COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES (GITI)

Especialidad Electrónica

TRABAJO DE FIN DE GRADO

CONTROL DE NAVEGACIÓN AUTÓNOMA DE UN CUADRICÓPTERO EN INTERIORES

Autor: Jorge Jacobo Bennasar Vázquez

Director: Juan Luis Zamora Macho

Co-Director: Javier García Aguilar

Madrid

Junio de 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. Jorge Jacobo Bennasar Vázquez DECLARA ser el titular de los derechos de propiedad intelectual de la obra: *Control de navegación autónoma de un cuadricóptero en interiores*, 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....7.....de.....junio.....de....2019

ACEPTA

Fdo.....Jorge Bennasar.....

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

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título "Control de navegación autónoma de un cuadricóptero en interiores" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018 / 2019 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.: BENNASAR VÁZQUEZ, JORGE JACOBO Fecha: 07/06/2019

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: ZAMORA MACHO, JUAN LUIS Fecha: 7/6/2019

Fdo.: GARCÍA AGUILAR, JAVIER Fecha: 7/6/2019

COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES (GITI)

Especialidad Electrónica

TRABAJO DE FIN DE GRADO

CONTROL DE NAVEGACIÓN AUTÓNOMA DE UN CUADRICÓPTERO EN INTERIORES

Autor: Jorge Jacobo Bennasar Vázquez

Director: Juan Luis Zamora Macho

Co-Director: Javier García Aguilar

Madrid

Junio de 2018

Resumen

Este trabajo de fin de grado, siendo una continuación del proyecto *"Desarrollo de sistemas de navegación autónoma para un UAV"* [*], trata de mejorar el vuelo manual de drones logrado anteriormente y desarrollar e implementar toda la tecnología necesaria para la navegación autónoma. Para la programación se ha empleado el entorno *MatLab/Simulink (R2018b)*, mientras que para el hardware se ha empleado una *Raspberry Pi Zero W* acompañada de otros dispositivos, como el *OpenPilot Revolution* o el *PXFmini*, que han ido variando a lo largo del proyecto. Como sensores se han utilizado una Unidad de Medida Inercial (IMU) (incorporada en el *OpenPilot* y en el *PXFmini*) para medir aceleraciones lineales y velocidades angulares, un sonar *MB1240 XL-MaxSonar-EZ4* para obtener la altura del UAV y un sistema de cámaras de *OptiTrack* para hallar los ángulos de Euler y la posición XYZ del dron. Para obtener las mejores estimaciones para el control se han introducido todas las medidas en un Filtro Extendido de Kalman (EKF).

Introducción

La revolución digital de este último siglo ha permitido el desarrollo de múltiples tecnologías prometedoras. Un ejemplo característico es el de los drones autónomos, vehículos aéreos no tripulados que cada día tienen nuevas aplicaciones. La mayoría de estos usos, sin embargo, son en zonas de exteriores, donde se puede acceder a la tecnología GPS. Para lograr un vuelo autónomo fiable en zonas de interiores es necesario incorporar nuevos sensores, ya sean externos (UWB, MCS…) o acoplados al dron (técnicas SLAM…). La navegación autónoma de drones en espacios de interiores abre un nuevo abanico de aplicaciones, como son el inventariado de almacenes, las operaciones de rescate o las inspecciones internas industriales.

Objetivos

El objetivo fundamental del proyecto es que un dron vuele autónomamente en un espacio de interiores. Para ello, se han de alcanzar los siguientes hitos:

- Revisar y optimizar todo lo realizado anteriormente.
- Lograr un vuelo manual de calidad.
- Incorporar la tecnología necesaria para la navegación autónoma:
    - A bordo del dron.

[*] J. García Aguilar, "Desarrollo de sistemas de navegación autónoma para un UAV," de *Trabajo de Fin de Máster - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2018.

- Externa: sistema de cámaras MCS de *OptiTrack*.

- Lograr la navegación autónoma.

Solución

Como solución se ha empleado un control que puede funcionar tanto en vuelo manual como en vuelo autónomo. Para el vuelo manual se usa un control de estabilización, que traduce las referencias transmitidas desde la emisora en los PWM enviados a los motores. En este caso se emplea como sensor una IMU, que ofrece las medidas de aceleraciones lineales y velocidades angulares. Para el vuelo autónomo se usan, además de una IMU, un sistema de cámaras MCS (de *OptiTrack*) y un sonar. La navegación autónoma cuenta con dos controles en cascada: al control de navegación entran las referencias de posición desde la misión, saliendo de éste las referencias de los ángulos de Euler, que a su vez entran en el control de estabilización. En ambos casos, se hace uso de un Filtro Extendido de Kalman para hallar las mejores estimaciones. El diagrama fundamental de la estructura del proyecto de muestra a continuación (Figura 1):

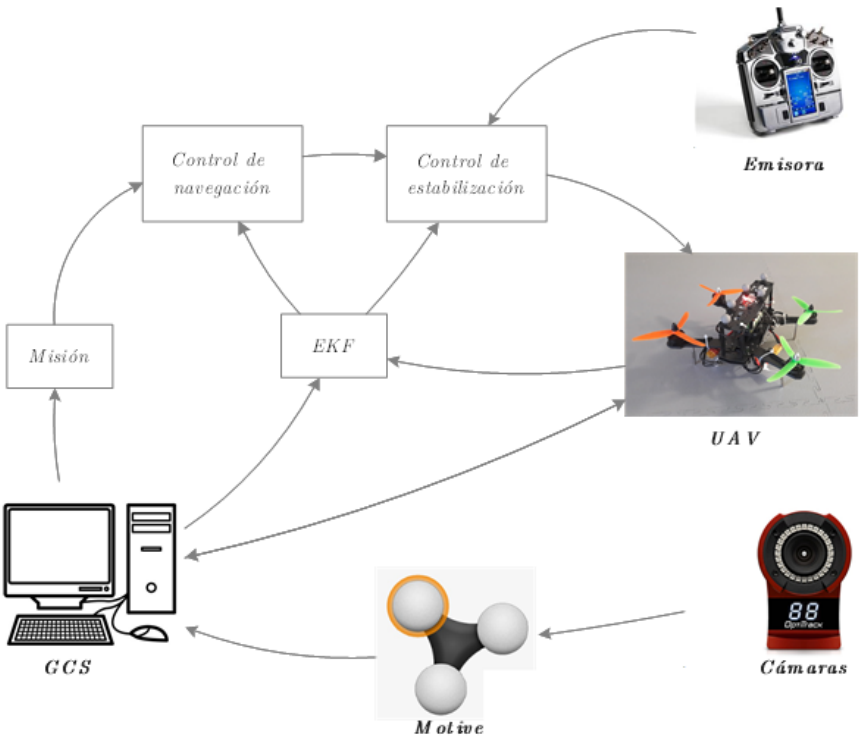

Figura 1. Diagrama fundamental del proyecto

Para realizar todo esto se han empleado distintos software: *MatLab/Simulink* para la programación y la estación de tierra (GCS), *Motive* para el sistema de cámaras y *BLHeli Suite* para el control de los Controladores Electrónicos de Velocidad (ESC). Además, como controladora se ha utilizado una *Raspberry Pi Zero W*, que ha sido acompañada por otros dispositivos, como:

- *PXFmini* de *Erle Robotics*.
- *OpenPilot Revolution*.
- *Waveshare Serial Expansion Hat*.

El resultado es el UAV mostrado a continuación (Figura 2):

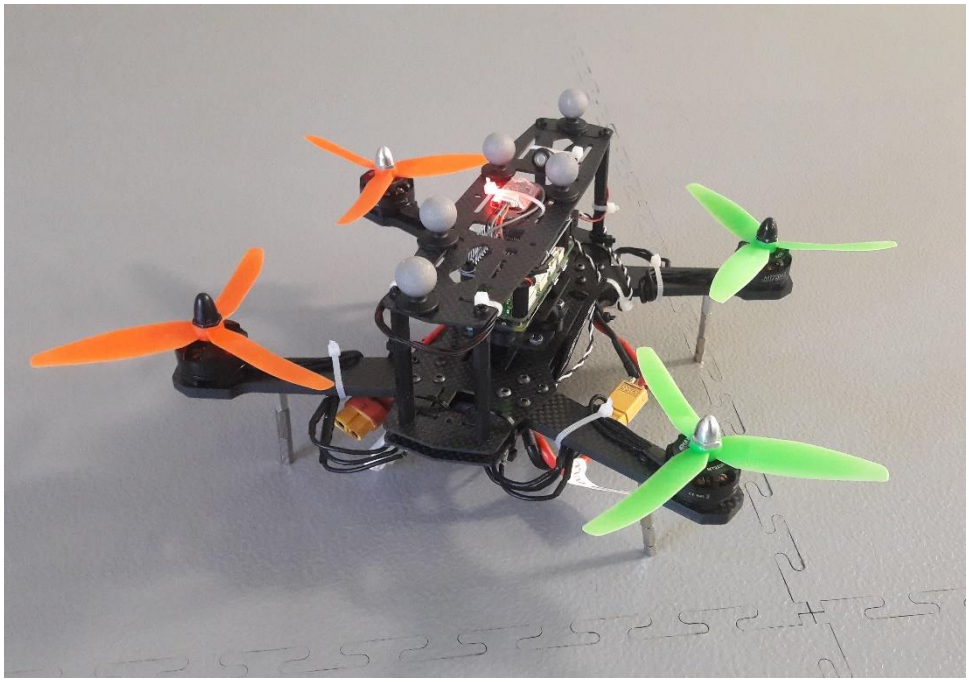

Figura 2. Dron final del proyecto

Resultados

Tras probar varias configuraciones se decidió que la que otorgaba una mayor estabilidad de vuelo era la siguiente:

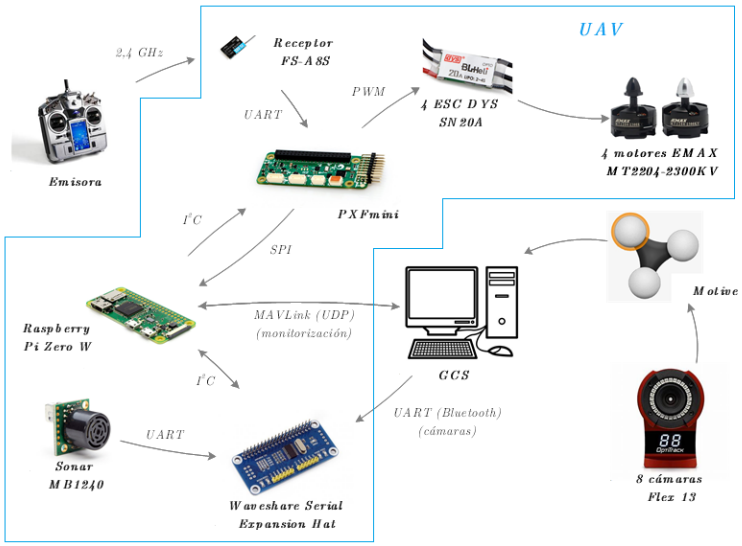

Figura 3. Configuración final de vuelo

Con ella se obtuvieron los siguientes resultados:

- Vuelo manual: resultados satisfactorios tanto en simulación como en vuelo. En la Figura 4 se puede observar la comparación entre las estimaciones del EKF y las medidas de las cámaras de los ángulos de Euler en una prueba de vuelo manual:

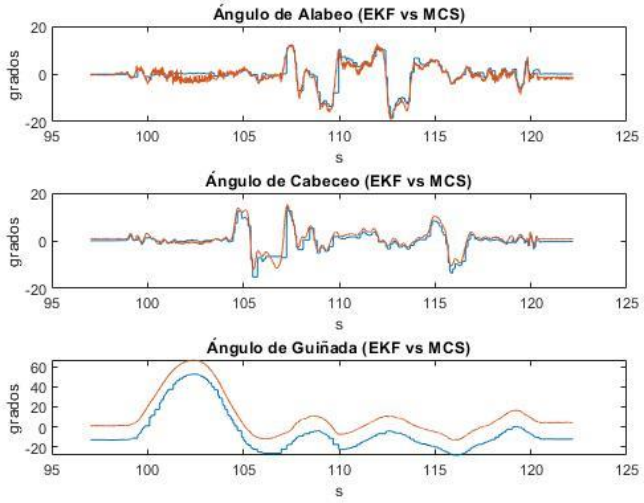

Figura 4. Ángulos de Euler (estimación y cámaras)

- Vuelo autónomo: resultados satisfactorios en simulación. No se consiguió realizar un vuelo autónomo principalmente debido a dos razones: problemas en las comunicaciones y la necesidad de una etapa de calibración de las medidas del sistema MCS que no dio tiempo a completar. En la Figura 5 se puede apreciar el seguimiento a la referencia de posición del dron en una simulación de navegación autónoma:

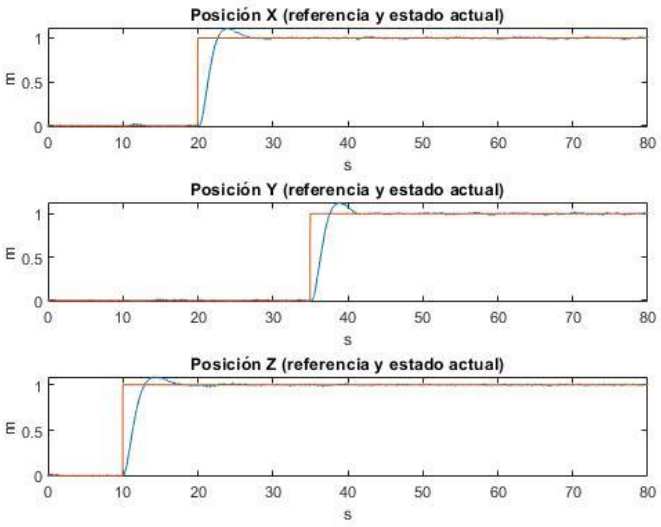

Figura 5. Posición (actual y referencia) del UAV en la simulación

Conclusiones

En este proyecto se han alcanzado una gran cantidad de logros: optimizar el vuelo manual, lograr una simulación de navegación autónoma exitosa, determinar la mejor configuración de hardware en cuento a calidad de vuelo se refiere... Sin embargo, no se ha conseguido que el dron vuele autónomamente. Esto es debido a dos principales razones:

- La posible incompatibilidad entre el *PXFmini*, encargado de obtener las medidas de la IMU y de mandar los PWM a los ESC, y el *Waveshare Serial Expansion Hat*, cuya misión era habilitar suficientes puertos UART para la navegación autónoma.

- La necesidad de una etapa de calibración de las medidas de las cámaras.

Si ambos escollos son superados, y con todo lo realizado en el proyecto, el dron será capaz de volar autónomamente.

Abstract

This project, being a continuation of *"Desarrollo de sistemas de navegación autónoma para un UAV"* [*], aims to improve the previously achieved manual flight of a drone and to develop and implement all the necessary technology for autonomous navigation. *MatLab/Simulink (R2018b)* has been the software used for programming, while a *Raspberry Pi Zero W* has been employed as the main controller. Other hardware components, such as *OpenPilot Revolution* and *PXFmini*, were also used. Sensor-wise, an Inertial Measurement Unit (IMU) has been employed, accompanied by an *OptiTrack* motion capture system and a sonar. To obtain the best estimations for the control, all the data has been introduced in an Extended Kalman Filter (EKF).

Introduction

The digital revolution of the last century has driven the development of multiple promising technologies. A clear example are autonomous drones, unmanned aerial vehicles that have new applications every day. Most of this uses, however, are in open air, where GPS technology is accessible. To achieve reliable indoor autonomous flight it is necessary to incorporate new sensors, external (UWB, MCS…) or integrated in the UAV (SLAM techniques). The indoor autonomous flight of drones enables a wide variety of new applications, such as warehouse inventory, rescue operations and industrial internal inspections.

Objectives

The main objective of the project is to achieve indoors autonomous navigation of a drone. For it, several milestones have to be reached:

- Revise and optimize all the previous work.
- Achieve a quality manual flight.
- Incorporate the necessary technology for autonomous navigation:
    - Integrated in the UAV.
    - External: motion capture system of *OptiTrack*.
- Achieve autonomous navigation.

---

[*]J. García Aguilar, "Desarrollo de sistemas de navegación autónoma para un UAV," de *Trabajo de Fin de Máster - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2018.

Solution

As a solution, a control capable of working in both manual and autonomous flight has been developed. For manual flight the attitude control is used. This control translates the information sent from the remote control into the PWM sent to the motors. Sensor-wise, an IMU, which gives information about linear accelerations and angular velocities, is used. For autonomous navigation, besides the IMU, a motion capture system (*OptiTrack*) and a sonar are used. The autonomous flight has two controls in cascade: the navigation and the attitude control. The navigation control receives the position references from the mission and sends the Euler angles references to the attitude control. In both controls an Extended Kalman Filter is used to improve estimations. The main diagram of the project is shown below (Figure 1):

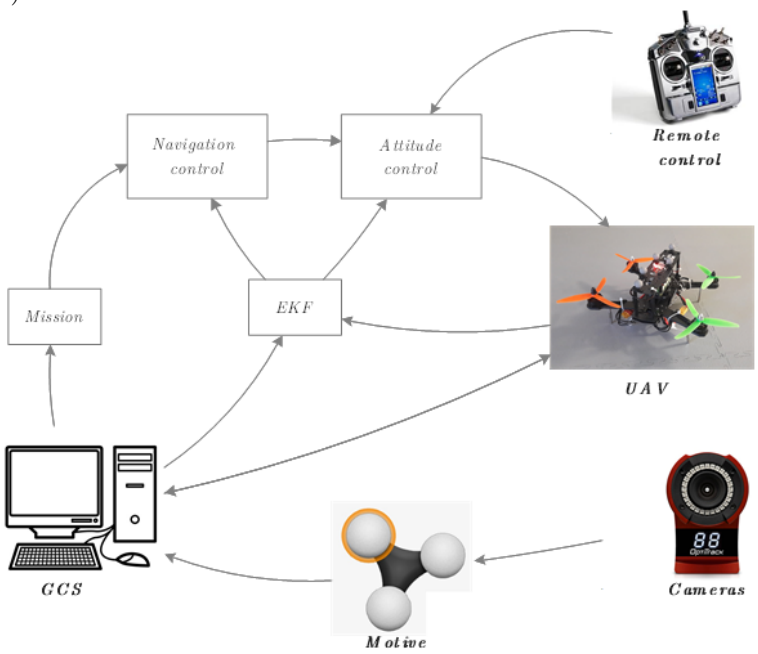

Figure 1. Project main diagram

To achieve all of this, different programs have been used: *MatLab/Simulink* for the programming and the ground control station (GCS), *Motive* for the motion capture system and *BLHeli Suite* for the Electronic Speed Controllers (ESC). The main controller was a Raspberry Pi Zero W, while other hardware components were:

- *PXFmini* (*Erle Robotics*).
- *OpenPilot Revolution.*
- *Waveshare Serial Expansion Hat.*

The result is the UAV shown below (Figure 2):

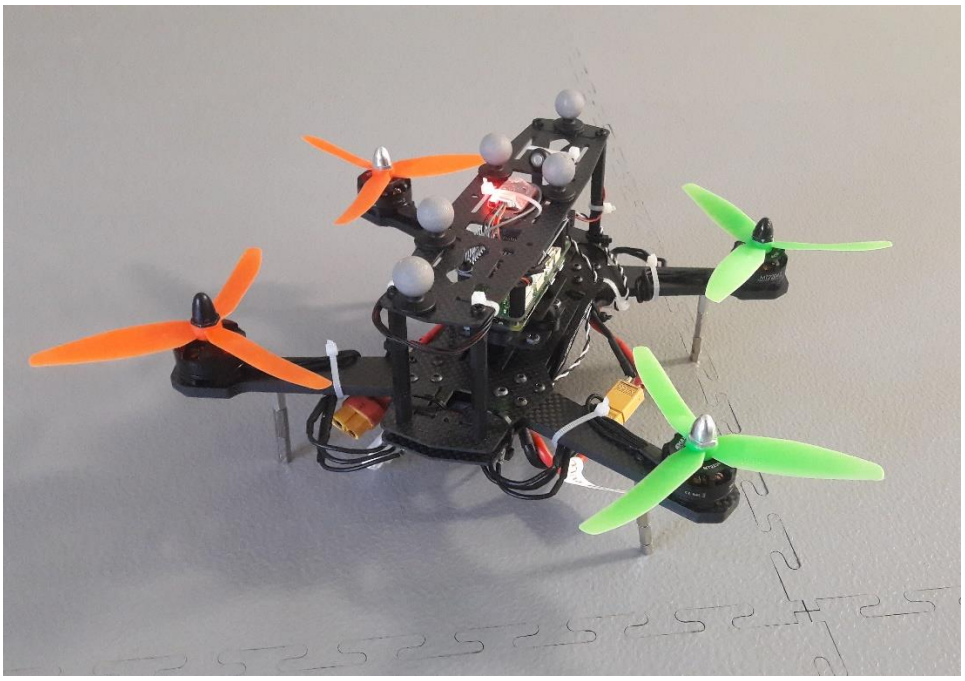

Figure 2. Final UAV

Results

After testing several configurations, the disposition shown below (Figure 3) was the one that offered the best stability inflight:

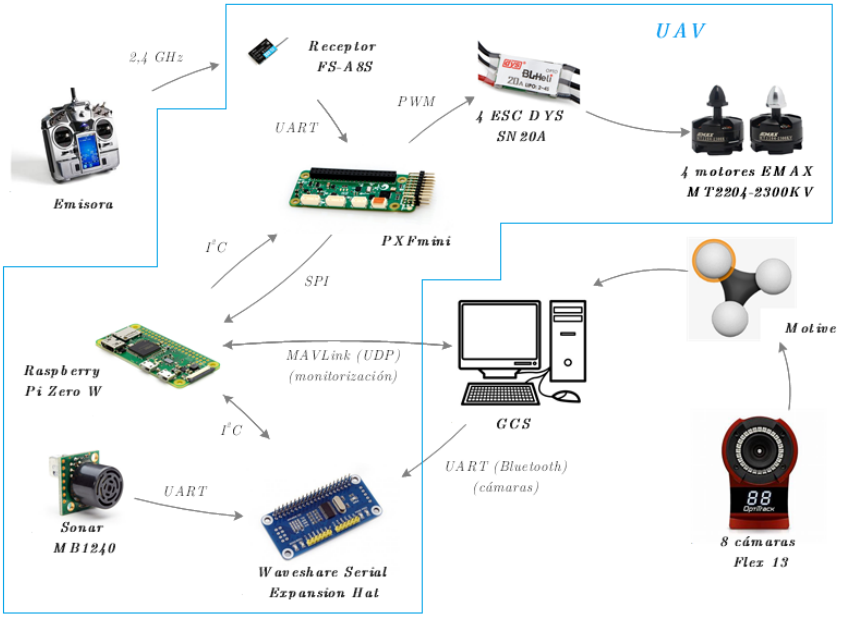

Figure 3. Final flight configuration

With this configuration, the results achieved were:

- Manual flight: satisfactory results in simulation and flight. In Figure 4 it is shown the comparison between the EKF estimations and the data received from the cameras regarding the Euler angles of the UAV during a manual flight:

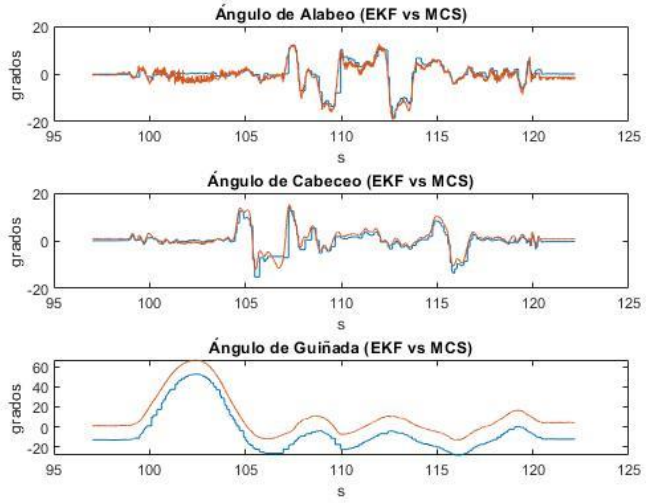

Figure 4. Euler angles (estimation and cameras)

- Autonomous flight: satisfactory results in simulation. A reliable autonomous flight was not achieved due to two main reasons: communication problems and the necessity of a calibration stage for the data received from the motion capture system. In Figure 5 it is shown the tracking of the position reference of the drone in a simulation of autonomous navigation:

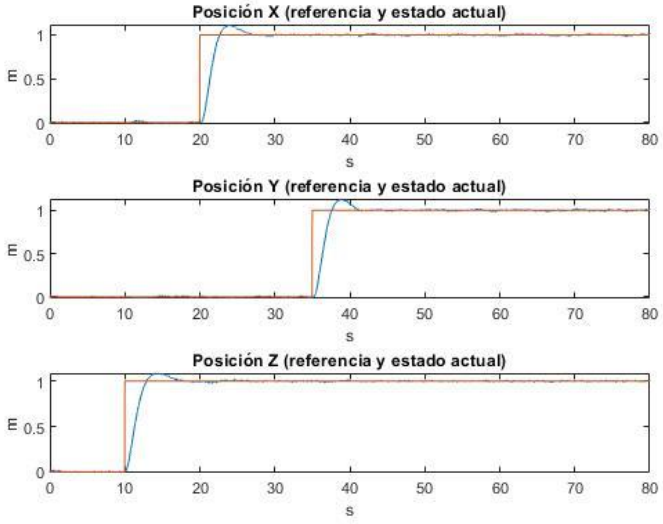

Figure 5. Position (simulated and reference) of the UAV

Conclusions

In this project, several achievements have been reached: the optimization of manual flight, successful simulation of autonomous navigation, determination of the best hardware configuration in terms of flight quality… However, the UAV has not flown autonomously. This is due to two main problems:

- The possible incompatibility between the *PXFmini* and the *Waveshare Serial Expansion Hat*.
- The necessity of a calibration stage for the motion capture system data.

If both complications are solved, the drone will finally fly autonomously.

Índice general

Índice de figuras

Índice de tablas

Siglas

BLDC: Brushless DC (corriente continua sin escobillas)
EKF: Extended Kalman Filter (Filtro Extendido de Kalman)
ESC: Electronic Speed Controller (Controlador Electrónico de Velocidad)
GCS: Ground Control Station (estación de control de tierra)
IMU: Inertial Measurement Unit (Unidad de Medida Inercial)
IR: Infrarrojos
PD: Proporcional-Derivativo
PI: Proporcional-Integral
PID: Proporcional-Integral-Derivativo
PWM: Pulse Width Modulation (modulación de ancho de pulso)
XYZ: Ejes de coordenadas

PARTE I:

MEMORIA DESCRIPTIVA

1. Introducción

La gran revolución digital de las últimas décadas ha permitido el desarrollo acelerado de nuevas tecnologías en múltiples ámbitos. Se ha conseguido aumentar mucho las prestaciones de los microprocesadores, siendo cada vez más potentes y rápidos. Todo esto ha abierto una enorme ventana de posibilidades en el campo del control. Además, debido a la mejora de sus capacidades, los microcontroladores cada vez pueden ser más pequeños, lo que nos permite darles aplicaciones que hace poco tiempo no hubiéramos imaginado. Una de ellas es la de los UAVs (Unmanned Aerial Vehicles), también conocidos como drones, que son vehículos aéreos con unas dimensiones suficientemente pequeñas como para poder volar en interiores.

Los UAVs han provocado una revolución debido a su gran versatilidad. Actualmente ya tienen bastantes aplicaciones (fotografía, video y cartografía aérea, aplicaciones militares, exploración de lugares de difícil acceso, agricultura…), pero se prevé que en un futuro cercano tendrán muchas más. El abanico de posibilidades, además, crece mucho si consideramos la posibilidad de que vuelen de manera autónoma. En exteriores esto se suele conseguir con la ayuda de localización GPS (Global Positioning System), pero en interiores esto no es posible. Es por ello que el vuelo autónomo de UAVs en zonas con un espacio restringido ha supuesto un verdadero reto en los últimos tiempos. Es necesario utilizar múltiples sensores, ya sean externos o incorporados al propio dron, con el fin de establecer un sistema de posicionamiento suficientemente preciso como para que la navegación tenga éxito.

En este proyecto, que se sustenta en otros realizados anteriormente en ICAI (véanse [1], [2], [3] y [4]), se ha implementado un control de vuelo autónomo en interiores para un cuadricóptero (UAV con cuatro rotores). Empleando un sistema MCS (Motion Capture System) compuesto por ocho cámaras *Flex 13* de *OptiTrack* se ha recogido información acerca de la localización del dron para, a través del software *Motive*, enviarla a la estación de tierra (GCS) y, posteriormente, al microcontrolador (*Raspberry Pi Zero W*). Además, se han utilizado el módulo *PXFmini* de *Erle Robotics* y la controladora de vuelo *OpenPilot Revolution*. Ambos incorporan una unidad de medida inercial (IMU) que proporciona información acerca de las aceleraciones y velocidades angulares del UAV. Por otro lado, también se han empleado un sonar *MB1240 XL-MaxSonar-EZ4* de *MaxBotix* para medir la altura del dron y un *Waveshare Serial Expansion Hat* para *Raspberry* para tener disponibles dos canales UART adicionales. Con todas las medidas recogidas, y haciendo uso de un filtro extendido de Kalman (EKF), el estimador de estado por excelencia en cuento a sistemas no lineales se refiere, se ha tratado de obtener una estimación precisa de las variables necesarias para la actuación exitosa del control y, en consecuencia, de la

navegación autónoma. Todo esto se ha realizado con la ayuda del entorno *MatLab/Simulink (R2018b)*, en el que existe soporte para la *Raspberry Pi Zero W* y para el *OpenPilot Revolution*, gracias al *Waijung Blockset*.

1.1. Motivación

Los UAVs son una tecnología con un enorme potencial, y la posibilidad de que vuelen de forma totalmente autónoma los convierte en actores principales de la gran revolución tecnológica que está ocurriendo en el siglo XXI. La principal motivación de este proyecto es crear un sistema fiable de navegación autónoma para drones haciendo uso de un sistema de cámaras MCS con el objetivo de que otros proyectos puedan centrarse en darle aplicaciones. Además, este proyecto requiere de un conocimiento integral de lo estudiado en una ingeniería, pues es necesario saber desenvolverse en áreas tan diversas como pueden ser el control digital, la mecánica, la electrónica o la aerodinámica. Que sea un proyecto tan completo es otro aliciente para su realización.

1.2. Objetivos

El objetivo final del proyecto era conseguir que un cuadricóptero vuele de forma autónoma en una zona de interiores. Para lograrlo se tuvieron que conseguir los siguientes hitos:

- Comprobar el funcionamiento y optimizar todo lo realizado en proyectos anteriores:
  - Unidad de medida inercial (IMU).
  - Comunicaciones (*MAVLink* y emisora).
  - Filtro Extendido de Kalman (EKF).
  - Controles de vuelo y modelado matemático.
- Lograr que el cuadricóptero vuele correctamente de forma manual.
- Incorporar el hardware, software y comunicaciones necesarias para hacer posible la navegación autónoma.
- Lograr que el dron vuele autónomamente.

1.3. Metodología

La metodología seguida en el proyecto ha sido la siguiente:

1. Configuración de la *Raspberry* para que fuera compatible con el entorno *MatLab/Simulink* y cumpliera todos los requisitos del proyecto.

2. Estudio de lo realizado en proyectos anteriores y comprobación de su funcionamiento:

    2.1. Unidad de medida inercial (IMU).
    2.2. Comunicaciones (*MAVLink* y emisora).
    2.3. Controles de vuelo y modelado matemático.
    2.4. Filtro Extendido de Kalman (EKF).

3. Navegación manual:

    3.1. Vuelo manual.
    3.2. Optimización de vuelo.

4. Navegación autónoma:

    4.1. Prueba del sistema MCS.
    4.2. Incorporación de alternativas para la navegación autónoma.
    4.3. Vuelo autónomo.
    4.4. Optimización de vuelo.

5. Memoria.

1.4. Recursos

1.4.1. Software

Los recursos software empleados en este proyecto son los mostrados a continuación:

MatLab/Simulink (R2018b): se utilizó *MatLab/Simulink* como herramienta para programar la *Raspberry Pi Zero W* y el *OpenPilot Revolution*. Además, se empleó para implementar la estación de tierra (GCS).

PuTTY: *PuTTY* se empleó para conectarse de forma remota a la *Raspberry Pi Zero W* por *SSH*.

Motive: *Motive* es una aplicación de *OptiTrack* que se empleó para recoger y analizar los datos obtenidos por las cámaras *Flex 13* y enviarlos a la GCS.

BLHeli Suite: la aplicación *BLHeli Suite* se usó para la programación de los ESC.

1.4.2. Hardware

Por otro lado, los recursos hardware que se emplearon son:

Raspberry Pi Zero W: como microcontrolador se empleó una *Raspberry Pi Zero W* debido a su buena relación potencia-tamaño con respecto a otras opciones.

Módulo PXFmini: el módulo *PXFmini* de *Erle Robotics* se usó junto con la *Raspberry Pi Zero W* ya que incorpora una unidad de medida inercial (IMU) y un controlador de servos.

OpenPilot Revolution: también se empleó la controladora de vuelo *OpenPilot Revolution* debido a la necesidad de incorporar más UARTs al sistema. También se probó su IMU, así como sus salidas PWM.

Sonar MB1240 XL-MaxSonar-EZ4: para lograr que la estimación de la altura en el EKF fuera más fiable que empleando únicamente las cámaras, se decidió utilizar un sonar *MB1240 XL-MaxSonar-EZ4* de *MaxBotix*.

Electronic Speed Controllers, motores y hélices: se emplearon cuatro motores BLDC (corriente continua sin escobillas) *EMAX MT2204-2300KV* y hélices 5030 de tres palas. Como ESC se escogieron cuatro *DYS SN20A*.

MPU 6050: durante un tiempo se utilizó también una IMU externa *MPU 6050*.

Waveshare Serial Expansion Hat: al final del proyecto se decidió emplear un *Waveshare Serial Expansion Hat* para *Raspberry Pi Zero W* para, descartando el *OpenPilot*, poder contar con canales UART adicionales.

Cámaras Flex 13 de OptiTrack: como método para facilitar la navegación autónoma del dron en interiores.

Batería, distribuidor de potencia y regulador de tensión: para la alimentación del UAV se empleó una batería LiPo de tres celdas y capacidad de 1000 mAh. Alimentaba a los controladores electrónicos de velocidad (ESC) a través de un distribuidor de potencia y a la *Raspberry Pi Zero W* gracias a un regulador de tensión.

Emisora Turnigy i10 y receptor FS-A8S: se usó una emisora *Turnigy i10*, que a su vez se enlazó con un receptor *FS-A8S* acoplado al dron.

Banco de ensayos y servo-tester: también se empleó un banco de ensayos junto con un servo-tester para caracterizar las curvas de empuje que proporcionaban los motores en función del PWM aplicado a cada ESC.

Estructura del UAV: por último, para la construcción del cuadricóptero se empleó una estructura de 250 mm (distancia entre dos motores opuestos) en configuración H.

1.5. Estructura de la memoria

Esta memoria está dividida en siete capítulos en los que se describe el trabajo desarrollado y los resultados obtenidos.

El capítulo 2 consiste en el estado del arte. En él se comentan algunos métodos utilizados en el vuelo autónomo de UAVs en interiores y, además, se mencionan algunas aplicaciones, como, por ejemplo, el inventariado de almacenes.

En el capítulo 3 se explican en detalle los diferentes componentes hardware que se han ido empleando a lo largo del proyecto, especificando las funciones que ha tenido cada uno de ellos.

En el capítulo 4 se presentan los recursos software que han sido utilizados.

Ya en el capítulo 5 se explica el sistema de control del UAV.

En el capítulo 6 se explican las comunicaciones empleadas, tanto a bordo del dron como en la estación base (GCS) y la emisora.

En el capítulo 7 se exponen la configuración final elegida para el dron así como los resultados obtenidos con la misma, tanto en vuelo manual como en navegación autónoma.

Por último, en el capítulo 8 se presentan las conclusiones, aportaciones adicionales y posibles futuros desarrollos.

2. Estado del Arte

En este capítulo se presentarán los distintos métodos o sistemas utilizados para la navegación autónoma de drones en interiores (sección 2.1). Además, se comentarán distintas aplicaciones que tienen ya los drones en zonas de interiores (sección 2.2).

2.1. Técnicas de navegación en interiores

El diseño del control de un UAV para vuelos interiores presenta nuevos desafíos debido a la imposibilidad de utilizar herramientas como el GPS o los magnetómetros (a causa de la gran cantidad de interferencias). Además, suelen ser zonas mucho más reducidas, con un mayor número de obstáculos, por lo que es necesaria muchísima más precisión en el sistema de posicionamiento. Las soluciones a estos inconvenientes implican la utilización de sensores adicionales. Éstos pueden ser sensores integrados en el propio UAV (cámaras, sensores de distancia, de flujo óptico, LIDAR...), en cuyo caso estamos hablando de técnicas SLAM (*Simultaneous Localization and Mapping*), o sensores externos. En esta sección se comentarán algunos de los sistemas más extendidos en la industria.

2.1.1. Ultra Wide Band (UWB)

Los sistemas de banda ultra ancha (UWB) se basan en la emisión de señales de anchos de banda superiores a los 500 MHz (o mayor del 20% de la frecuencia central) para la localización en interiores. Dichas frecuencias ofrecen múltiples ventajas para comunicaciones y aplicaciones relacionadas con la localización [5]. El gran ancho de banda hace que sean sistemas de mucha fiabilidad a cortas distancias. En [6] se ha tenido éxito al utilizar el sistema UWB en combinación con un Filtro Extendido de Kalman para la estimación de la altitud de UAVs. Sus principales ventajas e inconvenientes son:

Ventajas	**Inconvenientes**
Gran rendimiento: capaz de enviar muchos datos con bajo consumo de energía	Necesita de balizas en zona de vuelo y dron
Muy fiable para distancias cortas	No identifica la orientación del vehículo, por lo que no sirve para corregir el *yaw* (guiñada)

Tabla 2.1. Ventajas e inconvenientes del sistema UWB

2.1.2. Motion Capture System (MCS)

El sistema MCS (sistema de captura de movimiento) consta de varias cámaras de IR (infrarrojas) situadas en la zona de vuelo y reflectores sujetos al sólido rígido a rastrear. Las cámaras envían vectores dinámicos a la CPU y ésta posiciona los marcadores retro-reflexivos mediante triangulación. Es uno de los sistemas más utilizados para la navegación autónoma de drones en interiores (véanse [7] y [8]). Los principales sistemas MCS son los de *VICON* y *OptiTrack*. Esta tecnología es utilizada, por ejemplo, en un famoso video de TED en el que Raffaello D'Andrea muestra las asombrosas capacidades que pueden tener los UAVs con la ayuda de un sistema MCS [9]. Los drones no sólo resuelven complejos problemas individuales, sino que también pueden tomar decisiones en grupo. Esto es debido a que se pueden definir varios grupos de marcadores para que el sistema distinga entre distintos sólidos rígidos. Las principales ventajas e inconvenientes del sistema MCS son:

Ventajas	**Inconvenientes**
Identifica la orientación del vehículo, por lo que sirve para corregir el *yaw* (guiñada)	La zona de vuelo autónomo se limita a la rango de las cámaras
Ya disponible en la universidad (*OptiTrack*), por lo que no fue necesario adquirirlo	

Tabla 2.2. Ventajas e inconvenientes del sistema MCS

En este proyecto se empleó el sistema MCS de *OptiTrack*, con cámaras *Flex 13* y el software *Motive*.

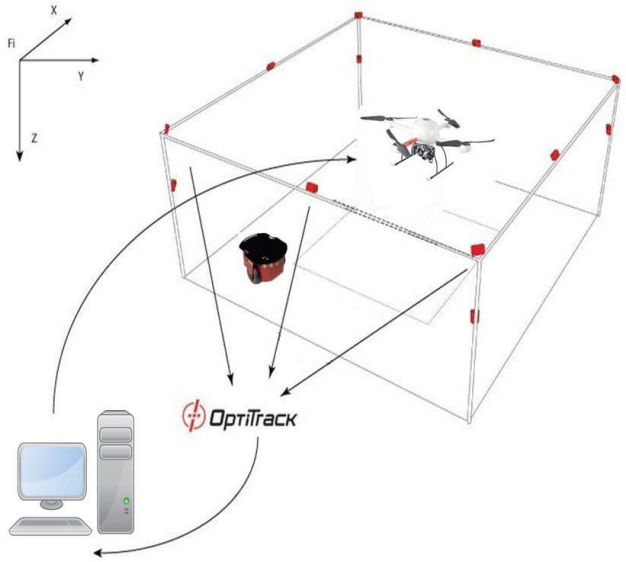

Figura 2.1. Diagrama de funcionamiento del MCS de *OptiTrack* [10]

Un sistema parecido es el *OTUS Tracker*. Consta de una estructura compuesta por múltiples sensores incorporada al dron y dos cámaras fijas situadas en un recinto de 5x5

metros. Con un precio más asequible, tiene unas prestaciones similares al sistema de captura de movimiento de *OptiTrack*. Sin embargo, el sistema de *OptiTrack* proporciona un recinto de trabajo mayor y es más flexible. Es por ello que se descartó el *OTUS Tracker*.

Figura 2.2. *OTUS Tracker* [11]

2.1.3. Navegación autónoma (SLAM)

Las técnicas de SLAM (*Simultaneous Localization and Mapping*) y Machine Learning permiten que pueda haber sistemas de navegación completamente autónomos a bordo de un dron [12]. Empresas como *DJI* y *Exyn Technologies* ya han desarrollado sistemas SLAM.

Sin ningún tipo de sensores externos, y para cumplir una determinada misión, el UAV ha de ser capaz de detectar por sí solo un camino libre de obstáculos desde el inicio hasta la meta. Para ello se sitúan distintos tipos de sensores en el dron, como pueden ser:

- Cámaras.
- Sensores de distancia.
- Sensores de flujo óptico.
- LIDAR (Light Detection and Ranging) [13].

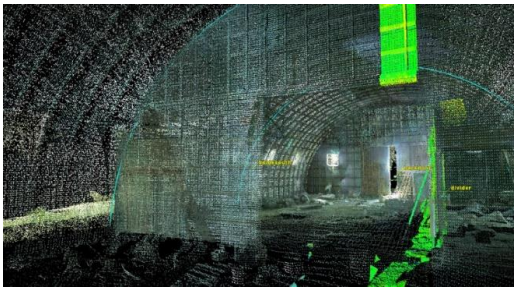

Figura 2.3. Mapeado de interiores por técnicas LIDAR [14]

Sus principales ventajas e inconvenientes son:

Ventajas	Inconvenientes
Autonomía total: el UAV no depende de sensores externos	Necesita de una mayor carga computacional

Tabla 2.3. Ventajas e inconvenientes de las técnicas SLAM

2.2. Aplicaciones de vuelo en interiores

La navegación autónoma de UAVs en interiores ha permitido que se desarrollen múltiples aplicaciones en diferentes ámbitos:

2.2.1. Inventariado de almacenes

El Instituto Fraunhofer de Flujo de Materiales y Logística propuso en el *Proyecto InventAIRy* sustituir la plantilla de trabajadores por drones que, utilizando técnicas de SLAM, pudieran navegar autónomamente. Desde entonces, el grupo francés Hardis [15] ha presentado una patente del dron "inventarista" *Eyesee* con el objetivo de automatizar el inventariado y corregir posibles errores en el almacenamiento. Geodis, un proveedor de servicios logísticos, también ha desarrollado un sistema de inventariado de almacenes innovador en colaboración con Delta Drone [16].

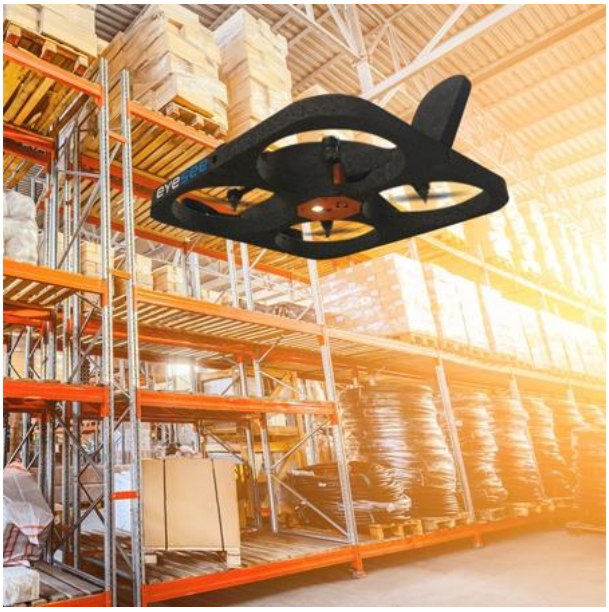

Figura 2.4. Dron *Eyesee* de Hardis [15]

2.2.2. En labores de rescate

Se han realizado múltiples estudios acerca de la viabilidad de aplicar la tecnología de los drones a labores de búsqueda y rescate en interiores (véanse [17] y [18]). En estos estudios se demuestra que en un futuro cercano los UAVs no solo podrían sustituir a los humanos en este tipo de tareas peligrosas, sino que serían también más eficientes.

2.2.3. Inspecciones internas

En los últimos años se ha estado estudiando la posibilidad de utilizar UAVs para la inspección y supervisión de instalaciones industriales. Un artículo publicado por un equipo de ingenieros de ETH Zürich [19] demostró en 2013 la viabilidad de utilizar a los drones como sustitutos de los humanos en tareas peligrosas relacionadas con la inspección de zonas industriales de difícil acceso. En el sector aeroespacial, por ejemplo, se está valorando la posibilidad de utilizar esta tecnología: un dron, haciendo uso de técnicas de reconocimiento de imágenes, podría ser de gran utilidad a la hora de detectar posibles defectos, como grietas, en el interior de una aeronave.

3. Hardware

En este capítulo se explica en más profundidad el hardware empleado en el proyecto. Consta de ocho secciones. En la sección 3.1 se indican las principales características de la controladora de vuelo escogida, la *Raspberry Pi Zero W*, y se indican las principales operaciones realizadas con respecto a la misma. En los puntos 3.2 y 3.3 se comentan el *PXFmini* y el *OpenPilot Revolution*, respectivamente, indicando los módulos de cada uno de ellos que se han utilizado. En la sección 3.4 se comentan los ESC, hélices y motores escogidos, así como los ensayos realizados a éstos. Ya en el punto 3.5 se comentan las cámaras escogidas para el sistema de localización y en la sección 3.6 se detalla la estructura del dron. En la sección 3.7 se indica el sonar elegido para el proyecto así como sus características y en la sección 3.8 se comenta una IMU externa que se utilizó durante un tiempo. Por último, en el punto 3.9 se comenta el *Waveshare Serial Expansion Hat* utilizado para tener más canales UART.

3.1. Raspberry Pi Zero W

La *Raspberry Pi Zero W* es un ordenador de placa simple o *single-board computer* (SBC) que opera en *Linux*, lo que permite la ejecución simultánea de diferentes tareas a distintos periodos de muestreo. Además, es posible programarla desde el entorno *MatLab/Simulink* en su última versión, la *R2018b*. Las características principales de la *Raspberry Pi Zero W* son [20]:

- Red Wireless LAN 802.11 b/g/n.
- Bluetooth 4.1.
- Bluetooth Low Energy (BLE).
- Procesador (CPU) de un núcleo, 1 GHz.
- Memoria RAM de 512 MB.
- Mini HDMI y puertos USB OTG (On-The-Go).
- Alimentación por micro USB.
- Compatible con HAT (Hardware Attached on Top) (40-pin header).

En la siguiente imagen (Figura 3.1) se pueden ver las principales especificaciones de la *Raspberry Pi Zero W* [21]:

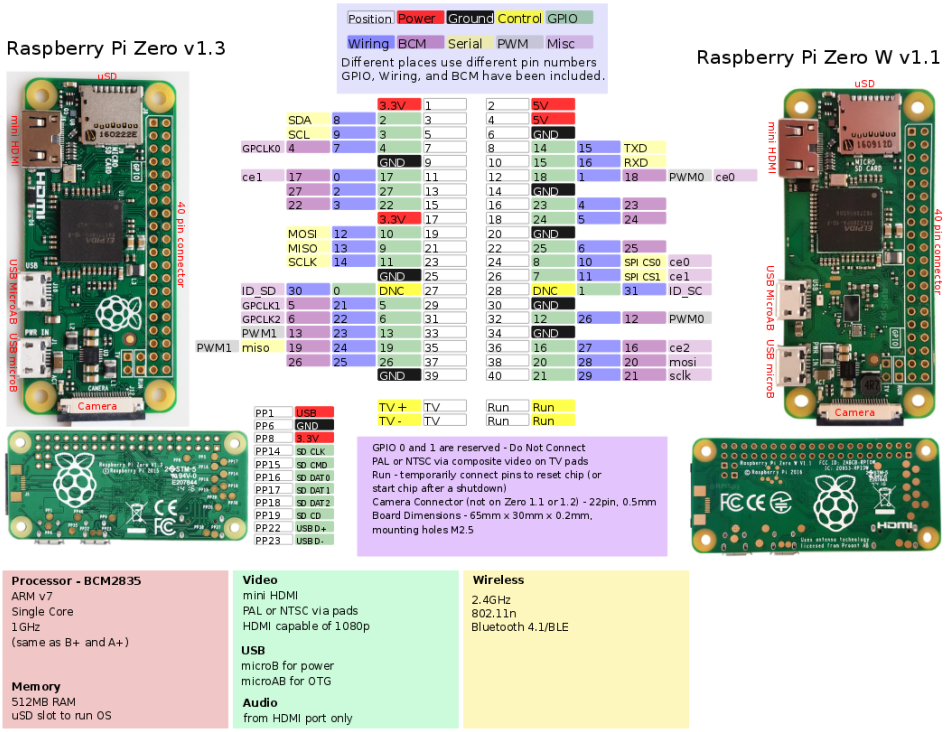

Figura 3.1. Especificaciones de la *Raspberry Pi Zero W*

En este proyecto se han utilizado los siguientes pines GPIO (con la numeración mostrada en la imagen de la Figura 3.1):

- 2: alimentación a 5 V para la IMU.
- 3: SDA de la conexión I²C con la IMU.
- 5: SCL de la conexión I²C con la IMU.
- 6: conexión de tierra (GND) con la IMU.
- 8: transmisión (Tx) de la conexión UART con el *OpenPilot Revolution*.
- 9: conexión a tierra (GND) con el *OpenPilot Revolution*.
- 10: recepción (Rx) de la conexión UART con el *OpenPilot Revolution*.

3.2. PXFmini

El *PXFmini* de *Erle Robotics* es un módulo de autopiloto diseñado para la *Raspberry Pi Zero W* que permite que un dron navegue autónomamente. Incorpora, entre otras cosas, una IMU y un controlador de servos *PCA9685*. También cuenta con un barómetro que no se ha utilizado. Sus principales características se muestran en la siguiente imagen (Figura 3.2) [22]:

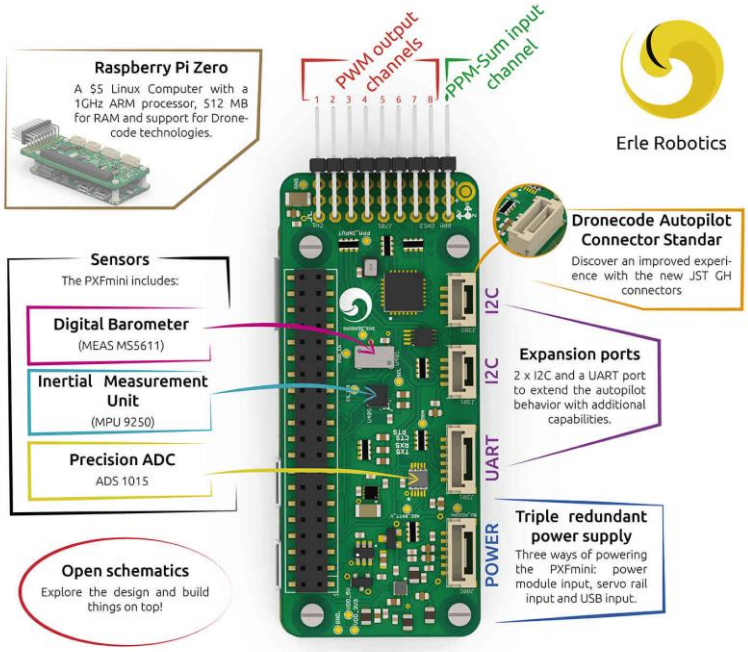

Figura 3.2. Principales características del *PXFmini*

En este proyecto se han utilizado diferentes módulos del *PXFmini*:

- *MPU 9250* (IMU): lleva incorporados un magnetómetro, un sensor de temperatura, un giróscopo y un acelerómetro. Se utilizaron las medidas del giróscopo y el acelerómetro, ya que el magnetómetro se ve expuesto a muchas interferencias en vuelos de interiores. Se optó por efectuar todas las medidas posibles con los sensores (cada 1 ms) para luego ir haciendo la media de las últimas 10 (cada 10 ms), reduciéndose así el ruido de medida considerablemente. También se implantó un filtro digital de 10 Hz con el mismo propósito. La información de la IMU se envió a través del protocolo SPI.

- LEDs: el *PXFmini* incorpora tres LEDs. Dos de ellos (azul y naranja) se utilizaron para diferenciar los distintos estados en los que se encontraba el dron.

- *PCA9685*: se trata de un controlador de PWM que se utilizó para controlar la potencia de los motores. Resumiendo, su función es señalar los tiempos en los que las salidas (que se dirigen a los ESC) se ponen a nivel alto y a nivel bajo. Como se indica en [1], los registros se escriben como se indica a continuación:

    - Registros de encendido: encienden la salida poniéndose a 0 al inicio del periodo de la señal.

- Registros de apagado: siguen la siguiente ecuación para determinar el tiempo de apagado en número de muestras (0 – 4095):

$$t_{apagado} = Pulso \times Frecuencia \times 10^{-6} \times 4096 - 1$$

Siendo $t_{apagado}$ el tiempo de apagado en número de muestras, $Pulso$ la amplitud del pulso PWM en microsegundos (de 1000 a 2000) y $Frecuencia$ la frecuencia de la señal PWM (432,5 Hz).

- *Safety switch*: para incorporar el *safety switch* se utilizó el canal de entrada PPM-Sum.

3.3. OpenPilot Revolution

El *OpenPilot Revolution* es una controladora de vuelo con un procesador *STM32F405RGT6 ARM Cortex-M4* que cuenta con IMU, salidas PWM, entrada para un sensor sonar, entradas digitales y múltiples UARTs, además de otros muchos módulos. En la Figura 3.3 se muestran sus principales puertos [23]:

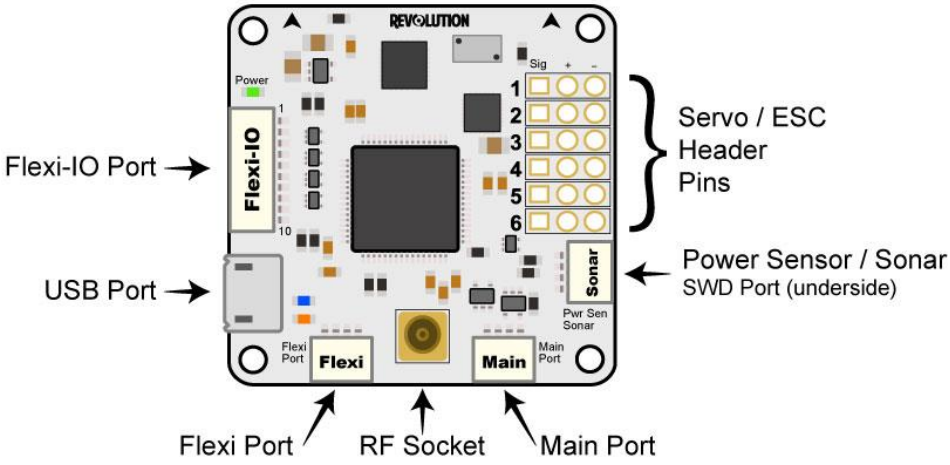

Figura 3.3. Principales puertos del *OpenPilot Revolution*

En este proyecto se utilizaron, a lo largo de diferentes etapas, los siguientes módulos:

- *MPU 6000* (IMU): incorpora un giróscopo y un acelerómetro. Al igual que con la IMU del *PXFmini*, se optó por realizar todas las medidas posibles (cada 1 ms) para luego hacer la media (cada 10 ms), además de implantarse un filtro digital de 10 Hz. En este caso la comunicación también se llevó a cabo por el protocolo SPI.

- Salidas PWM: se utilizaron cuatro salidas PWM para los motores. Éstas fueron las siguientes:

  - Puerto B1 (Timer 3)
  - Puerto A3 (Timer 9)
  - Puerto A2 (Timer 2)
  - Puerto A1 (Timer 5)

- Sonar: se hizo uso del *Sonar Port*, que alimenta al sonar *MB1240* y convierte el voltaje emitido por el mismo en una medida digital cruda (conversión ADC) de la altura del dron que, tras ser acondicionada[1], es usada en el control.

- *Safety switch*: para el *safety switch* se utilizó el puerto B0 como entrada digital.

- UARTs: en el proyecto se utilizaron tres diferentes UARTs en el *OpenPilot Revolution*:

  - UART 1: en el *Main Port* (Tx/Rx en los puertos A9/A10), a 115200 baudios. Para la comunicación con la *Raspberry Pi Zero*.
  - UART 2: en el *Flexi Port* (sin Tx, Rx en el puerto B11), a 115200 baudios. Para la comunicación con la emisora.
  - UART 3: en el *Input Header* (sin Tx, Rx en el puerto C7), a 57600 baudios. Para la comunicación con la GCS.

3.4. Electronic Speed Controllers, motores y hélices

Los motores elegidos para el cuadricóptero fueron cuatro BLDCs (corriente continua sin escobillas) *EMAX MT2204-2300KV* equipados con hélices 5030 de tres palas. Como ESC, que son los dispositivos encargados de controlar la potencia o empuje de los motores en función del pulso de PWM recibido, se escogieron cuatro (uno por motor) *DYS SN20A*. Para configurarlos se utilizó el programa *BLHeli Suite*, decidiéndose que el empuje fuera mínimo (nulo) con un PWM de 1000 µs y máximo con uno de 2000 µs.

Antes de comenzar los ensayos de vuelo manual se decidió comprobar el correcto funcionamiento del sistema ESC-motor-hélice en un banco de ensayos. Para ello se ancló el motor a una estación (como la mostrada en la Figura 3.4) capaz de medir el empuje provocado por el mismo en gramos.

[1]Véase la sección 3.7.

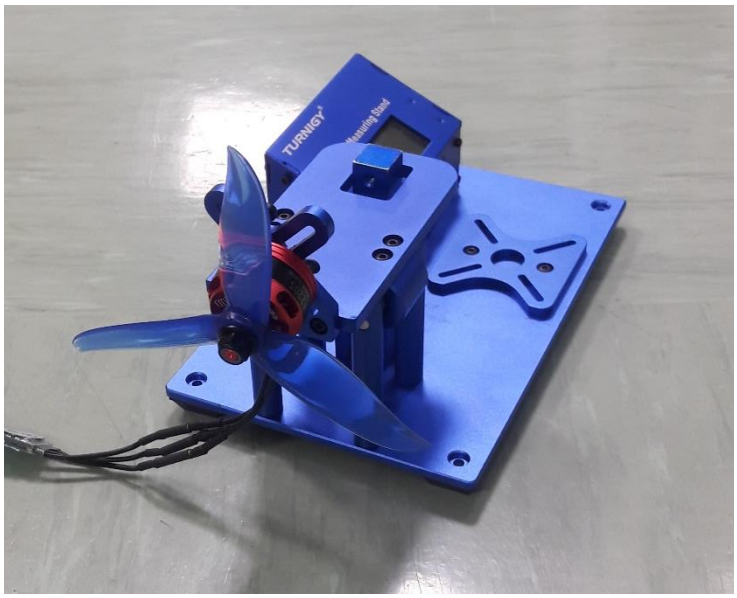

Figura 3.4. Banco de ensayos de motores

Los resultados obtenidos en función del motor y el PWM a plena batería fueron:

Empuje (g)							
PWM	PROA-ESTRIBOR	PROA-BABOR	POPA-ESTRIBOR	POPA-BABOR	Media	Varianza	Desviación típica
1000	0	0	0	0	0	0	0
1100	18	17	20	19	18,50	1,25	1,12
1200	41	36	39	43	39,75	6,69	2,59
1300	69	67	68	72	69,00	3,50	1,87
1400	99	97	97	104	99,25	8,19	2,86
1500	134	132	133	138	134,25	5,19	2,28
1600	162	160	164	170	164,00	14,00	3,74
1700	204	199	206	209	204,50	13,25	3,64
1800	233	229	239	242	235,75	25,69	5,07
1900	272	268	281	290	277,75	72,19	8,50
2000	320	311	329	338	324,50	101,25	10,06

Tabla 3.1. Resultados de los ensayos de los motores

El empuje de cada uno de los motores (en gramos) y su media en función del PWM fueron:

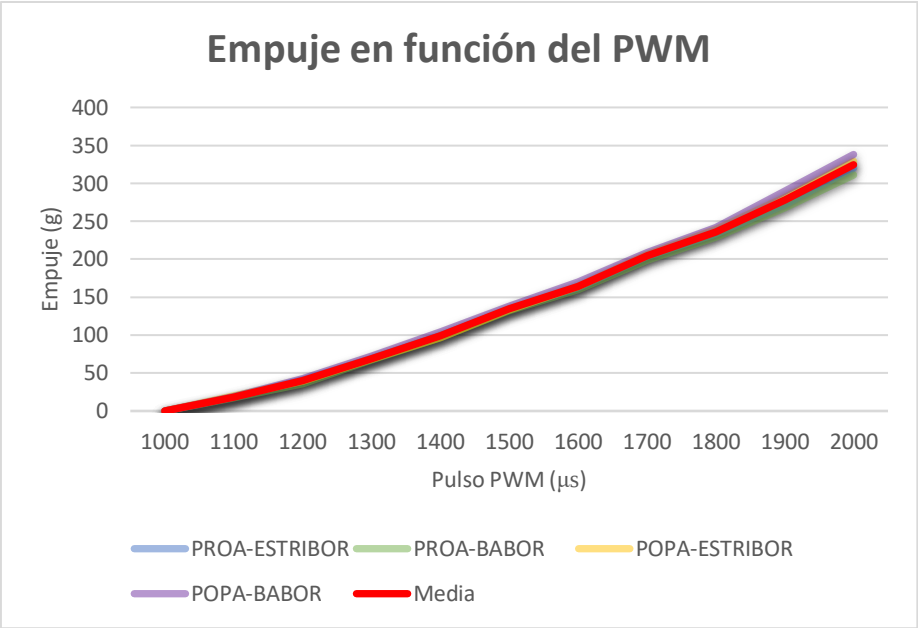

Figura 3.5. Empuje en función del PWM

Por otro lado, la desviación típica (σ) de los empujes de los distintos motores en función del PWM fue:

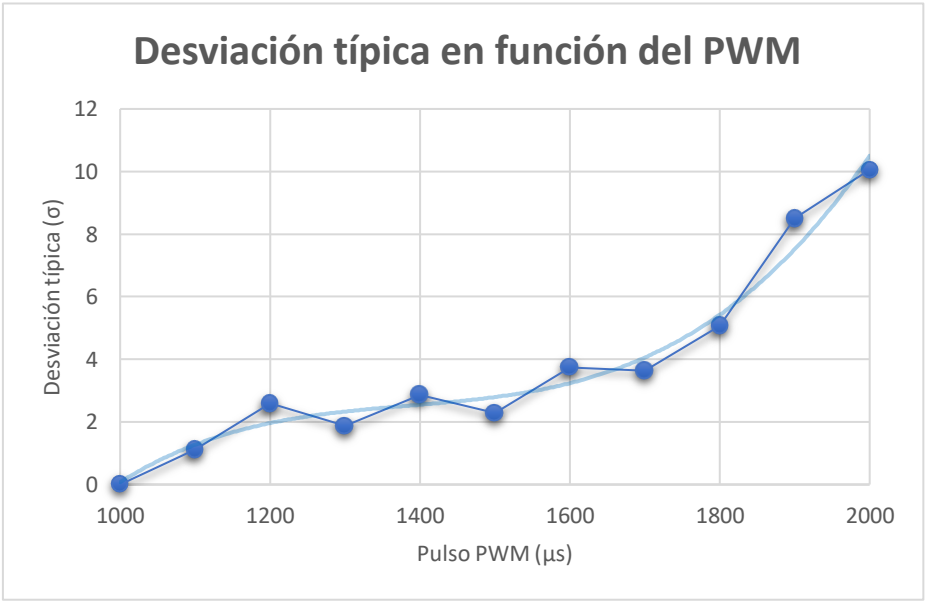

Figura 3.6. Desviación típica en función del PWM

A la vista de estos resultados (Figura 3.6), es posible decir que, hasta un PWM de 1700, la desviación típica entre motores es baja, lo que implica que podemos suponer que estos se van a comportar de manera suficientemente parecida como para que el cuadricóptero vuele adecuadamente. Por otro lado, la desviación típica crece considerablemente en la zona más alta del PWM (de 1700 a 2000), pero el cuadricóptero

rara vez va a necesitar tanta potencia para volar (su peso es de, aproximadamente, 520 g, siendo el empuje combinado a un PWM de 1700 aproximadamente 820 g).

3.5. Cámaras OptiTrack (Flex 13)

Para el sistema de localización externo del dron se utilizaron 8 cámaras *Flex 13* de *OptiTrack* como las mostradas en la Figura 3.7 [24].

Figura 3.7. Cámaras *Flex 13* de *OptiTrack*

Las principales características de las cámaras *Flex 13* de *OptiTrack* se muestran a continuación [25]:

- Anillo de 28 LEDs IR (infrarrojos) de 850 nm.
- Brillo ajustable e iluminación continua o estroboscópica.
- Lente de 5,5 mm (F1.8) con un campo de visión horizontal de 56º y vertical de 46º.
- Resolución de imagen de 1,3 megapíxeles, con un tamaño de imagen de 6,144 x 4,9152 mm y un tamaño de pixel de 4,8 x 4,8 µm.
- Fotogramas por segundo ajustables: 30-120 FPS.
- Latencia de 8,3 ms.
- Filtro paso alto de 800 nm.
- Dimensiones de 53,8 x 81 x 42,4 mm y peso de 187 g.

A través de tres *OptiHub 2* las cámaras *Flex 13* envían la información a *Motive*, el programa encargado de analizarla y obtener la posición y la orientación del dron (en cuaterniones). Posteriormente esta información es enviada a la GCS, que traduce los cuaterniones a ángulos de Euler y envía la orientación y la posición al UAV por UART.

3.6. Estructura del dron

Dentro de la tecnología de los UAVs existen múltiples tipos de estructuras diferentes (Figura 3.8) [26]:

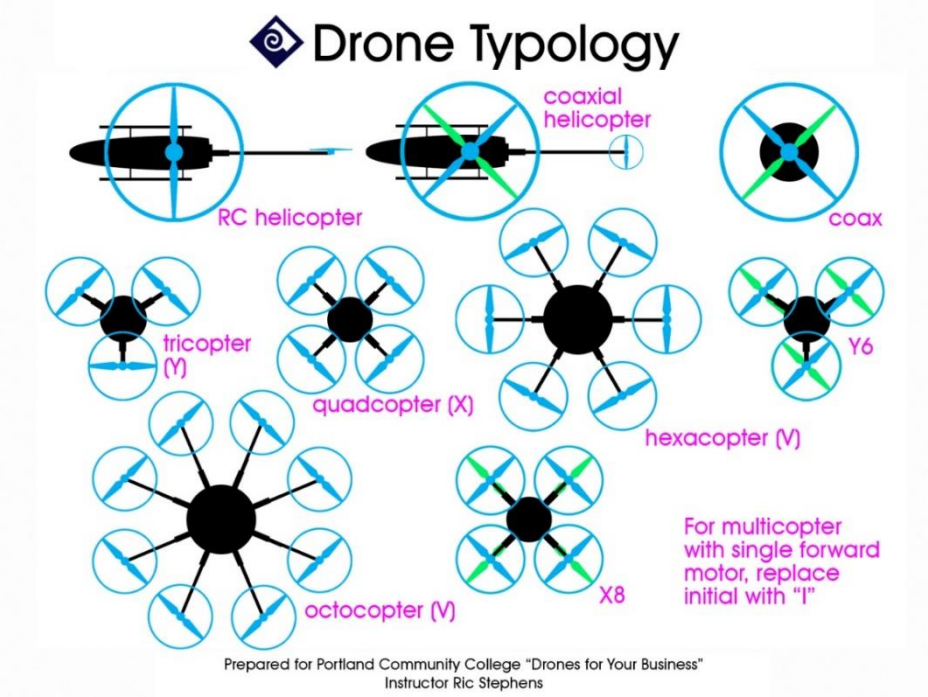

Figura 3.8. Diferentes tipologías de drones

El número de rotores de un dron depende principalmente de dos factores: el rango y el peso. Los cuadricópteros (cuatro rotores) tienen una disposición ideal para un corto rango de vuelo y un peso moderado. Es por ello, y teniendo en cuenta que se va a volar en un espacio de interiores, que se ha decidido utilizar una estructura de cuatro rotores en este proyecto. Los cuadricópteros, sin embargo, también pueden ser de distintos tipos (Figuras 3.9 y 3.10) [27] [28]:

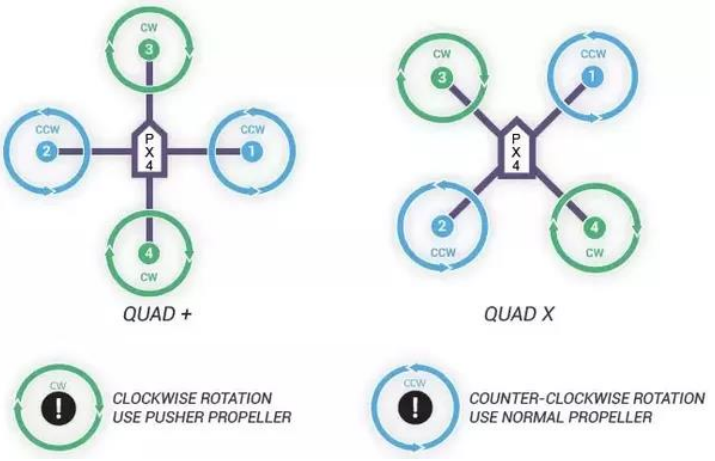

Figura 3.9. Disposiciones en X y + de un cuadricóptero

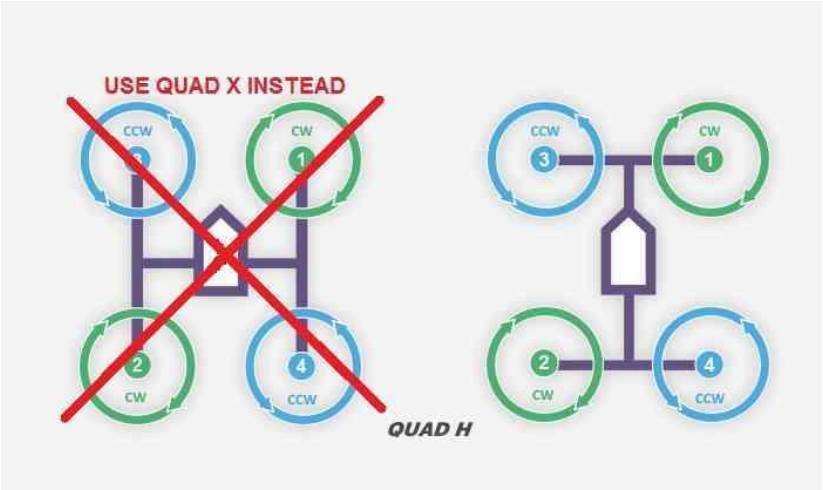

Figura 3.10. Disposición H de un cuadricóptero

Cómo se puede ver en las imágenes, la única diferencia sustancial a nivel estructural entre las disposiciones X y H es el sentido de giro de los rotores. Ambos ofrecen, como bien se indica en [1], mayor comodidad que la configuración + si en un futuro se desea situar una cámara frontal en el dron. En este proyecto finalmente se optó por la disposición H, siguiendo los pasos de [1].

3.7. Sonar MB1240

Para hacer más preciso el control en el vuelo autónomo se decidió, a parte de las cámaras MCS y de la IMU, incorporar un sonar. Para ello se escogió el *MB1240 XL-MaxSonar-EZ4* de *MaxBotix*, cuyas principales características son:

- Alimentación entre 3,3 y 5 V.
- Lecturas cada 100 ms (ratio de 10 Hz).
- Rango máximo de 7,6 m.
- En caso de que se envíe la medida por ADC, la conversión de V a cm depende de la alimentación según la siguiente ecuación:

$$altura\ cruda \sim \frac{V_{CC}\ (V)}{1024}\ (cm)$$

- Si la información se envía por UART, el mensaje consiste en:
  - Un byte ASCII “R” mayúscula.
  - Tres bytes ASCII representando la medida en cm hasta un máximo de 765.
  - Un byte ASCII 13.

El sonar se empezó a utilizar cuando se introdujo el *OpenPilot Revolution*, comunicándose con este a través de tres pines: tierra, alimentación a 5 V y conversor ADC (Figura 3.11) [29].

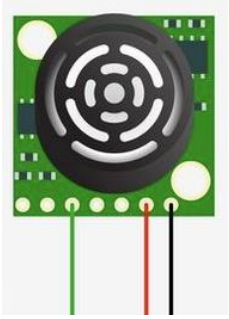

Figura 3.11. Sonar *MB1240 XL-MaxSonar-EZ4*

Posteriormente se prescindió del *OpenPilot* y se empezaron a enviar las medidas del sonar por UART al *Waveshare Serial Expansion Hat*.

En la práctica se observó que las medidas crudas del sonar eran bastante inexactas. Además, solo medía correctamente por encima de los 35 cm aproximadamente. Es por ello que se realizó el siguiente acondicionamiento lineal de medida por comparación con las medidas de las cámaras *Flex 13*:

- Si la altura cruda es inferior a 35 cm:

$$altura\ acondicionada = 0\ cm$$

  Esto se debe a que, si el control recibe una medida de 0 exacto, no la tiene en cuenta.

- Si la altura cruda es superior a 35 cm:

$$altura\ acondicionada = 14{,}64 + 0{,}73 \times altura\ cruda\ (cm)$$

3.8. MPU 6050

Al comprobar que la IMU del *OpenPilot Revolution* no arrojaba buenos resultados, en una primera instancia se decidió incorporar una IMU externa *MPU 6050*, que posteriormente fue sustituida por el *PXFmini*.

La *MPU 6050* cuenta con un acelerómetro y un giróscopo, ambos de 3 ejes. Además, cuenta con un canal I^2C (no SPI). Estaba conectada a la Raspberry Pi Zero W a través de cuatro pines:

- Alimentación a 5 V.

- Tierra (GND).
- SCL (I^2C).
- SDA (I^2C).

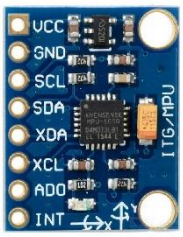

Figura 3.12. IMU *MPU 6050* [30]

3.9. Waveshare Serial Expansion Hat

Ya al final del proyecto se optó por incorporar un *Waveshare Serial Expansion Hat* a la *Raspberry* para tener todos los canales UART necesarios (GCS, emisora y sonar). El *Hat* cuenta con dos canales UART adicionales que, sumados al canal de la *Raspberry Pi Zero W*, suman tres en total. La información recibida por UART en el *Hat* es posteriormente enviada por I^2C a la *Raspberry*.

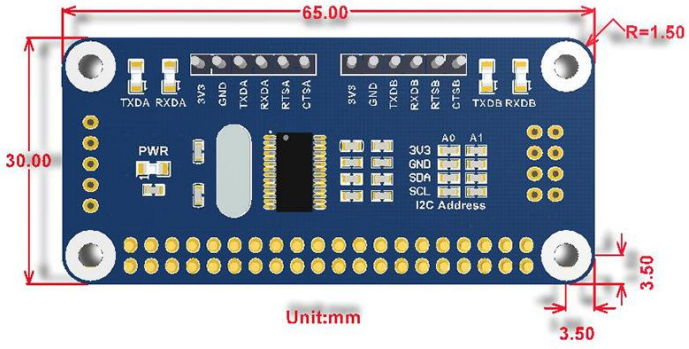

Figura 3.13. *Waveshare Serial Expansion Hat* [31]

4. Software

En este capítulo se explica en detalle el software empleado. Está formado por tres secciones. En la sección 4.1 se comenta el principal entorno de programación del proyecto, *MatLab/Simulink*. En el punto 4.2 se habla en detalle de *Motive*, el programa que se encarga de procesar la información captada por las cámaras *Flex 13*. Por último, en la sección 4.3 se comenta la aplicación *BLHeli Suite*, que se ha usado para la programación de los ESC.

4.1. MatLab/Simulink

Para programar el software se ha utilizado el entorno *MatLab/Simulink*, en concreto la versión *R2018b*. Ésta cuenta con soporte para *Raspberry Pi Zero W* y, gracias al *Waijung Blockset*, también para microcontroladores *STM32F4*, lo que facilita la programación del *OpenPilot Revolution*.

En este proyecto el software de *MatLab/Simulink* se ha dividido en varias secciones. Éstas son:

- *Raspberry Pi Zero W*
- *OpenPilot Revolution*
- Estación de control de tierra (GCS)

4.1.1. Raspberry Pi Zero W

En cuanto a la *Raspberry Pi Zero W*, para permitir su programación desde *MatLab/Simulink R2018b* se hizo lo siguiente:

1. Descargar la imagen *Raspbian Stretch with desktop* en el siguiente enlace: https://www.raspberrypi.org/downloads/raspbian/.

2. Utilizar *Etcher* para grabar la imagen en una tarjeta Micro SD de 16 GB.

3. Conectar la *Raspberry Pi Zero W* a un monitor (mini HDMI a HDMI) y usar un teclado y un ratón (micro USB a USB) para configurar la conexión WiFi. A continuación se muestra el código:

/etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network = {
ssid="SSID_RED_1"
psk="CONTRASEÑA_1"
key_mgmt=WPA-PSK
}

network = {
ssid="SSID_RED_2"
psk="CONTRASEÑA_2"
key_mgmt=WPA-PSK
}

De esta forma ya era posible configurar la *Raspberry Pi Zero W* de forma inalámbrica usando *PuTTY* (conexión SSH) o a través de VNC (Virtual Network Computing). Además, fue necesario realizar este paso antes de instalar el *Simulink Support Package for Raspberry Pi Hardware* porque la otra alternativa que no requería conexión WiFi para hacerlo causaba errores.

4. Instalar a través de *MatLab/Simulink* el *Simulink Support Package for Raspberry Pi Hardware* y hacer las pruebas pertinentes para comprobar su correcto funcionamiento (encender un LED…).

Además, hubo que realizar configuraciones adicionales para que todo funcionara correctamente:

- Habilitar las siguientes opciones de interconexión: SSH, VNC, SPI, I²C y serie. Esto se realizó fácilmente a través del comando *sudo raspi-config*, en *Interfacing Options*.

- Además para permitir la comunicación UART fue necesario modificar el fichero */boot/config.txt* añadiendo *enable_uart = 1*.

- Por otro lado, también fue necesario cambiar la velocidad de la conexión I²C. Esto se consiguió, de nuevo, modificando el fichero */boot/config.txt*. En este caso, para que funcionara a la velocidad deseada (400 kbps), se añadió lo siguiente: *dtparam = i2c_baudrate = 400000*.

4.1.2. Waijung Blockset (OpenPilot Revolution)

Para poder programar el *OpenPilot Revolution* en el entorno *MatLab/Simulink* fue necesaria la instalación del *Waijung Blockset* (junto con el *UC3M Add-on Blockset*), que consiste en los siguientes pasos [32]:

1. Descargar e instalar el programa *STM32 ST-Link Utility* y el driver *ST-Link/V2 USB*. El *ST-Link Utility* sirve para descargar archivos ejecutables en la memoria flash del *STM32*.

2. Descargar e instalar el driver *FTDI USB*.

3. Conectar el *OpenPilot* al ordenador a través del *ST-Link V2* y comprobar la conexión desde la aplicación *STM32 ST-Link Utility*.

4. Instalar el *Waijung Blockset*:

    a. Descargar el paquete de instalación del *Waijung Blockset* y el *STM32F4 Target*.
    b. Extraer la carpeta *.7z* descargada y abrir desde *MatLab* el fichero *install_waijung.m*.
    c. Correr el archivo como administrador.
    d. Utilizar la opción *Set Path* para añadir la carpeta *waijung* al *path* de *MatLab/Simulink*.

Tras la instalación del *Waijung Blockset* ya fue posible programar el *OpenPilot* desde *MatLab/Simulink*.

4.2. Motive

La aplicación que se usa para el procesamiento de la información captada por las cámaras y el envío de ésta a la GCS (por *streaming*) es *Motive*. La GCS, por su parte, utiliza una función proporcionada por *OptiTrack* (*FuncMotive*) para leer correctamente la información enviada desde *Motive*.

Lo primero que se debe de hacer en *Motive* es la calibración de las cámaras para que las medidas obtenidas sean más precisas. Esto se realizó moviendo la *CW-500 Calibration Wand* por la zona de detección de las cámaras. Posteriormente se empleó el *CS-200 Calibration Square* para fijar el origen y el sistema de coordenadas deseado.

Tras haber realizado la calibración es necesario definir el sólido rígido con el que se va a trabajar (en este caso el dron). Con todo esto realizado, la aplicación envía la posición XYZ y la orientación en cuaterniones, que luego es transformada a ángulos de Euler en la GCS. Además, al trabajar con el dron en la zona de vuelo autónomo, es recomendable realizar una calibración continua (*Continuous Calibration*) para que la información enviada sea lo más fiable posible.

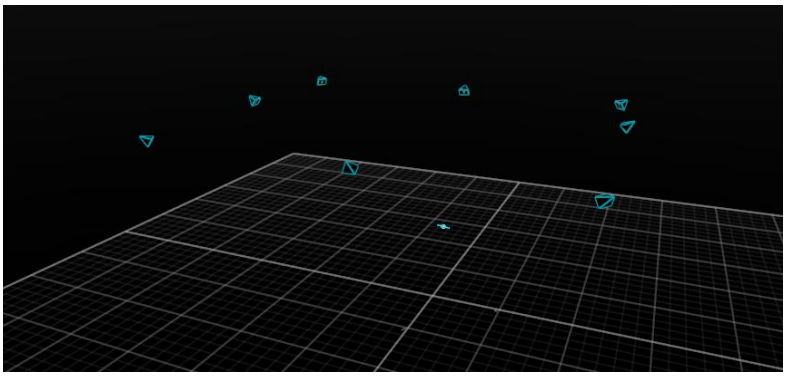

Figura 4.1. Las cámaras y el dron vistos desde *Motive*

4.3. BLHeli Suite

BLHeli Suite es la aplicación que se ha empleado para la programación de los ESC. La configuración escogida se puede observar en la Figura 4.10:

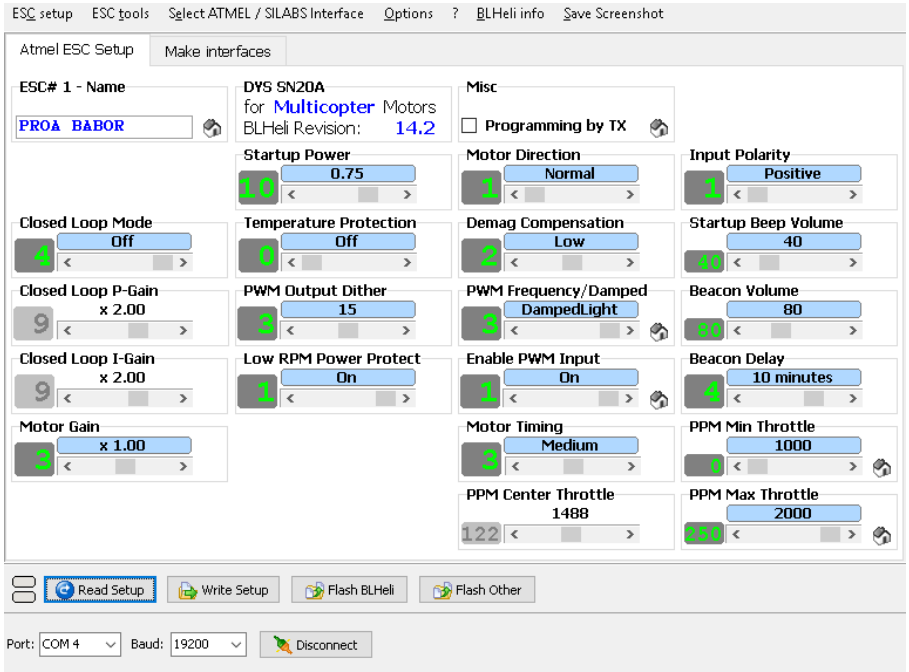

Figura 4.2. Configuración de los ESC en *BLHeli Suite*

5. Sistema de control

En este capítulo se explica la estructura del sistema de control del UAV. Consta de dos secciones. En la sección 5.1 se aborda la parte del sistema de control en la *Raspberry Pi Zero W*. En el punto 5.2 se mencionan las funciones del *OpenPilot Revolution*, mientras que la sección 5.3 trata de la estación de control de tierra (GCS). Téngase en cuenta que todo lo relacionado con las comunicaciones no será comentado en detalle en este capítulo, sino más adelante, en el capítulo 6.

5.1. Raspberry Pi Zero W

En cuanto a la *Raspberry Pi Zero W*, el software se carga desde el archivo *RASPI_CONTROL_SYSTEM.slx*. Éste principalmente se encarga del control, la máquina de estados y el EKF. Sin embargo, también tiene otras funciones a lo largo de diferentes etapas del proyecto:

- Control de los PWM enviados a los ESC a partir del *PXFmini*.
- Comunicación UART con el *OpenPilot Revolution* o con la emisora.
- Comunicación *MAVLink* con la GCS.
- Control de LEDs y recepción del *safety switch*.
- Procesamiento de los canales enviados desde la emisora.
- Simulación.
- Cálculo de la carga computacional.
- Procesamiento de las medidas crudas recibidas desde la IMU[2] del *PXFmini*.

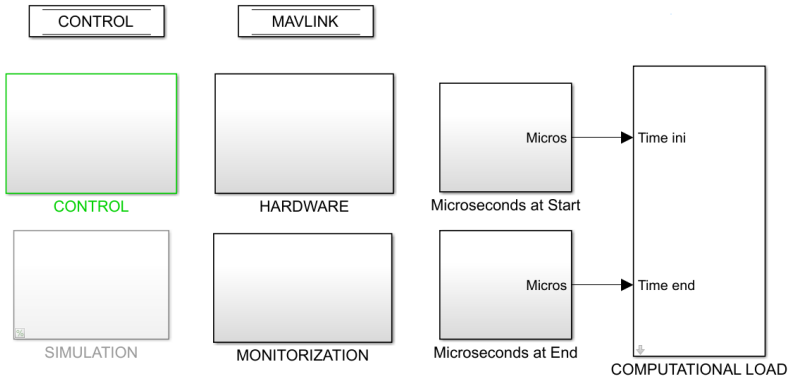

Figura 5.1. Ventana principal de *RASPI_CONTROL_SYSTEM.slx*

[2]Véase la sección 3.2.

Las tres principales secciones del software de la *Raspberry* se explican a continuación. Téngase en cuenta que el modelo matemático utilizado en el control se explica en detalle en el Anexo.

5.1.1. Control

El principal objetivo del control es garantizar la estabilidad del UAV. Dado a que el dron ha de estar preparado para volar manual y autónomamente, podemos distinguir dos principales tipos de controles: el de vuelo manual y el de navegación autónoma.

Para facilitar la implantación del control en distintos drones se siguió una estrategia que consistía en realizar un control que no dependiera de las características físicas del dron [1]. De esta forma, tan solo es necesario cambiar los parámetros físicos del UAV en el modelo, sin que esto afecte al control genérico diseñado, optimizado por mínimos cuadrados. Esto además permite desacoplar los controles de estabilización en cada eje.

5.1.1.1. Vuelo manual

El esquema general de vuelo manual es el siguiente:

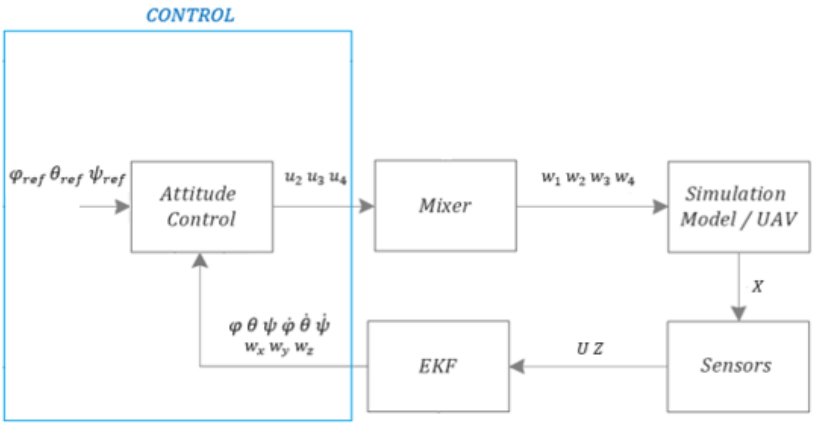

Figura 5.2. Esquema general de vuelo manual

Como se puede observar en la Figura 5.2, en este caso el control tan sólo cuenta con una etapa: el control de estabilización (*Attitude Control*). El control de estabilización se encarga, como bien dice su nombre, de estabilizar el cuadricóptero. Recibe las referencias del *roll*, *pitch* y *yaw* ($\varphi_{ref}, \theta_{ref}, \psi_{ref}$) desde la emisora, así como los ángulos y velocidades de *roll*, *pitch* y *yaw* ($\varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}$) y las velocidades angulares vistas desde los ejes del dron (w_x, w_y, w_z) actuales desde el EKF. El esquema del control de estabilización detallado se muestra a continuación:

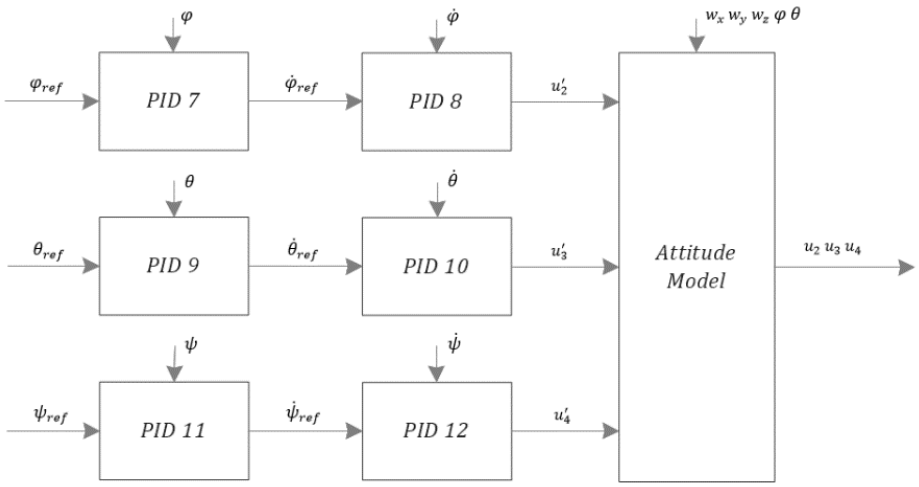

Figura 5.3. Control de estabilización

Se puede observar (Figura 5.3) como el control de estabilización consta de seis controles Proporcional-Integral-Derivativo (PID), dos PID en cascada por cada ángulo de referencia de entrada. Las salidas del control en cascada son las aceleraciones de *roll, pitch* y *yaw* de referencia ($\ddot{\varphi}_{ref} = u_2', \ddot{\theta}_{ref} = u_3', \ddot{\psi}_{ref} = u_4'$), que pasan al *Attitude Model*, un modelo matemático con el que, a partir de estas aceleraciones y de las velocidades angulares vistas desde los ejes del dron (w_x, w_y, w_z) actuales, se determinan las fuerzas unitarias de *roll*, *pitch* y *yaw*, que a su vez entran al *Mixer*, que posteriormente determina la potencia requerida en cada uno de los motores del dron. Si se quiere más información acerca del *Attitude Model* véase el Anexo.

5.1.1.2. Navegación autónoma

El esquema general de navegación autónoma es el mostrado a continuación:

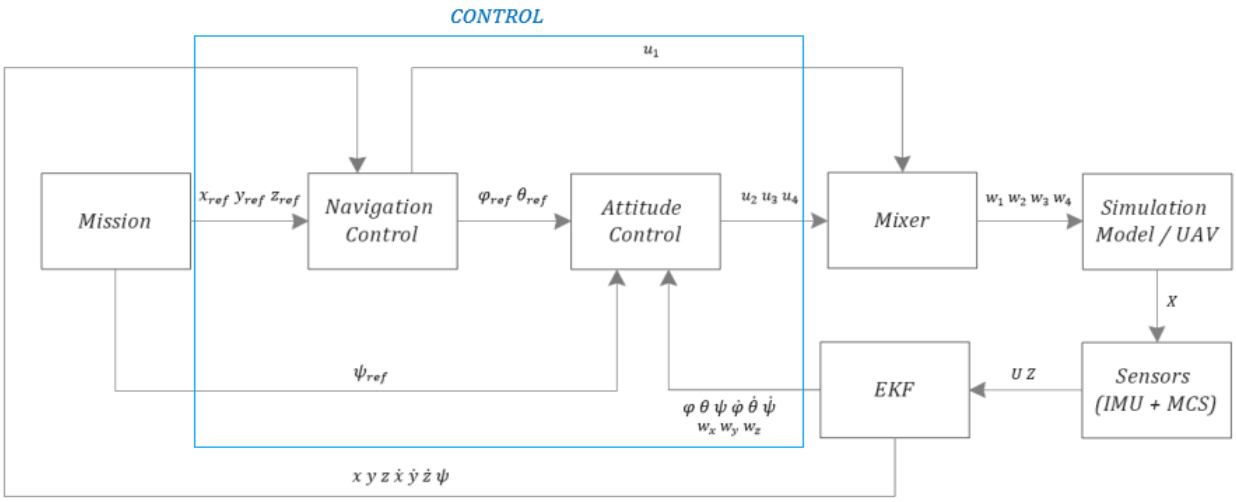

Figura 5.4. Esquema general de navegación autónoma

En la Figura 5.4 se puede notar como, en esta ocasión, el control cuenta con dos bloques diferentes: el control de estabilización (*Attitude Control*) y el control de navegación (*Navigation Control*). Las funciones del control de estabilización son idénticas a lo mostrado anteriormente con la salvedad de que, en esta ocasión, recibe las referencias del *roll* y del *pitch* ($\varphi_{ref}, \theta_{ref}$) desde el modelo de navegación y la referencia del *yaw* (ψ_{ref}) desde la misión. El esquema es el mismo que el mostrado en la Figura 5.3.

El control de navegación, por otro lado, es el encargado del vuelo autónomo. Recibe las referencias de posición ($x_{ref}, y_{ref}, z_{ref}$) desde la misión y las posiciones, velocidades lineales y el *yaw* ($x, y, z, \dot{x}, \dot{y}, \dot{z}, \psi$) actuales estimados por el Filtro Extendido de Kalman. El esquema del control de navegación es el siguiente:

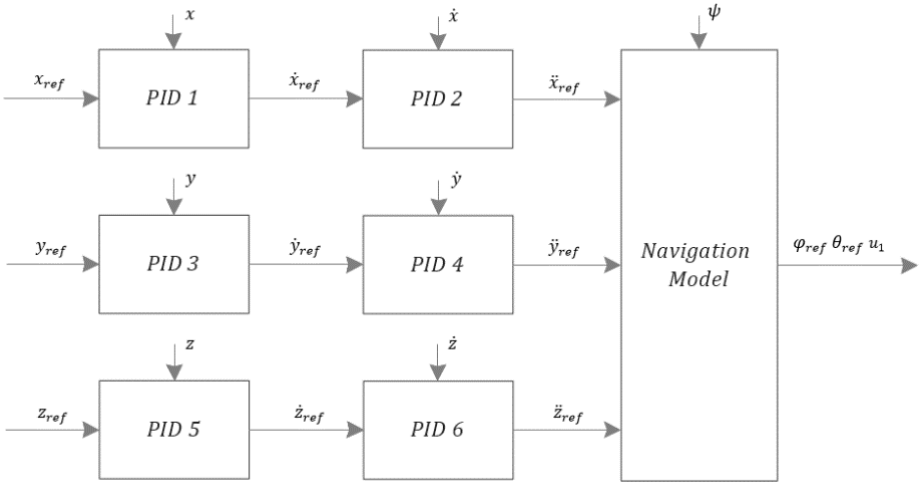

Figura 5.5. Control de navegación

Se puede observar (Figura 5.5) como el control de navegación consta también de seis controles PID, dos PID en cascada por coordenada. Las salidas del control en cascada son las aceleraciones lineales de referencia ($\ddot{x}_{ref}, \ddot{y}_{ref}, \ddot{z}_{ref}$), que pasan al *Navigation Model*, un modelo matemático con el que, a partir de estas aceleraciones y del *yaw* actual, se determinan el *roll* y el *pitch* de referencia ($\varphi_{ref}, \theta_{ref}$), que entran en el *Attitude Control*, y el *thrust* o empuje unitario (u_1), que entra directamente al *Mixer*. Si se quiere más información acerca del *Navigation Model* véase el Anexo.

5.1.2. Máquina de estados

La máquina de estados utilizada en el control fue la mostrada en el siguiente esquema (Figura 5.6):

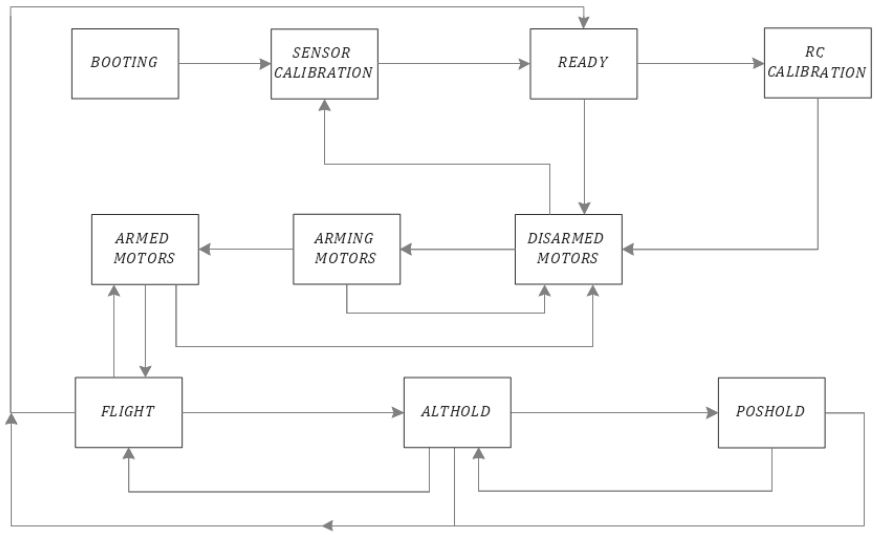

Figura 5.6. Máquina de estados

El funcionamiento de la máquina de estados es el siguiente:

1. *BOOTING*: inicialización de 3 segundos de duración.

2. *SENSOR CALIBRATION*: calibración de las medidas de los sensores: durante 30 segundos se hace una media de cada una de las medidas de la IMU y, posteriormente, estas medias se restan a las nuevas medidas. Esto es debido a que, como el dron está en reposo, las medias deberían coincidir con el *offset* de los sensores.

3. *READY*: tras la calibración, pulsando el *safety switch* se puede pasar directamente al proceso de armado de los motores (*DISARMED MOTORS*). Sin embargo, y de forma opcional, es posible también realizar un calibrado de la emisora (*RC CALIBRATION*) con el *rc calibration switch*.

4. *RC CALIBRATION*: calibrado opcional de la emisora.

5. *DISARMED MOTORS*: para el armado de los motores se ha de enviar una señal determinada desde la emisora durante 2 segundos. Desde este estado es posible volver a calibrar los sensores (*SENSOR CALIBRATION*) pulsando el *safety switch*.

6. *ARMING MOTORS*: estado de 2 segundos de duración en el que se arman los motores.

7. *ARMED MOTORS*: tras haber armado los motores ya es posible despegar el dron, activándose los controles. Para volver a *DISARMED MOTORS* hay que enviar otra señal desde la emisora.

8. *FLIGHT*: que se esté en el estado *FLIGHT* o *ARMED MOTORS* simplemente depende de si se está volando o no. Éste es el estado de vuelo manual.

9. *ALTHOLD*: éste es el estado de vuelo manual con control de posición autónomo. Para llegar a él es necesario que el canal de la emisora *althold* esté activo. Para volver a *FLIGHT* es necesario, por seguridad, que se cumplan las siguientes condiciones:

    a. Desactivar *althold*.
    b. Hacer que el *throttle* que se envíe desde la emisora coincida dentro de un rango (de 0,9 a 1,1 veces) con el del control.

10. *POSHOLD*: estado en el que el dron trata de mantener su posición de forma autónoma. Para pasar a él es necesario estar en el estado *ALTHOLD* y que el canal de la emisora *poshold* esté activo. Para volver a *ALTHOLD* es necesario que se cumplan las siguientes condiciones:

    a. Desactivar *poshold*.
    b. Hacer que el *target* del *roll* enviado desde la emisora coincida dentro de un rango (de 0,9 a 1,1 veces) con el del control.
    c. Hacer que el *target* del *pitch* enviado desde la emisora coincida dentro de un rango (de 0,9 a 1,1 veces) con el del control.

Además existe la posibilidad de una parada de seguridad. Si en cualquiera de los tres estados de vuelo (*FLIGHT*, *ALTHOLD* o *POSHOLD*) se activa desde la emisora *stop*, se vuelve al estado *READY*.

5.1.3. Filtro Extendido de Kalman (EKF)

Para el control de la navegación autónoma de un UAV es imprescindible implementar un estimador de estado. Esto es debido a que las medidas de los sensores por separado no son suficientemente fiables, es decir, están sujetas a un error considerable. El estimador de estado más utilizado en sistemas lineales es el Filtro de Kalman (KF), un algoritmo desarrollado por el ingeniero eléctrico, matemático e inventor Rudolf E. Kálmán. Una de las primeras aplicaciones de este filtro fue en el programa Apollo, aunque posteriormente se ha convertido en el estimador de estado por excelencia. Sin embargo, un UAV se caracteriza por ser un sistema no lineal. Es por ello que hemos de utilizar un Filtro Extendido de Kalman (EKF) [33], válido en sistemas no lineales diferenciables en los que los errores de linealización (por aproximaciones de Taylor) sean suficientemente pequeños.

A lo largo de los últimos años se han desarrollado otros estimadores no lineales derivados del Filtro de Kalman, como el "Square Root Unscented Kalman Filter" (SRUKF) [34]. Sin embargo, y a pesar de que algunos de estos métodos demuestran ser más precisos (aunque requieren una mayor potencia de cálculo), en principio se usará el EKF en este proyecto.

En el EKF se integran las medidas de la IMU (giróscopo y acelerómetro) y de las cámaras MCS para corregir los errores individuales en cada una de ellas y así obtener una estimación más precisa de los valores reales. Dos ejemplos claros de la actuación del Filtro Extendido de Kalman serían:

- La corrección ocasional de la medida del giróscopo por parte del acelerómetro, que ocurre cuando éste último detecta una aceleración equivalente a la gravedad en el eje Z, siendo los ángulos de cabeceo (*pitch*) y alabeo (*roll*) cercanos a cero.
- La estimación del ángulo de guiñada (*yaw*) por parte de la IMU, a diferencia del *pitch* y el *roll*, acumula mucho error, por lo que las medidas del sistema MCS ayudan a su corrección.

El EKF cuenta con dos etapas: predicción y actualización. Teniendo en cuenta las covarianzas de la predicción del estado estimado (modelo matemático) y de las medidas de los sensores realiza una estimación ponderada de las variables necesarias para el control.

La formulación general del Filtro Extendido de Kalman es la siguiente [35]:

$$x_t = f(x_{k-1}, u_k) + w_k$$

$$z_k = h(x_k) + v_k$$

Siendo x_t el vector de estados, z_k el vector de observaciones y u_k el vector de control. Por otro lado, w_k y v_k son ruidos que siguen una distribución normal con media cero.

El EKF de este proyecto tiene varios estados [1]:

A. Funcionamiento fiable con la IMU:

- *Deshabilitado:* no se estima ninguna variable. Se sale de este estado una vez se hayan calibrado los sensores.

- *Pitch & Roll:* el Filtro Extendido de Kalman tan solo estima los ángulos de *pitch* (cabeceo) y *roll* (alabeo). Este es el modo de funcionamiento adecuado para vuelo manual cuando solo se tiene como sensor la Unidad de Medida Inercial, ya que no se tendría medida directa del *yaw*, por lo que no sería posible corregirlo de forma automática fiablemente.

B. Funcionamiento fiable con la IMU, el sistema MCS y el sonar:

- *Pitch, Roll & Yaw:* se estiman los ángulos de cabeceo, alabeo y guiñada. Este es el modo de funcionamiento que se ha de utilizar para vuelo manual si se cuenta, además de con la IMU, con algún sensor que permita corregir de manera adecuada el *yaw*, como un magnetómetro o el sistema MCS. La primera opción no fue posible debido a las interferencias electromagnéticas en interiores.

- *Pitch, Roll y Altura:* en este estado el EKF mide, además del alabeo y el cabeceo, la altura. Para ello, al igual que con la guiñada, es necesario contar con un sensor que mida de forma directa la altura del cuadricóptero. En interiores un barómetro no es una buena opción, y ésta es una de las razones por las que se recurrió al sistema de cámaras.

- *Pitch, Roll, Yaw y Altura:* se estiman los ángulos de cabeceo, alabeo y guiñada y la altura del dron.

- *Pitch, Roll, Yaw y Velocidades XY:* en este modo se calculan el *roll*, el *pitch*, el *yaw* y las velocidades horizontales del cuadricóptero.

- *Pitch, Roll, Yaw, Velocidades XY y Altura:* estado igual que el anterior pero con estimación adicional de altura.

- *Posicionamiento general en el espacio:* se estiman los ángulos de *roll*, *pitch* y *yaw*, la posición en el espacio y la velocidad.

5.2. OpenPilot Revolution

En *OpenPilot Revolution* se programó para que realizara las siguientes funciones[3]:

- Comunicaciones UART[4] con la emisora, la GCS y la *Raspberry*.
- Procesamiento de las medidas del sonar[5] y de la IMU[4].
- Control de LEDs y de las salidas PWM.
- Recepción del *safety switch*.

---

[3]Téngase en cuenta que algunas también han sido mencionadas en la sección 5.1 Esto es debido a que se probó de ambas formas. Para más detalle con respecto a esto véase el capítulo 6.

[4]Véase la sección 3.3.

[5]Véase la sección 3.7.

El *OpenPilot* se carga desde *OPENPILOT_REVO.slx*.

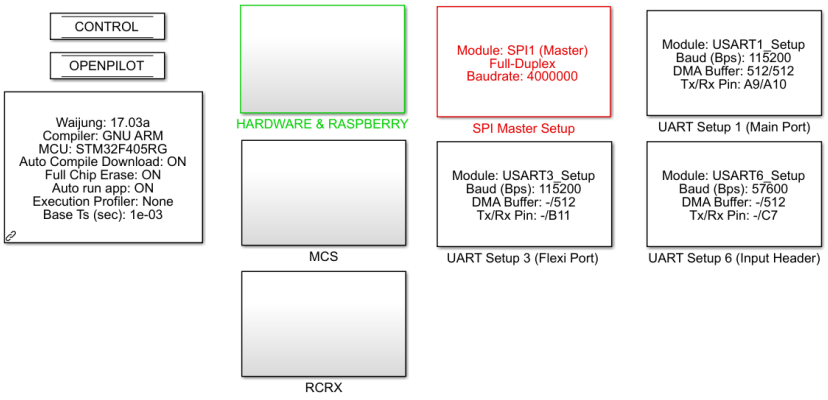

Figura 5.7. Ventana principal de *OPENPILOT_REVO.slx*

5.3. Estación de control de tierra (GCS)

Las principales funciones de la estación de tierra (GCS) en este proyecto han sido:

- Comunicación *MAVLink* con la *Raspberry Pi Zero W*. Esto ha servido para:
  - Monitorización del dron.
  - Cambio de las inercias en el modelo del dron en tiempo real para ir probando aquellas que ofrecían un control más amortiguado.
- Procesamiento de los datos recibidos desde *Motive* y envío por UART de la posición XYZ y los ángulos de Euler del dron al *OpenPilot Revolution*.

Todo esto se realizó desde el archivo *PC_CONTROL_STATION.slx*, cuya ventana principal se muestra a continuación:

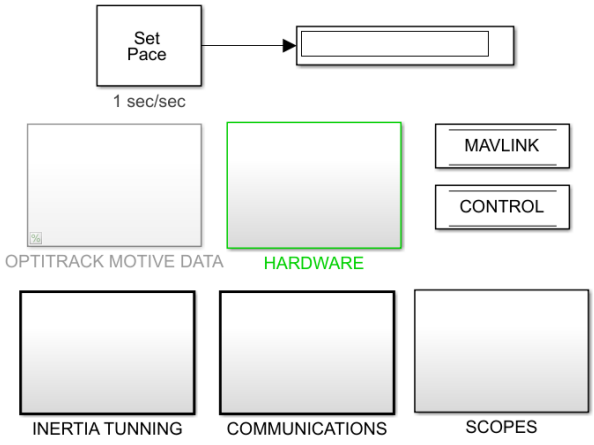

Figura 5.8. Ventana principal de *PC_CONTROL_STATION.slx*

6. Comunicaciones

En este capítulo se explican en profundidad las comunicaciones. Está formado por 2 secciones. En la sección 6.1 se comentan las diferentes comunicaciones UART empleadas en el proyecto. En el punto 6.2 se habla del protocolo *MAVLink*.

6.1. UART

En este proyecto se han empleado diferentes comunicaciones UART entre distintos dispositivos. Además, es conveniente empezar indicando que, en un principio, para vuelo manual, la única comunicación UART utilizada fue la que enviaba la información de la emisora a la *Raspberry*. Posteriormente, debido a que la *Raspberry* no podía soportar dos UARTs diferentes (emisora y GCS (cámaras)), se tuvo que recurrir al *OpenPilot Revolution*. Finalmente se optó por descartar el *OpenPilot* y emplear un *Waveshare Serial Expansion Hat*, que se comunicaba con la *Raspberry* por I^2C.

El esquema de comunicaciones UART al inicio del proyecto fue:

Figura 6.1. Esquema inicial de comunicaciones UART

Posteriormente se pasó al esquema mostrado a continuación:

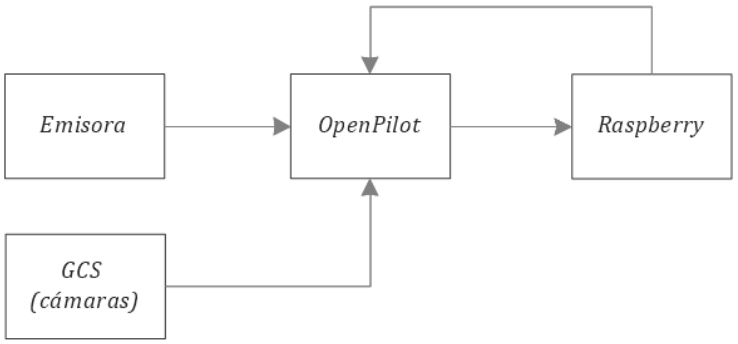

Figura 6.2. Esquema de comunicaciones UART con el *OpenPilot*

Finalmente se empleó el siguiente esquema:

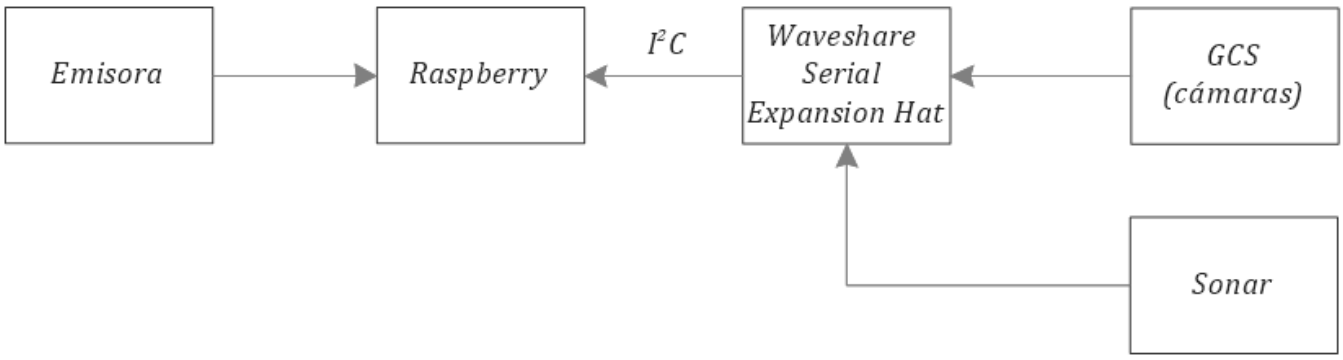

Figura 6.3. Esquema final de comunicaciones UART

6.1.1. Emisora – OpenPilot/Raspberry (protocolo IBUS)

Para la recepción de datos desde la emisora, ya sea en el *OpenPilot* o en la *Raspberry*, es necesario un receptor de radio enlazado con ésta. Este receptor se conecta con el *OpenPilot* o con la *Raspberry*. El protocolo utilizado para el envío de información fue el IBUS, un protocolo serie de *FlySky* que funciona a 115200 baudios y con una cabecera de dos bytes (0x20, 0x40). Los canales de radio enviados fueron (dos bytes por canal):

- *Throttle*: determina la aceleración de las palas.
- *Rudder*: timón, determina el giro de guiñada.
- *Pitch*: para el movimiento proa – popa.
- *Roll*: para el movimiento estribor – babor.
- *Althold*: para activar el mantenimiento de altura autónomo.
- *Poshold*: para activar el mantenimiento de posición autónomo.
- *Stop*: parada de seguridad.

Mensaje emisora – OpenPilot/Raspberry (16 bytes)							
Cabecera (2 bytes)	*Throttle* (2 bytes)	*Rudder* (2 bytes)	*Pitch* (2 bytes)	*Roll* (2 bytes)	*Althold* (2 bytes)	*Poshold* (2 bytes)	*Stop* (2 bytes)

Tabla 6.1. Mensaje emisora – *OpenPilot/Raspberry*

6.1.2. GCS – OpenPilot

Para el envío de los datos recabados por las cámaras desde la GCS al dron en un principio se optó por utilizar el *OpenPilot*, una controladora de vuelo que, a diferencia de la *Raspberry Pi Zero W*, puede soportar varias UART simultáneas. Para esta comunicación se estableció también una cabecera de dos bytes (0x20, 0x40), siendo los datos enviados los mencionados a continuación (cuatro bytes por medida):

- Posición XYZ del dron (tres medidas).
- Ángulos de Euler del dron (tres medidas).

Mensaje GCS – OpenPilot (26 bytes)						
Cabecera (2 bytes)	Posición X (4 bytes)	Posición Y (4 bytes)	Posición Z (4 bytes)	Alabeo (4 bytes)	Cabeceo (4 bytes)	Guiñada (4 bytes)

Tabla 6.2 Mensaje GCS – *OpenPilot*

6.1.3. OpenPilot – Raspberry

Dentro de las comunicaciones entre el *OpenPilot* y la *Raspberry* podemos distinguir tres casos:

1. Utilización de la IMU, de las salidas PWM y de los LEDs del *OpenPilot*. En este caso las comunicaciones son bilaterales:

    - Del *OpenPilot* a la *Raspberry*: con una cabecera de dos bytes (0x20, 0x40), se envían los siguientes datos:

Mensaje OpenPilot – Raspberry (1) (43 bytes)							
Cabecera (2 bytes)	Canales de la emisora (14 bytes)	Posición XYZ (6 bytes)	Ángulos de Euler (6 bytes)	Altura del sonar (2 bytes)	*Safety switch* (1 byte)	Velocidades angulares procesadas de la IMU (6 bytes)	Aceleraciones lineales procesadas de la IMU (6 bytes)

Tabla 6.3 Mensaje *OpenPilot* – *Raspberry* (1)

Ésta comunicación resultó ser demasiado lenta (no se consiguió reducir a menos de 10 ms satisfactoriamente), lo que provocó que el vuelo fuera menos estable que cuándo se utilizaba el *PXFmini*. Es por ello que se descartó.

- De la *Raspberry* al *OpenPilot*: con la misma cabecera, se envían:

Mensaje Raspberry – OpenPilot (11 bytes)					
Cabecera (2 bytes)	Estado actual (1 byte)	PWM 1 (2 bytes)	PWM 2 (2 bytes)	PWM 3 (2 bytes)	PWM 4 (2 bytes)

Tabla 6.4 Mensaje *Raspberry – OpenPilot*

2. Utilización de una IMU externa, conectada por I^2C a la *Raspberry Pi Zero W*, y de las salidas PWM y de los LEDs del *OpenPilot Revolution*. Esto se probó tras comprobar que no era factible enviar los datos de la IMU por UART (por su lentitud). Sin embargo, tampoco arrojó resultados tan satisfactorios como el *PXFmini*. La comunicación también es bilateral, aunque más relajada:

   - Del *OpenPilot* a la *Raspberry*: igual que en el caso anterior con la salvedad de que no se envían las medidas de la IMU:

Mensaje OpenPilot – Raspberry (2) (31 bytes)					
Cabecera (2 bytes)	Canales de la emisora (14 bytes)	Posición XYZ (6 bytes)	Ángulos de Euler (6 bytes)	Altura del sonar (2 bytes)	*Safety switch* (1 byte)

Tabla 6.5 Mensaje *OpenPilot – Raspberry* (2)

   - De la *Raspberry* al *OpenPilot*: igual que en el caso anterior (Tabla 6.4).

3. Utilización de la IMU, de las salidas PWM y de los LEDs del *PXFmini*. En este caso, sin embargo, la comunicación es unilateral, de *OpenPilot* a *Raspberry*. El mensaje es el mismo que el indicado en la Tabla 6.5. Con esta disposición se consiguió la mayor estabilidad de vuelo con la utilización del *OpenPilot*. Sin embargo, resultó ser menos estable que cuando sólo se utilizaban *Raspberry* y *PXFmini*, por lo que se optó finalmente por descartar el *OpenPilot* y emplear un *Waveshare Hat* con dos puertos serie adicionales.

6.1.4. GCS – Waveshare Hat – Raspberry

Como se ha mencionado previamente, finalmente se optó por utilizar un *Waveshare Serial Expansion Hat* para poder prescindir así del *OpenPilot Revolution*. Esto se hizo por dos razones:

- No parecía razonable utilizar una controladora de vuelo para sólo tratar temas de comunicación.
- El *OpenPilot Revolution* no ofrecía la misma calidad de vuelo manual que la obtenida con la utilización del *PXFmini*.

Desde la estación de tierra se decidió enviar al *Waveshare Serial Expansion Hat* lo mismo que lo mencionado en la Tabla 6.2. La información pasa del *Hat* a la *Raspberry* por I^2C.

6.1.5. Sonar – Waveshare Hat – Raspberry

Por otro lado, las medidas del sonar pasaron de enviarse por ADC al *OpenPilot* a comunicarse al *Waveshare Serial Expansion Hat* por UART. Si se desea saber cómo funciona la comunicación UART en el sonar *MB1240* véase el apartado 3.7. Por supuesto, en este caso la información también pasa del *Hat* a la *Raspberry* por I^2C.

6.2. MAVLink

MAVLink, o *Micro Air Vehicle Link*, es un protocolo de comunicación para pequeños UAVs. Éste se usa, fundamentalmente, para la comunicación entre un UAV y una estación de control de tierra (GCS). En este proyecto tiene dos principales funciones:

- Monitorización.
- Cambio de inercias en el modelo en tiempo real.

La estructura de un paquete de *MAVLink* se muestra a continuación (Figura 6.4) [36] (Tabla 6.6) [37] [38]:

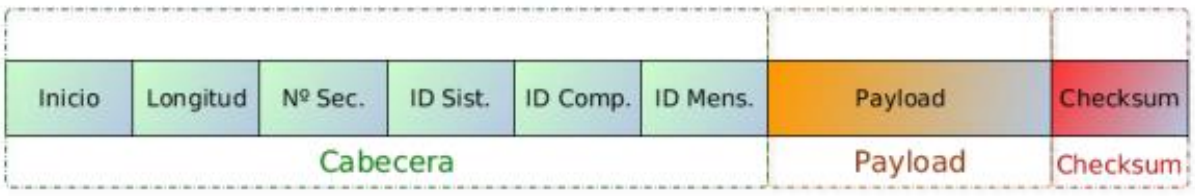

Figura 6.4. Estructura de un paquete de *MAVLink*

Parte del mensaje	Campo	Byte	Descripción
Cabecera	STX	0	Indica el inicio de un nuevo mensaje
	LEN	1	Indica la longitud del *payload* (contenido)
	SEQ	2	Número de secuencia. Sirve para dividir mensajes muy largos y para detectar pérdida de paquetes
	SYS	3	ID del sistema que envía el mensaje. Sirve para diferenciar diferentes comunicaciones *MAVLink* en el mismo entorno
	COMP	4	ID del componente emisor dentro del propio sistema que envía el mensaje
	MSG	5	ID del mensaje enviado. Sirve para que el sistema de recepción sepa codificarlo
Payload	PAYLOAD	6 : n + 6	Contenido del mensaje
Checksum	CKA	n + 7	Checksum LSB. El checksum protege el paquete de ser decodificado de una versión diferente del mismo
	CKB	n + 8	Checksum MSB

Tabla 6.6. Estructura de un paquete de *MAVLink*

Dentro del propio protocolo *MAVLink* se han probado diferentes medios de comunicación:

- UDP: es el más rápido de los tres, aunque el más inseguro. No ocupa periféricos y requiere de conexión WiFi. Fue el más utilizado a lo largo del proyecto debido a que la monitorización no es crítica.

- TCP/IP: más lento que el UDP debido a que en este sí se comprueban los errores. Esto hace que sea un medio más seguro que el UDP, aunque más propenso a ralentizarse. Tampoco ocupa periféricos y también requiere conexión WiFi.

- UART: ocupa el periférico de la UART de la *Raspberry*, necesario para otras comunicaciones. Es por ello que fue rápidamente descartado.

7. Resultados

En este capítulo se muestran los resultados obtenidos. En la sección 7.1 se comentan los resultados de vuelo manual, mientras que en la 7.2 se trata el vuelo autónomo.

La configuración final escogida fue la mostrada a continuación (Figura 7.1):

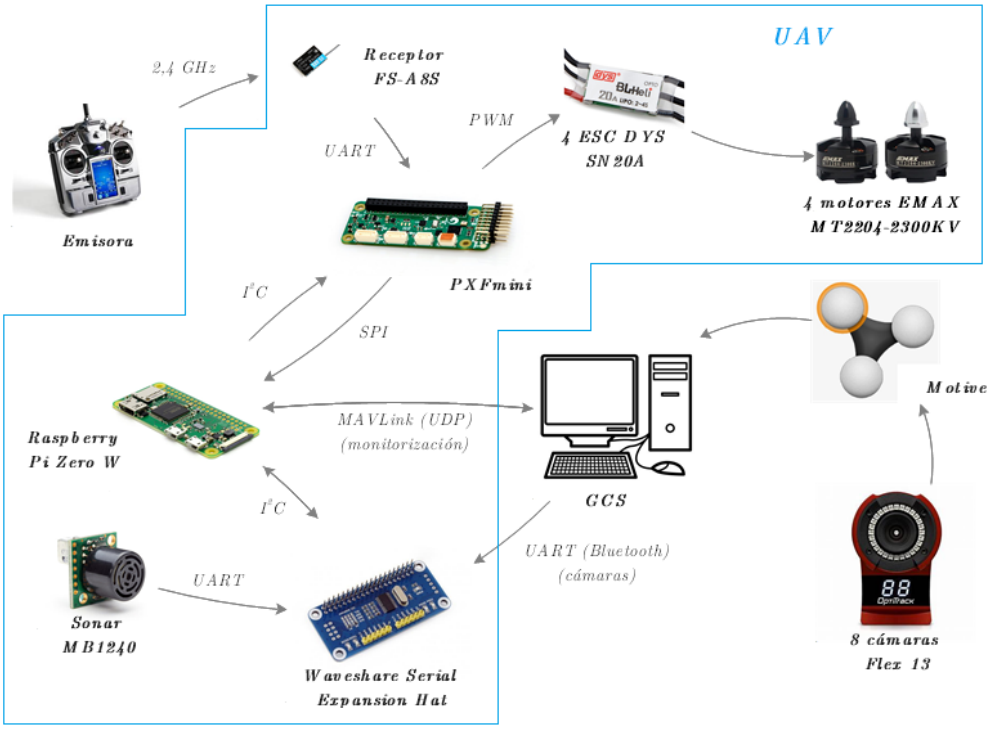

Figura 7.1. Configuración final de vuelo

Con ella se consiguió el vuelo manual más estable, además de que se posibilitaron todas las comunicaciones necesarias para la navegación autónoma.

7.1. Vuelo manual

El vuelo manual del dron acabó funcionando sin ningún problema. Para ello, se ajustaron las inercias (desacopladas del control) en vuelo (por *MAVLink*) para obtener aquellas que otorgaban mayor estabilidad.

Para analizar la navegación manual, que consistió en el control del *roll* y el *pitch* (el *yaw* no era estimado correctamente por el EKF, siendo ésta una de las razones por las que se recurrió a las cámaras), se pueden comparar las diferencias entre la referencia (en rojo) y la estimación (en azul) del EKF en ángulos y velocidades angulares (Figuras 7.2 y 7.3):

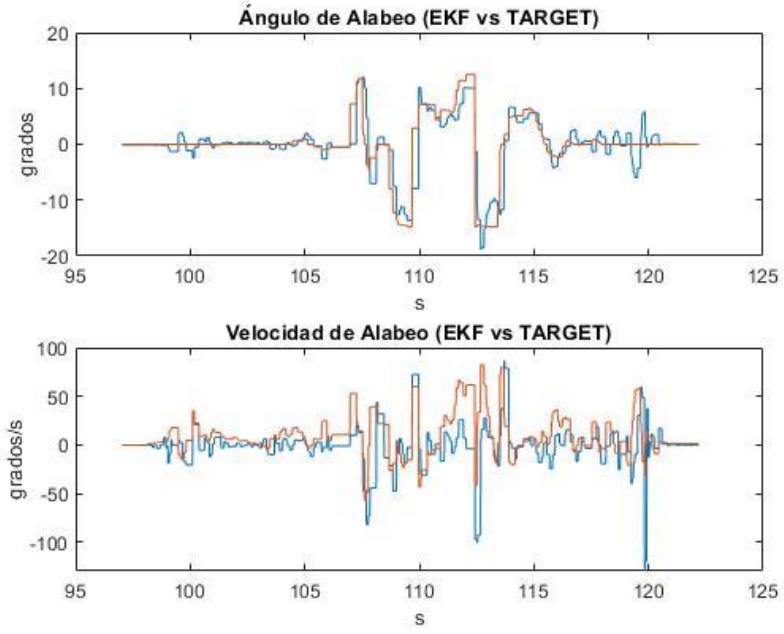

Figura 7.2. Ángulo y velocidad de alabeo (estimación y referencia)

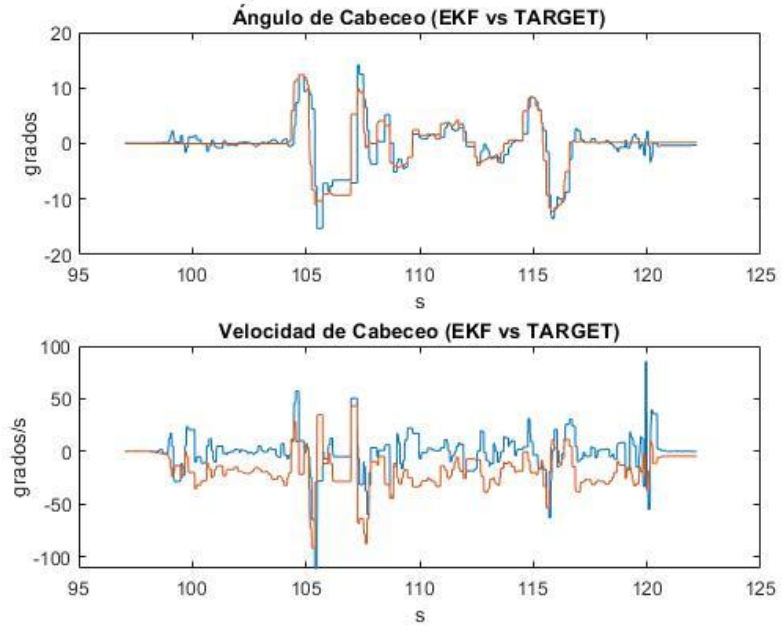

Figura 7.3. Ángulo y velocidad de cabeceo (estimación y referencia)

En ambas figuras (7.2 y 7.3) se puede comprobar como el control de los ángulos de *roll* y *pitch* funciona a la perfección. En las velocidades angulares se puede observar cierto *offset*. Esto se debe a que, a diferencia del control de ángulos, formado por controladores de tipo Proporcional-Integral (PI), el control de velocidades angulares cuenta con controladores Proporcional-Derivativo (PD), que no anulan por completo el error de perturbación.

Para comprobar que la estimación de ángulos realizada por el EKF es la correcta también se pueden comparar los ángulos de Euler estimados (en rojo) con los medidos por las cámaras (en azul) (Figura 7.4):

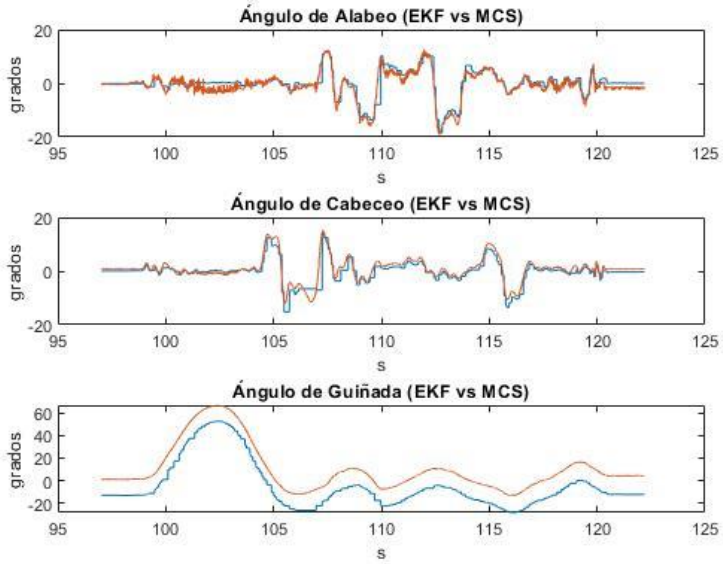

Figura 7.4. Ángulos de Euler (estimación y cámaras)

Como se puede observar en la Figura 7.4, el *roll* y el *pitch* son estimados a la perfección. Las variaciones del *yaw* también se estiman correctamente, pero existe un *offset* entre el ángulo estimado y el medido por el sistema MCS. Esto se debe a que el eje de coordenadas del sistema de cámaras es fijo, y a la hora de realizar la prueba no se alineó perfectamente con el del dron. Si el vuelo hubiera sido más extenso la estimación del *yaw* hubiera empezado a derivar: es ésta una de las razones por las que se ha introducido el sistema MCS.

Además, también se puede mostrar la posición medida por las cámaras a lo largo del vuelo, que es enviada en la navegación autónoma al dron (Figura 7.5):

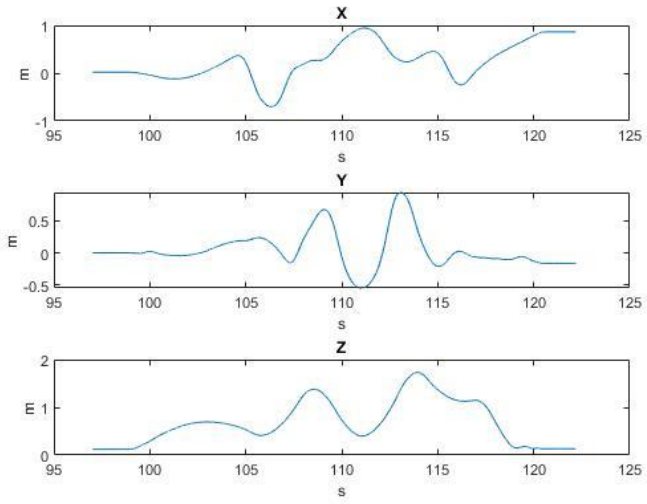

Figura 7.5. Posición del dron durante el vuelo (cámaras)

Los PWM, calculados a partir de las fuerzas unitarias de *roll*, *pitch* y *yaw* (salidas del control de estabilización), así como del *throttle* enviado directamente desde la emisora, enviados a los motores a lo largo del vuelo fueron (Figura 7.6):

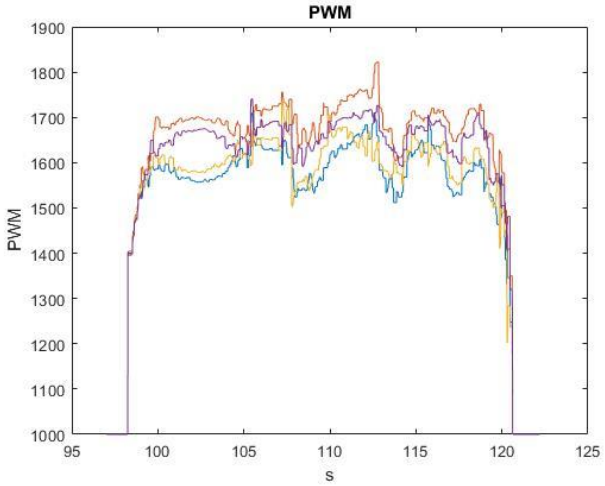

Figura 7.6. PWM enviados a los motores en el vuelo manual

7.2. Vuelo autónomo

Finalmente no se consiguió realizar un vuelo autónomo, esto se debió principalmente a dos factores:

- A lo largo del proyecto se acabó invirtiendo mucho tiempo en la incorporación del *OpenPilot Revolution* como sustituto del *PXFmini* para así contar con suficientes puertos serie para la navegación autónoma. Sin embargo, las pruebas de vuelo con el *OpenPilot* no llegaron a resultar satisfactorias, estando el vuelo manual menos amortiguado que con el *PXFmini*. Por ello se decidió reincorporar el *PXFmini*, descartando el *OpenPilot*, y emplear el *Waveshare Serial Expansion Hat*. En última instancia no se consiguió que las comunicaciones UART entre la GCS y el *Waveshare Serial Expansion Hat* funcionaran correctamente debido a una incompatibilidad entre éste y el *PXFmini*.

- Para lograr un vuelo autónomo satisfactorio también sería necesario realizar una etapa de calibración de las medidas de las cámaras.

En ambos problemas se profundizará más en la sección 8.2 (futuros desarrollos).

En esta sección, sin embargo, se muestran los resultados obtenidos en una simulación de la navegación autónoma. La misión en esta simulación consta de cuatro fases:

1. Se empieza en el eje de coordenadas (0, 0, 0). La primera orden que se le da al dron, a los 10 segundos, es que ascienda hasta una altura de 1 metro (0, 0, 1).
2. Posteriormente, al segundo 20, se le ordena al UAV que se mueva 1 metro en el eje X (1, 0, 1).
3. La tercera orden, en el segundo 35, consiste en que el dron se mueva a la coordenada (1, 1, 1).
4. Por último, ya en el segundo 50, el dron debe girar 25º con respecto al eje Z (guiñada).

La posición del dron en cada eje, junto a su referencia correspondiente, obtenidas en la simulación se pueden observar a continuación (Figura 7.7):

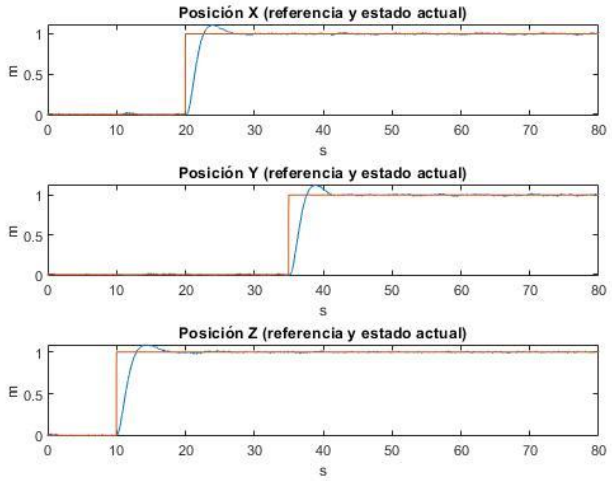

Figura 7.7. Posición (actual y referencia) del UAV en la simulación

Como se puede observar en la Figura 7.7, el control funciona a la perfección: el dron alcanza la posición deseada en unos pocos segundos. Las velocidades, también junto a sus referencias, obtenidas se muestran en la Figura 7.8:

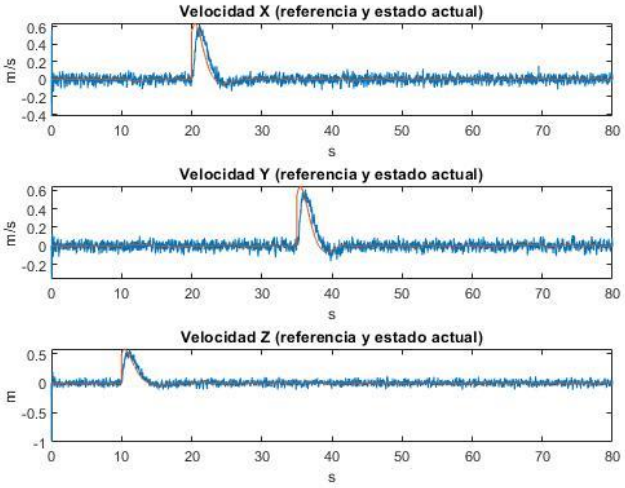

Figura 7.8. Velocidad (actual y referencia) del UAV en la simulación

Por otro lado, en la Figura 7.9 se puede observar como la orden de giro de guiñada del dron funciona también a la perfección en la simulación:

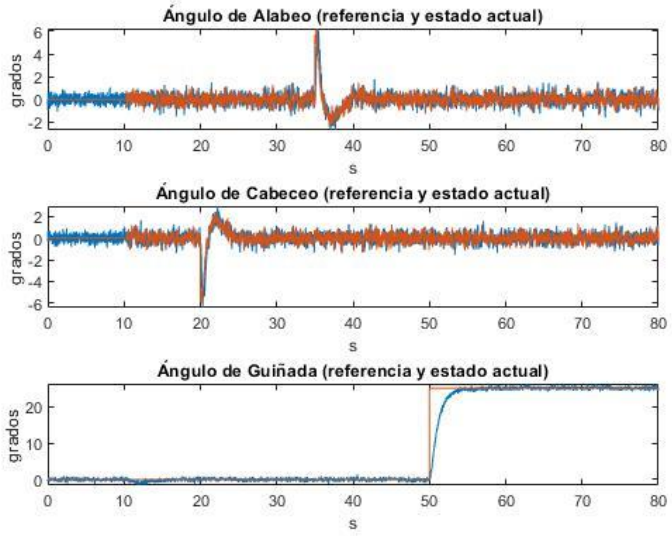

Figura 7.9. Ángulos de Euler (actuales y referencias) del UAV en la simulación

En la Figura 7.9 se puede apreciar también como las referencias de alabeo y cabeceo sólo varían sustancialmente cuando se le ordena al dron un cambio de posición:

- Cuando el UAV ha de moverse en la dirección positiva del eje X el ángulo de cabeceo se vuelve negativo.
- Si se le ordena al dron que se mueva en la dirección positiva del eje Y, el ángulo de alabeo se vuelve positivo.

8. Conclusiones y futuros desarrollos

En este capítulo se encuentra la conclusión final del proyecto (sección 8.1), enumerándose también posibles futuros desarrollos que, no habiéndose podido completar a lo largo de este proyecto, harían posible la navegación autónoma (sección 8.2).

8.1. Conclusiones

A lo largo de este proyecto se han logrado una gran cantidad de hitos. Por un lado, se ha conseguido optimizar el vuelo manual, haciéndolo más estable de lo que era previamente en [1]. Además, se han probado una gran variedad de configuraciones para hallar aquella que, posibilitando la navegación autónoma, ofrecía la mayor calidad de vuelo. Para ello, se han tenido que probar múltiples componentes de hardware, como el *PXFmini*, el *OpenPilot Revolution* o el *Waveshare Serial Expansion Hat*. Con todo ello se ha conseguido tener suficientes puertos UART en el dron, posibilitando así comunicaciones con la GCS (para transmitir información del sistema de cámaras), con la emisora y con el sonar. Se han conseguido resultados satisfactorios en cuanto a la simulación de la navegación autónoma y los resultados de vuelo manual obtenidos son buenos. Por otro lado, se ha programado gran parte de lo que restaba del control para facilitar el vuelo autónomo. Sin embargo, y a falta de solventar algunos pequeños problemas que se explicarán más adelante (sección 8.2), no se ha conseguido que el dron volase autónomamente.

8.2. Fututos desarrollos

En esta sección de mencionan los futuros desarrollos necesarios para que el dron vuelo autónomamente.

8.2.1. Incompatibilidad PXFmini - Waveshare Serial Expansion Hat

Se llegó a la conclusión de que hay incompatibilidad entre el *PXFmini* y el *Waveshare Serial Expansion Hat*. Para solucionarlo, se propone incorporar de nuevo una IMU externa, así como un controlador de servos, descartando el *PXFmini*, y seguir usando el *Waveshare*

Serial Expansion Hat para tener así suficientes puertos UART. En la Figura 8.1 se muestra el esquema de la configuración propuesta:

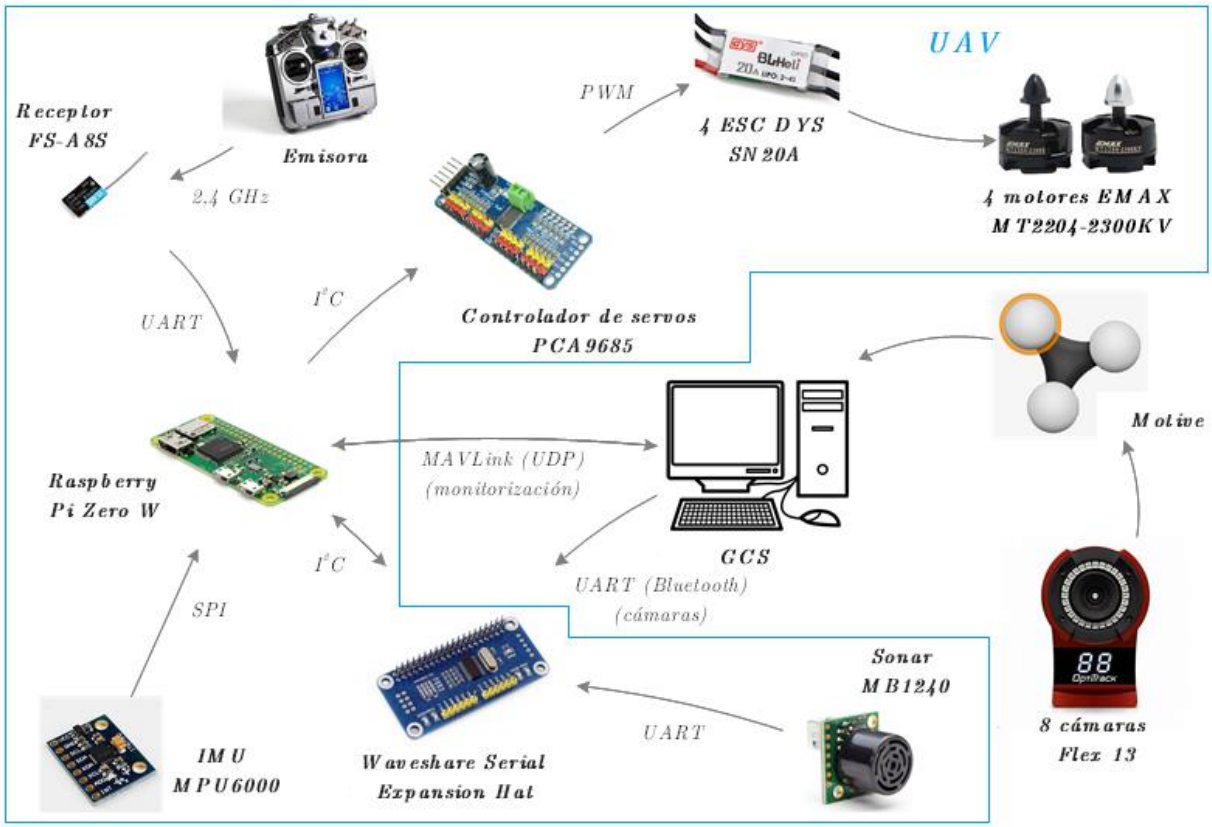

Figura 8.1. Configuración hardware propuesta

8.2.2. Etapa de calibración para las medidas de las cámaras

Por último, es también necesario programar una etapa de calibración para las medidas del sistema MCS (ángulos de Euler y posiciones). Durante el estado de *SENSOR CALIBRATION* se habría de realizar lo siguiente:

- Según sea el ángulo de guiñada detectado por las cámaras (éstas tienen un eje de coordenadas fijo), en la calibración se ha de aplicar un giro al eje de coordenadas (en el eje Z). Con esto se ha de lograr que, tras la calibración y antes de despegar, el ángulo de guiñada sea nulo, coincidiendo así con el de la IMU calibrada. Este giro ha de aplicarse tanto a posición como a ángulos a lo largo de todo el vuelo.

- Tras haberse realizado el giro correspondiente, es necesario que se calcule el *offset* de cada una de las medidas durante la calibración y que éste posteriormente se reste a las mismas.

8.2.3. Otros desarrollos

Tras haberse realizado los dos ajustes previamente mencionados el dron ya debería ser capaz de volar autónomamente sin ningún problema. Cuando esto se haya logrado, el dron de vuelo autónomo podría formar parte de la asignatura de *Control Avanzado*. Con ello, y como se menciona en [1], se podrían implementar las siguientes actividades:

- Realización de una carrera contrarreloj en un circuito predeterminado.
- Minimización del error de seguimiento en trayectorias previamente definidas.
- Realizar la programación necesaria para que el dron mantenga un objeto encima suyo durante el vuelo.

Referencias

[1] J. García Aguilar, "Desarrollo de sistemas de navegación autónoma para un UAV," de *Trabajo de Fin de Máster - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2018.

[2] D. Menéndez Botella, "Desarrollo de un sistema de navegación autónoma en interiores para un cuadricóptero," de *Trabajo de Fin de Máster - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2017.

[3] L. Sánchez-Quiñones Campuzano, "Control de un cuadricóptero para seguimiento automático de personas," de *Trabajo de Fin de Grado - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2017.

[4] N. González García, "Control de un cuadricóptero para navegación en interiores usando un sensor de flujo óptico," de *Trabajo de Fin de Grado - Escuela Técnica Superior de Ingeniería (ICAI)*, Madrid, 2016.

[5] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar y H. S. Al-Khalifa, "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances," de *Sensors (Basel)*, vol. 16, no. 5, pp. 707, 2016.

[6] T. Minh Nguyen, A. Hanif Zaini, K. Guo y L. Xie, "An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied environments," de *International Micro Air Vehicles Conference*, pp. 1-6, 2016.

[7] L. Zhang, J.-Z. Lai, P. Shi, S. Bao y L. Jian-Ye, "An improved MCS/INS integrated navigation algorithm for multi-rotor UAV indoor flight," de *IEEE 2016 Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 2099-2104, 2016.

[8] F. Yu, G. Chen, N. Fan, Y. Song y L. Zhu, "Autonomous Flight Control Law for an Indoor UAV Quadrotor," de *IEEE 2017 29th Chinese Control and Decision Conference (CCDC)*, pp. 6767-6771, 2017.

[9] *The astounding athletic power of quadcopters / Rafaello D'Andrea (TED Talk)*. https://www.youtube.com/watch?v=w2itwFJCgFQ (último acceso: 21/05/2019).

[10] *OptiTrack*. https://sites.google.com/a/nd.edu/discoverlab/robot-platform/optitrack (último acceso: 21/05/2019).

[11] *OTUS Tracker*. https://www.rcbenchmark.com/products/otus-tracker (último acceso: 21/05/2019).

[12] S. Grzonka, G. Grisetti y W. Burgard, "Towards a Navigation System for Autonomous Indoor Flying," de *IEEE 2009 International Conference on Robotics and Automation (ICRA '09)*, pp. 1679-1684, 2009.

[13] R. Li, J. Liu, L. Zhang y Y. Hang, "LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments," de *IEEE 2014 DGON Inertial Sensors and Systems Symposium (ISS)*, pp. 1-15, 2014.

[14] *LIDAR 3D High-Definition Surveying.* https://www.flatironsinc.com/services/lidar.php (último acceso: 21/05/2019).

[15] *Eyesee: the drone allowing to automate inventory in warehouses.* https://www.hardis-group.com/en/our-activities/logistics-solutions/eyesee-drone-allowing-automate-inventory-warehouses (último acceso: 21/05/2019).

[16] *Warehouse inventory using drones: GEODIS and DELTA DRONE have entered the industrialization production phase of their completely automated solution.* https://geodis.com/fr-en/newsroom/press-releases/warehouse-invenotry-using-drones-geodis-and-delta-drone-have-entered-the-industrialization-production-phas-of-their-completely-automated-solution (último acceso: 21/05/2019).

[17] R. J. D'Angelo y R. Cooper Levin, "Design of an Autonomous Quadrotor UAV for Urban Search and Rescue," de *Major Qualifying Project - Worcester Polytechnic Institute*, Worcester, 2011.

[18] C. Sampedro, A. Rodríguez-Ramos, H. Bavle, A. Carrio, P. de la Puente y P. Campoy, "A Fully-Autonomous Aerial Robot for Search and Rescue Applications in Indoor Environments using Learning-Based Techniques," de *Journal of Intelligent & Robotic Systems*, pp. 1-27, 2018.

[19] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Hürzeler y R. Siegwart, "A UAV System for Inspection of Industrial Facilities," de *IEEE Aerospace Conference (AERO 2013)*, pp. 1-8, 2013.

[20] *Raspberry Pi Zero W.* https://www.raspberrypi.org/products/raspberry-pi-zero-w/ (último acceso: 21/05/2019).

[21] *Raspberry Pi Zero W pins.* https://cdn.sparkfun.com/assets/learn_tutorials/6/7/4/PiZero.png (último acceso: 21/05/2019).

[22] *PXFmini.* http://ardupilot.org/copter/_images/pxfmini.jpg (último acceso: 21/05/2019).

[23] *OpenPilot Revolution.* https://opwiki.readthedocs.io/en/latest/_images/revo-ports-2.jpg (último acceso: 30/05/2019).

[24] *Flex 13.* https://www.optitrack.com/products/flex-13/ (último acceso: 21/05/2019).

[25] *Flex 13 specifications.* https://www.optitrack.com/products/flex-13/specs.html (último acceso: 21/05/2019).

[26] *Drone Typology.* https://api.ning.com/files/zcq5CJIee48StFpESnJ43E3VQA3q4y3UvHVkF1q77WBpvblcG9kpM16s8PElc5BfRugDGsyEpYOTGbVttzvCs2TN6-7kd*kW/DroneTypology.jpg (último acceso: 21/05/2019).

[27] *Quad + and X structures.* https://qph.fs.quoracdn.net/main-qimg-006f50df374059a3fd371cb0ae152f6e (último acceso: 21/05/2019).

[28] *Quad H structure.* https://discuss.ardupilot.org/uploads/default/original/2X/3/374631d1fc12eabbe7bbb8e97bd68628295f3163.jpg (último acceso: 21/05/2019).

[29] *Sonar MB1240 XL-MaxSonar-EZ4.* https://cdn.instructables.com/FGJ/JZS1/HJKBVFQN/FGJJZS1HJKBVFQN.MEDIUM.jpg?width=614 (último acceso: 21/05/2019).

[30] *MPU 6050.* https://www.makerlab-electronics.com/my_uploads/2015/05/mpu-6050-1.jpg (último acceso: 23/05/2019).

[31] *Waveshare Serial Expansion Hat.* https://www.waveshare.com/img/devkit/accBoard/Serial-Expansion-HAT/Serial-Expansion-HAT-size.jpg (último acceso: 23/05/2019).

[32] *Waijung Software Installation.* http://waijung.aimagin.com/software_installation.htm (último acceso: 21/05/2019).

[33] C. Luo, S. I. McClean, G. Parr, L. Teacy y R. De Nardi, "UAV Position Estimation and Collision Avoidance Using the Extended Kalman Filter," de *IEEE Transactions on Vehicular Technology*, vol. 62, no. 6, pp. 2749-2762, 2013.

[34] J. Goslinski, W. Giernacki y A. Krolikowski, "A Nonlinear Filter for Efficient Attitude Estimation of Unmanned Aerial Vehicle (UAV)," de *Journal of Intelligent & Robotic Systems*, pp. 1-17, 2018.

[35] *Extended Kalman Filter.* https://en.wikipedia.org/wiki/Extended_Kalman_filter (último acceso: 21/05/2019).

[36] Paquete *MAVLink*. https://www.xdrones.es/wp-content/uploads/2017/02/paquete-mavlink-1.png (último acceso: 30/05/2019).

[37] *MAVLink: protocolo de comunicación para drones*. https://www.xdrones.es/mavlink/ (último acceso: 21/05/2019).

[38] *MAVLink*. https://erlerobotics.gitbooks.io/erlerobot/es/mavlink/mavlink.html (último acceso: 21/05/2019).

Anexo

Attitude Model

Considerando las siguientes variables y parámetros:

L_r	Longitud de *roll*
L_{pf}	Longitud de *pitch* delantera
L_{pr}	Longitud de *pitch* trasera
$u_2 = \frac{T_1+T_2-T_3-T_4}{T_b}$	Fuerza unitaria de *roll*
$u_3 = \frac{T_1-T_2\times\frac{L_{pr}}{L_{pf}}-T_3\times\frac{L_{pr}}{L_{pf}}+T_4}{T_b}$	Fuerza unitaria de *pitch*
$u_2 = \frac{T_1-T_2+T_3-T_4}{T_b}$	Fuerza unitaria de *yaw*
I	Matriz de inercias

Siendo la matriz de pares base la definida a continuación:

$L = \begin{bmatrix} T_b L_r & 0 & 0 \\ 0 & T_b L_{pf} & 0 \\ 0 & 0 & T_b L_{pr} \end{bmatrix}$	Matriz de pares base

Y las matrices de Euler de velocidades y aceleraciones angulares las siguientes:

$E = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\varphi & s_\varphi c_\theta \\ 0 & -s_\varphi & c_\varphi c_\theta \end{bmatrix}$	Matriz de cuerpo a tierra de velocidades
$\dot{E} = \begin{bmatrix} 0 & 0 & -c_\theta \dot{\theta} \\ 0 & -s_\varphi \dot{\varphi} & c_\varphi c_\theta \dot{\varphi} - s_\varphi s_\theta \dot{\theta} \\ 0 & -c_\varphi \dot{\varphi} & -s_\varphi c_\theta \dot{\varphi} - c_\varphi s_\theta \dot{\theta} \end{bmatrix}$	Matriz de cuerpo a tierra de aceleraciones

Podemos asumir lo siguiente:

$$\frac{d(I\cdot\overline{w})}{dx}=\sum T=L\cdot u\rightarrow\frac{d\overline{w}}{dx}=\dot{\overline{w}}=I^{-1}[L\cdot u-\overline{w}\times(I\cdot\overline{w})]=\dot{E}\begin{bmatrix}\dot{\varphi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}+E\begin{bmatrix}\ddot{\varphi}\\\ddot{\theta}\\\ddot{\psi}\end{bmatrix}\rightarrow\begin{bmatrix}\ddot{\varphi}\\\ddot{\theta}\\\ddot{\psi}\end{bmatrix}=$$

$$E^{-1}\left[\dot{\overline{w}}-\dot{E}\begin{bmatrix}\dot{\varphi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}\right]=u' \qquad \text{Cálculos intermedios}$$

$$\overline{w}=E\begin{bmatrix}\dot{\varphi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}\rightarrow\begin{bmatrix}\dot{\varphi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}=E^{-1}\overline{w} \qquad \text{Cálculos intermedios}$$

Así, el vector u' sería:

$$u'=\begin{bmatrix}u_2'\\u_3'\\u_4'\end{bmatrix}=E^{-1}\left[I^{-1}[L\cdot u-\overline{w}\times(I\cdot\overline{w})]-\dot{E}E^{-1}\overline{w}\right]=\begin{bmatrix}\ddot{\varphi}\\\ddot{\theta}\\\ddot{\psi}\end{bmatrix} \qquad \text{Ac. de } \textit{roll, pitch} \text{ y } \textit{yaw} \text{ de ref.}$$

Y utilizando estas tres matrices auxiliares:

$$MA_1=L^{-1}IE \qquad \text{Matriz auxiliar 1}$$

$$MA_2=L^{-1}I \qquad \text{Matriz auxiliar 2}$$

$$MA_3=L^{-1}I\dot{E}E^{-1} \qquad \text{Matriz auxiliar 3}$$

Podemos finalmente despejar los valores de salida deseados:

$$u=\begin{bmatrix}u_2\\u_3\\u_4\end{bmatrix}=MA_1u'+MA_2\left[\begin{bmatrix}w_x\\w_y\\w_z\end{bmatrix}\times\left(I\cdot\begin{bmatrix}w_x\\w_y\\w_z\end{bmatrix}\right)\right]+MA_3\begin{bmatrix}w_x\\w_y\\w_z\end{bmatrix} \qquad \text{Fuerzas un. de } \textit{roll, pitch} \text{ y } \textit{yaw}$$

Nótese que u está en función de los valores de entrada $(u_2', u_3', u_4', w_x, w_y, w_z, \varphi, \theta)$ y de parámetros predefinidos (L, I).

Navigation Model

Considerando las siguientes variables y parámetros, y teniendo en cuenta que aproximamos $fb_x \sim 0$ y $fb_y \sim 0$ en un cuadricóptero:

$T_b = m \times g$	Empuje base
$T_{ref} = -\frac{fb_Z}{T_b} = \frac{T_1+T_2+T_3+T_4}{T_b} = u_1$	Empuje unitario total negativo de referencia
$fb_x \sim 0$	Fuerza en el eje X del cuerpo
$fb_y \sim 0$	Fuerza en el eje Y del cuerpo
fb_Z	Fuerza en el eje Z del cuerpo

Y siendo la matriz de conversión de Euler de aceleraciones lineales (con $\cos\alpha = c_\alpha$ y $\sin\alpha = s_\alpha$):

$$DCM_{b-e} = \begin{bmatrix} c_\theta c_\psi & s_\varphi s_\theta c_\psi - c_\varphi s_\psi & c_\varphi s_\theta c_\psi + s_\varphi s_\psi \\ s_\psi c_\theta & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi \\ -s_\theta & s_\varphi c_\theta & c_\varphi c_\theta \end{bmatrix}$$ Matriz de cuerpo a tierra

$$DCM_{b-e,lin} = \begin{bmatrix} c_\psi & \varphi_{ref}\,\theta_{ref} c_\psi - s_\psi & \theta_{ref} c_\psi + \varphi_{ref} s_\psi \\ s_\psi & \varphi_{ref}\,\theta_{ref} s_\psi + c_\psi & \theta_{ref} s_\psi - \varphi_{ref} c_\psi \\ -\theta_{ref} & \varphi_{ref} & 1 \end{bmatrix}$$ Matriz linealizada

Podemos decir que la aceleración lineal vista desde tierra es:

$$[\ddot{x}\ddot{y}\ddot{z}]_{earth} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + DCM_{b-e} \times \frac{1}{m} \times \begin{bmatrix} fb_x \\ fb_y \\ fb_z \end{bmatrix} \sim \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + DCM_{b-e,lin} \times \frac{1}{m} \times \begin{bmatrix} 0 \\ 0 \\ fb_z \end{bmatrix}$$ Aceleración

Desarrollando cada una obtenemos:

$\ddot{x}_{earth} = g \times (-s_\psi \varphi_{ref} T_{ref} - c_\psi \theta_{ref} T_{ref})$	Aceleración en el eje X vista desde tierra
$\ddot{y}_{earth} = g \times (c_\psi \varphi_{ref} T_{ref} - s_\psi \theta_{ref} T_{ref})$	Aceleración en el eje Y vista desde tierra

$\ddot{z}_{earth} = g \times (-T_{ref} + 1)$ Aceleración en el eje Z vista desde tierra

Finalmente, despejando los valores de salida deseados:

$$\varphi_{ref} = \frac{-\frac{\ddot{x}_{earth}}{g} \times s_{\psi} + \frac{\ddot{y}_{earth}}{g} \times c_{\psi}}{1 - \frac{\ddot{z}_{earth}}{g}}$$ Ángulo de alabeo de referencia

$$\theta_{ref} = -\frac{\frac{\ddot{x}_{earth}}{g} \times c_{\psi} + \frac{\ddot{y}_{earth}}{g} \times s_{\psi}}{1 - \frac{\ddot{z}_{earth}}{g}}$$ Ángulo de cabeceo de referencia

$$T_{ref} = 1 - \frac{\ddot{z}_{earth}}{g} = u_1$$ Empuje unitario total negativo de referencia

Como se puede observar, $\varphi_{ref}, \theta_{ref}$ y u_1 están solo en función de los valores de entrada ($\ddot{x}_{ref}, \ddot{y}_{ref}, \ddot{z}_{ref}, \psi$) y la aceleración de la gravedad (g).

PARTE II:

PRESUPUESTO

Elemento	**Precio unitario aproximado (€)**	**Unidades**	**Precio total aproximado (€)**
Raspberry Pi Zero W	11	1	11
PXFmini	62	1	62
OpenPilot Revolution	48	1	48
Waveshare Serial Expansion Hat	17	1	17
Sonar MB1240 XL-MaxSonar-EZ4	40	1	40
IMU MPU-6050	3	1	3
Motor EMAX MT2204-2300KV	15	4	60
Hélice 5030	0,25	4	1
ESC DYS SN20A	11	4	44
Receptor FS-A8S	8	1	8
Batería LiPo	6	1	6
Emisora Turnigy i10	240	1	240
Cámara Flex 13	891	8	7128
OptiHub	267	3	801
Licencia Motive Tracker	891	1	891
CW-500 Calibration Wand	267	1	267
CS-200 Calibration Square	133	1	133
Licencia MatLab/Simulink	800	1	800
Horas de trabajo	10	300	3000
TOTAL	**13560 €**		

Tabla 1. Presupuesto del proyecto