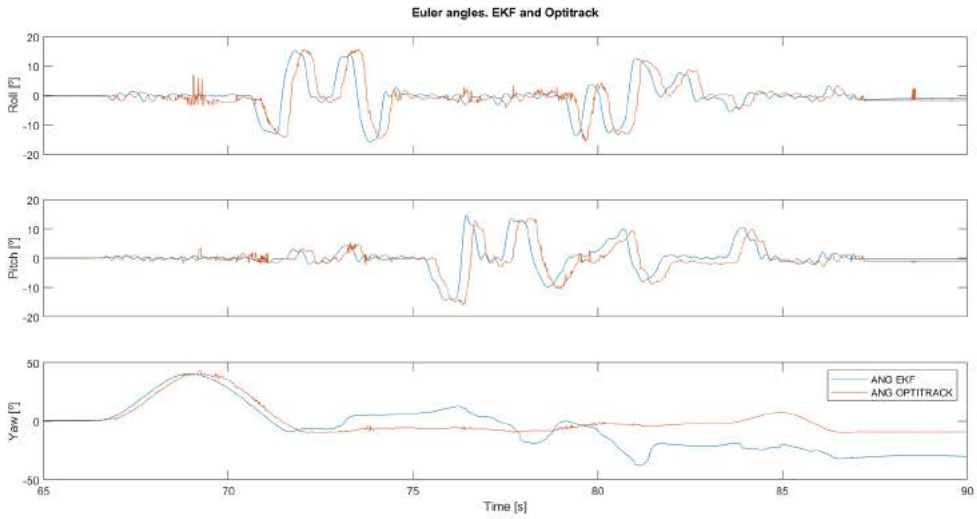


Figura 6.7. Comparativa entre los ángulos de Euler del EKF y del MCS.

En la Figura 6.8 se pueden observar las posiciones en el espacio captadas por *Motive* durante una de las pruebas de vuelo manual. Se denota que el sistema no pierde de vista al dron en la zona en la que se ha trabajado. Ésto implica que, si se pudiese eliminar el retardo en la comunicación, sería factible cerrar el lazo con este sistema.

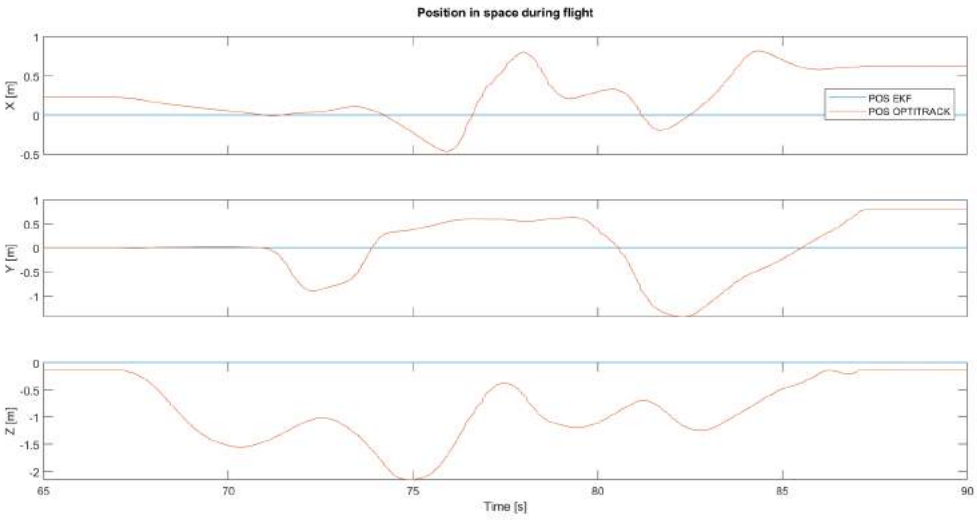


Figura 6.8. Comparativa entre los ángulos de Euler del EKF y del MCS.

En la Figura 6.9 se muestra una comparativa entre los ángulos de Euler estimados con el EKF y los obtenidos mediante el sistema de cámaras de Optitrack al comunicar la GCS por Bluetooth. Puede observarse que el retardo prácticamente desaparece, pero los ángulos los obtenidos con Motive son ligeramente mayores. Ésto puede deberse, entre otros, a la cantidad de filtrado que se ha debido de incorporar a las medidas de la IMU para eliminar el ruido inherente del dron. No pudo llegar a implantarse este sistema debido a la escasez de puertos serie de la *Raspberry*.

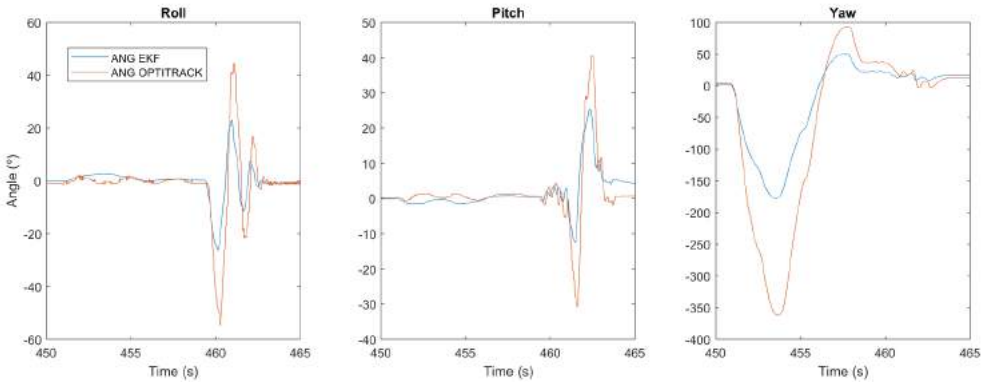


Figura 6.9. Comparativa entre los ángulos de Euler del EKF y del MCS.

7

Conclusiones y futuros desarrollos

En este capítulo se efectúa una conclusión general del proyecto y se enumeran posibles futuros desarrollos. Como futuros desarrollos, se va a enfocar la mira em los problemas no resueltos que se han detectado en este proyecto

7.1. Conclusiones

En este proyecto se ha conseguido implantar una estructura de control de vuelo en una *Raspberry Pi Zero W* programada desde *Matlab/Simulink*. Para ello, se han tenido que programar tanto la gestión de las comunicaciones como la lectura y tratamiento de datos de los sensores.

En cuanto a consecución de resultados, se ha obtenido un control de vuelo robusto que ha permitido a un piloto no experto efectuar varias pruebas en modo manual. Además, se ha optimizado el EKF para el control de cabeceo y guiñada, obteniendo ángulos muy similares a los captados con le *MCS* simplemente con las medidas de la IMU.

7.2. Futuros desarrollos

1. Por un lado, está la incompatibilidad entre la comunicación MAVLink via Bluetooth y la recepción de los datos de la emisora. Para solventarlo, se propone o bien emplear la miniuart para recibir el protocolo IBUS por el GPIO y conectar el Bluetooth directamente al de la Raspberry, o bien implementar un código en C que permita manejar interrupciones desde Matlab.

2. Por otro lado, se debería de efectuar un ajuste fino al control de estabilidad, a pesar de que funcione lo suficientemente bien como para que un piloto no experto sea capaz de manejar la aeronave con el control actual.

Una vez se implementasen estos desarrollos, el dron ya debería de estar suficientemente capacitado como para volar de forma completamente autónoma, y se podría enfocar los esfuerzos en darle utilidad. Por ejemplo, como parte lectiva de las asignaturas de control de la universidad, se podrían implementar las siguientes prácticas:

- Control de un péndulo invertido situado encima del cuadricóptero.
- Minimizar el error de seguimiento de trayectorias predefinidas.
- Contrarreloj en un circuito con obstáculos previamente conocido.

References

[ATD14] L. Apvrille, T. Tanzi, and J. L. Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. In *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, pages 1–4, Aug 2014.

[BHCL16a] S. M. Bae, K. H. Han, C. N. Cha, and H. Y. Lee. Development of inventory checking system based on uav and rfid in open storage yard. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–2, Dec 2016.

[BHCL16b] Sung Moon Bae, Kwan Hee Han, Chun Nam Cha, and Hwa Yong Lee. Development of inventory checking system based on uav and rfid in open storage yard. In *International conference on Information Science and Security*. IEEE, 2016.

[CMBH16] Anjan Chakrabarty, Robert Morris, Xavier Bouyssounouse, and Rusty Hunt. Autonomous indoor object tracking with the parrot ar.drone. In *International Conference on WeATT1.4 Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016.

[Dij12] Nick Dijkshoorn. Simultaneous localization and mapping with the ar. drone. *PhD diss., Masters thesis, Universiteit van Amsterdam*, 2012.

[Gon16] Nestor González. Control de un cuadricóptero para navegación en interiores usando un sensor de flujo óptico. Master's thesis, Escuela Técnica Superior de Ingeniería (ICAI), 2016.

[HGG+16] E. H. C. Harik, F. GuÃ©rin, F. Guinand, J. F. BrethÃ©, and H. Pelvillain. Towards an autonomous warehouse inventory scheme. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.

[HSA+15] Shaima Al Habsi, Maha Shehada, Marwah Abdoon, Ahmed Mashood, and Hassan Noura. Integration of a vicon camera system for indoor flight of a parrot and ar drone. In *10th International Symposium on Mechatronics and its Applications (ISMA)*, 2015.

[KK18] P. Kaniewski and T. Kraszewski. Drone-based system for localization of people inside buildings. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pages 46–51, Feb 2018.

[LLZH14] R. Li, J. Liu, L. Zhang, and Y. Hang. Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments. In *2014 DGON Inertial Sensors and Systems (ISS)*, pages 1–15, Sept 2014.

[Men17] Diego Menéndez. Desarrollo de un sistema de navegación autónoma en interiores para un cuadricóptero. Master's thesis, Escuela Técnica Superior de Ingeniería (ICAI), 2017.

[PHSA17] James A Preiss, Wolfgang Honig, Gaurav S Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3299–3304. IEEE, 2017.

[SQ17] Laura Sánchez-Quiñones. Control de un cuadricóptero para seguimiento automático de personas. Master's thesis, Escuela Técnica Superior de Ingeniería (ICAI), 2017.

[TW17] J. Tiemann and C. Wietfeld. Scalable and precise multi-uav indoor navigation using tdoa-based uwb localization. In *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, Sept 2017.

[XKM18] Lichao Xu, Vineet R Kamat, and Carol C Menassa. Automatic extraction of 1d barcodes from video scans for drone-assisted inventory management in warehousing applications. *International Journal of Logistics Research and Applications*, 21(3):243–258, 2018.

[YCF+17] F. Yu, G. Chen, N. Fan, Y. Song, and L. Zhu. Autonomous flight control law for an indoor uav quadrotor. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 6767–6771, May 2017.

[ZWC+16] Yu Zhang, Tingting Wang, Zhihao Cai, Yingxun Wang, and Zhenxing You. The use of optical flow for uav motion estimation in indoor environment. In *Proceedings of 2016 IEEE Chinese Guidance, Navigation and Control Conference August 12-14, 2016 Nanjing, China*. IEEE, 2016.

PARTE II

PRESUPUESTO

ELementos y precios unitarios

Elemento	Coste unitario [€]
Dron	
Raspberri Pi Zero W	11
PXFmini	69
Motor mt2204	7.75
ESC DYSSN16	9.34
Hélices 3x5030	0.20
Receptor FS-A8S	7.47
Batería LiPo	5.66
Emisora Turnigy i10	238.53
MCS	
Cámara Flex 13	999
OptiHub	299
Licencia Motive Tracker	999
CW-500	299
CS-200	149
Software - Licencias	
Matlab / Simulink	800
Horas de trabajo	
Programación	10
Construcción y mantenimiento del dron	8
Pruebas de comunicaciones	9
Pruebas de vuelo	8
Depuración del ekf y control	10

Tabla 1. Elementos empleados y precios unitarios

Cantidad de cada elemento

Elemento	unidaeds [€]
Dron	
Raspberri Pi Zero W	1
PXFmini	1
Motor mt2204	4
ESC DYSSN16	4
Hélices 3x5030	4
Receptor FS-A8S	1
Batería LiPo	1
Emisora Turnigy i10	1
MCS	
Cámara Flex 13	8
OptiHub	3
Licencia Motive Tracker	999
CW-500	1
CS-200	1
Software - Licencias	
Matlab / Simulink	1
Horas de trabajo	
Programación	200
Construcción y mantenimiento del dron	50
Pruebas de comunicaciones	50
Pruebas de vuelo	40
Depuración del ekf y control	50

Tabla 2. Cantidad de elementos empleados

Presupuesto final

Elemento	Coste total [€]
Dron	
Raspberri Pi Zero W	11
PXFmini	69
Motor mt2204	31
ESC DYSSN16	37.36
Hélices 3x5030	0.80
Receptor FS-A8S	7.47
Batería LiPo	5.66
Emisora Turnigy i10	238.53
MCS	
Cámara Flex 13	7992
OptiHub	897
Licencia Motive Tracker	999
CW-500	299
CS-200	149
Software - Licencias	
Matlab / Simulink	800
Horas de trabajo	
Programación	2000
Construcción y mantenimiento del dron	400
Pruebas de comunicaciones	450
Pruebas de vuelo	320
Depuración del ekf y control	500
Presupuesto total	15 206.82€

Tabla 3. Elementos empleados y precios unitarios