Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Sistema de coordinación y control de múltiples vehículos aéreos no tripulados en testbed de interiores

Autor: Raúl Zahínos Marín
Tutor: Aníbal Ollero Baturone

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Sistema de coordinación y control de múltiples vehículos aéreos no tripulados en testbed de interiores

Autor:
Raúl Zahínos Marín

Tutor:
Aníbal Ollero Baturone
Catedrático

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo de Fin de Grado: Sistema de coordinación y control de múltiples vehículos aéreos no tripulados en testbed de interiores

Autor: Raúl Zahínos Marín

Tutor: Aníbal Ollero Baturone

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Me gustaría empezar dando las gracias a todos los docentes que han participado en mi formación. Especialmente, quiero agradecer a mi tutor, Aníbal, la atención y los consejos que me ha dado durante la redacción de este trabajo.

Le debo también un enorme agradecimiento a Miguel Ángel Trujillo, en primer lugar, por proponerme este proyecto, pero sobre todo, por la motivación que ha sabido transmitirme en aquellos momentos en los que se me hizo cuesta arriba el camino. También tengo que darle las gracias a Jesús Parra, que ha conseguido que me sintiera como en casa desde que llegué a CATEC.

Gracias a mis amigos de toda la vida y a los nuevos que han surgido en estos cuatro años. Espero seguir compartiendo momentos a vuestro lado durante muchos años más.

Finalmente, doy las gracias a mis padres, que han sido quienes me han dado todo en esta vida; a mi hermana, por demostrarme que puedo contar con ella para cualquier cosa; a ellos y al resto de mi familia, por el cariño que siempre me han dado.

Raúl Zahínos Marín

Sevilla, 2020

Resumen

En este trabajo se ha llevado a cabo la puesta en marcha de un sistema que permite volar simultáneamente un elevado número de vehículos aéreos no tripulados en un entorno controlado. El propósito de dicho sistema es servir de infraestructura que habilite futuras investigaciones relacionadas con los sistemas multi-UAV, permitiendo llevar a cabo pruebas de concepto para la validación experimental de nuevos algoritmos.

Los UAVs empleados se encuentran disponibles comercialmente y son especialmente adecuados para este fin, por tratarse de una plataforma de código abierto. A pesar de esto, su uso como sistema multi-UAV no es inmediato y está fuertemente condicionado por el hardware adicional del que se disponga. En nuestro caso, disponemos de un sistema de posicionamiento en interiores que nos permite obtener la localización de cada dron de manera precisa.

A la fecha de este documento el sistema cuenta con 10 UAVs, sin embargo, es previsible que en el futuro se plantee incrementar esta cifra. Por este motivo, desde el comienzo se ha tenido presente el asunto de la escalabilidad. Esto plantea ciertos retos, tales como la imposibilidad de hacer uso de tantas antenas como UAVs cuando la cantidad de vehículos comience a ser significativa. De dicha restricción se han derivado numerosos problemas, por lo que ha sido necesario adquirir un profundo conocimiento del sistema para identificar las causas y buscar soluciones. En este documento se detalla todo el proceso y se justifican las decisiones tomadas. Finalmente, se presentan resultados experimentales con hasta 9 UAVs.

Abstract

In this project, it has been carried out the set-up of a system that enables the simultaneous fly of a large number of Unmanned Aerial Vehicles (UAVs) in a structured environment. The main goal is to serve as an infrastructure for future researches related to multi-UAV systems, allowing the execution of proof-of-concept tests to validate new algorithms.

The UAVs employed are commercially available and especially well-suited for this end, as they constitute an open-source platform. Despite this, their usage as a multi-UAV system is not an out-of-the-box feature and is strongly conditioned by the additional hardware that is available. In our case, we use an indoor positioning system that provides the localization of each drone with high accuracy.

At the time of writing this document the system has 10 UAVs. However, this number is expected to be increased in the future. For this reason, we have kept in mind the scalability issue from the beginning. It poses certain challenges, such as the impossibility of using as many antennas as UAVs when the number of these becomes higher. Several problems arose because of this constraint; therefore, a deep understanding of the system was required to identify the causes and propose solutions. In this document, we detail the whole process and justify the decisions taken. Finally, experimental results are provided for up to nine UAVs.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En los últimos años ha crecido el interés por los sistemas multi-UAV y sus potenciales aplicaciones, lo que ha dado lugar a un alto número de publicaciones en esta materia. Sin embargo, debido a las dificultades técnicas que entrañan este tipo de sistemas, gran parte de ellas están aún restringidas al ámbito teórico y de simulación.

El centro de investigación CATEC cuenta con una reconocida experiencia en el desarrollo de nuevas tecnologías, caracterizándose además por perseguir un alto nivel de madurez tecnológica en sus proyectos. Como no fueron ajenos a esta tendencia, desarrollaron en sus instalaciones un sistema que permitiera el vuelo simultáneo de múltiples vehículos aéreos, con el propósito de validar nuevos algoritmos mediante experimentos reales, elevando así su nivel TRL[1]. Dicho sistema ha sido de gran utilidad desde entonces, pero con el paso de los años, la inevitable degradación de los componentes físicos y la aparición de nuevas herramientas software obligan a preguntarse si vale la pena mantenerlo (reparándolo y actualizándolo) o si es preferible sustituirlo por uno totalmente nuevo.

Este proyecto parte de la decisión –ya tomada– de remplazar al sistema existente. Para ello, se dispone de 10 nuevos UAVs, un sistema de posicionamiento en interiores y un PC destinado a funcionar como GCS[2].

1.1 Objetivos

El objetivo principal es llevar a cabo la puesta en marcha de una infraestructura que permita realizar vuelos con múltiples vehículos aéreos no tripulados de manera coordinada. Los siguientes objetivos secundarios se definen con la intención de igualar o mejorar las características del sistema previamente existente:

Se prestará especial atención al asunto de la escalabilidad. El nuevo sistema deberá permitir que se incremente el número de UAVs en el futuro sin tener que llevar a cabo importantes modificaciones. Asimismo, es deseable que el coste unitario de los nuevos UAVs no sea demasiado elevado y que se minimice la necesidad de adquirir hardware adicional.

El tiempo y esfuerzo necesario para lograr una infraestructura plenamente funcional no es despreciable. Por ello, es también una pretensión de este trabajo desarrollar un sistema que sea fácilmente accesible por otros investigadores en futuros proyectos, de manera que no tengan que preocuparse por solucionar problemas técnicos y puedan centrarse en el desarrollo de nuevos algoritmos y estrategias de control.

[1] *Technology Readiness Level* (nivel de madurez tecnológica).
[2] *Ground Control Station* (estación de control en tierra).

1.2. Estado de la técnica

El interés de desarrollar sistemas multi-UAV radica en la existencia de misiones en las que, por su propia naturaleza, el uso de varios UAVs supone una importante ventaja frente al uso de uno solo. En [1] se examinan las publicaciones más relevantes en este ámbito durante los últimos años y se enumeran las siguientes ventajas:

- Mayor eficiencia. Capacidad para completar la misión en un tiempo menor. Esta característica puede ser especialmente relevante, por ejemplo, en aplicaciones de búsqueda y rescate en las que se necesite examinar un área muy extensa.
- Menor coste. Emplear varios UAVs de menor tamaño puede ser menos costoso que recurrir a un único vehículo de mayores dimensiones y peso, tanto por el precio de los vehículos como por los costes administrativos asociados al cumplimiento de las regulaciones que rigen el vuelo de los vehículos pesados.
- Actuación simultánea. Se habilita la posibilidad de actuar simultáneamente en diferentes localizaciones geográficas.
- Tolerancia a fallos. La misión puede continuar aun cuando tiene lugar la pérdida de algún UAV. Es posible incorporar UAVs redundantes para aumentar la robustez.

En relación con su actividad, pueden distinguirse dos importantes conceptos: coordinación y cooperación [2]. La coordinación está relacionada con la necesidad de compartir recursos, tales como el espacio físico; o de llevar a cabo tareas de manera sincronizada. La cooperación involucra comportamientos colaborativos: cada individuo realiza una serie de tareas para cumplir con un objetivo común. En este último caso, puede ser necesario establecer una metodología para la toma de decisiones y asignación de tareas. Dependiendo de la disponibilidad de la información, las estrategias pueden plantearse de manera centralizada o distribuida. En general, las segundas favorecen más la escalabilidad del sistema.

La implementación práctica de esta clase de algoritmos depende de dos aspectos fundamentales: la estimación del estado y la infraestructura de comunicaciones [3]. Dentro de las tecnologías que pueden ser empleadas para obtener la localización de los UAV se distinguen dos grupos:

- Aquellas que dependen de una infraestructura externa. Es el caso del GPS (y RTK GPS[3]), los sistemas de captura de movimiento basados en cámaras infrarrojas, o los sistemas basados en balizas de radiofrecuencia. Para estos últimos, la tecnología UWB[4] ha ganado especial relevancia en los últimos años.
- Aquellas que permiten realizar la estimación a bordo. Generalmente visión monocular, visión estéreo y LIDAR, en ocasiones combinados con medidas de sensores inerciales. Requieren del uso de técnicas como la odometría visual, la odometría basada en LIDAR o las técnicas de SLAM[5]. Las estrategias de cooperación multi-UAV también se extienden a este ámbito, siendo un ejemplo de esto los algoritmos de SLAM multi-agente.

En cuanto a las comunicaciones, es bastante común el uso de WiFi y de módulos como los XBee-Pro [1], que emplean los protocolos IEEE 802.11 y IEEE 802.15.4 respectivamente. Se espera que en los próximos años el uso de la red 5G se extienda a este tipo de aplicaciones, dadas sus características de baja latencia.

A continuación, se citan algunos ejemplos de aplicaciones en los que se propone el uso de sistemas multi-UAV. En [4] se aplican al caso de un sistema de videovigilancia. Otro caso práctico es la utilización de UAVs para dar cobertura a una zona en la que la red de comunicaciones se haya visto dañada tras una catástrofe natural [5]. También se han aplicado al transporte de una carga de manera conjunta [6], en aplicaciones de cinematografía [7] y en detección de incendios forestales [8].

[3] *Real Time Kinematics Global Positioning System* (Navegación cinética satelital en tiempo real)
[4] *Ultra-wideband* (Banda ultraancha)
[5] *Simultaneous localization and mapping* (Localización y modelado simultáneos)

Existen también múltiples trabajos académicos enfocados no tanto en las aplicaciones prácticas sino en la puesta en marcha de un banco de pruebas que habilite futuras investigaciones. En general, las diferencias que hay entre ellos están relacionadas con el método empleado para obtener la localización de los UAVs y la arquitectura de control empleada.

En [9] se detalla una infraestructura para el control de 20 *quadrotors* en interiores, haciendo uso de un sistema de captura de movimiento basado en cámaras infrarrojas (VICON) y centralizando el control en un PC desde el que se envían comandos a todos los drones. Otros centros tecnológicos han desarrollado arquitecturas similares: The Flying Machine Arena [10] en el ETHZ, RAVEN [11] en el MIT, o el sistema que antecede a este proyecto, en CATEC.

En [12] exponen la puesta en marcha de un testbed de interiores que no requiere de unas instalaciones dedicadas y puede ser montado y desmontado en poco tiempo. Enfatizan el ahorro que supone prescindir de un sistema de posicionamiento como VICON; en su lugar, hacen uso del sistema Lighthouse[6] de HTC. Además, con dicho sistema la estimación de la posición puede realizarse a bordo, lo que reduce la dependencia con el PC central. El trabajo presentado en [13] parte de un entorno de pruebas basado en simulación, para después incorporar drones comerciales que pueden volar en exteriores haciendo uso de GPS. En [14] deciden prescindir de cualquier tipo de sistema de posicionamiento externo y confían en la localización obtenida por cada uno de los drones mediante técnicas de odometría visuo-inercial.

Han surgido también en los últimos años aplicaciones dentro del ámbito del entretenimiento. Dado que suelen estar llevadas a cabo por empresas privadas, no existe apenas información disponible sobre sus detalles técnicos. Los espectáculos que empresas como Intel o EHang realizan en exteriores, hacen uso de tecnologías GNSS[7] para la localización de los UAVs. Aquellos que tienen lugar en interiores necesitan de sistemas de posicionamiento alternativos, por ejemplo, la empresa Verity Studios ha desarrollado un sistema basado en UWB. En ambos casos, suelen utilizarse coreografías pre-programadas.

Figura 1-1. A la izquierda, una demostración de la empresa Verity Studios, a la derecha, un espectáculo ofrecido por Intel.

[6] https://www.vive.com/eu/accessory/base-station/
[7] *Global Navigation Satellite System* (Sistema global de navegación por satélite)

1.3. Plataforma

La plataforma elegida para el desarrollo del proyecto es el Crazyflie 2.1, fabricado por la empresa sueca Bitcraze[8]. El principal punto a su favor es que se trata de una plataforma de código abierto en su totalidad, desde el firmware que ejecuta el microcontrolador a bordo hasta el software de alto nivel que permite su teleoperación desde un PC. Además, su reducido peso y tamaño hacen que sea extremadamente seguro trabajar con ellos, y la flexibilidad de su estructura les permite resistir colisiones o caídas desde varios metros de altura sin fracturarse. Es por ello por lo que, en los últimos años, numerosos investigadores de instituciones de renombre han sabido ver su potencial y han dado lugar a un alto número de publicaciones en las que se ha usado esta plataforma. En [15] puede encontrarse un listado que es actualizado con frecuencia.

1.3.1. Crazyflie 2.1

El Crazyflie 2.1 es un cuadricóptero (o *quadrotor*) de reducidas dimensiones. Presenta una configuración en forma de X, siendo la separación lateral de los motores de tan solo 92mm y su altura de 29mm. Pesa 27g y admite una carga útil de 15g adicionales (máximo recomendado). Dispone de una batería LiPo de 250 mAh de capacidad, que lo dota de una autonomía de vuelo de aproximadamente 7 minutos.

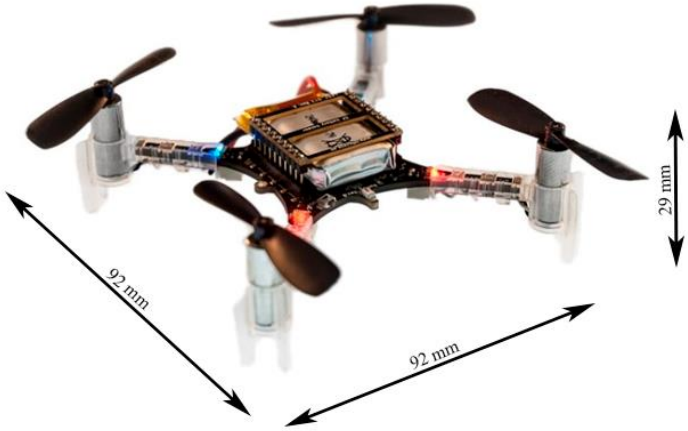

Figura 1-2. Dimensiones del Crazyflie 2.1

Para la propulsión utiliza 4 motores de corriente continua de tipo *coreless*, cuya construcción se caracteriza por prescindir del núcleo de hierro interno al rotor. Este tipo de motores presentan menor inercia que los motores convencionales, lo que les permite modificar su velocidad de giro en un tiempo menor.

Está equipado con dos microcontroladores. El STM32F405 es el microcontrolador principal y el más potente de los dos (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash); se encarga de ejecutar el código del autopiloto. Por otra parte, el nRF51822 (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash) lleva a cabo la comunicación por radio y de la gestión de la energía. Ambos microcontroladores se comunican entre sí mediante sus respectivas UARTs (comunicación serie).

[8] https://www.bitcraze.io/

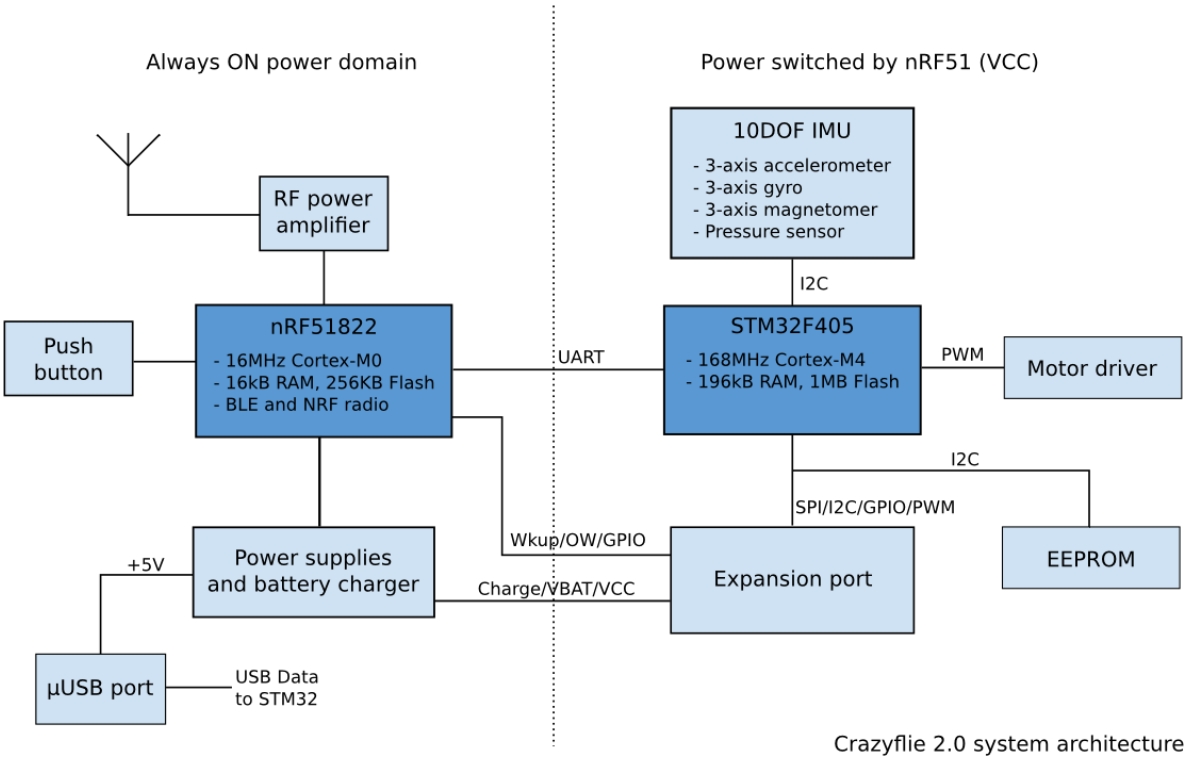

Figura 1-3. Arquitectura hardware

La IMU (Unidad de Medidas Inerciales) está compuesta por el BMI088 (acelerómetro y giróscopo de 3 ejes) y el BMP388 (barómetro). Nótese que el esquema (Fig 1-3) pertenece a la versión 2.0; en la nueva versión los desarrolladores decidieron prescindir del magnetómetro.

Los sensores inerciales son lo bastante rápidos para lograr un control de estabilización preciso, sin embargo, debido a su naturaleza no son suficiente para llevar a cabo una navegación plenamente autónoma. La estimación de la altura que se obtiene al usar únicamente el barómetro es bastante oscilante y no permite lograr un *hovering* estable. Por otra parte, no dispone –por sí mismo– de ningún método para conocer su posición y orientación de forma absoluta, y cualquier intento de odometría inercial presentará derivas por los errores acumulativos debidos al ruido de los sensores.

Para suplir estas limitaciones, la empresa Bitcraze comercializa una serie de tarjetas de expansión destinadas a aumentar las capacidades del dron en lo referente a localización:

- Loco positioning deck. Permite la recepción de medidas de rango procedentes de un sistema de balizas de radiofrecuencia basado en la tecnología UWB (banda ultraancha). Proporciona una estimación de la posición de manera absoluta con una precisión de 10 cm.
- Lighthouse positioning deck. Incorpora cuatro fotodiodos para la recepción de pulsos de luz infrarroja emitidos por el sistema de posicionamiento comercial HTC Vive Base Station. Proporciona una estimación absoluta con precisión milimétrica.
- Flow deck. Aplica el concepto de flujo óptico para medir la velocidad de traslación en el plano horizontal. Para ello, incorpora una cámara dirigida hacia el suelo y un láser que permite estimar la altura. La estimación de la posición se realiza, en este caso, de manera relativa.

1.3.2. Crazyradio PA

La comunicación inalámbrica entre el PC y el Crazyflie tiene lugar a través de este dispositivo. Se trata de un adaptador USB con un circuito de amplificación y una antena. Tiene una potencia de 20dBm y un alcance de 1km en ausencia de obstáculos.

Opera en la banda de 2.4GHz, lo que la hace susceptible a interferencias con las redes WiFi o las conexiones Bluetooth. Sin embargo, es posible seleccionar hasta 125 canales diferentes (separados por 1MHz) para mitigar dicho problema. También permite configurar el ancho de banda (2Mbps, 1Mbps o 250 Kbps). Para su uso en interiores se recomienda elegir el mayor ancho de banda posible, de cara a aumentar la robustez frente a interferencias (a costa de reducir el alcance).

Está basado en el chip nRF24LU1+ de Nordic Semiconductor, que permite enviar paquetes de hasta 32 bytes e implementa la comprobación del ACK y el reenvío en caso de pérdida de paquetes.

Figura 1-4. Crazyradio PA

1.3.3. Software

Para controlar el dron desde un PC es deseable disponer de una interfaz software que aporte cierto nivel de abstracción en el uso de la antena. La comunicación se basa en el protocolo CRTP (Crazy RealTime Protocol), cuyos detalles pueden consultarse en [16]. El PC envía a través del enlace una serie de mensajes de acuerdo con dicho protocolo, que el microcontrolador principal del dron interpreta para dar lugar a unas determinadas acciones. Para simplificar esta tarea, es preferible hacer uso de funciones de más alto nivel que generen automáticamente los paquetes y sean fácilmente identificables con las acciones a realizar. Existen dos alternativas ya desarrolladas:

Crazyflie-lib-python

Se trata de la API oficial proporcionada por la empresa que comercializa el Crazyflie. Esta librería da soporte a funcionalidades básicas como el envío de comandos de movimiento, el *logging* de variables del microcontrolador a bordo o la modificación de parámetros del firmware en tiempo de ejecución. Está totalmente programada en Python, incluido el *driver* de la radio. Si bien es cierto que permite controlar a un único dron con relativa facilidad, su desempeño empeora sustancialmente en situaciones en las que se requiere hacer uso de un gran ancho de banda; por ejemplo, cuando se intenta controlar varios drones desde una misma antena. A pesar de que la librería soporta esta funcionalidad, su éxito está condicionado a que la cantidad de información que necesiten intercambiar el PC y los drones sea mínima.

Crazyswarm

El proyecto Crazyswarm [17] ha sido desarrollado por investigadores de la USC[9]. En él se describe una arquitectura para el control de un alto número de Crazyflies en interiores y se abordan los principales retos que supone trabajar con dichos drones, tales como la estimación de la posición y las limitaciones en las comunicaciones. En ese sentido, destaca por implementar un esquema de comunicaciones más eficiente que el de la librería oficial. En contraposición con esta, prescinde de la capa de transporte (no asegura que el paquete se haya recibido correctamente) a fin de garantizar una latencia reducida. Además, el envío del *feedback* de posición se lleva a cabo comprimiendo en un mismo paquete la posición y orientación de hasta dos drones, y haciendo uso de *broadcasting* para lograr una alta tasa de envío.

En [18] consiguieron volar simultáneamente 49 drones usando únicamente 3 radios. Sin embargo, en este caso la única información que se envía durante el vuelo es el *feedback* de posición, mientras que la trayectoria se carga al inicio y no puede ser modificada posteriormente. Recientemente, han incorporado la posibilidad de enviar referencias de posición en tiempo real, pero dado que el ancho de banda requerido es mayor, el número de drones por antena ha de ser necesariamente menor.

Todo el software se encuentra disponible en su página web como código abierto. El núcleo del proyecto ha sido desarrollado en C++, aunque gracias al framework de ROS es posible aprovechar también sus funcionalidades desde scripts de Python.

1.4. Testbed

Todas las pruebas de vuelo han sido realizadas en el testbed de CATEC[10]. Se trata de un espacio de 15x15x5m delimitado por una red de seguridad. Está equipado con un sistema de posicionamiento de interiores compuesto por 20 cámaras VICON. Dicho sistema permite obtener la posición y orientación de un objeto con una precisión del orden de milímetros y a una tasa de refresco de hasta 500 Hz. Para ello, deben colocarse unos marcadores pasivos sobre el objeto, que serán captados simultáneamente por varias cámaras infrarrojas para reconstruir su posición 3D.

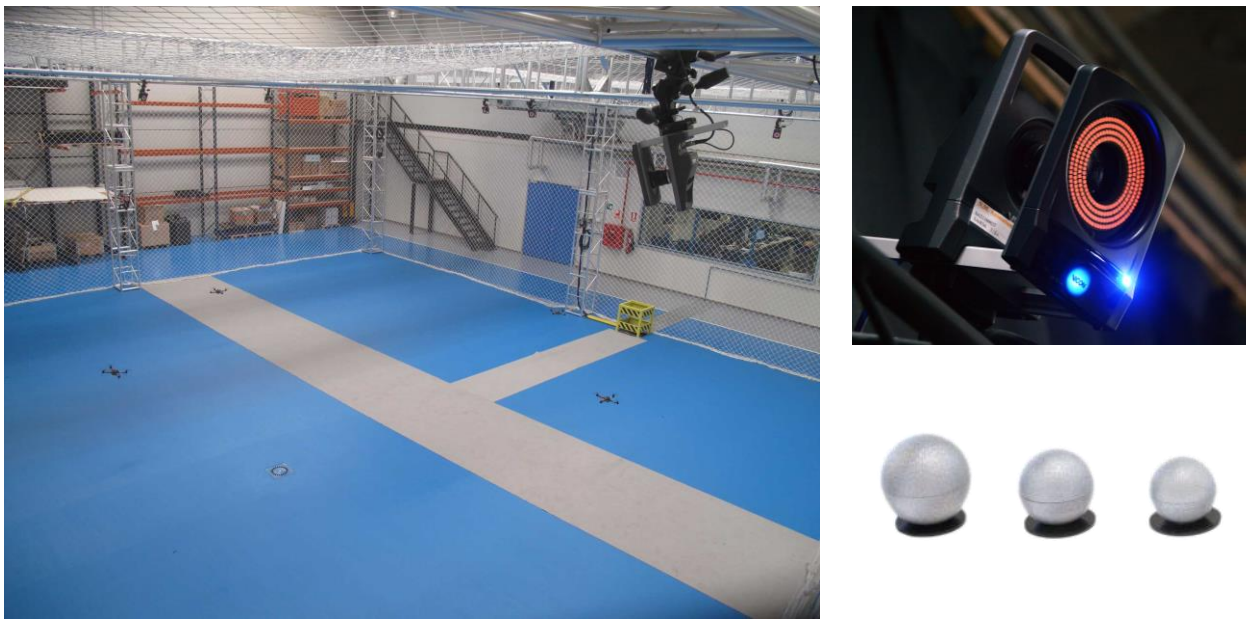

Figura 1-5. Testbed de CATEC, cámara VICON T-40 y marcadores retrorreflectores.

[9] University of Southern California
[10] Centro Avanzado de Tecnologías Aeroespaciales. http://www.catec.aero/es

El PC empleado como estación de control en tierra cuenta con las siguientes características: Procesador Intel Xeon(R) CPU E5620 @ 2.4 GHz x 8, memoria RAM de 12 GB, Sistema Operativo Ubuntu 18.04.4 LTS 64 bits

2 CONTROL Y ESTIMACIÓN

El control de un sistema multi-UAV es un problema que debe ser abordado a múltiples niveles. Por una parte, es necesario que cada dron, a nivel individual, sea capaz de lograr la estabilización y realizar el seguimiento de las referencias de posición que se le comanden. Por otra, a un nivel superior, surge la necesidad de decidir cómo se comporta cada uno de los individuos. En este capítulo se tratará únicamente el primer caso. El segundo problema es un tema extenso que escapa del alcance de este proyecto y será abordado de manera muy superficial en próximos capítulos.

En primer lugar, se definirán dos sistemas de referencia: el sistema de referencia inercial {W}, que está situado en el suelo del *testbed* y su orientación sigue la convención ENU (*East – North – Up*) y el sistema de referencia del cuerpo {B}, cuyo origen se sitúa en el centro de masas del vehículo y cuya orientación es tal que el eje Z apunta hacia la parte superior y el eje X hacia el frente. Se expresará la orientación de {B} respecto de {W} mediante tres rotaciones en ejes móviles, según los ángulos roll, pitch, yaw.

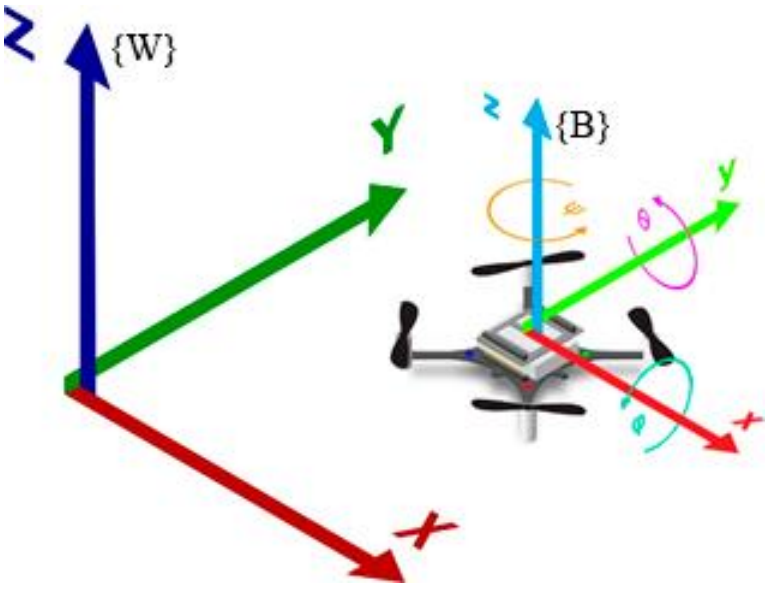

Figura 2-1. Sistemas de referencia

Antes de adentrarnos en el problema de control comentaremos brevemente cómo tiene lugar la generación de las acciones de control.

Al girar, cada uno de los motores produce una fuerza (empuje) proporcional al cuadrado de la velocidad angular:

$$f_i = k_t \omega_i^2$$

Y un momento de reacción:

$$\tau_i = k_d \omega_i^2$$

Siendo k_t y k_d dos coeficientes aerodinámicos cuyos valores pueden ser ajustados experimentalmente.

Como puede verse en la Figura 2-2, los motores situados en una diagonal giran en el mismo sentido y los restantes giran en sentido contrario. Para lograr que las cuatro fuerzas tengan signo positivo se utilizan hélices de perfiles opuestos según el sentido de giro. De esta forma, además se consigue que los momentos τ_i se cancelen entre sí cuando la velocidad angular sea la misma.

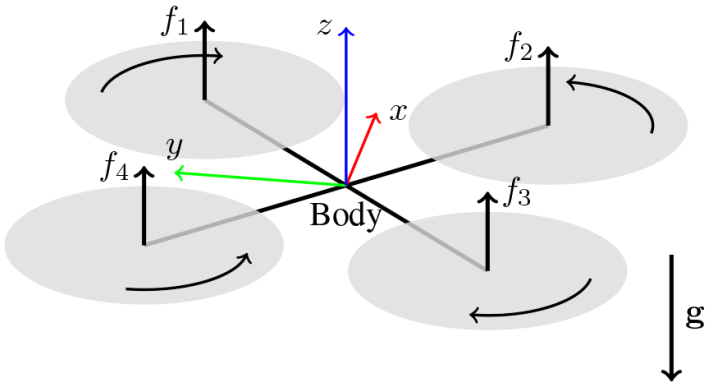

Figura 2-2. Sistema de referencia del cuerpo, empuje y sentido de giro de los motores. [19]

Debido a la disposición geométrica de los motores, sobre el centro de masas del *quadrotor* actúan las siguientes fuerzas y momentos:

$$^{B}\boldsymbol{F} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f_1 + f_2 + f_3 + f_4 \end{bmatrix}$$

$$^{B}\boldsymbol{M} = \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} d\,(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} d\,(-f_1 - f_2 + f_3 + f_4) \\ \frac{k_d}{k_t}(-f_1 + f_2 - f_3 + f_4) \end{bmatrix}$$

Siendo d la distancia desde el motor hasta el centro de masas del *quadrotor*.

De las relaciones anteriores se obtiene un sistema de cuatro ecuaciones y cuatro incógnitas cuya solución permite hallar las velocidades angulares a las que debe girar cada motor para lograr un vector de actuaciones $\boldsymbol{U} = [T \quad \tau_\varphi \quad \tau_\theta \quad \tau_\psi]$ sobre el centro de masas del sólido rígido.

El control de la posición y la orientación de un *quadrotor* (o en general, de un sólido rígido en el espacio) involucra 6 grados de libertad. Como se ha visto, este tipo de vehículos solo puede dar lugar a 4 acciones de control independientes; se trata, por tanto, de un sistema sub-actuado. Además, en ausencia de control se comporta como un sistema inestable.

Por estos motivos, es bastante común la implementación de una arquitectura de control consistente en dos bucles en cascada: el bucle interno controla la orientación (manteniendo la estabilidad) y el externo logra el control de posición a partir de la generación de referencias que deben ser alcanzadas por el bucle interno.

Concretamente, en el firmware del Crazyflie está implementada la siguiente arquitectura de control:

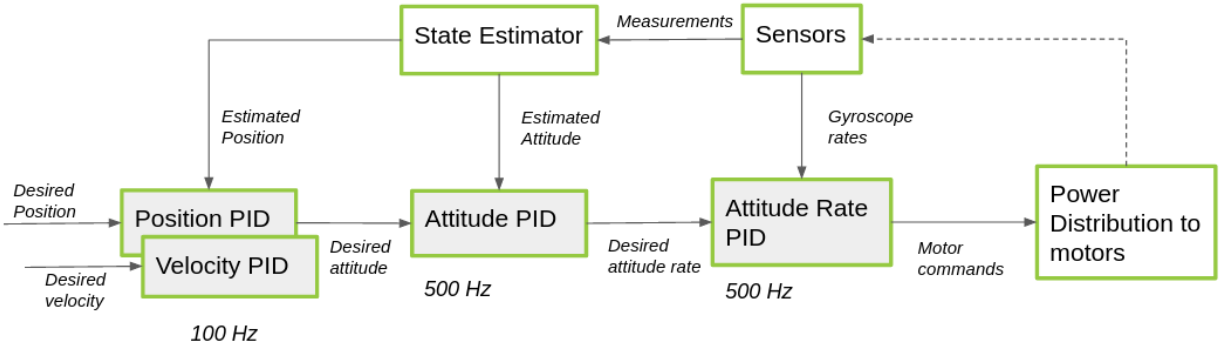

Figura 2-3. Arquitectura de control implementada en el firmware.

2.1. Control de estabilización

Aunque el sistema a controlar sea no lineal, es posible hacer uso de técnicas clásicas, como el control PID, bajo la suposición de que el *quadrotor* va a estar funcionando entorno al punto de operación correspondiente a una posición horizontal cercana al *hovering*. Este enfoque es el más habitual y el que se utilizará en este trabajo. Por otra parte, en trabajos como [20] pueden encontrarse controladores no lineales que permiten realizar maniobras más agresivas al no requerir la saturación de los ángulos de inclinación a valores pequeños.

El control de estabilización se corresponde con el bucle interno, y debe ser lo suficientemente rápido como para no perjudicar al control del bucle externo, por ello, requiere ser ejecutado a una frecuencia mayor.

Bajo la simplificación de que la dinámica de rotación está desacoplada en los tres ejes, el sistema puede modelarse con las siguientes ecuaciones diferenciales, que relacionan las variables a controlar con las acciones de control:

$$\ddot{\varphi} = \frac{\tau_{\varphi}}{I_{xx}}$$

$$\ddot{\theta} = \frac{\tau_{\theta}}{I_{yy}}$$

$$\ddot{\psi} = \frac{\tau_{\psi}}{I_{zz}}$$

Cada una de ellas representa un sistema de segundo orden que podría ser controlado por un PID, sin embargo, es bastante común realizar este control en dos etapas. En el Crazyflie tiene lugar de la siguiente manera: a nivel de velocidades angulares implementa tres PIDs que operan directamente con las medidas del giróscopo; a nivel de ángulos otros tres PIDs que utilizan la estimación del Filtro de Kalman o del filtro complementario, según se configure. Es decir, el resultado del control de ángulos pasa a ser la referencia del control de velocidad angular.

Por otra parte, los ángulos de roll y pitch están saturados a 20º para no comprometer la estabilidad del vehículo.

2.2. Control de posición/velocidad

Como se señaló al inicio del capítulo, el vehículo es capaz de generar tres pares ($\tau_\varphi, \tau_\theta, \tau_\psi$) y una fuerza cuya única componente no nula en el sistema {B} se halla en la dirección del eje Z. El movimiento en el plano XY del sistema {W} se logra al variar la inclinación, de manera que dicha fuerza obtenga valores distintos de cero en sus proyecciones sobre los ejes de {W}.

Las siguientes ecuaciones modelan la dinámica de traslación:

$$\ddot{x} = \frac{{}^{W}F_x}{m}$$

$$\ddot{y} = \frac{{}^{W}F_x}{m}$$

$$\ddot{z} = -g + \frac{{}^{W}F_z}{m}$$

Donde ${}^{W}F$ se relaciona con ${}^{B}F$ mediante una matriz de rotación.

La implementación presente en el firmware permite controlar en velocidades (que son alcanzadas mediante un PID por cada componente), o bien realizar además un control de posición, en cuyo caso, la salida de cada PID de posición se convierte en la referencia del control de velocidad.

El resultado del control de velocidad, $u_{\dot{x}}$, $u_{\dot{y}}$, no se corresponde de manera directa con referencias de θ, φ respectivamente; en general, estas variables estarán acopladas y solo coincidirían en el caso de que los ejes {W} estuviesen alineados con los ejes {B}. Por tanto, es necesario tener en cuenta el ángulo ψ para calcular las referencias de roll y pitch que serán entregadas al control de actitud. Para ello, se utiliza la siguiente relación:

$$\varphi^{ref} = u_{\dot{x}} \sin(\psi) - u_{\dot{y}} \cos(\psi)$$

$$\theta^{ref} = -(u_{\dot{x}} \cos(\psi) + u_{\dot{y}} \sin(\psi))$$

Nótese que se trata de una simplificación que considera únicamente el giro en ψ, asumiendo que los ángulos θ, φ tienen valores cercanos a cero.

En cuanto al control en z, además del valor calculado por el PID, se incorpora un término de *feedforward* que pretende compensar el peso.

2.3. Estimación del estado

Para llevar a cabo cualquier tipo de control es imprescindible disponer de información del estado que se desea controlar. En algunas ocasiones es posible medirlo directamente; en otras es preciso recurrir a técnicas de estimación que permitan inferir su valor a partir de otras medidas y un modelo matemático.

A menudo se emplean técnicas de fusión sensorial, que consisten en combinar medidas de sensores de distinta naturaleza para lograr una estimación más precisa de la que se obtendría con cada uno de ellos por separado. En el caso del Crazyflie, se encuentran implementadas en el firmware dos técnicas ampliamente utilizadas: el filtro complementario y el filtro de Kalman.

2.3.1. Filtro complementario

La estrategia del filtro complementario consiste en combinar dos señales con diferentes características de ruido. La señal en la que predomina el ruido de alta frecuencia es filtrada mediante un filtro paso bajo, $G(s)$, y la señal en la que lo hace el de baja frecuencia mediante su complementario, $1 - G(s)$. Finalmente, las dos señales filtradas se suman.

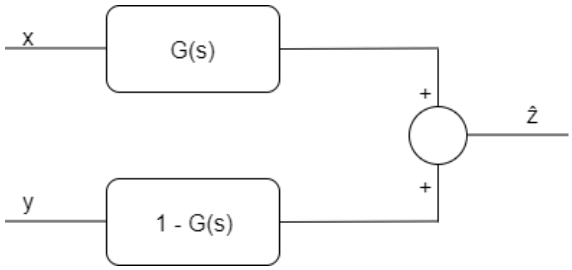

Figura 2-4. Filtro complementario genérico

En el caso de los vehículos aéreos es bastante común emplear esta estrategia para estimar la orientación a partir de medidas de sensores inerciales (acelerómetro y giróscopo).

Por una parte, el giróscopo proporciona medidas de la velocidad angular en el sistema de referencia del cuerpo, cuya integración permite obtener una estimación de la posición angular de forma relativa. Sin embargo, debido al ruido del sensor, dicha estimación presentará importantes derivas a largo plazo.

El acelerómetro puede emplearse para medir directamente la orientación. Cuando el dron está parado, la única aceleración que detecta el sensor es la de la gravedad. Dado que la dirección de este vector es conocida en ejes inerciales, puede utilizarse como referencia absoluta para inferir el valor de los ángulos *roll* y *pitch*. El principal inconveniente es que esta información deja de ser fiable cuando el dron entra en movimiento y queda sometido a otras aceleraciones. Es decir, no se puede confiar en las medidas a corto plazo.

De acuerdo con el razonamiento anterior, parece coherente aplicar un filtro de paso bajo a la estimación del acelerómetro y uno de paso alto a la del giróscopo. Por lo tanto, podría aplicarse el siguiente esquema a las variables *roll* y *pitch* de manera independiente:

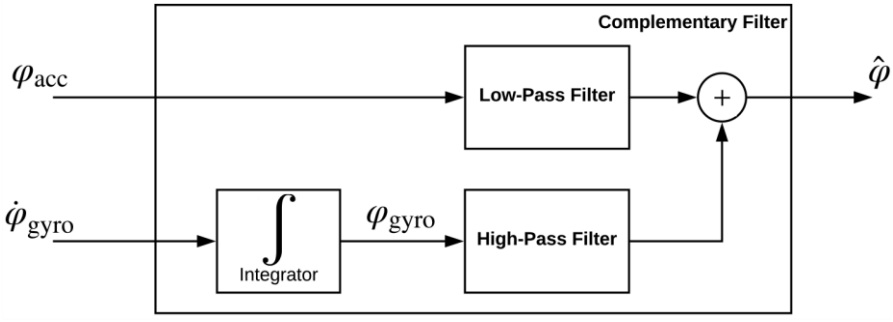

Figura 2-5. Filtro complementario para la estimación de la orientación mediante acelerómetro y giróscopo.

El planteamiento que se ha descrito corresponde a la formulación clásica del filtro complementario para un sistema SISO. Si bien es cierto que dicho enfoque es ampliamente utilizado debido a su sencillez y su bajo coste computacional, existen otras formulaciones más novedosas que han demostrado ser capaces de lograr una estimación de mejor calidad y con mayor robustez.

En el Crazyflie se encuentra implementado el filtro descrito en [21], al que sus autores han denominado Filtro Complementario Explícito.

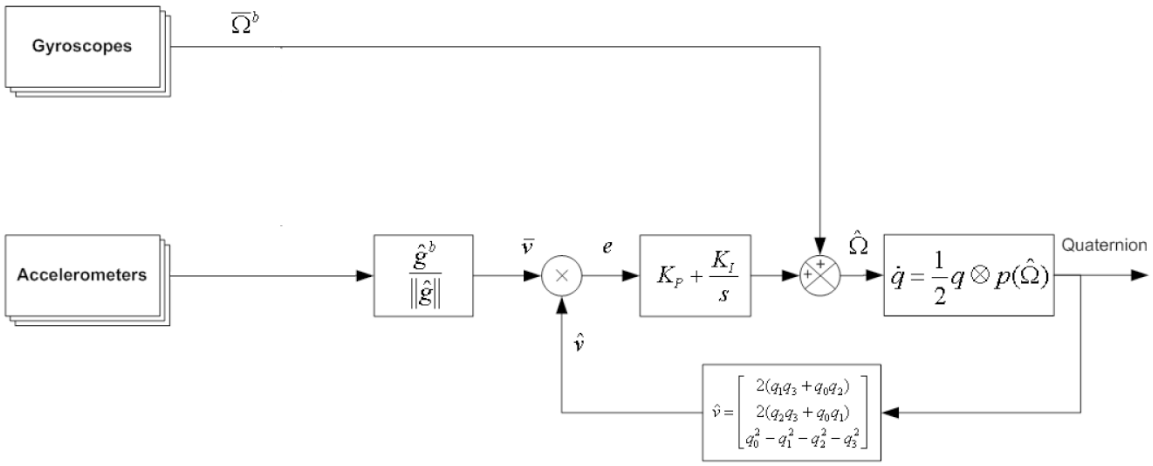

Figura 2-6. Implementación del Filtro Complementario Explícito en el Crazyflie

En esta implementación la orientación se representa mediante cuaternios y es estimada conjuntamente (en contraposición con el planteamiento clásico que considera los ángulos *roll* y *pitch* de manera desacoplada). Mediante el acelerómetro se obtiene una medida de la dirección de la gravedad, que es comparada con la estimación del filtro para obtener una métrica del error en términos de rotación. Posteriormente, un regulador PI determina la contribución del acelerómetro (mediante el término proporcional) y realiza una corrección del *bias* del giróscopo (mediante el término integral). La salida del PI se combina con la medida del giróscopo para obtener una estimación de la velocidad angular, la cual permite calcular la tasa de cambio del cuaternión, que finalmente es integrada para calcular la orientación.

En la publicación original [21] también se lleva a cabo una precompensación de la aceleración del cuerpo con el fin de obtener una mejor estimación de la dirección de la gravedad. Esta aceleración es calculada a partir de un modelo aerodinámico. En la implementación del Crazyflie se ha prescindido de esta característica.

Las estimaciones obtenidas con este método permiten cerrar el bucle de control de estabilización y, por tanto, realizar vuelos en modo manual. Es decir, sería el piloto quien cerrase el bucle de control de posición. Sin embargo, el uso de este estimador es insuficiente para la realización de vuelos autónomos.

2.3.2. Filtro de Kalman Extendido

El filtro de Kalman es un algoritmo recursivo que permite obtener estimaciones de los estados de un sistema a partir de un modelo del proceso y de las medidas proporcionadas por los sensores. El planteamiento original está definido para sistemas lineales y asume que tanto el proceso como la medida están afectados por ruidos blancos, de media nula, aditivos, e independientes entre sí. Está demostrado que, bajo estas suposiciones, la estimación obtenida por el filtro es óptima. El filtro de Kalman extendido (EKF) surge por la necesidad de aplicar este método al caso de los sistemas no lineales, puesto que aparecen en multitud de aplicaciones prácticas. A pesar de que el EKF no garantiza la optimalidad de la estimación, en la práctica es capaz de lograr resultados aceptables. A continuación, se explicará su funcionamiento.

La evolución del proceso, en un instante k, está definida por:

$$\boldsymbol{\xi_k} = \boldsymbol{f}(\boldsymbol{\xi_{k-1}}, \boldsymbol{u_k}) + \boldsymbol{w_k}$$

Y el modelo de medida:

$$\boldsymbol{z_k} = \boldsymbol{h}(\boldsymbol{\xi_k}) + \boldsymbol{v_k}$$

Siendo $\boldsymbol{\xi}$ el vector de estados, $\boldsymbol{u}$ el vector de entradas, $\boldsymbol{z}$ el vector de medidas y $\boldsymbol{f}$ y $\boldsymbol{h}$ dos funciones diferenciables y no lineales. El ruido del proceso, $\boldsymbol{w_k}$, representa la incertidumbre debida a los fenómenos no modelados por $\boldsymbol{f}$; el ruido de la medida, $\boldsymbol{v_k}$, se debe a las imperfecciones del sensor. Ambos ruidos cumplen las características enumeradas anteriormente.

El algoritmo consiste en dos etapas: predicción y corrección.

Durante la etapa de predicción, las nuevas estimaciones se obtienen haciendo uso del modelo del sistema y del valor estimado en el instante inmediatamente anterior:

$$\hat{\boldsymbol{\xi}}_{\boldsymbol{k|k-1}} = \boldsymbol{f}(\hat{\boldsymbol{\xi}}_{\boldsymbol{k-1|k-1}}, \boldsymbol{u_k})$$

La incertidumbre asociada a la estimación, modelada mediante la matriz de covarianzas P, se proyecta hacia el futuro mediante la siguiente relación:

$$\boldsymbol{P_{k|k-1}} = \boldsymbol{F_k P_{k-1|k-1} F_k^T} + \boldsymbol{Q_k}$$

Donde $\boldsymbol{F_k}$ se define como el jacobiano de $\boldsymbol{f}$ evaluado en $(\hat{\boldsymbol{\xi}}_{\boldsymbol{k-1|k-1}}, \boldsymbol{u_k})$. La matriz de covarianzas asociada al ruido del proceso, $\boldsymbol{Q_k} = E[\boldsymbol{w_k w_k^T}]$.

Nótese que, tras cada etapa de predicción, la incertidumbre de la estimación debe aumentar necesariamente.

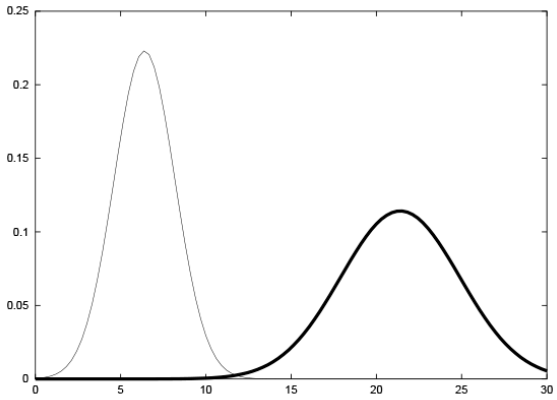

Figura 2-7. Evolución de la estimación durante la etapa de predicción. [25]

La etapa de corrección tiene lugar cuando se recibe una nueva medida. Comprende las siguientes operaciones:

1. Cálculo de la ganancia

$$\boldsymbol{K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R_k\right)^{-1}}$$

2. Corrección de la estimación del estado

$$\boldsymbol{\hat{\xi}_{k|k} = \hat{\xi}_{k|k-1} + K_k \left(z_k - h\left(\hat{\xi}_{k|k-1}\right)\right)}$$

3. Corrección de la estimación de la covarianza

$$\boldsymbol{P_{k|k} = (I - K_k H_k) P_{k|k-1}}$$

Donde $\boldsymbol{H_k}$ se define como el jacobiano de $\boldsymbol{h}$ evaluado en $(\hat{\boldsymbol{\xi}}_{\boldsymbol{k|k-1}})$.

Las siguientes gráficas ilustran el desarrollo de esta etapa. En la imagen de la izquierda se representa el resultado de la etapa de predicción junto a la recepción de una nueva medida. A la derecha, el resultado de la etapa de corrección. Se observa que, tras la ejecución de esta etapa, la incertidumbre de la estimación disminuye.

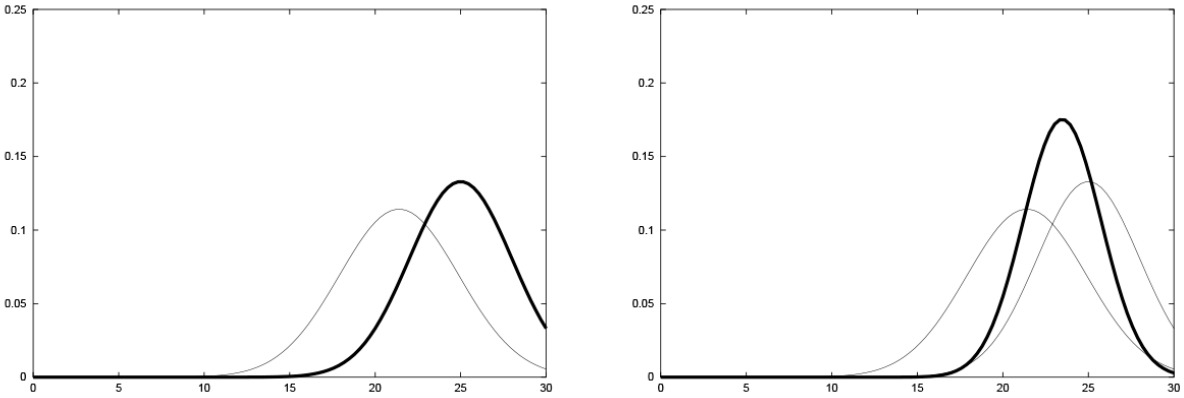

Figura 2-8. Evolución de la estimación durante la etapa de corrección.

La implementación presente en el Crazyflie está basada, fundamentalmente, en dos publicaciones [22], [23]. En ellas se hace uso de un modelo del sistema obtenido mediante la formulación de Newton-Euler:

$$m\ddot{\boldsymbol{x}} = \mathbf{R}(f\boldsymbol{e_3} + \boldsymbol{f_a}) + m\boldsymbol{g}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,[\boldsymbol{\omega}]_\times$$

Siendo $\boldsymbol{x}$ la posición del *quadrotor* expresada en el sistema de referencia inercial y $\mathbf{R}$ la matriz de rotación que representa la orientación del sistema de referencia del cuerpo respecto del sistema inercial. Las fuerzas que actúan sobre el sólido rígido son: el empuje total producido por los rotores, de módulo f y dirección el eje vertical del sistema de referencia del cuerpo, $\boldsymbol{e_3}$; el resto de fuerzas aerodinámicas, $\boldsymbol{f_a}$, también expresadas en dicho sistema de referencia; y el peso, $m\boldsymbol{g}$, expresado en el sistema de referencia inercial. La segunda ecuación constituye una relación puramente cinemática que involucra la matriz de rotación, su derivada temporal y la velocidad angular del sólido rígido (expresada en el sistema de referencia del cuerpo), siendo $[\boldsymbol{\omega}]_\times$ la matriz antisimétrica que cumple $[\boldsymbol{\omega}]_\times \boldsymbol{v} = \boldsymbol{\omega} \times \boldsymbol{v}$. Se ha prescindido de la ecuación que modela la dinámica de rotación (relacionando pares con aceleraciones angulares), puesto que no será empleada por el filtro propuesto.

El vector de estados que se pretende estimar comprende: posición, velocidad lineal, orientación y velocidad angular. Dado que cada una de ellas viene representada por 3 parámetros se tiene un total de 12 componentes. Para reducir la complejidad, se decide no incluir la velocidad angular en la estimación a realizar por el filtro; en su lugar se considera directamente la medida del giróscopo bajo la suposición de que el ruido es despreciable. Por lo tanto, el EKF deberá estimar 9 estados:

$$\boldsymbol{\xi} = (\boldsymbol{x}, \boldsymbol{\rho}, \boldsymbol{\delta})$$

La velocidad lineal se estima en el sistema de referencia del cuerpo y queda definida por $\boldsymbol{\rho} = \mathbf{R}^{-1}\dot{\boldsymbol{x}}$. Los estados correspondientes a la orientación son estimados mediante una formulación indirecta del filtro de Kalman, es decir, el vector $\boldsymbol{\delta}$ representa el error en orientación (y no la orientación en sí). Esta elección viene motivada por el hecho de que las representaciones de la orientación basadas en 3 parámetros pueden dar lugar a singularidades. Para evitar esto, la estimación de la orientación se actualiza tras cada etapa utilizando el vector de errores $\boldsymbol{\delta}$, que seguidamente es reseteado a cero.

$$\hat{\mathbf{R}} = \hat{\mathbf{R}}_{\mathbf{ref}}(\mathbf{I} + [\boldsymbol{\delta}]_{\times})$$

Siendo $\hat{\mathbf{R}}_{\mathbf{ref}}$ la última estimación de la orientación disponible e $\mathbf{I}$ una matriz identidad 3x3.

Por razones de eficiencia computacional, en la implementación práctica del Crazyflie esta operación se realiza mediante el producto de dos cuaternios: el primero de ellos representa la orientación de referencia y el segundo la rotación indicada por el vector $\boldsymbol{\delta}$, expresada como cuaternión a través de los parámetros de Euler-Rodrigues.

Las ecuaciones que se emplean en la etapa de predicción son las siguientes:

$$\dot{\hat{\boldsymbol{x}}} = \hat{\mathbf{R}}\,\hat{\boldsymbol{\rho}}$$

$$\dot{\hat{\boldsymbol{\rho}}} = \frac{1}{m} f\boldsymbol{e_3} - [\hat{\boldsymbol{\omega}}]_{\times}\hat{\boldsymbol{\rho}} - \mathbf{R}^{-1}\boldsymbol{g}$$

$$\dot{\hat{\boldsymbol{\delta}}} = \hat{\boldsymbol{\omega}}$$

La contribución de $\boldsymbol{f_a}$ se ha considerado despreciable. Para el cálculo de f existen varias opciones:

- Utilizar el *thrust* comandado por el controlador. Sin embargo, esto implicaría realizar un mapeo entre el valor del PWM y la magnitud física, el cual puede dejar de ser válido bajo ciertas condiciones, por ejemplo, cuando se modifica la masa o disminuye el nivel de la batería.
- Calcularlo mediante la velocidad angular de los rotores y el coeficiente aerodinámico κ_t. Dado que no se dispone de dichos valores también se descarta esta opción.
- Utilizar los valores medidos por el acelerómetro.

El acelerómetro presenta el siguiente modelo de medida, que puede relacionarse con f:

$$\boldsymbol{z_{acc}} = \mathbf{R}^{-1}(\ddot{\boldsymbol{x}} - \boldsymbol{g}) + \boldsymbol{w_{acc}} = \frac{1}{m}(f\boldsymbol{e_3} + \boldsymbol{f_a}) + \boldsymbol{w_{acc}}$$

Por tanto, la medida del acelerómetro puede identificarse con $\frac{1}{m} f\boldsymbol{e_3}$.

Dado que la ejecución del filtro se realiza en tiempo discreto, las ecuaciones de predicción que se implementan en el microcontrolador son la versión discretizada (mediante el método de Euler) de las presentadas anteriormente. Por otra parte, a partir de ellas es posible calcular analíticamente el jacobiano que ha de emplearse en la etapa de predicción de la matriz de covarianzas asociada a la estimación.

El ruido del proceso puede dividirse en dos partes: una afecta al movimiento traslacional y otra al rotacional. El movimiento traslacional se considera afectado por una aceleración, modelada como una variable aleatoria que sigue una distribución gaussiana con media nula. El efecto de dicha aceleración se propaga a los estados de velocidad y posición según los factores T y $\frac{1}{2}T^2$ respectivamente (siendo T el tiempo de muestreo). En cuanto a la rotación, se sabe que la velocidad angular está afectada por un ruido aleatorio, puesto que se toma como estimación el valor medido por el giróscopo, como se indicó anteriormente. El valor de la covarianza que caracteriza dicho ruido puede obtenerse experimentalmente. Los estados comprendidos en $\boldsymbol{\delta}$ se verán afectados por dicho ruido multiplicado por un factor T.

La etapa de predicción se ejecuta a una frecuencia constante, mientras que la de corrección tiene lugar de manera asíncrona cuando se recibe una medida determinada. Existen numerosos sensores capaces de aportar información sobre el estado del Crazyflie: altura basada en barómetro/láser, velocidad obtenida mediante flujo óptico, medidas de rango mediante UWB, etc. Cada uno de ellos presenta un modelo de medida característico que relaciona la información percibida con algunos de los estados del sistema. Dichos modelos no serán detallados aquí, ya que los sensores nombrados no han sido utilizados en el proyecto, pero se encuentran implementados en firmware y pueden ser consultados directamente en el código [24]. La única medida externa empleada procede del sistema VICON, que proporciona directamente la posición en el sistema de referencia inercial, por lo que su modelo de medida es trivial.

3 DESARROLLO DE LA INFRAESTRUCTURA

En este capítulo se describirá la solución que se ha adoptado y se presentará la estructura de los programas desarrollados. Se ha pretendido desarrollar un sistema que pueda ser utilizado con facilidad por otros investigadores en el futuro, de manera que no necesiten invertir mucho tiempo en aspectos relativos a la programación o la solución de problemas técnicos. Por ello, se ha incorporado una interfaz gráfica que agiliza la configuración inicial.

Antes de presentar la solución final se detallará el recorrido seguido, incluyendo también las decisiones que se han descartado. Con esto, se pretende poner de manifiesto los problemas que han ido surgiendo y las soluciones que se han propuesto, de manera que la lectura del capítulo pudiera servir de ayuda a aquellas personas que deseen embarcarse en un proyecto similar.

3.1. Desarrollo basado en la librería oficial

Al inicio del proyecto, se decidió hacer uso únicamente de la librería oficial para llevar a cabo el desarrollo software. Dicha librería permite, a grandes rasgos, controlar el dron de dos formas distintas: o bien cargando la trayectoria completa (previamente al inicio del vuelo), o bien enviando comandos en tiempo real.

La primera de las opciones tiene como ventaja la simplicidad en las comunicaciones: una vez cargada la trayectoria solo sería necesario dar la orden de empezar, de manera sincronizada, a todos los drones. Si además se dispusiera de alguna de las tarjetas de expansión[11] se podría realizar el seguimiento de estas trayectorias sin tener que intercambiar más información con el PC. En nuestro caso, la única información que mandaría la antena sería el *feedback* de posición procedente del sistema de posicionamiento VICON. Sin embargo, la opción de cargar previamente la trayectoria se descartó por estar limitada a un caso de uso muy concreto y no permitir la realización de experimentos más complejos, por ejemplo, en entornos dinámicos.

En cambio, se optó por la segunda opción, esto es, por el envío de comandos durante el vuelo. Para ello, la librería da soporte de tres maneras distintas:

1. Mediante comandos de alto nivel de tipo "*go to*". Esta orden permite especificar el punto de destino y la duración del movimiento. Al hacerlo, el microcontrolador a bordo ejecuta un generador de trayectorias basadas en polinomios de orden 7 (con condiciones de velocidad inicial y final nulas). Las trayectorias generadas son utilizadas como referencias del bucle de control de posición.
2. Mediante el envío de referencias de posición/velocidad. En este caso no se especifica la duración del movimiento ni se ejecuta el generador de trayectorias, sino que el valor recibido es interpretado directamente como la referencia a alcanzar por el control de posición o velocidad.
3. Mediante el envío de referencias de actitud para el bucle interno. En este modo se especifican las referencias de los ángulos *roll* y *pitch*, la velocidad angular en *yaw* y el *thrust* total.

[11] Las tarjetas de expansión fueron presentadas en el apartado 1.3.

De los tres modos citados, el primero es el que concentra la mayor parte de los cálculos en el microcontrolador a bordo, mientras que el último implica trasladar al PC tanto la generación de trayectorias como el control de posición. Este hecho tiene las siguientes repercusiones:

- Desde el punto de vista funcional: el uso de comandos de alto nivel es el modo menos flexible de los tres, dado que el movimiento queda restringido a segmentos de recta en cuyos extremos se fuerza al dron a detenerse. Con los otros dos modos sí podrían definirse trayectorias completamente arbitrarias.
- Desde el punto de vista computacional: el PC cuenta con más recursos que el microcontrolador. Si se usase el modo 3, podría aprovecharse esta característica para ejecutar algoritmos de control más complejos.
- Desde el punto de vista de las comunicaciones: para un mismo tiempo de muestreo en el control de posición, T_s, la cantidad de información a transmitir por la antena sería distinta en cada caso. Usando el modo 1 habría que transmitir el *feedback* de posición a esa frecuencia y los *waypoints* en instantes puntuales. Con el modo 2 sería necesario además enviar las referencias a la misma frecuencia, es decir, se estaría duplicando el ancho de banda utilizado. El modo 3, sería el que menos coste tendría, dado que únicamente necesitaría enviar las acciones de control (calculadas en el PC) cada T_s.

3.1.1 Control de posición implementado en la estación de tierra

De acuerdo con el razonamiento anterior, en un primer momento se optó por utilizar el tercer modo, que parecía ser el que más ventajas ofrecía (a pesar de que implicaba un trabajo adicional en el desarrollo del código de los controladores de posición y del generador de trayectorias). Si bien esta estrategia era válida conceptualmente, en la implementación práctica surgieron ciertas complicaciones que después llevarían a descartarla.

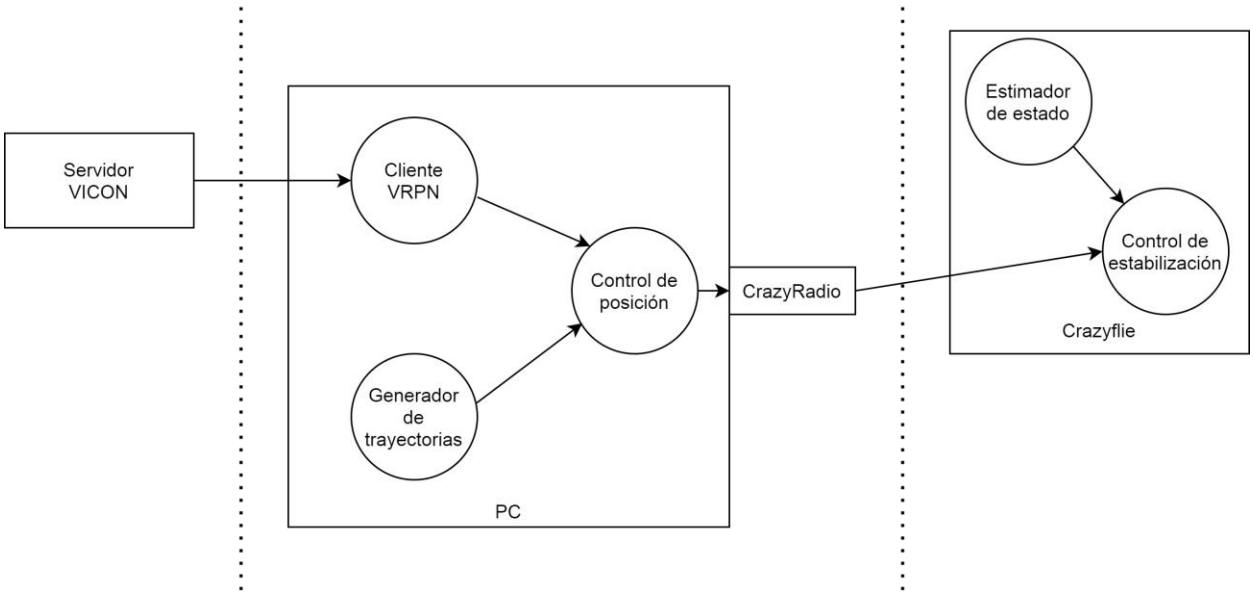

Figura 3-1. Flujo de información del control de posición implementado en la estación de tierra

La estructura de los programas a ejecutar en el PC se diseñó persiguiendo cierta modularidad: el generador de trayectorias y el control de posición tienen lugar en procesos separados. De esta manera, si en el futuro se deseara modificar o sustituir alguno de ellos por otra versión más compleja, la integración del código sería

sencilla. La comunicación entre procesos se lleva a cabo mediante ROS[12], que se trata de una herramienta que ha adquirido gran popularidad en los últimos años por facilitar la modularidad y la reutilización de código.

El software de localización de VICON se ejecuta en otro ordenador, conectado a la misma red local mediante ethernet, y que cumple la función de servidor. Para acceder a esta información desde el PC de control, se necesita establecer una conexión TCP actuando como cliente. Dado que el software oficial de VICON incorpora de forma nativa una interfaz VRPN, la opción más sencilla consiste en utilizar el paquete de ROS *vrpn_client_ros*, que establecerá dicha conexión y publicará en un *topic* la posición y orientación del objeto.

El programa desarrollado para ejecutar el control realiza, de manera cíclica, las siguientes acciones:

1. Leer la última posición obtenida por el sistema VICON.
2. Leer las referencias en X, Y, Z y *Yaw*.
3. Calcular las acciones de control (*roll, pitch, yaw rate, thrust*)
4. Enviar las acciones de control por la antena.
5. Esperar hasta siguiente ciclo

Se ajustaron experimentalmente las constantes de los PIDs para conseguir una respuesta rápida y sin demasiada sobreoscilación. Sin embargo, se observó que al repetir varias veces los experimentos en las mismas condiciones (mismas ganancias y referencias), los resultados diferían significativamente en cuanto a sobreoscilación y tiempo de subida. Esta situación se acentuaba cuando se trataba de controlar a dos drones simultáneamente.

Para encontrar la causa del problema se examinó el funcionamiento del firmware del dron: cuando éste recibe un comando a través de la antena, el comportamiento por defecto es aplicarlo hasta que se reciba el siguiente. Para evitar que se pierda el control ante una desconexión cuenta con el siguiente mecanismo de seguridad: si transcurren 500 ms sin que se reciba ningún comando, trata de estabilizarse en una posición horizontal durante un máximo de 2 segundos, momento en el cual se desactivan los motores.

Se llegó entonces a la conclusión de que el problema estaba siendo ocasionado por un fallo en las comunicaciones. El hecho de perder varios paquetes seguidos implica que la acción de control que se aplica en un instante no sea la que debería aplicarse, sino la de varios milisegundos atrás. En otras palabras, se están introduciendo (de manera aleatoria) retrasos en el bucle de control, lo cual explicaría el comportamiento sobreoscilatorio observado.

3.1.2 Control de posición implementado en el microcontrolador a bordo

Ante la falta de robustez que suponía usar el método anterior se decidió cambiar de estrategia. En esta ocasión el control de posición se ejecuta en el microcontrolador a bordo, mientras que el PC se limita a establecer la comunicación con el dron y enviar periódicamente la posición actual y las referencias de posición.

[12] Robot Operating System https://www.ros.org/

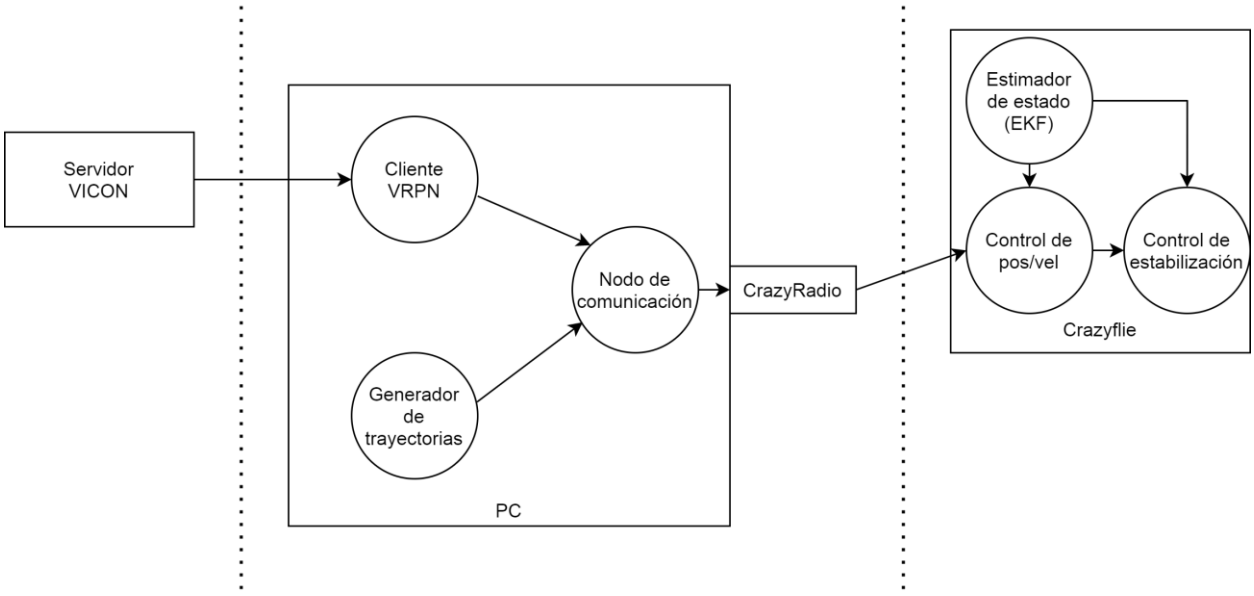

Figura 3-2. Flujo de información del control de posición implementado a bordo

Podría argumentarse que sigue presente la problemática del apartado anterior, puesto que la pérdida de paquetes en las comunicaciones va a seguir ocurriendo. Sin embargo, en este caso es mucho menos crítica y apenas tiene impacto por los motivos que se expondrán a continuación:

1. Cuando el dron recibe un paquete con la información de su localización esta no es utilizada directamente por el control de posición. En su lugar, se utiliza la estimación obtenida mediante el Filtro de Kalman, el cual se ejecuta periódicamente y a una frecuencia fija. Dicho algoritmo hace uso de las medidas externas de posición para corregir las estimaciones de los estados del sistema. Durante los ciclos en los que no se reciben nuevas medidas, el filtro continúa proporcionando estimaciones en base a un modelo del sistema. Además, actúa como algoritmo de fusión sensorial, al incorporar también las medidas de los sensores inerciales (que siempre están disponibles).
2. La pérdida de paquetes de referencias de posición tampoco es algo crítico, puesto que en el tiempo que el dron tarda en alcanzarlas estas se mantienen aproximadamente constantes.

Para el envío de la información de localización, la librería permite elegir entre transmitir únicamente las posiciones o incluir además la orientación. Utilizando la primera opción se conseguía un *hovering* estable, mientras que con la segunda el dron comenzaba a oscilar ligeramente entorno a su posición. Esto se debe a que existe una diferencia de varios grados entre los ángulos de *roll* y *pitch* medidos por el sistema de posicionamiento y la orientación real del sistema de referencia del cuerpo. Como consecuencia, el control de estabilización empeoraba su desempeño.

Podría optarse entonces por enviar solo la posición, sin embargo, en ese caso la orientación en *yaw* se estaría estimando únicamente de manera incremental. La estimación de los ángulos *roll* y *pitch* no presenta apenas derivas, ya que combina las medidas del giróscopo y el acelerómetro (cuando el dron está quieto, puede conocerse la dirección de la aceleración de la gravedad, que actúa como referencia absoluta). En el caso de *yaw*, las derivas sí son importantes.

Dado que el control en XY está acoplado con el ángulo *yaw*, no parece razonable prescindir de la orientación capturada por el sistema de posicionamiento. Lo ideal sería poder enviar únicamente el valor de *yaw* e ignorar *roll* y *pitch* (que no son precisos), sin embargo, el tipo de mensaje obliga a que la orientación se represente mediante quaternios. A pesar de esto, en el firmware se ejecuta una conversión a *roll*, *pitch*, *yaw,* antes de incorporar las medidas al Filtro de Kalman. Aumentando el valor de las covarianzas asociadas a las medidas externas de *roll* y *pitch* se logró que estas no fueran tenidas en cuenta.

Se realizaron pruebas de vuelo y se comprobó que los resultados eran aceptables.

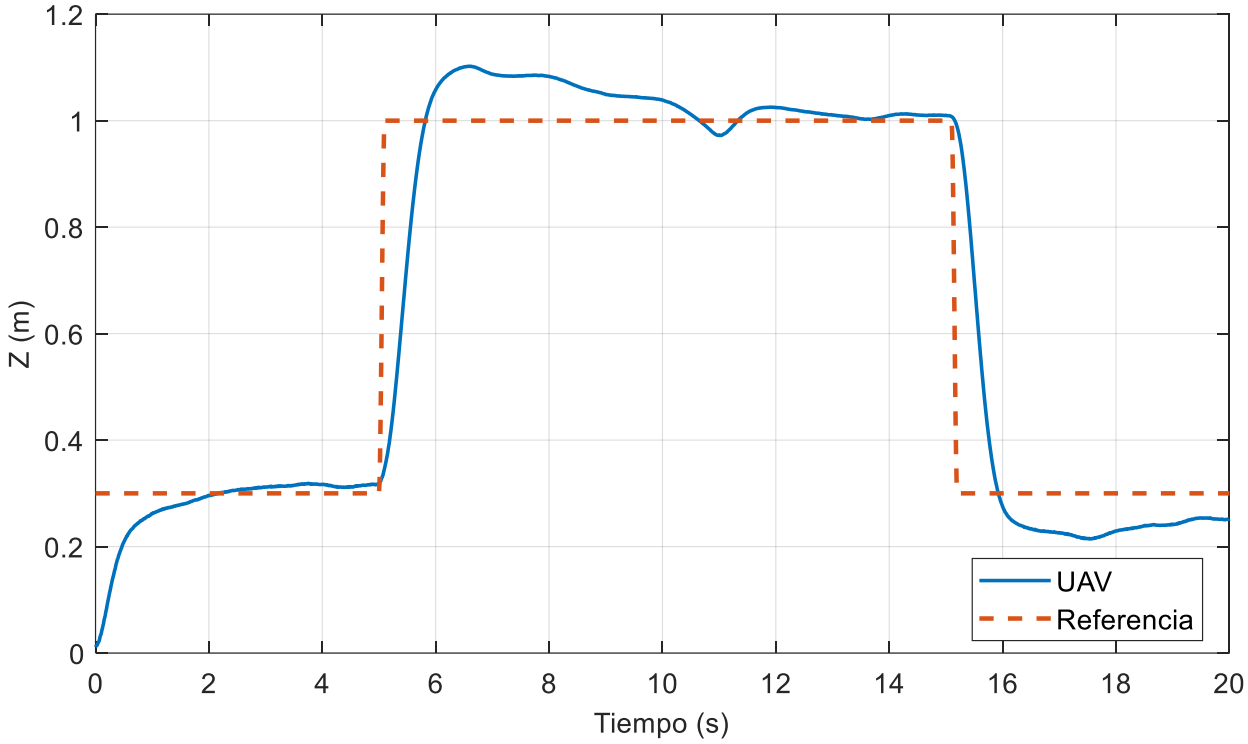

Figura 3-3. Respuesta ante escalón del bucle cerrado del control en Z.

El control de posición funciona correctamente, sin embargo, puede intuirse que variar la referencia mediante escalones no es lo más aconsejable. Si la magnitud del escalón no es demasiado grande, el control es capaz de responder adecuadamente; en caso contrario, podrían demandarse unas acciones de control inalcanzables por los actuadores, provocando importantes sobreoscilaciones e incluso llegando a comprometer la estabilidad del sistema. Además, de esta forma no es posible especificar el tiempo en el que el dron debe alcanzar el punto de destino.

Por ello, se desarrolló un generador de trayectorias que consigue hacer variar las referencias de manera suave. Se basa en una interpolación entre el punto de origen y el de destino mediante polinomios de orden 3.

$$q(t) = a(t-t_i)^3 + b(t-t_i)^2 + c(t-t_i) + d$$

Eligiendo como condiciones de contorno las posiciones de origen y destino, con velocidad inicial y final nula, los coeficientes pueden calcularse como:

$$a = -\frac{2}{T^3}(q_f - q_i)$$

$$b = \frac{3}{T^2}(q_f - q_i)$$

$$c = 0$$

$$d = q_i$$

Siendo T la duración del movimiento.

El nodo del generador de trayectorias lee un fichero de texto en el que se especifica una lista de *waypoints* (X, Y, Z, Yaw) y la duración de cada tramo. En cada tramo, se calculan los coeficientes de los polinomios y se genera la trayectoria en tiempo real.

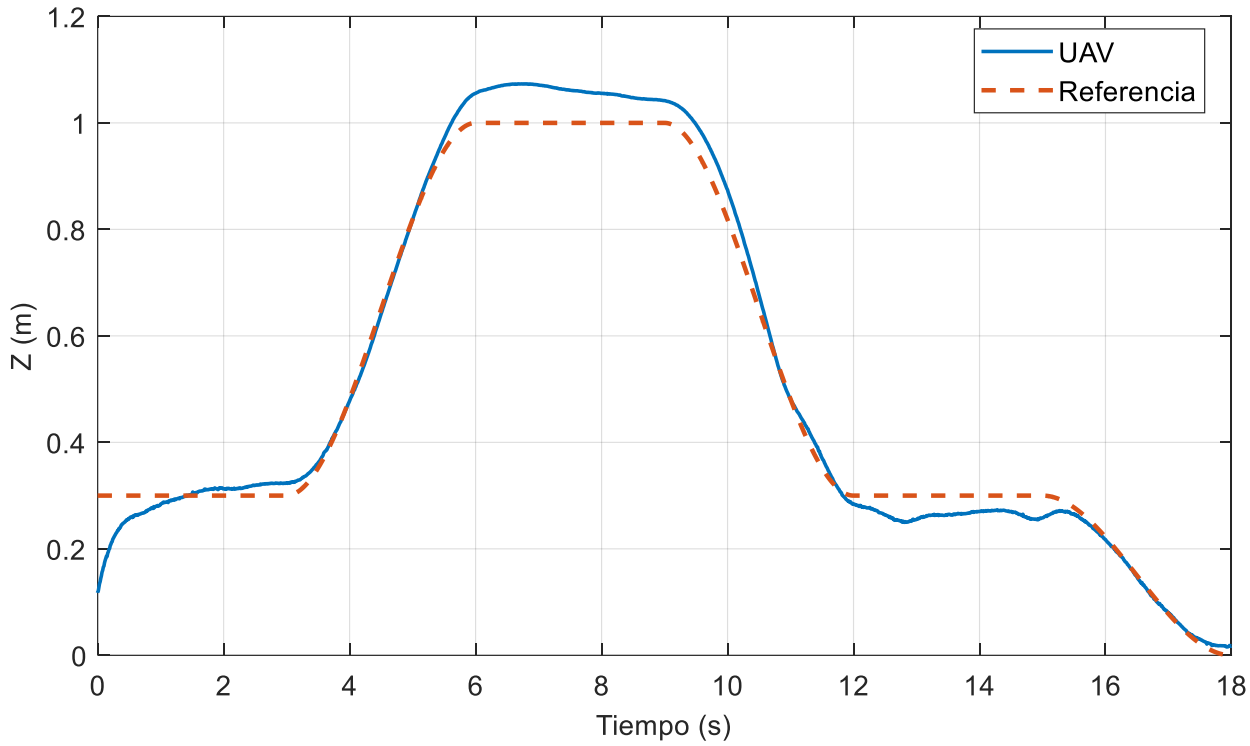

Figura 3-4. Seguimiento de una referencia generada mediante polinomios de orden 3.

A la hora de realizar la extensión al caso de múltiples drones fue necesario replantear la obtención de la localización a través del sistema de posicionamiento. El software oficial del sistema VICON permite realizar el seguimiento de múltiples objetos simultáneamente, sin embargo, para identificarlos de manera unívoca es necesario que los marcadores reflectantes de cada uno de ellos sigan distribuciones espaciales distintas. Además, es importante que las configuraciones no presenten simetrías, ya que podrían dar lugar a ambigüedades a la hora de determinar la orientación. Debido al reducido tamaño del Crazyflie no hay muchas opciones para la colocación de los reflectores, por ello, se decidió hacer uso de una estructura que permitiera aumentar la variabilidad. Se encontró un modelo en Internet y se fabricó mediante impresión 3D.

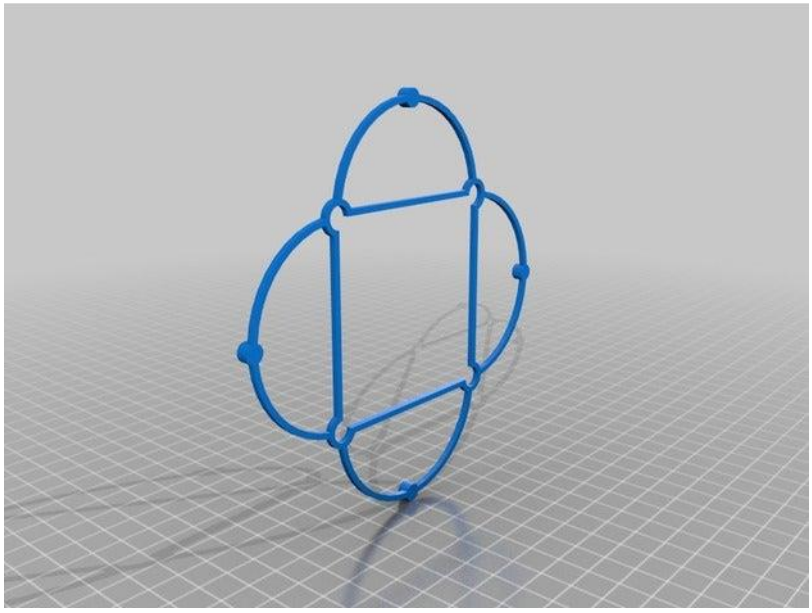

Figura 3-5. Estructura para la colocación de marcadores retrorreflectivos

Sin embargo, esto solo sería una solución provisional. De cara a la escalabilidad sigue existiendo un problema, ya que las opciones terminarían agotándose a partir de un determinado número de drones. En el proyecto Crazyswarm [17] abordaron este problema con un enfoque distinto. En su lugar, decidieron utilizar la misma configuración de reflectores para todos los drones y prescindir del software oficial de VICON. Desarrollaron un software propio que opera directamente sobre la nube de puntos y cuyo funcionamiento consiste en identificar los drones al comienzo, a través de unas posiciones iniciales definidas, y posteriormente realizar un seguimiento basándose en la posición ocupada por el dron en el instante inmediatamente anterior. Los detalles pueden consultarse en [18]. Puesto que el software que desarrollaron (bajo el nombre de libobjecttracker) se encuentra publicado como código abierto, se decidió hacer uso de esta herramienta para eliminar el problema de la distribución espacial de los reflectores.

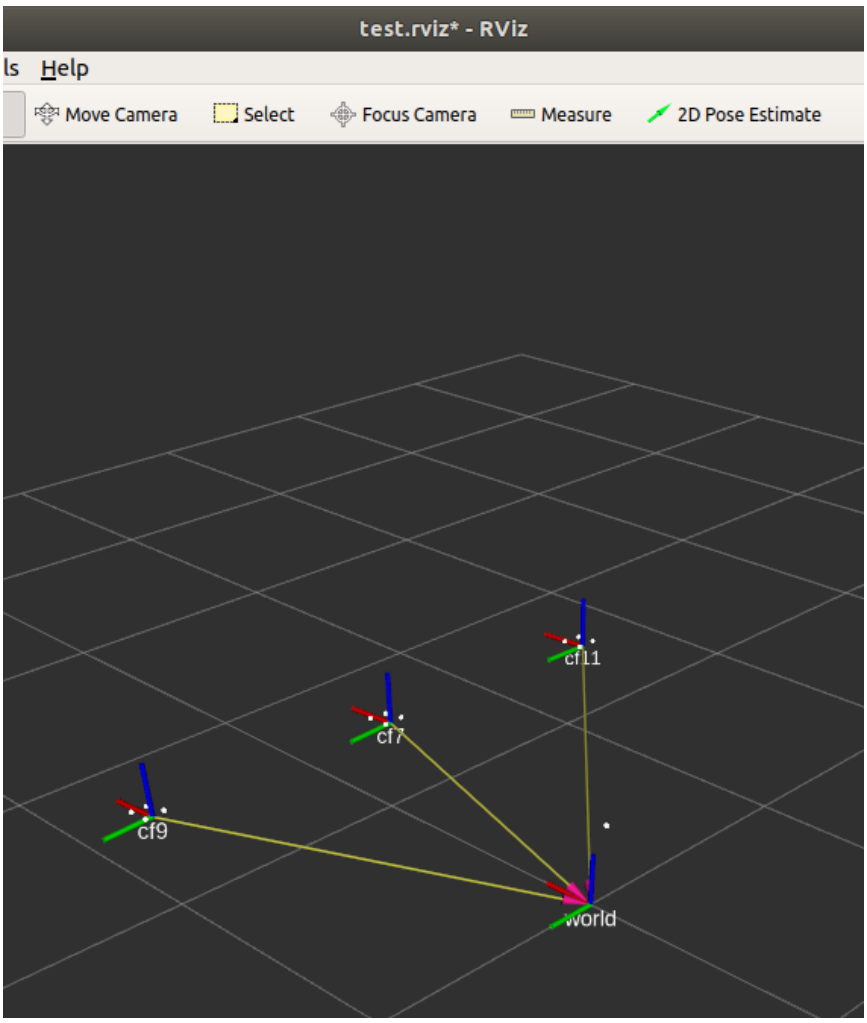

Figura 3-6. Visualización en Rviz de las posiciones obtenidas

Figura 3-7. Colocación de los marcadores.

Para extender el procedimiento expuesto en el aparatado anterior al caso de múltiples drones, se desarrolló un programa multihilo que gestionara el envío de comandos de manera concurrente. Se observó que el tiempo requerido por cada dron para establecer la conexión era muy variable, por ello, fue necesario sincronizar el inicio del movimiento. Cada hilo notifica (a través de una variable de condición) que se encuentra listo para el despegue, y se mantiene a la espera hasta que el resto de los hilos también lo estén.

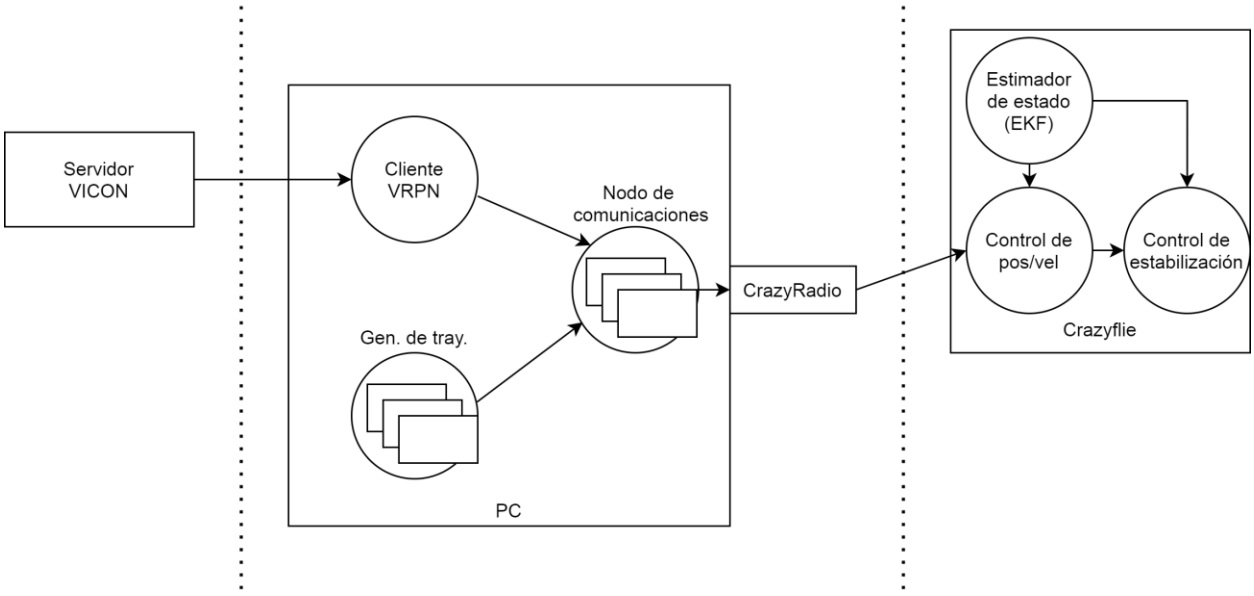

Figura 3-8. Flujo de información del control de posición implementado a bordo. Múltiples Crazyflies.

Nuevamente, volvieron a aparecer dificultades relacionadas con las comunicaciones: a los pocos segundos de iniciar el vuelo, algunos de los drones experimentaban desconexiones repentinas. En esta ocasión, se buscó la causa del problema en la propia librería. La versión actual de crazyflie-lib-python (0.1.12.1) soporta, en principio, el uso compartido de una radio por parte de varios drones. Sin embargo, algunos aspectos de su programación hacen que esto se vuelva inviable cuando se envían comandos a una frecuencia elevada (como ocurre en nuestro caso). En líneas generales, el funcionamiento del módulo radiodriver.py es el siguiente:

Para cada Crazyflie, se crea una cola de mensajes de salida (con capacidad para un único paquete) y se lanza un hilo que pretende hacer uso del recurso compartido (la radio) de manera segura. La exclusión mutua en el acceso a la radio se logra mediante el uso de un objeto threading.Lock(). Cuando alguno de los hilos adquiere el *Lock*, lee un mensaje de su cola y lo envía a través del puerto USB asociado a su radio; mientras, el resto de los hilos permanecen bloqueados a la espera.

Por otra parte, con el protocolo CRTP se intenta garantizar que todos y cada uno de los paquetes sean recibidos por el dron. Para ello, el dron debe devolver un ACK (asentimiento) verificando que ha recibido el paquete. En caso de no obtener el ACK, el PC vuelve a intentar el envío. Dado que el ACK también podría perderse, en la cabecera del mensaje se incorpora un número de secuencia que permite distinguir los paquetes duplicados, así como volver a enviar el ACK si fuera necesario.

Por último, se implementa también un mecanismo de desconexión que provoca la terminación de los hilos asociados a un determinado Crazyflie si no se recibe ningún ACK tras un máximo de 100 reintentos.

De este planteamiento surgen varios problemas:

Cada vez que se pone un paquete en la cola de mensajes (al enviar un comando o la información de localización) se hace mediante una llamada a función que actúa de modo bloqueante cuando esta se ha llenado. Idealmente, esta situación no tendría que darse nunca. Sin embargo, cronometrando la ejecución se observó que, al conectar 2 Crazyflies en la misma radio, aparecían retrasos en el bucle del hilo de control de cada uno de ellos. Cuanto más rápido se pretendía enviar los comandos, mayor era el retraso medio introducido.

Periodo deseado (ms)	Retraso medio (ms)
10	250
20	50
30	25
40	20
50	5

Tabla 3-1. Retraso debido a las llamadas bloqueantes.

Este hecho ponía de manifiesto que se estaba escribiendo en las colas a una tasa mayor que la tasa de lectura. Sin embargo, la radio había demostrado ser capaz de enviar paquetes a 100 y 200 Hz con un solo dron conectado, por lo que el problema debía estar ocasionado por la implementación software. Se observaron dos posibles causas:

1. Por una parte, al utilizar un *mutex* (*lock*) para proteger el recurso compartido no se garantiza que todos los hilos consigan acceder a él dentro del plazo. Es más, podría darse el caso de que alguno de los hilos no lo lograse nunca, ya que el sistema operativo no tiene configurada una política de planificación de tiempo real y, por tanto, el orden en el que se desbloquean los hilos no es determinista.
2. Por otra, el modo en el que se gestionan los ACK parece ser problemático. Cuando el paquete enviado no obtiene un ACK, se vuelve a enviar el mismo en lugar de leer de la cola, lo que provoca que esta se llene. Como no parece que dichas pérdidas ocurran de manera esporádica, esto es indicativo de que existe otro problema subyacente:

   A nivel de USB, cada vez que se envía un paquete, se configura la dirección del dron y seguidamente se copia al buffer de salida del chip. Después, se lee del buffer de entrada para comprobar el ACK y el número de secuencia. A pesar de que es posible configurar dinámicamente la dirección de destino, los buffers de entrada y salida son comunes para todos los drones que se comunican con la radio. En consecuencia, el hilo que gestiona la radio para un determinado dron podría leer del buffer de entrada un paquete con ACK y número de secuencia enviado por alguno de los demás drones. En ese caso, la comprobación fallaría y se ordenaría el reenvío. Esta situación, repetida en el tiempo a una alta frecuencia, acabaría provocando la desconexión de alguno de los drones una vez se llegase al límite de reintentos.

Por todo lo expuesto anteriormente, queda claro que en una aplicación de este tipo es preferible asumir la pérdida ocasional de algún paquete a garantizar su recepción con posibles retrasos. Persiguiendo esta idea, se llevaron a cabo las siguientes modificaciones en la librería:

- Se limitó el tiempo máximo que las llamadas a funciones bloqueantes pueden esperar para escribir en la cola. Dado que estas llamadas se realizan dos veces por ciclo, dicho *timeout* se fijó como la mitad de la duración de un ciclo.
- Se desactivó la comprobación del ACK y número de secuencia; y, por tanto, también el reenvío de paquetes.
- Se desactivó el mecanismo de desconexión que dependía del número de paquetes perdidos.

Al tratar de verificar que las modificaciones tenían un impacto positivo, se observó que en ocasiones surgían dificultades para establecer la conexión inicial con algunos de los drones. Esto se debe a que previamente al envío de los comandos, el dron y el PC necesitan intercambiar cierta información referente a configuraciones y *logging* de variables. Es necesario que este proceso se complete sin pérdidas para poder iniciar el vuelo, por ello, se volvió a habilitar la comprobación de ACK y el reenvío de paquetes únicamente durante esta fase.

Tras realizar estas modificaciones se logró una importante mejoría y fue posible volar varios drones simultáneament, desde una misma antena y sin experimentar desconexiones. Se hicieron varias pruebas para determinar el número de drones por antena, que se acabó fijando en un máximo de tres. A partir de ese número, se apreciaba a simple vista una progresiva degradación del control, debido a que las pérdidas de paquetes de localización eran más frecuentes.

Llegados a este punto, para añadir más drones, teóricamente bastaría con hacer grupos de tres y asociarlos a distintas antenas. Sin embargo, al tratar de hacerlo se volvieron a experimentar casos de desconexiones repentinas, por encima de un total de siete drones distribuidos en tres antenas. Dichas desconexiones son provocadas desde el lado del Crazyflie y son indicativas de que se ha alcanzado el tiempo máximo admisible sin recibir paquetes. La causa de estas pérdidas no está del todo clara, ya que no se está excediendo el máximo de drones por antena anteriormente determinado. Se sospecha que podría estar debida a las limitaciones que existen en la ejecución de aplicaciones multihilo desde el intérprete de Python. Para cada Crazyflie se lanzan los siguientes hilos:

- Hilo principal: bucle de envío de comandos y localización.
- Suscriptor de ROS para la obtención de la localización procedente del sistema de posicionamiento.
- Suscriptor de ROS para la obtención de la trayectoria.
- Hilo de gestión de la radio.

Es decir, para 7 drones se tendría un total de 28 hilos dentro del mismo proceso. Ejecutar un alto número de hilos no es necesariamente algo problemático, sin embargo, todos ellos funcionan a una frecuencia elevada y solo pueden hacer uso de uno de los núcleos del procesador, debido al GIL[13] de Python. Además, como se comentó anteriormente, la ausencia de una política de planificación de tiempo real puede dar lugar a que algunos de los hilos no cumplan con las restricciones temporales necesarias para el correcto funcionamiento del sistema.

Teniendo presentes los problemas de eficiencia de la implementación propuesta (y de la propia librería), en lugar de invertir más tiempo en intentar solucionarlos, se decidió explorar el proyecto Crazyswarm para conocer cómo habían sido abordados por sus desarrolladores.

[13] *Global Interpreter Lock*

3.2. Desarrollo basado en el proyecto Crazyswarm

En el proyecto Crazyswarm también se emplea el enfoque basado en ejecutar el control de posición en el microcontrolador a bordo. Destaca por la implementación de un esquema de comunicaciones eficiente que logra maximizar el número de drones conectados por antena. Cumple las siguientes características:

- Prescinde de la comprobación de ACK y reenvío de aquellos paquetes que son enviados a una alta frecuencia (información de localización y comandos).
- Consigue comprimir en un único paquete la posición y orientación de hasta dos drones.
- Uso de *broadcasting* para el envío de paquetes de localización.

Debido a la compresión y el uso de *broadcasting,* los drones que se encuentran conectados a una misma antena reciben la información de localización de todos los demás y, por lo tanto, deben identificar qué paquetes les corresponden.

En un primer momento, podría cuestionarse la utilidad de este método, ya que la cantidad total de información que ha de enviarse es la misma. Sin embargo, existe una mejora del rendimiento debida a la reducción de solicitudes de escritura en el dispositivo USB. Sirva como prueba la siguiente comparación:

- Cuando se hace uso de la librería oficial, se realizan 2 solicitudes de escritura cada vez que se envía la localización de un dron: la primera configura la dirección de destino y la segunda transfiere el paquete que contiene la posición y orientación.
- Con el software de Crazyswarm, en una única solicitud se consigue transferir 2 paquetes de localización, cada uno de ellos con la posición y orientación de 2 drones. Además, como se usa *broadcasting* no se necesita configurar la dirección de destino.

Es decir, se consigue transferir la misma cantidad de información reduciendo hasta en un factor de 8 el número de solicitudes USB. De esta manera, se maximiza el número de paquetes que pueden ser enviados por unidad de tiempo.

Por otra parte, la implementación del programa principal es también más eficiente: centraliza en el mismo proceso el envío de paquetes y el algoritmo de localización, en lugar de utilizar varios procesos comunicados mediante ROS, por lo que se reduce el número de suscriptores. Además, al estar implementado completamente en C++ se evitan los problemas que en el apartado anterior se asociaban al GIL de Python. El uso de ROS se restringe al envío de comandos de movimiento, es decir, se proporciona una interfaz al usuario para la ejecución de trayectorias que pueden haber sido programadas en diferentes lenguajes, como Python o MATLAB.

3.2.1. Interfaz gráfica

Uno de los inconvenientes que presenta este proyecto es lo tedioso que resulta establecer la configuración al inicio de cada vuelo. Como se comentó anteriormente, emplean un algoritmo propio para obtener la posición de los drones a partir de la nube de puntos generada por los marcadores. En la inicialización de dicho algoritmo es necesario proporcionar la posición inicial de cada dron con bastante exactitud; si se comete un error demasiado grande, es posible que no se logre localizarlo. Además, esta configuración se lleva a cabo rellenando manualmente un fichero de texto.

Para facilitar esta tarea, se ha desarrollado una interfaz gráfica que agilice dicha configuración. Para ello se ha empleado la librería gráfica Qt5 sobre C++.

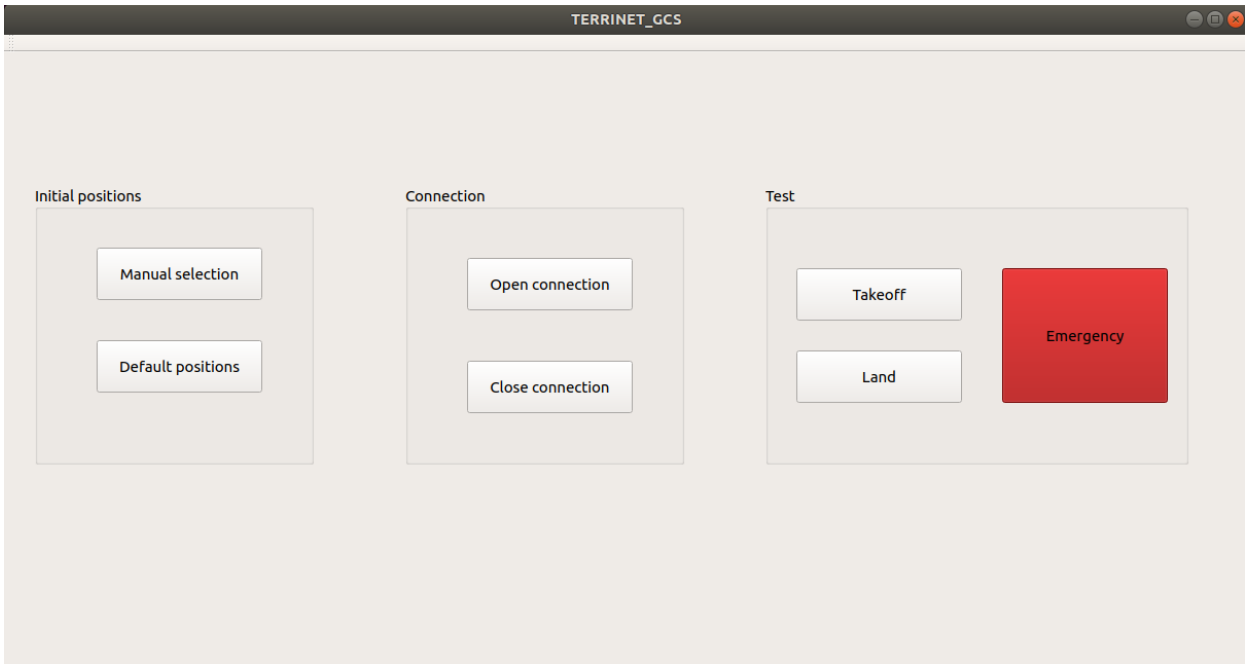

Figura 3-9. Ventana principal de la interfaz gráfica.

Por una parte, se ha incluido un modo que permite visualizar la nube de puntos y asignar los drones de manera interactiva. De esta forma, el usuario puede iniciar el vuelo desde cualquier lugar sin necesidad de conocer las coordenadas, únicamente necesita seleccionar el dron e indicar el número que lo identifica. Además, se garantiza que el algoritmo será capaz de localizar al dron, ya que la posición inicial es capturada de manera exacta.

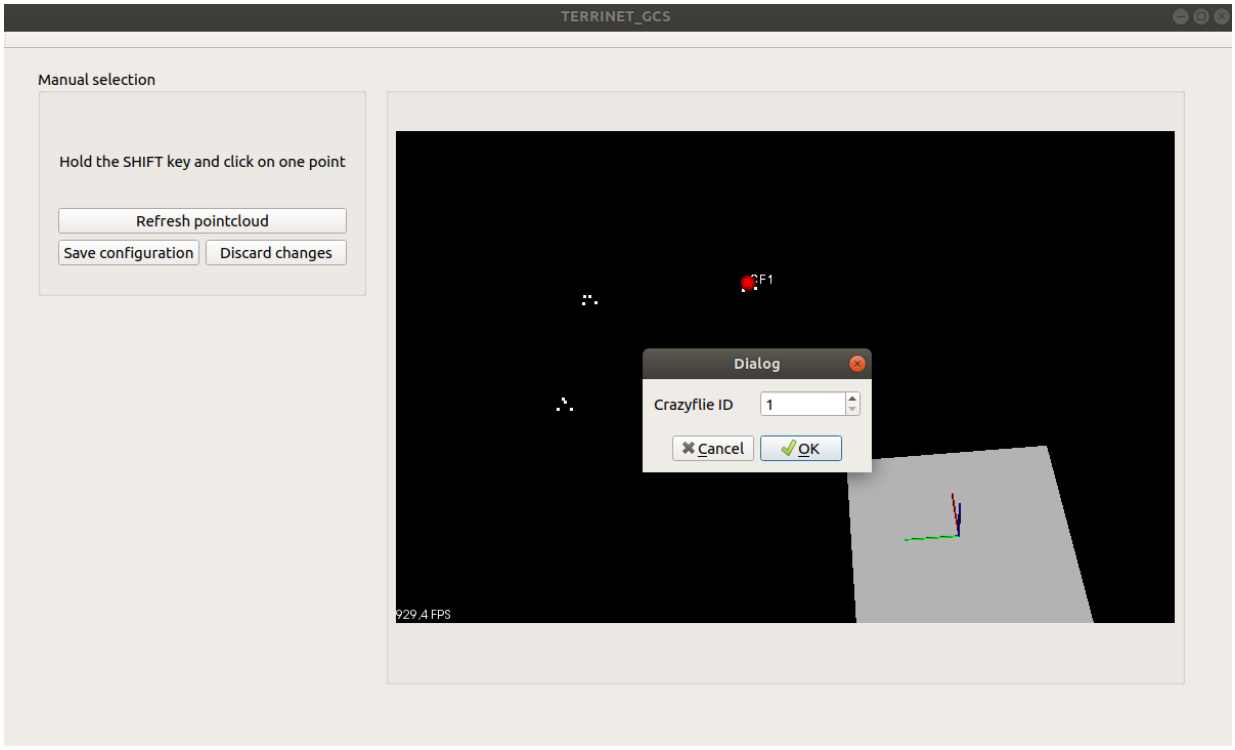

Figura 3-10. Ventana de selección manual.

Por otra parte, se añade también la posibilidad de empezar en unas posiciones definidas por defecto. En este caso, únicamente se ha de seleccionar el número de identificación de los drones que van a activarse.

Al pulsar el botón *Open connection* se inicia el nodo principal de Crazyswarm, crazyswarm_server, que se encarga de establecer la comunicación con los drones previamente seleccionados, así como de localizarlos y enviarles su posición a una frecuencia de 100 Hz. Este nodo de ROS también centraliza el envío de comandos; para ello, inicia una serie de suscriptores destinados a recibir las referencias de posición. De esta forma, el usuario únicamente ha de publicarlas en el *topic /cfX/cmd_position*, siendo X el identificador del dron correspondiente.

Los botones *takeoff* y *land* tienen como propósito servir como verificación de que todo funciona correctamente. El botón *emergency* permite enviar la orden de parada a todos los drones.

3.2.2. Nodo de envío de comandos

A modo de ejemplo de uso, se ha desarrollado un nodo que tiene como finalidad coordinar el movimiento de varios drones e iniciar el seguimiento de una trayectoria. De cara a mantener una estructura modular, la generación de trayectorias sigue teniendo lugar en un nodo independiente.

El programa inicia un hilo por cada uno de los Crazyflies activos. Cuando el usuario da la orden, despegan todos simultáneamente hasta una altura predefinida y vuelven a permanecer a la espera hasta que se inicie la trayectoria. Esta decisión pretende evitar problemas asociados al despegue, por ejemplo, que las referencias en XY al comienzo no coincidan de manera exacta con la posición inicial del dron. En ese caso, el control intentaría alcanzarlas cuando el dron aun no ha despegado, lo que podría provocar que se desestabilizase al empezar a volar.

Se han configurado dos modos de funcionamiento:

- En el modo absoluto, el generador de trayectorias calcula una trayectoria para cada dron, especificada en coordenadas del sistema de referencia del suelo.
- En el modo relativo, el generador produce una única trayectoria, a la que posteriormente se le suma un *offset* igual a la posición inicial de cada dron.

La función de cada hilo consiste en una máquina de estados que comprende los siguientes:

1. Despegue. Se entra en este estado cuando el usuario da la orden y sólo si el nodo del generador de trayectorias está disponible.
2. Vuelo. Cuando el usuario da la orden se inicia el vuelo. En este estado la trayectoria es publicada en el topic *cmd_position* a una frecuencia fija.
3. Aterrizaje. La trayectoria de cualquiera de los drones puede ser interrumpida en cualquier momento mediante la llamada a una servicio. En este estado, en lugar de comandar posiciones se controla en velocidad para no depender de unas coordenadas concretas. Se comanda una velocidad de descenso hasta que la altura es lo suficientemente cercana al suelo.

3.2.3. Determinación del número de UAVs por antena

Mediante la herramienta comCheck (incluida en Crazyswarm) es posible medir el número de paquetes por segundo que es capaz de enviar la antena desde un PC determinado. Se realizaron dos pruebas:

- Conectando la antena directamente al puerto USB se puede enviar aproximadamente 500 paquetes por segundo.
- Al conectar la antena a un *hub* USB con alimentación externa, el número de paquetes ascendía hasta 700.

La explicación de este inesperado comportamiento no está muy clara, pero podría tener relación con los niveles de alimentación o con el método empleado por el *hub* para gestionar las peticiones USB.

Teniendo en cuenta el mecanismo de compresión empleado por Crazyswarm, se calculó el número de paquetes por segundo que deberían enviarse para realizar un control de posición a 100 Hz.

Número de drones	Número de paquetes de localización por ciclo	Número de paquetes de comandos por ciclo	Número de paquetes por segundo
1	1	1	200
2	1	2	300
3	2	3	500
4	2	4	600
5	3	5	800

Tabla 3-2. Estimación del tráfico de la antena.

Poniéndonos en el caso más desfavorable, número de drones por antena estaría limitado a 3. Por encima de ese número podría empezar a tener lugar la pérdida de paquetes. Se dispone de 4 antenas y 10 drones, por lo que es posible distribuirlos cumpliendo con dicha restricción.

Si se desease aumentar el número de drones por antena, podrían relajarse las especificaciones temporales de dos formas:

- Reduciendo la tasa de envío de paquetes de localización. Es posible llevar esto a cabo sin afectar significativamente al control, debido a que en el microcontrolador se está ejecutando el filtro de Kalman a una frecuencia fija. Implica modificar el código del nodo crazyswarm_server.*cpp*
- Reduciendo la tasa de envío de paquetes de comandos. El muestreo de la referencia también puede hacerse a una frecuencia menor. Para ello, basta con modificar la frecuencia con la que se publica en el *topic* /*cmd_position*.

4 RESULTADOS EXPERIMENTALES

En este capítulo se mostrarán los resultados de varias pruebas de vuelo, con el propósito de validar experimentalmente el buen funcionamiento del sistema. Se ha hecho uso de la herramienta rosbag para capturar la información publicada en tiempo real en los diferentes *topics*.

4.1. Un único dron. Movimiento relativo.

En este experimento se ha especificado una trayectoria en coordenadas relativas a la posición de inicio. El UAV asciende 30 cm y recorre un cuadrado de 1 metro de lado en el plano XY, a una velocidad de aproximadamente 0.5 m/s. Los máximos errores cometidos se encuentran en torno a los 5 cm.

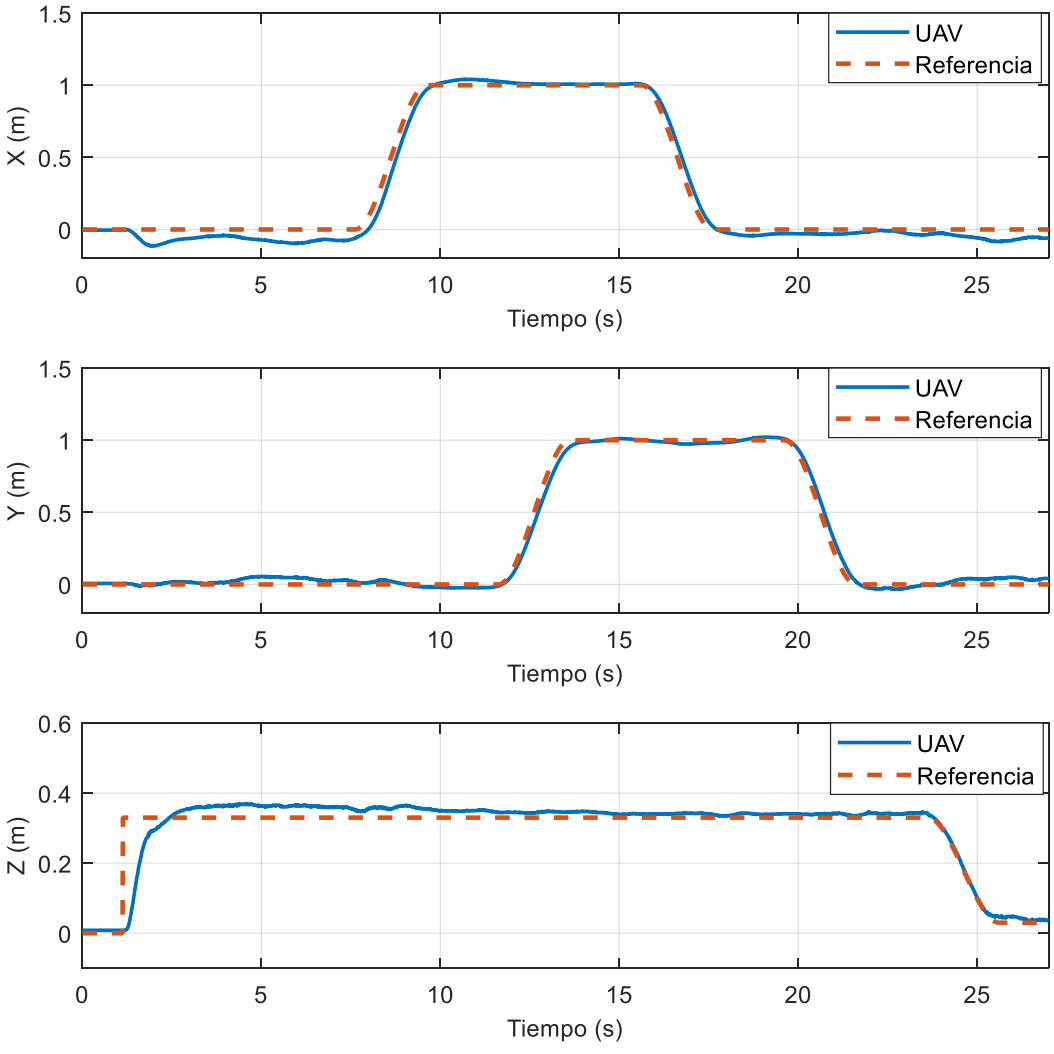

Figura 4-1. Seguimiento de una trayectoria.

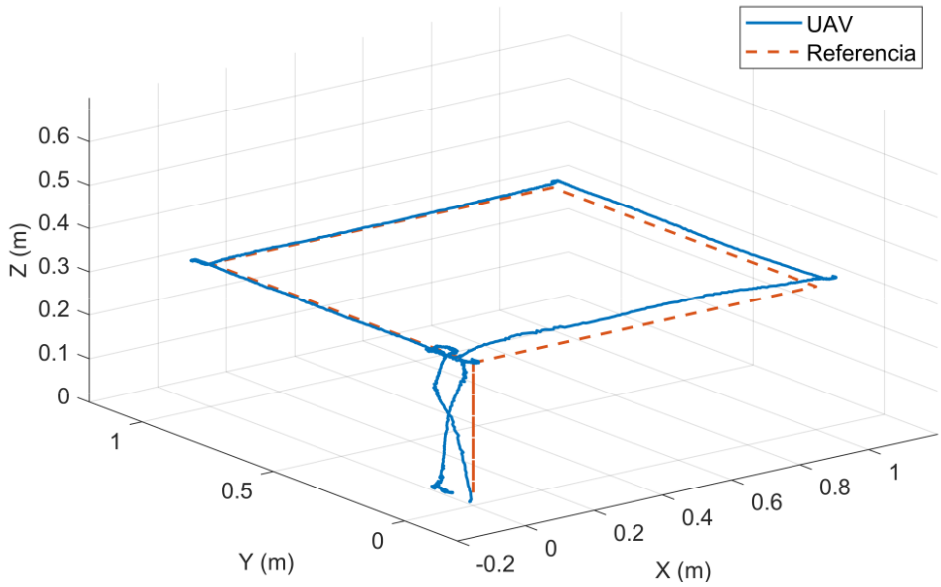

Figura 4-2. Seguimiento de una trayectoria. Vista 3D.

4.2. Tres drones. Movimiento relativo.

Para verificar que no existen problemas relacionados con las comunicaciones, se repite el experimento anterior con 3 UAVs conectados a la misma antena. Se especifica una única trayectoria en coordenadas relativas, que será convertida a coordenadas absolutas, para cada dron, sumando su posición inicial.

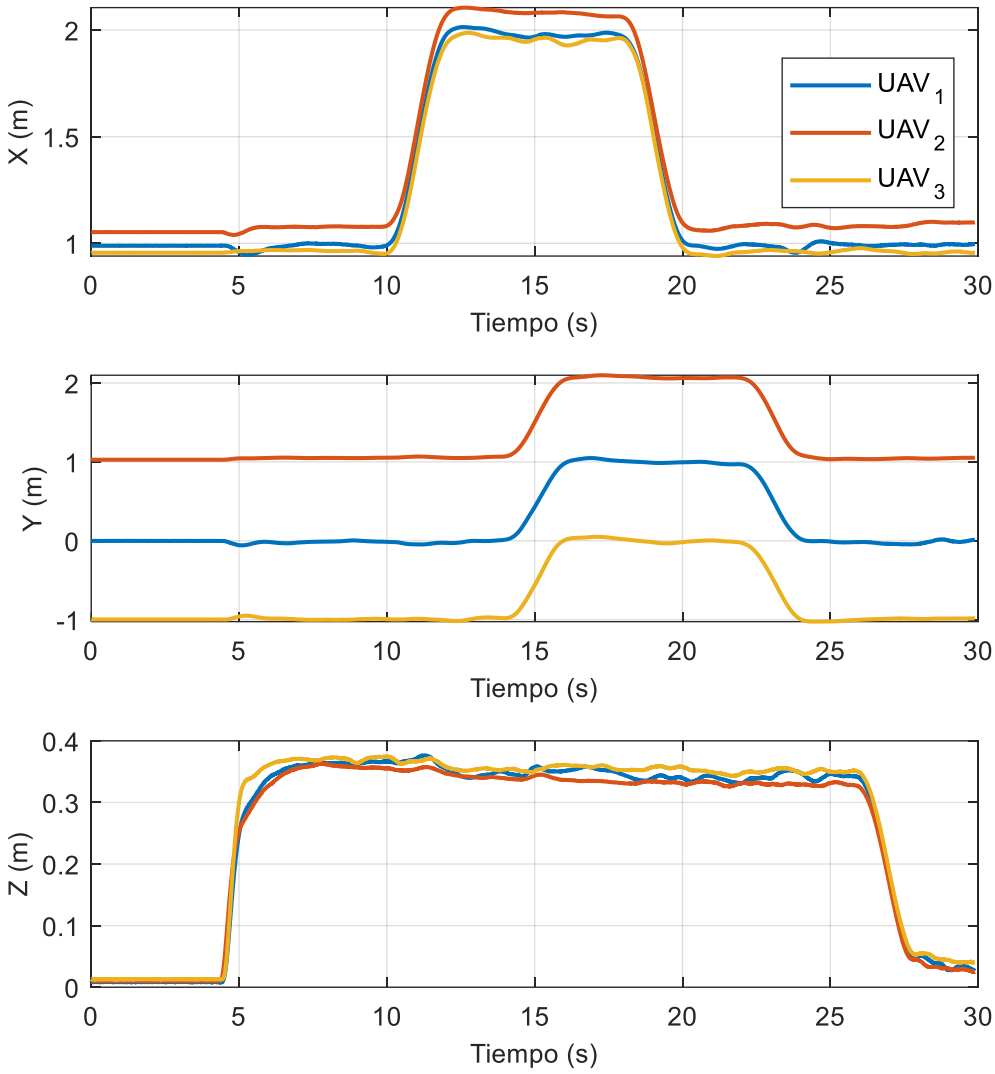

Figura 4-3. Seguimiento de una trayectoria por tres UAVs simultáneamente.

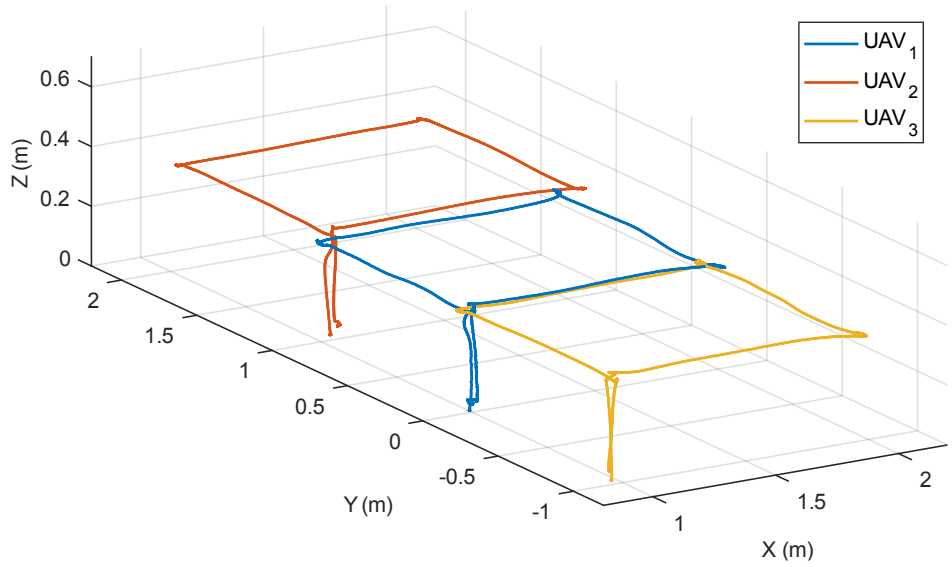

Figura 4-4. Seguimiento de una trayectoria por tres UAVs simultáneamente. Vista 3D

4.3. Tres drones. Vuelo en formación.

En este experimento se pretende lograr un movimiento coordinado algo más complejo. Una opción sería planificar cada una de las trayectorias de manera independiente, pero es una tarea compleja que requiere poner especial atención en la evitación de colisiones. En su lugar, la estrategia empleada consiste en considerar que los drones forman parte de una estructura virtual, la cual se comporta como un sólido rígido. En el centro de dicha estructura situamos su sistema de referencia y, en él, definimos las posiciones que ocupan los drones dentro de la formación (en este caso, los vértices de un triángulo equilátero). Se planifica una única trayectoria, la cual definirá, en cada instante, la posición y orientación del sistema de referencia de la formación con respecto al sistema de referencia inercial; a partir de ella, se obtienen las trayectorias de cada uno de los drones. La trayectoria definida para esta prueba consiste en una traslación en el plano XZ y una rotación de 90º en torno al eje Z móvil.

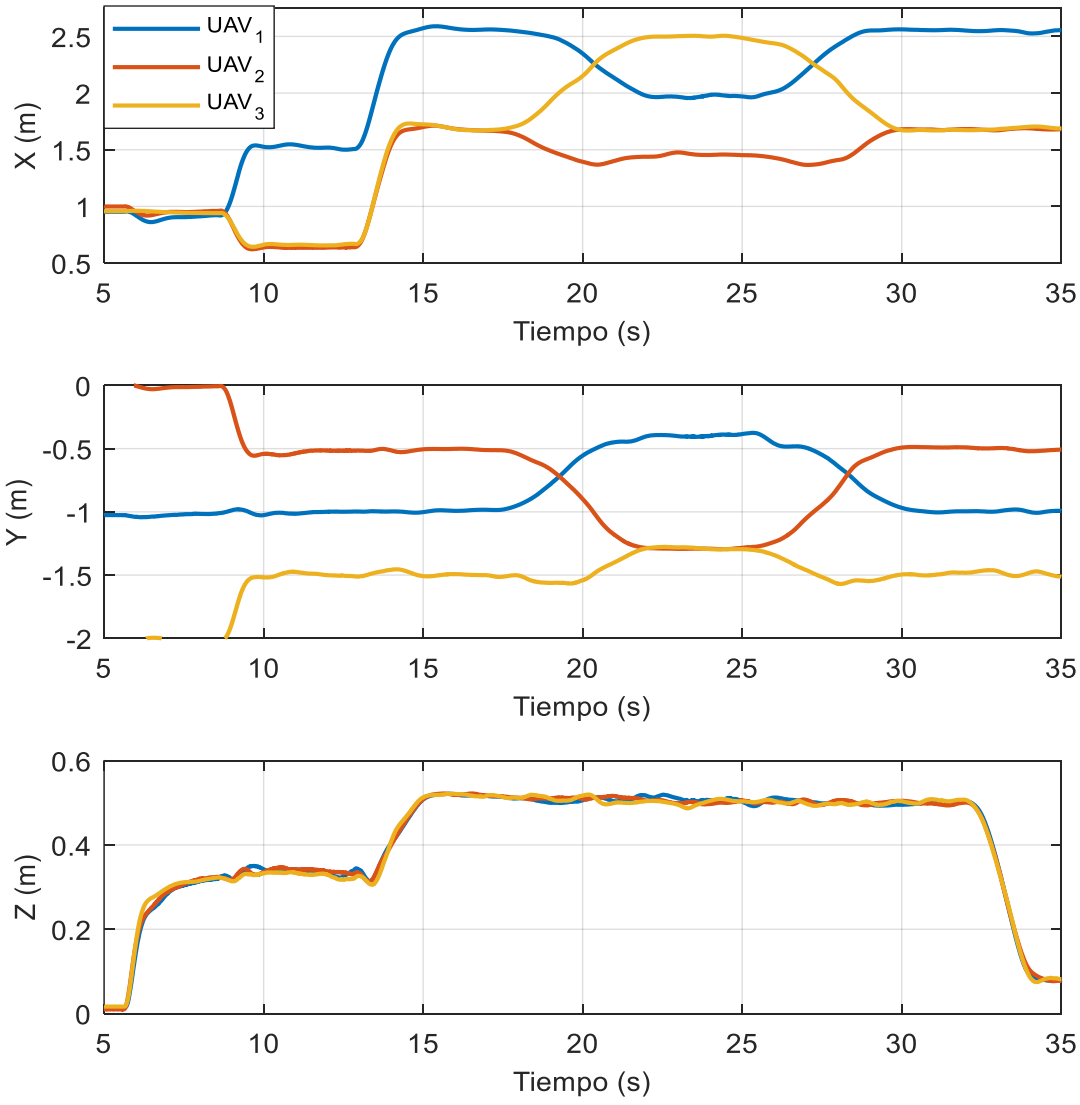

Figura 4-5. Vuelo en formación de tres UAVs.

Los drones despegan simultáneamente y desde posiciones arbitrarias, por ello, en el momento de iniciar la formación surge el problema de asignar un vértice a cada dron. La solución adoptada consiste en hacer coincidir, inicialmente, el origen del sistema de referencia de la formación con el centro geométrico definido por las posiciones de los drones, y asignar a cada uno el vértice más cercano. De esta forma, se asegura que no habrá colisiones durante la reorganización.

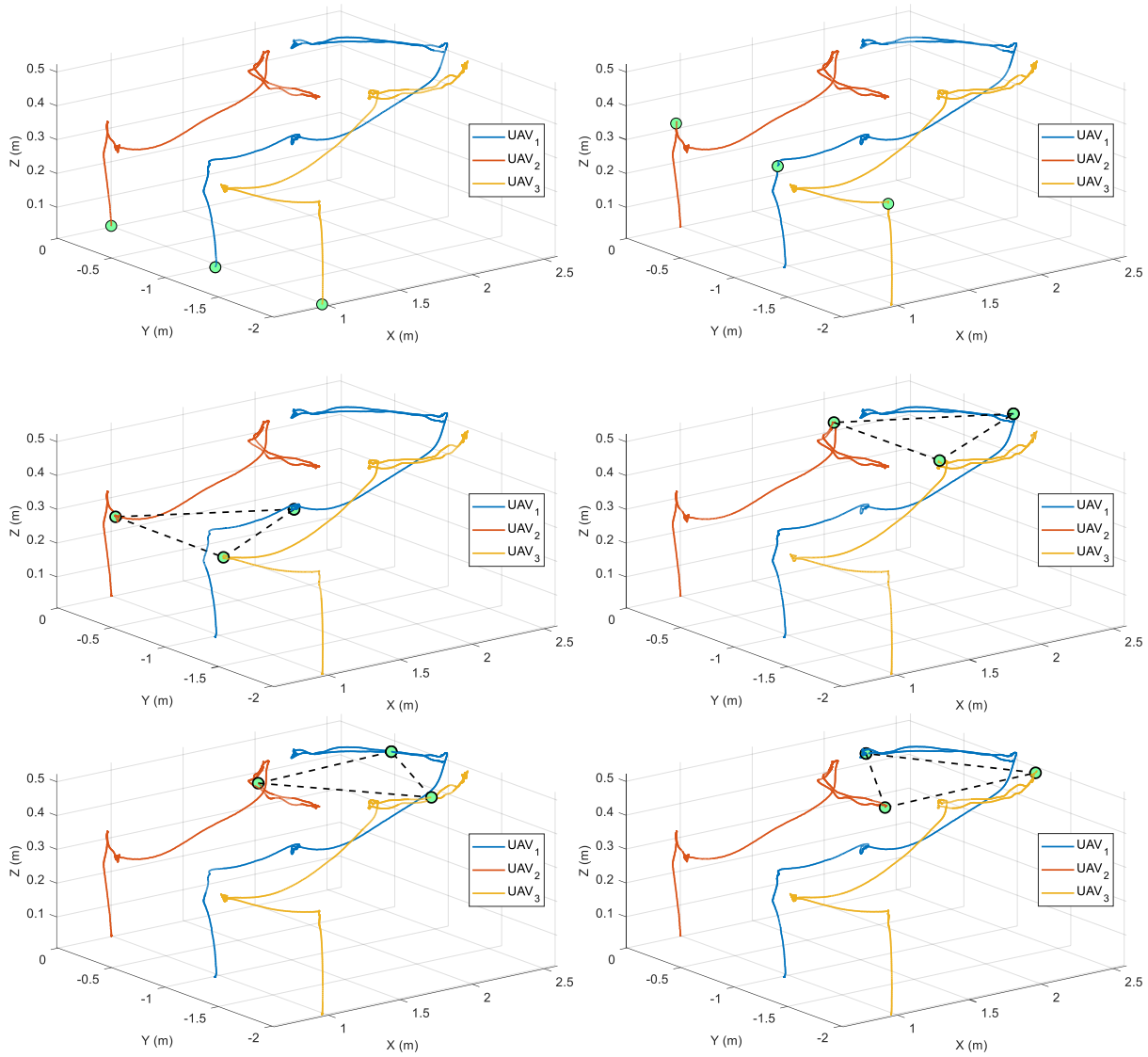

Figura 4-6. Vuelo en formación de tres UAVs. Vista 3D.

4.4. Nueve drones. Movimiento relativo.

Repetición del experimento del apartado 4.2, en este caso, con un total de 9 UAVs (3 por antena). Se verifica que no existen problemas de escalabilidad al aumentar el número de UAVs en esta proporción.

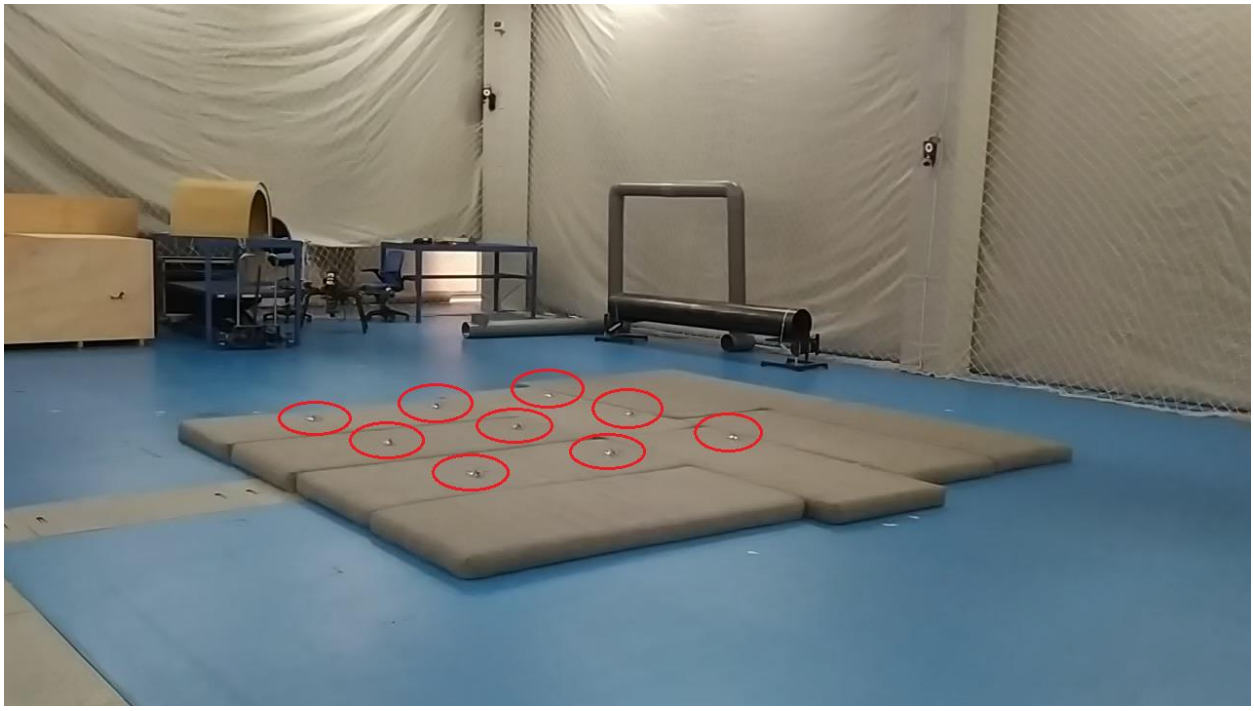

Figura 4-7. Nueve UAVs antes del despegue.

Figura 4-8. Nueve UAVs tras el despegue.

5 CONCLUSIONES Y TRABAJO FUTURO

Se ha llevado a cabo la puesta en marcha de una infraestructura que permite el vuelo simultáneo de hasta 10 drones. En este proceso, se ha adquirido un profundo conocimiento de todas las partes que componen el sistema como consecuencia de los problemas que han surgido y las decisiones que han tenido que tomarse para solventarlos. Se ha comprobado que, en una aplicación de este tipo, aspectos como el control, la estimación de estado, las comunicaciones o la programación de Sistemas de Tiempo Real juegan papeles fundamentales.

A pesar de que los drones que se han empleado sean un producto comercial, su uso en aplicaciones multi-UAV no está exento de problemas y se encuentra fuertemente condicionado por el hardware adicional del que se disponga. La filosofía de la empresa que lo comercializa (Bitcraze) hace que se trate de un producto que, en cierto sentido, esté en constante fase de desarrollo (proceso al que invitan a participar a todos sus usuarios). Por ello, a día de hoy no es una solución "*out-of-the-box*[14]", pero sí se hace posible abordar por uno mismo dichos problemas gracias a la accesibilidad que proporciona el código abierto.

Concretamente, con la librería oficial se experimentaron dificultades a la hora de volar múltiples UAVs desde una misma antena. Se examinó en profundidad el funcionamiento del código para detectar las causas y se llevaron a cabo modificaciones que permitieron mitigarlas hasta cierto punto. Como la solución aún presentaba problemas de robustez, se decidió prescindir de la librería oficial y hacer uso del proyecto Crazyswarm. Dicho proyecto ha sido desarrollado por investigadores ajenos a la empresa y pretende suplir las carencias de la librería oficial, además, también está disponible como código abierto. Si bien es cierto que Crazyswarm ha demostrado comportarse adecuadamente frente a un número elevado de drones, su uso y configuración pueden resultar difíciles a aquellos usuarios que no estén demasiado familiarizados con Linux y ROS, por ello, se creyó conveniente desarrollar una interfaz gráfica que facilitase dicha tarea.

A pesar de que se sabía de la existencia de Crazyswarm prácticamente desde el inicio, se trató de seguir el camino lógico que hubiese sido necesario recorrer si dicho proyecto no existiese o, alternativamente, si se utilizasen otros drones distintos a los Crazyflies. Durante este proceso, se ha perseguido un cierto compromiso entre prestaciones, escalabilidad y robustez. Teniendo esto presente, pueden extraerse las siguientes conclusiones:

1. De cara a la escalabilidad, es deseable maximizar el número de UAVs conectados a una misma antena. En la práctica, el número de antenas que pueden utilizarse simultáneamente va a venir limitado por la disponibilidad de los canales de radiofrecuencia. Para disminuir las interferencias, las antenas deben operar en canales suficientemente espaciados entre sí. Dado que la banda de 2.4 GHz es utilizada por multitud de dispositivos, dependiendo del entorno, puede ser difícil encontrar suficientes canales de calidad para un número elevado de UAVs. Por otra parte, el máximo número de

[14] En el contexto del software, la expresión *out-of-the-box* (OOTB) se refiere a aquellos productos que funcionan correctamente de manera inmediata, sin requerir ninguna modificación o configuración especial.

dispositivos USB que un PC convencional es capaz de gestionar puede ser moderado, aun haciendo uso de *hubs*. La principal desventaja es que el ancho de banda efectivo para cada UAV se reduce, ya que la compartición de la antena se lleva a cabo multiplexando su uso en el tiempo. Es importante asegurar que se dispone de suficiente ancho de banda como para cumplir con las restricciones temporales.

2. Se ha seguido un enfoque centralizado: toda la información se encuentra disponible en el PC y desde ahí se dan las órdenes a los UAVs. La principal ventaja es la facilidad para probar nuevos algoritmos, que pueden ser programados mediante un lenguaje de alto nivel y ejecutados en un equipo con mayores capacidades computacionales. El problema asociado a la centralización es la necesidad de mantener una comunicación constante. En consecuencia, el sistema es poco robusto ante pérdidas de conexión. Dado que la estimación de estado del Crazyflie depende en gran medida del sistema de posicionamiento externo, no es posible implementar con seguridad una rutina de emergencia como aterrizar o mantener la posición; la opción más segura es cortar la alimentación a los motores.

3. El uso del sistema de posicionamiento VICON tiene la ventaja de ofrecer una estimación con una precisión difícil de igualar por otros métodos. A pesar de esto, presenta varios inconvenientes a la hora de trabajar con un número elevado de UAVs. Por una parte, el software oficial no soporta la detección de múltiples objetos con igual distribución de marcadores (lo cual es una necesidad, debido al reducido tamaño de los Crazyflies). Esto fue abordado haciendo uso de un software de terceros [18] que, si bien es cierto que funciona correctamente, requiere que se especifiquen las coordenadas iniciales con la suficiente exactitud. Otro problema es la falta de robustez ante situaciones en las que se pierda la visibilidad de algunos marcadores durante un periodo de tiempo prolongado.

4. La localización proporcionada por VICON puede usarse para cerrar el bucle de control de posición, bien en el PC central, bien en el microcontrolador a bordo. La segunda opción dio mejor resultado y es más robusta ante pérdidas puntuales de paquetes, ya que el estimador interno (EKF) es capaz de seguir proporcionando estimaciones de calidad durante los ciclos en los que no se reciben medidas. En ambos casos, se experimentaron problemas ocasionados por la implementación del protocolo CRTP (*Crazy RealTime Protocol*) de la librería oficial. A pesar de su nombre, posee características que interfieren con las restricciones de tiempo real del problema de control, por ejemplo, trata de asegurar la recepción de un paquete reenviándolo las veces que sea necesario (introduciendo retrasos). Se llegó a la conclusión de que era preferible asumir la pérdida de paquetes a recibirlos fuera de plazo, y se realizaron las pertinentes modificaciones en la librería para que esto fuera así.

Como trabajo futuro se propone estudiar alternativas para reducir la dependencia de los Crazyflies con el PC central. Por ejemplo, podría plantearse la sustitución del sistema de posicionamiento VICON por un sistema basado en UWB, ya que el segundo permitiría estimar la posición a bordo, disminuyendo así la congestión en las comunicaciones y favoreciendo la escalabilidad. Además, se evitarían problemas asociados a la falta de visibilidad de los reflectores y se simplificaría el uso del sistema, pudiéndose prescindir de la configuración inicial que el usuario ha de llevar a cabo para identificar a cada UAV.

Otra posible línea de desarrollo futuro sería habilitar la comunicación entre UAVs. Actualmente existe una API P2P en el firmware pensada para este fin, sin embargo, aún se encuentra en fase de desarrollo. El uso de esta API permitiría la ejecución de algoritmos cooperativos de una forma verdaderamente descentralizada.

REFERENCIAS

[1] Skorobogatov, Georgy & Barrado, Cristina & Salamí, Esther. (2019). Multiple UAV Systems: A Survey. Unmanned Systems. 08. 10.1142/S2301385020500090.

[2] Maza, I., Ollero, A., Casado, E., Scarlatti, D. (2015) Classification of Multi-UAV Architectures. In: Valavanis K., Vachtsevanos G. (eds) Handbook of Unmanned Aerial Vehicles. Springer, Dordrecht. https://doi.org/10.1007/978-90-481-9707-1_119.

[3] S. Chung, A. A. Paranjape, P. Dames, S. Shen and V. Kumar, "A Survey on Aerial Swarm Robotics," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 837-855, Aug. 2018, doi: 10.1109/TRO.2018.2857475.

[4] Perez, D., Maza, I., Caballero, F. et al. A Ground Control Station for a Multi-UAV Surveillance System. J Intell Robot Syst 69, 119–130 (2013). https://doi.org/10.1007/s10846-012-9759-5

[5] N. Zhao et al., "UAV-Assisted Emergency Networks in Disasters," in IEEE Wireless Communications, vol. 26, no. 1, pp. 45-51, February 2019, doi: 10.1109/MWC.2018.1800160.

[6] R. Ritz and R. D'Andrea, "Carrying a flexible payload with multiple flying vehicles," 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, 2013, pp. 3465-3471, doi: 10.1109/IROS.2013.6696850.

[7] Torres-González, A. & Capitán, Jesús & Cunha, Rita & Ollero, Anibal & Mademlis, Ioannis. (2018). A Multidrone Approach for Autonomous Cinematography Planning. Advances in Intelligent Systems and Computing. 337-349. 10.1007/978-3-319-70833-1_28.

[8] Viguria, A., Maza, I., Ollero, A.: Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions. Advanced Robotics 24 (1–2), 1–23 (2010)

[9] Kushleyev, A., Mellinger, D., Powers, C. et al. Towards a swarm of agile micro quadrotors. Auton Robot 35, 287–300 (2013). https://doi.org/10.1007/s10514-013-9349-9

[10] S. Lupashin, A. Schöllig, M. Hehn and R. D'Andrea, "The Flying Machine Arena as of 2010," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 2970-2971, doi: 10.1109/ICRA.2011.5980308.

[11] J. P. HOW, B. BEHIHKE, A. FRANK, D. DALE and J. VIAN, "Real-time indoor autonomous vehicle test environment," in IEEE Control Systems Magazine, vol. 28, no. 2, pp. 51-64, April 2008, doi: 10.1109/MCS.2007.914691.

[12] Amador, J.M. & Martinez-de Dios, J. Ramiro & L. Paneque, Julio & Ollero, A.. (2019). An Agile Low-cost Testbed for Multi-Drone Target Tracking. 344-351. 10.1109/REDUAS47371.2019.8999718.

[13] Purta, R. & Dobski, Mikołaj & Jaworski, A. & Madey, G.. (2013). A testbed for investigating the UAV swarm command and control problem using DDDAS. Procedia Computer Science. 18. 2018-2027. 10.1016/j.procs.2013.05.371.

[14] A. Weinstein, A. Cho, G. Loianno and V. Kumar, "Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors," in IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1801-1807, July 2018, doi: 10.1109/LRA.2018.2800119.

[15] https://www.bitcraze.io/portals/research/

[16] https://wiki.bitcraze.io/projects:crazyflie:crtp

[17] https://crazyswarm.readthedocs.io/en/latest/

[18] J. A. Preiss, W. Honig, G. S. Sukhatme and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 3299-3304, doi: 10.1109/ICRA.2017.7989376.

[19] M. Faessler, D. Falanga and D. Scaramuzza, "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight," in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 476-482, April 2017, doi: 10.1109/LRA.2016.2640362.

[20] Mellinger, Daniel, "Trajectory generation and control for quadrotors" (2012). Dissertations available from ProQuest. AAI3509215. https://repository.upenn.edu/dissertations/AAI3509215

[21] M. Euston, P. Coote, R. Mahony, J. Kim and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, 2008, pp. 340-345, doi: 10.1109/IROS.2008.4650766.

[22] Mueller, Mark W., Michael Hamer, and Raffaello D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation." 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015.

[23] Mueller, Mark W., Markus Hehn, and Raffaello D'Andrea. "Covariance correction step for kalman filtering with an attitude." Journal of Guidance, Control, and Dynamics 40.9 (2017): 2301-2306.

[24] https://github.com/bitcraze/crazyflie-firmware/blob/2020.04/src/modules/src/kalman_core.c

[25] Thrun, S., Burgard, W.,, Fox, D. (2005). Probabilistic robotics.. MIT Press. ISBN: 978-0-262-20162-9