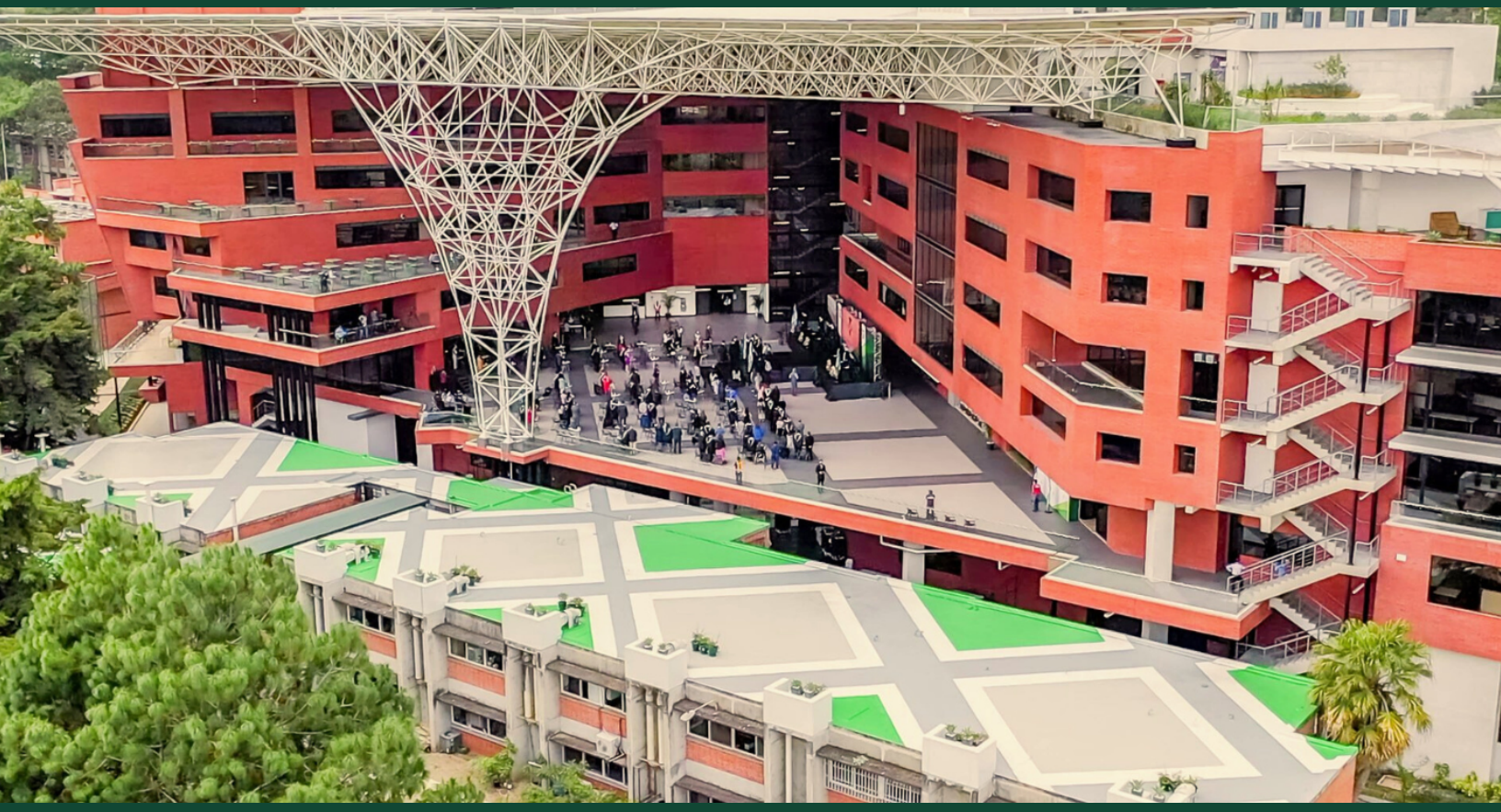
Desarrollo de herramientas de software para el control individual y seguro del cuadricóptero Crazyflie 2.1 utilizando la placa de expansión de posicionamiento con odometría visual Flow Deck

Pablo Javier Caal Leiva



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de herramientas de software para el control individual y seguro del cuadricóptero Crazyflie 2.1 utilizando la placa de expansión de posicionamiento con odometría visual Flow Deck

Trabajo de graduación presentado por Pablo Javier Caal Leiva para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

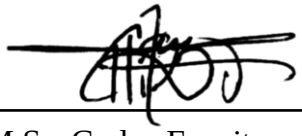
2025

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

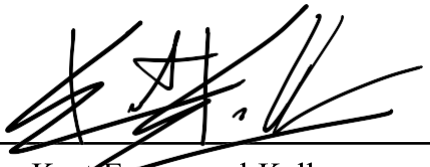


(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales



(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Desde que era pequeño, descubrí una facilidad natural por las matemáticas y un profundo interés por la tecnología. Con el paso de los años, estos intereses evolucionaron y se fusionaron con nuevas aspiraciones que despertaron en mí una fascinación por la ingeniería. Fue entonces cuando definí una meta clara: convertirme en un profesional capaz de innovar y contribuir al desarrollo tecnológico. Este proyecto representa un paso importante hacia el cumplimiento de esa meta, un logro que no habría sido posible sin el apoyo invaluable de las personas y las instituciones que me han acompañado en este camino.

En primer lugar, quiero expresar mi sincero agradecimiento a mi asesor, Msc. Miguel Zea, por su guía constante y sus valiosas aportaciones durante el desarrollo de este proyecto. Su experiencia y conocimientos fueron fundamentales para la realización de este trabajo.

A mi familia, quienes han sido fuente de motivación y fortaleza; mis padres, hermanos y abuelos, gracias por su amor incondicional, su apoyo constante y su confianza en cada paso de mi camino. Este logro también es suyo.

Mi más profundo agradecimiento también a la Fundación Juan Bautista Gutiérrez, en especial a Doña Isabelita, por darme la oportunidad de estudiar la carrera de mi elección en una de las mejores universidades del país. Mi gratitud se extiende a mis mentoras de la fundación, quienes fueron un apoyo esencial durante mis años universitarios, ofreciéndome orientación y respaldo en los momentos más importantes.

Al Centro Universitario Ciudad Vieja, gracias por acogerme como un joven proveniente del interior del país y por formarme integralmente como profesional. A las personas responsables de este centro y a quienes tuve el honor de conocer allí, gracias por ser parte de esta experiencia transformadora.

Asimismo, agradezco a la Universidad del Valle de Guatemala, que me brindó un entorno para crecer académica y profesionalmente. Finalmente, a mis amigos y compañeros, gracias por su apoyo en los momentos de estrés y alegría. Cada uno de ustedes contribuyó a que este proceso fuera más llevadero y significativo.

A todos, gracias por formar parte de este viaje.

Índice general

Índice de figuras

Resumen

El objetivo principal de este trabajo fue desarrollar herramientas de *software* que permitieran el control individual y seguro del dron Crazyflie 2.1 mediante la integración de la placa de posicionamiento Flow Deck, basada en odometría visual. Esto con el fin de ampliar las aplicaciones educativas en la Universidad del Valle de Guatemala, facilitando su uso en prácticas de laboratorio y en entornos de aprendizaje relacionados con robótica y sistemas de control.

Durante el desarrollo del proyecto, se completó exitosamente la integración de *hardware*, logrando que la placa de expansión Flow Deck funcionara de forma óptima como sistema de posicionamiento del dron. También, se desarrollaron funciones de control utilizando la librería cflib, adaptadas para ser utilizadas desde el entorno de trabajo de Matlab. Estas funciones permitieron ejecutar experimentos que van desde el despegue hasta el aterrizaje, incluyendo el seguimiento de trayectorias planificadas mediante métodos de interpolación en una pista de obstáculos. Además, utilizando las herramientas desarrolladas, se elaboraron guías de laboratorio para los cursos de Sistemas de Control y Robótica. Finalmente, se creó un manual de usuario que detalla el proceso de ensamblaje del dron Crazyflie, la integración de la placa de expansión Flow Deck, la instalación de las dependencias y el uso del paquete de herramientas de *software*.

Este trabajo sienta las bases futuras para más aplicaciones de robótica educativa utilizando las herramientas de *software* desarrolladas y proporciona recursos valiosos para la enseñanza de los cursos mencionados, mejorando la experiencia práctica de los estudiantes.

Palabras clave: robótica, drones, sistemas de control, sistemas de posicionamiento, *Robotics, drone, control systems, positioning system.*

Abstract

The main objective of this work was to develop software tools that enable the individual and safe control of the Crazyflie 2.1 drone by integrating the Flow Deck positioning board, based on visual odometry. This was done to expand the educational applications of the Crazyflie drone at Universidad del Valle de Guatemala, facilitating its use in laboratory practices and learning environments related to robotics and control systems.

During the development of the project, the hardware integration was successfully completed, ensuring the Flow Deck expansion board functioned optimally as a positioning system for the drone. Additionally, control functions were developed using the cflib library, adapted for use in the Matlab environment. These functions allowed the execution of experiments ranging from take-off and landing to the tracking of trajectories planned using interpolation methods in an obstacle course. Furthermore, using the developed tools, laboratory guides were created for the Control Systems and Robotics courses. Finally, in addition to the main objectives, a user manual was created detailing the assembly process of the Crazyflie drone, the integration of the Flow Deck expansion board, the installation of dependencies, and the usage of the software tools package.

This work lays the groundwork for future applications in educational robotics by utilizing the software tools developed and providing valuable resources for teaching the aforementioned courses, enhancing the students' practical learning experience.

Keywords: Robotics, drone, control systems, positioning system.

CAPÍTULO 1

Introducción

En los últimos años, los drones han incrementado su relevancia en diversas áreas, desde la investigación científica hasta aplicaciones industriales. Esta importancia se debe a su notable capacidad de adaptación a distintos entornos y las soluciones innovadoras que pueden ofrecer.

La Universidad del Valle de Guatemala (UVG) dispone de un conjunto de micro-drones Crazyflie, adquiridos desde hace algunos años con fines educativos. Además, cuenta con un laboratorio equipado con un ecosistema adaptado para estudios de ingeniería mecatrónica y electrónica, donde se realizan prácticas de laboratorio que involucran agentes autónomos como robots móviles, brazos robóticos y drones.

Este proyecto explora una alternativa de control para los drones Crazyflie 2.1 mediante la implementación de un sistema de posicionamiento basado en odometría visual. Para ello, se presetan la integración, las pruebas y la validación del *hardware* requerido; se explican herramientas de *software* para facilitar la maniobrabilidad del dron y, finalmente, se proporciona un manual de usuario y guías de laboratorio orientadas a los cursos de Robótica y Sistemas de Control.

CAPÍTULO 2

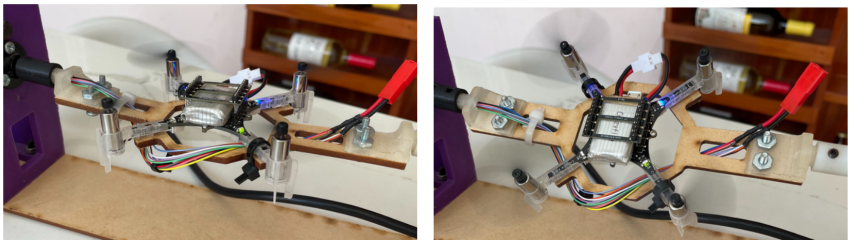
Antecedentes

Los drones Crazyflie han demostrado ser herramientas altamente efectivas en investigación de sistemas de control, ya sea como agentes individuales o dentro de un enjambre. Algunos ejemplos de investigación incluyen estudios sobre algoritmos de evasión de obstáculos, vuelo en enjambre y sistemas de navegación autónoma. Esta eficacia y versatilidad de los drones fue la razón principal por la cual la Universidad del Valle de Guatemala adquirió hace algunos años un conjunto de drones Crazyflie y, desde entonces, se han desarrollado diversos proyectos de investigación con ellos.

2.1. Investigación con drones Crazyflie en la Universidad del Valle de Guatemala

Sanabria en [1] desarrolló la fase inicial en la línea de investigación con drones Crazyflie. Su trabajo consistió en la implementación de una plataforma de pruebas 1 para cuadricópteros Crazyflie 2.0 sobre la cual se pueden verificar algoritmos de control de actitud para un grado de libertad. Durante su investigación, implementó un conjunto de herramientas de *software* necesarias para comunicar al dron con la computadora a través de Python y por medio del dispositivo Crazyradio. Asimismo, elaboró una interfaz gráfica capaz de recuperar y procesar ángulos de inclinación, manipular la orientación y modificar los parámetros del controlador del dron.

Figura 1. Plataforma de pruebas de la investigación de Sanabria [1].

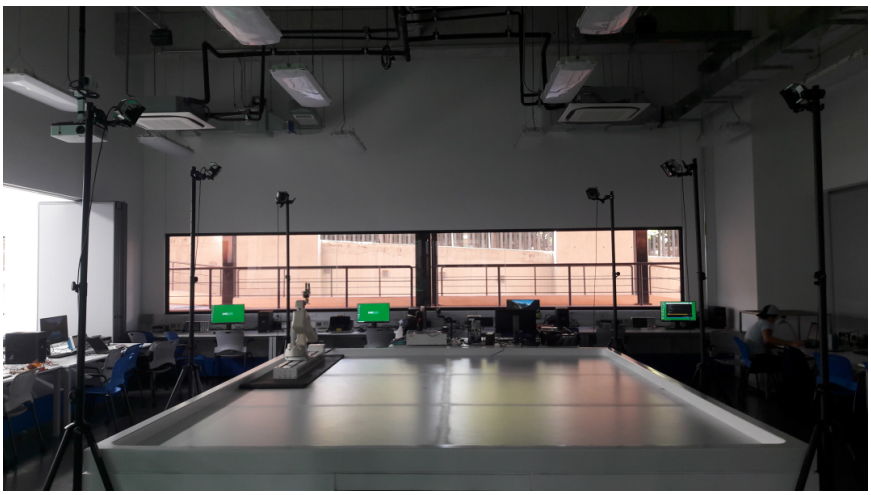


Como resultado de la investigación se crearon guías de laboratorio para los cursos de sistemas de control 1 y 2, con la limitante de que el dron únicamente podría utilizarse junto a la plataforma de pruebas.

2.2. Ecosistema Robotat

El ecosistema Robotat 2 es un entrono tecnológico de investigación y experimentación ubicado en la Universidad del Valle de Guatemala [2]. Este ecosistema consiste de una plataforma sólida de aproximadamente 400×500 cm, un sistema de captura de movimiento compuesto por seis cámaras OptiTrack y una red local Wi-Fi estblecida mediante el protocolo MQTT. La infraestructura permite la conexión y control de múltiples agentes simultáneamente, con una capacidad máxima de 11 agentes operando a una frecuencia de recepción y decodificación de datos superior a 10 Hz.

Figura 2. Ecosistema de pruebas Robotat [2].



2.3. Incorporación de drones Crazyflie al ecosistema Robotat

Denny Otzoy [3] y José Gordillo [4] se centraron en desarrollar la infraestructura y herramientas necesarias para utilizar el cuadricóptero dentro del ecosistema Robotat. Otzoy se enfocó en el desarrollo de herramientas para la integración con el sistema de captura de movimiento. Empleó una máquina física para la transmisión de datos, desarrolló la representación del dron como cuerpo rígido dentro del ecosistema y codificó las trayectorias en el formato adecuado. Por otro lado, Gordillo implementó un paquete de herramientas de *software* para la ejecución de trayectorias de enjambre. Evaluó dos alternativas para el sistema de control y con base en un listado de ventajas y desventajas, descartó la opción de la librería Crazyswarm y en su lugar optó por implementar un sistema basado en una antena de comunicación WiFi. En ambos casos, a pesar de los esfuerzos, no se logró el control adecuado o seguimiento de trayectorias debido a limitantes en la metodología empleada para el sistema de control del dron.

A raíz de las limitaciones encontradas, Julio Avila [5] y Brandon Garrido [6] buscaron integrar la librería de control Crazyswarm 2 al ecosistema Robotat. El trabajo de Avila estuvo centrado principalmente en el desarrollo de un servidor para comunicación entre los drones, el sistema de captura de movimiento y Matlab, con el fin de enviar comandos para realizar trayectorias de drones individuales o de enjambre. Por su parte, Garrido se enfocó en el desarrollo de infraestructura para la experimentación y control de múltiples drones desde el sistema operativo ROS2 y dentro del ecosistema Robotat. Los resultados obtenidos en ambas investigaciones demostraron estabilidad y precisión en el control de los drones.

Figura 3. Prueba de vuelo con tres drones utilizando el *software* Crazyswarm 2 [6].

Aunque el manejo de los drones mediante Crazyswarm demostró ser efectivo, requiere conocimientos de los sistemas operativos Linux y ROS2. Esta tarea vuelve complicado el proceso debido a la complejidad de dichas herramientas y la escasa documentación disponible de la librería Crazyswarm 2. Adicionalmente, la transición de Crazyflie a ROS2 es reciente, al igual que la librería Crazyswarm 2, lo que dificulta su implementación en laboratorios.

Por otro lado, el uso individual de los drones Crazyflie plantea otro desafío, ya que, sin un medio para conocer su posición, los drones son vulnerables a colisiones debido a la fragilidad de su manipulación. Para abordar dicho desafío, una solución viable es complementar el funcionamiento de los drones con alguna de las placas de expansión de Bitcraze. Una de ellas es el Flow Deck, que ha demostrado ser altamente eficiente en proporcionar estabilidad al dron y con el cual se han logrado proyectos de investigación como vuelo autónomo y seguimiento de trayectorias.

2.4. Vuelo autónomo del dron Crazyflie 2.1 empleando la placa de posicionamiento Flow Deck

En la Universidad Uppsala [7] se realizó una investigación utilizando drones Crazyflie 2.1 con el objetivo de incorporar vuelo autónomo a través de trayectorias con obstáculos. Para ello se exploraron dos alternativas para el sistema de navegación del dron: un sistema de posicionamiento local mediante la herramienta LPS (*Loco Positioning System*) y un sistema de navegación óptico mediante el dispositivo Flow Deck. Para la detección de obstáculos en las trayectorias se utilizó el sensor de detección Multiranger.

Durante los experimentos realizados se observó que el sistema de navegación óptico superó al LPS en términos de estabilidad de vuelo y capacidad para completar las trayectorias. No se logró completar pruebas realizadas con el LPS como sistema de posicionamiento, siendo la razón principal el vuelo inestable causado por perturbaciones inusuales en el entorno de experimentación. Por otro lado, se completaron satisfactoriamente las pruebas al emplear el sistema de navegación óptico con el Flow Deck. Como recomendación, se sugirió realizar pruebas adicionales para mejorar la precisión del vuelo y considerar la posibilidad de utilizar múltiples sistemas de navegación en conjunto para obtener resultados más robustos.

CAPÍTULO 3

Justificación

Los avances en investigación de ingeniería de control han sido significativamente influenciados por el uso de drones, que se han convertido en herramientas indispensables debido a su alta capacidad para implementar algoritmos de control. Su versatilidad y facilidad de adaptación hacen de los drones un recurso valioso para la enseñanza de cursos relacionados con ingeniería de control y robótica.

Como se ha mencionado, en la Universidad del Valle de Guatemala se han desarrollado distintos proyectos de investigación que han aprovechado las capacidades de los drones Crazyflie. Estos proyectos han explorado distintas aplicaciones, desde su uso en plataformas de pruebas hasta la formación de enjambres de drones en entornos controlados.

Recientemente, se ha desarrollado un conjunto de herramientas que, si bien demostraron ser efectivas, resultan complicadas de utilizar debido a su alta curva de aprendizaje y escasa documentación. En particular, la herramienta Crazyswarm 2 en ROS2 ha demostrado ser efectiva pero difícil de manejar debido a los requerimientos de conocimientos avanzados en Linux y ROS2, así como la falta de documentación disponible. Por ello, surge la necesidad de simplificar las herramientas de control para facilitar el uso práctico y seguro de los drones.

El propósito de este trabajo de graduación se centra en la oportunidad de ampliar el uso de los drones Crazyflie en la Universidad del Valle de Guatemala, permitiendo su uso independiente y seguro mediante la integración de la placa de expansión Flow Deck como nuevo sistema de posicionamiento. De esta manera, los drones podrán ser utilizados de forma sencilla en prácticas de laboratorio en cursos de ingeniería electrónica y mecatrónica, tales como Sistemas de Control 1 y Robótica 1.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar herramientas de *software* para controlar de manera individual y segura el cuadricóptero Crazyflie 2.1 utilizando la placa de expansión de posicionamiento con odometría visual Flow Deck.

4.2. Objetivos específicos

- Realizar la integración, pruebas y validación de la placa de expansión Flow Deck en el dron Crazyflie 2.1 sobre la mesa de pruebas del ecosistema Robotat.
- Desarrollar herramientas de *software* que permitan simular, controlar y monitorear a los drones Crazyflie 2.1 de forma independiente y segura.
- Desarrollar un conjunto de experimentos que permitan estudiar temas de control de orientación y posición del dron Crazyflie 2.1 en la mesa de trabajo del ecosistema Robotat y que puedan emplearse dentro de los cursos de Sistemas de Control 1 y Robótica 1.

CAPÍTULO 5

Alcance

Este proyecto involucró la integración y validación de la placa de expansión Flow Deck en los drones Crazyflie 2.1, junto con el desarrollo de herramientas de *software* diseñadas para facilitar el control individual y seguro del dron en entornos educativos. Además, se elaboraron guías de laboratorio y un manual de usuario para apoyar su implementación en cursos de Sistemas de Control y Robótica.

Para el desarrollo de las herramientas de *software*, se priorizó la practicidad de uso sobre la eficiencia computacional. El objetivo fue garantizar que las herramientas fueran fáciles de implementar y utilizar en un entorno académico, incluso si esto implicaba algunas concesiones en cuanto al rendimiento óptimo o la implementación de características avanzadas.

En el proceso de fusión de sensores, se enfrentó la limitante de que el ecosistema Robotat actualmente opera mediante un esquema de solicitud y recepción de datos, en lugar de un flujo continuo de datos (*streaming*). Esto impidió implementar una fusión de sensores basada en eventos continuos. Además, para la fusión lograda, fue necesario que el dron se inicializara con sus ejes alineados de forma aproximada con los del sistema de captura de movimiento del ecosistema Robotat, debido a la naturaleza de las funciones desarrolladas.

Por otro lado, se presentaron dificultades inherentes al *hardware* de los drones Crazyflie, que resultaron ser especialmente frágiles durante los experimentos. Uno de los problemas más frecuentes fue el daño constante de las hélices, lo que limitó la continuidad de las pruebas y aumentó la necesidad de mantenimiento.

CAPÍTULO 6

Marco teórico

Para cumplir los objetivos establecidos en este proyecto, es necesario desarrollar un marco teórico que aborde diversos temas clave relacionados con el dron Crazyflie. Entre estos se incluyen el análisis de sus componentes principales, el sistema de coordenadas, la comunicación inalámbrica y la funcionalidad y operación de la placa de expansión Flow Deck. También, se describen los principios de odometría visual que sustentan el sistema de posicionamiento. Asimismo, se explora la estructura y funcionalidad de la biblioteca cflib en Python, utilizada para implementar las herramientas de *software* desarrolladas.

6.1. Crazyflie 2.1

Los drones Crazyflie son plataformas de desarrollo aéreo de código abierto desarrollados por Bitcraze. Están diseñados principalmente para investigación, desarrollo y educación en ingeniería de control y robótica. Como se observa en la Figura 4, tienen un tamaño reducido, pero están equipados con una variedad de sensores que lo vuelven ideal para explorar algoritmos de control y otras aplicaciones.

Figura 4. Dron Crazyflie 2.1 [8].

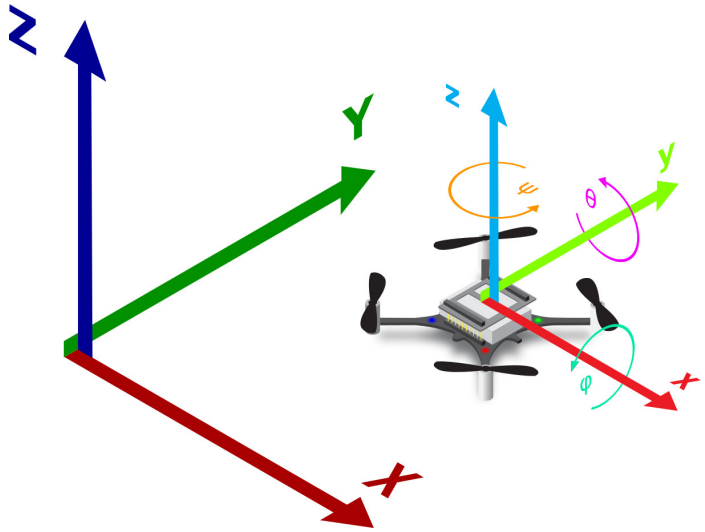
El Crazyflie 2.1 es un mini dron que pesa aproximadamente 27 gramos y tiene dimensiones generales de 92×92×29 mm. Su control se realiza mediante *Bluetooth* o radiofrecuencia, lo que le permite ser controlado desde dispositivos móviles, así como desde sistemas operativos Windows, Mac OSX y Linux utilizando Crazyradio o Crazyradio PA. En cuanto a sus características eléctricas, el Crazyflie 2.1 dispone de una batería de litio-polímero (Li-Po) con modos desde 100 mA hasta 980 mA, que alimenta motores, microcontroladores y demás componentes. Utiliza un microcontrolador STM32F405 para el control de vuelo y un microcontrolador nRF51 para la comunicación inalámbrica. Está equipado con un acelerómetro/giroscopio de 3 ejes BMI088 y un sensor de presión de alta precisión BMP388, pero también permite la integración de otras placas de expansión para ampliar sus capacidades, con la restricción de que soporta una carga adicional de hasta 15 gramos que puede afectar el tiempo de vuelo debido al aumento de demanda de energía [8].

6.2. Sistema de coordenadas del dron Crazyflie

Los drones Crazyflie, al igual que la mayoría de los drones, utilizan la convención de sistema de coordenadas tridimensional ENU (*East North Up*) para determinar su posición y orientación en el espacio. Como se observa en la Figura 5, el dron tiene tres ejes principales de referencia: el eje X es horizontal y apunta hacia adelante, el eje Y es horizontal y apunta hacia la derecha y el eje Z es vertical y apunta hacia arriba.

Por otro lado, la orientación del dron se describe en términos de ángulos de inclinación respecto a los ejes de referencia. El ángulo de balanceo (*roll*) se refiere a la inclinación del dron hacia los lados en torno al eje X, el ángulo de cabeceo (*pitch*) se refiere a la inclinación hacia adelante o hacia atrás en torno al eje Y, y el ángulo de guiñada (*yaw*) se refiere a la rotación del dron en torno al eje Z. Según la documentación oficial de Bitcraze, estos ángulos siguen las siguientes reglas de rotación: *roll* y *yaw* rotan en sentido horario alrededor del eje al mirar desde el origen, mientras que *pitch* rota en sentido antihorario alrededor del eje al mirar desde el origen [9].

Figura 5. Sistema de coordenadas del Crazyflie 2.1 [9].



6.3. Comunicación inalámbrica del dron Crazyflie

Los drones Crazyflie establecen una comunicación inalámbrica por medio del dispositivo Crazyradio, mostrado en la Figura 6. Esta comunicación se basa en un enlace de radiofrecuencia bidireccional que permite enviar comandos de control al dron y recibir datos en tiempo real sobre su posición, orientación y otros parámetros relevantes.

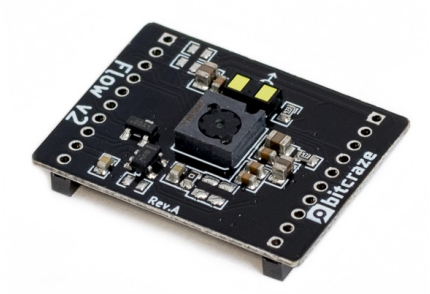
Figura 6. Dispositivo de comunicación inalámbrica Crazyradio [10].

El Crazyradio es un transceptor de radio USB de código abierto, baja latencia y largo alcance. Funciona en la banda de 2.4 GHz con un rango de transmisión de hasta 1 km (en condiciones ideales) gracias a su amplificador de radio de 20 dBm. Está basado en el microcontrolador nRF24LU1 de Nordic Semiconductor y se comunica por medio del protocolo "*Enhanced ShockBurst*" compatible con los microcontroladores nRF24L01p, nRF51 y nRF52. En una capa de nivel más alto, utiliza el protocolo de paquetes CRTP para comunicarse con el Crazyflie y poder actualizar el *firmware*, enviar comandos y recibir información de los drones. Dependiendo del sistema operativo, es necesario instalar controladores o realizar configuraciones específicas [10].

6.4. Placa de expansión Flow Deck

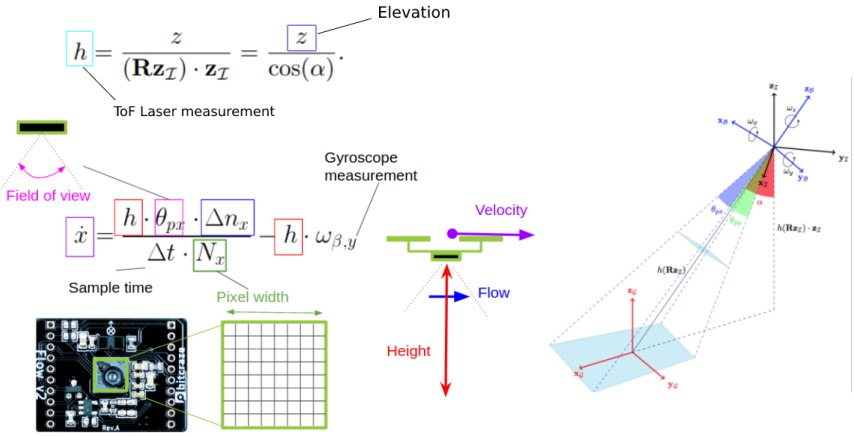
El Flow Deck es una placa de expansión diseñada específicamente para mejorar las capacidades de percepción de los drones Crazyflie. Esta placa funciona como un sistema de posicionamiento para el dron, otorgándole la capacidad de comprender su posición y velocidad en un entorno tridimensional.

Figura 7. Placa de expansión Flow Deck v2 [11].



La placa Flow Deck (Figura 7), tiene un peso aproximado de 1.6 gramos y dimensiones generales de 21×28×4 mm. Para integrarlo, se requiere actualizar el *firmware* del Crazyflie a su última versión y montarlo en la parte inferior del dron. Utiliza el sensor de medición de distancias VL53L1x para determinar su altura junto al sensor de flujo óptico PMW3901 que mide movimientos horizontales sobre la superficie mediante odometría visual [11]. Como se observa en la Figura 8, la combincación de información obtenida de ambos sensores permite al dron estimar su posición en un entorno tridimensional.

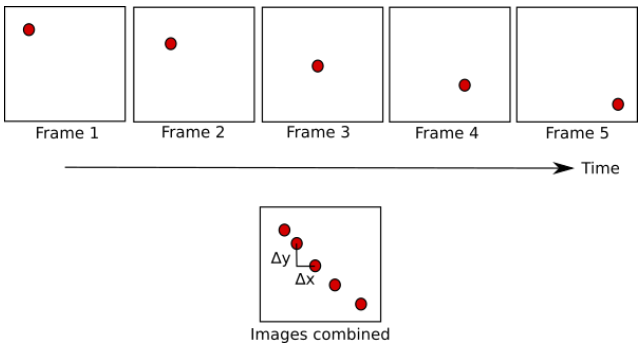
Figura 8. Funcionamiento de la placa Flow Deck y ecuación de estimación de posición [12].



6.5. Odometría visual

Es un método que permite estimar el movimiento de un agente robótico utilizando infomación de imágenes capturadas por cámaras. Esta técnica de estimación de posición se basa en el análisis de una secuencia de imágenes para identificar y rastrear rasgos visuales distintivos. Tal como se muestra en la Figura 9, se realiza un seguimiento de los rasgos de fotograma a fotograma; el resultado es la distancia que la entidad se ha movido desde el fotograma anterior [13].

Figura 9. Diagrama de flujo óptico para estimación de posición con odometría visual [14].



Este método resulta útil para entornos donde otras formas de localización no son viables o precisas, como en interiores o áreas con estructuras complejas.

Como se mencionó, el sensor PMW3901 en la placa Flow Deck utiliza odometría visual para estimar las distancias desplazadas por rasgos distintivos en las imágenes capturadas. Esta información, junto con las mediciones de altura del sensor de distancia VL53L1x, son combinadas en la ecuación de la Figura 8 para estimar el movimiento del dron relativo a su punto de partida [12].

6.6. Sensor de flujo óptico PMW3901

Es un sensor de flujo óptico desarrollado por Pixart Imaging, ampliamente utilizado en aplicaciones de robótica para estimar movimiento. Resulta de particular interés debido a que forma parte de la placa de expansión Flow Deck de Bitcraze y es el responsable de darle sentido de posición bidimensional (en los ejes X-Y) a los drones Crazyflie. Dentro de sus detalles técnicos. Dispone un microcontrolador de bajo consumo que determina internamente el flujo óptico con algoritmos de odometría visual, proporcionando posición con base en diferencias en pixeles entre fotogramas. Opera con un voltaje que varía entre 1.8 y 2.1 V. Se comunica por medio de una interfaz SPI de 4 hilos que opera a 2 MHz, transmitiendo datos de movimiento almacenados en registros de 16 bits. Tiene un rango de operación desde 80 mm hasta el infinito y cuenta con una tasa de fotogramas de 121 FPS, detectando movimientos de hasta 7.4 radianes por segundo. Está encapsulado en un paquete *chip-on-board* de 28 pines, lo que permite su integración en distintas aplicaciones [15].

6.7. Sensor de medición de distancias VL53L1x

Es un sensor de medición de distancias desarrollado por STM electronics. La placa de expansión Flow Deck tiene integrado este sensor para proporcionar sentido de altura de vuelo (eje Z) a los drones Crazyflie. En cuanto a sus especificaciones técnicas, el VL53L1x está basado en la tecnología de tiempo de vuelo (*Time-of-Flight*, ToF) que permite medir el tiempo que tarde un pulso de luz en reflejarse desde un objeto y volver al sensor, proporcionando una medida precisa de la distancia del objeto. El sensor funciona emitiendo un láser invisible de 940 nm y utiliza una matriz de recepción SPAD (diodo de avalancha de fotón único) para detectar la luz reflejada, ofreciendo un campo de visión típico de 27 grados. Incluye un microcontrolador de bajo consumo que permite una medición de distancia de hasta 4 metros con una frecuencia de hasta 50 Hz y se comunica por medio de una interfaz I2C que soporta velocidades de hasta 400 kHz [16].

6.8. Librería cflib en Python para Crazyflie

La librería cflib es una herramienta en Python desarrollada por Bitcraze para facilitar la comunicación y control del dron Crazyflie. Esta librería proporciona una API robusta que permite implementar distintas operaciones en el dron, que van desde vuelo autónomo hasta la gestión y recopilación de lecturas de sensores. Además, cflib simplificia la interacción con el *hardware* a través de comandos y funciones de alto nivel. La librería es asíncrona y se basa

en devoluciones de llamada para eventos. Sin embargo, existen contenedores síncronos para clases seleccionadas que crean una API síncrona encapsulando las clases asíncronas [17].

6.8.1. Estructura de la librería

Identificador uniforme de recursos (URI)

Todos los enlaces de comunicación se identifican mediante una dirección URI estructurada con el formato [17]: **InterfaceType://InterfaceId/InterfaceChannel/InterfaceSpeed**

Los tipos de interfaz actualmente disponibles con la estructuras de formato adecuada son los siguientes:

- **radio://0/10/2M:** interfaz de radio, dongle USB número 0, canal de radio 10 y radio velocidad 2 Mbit/s
- **debug://0/1:** interfaz de depuración, id 0, canal 1
- **usb://0:** cable USB a puerto microusb, id 0
- **serial://ttyAMA0:** puerto serie, id ttyAMA0
- **tcp://tcp-1.local:300:** conexión de red TCP, Nombre: tcp-1.local, puerto 300

Variables y parámetros

En la librería Crazyflie, tanto las variables de registro como los parámetros son fundamentales para monitorear y controlar el dron en tiempo real. Ambas se agrupan y acceden de manera estructurada siguiendo la misma convención de nombres: grupo.nombre [17].

- **Variables:** las variables de registro se utilizan para supervisar continuamente datos del dron. Se configuran para que el *firmware* envíe estos datos al host a intervalos regulares.
- **Parámetros:** los parámetros permiten leer y escribir configuraciones en el *firmware* durante la ejecución. Son útiles para modificar ajustes que no cambian constantemente, como los valores de controladores PID o el resultado de las autocomprobaciones.

6.8.2. Módulos

La librería está organizada en distintos módulos que permiten al usuario acceder a funcionalidades específicas del dron. Estos módulos completan la estructura de la librería cflib, proporcionando una API robusta para controlar, gestionar y optimizar el uso del dron Crazyflie en distintos entornos.

Módulo Bootloader

El módulo Bootloader es responsable de gestionar el proceso de actualización del *firmware* del dron Crazyflie. Este módulo proporciona las herramientas necesarias para realizar una actualización segura, tanto a través de interfaces USB como de radio. Utiliza el protocolo *bootloading* del Crazyflie para coordinar la transferencia de datos, gestionar las versiones y garantizar la correcta escritura en la memoria flash del dron [18].

Módulo CPX

El módulo CPX proporciona un conjunto de herramientas para gestionar la comunicación entre el dron Crazyflie y otras plataformas mediantes distintos mecanismos de transferencia, como *sockets* TCP y UART. Permite el intercambio de datos entre el dron y dispositivos externos [19].

Módulo Crazyflie

El módulo Crazyflie es el núcleo de la librería y posee una amplia colección de herramientas diseñadas para controlar al dron Crazyflie. Este módulo ofrece una interfaz para interactuar con el dron, permitiendo el control de vuelo, la gestión de parámetros, la adquisición de datos y otras funciones. Está compuesto por la clase principal Crazyflie y otras clases auxiliares que gestionan diferentes aspectos del control del dron, como la conexión y la configuración [20].

Módulo CRTP

El módulo CRTP (*Crazy Real-Time Protocol*) conforma al protocolo de comunicación entre el controlador y el dron Crazyflie. Este protocolo es el responsable de estructurar los paquetes de datos en la comunicación con el dron, permitiendo una interacción a nivel bajo entre el *hardware* y el *software*. Además, dispone de las herramientas necesarias para configurar y gestionar las conexiones, asegurando una comunicación segura y eficiente [21].

Módulo Drivers

El módulo de Drivers contiene las implementaciones necesarias para la interacción con el *hardware* del dron Crazyflie. Estas herramientas permiten gestionar las distintas interfaces de comunicación y control entre el *software* y el *hardware*. Dentro del módulo Drivers se encuentran componentes que habilitan la conexión física mediante interfaces como USB y radio, garantizando una transferencia de datos segura y eficiente [22].

Módulo Localization

El módulo Localization permite obtener y gestionar datos relacionados con la posición y localización del dron Crazyflie. A través de este módulo, el usuario puede utilizar sistemas de localización externa, como cámaras o estaciones de anclaje, para rastrear el movimiento del dron con precisión en tiempo real [23].

Módulo Positioning

El módulo Positioning se encarga de gestionar los sistemas de posicionamiento de Crazyflie. Incluye algoritmos y configuraciones para obtener la posición precisa del dron a través de sensores de posicionamiento [24].

Módulo Utils

El módulo Utils contiene utilidades generales y herramientas de apoyo que facilitan el desarrollo de aplicaciones con el dron Crazyflie. Este módulo incluye funciones para el manejo de errores, herramientas de depuración y otras funciones auxiliares necesarias para mejorar la eficiencia y fiabilidad del desarrollo [25].

Los módulos anteriormente presentados conforman en totalidad a la librería cflib en Python y cada uno otroga funcionalidades específicas a la libería. Sin embargo, por la finalidad de este trabajo de graduación, algunos módulos presentan mayor relevancia que otros. En especial, los módulos Crazyflie y CRTP son particularmente importantes.

6.9. Módulo Crazyflie

El módulo Crazyflie es relevante porque centraliza todas las operaciones esenciales para interacturar y controlar el dron. Coordina la interacción con el resto de módulos para formar una interfaz simple capaz de ejecutar funcionalidades específcias. Gracias a este módulo, es posible realizar tanto operaciones báscias como avanzadas con el dron Crazyflie [20].

Clase principal crazyflie
Es la clase que centraliza las operaciones necesarias para controlar el dron Crazyflie. Actúa como la interfaz directa entre el usuario y el dron, gestionando la conexión, el envío de comandos, el registro de datos y la configuración de parámetros. Además, la clase Crazyflie instancia y coordina varias clases auxiliares que facilitan tareas específicas.

Clases secundarias

- **appchannel:** facilita la comunicación personalizada entre el dron y aplicaciones externas a través de un canal de aplicaciones.

- **commander:** controla el movimiento y los comandos de vuelo del dron.
- **console:** proporciona una interfaz para interactuar con la consola del dron, donde se pueden visualizar mensajes del *firmware* del dron en tiempo real.
- **extpos:** maneja la información de posicionamiento externo del dron, generalmente a través de sistemas de localización como cámaras externas.
- **high_level_commander:** permite controlar el dron de manera autónoma, con comandos de alto nivel, como trayectorias predefinidas, despegue y aterrizaje.
- **localization:** maneja los aspectos relacionados con la localización del dron Crazyflie, ayudando a rastrear su posición en el espacio.
- **log:** gestiona el registro de datos del dron, como el estado de los sensores.
- **param:** gestiona los parámetros configurables del dron, como la calibración de sensores y ajustes de vuelo.
- **plataformservice:** gestiona la interacción entre el *hardware* y el *software* del dron, facilitando la integración de diferentes plataformas.
- **swarm:** maneja la coordinación de múltiples drones Crazyflie, permitiendo controlarlos simultáneamente como un enjambre.
- **syncCrazyflie:** proporciona una interfaz síncrona para la clase principal Crazyflie, facilitando la escritura de scripts secuenciales.
- **syncLogger:** permite sincronizar el registro de datos de vuelo para garantizar la consistencia y precisión de los datos recolectados durante el vuelo.
- **toc:** almacena un índice de las variables y parámetros disponibles en el dron, que pueden ser registrados o modificados.
- **toccache:** almacena en caché las Tablas de Contenidos (TOC) del dron, optimizando la recuperación de datos y parámetros para evitar su descarga en cada nueva conexión.

6.10. Módulo CRTP

CRTP es el protocolo de empaquetado de datos utilizado para comunicarse con los drones Crazyflie. Este protocolo estructura los paquetes de información para dirigirlos a funcionalidades específicas del dron como el registro de datos, control de movimiento y parámetros de configuración.

La comunicación con Crazyflie se implementa como una pila de capas independientes. En la base se encuentra el medio físico (radio o USB), seguido por el enlace que garantiza la transmisión segura y ordenada de paquetes. Encima de esto está CRTP que maneja información de puerto y canal para dirigir los paquetes de datos a los distintos subsistemas del Crazyflie. Por último, los subsistemas implementan las funcionalidades del dron, controladas a través de CRTP.

Cada paquete CRTP incluye un número de puerto, un número de canal y una carga útil. Los puertos oscilan entre 0 y 15 y los canales entre 0 y 3, con la carga útil limitada

a 31 bytes. Estos metadatos permiten dirigir los paquetes a los subsistemas correctos y las funcionalidades específicas dentro del Crazyflie [26].

El módulo dentro de la libería se ecuentra estructurado en submódulos con funcionalidades espcíficas:

Submódulos

- **cflib.crtp.crtpstack:** administra el flujo de paquetes entre el dron y el controlador.
- **cflib.crtp.exceptions:** maneja los errores de comunicación CRTP, como pérdida de paquetes o tiempos de espera.
- **cflib.crtp.pcap:** captura paquetes CRTP para depuración y análisis.
- **cflib.crtp.prrtdriver:** implementa el PRRT para garantizar la entrega fiable de datos mediante retransmisiones.
- **cflib.crtp.radiodriver:** gestiona la comunicación inalámbrica por radio con el dron.
- **cflib.crtp.serialdriver:** maneja la comunicación por conexiones serie.
- **cflib.crtp.tcpdriver:** gestiona las conexiones TCP/IP.
- **cflib.crtp.udpdriver:** gestiona las conexiones UDP.
- **cflib.crtp.usbdriver:** administra la comunicación por puerto USB.

CAPÍTULO 7

Integración y validación de la placa de expansión Flow Deck

En este capítulo se presenta una explicación detallada del procedimiento realizado para lograr la integración exitosa de la placa de expansión Flow Deck en el dron Crazyflie 2.1. Para realizar esta tarea, se utilizó un ordenador con sistema operativo Windows 11 y *software* de código abierto proporcionado por el grupo Bitcraze. También, se describen los experimentos de vuelo reaizados con y sin la placa de expansión Flow Deck y se proporcionan observaciones importantes, relacionadas con el entorno de experimentación, para que la placa funcione correctamente.

7.1. Validación de funcionamiento de Crazyflie

Previo a la instalación de la placa de expansión, fue necesario realizar una verificación del ensamble y ajustes de configuración con el fin de garantizar el funcionamiento adecuado del dron Crazyflie 2.1 para los fines del proyecto.

7.1.1. Ensamble de Crazyflie

Respecto al ensamble del dron, se consultó el tutorial de armado y pruebas de vuelo disponible en la página oficial de Bitcraze [27]. Se examinó que cada parte del dron estuviera correctamente instalada según la información en la página. El resultado obtenido se observa en la Figura 10 donde se muestra al dron correctamente ensamblado.

Luego de examinar y validar el ensamble, se ejecutó la secuencia de auto-prueba presionando el botón de encendido. Esta secuencia preparó al dron para volar, verificando que el *hardware* se encontrara en buen estado y calibró los sensores con valores base. Para ejecutar la prueba, se colocó al dron en una superifice nivelada y se mantuvo absolutamente quieto.

Figura 10. Ensamble de dron Crazyflie 2.1.

La ejecución de la autoprueba indicó que el estado del dron se encontraba a la perfección y listo para volar. Esto se determinó con la siguiente lista de resultados que se interpretan con la modalidad de encendido de los LED presentes en el dron:

- **Encendido y listo para volar:** LED azules permanecen encendidos y el LED delantero derecho parpadea en rojo dos veces por segundo.
- **Encendido pero con sensores descalibrados:** LED acules permanecen encendidos y el LED delantero derecho parapadea en rojo cada dos segundos.
- **Crazyradio conectada:** LED delantero izquierdo parpadea en rojo y/o verde.
- **Batería baja:** LED delantero derecho permanece completamente encendido en rojo.
- **Carga:** LED azul izquierdo parpadea mientras que el LED azul derecho está encendido.
- **Modo cargador de arranque:** LED azules parpadean una vez por segundo.
- **Error de autoprueba:** LED delantero derecho parpadea repetidamente en rojo con una pausa larga entre parpadeos.

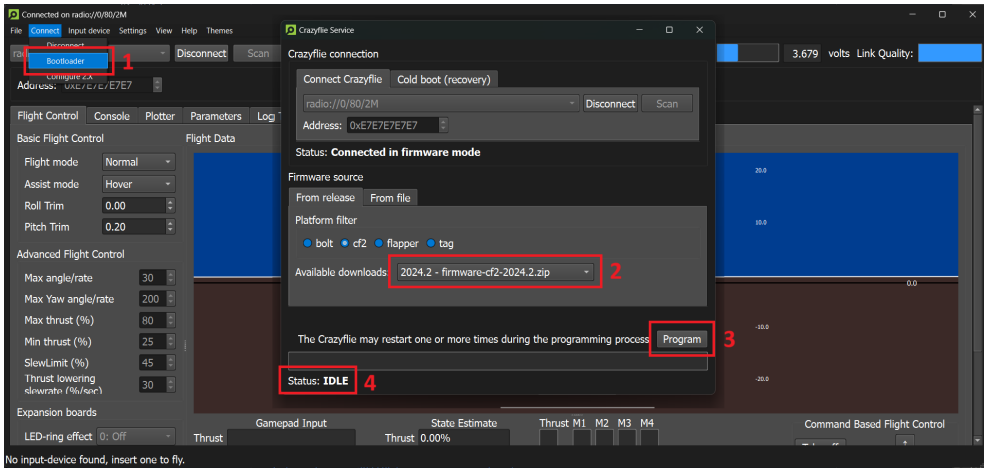
Figura 11. Interfaz de control de vuelo del *software* Crazyclient.

7.1.2. Software Crazyclient y actualización de firmware de Crazyflie

Tras verificar que el dron iniciara correctamente, se instaló el *software* de código abierto Crazyclient del grupo Bitcraze según el manual de instalación provisto en su página oficial [28], versión para Windows 10/11. Este programa dispone de diversas herramientas para el control, la calibración, la visualización de datos en tiempo real y la configuración de parámetros del dron Crazyflie. Además, como se observa en la Figura 11, dispone de una interfaz gráfica de usuario que facilita la interacción.

Una vez instalado el programa Crazyclient, lo primero que se realizó fue la actualización de *firmware* siguiendo las instrucciones de la página oficial [29]. Este paso es fundamental para que el dron Crazyflie reconozca a la placa de expansión Flow Deck. Es necesario que tenga instalada la versión más reciente disponible para garantizar un rendimiento eficiente del *hardware*.

Figura 12. Actualización de *firmware* con el programa Crazyclient.



Antes de la actualización de *firmware*, fue necesario instalar los controladores USB del dispositivo Crazyradio utilizando la guía de instalación del fabricante [30]. Posteriormente, se empleó el Crazyradio para establecer la conexión con el Crazyflie. Una vez conectado, se siguieron los pasos mostrados en la Figura 12. Cabe mencionar que el último paso tan solo consistió en esperar a que terminara el proceso y que el estado del dron fuera IDLE.

7.1.3. Configuración de parámetros

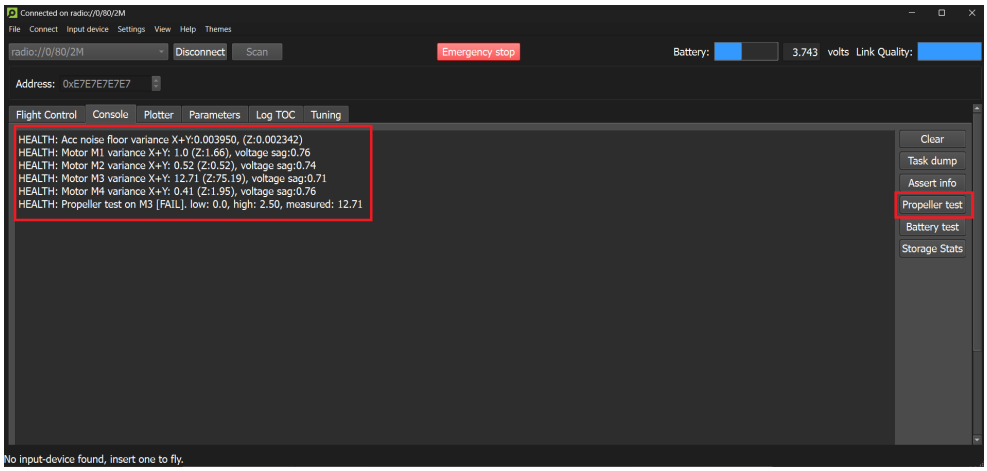
Como siguiente paso, se habilitó la vista de parámetros del cfclient para ajustar los parámetros del dron con el fin de asegurar el comportamiento esperado.

- **stabilizer.controller:** se colocó con un valor de 1 para utilizar el controlador PID.
- **stabilizer.estimator:** se colocó un valor de 2 para utilizar el estimador extendido de Kalman.
- **commander.enHighLevel:** se colocó un valor de 1 para permitir el envío de comandos de forma remota al dron.

7.1.4. Prueba de hélices y motores

Manteniendo la conexión del dron con el cliente, se utilizó la vista de consola para realizar una prueba de hélices y motores. Para ello, se colocó al Crazyflie sobre una superifcie plana y dura y luego se seleccionó la opción "*Propeller test*" para realizar una prueba de vibración en motores y hélices (Figura 13). El dron accionó levemente los motores, de forma secuencial, para determinar el estado de estos y sus hélices. Para los motores que presentaron vibraciones por encima del umbral permitido se emitió un pitido y se imprimió la advertencia en consola.

Figura 13. Vista de consola y prueba de vibración en hélices y motores.



Las vibraciones excesivas en los motores y hélices perjudican significativamente la calidad de vuelo del dron, por tal motivo fue necesario realizar ajustes para reducirlas o eliminarlas. Hay tres posibles soluciones al problema y deben intentarse en orden:

1. **Balanceo de hélices:** la primera solución para abordar el problema es balancear las hélices, siguiendo el tutorial disponible en la página de Bitcraze [31].
2. **Reemplazo de hélices:** si el balanceo de las hélices no resuelve el problema, se recomienda reemplazarlas con hélices nuevas de repuesto.
3. **Reemplazo de motores:** si las vibraciones persisten después de cambiar las hélices, se debe considerar el reemplazo de los motores.

7.1.5. Prueba de vuelo simple

Luego de examinar los aspectos anteriores, se procedió a realizar una prueba de vuelo utilizando la GUI del cfclient y un mando de PS4. El objetivo del experimento fue elevar un poco el dron aplicando una cantidad considerable de *thrust* para elevarlo y poder observar su comportamiento en ausencia de un sistema de posicionemiento (lazo abierto).

El desepgue del dron fue exitoso sin embargo, como se esperaba, no logró mantener su posición durante la elevación presentando una cantidad considerable de deriva y descompensación de vuelo.

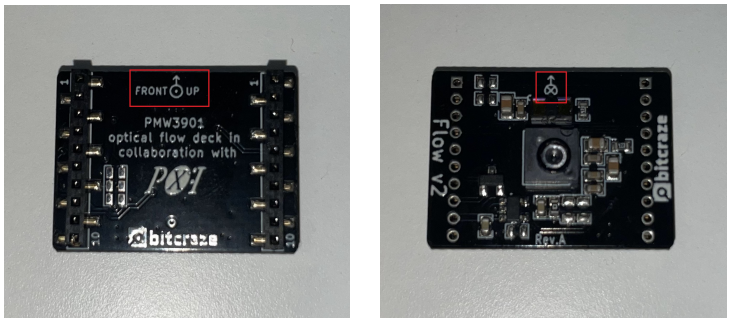
7.2. Instalación de la placa de expansión Flow Deck

Habiendo verificado el funcionamiento propio del dron Crazyflie y familiarizado con el *software* de control cfclient, se continuó realizando la instalación de la placa de expansión Flow Deck y su validación mediante pruebas de posicionamiento.

7.2.1. Montaje y reconocimiento de la placa

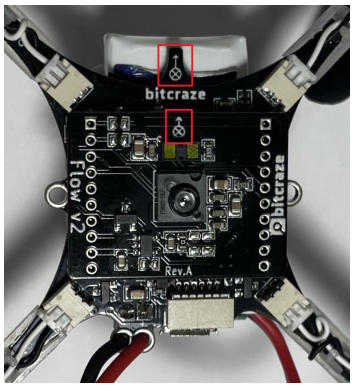
Para proceder con la instalación de la placa de expansión Flow Deck, se siguieron cuidadosamente las instrucciones provistas por Bitcraze en su página oficial [32]. Puede resultar intuitivo, pero es importante mencionar que la placa Flow Deck tiene una orientaión específica para ser instalada. Como se observa en la Figura 14, en una de sus vistas tiene indicado que es la vista frontal del sensor y que debe ser orientado hacia arriba. Este lado es el que se acopla directamente sobre el dron Crazyflie utilizando los conectores presentes en la parte inferior del dron.

Figura 14. Vista frontal y trasera de la placa de expansión Flow Deck.



Además de la orientación, también se debe considerar que la placa tiene una dirección específica para ser instalada. En ambas vistas de la Figura 14 se observa un símbolo con un círculo y una flecha. Este mismo símbolo se encuentra en la parte inferior del Crazyflie por lo que, al momento de acoplar el sensor, es importante asegurarse que la dirección de los símbolos en ambas placas (Crazyflie y Flow Deck) coincidan, justo como se observa en la Figura 15. De lo contrario, la placa podría experimentar daños irreversibles en el *hardware*.

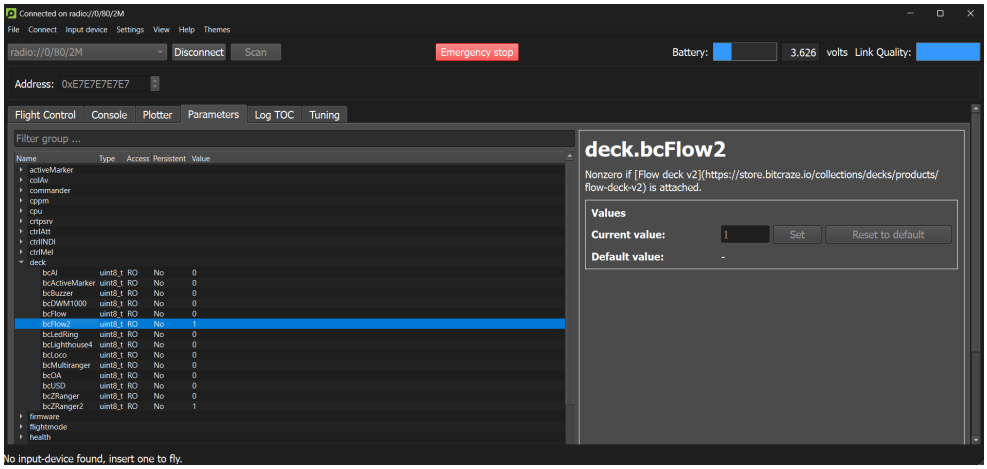
Figura 15. Vista inferior de placa Flow Deck sobre dron Crazyflie.



El siguiente aspecto que se consideró luego de instalar la placa en el dron fue la verificación de reconocimiento del sensor por parte del *firmware*. Esto se realizó utilizando nuevamente el programa cfclient y conectando el Crazyflie con la placa Flow Deck instalada.

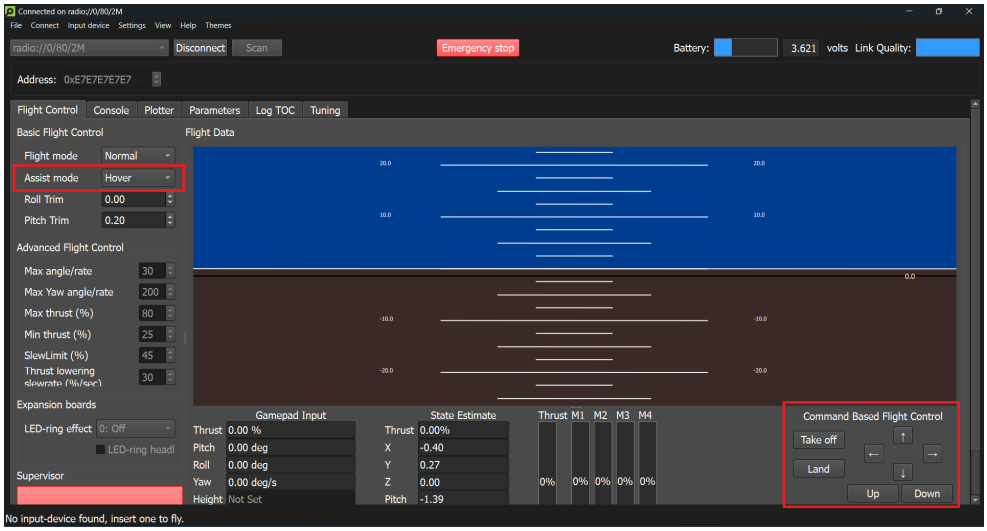
Una vez conectado el dron al cliente, se consultó la vista de parámetros y en la sección de placas, o como aparece en el cliente "*decks*", se identificó el registro **deck.bcFlow2**. Justo como se aprecia en la Figura 16, este registro es un indicador del reconocimiento de la placa de expansión Flow Deck v2 y como este automáticamente adquirió un valor de 1 se puede asegurar que el Crazyflie detectó exitosamente al sensor.

Figura 16. Reconocimiento de placa Flow Deck en vista de parámetros en cfclient.



Otra forma de identificar si el dron Crazyflie ha detectado a la placa Flow Deck es observar si en la vista de control de vuelo se habilitó el modo asistido flotante, o bien, si se habilitaron los controles de vuelo basado en comandos (ver Figura 17).

Figura 17. Reconocimiento de placa Flow Deck en vista de control de vuelo en cfclient.

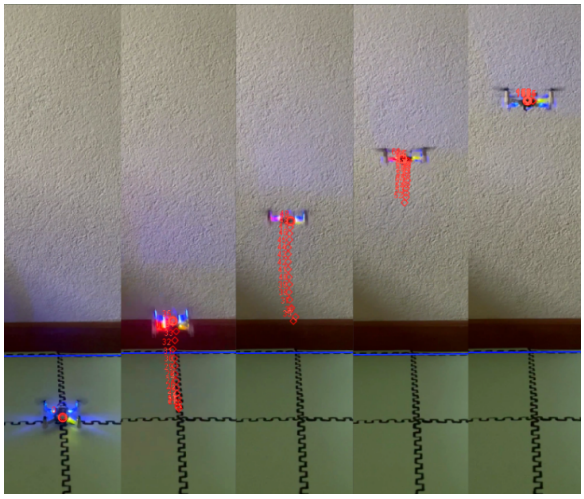


7.2.2. Pruebas de posicionamiento

Luego de verificar que la placa Flow Deck había sido reconocida por el dron, se procedió a realizar pruebas de vuelo enfocadas en evaluar la mejora en el posicionamiento y estabilidad del dron. El objetivo de estas pruebas fue comparar el desempeño del Crazyflie con y sin la placa de expansión.

Tal como se realizó el experimento sin la placa, la prueba únicamente consistió en elevar el dron a cierta altura y mantenerlo levitando por algunos segundos para despues aterrizarlo. Para ello, el dron se colocó en una superficie plana y, a diferencia de las pruebas sin placa, en estas ya no fue necesario un mando pues se encontraban habilitados los controles de vuelo basados en comandos en la GUI del cfclient (como se observó en la Figura 17). Utilizando el comando "*take off*", se logró despegar al dron y este mantuvo una altura constante hasta que se envió el comando "*land*" para aterrizarlo.

Figura 18. Prueba de vuelo y posicionamiento con placa Flow Deck integrada.



Justo como se observa en la Figura anterior 18, el desempeño de vuelo del dron fue significativamente mejor que el obtenido para la misma prueba pero sin la placa de posicionamiento. Validando su instalación y reconocmiento en el *firmware* del dron.

7.3. Adaptación de entorno

Al realizar las pruebas de posicionamiento con la placa Flow Deck, se observó que para distintas condiciones del entorno y para diferentes superficies el comportamiento de posicionamiento del dron Crazyflie fue variable. Esto se debió a las características de los sensores en la placa que mejoran su funcionamiento en condiciones específicas. Por ello, surgió la necesidad de identificar un entorno de pruebas con las condiciones óptimas para el correcto desempeño de la placa Flow Deck.

7.3.1. Superficies y condiciones desfavorables

Durante las pruebas realizadas, se observaron dificultades en la capacidad del dron para mantener su posición cuando se encontraba volando sobre superficies con condiciones de alta reflectividad, demasiada homogeneidad y con condiciones de luz intensas.

- La homogeneidad en las superficies representó un problema en el desempeño del dron debido a que el sensor de flujo óptico de la placa no era capaz de interpretar movimiento relativo al no identificar diferencias significativas en las imágenes de la superficie.
- La condición de luz intensa con sombras muy marcadas también afectó al sensor de flujo óptico, ya que la propia sombra del dron sobre la superficie generaba errores en el cálculo de movimiento relativo.
- La reflectividad en las superficies perjudicó el funcionamiento del sensor infrarrojo de la placa, provocando que esta no determinara correctamente el movimiento relativo por no tener una lectura correcta de la posición vertical del dron.

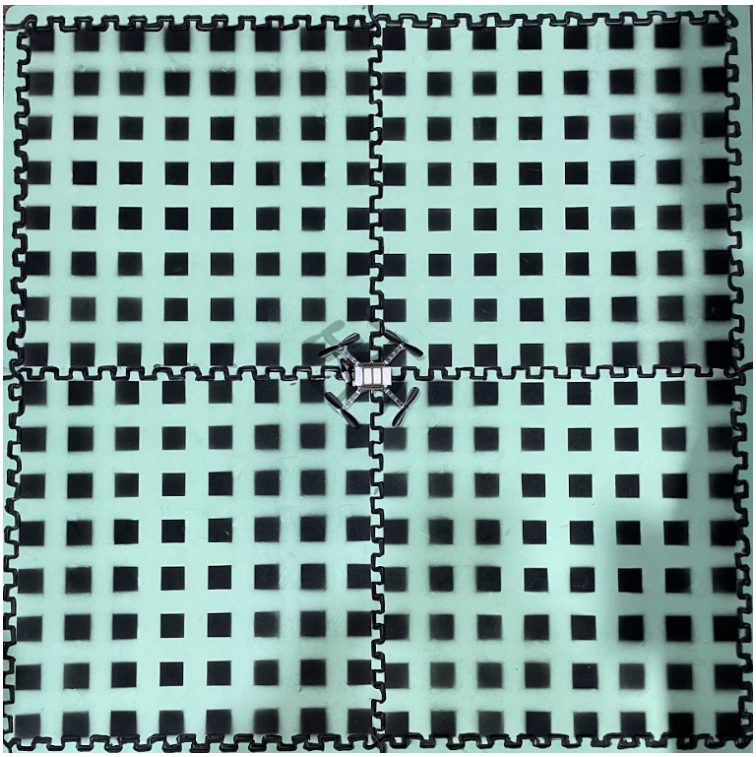
Figura 19. Superficies de prueba.

En la Figura 19 se muestran algunas de las superficies que fueron probadas. La primer imagen es en la superficie natural del ecosistema Robotat, pero la homogeneidad de la superficie presentó un pésimo posicionamiento, provocando que el dron volara erráticamente. La segunda y tercera figura corresponden a papel mate con patrones impresos, sin embargo, la pintura utilizada presentó alta reflectividad lo que produjo que el dron no fuera capaz de estimar correctamente su posición. La última superficie, una alfombra con cubierta impresa de grama artificial, presento el mismo problema debido a ser muy reflectiva.

7.3.2. Solución de entorno y observaciones

Para mitigar los problemas de funcionamiento debido a las condiciones del entorno, se tomaron varias medidas. Primero, se ajustó el entorno de vuelo empleando una superficie antirreflejante con patrones visibles, lo que mejoró significativamente la capacidad de los sensores del Flow Deck. Las superficies antirreflejantes utilizadas consistieron en alfombras modulares de foami con patrones situados con pintura antireflejante, justo como se observa en la Figura 21.

Figura 21. Prototipo de superficie para pruebas de funcionamiento de la placa Flow Deck.



Estas alfombras pueden ser adaptadas en distintos entornos y resultan especialmente prácticas para ser instaladas en el ecosistema Robotat, pues pueden ser colocadas y retiradas con facilidad. Además, al ser de material foami tienen la característica de ser amortiguadoras, ayudando a reducir el daño en el *hardware* del dron en caso de colisiones por fallas o errores.

Otro aspececto importante que se tuvo en consideración fue la ilumincación del entorno. Fue importante asegurarse que la distribución de luz fuese adecuada, de forma que se minimizara al máximo la propia sombra del dron para que no terminara afectando a las lecturas del sensor de flujo óptico.

Estas mejoras permitieron un rendimiento más estable del dron, asegurando que el Flow Deck pudiera operar de manera óptima. Estas observaciones son fundamentales para futuros experimentos, recomendando realizar vuelos en entornos controlados y adecuadamente preparados para evitar problemas de posicionamiento.

7.4. Conclusión

En conclusión, la integración y validación de la placa de expansión Flow Deck en el dron Crazyflie 2.1 permitió establecer una base sólida para mejorar la estabilidad y el control del dron en aplicaciones experimentales. A través de la instalación cuidadosa y la optimización de las condiciones del entorno, se lograron avances significativos en la precisión del posicionamiento y las capacidades de vuelo autónomo del dron. Estos resultados abren la puerta al desarrollo de herramientas de *software* que permitan maniobrar al dron de forma individual y segura.

En el siguiente capítulo, se detallará el desarrollo, implementación y validación de estas herramientas, con un enfoque en facilitar su uso y expandir el potencial del dron en aplicaciones educativas y experimentales.

CAPÍTULO 8

Herramientas de software para control individual de dron Crazyflie

En este capítulo se presenta el desarrollo e implementación de algoritmos de control básicos para el dron Crazyflie con la placa Flow Deck integrada. Se detallan los pasos de instalación de las dependencias, desarrollo de funciones de control específicas en Python y su implementación desde el entorno de Matlab. Además, se presenta un método para mejorar el rendimiento de vuelo al mezclar las lecturas de posicionamiento de la placa Flow Deck con lecturas de un sistema de captura de movimiento. Por último, se presentan herramientas de simulación para experimentar con el dron Crazyflie, detallando los alcances de estas.

8.1. Algoritmos de control básico en Python

Con el fin de lograr un control efectivo del dron Crazyflie, se desarrollaron una serie de algoritmos básicos en Python utilizando la librería oficial de Crazyflie en Python, cflib. Estos algoritmos permitieron ejecutar comandos para conexión, lectura de variables, configuración de parámetros y movimiento básicos de vuelo. Estas proporcionan una base para realizar una amplia variedad de experimentos.

8.1.1. Instalación de librería cflib

La librería Crazyflie es una herramienta fundamental para interactuar y controlar al dron mediante algoritmos en Python. Para su instalación es necesario asegurarse previamente de tener instalada una versión de Python entre las versiones 3.7 y 3.11. Para el desarrollo del proyecto se instaló la versión 3.11.0 disponible para Windows 11 en la página oficial de Python [33]. Al instalarlo se tomó el cuidado de asegurarse que la ruta a Python estuviera agregada al PATH.

Para confirmar la versión de Python instalada en el sistema, se ejecutó el siguiente comando en la terminal de Windows:

```
python --version
```

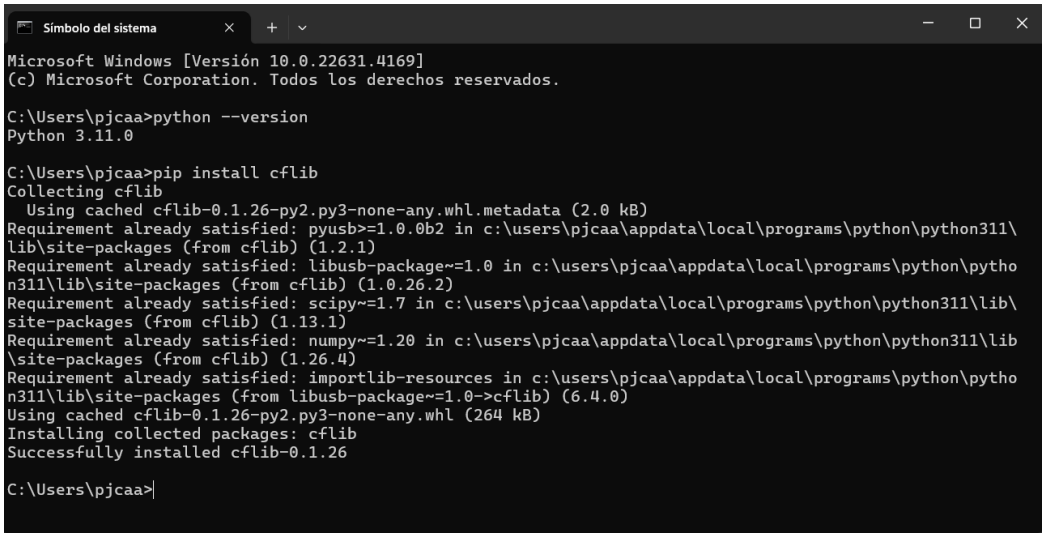
El resultado en consola confirma que la versión en uso es la **3.11.0**, como se muestra en la Figura 22.

Una vez verificada la versión de Python, se procedió a la instalación de cflib mediante el gestor de paquetes pip, ejecutando el siguiente comando:

```
pip install cflib
```

Al ejecutar este comando, se descargan e instalan automáticamente las dependencias necesarias, como **pyusb**, **libusb-package**, **scipy**, **numpy**, entre otras. La versión de la librería instalada fue **cflib-0.1.26**, como se observa en la Figura 22.

Figura 22. Ejecución de comandos en la terminal de Windows.



Tras la instalación, se llevaron a cabo pruebas básicas para garantizar que la librería cflib se instaló correctamente y que el dron puede recibir comandos desde un script de Python. Es importante mencionar que, a lo largo de este proyecto, se utilizó el programa Visual Studio Code como editor de texto para los algoritmos en Python.

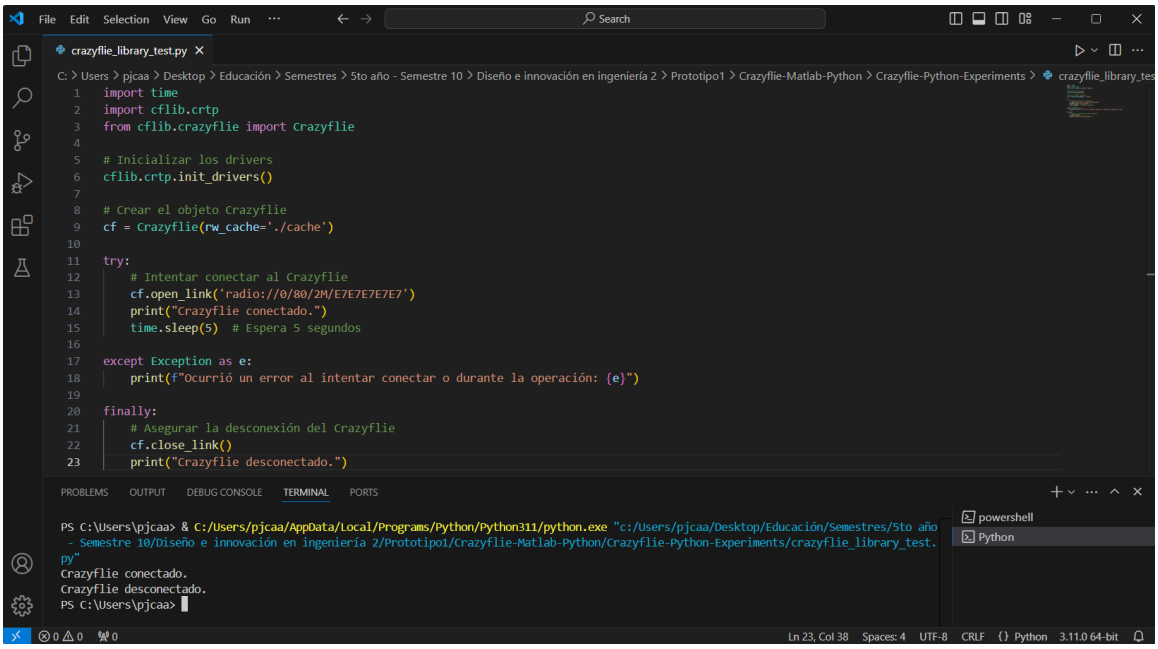
A continuación, se presenta el código de prueba que realiza la conexión y desconexión del dron (ver Código 8.1).

Código 8.1. Algoritmo de prueba en Python utilizando la librería Cflib.

```python
import time
import cflib.crtp
from cflib.crazyflie import Crazyflie

# Inicializar los drivers
cflib.crtp.init_drivers()

# Crear el objeto Crazyflie
cf = Crazyflie(rw_cache='./cache')

try:
# Intentar conectar al Crazyflie
cf.open_link('radio://0/80/2M/E7E7E7E7E7')
print("Crazyflie conectado.")
time.sleep(5)  # Espera 5 segundos
except Exception as e:
print(f"Ocurrió un error al intentar la conexión: {e}")
finally:
# Asegurar la desconexión del Crazyflie
cf.close_link()
print("Crazyflie desconectado.")
```

El algoritmo presentado realiza un proceso simple de conexión con el dron Crazyflie, manteniendo activa la conexión por 5 segundos para luego cerrar la conexión. Para lograrlo se utilizó la libería time y los módulos CRTP y Crazyflie de la libería cflib. El móudlo CRTP se utilizó para inicializar los drivers para el dispositivo Crazyradio y el módulo Crazyflie se empleó para instanciar un objeto de la clase principal Crazyflie, este fue la representación del dron en el código. En la Figura 23, se muestra el entorno de desarrollo de *VS Code* y la correcta compilación y ejecución del algoritmo de prueba.

Figura 23. Ejecución de algoritmo de prueba en Visual Studio Code.



8.1.2. Uso de librería cflib y recursos adicionales

Para el desarrollo de los algoritmos de control, se utilizaron módulos específicos de la librería cflib para Crazyflie y algunas librerías estándar de Python. A continuación, se detallan las librerías y módulos utilizados con su propósito:

Módulos de cflib utilizados

- **CRTP:** el módulo cflib.crtp es responsable de iniciar los drivers necesarios para la comunicación mediante *Crazy Real-Time Protocol*, utilizado para establecer la conexión con el dron a través del dispositivo Crazyradio.
- **Crazyflie:** el módulo cflib.crazyflie permite controlar y comunicarse con el dron.
  - Submódulo Log: el submódulo cflib.crazyflie.log se empleó para configurar los registros de variables en tiempo real.
  - Submódulo SyncCrazyflie: el submódulo cflib.crazyflie.synccrazyflie se empleó para simplificar el manejo de conexiones mediante el uso de la subclase SyncCrazyflie.
  - Submódulo HighLevelCommander: el submódulo cflib.crazyflie.high_level_commander se empleó como una interfaz de alto nivel para el envío de comandos de control de vuelo hacia el dron.

Es importante mencionar que se utilizó la versión síncrona de la librería Crazyflie en lugar de la versión asíncrona, como se evidencia en el uso de la subclase SyncCrazyflie. Esta convierte a las funciones no bloqueantes de la versión asíncrona en funciones bloqueantes, lo que significa que el programa espera que una tarea termine antes de continuar con la siguiente.

Esta elección se realizó por la necesidad de ejecutar comandos de control de manera secuencial y confiable. Al utilizar una versión síncrona, se asegura que cada acción enviada sea completada antes de ejecutar la siguiente. De esta forma, el dron puede completar algoritmos de forma secuencial, tal como se esperaría para un algoritmo de seguimiento de trayectorias.

Recursos adicionales de Python utilizados

- **logging:** se utilizó para configurar el nivel de registro de mensajes dentro del programa.
- **time:** se utilizó para gestionar el tiempo en la ejecución de las funciones de control. Esto asegurpo que el dron tuviera el tiempo suficiente para ejecutar las acciones solicitadas antes del siguiente comando.
- **sys:** se utilizó para forzar la salida inmediata de los mensajes en la consola. Esto evitó que el sistema almacenara mensajes en *buffer* y retrasara su impresión en consola.
- **threading:** se utilizó la clase *Event* del módulo para sincronizar la recepción de datos del dron.

8.1.3. Desarrollo de algoritmos de control

A continuación, se presentan las funciones desarrolladas en Python con algoritmos de control de funcionalidades básicas del dron Crazyflie. Están clasificadas en cuatro grupos: conexión, lectura de variables, configuración de parámetros y movimiento general.

Funciones de conexión y desconexión

- **connect(uri)** (Código 13.1)
  Esta función se utiliza para establecer y mantener una conexión activa con el dron Crazyflie de manera síncrona. Requiere que sea especificado el Identificador Unifrome de Recursos (URI) del Crazyflie objetivo.
  - Atributos:
    - uri (str): Dirección URI del Crazyflie a conectar.
  - Valor de retorno:
    - Retorna una instancia de la clase SyncCrazyflie si la conexión es exitosa.
    - Imprime en consola un mensaje de confirmación de conexión o de alerta de error con los detalles correspondientes.
- **disconnect(SyncCrazyflie)** (Código 13.2)
  Esta función se utiliza para cerrar la conexión activa con el dron Crazyflie. Requiere que sea especificada la instancia de la clase SyncCrazyflie generada al conectarse .
  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de desconexión o de alerta de error con los detalles correspondientes.

Funciones de lectura de variables y configuración de parámetros

- **get_pose(SyncCrazyflie)** (Código 13.3)
  Esta función se utiliza para obtener los valores actuales de los registros de la estimación de posición del dron Crazyflie.
  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
  - Valor de retorno:
    - Retorna un diccionario con los valores de posición actuales del Crazyflie. Las llaves del diccionario correspondiente son: “x”, “y”, “z”, “roll”, “pitch”, “yaw”.
    - Imprime en consola un mensaje con los valores de posición actual o un mensaje de alerta de error con los detalles correspondientes.

- **get_pid_values(SyncCrazyflie)** (Código 13.4)
  Esta función se utiliza para obtener los valores actuales de los registros de los controladores PID de posición del Crazyflie.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
  - Valor de retorno:
    - Retorna un diccionario con los valores actuales de los controladores PID de posición del Crazyflie. Las llaves del diccionario correspondiente son: "X", "Y", "Z" y cada una posee un arreglo con las constantes P, I y D.
    - Imprime en consola un mensaje con los valores actuales de los controladores PID de posición o un mensaje de alerta de error con los detalles correspondientes.

- **get_pid_x(scf)**, **get_pid_y(scf)** y **get_pid_z(scf)** (Código 13.5)
  Estas funcones se utilizan para obtener los valores actuales del controlador PID de posición en el eje correspondiente del dron Crazyflie.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
  - Valor de retorno:
    - Retorna un arreglo con los valores actuales de las tres constantes del controlador PID de posición del eje correspondiente.
    - Imprime en consola un mensaje con los valores del controlador PID de posición del eje correspondiente o un mensaje de alerta de error con los detalles correspondientes.

- **detect_flow_deck(SyncCrazyflie)** (Código 13.6)
  Esta función se utiliza verificar que el dron Crazyflie conectado esté detectando correctamente a la placa de expansión Flow Deck. Fue desarrollada como una función de prevención de accidentes, ya que si no se detecta dicha placa el comportamiento del dron Crazyflie es errático.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
  - Valor de retorno:
    - Retorna un valor True si se detecta correctamente la placa Flow Deck y False si esta no es detectada.
    - Imprime en consola un mensaje confimando o negando la detección de la placa de expansión Flow Deck.

- **set_position(SyncCrazyflie, x, y, z)** (Código 13.7)
  Esta función se utiliza para configurar el valor en los registros de la estimación de posición del dron Crazyflie, es decir, actualiza la posición actual en el marco de referencia relativo del dron Crazyflie.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - x (float): Coordenada X en metros de la posición a configurar.
    - y (float): Coordenada Y en metros de la posición a configurar.
    - z (float): Coordenada Z en metros de la posición a configurar.
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de actualización de posición relativa o un mensaje de alerta de error con los detalles correspondientes.

- **set_pid_values(SyncCrazyflie, p_gains, i_gains, d_gains)** (Código 13.9)
  Esta función se utiliza para configurar los valores actuales de los registros de los controladores PID de posición del Crazyflie.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - p_gains (diccionario): Diccionario con constantes proporcionales de los controladores PID de posición para los ejes XYZ. Las llaves del diccionario deben ser: "X", "Y" y "Z".
    - i_gains (diccionario): Diccionario con constantes integrativas de los controladores PID de posición para los ejes XYZ. Las llaves del diccionario deben ser: "X", "Y" y "Z".
    - d_gains (diccionario): Diccionario con constantes derivativas de los controladores PID de posición para los ejes XYZ. Las llaves del diccionario deben ser: "X", "Y" y "Z".
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de actualización de constantes de controladores PID de posición o un mensaje de alerta de error con los detalles correspondientes.

- **set_pid_x(scf, P, I, D)**, **set_pid_y(scf, P, I, D)** y **set_pid_z(scf, P, I, D)** (Código 13.8)
  Estas funciones se utilizan para configurar los valores de las constantes del controlador PID de posición en el eje correspondiente del dron Crazyflie.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - P (float): Constante proporcional para el controlador PID de posición del eje correspondiente.

    - I (float): Constante integrativa para el controlador PID de posición del eje correspondiente.
    - D (float): Constante derivativa para el controlador PID de posición del eje correspondiente.
  - Valor de retorno:
    - Imprime en consola un mensaje con los valores del controlador PID de posición del eje correspondiente o un mensaje de alerta de error con los detalles correspondientes.

Funciones de movimiento

- **takeoff(SyncCrazyflie, height, duration)** (Código 13.10)
  Esta función se utiliza para realizar el despegue del dron Crazyflie a una altura dada durante una duración especificada. En caso de no ser especificados la altura y duración, se utilizan los valores por defecto de 0.3 metros y 1 segundo.
  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - height (float): Altura en metros que alcanzará durante el despegue.
    - duration (float): Tiempo en segundos que tardará en alcanzar la altura de despegue indicada.
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de despegue completado o de alerta de error con los detalles correspondientes.

- **land(SyncCrazyflie, height, duration)** (Código 13.11)
  Esta función se utiliza para realizar el aterrizaje del dron Crazyflie a una altura dada durante una duración especificada. En caso de no ser especificados la altura y duración, se utilizan los valores por defecto de 0 metros y 2 segundos.
  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - height (float): Altura en metros que alcanzará durante el despegue.
    - duration (float): Tiempo en segundos que tardará en alcanzar la altura de aterrizaje indicada.
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de aterrizaje completado o de alerta de error con los detalles correspondientes.

- **move_to_position(SyncCrazyflie, x, y, z, velocity)** (Código 13.12)
  Esta función se utiliza para mover al dron a una posición nueva con una velocidad dada. Requiere de las coordenadas XYZ de la nueva posición y la velocidad del movimiento. La velocidad puede ser omitida y toma el valor por defecto de 1 metro por segundo.

  - Atributos:
    - SyncCrazyflie (scf): Instancia de la clase SyncCrazyflie generada al establecer la conexión.
    - x (float): Coordenada X en metros de la nueva posición.
    - y (float): Coordenada Y en metros de la nueva posición.
    - z (float): Coordenada Z en metros de la nueva posición.
    - velocity (float):Velocidad en metros por segundo del movimiento.
  - Valor de retorno:
    - Imprime en consola un mensaje de confirmación de posición alcanzada o de alerta de error con los detalles correspondientes.

8.1.4. Experimentos con algoritmos de control

Una vez desarrolladas las rutinas de control básico en forma de funciones de Python, se desarrollaron algoritmos utilizando dichas funciones para validar su uso en experimentos simples. Se realizaron tres experimentos: prueba de despegue simple (Código 8.2), prueba de despegue y aterrizaje con modificación del PID de posición Z (Código 8.3) y prueba movimiento de punto a punto (Código 8.4).

Experimento simple de despegue y aterrizaje

Código 8.2. Algoritmo de prueba de takeoff y land con Crazyflie.

```
1  def main():
2  uri = 'radio://0/80/2M/E7E7E7E7E7'
3  scf = connect(uri)
4
5  if scf:
6  takeoff(scf, height=0.5, duration=3.0)
7  land(scf, height=0.0, duration=2.0)
8  disconnect(scf)
9  else:
10 print("Failed to establish a connection to the Crazyflie.")
11
12 if __name__ == '__main__':
13 main()
```

Experimento de despegue y aterrizaje con modificación del PID de posición Z

Código 8.3. Algoritmo de prueba de takeoff y land con Crazyflie.

```
def main():
uri = 'radio://0/80/2M/E7E7E7E7E7'
scf = connect(uri)

if scf:
set_pid_z(scf, P=4.0, I=2.5, D=0.01)
takeoff(scf, height=0.5, duration=3.0)
land(scf, height=0.0, duration=2.0)
disconnect(scf)
else:
print("Failed to establish a connection to the Crazyflie.")

if __name__ == '__main__':
main()
```

Experimento de movimiento de punto a punto

Código 8.4. Algoritmo de prueba de takeoff y land con Crazyflie.

```
def main():
uri = 'radio://0/80/2M/E7E7E7E7E7'
scf = connect(uri)

if scf:
get_pid_z(scf)
takeoff(scf, height=0.5, duration=3.0)
move_to_position(scf, x=1.0, y=1.0, z=0.5)
move_to_position(scf, x=0.0, y=0.0, z=0.5)
land(scf, height=0.0, duration=2.0)
disconnect(scf)
else:
print("Failed to establish a connection to the Crazyflie.")

if __name__ == '__main__':
main()
```

El desarrollo de las funciones de control fue un proceso iterativo, por lo que estas pruebas fueron realizadas constantemente durante dicho proceso. Estas funciones fueron modificadas hasta alcanzar un resultado aceptable en el comportamiento de vuelo del dron Crazyflie.

8.2. Algoritmos de control básico desde Matlab

Aunque los algoritmos de control fueron desarrollados en Python, el entorno comúnmente utilizado en cursos y laboratorios de la Universidad del Valle de Guatemala es Matlab. Por esta razón se exploró la posibilidad de integrar las funciones desde Matlab, permitiendo a los estudiantes y profesores utilizar una interfaz conocida para experimentar con el dron.

8.2.1. Interfaz de Matlab para interacción con Python

Matlab proporciona la opción de interactuar directamente con secuencias, funciones y bibliotecas en Python, lo que facilita la integración de ambos lenguajes. Sin embargo, para utilizar esta característica deben tenerse en cuenta las siguientes condiciones:

- Matlab: tener instalada una versión reciente de Matlab ya que la compatibilidad con Python se ha mejorado en las versiones más recientes.
- Python: tener instalada una versión de Python en el sistema operativo. Matlab es compatible con distintas versiones de Python, pero se recomienda usar una versión que sea compatible con la versión de Matlab instalada. Consultar la matriz de compatibilidad de versiones en la página oficial de Matlab [34].

Es importante mencionar que para fines de este trabajo de graduación se utilizó la versión '24.1.0.2603908 (R2024a) Update 3' de Matlab y la versión 3.11.0 de Python (compatibles según la matriz de compatibilidad de Matlab).

8.2.2. Configuración del entorno en Matlab

Matlab es capaz de reconocer automáticamente la instalación de Python cuando esta fue correctamente instalada. Por lo que, para verificar que Matlab fue capaz de detectar la instalación de Python, bastó con ejecutar el comando mostrado en el Código 8.5 en la terminal de Matlab.

Código 8.5. Comando en terminal de Matlab para verificar la detección de Python.

```
1   pyenv
```

El resultado de ejecutar el comando fue impreso en consola de Matlab, como se observa en la Figura 24. La versión detectada fue la 3.11 con las rutas y el indicador *status* con valor de 1, lo que significa que la versión de Python fue cargada correctamente. De esta forma, se aseguró que Matlab era capaz de ejecutar *scripts* y código de Python. Si la versión de Python no fuese la correcta puede emplearse el comando mostrado en el Código 8.6 para cambiar la versión en uso.

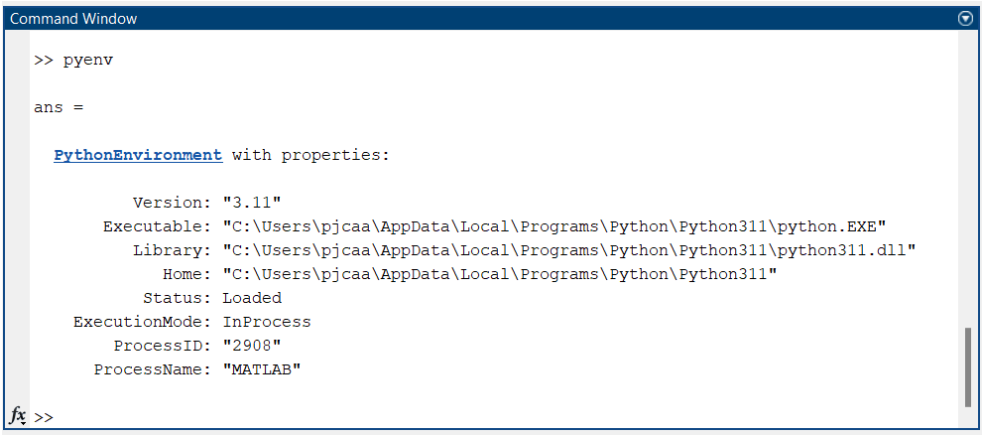
Código 8.6. Comandos para identificar la versión de Python detetada por Matlab.

```
1   pyenv('Version', 'ruta/a/python')
```

Ejecución de algoritmos de Python desde Matlab

Una vez comprobado que el entorno de trabajo de Matlab había cargado correctamente la versión de Python deseada, se procedió a verificar la capacidad de importar módulos de Python directamente en Matlab. Para ello, se realizó nuevamente un experimento de conexión y desconexión del Crazyflie.

Figura 24. Terminal de comandos de Matlab con ejecución del comando pyversion.



Código 8.7. Algoritmo de prueba de conexión con Crazyflie.

```
import time
import logging
import cflib.crtp
from cflib.crazyflie import Crazyflie

def connect_crazyflie():
cflib.crtp.init_drivers()
logging.basicConfig(level=logging.CRITICAL)
cf = Crazyflie(rw_cache='./cache')

try:
cf.open_link('radio://0/80/2M/E7E7E7E7E7')
print(f"Crazyflie conectado.")
time.sleep(5)  # Espera 5 segundos

except Exception as e:
print(f"Ocurrió un error al intentar conectar o durante la operación: {
    e}")

finally:
cf.close_link()
print(f"Crazyflie desconectado.")
```

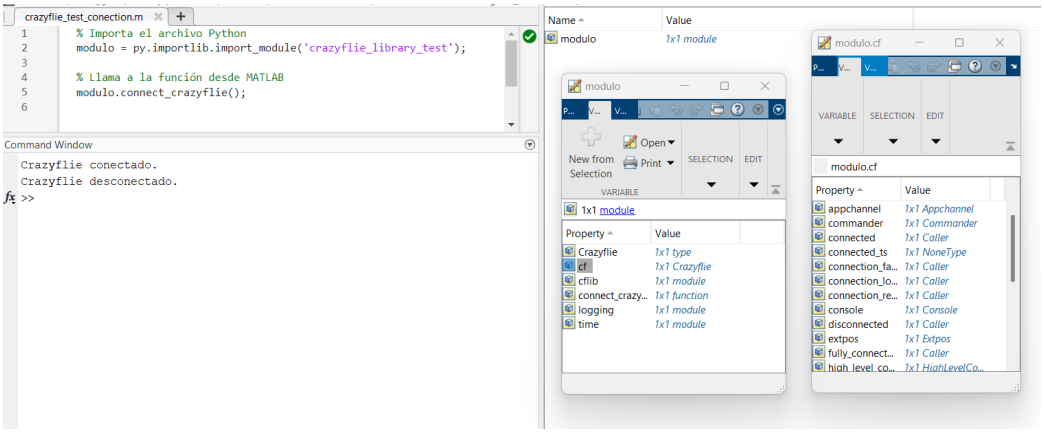
El Código 8.7 corresponde a un script en Python con una función que realiza la conexión y desconexión del Crazyflie. Para realizar la llamada del script de Pyhton desde Matlab y ejecutar la función se utilizaron los siguientes comandos propios de Matlab:

```
% Importa el archivo Python
modulo = py.importlib.import_module('crazyflie_library_test');

% Llama a la función desde MATLAB
modulo.connect_crazyflie();
```

El resultado obtenido se muestra en la Figura 25, en donde se observa que en la consola de Matlab se mostraron los mensajes programados en el algoritmo de Python y se creó

un objeto llamado módulo que internamente poseía las instancias de las librerías y objetos creados en el algortimo de Python. Lo que demostró que fue efectiva la ejecución y carga de algoritmos de Python directamente desde Matlab.

Figura 25. Ejecución de script en Python de código 8.1 desde Matlab.



8.2.3. Algoritmos de control en Python desde Matlab

Una vez demostrada la correcta ejecución de algoritmos de Python desde Matlab, se desarrolló un conjunto de comandos en Matlab que permitieron ejecutar funciones escritas en Python de forma directa y sencilla. Estos comandos actuaron como funciones de Matlab que, internamente, llaman de manera individual a los módulos y funciones correspondientes en Python. De esta manera, fue posible enviar comandos al Crazyflie sin la necesidad de interactuar directamente con Python, facilitando el uso de algoritmos de control y la realización de experimentos de vuelo complejos en el entorno conocido de Matlab.

Funciones de conexión y desconexión en Matlab

- **crazyflie_connect(uri):** establece una conexión con el dron Crazyflie utilizando la dirección uri proporcionada. Retorna un objeto SyncCrazyflie que representa la conexión activa.
- **crazyflie_disconnect(SyncCrazyflie):** cierra de forma segura la conexión con el dron representada por el objeto SyncCrazyflie.

Funciones de lectura de variables en Matlab

- **crazyflie_get_pose(SyncCrazyflie):** recupera la posición actual (x, y, z) y la orientación (roll, pitch, yaw) del dron en el espacio.
- **crazyflie_get_pid_values(SyncCrazyflie):** obtiene los valores actuales de los controladores PID (Proporcional, Integral y Derivativo) para los ejes x, y y z.

- **crazyflie_get_pid_x(SyncCrazyflie):** devuelve los valores PID para el control de posición en el eje x.
- **crazyflie_get_pid_y(SyncCrazyflie):** devuelve los valores PID para el control de posición en el eje y.
- **crazyflie_get_pid_z(SyncCrazyflie):** devuelve los valores PID para el control de posición en el eje z.
- **crazyflie_detect_flow_deck(SyncCrazyflie):** verifica si la placa de expansión Flow Deck está correctamente instalada y detectada por el dron.

Funciones de configuración de parámetros en Matlab

- **crazyflie_set_position(SyncCrazyflie, x, y, z):** configura la posición actual del dron en coordenadas relativas (x, y, z), actualizando sus registros de posición.
- **crazyflie_set_pid_values(SyncCrazyflie, p_gains, i_gains, d_gains):** ajusta los valores PID para los ejes x, y y z según los valores especificados en los diccionarios p_gains, i_gains y d_gains.
- **crazyflie_set_pid_x(SyncCrazyflie, P, I, D):** configura los valores PID específicos para el eje x.
- **crazyflie_set_pid_y(SyncCrazyflie, P, I, D):** configura los valores PID específicos para el eje y.
- **crazyflie_set_pid_z(SyncCrazyflie, P, I, D):** configura los valores PID específicos para el eje z.

Comandos de movimiento en Matlab

- **crazyflie_takeoff(SyncCrazyflie, height, duration):** realiza un despegue controlado, elevando el dron a una altura height durante un tiempo especificado por duration.
- **crazyflie_land(SyncCrazyflie, height, duration):** aterriza el dron de manera controlada, descendiendo a una altura height durante un tiempo especificado por duration.
- **crazyflie_move_to_position(SyncCrazyflie, x, y, z, velocity):** mueve el dron a una nueva posición definida por las coordenadas (x, y, z) con una velocidad constante velocity.

8.2.4. Experimentos con algoritmos de control desde Matlab

El desarrollo de los comandos de control en Matlab fue un proceso iterativo, ya que para cada comando desarrollado fue necesario realizar una serie de pruebas distintas para validar su funcionamiento bajo distintas condiciones. Una vez concluidos los comandos y verficado

el funcionamiento individual, fue posible realizar experimentos más complejos utilizando todos los comandos en conjunto. A continuación, se presentan algunos de los experimentos realizados:

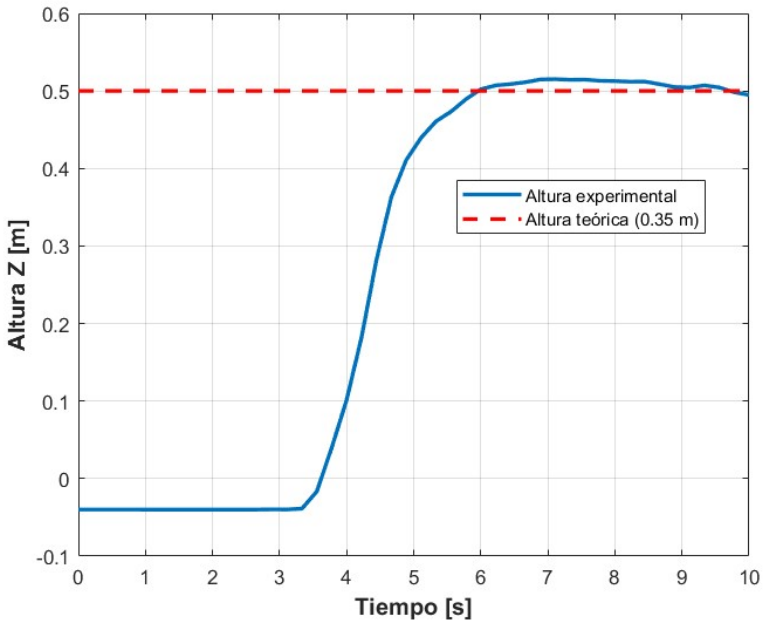
Experimento de despegue simple

Este experimento consistió en un despegue básico, evaluando la estabilidad del dron durante la elevación y comprobando la capacidad de mantenerse en vuelo estacionario.

Código 8.8. Algoritmo de experimento de despegue simple con Crazyflie en Matlab.

```
dron_id = 8;
crazyflie_1 = crazyflie_connect(dron_id);
crazyflie_takeoff(crazyflie_1, 0.5, 2.5);
crazyflie_land(crazyflie_1);
crazyflie_disconnect(crazyflie_1);
```

Figura 26. Altura contra tiempo del experimento de despegue simple.



En la Figura 26, se muestra el comportamiento de vuelo para el experimento de despegue simple y evidencia el comportamiento esperado. Inicia en una altura aproximada de 0 metros y comienza a elevarse de manera progresiva hasta alcanzar la altura indicada de 0.5 metros. Este experimento se desarrolló con los valores normales para el controlador PID, por lo que se observó un comportamiento bastante estable y con error mínimo.

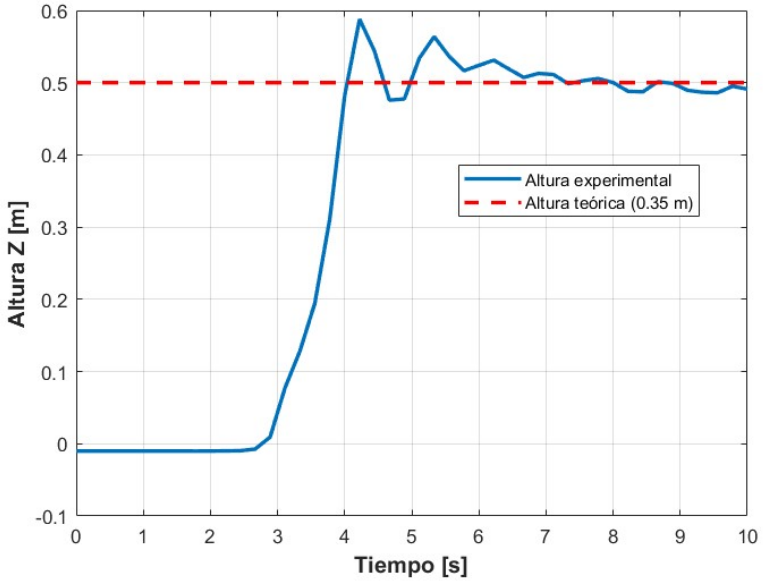
Experimento de despegue con modificación en el control PID de posición z

Se realizó una modificación de los parámetros PID para el control en el eje Z durante el despegue, evaluando el impacto en la precisión y estabilidad del vuelo.

Código 8.9. Algoritmo de experimento de despegue con modificación en el control PID de altura con Crazyflie en Matlab.

```
dron_id = 8;
crazyflie_1 = crazyflie_connect(dron_id);
crazyflie_set_pid_z(crazyflie_1, 2.5, 1.5, 0.01);
crazyflie_takeoff(crazyflie_1, 0.5, 2.5);
crazyflie_land(crazyflie_1);
crazyflie_disconnect(crazyflie_1);
```

Figura 27. Altura contra tiempo del experimento de despegue con modificación de control PID de altura.



En la gráfica de la Figura 27, se muestra el despegue del dron Crazyflie con la modificación del control PID de altura. Tal como era esperado, se observa una variación en el comportamiento de vuelo del dron que, para el caso de los parámetros seleccionados se observó mayor sobre-elevación (*overshoot*) y oscilaciones. Esto se evidencia al ver que el vuelo presenta sobreimpulso al alcanzar la altura deseada, es decir, sobrepasa la altura y luego intenta estabilizarse. De esta forma, se validaron las funciones de configuración de parámetros de los controladores PID de posición.

Experimento de seguimiento de trayectoria circular

En este experimento, se programó al Crazyflie para seguir una trayectoria circular.

Código 8.10. Algoritmo de experimento de seguimiento de trayectoria circular con Crazyflie en Matlab.

```
circle_center = [0,0,0.5];
N = 20;
radio = 0.3;
theta = linspace(0, 2*pi, N);
x = circle_center(1) + radio * cos(theta);
y = circle_center(2) + radio * sin(theta);
z = circle_center(3) * ones(1, N);

dron_id = 8;
crazyflie_1 = crazyflie_connect(dron_id);
crazyflie_takeoff(crazyflie_1, 0.3, 2.5);
for i = 1:N
crazyflie_move_to_position(crazyflie_1, x(i), y(i), z(i), 0.2);
end
crazyflie_land(crazyflie_1);
crazyflie_disconnect(crazyflie_1);
```

Figura 28. Seguimiento de trayectoria circular.

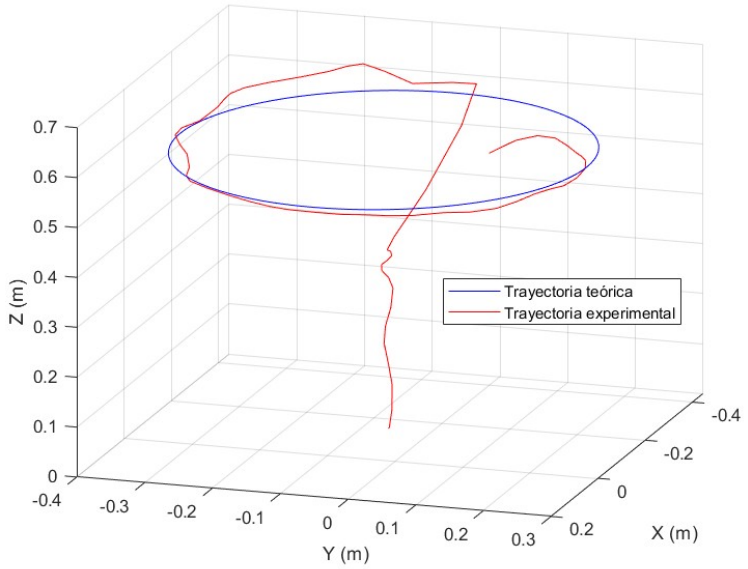
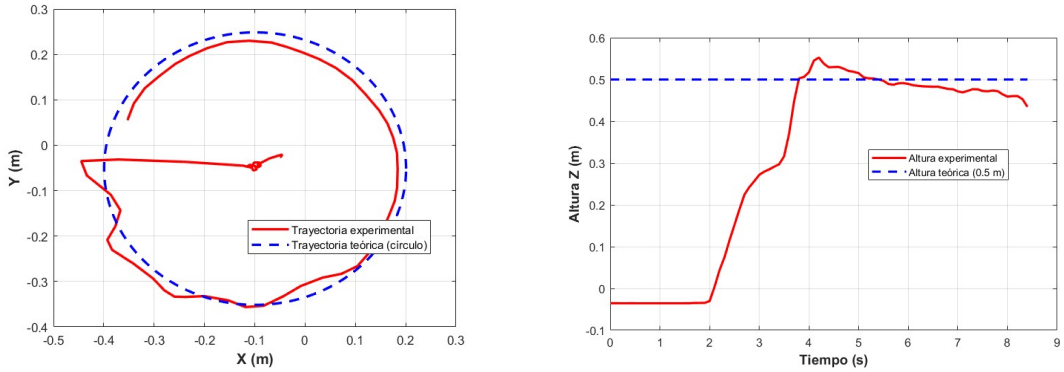


Figura 29. Posición X contra Y y altura contra tiempo para el seguimiento de trayectoria circular.



Al analizar las gráficas mostradas en las Figuras 28 y 29, se observa que el dron Crazyflie fue capaz de realizar el seguimiento de las trayectorias generadas aunque no perfectamente. En la gráfica del plano X-Y, se observa que el dron experimenta desviaciones significativas en la trayectoria circular y en la gráfica de altura conta tiempo, se ve que el dron presenta una disminución progresiva de altura conforme pasa el tiempo.

Las posibles fuentes de error incluyen limitaciones con el sensor Flow Deck, que puede verse afectado por la superficie de experimentación o la iluminación del entorno de pruebas. Aunque se intentó adaptar las condiciones del entorno para que fueran óptimas para su funcionamiento, parece que este aún presenta dificultades para posicionarse con precisión, lo que resulta en dificultades para mantener un seguimiento perfecto de la trayectoria generada. Otra posible fuente de error podría encontrarse en la integridad de las baterías de alimentación del dron ya que, al no ser las originales de Crazyflie estas podrían no ser totalmente compatibles y haberse dañado con el tiempo.

8.3. Fusión de sensores

Derivado de los resultados obtenidos, se identificó que el comportamiento de vuelo podía ser mejorado de forma considerable al realizar una fusión de sensores con un sistema de posicionamiento absoluto. A raíz de ello, se intentó fusionar las lecturas de posicionamiento de la placa de expansión Flow Deck con los datos de pose obtenidos a través del sistema de captura de movimiento del ecosistema Robotat.

De lograrse esto, permitiría optimizar significativamente el rendimiento del dron ya que las lecturas de posición relativa proporcionadas por la palca Flow Deck se complementarían con las lecturas absolutas del sistema MoCap, brindando al dron una referencia precisa de su posición dentro del entorno. Los datos de ambas fuentes serían procesados mediante el filtro extendido de Kalman del dron para una estimación más exacta de su posición, mejorando considerablemente la precisión de vuelo del Crazyflie en el seguimiento de trayectorias.

8.3.1. Sistema de captura de movimiento y funcionamiento desde Matlab

El MoCap está compuesto por una red de cámaras que rastrean marcadores reflectivos adheridos a los objetos que se desean monitorear dentro del espacio controlado. Dichos marcadores permiten que el sistema calcule la posición y orientación exacta de los objetos en tiempo real respecto a una referencia absoluta.

En el ecosistema Robotat, se utiliza una red TCP para la comunicación con el Mocap. Matlab es el *software* mayormente utilizado como panel de control en los laboratorios y experimentos de la universidad. Por lo tanto, es por medio de Matlab que se establece una conexión con el sistema MoCap para solicitar las poses de los marcadores reflectivos. A continuación, se listan las funciones principales para controlar el sistema:

- **tcp_obj = robotat_connect()**: esta función establece la conexión con la red TCP del ecosistema Robotat.
- **pose = robotat_get_pose(tcp_obj, id_agent)**: solicita y devuelve la posición y orientación del marcador reflectivo especificado por id_agent.
- **robotat_disconnect(tcp_obj)**: finaliza la conexión con el sistema.

Marcador reflectivo

Un marcador reflectivo es un conjunto de pequeñas esferas reflectantes sujetadas sobre los cuerpos que se desean rastrear dentro del sistema. Estos marcadores están diseñados para reflejar luz emitida por las cámaras, permitiendo que el *software* reconozca su posición en el espacio con alta precisión.

Para llevar a cabo la fusión de sensores de la placa Flow Deck y el sistema MoCap en el dron Crazyflie, fue necesario colocar un marcador reflectivo sobre el dron. Para ello, se desarrollaron distintos prototipos de marcador que pudieran colocarse en la parte superior del Crazyflie sin afectar a las hélices y que fuera fácilmente perceptible por el sistema MoCap.

Figura 30. Versiones iniciales del marker para Crazyflie.

En la Figura 30 se muestran las primeras versiones desarrolladas para sujetar a las esferas reflectantes que conformarían al marcador reflectivo del dron Crazyflie. Sin embargo, presentaron cirtas dificultades al utilizar el sistema MoCap, pues en ocasiones no se registraban lecturas del cuerpo rígido. Un factor que causo esto fue que las esferas eran sobrepuestas en agujeros pero sin sostenerlas de forma definitiva, lo que ocasionaba que se movimieran milimétricamente. También, se cree que la poca distancia entre esferas dificultaba el reconocimiento del cuerpo rígido en sistema MoCap.

Figura 31. Versión final de marcador reflectivo para dron Crazyflie.

Tras observar estos factores, se realizó el diseño final del marcador para el dron Crazyflie, mostrado en la Figura 31. Este dispone de agujeros para sujetar a las esferas mediante tornillos M3, de forma que la posición de estas queda totalmente rígida. Además, se utilizó una distancia considerablemente mayor entre esferas reflectivas y se añadió una esfera extra para disminuir la pérdida de reconocimiento por parte del sistema MoCap.

8.3.2. Implementación de fusión de sensores

Para la implementación de la fusión de sensores previamente descrita, se utilizó el submódulo *extpos* de la clase Crazyflie en la librería cflib de Python. Los datos obtenidos del sistema MoCap se enviaron mediante las funciones de este submódulo para actualizar la posición absoluta del dron, por medio del comando crazyflie_set_pose en Matlab.

Event Callback en Matlab

Se intentó desarrollar una rutina de Event CallBack en Matlab, en la cual el evento sería la recepción de datos desde la red TCP. Sin embargo, debido a la naturaleza del sistema Robotat, que opera bajo un esquema de solicitud y recepción de información, no fue posible implementar un evento continuo ya que la red TCP no envía datos de manera constante sin una solicitud previa. Para lograr implementar una rutina Event Callback es necesario que

el Robotat pueda funcionar bajo otro esquema de comunicación, por ejemplo un caso de suscripción para la recepción continua de información.

Función de movimiento con corrección de posición

Debido a la imposibilidad de implementar una suscripción para la recepción continua de datos en tiempo real, se utilizó un método de corrección de pose durante el movimiento del dron. La actualización de pose no se realizaría de manera periódica, sino que se realizaría conforme el dron se desplaza sobre una trayectoria. Es decir, las lecturas de pose del Robotat se utilizan como compensaciones de las desviaciones que ocurren durante el uso de la placa Flow Deck, de esta forma se mejora la precisión general de vuelo del dron dentro del ecosistema Robotat.

8.4. Simulador para dron Crazyflie 2.1

Los simuladores desempeñan un papel fundamental en el desarrollo y la validación de algoritmos para robots, permitiendo probar y ajustar sistemas en entornos controlados sin riesgo para el *hardware* ni para los experimentadores. En esta sección, se evaluará la viabilidad del uso del simulador Webots para representar el modelo del dron Crazyflie 2.1 y sus aplicaciones en el contexto de este proyecto.

8.4.1. Webots

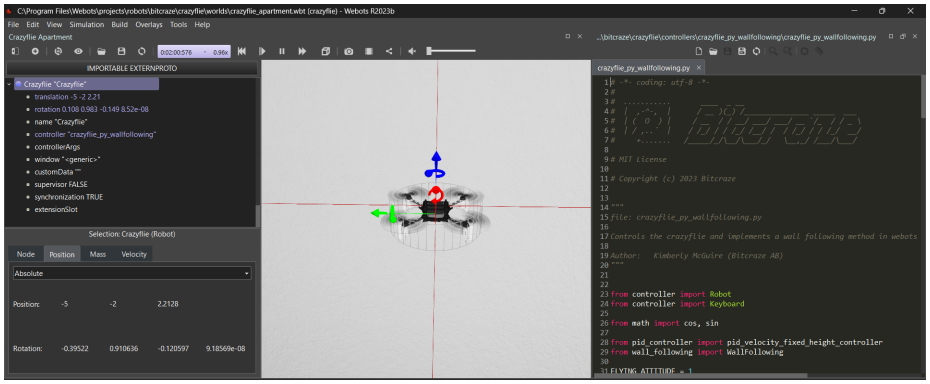
Webots es un simulador de robots de código abierto ampliamente reconocido por su capacidad para emular diversos agentes robóticos en entornos tridimensionales. Desarrollado por Cyberbotics, Webots es una herramienta versátil que soporta la programación en varios lenguajes, incluyendo Python, C++, Java y Matlab, lo que le ha otorgado un uso extenso en la investigación y la academia debido a su capacidad para recrear escenarios complejos.

Modelo de dron Crazyflie

En Webots, existe un modelo inicial del dron Crazyflie 2.1 (ver Figura32), desarrollado y publicado en el repositorio oficial de Bitcraze [35]. Aunque el modelo permite una simulación básica del comportamiento dinámico del dron, su desarrollo aún se encuentra en etapas tempranas, como se especifica en la documentación de Bitcraze. Por ello, se llevó a cabo una evaluación de los archivos disponibles para determinar su adecuación en las herramientas de control que usa este proyecto.

Tras la revisión del modelo del dron en Webots, se observó que, aunque ofrece una aproximación razonable al comportamiento real del Crazyflie 2.1, su control no está basado en la librería cflib en Python, sino en archivos en C que emplean un método de control diferente. Esto representa una limitación ya que el proyecto actual se basa en la librería cflib

Figura 32. Simulación de modelo Crazyflie 2.1 en Webots.



como núcleo de su estructura de control, por lo que el método de control en Webots no es compatible con la metodología del proyecto.

Adicionalmente, el simulador no ofrece un modelo específico de la placa de expansión Flow Deck, esencial para la navegación y estabilidad del Crazyflie en aplicaciones prácticas. En su lugar, se emplea un modelo genérico de cámara en Webots, el cual es importado en los archivos de control en C pero no se integra como un componente del sistema de control. Esto implica que el simulador actualmente no cuenta con una representación adecuada de la placa Flow Deck para simular sus funcionalidades en el dron.

Debido a estas limitaciones, se decidió no utilizar Webots como herramienta de simulación en este proyecto. Sin embargo, en el futuro, si el modelo de Crazyflie 2.1 en Webots es ampliado para incluir un control compatible con cflib y una representación más precisa de la placa Flow Deck, podría ser considerado como una opción viable para el desarrollo y prueba de algoritmos en este contexto.

8.5. Conclusión

En este capítulo se detalló el desarrollo, implementación y validación de herramientas de software para el control individual del dron Crazyflie 2.1. Estas herramientas facilitaron la ejecución de comandos básicos de vuelo, lectura de variables, configuración de parámetros y seguimiento de trayectorias. Además, se presentó la implementación de la fusión de sensores mediante el uso de datos de la placa Flow Deck y el sistema de captura de movimiento Robotat, logrando mejorar la precisión del dron durante el vuelo.

El capítulo permitió observar tanto las capacidades como las limitaciones del sistema, especialmente en lo referente a la sensibilidad de la placa Flow Deck a las condiciones del entorno. Las soluciones propuestas, como la adaptación del entorno y la integración del sistema MoCap, evidenciaron mejoras significativas en la estabilidad y el control del dron, estableciendo una base sólida para experimentos futuros.

En el siguiente capítulo, se aprovecharán estas herramientas para desarrollar guías de laboratorio destinadas a cursos de sistemas de control y robótica.

CAPÍTULO 9

Desarrollo de guías de laboratorio para cursos de sistemas de control y robótica

9.1. Guía de laboratorio 1: análisis de control de altura del dron Crazyflie

Este laboratorio se posiciona como una práctica útil dentro del curso de Sistemas de Control 1, brindando a los estudiantes una experiencia integral que conecta los principios teóricos del control con aplicaciones prácticas en un sistema real. Utilizando el dron Crazyflie 2.1 como planta experimental, los estudiantes tendrán la oportunidad de aplicar conceptos de ajuste de controladores PID, evaluar el impacto de diferentes configuraciones y comprender las complejidades inherentes a los sistemas dinámicos reales.

9.1.1. Descripción

Este laboratorio utiliza el dron Crazyflie 2.1 como planta de estudio para implementar y ajustar controladores PID que permitan controlar su altura de vuelo. En primer lugar, los estudiantes realizarán simulaciones en Matlab para modelar el comportamiento del dron bajo diferentes condiciones. Luego, realizarán experimentos con el dron físico, ajustando los parámetros del controlador PID y utilizando el sistema de captura de movimiento Robotat para medir y analizar su desempeño real. Finalmente, realizarán una comparativa entre los resultados de las simulaciones y la experimentación real con el fin de observar las diferencias entre el comportamiento simulado del modelo simplificado y el comportamietno del dron físico.

9.1.2. Objetivos

Este laboratorio busca proporcionar a los estudiantes una comprensión práctica de los conceptos de control automático mediante el uso de controladores PID, aplicados al dron Crazyflie 2.1. A través de simulaciones y experimentación física, los estudiantes desarrollarán habilidades para analizar y ajustar sistemas dinámicos. Los objetivos específicos de esta práctica, así como su relevancia dentro del curso de Sistemas de Control, se detallan en la guía de laboratorio disponible en Anexos.

9.1.3. Material y equipo necesario

El laboratorio requiere un conjunto de herramientas y equipos esenciales, incluyendo el dron Crazyflie 2.1, el sistema Robotat y *software* como Matlab y Python. La lista completa y las configuraciones necesarias se encuentran descritas en la guía ubicada en Anexos.

9.1.4. Procedimiento

Este laboratorio se divide en tres partes principales: simulación con modelo simplificado en condiciones ideales, simulación con modelo simplificado en condiciones no ideales y experimentación física con el dron Crazyflie. A continuación, se describen las distintas secciones de guía de laboratorio:

Parte 1: simulación del modelo simplificado en condiciones ideales

En esta primera sección, se espera que los estudiantes implementen una simulación del modelo de vuelo vertical del dron Crazyflie 2.1 bajo condiciones ideales. El objetivo es observar cómo el controlador PID afecta la altura del dron en un entorno simulado donde no existen factores externos que interfieran, como la resistencia al aire o retrasos en los motores. Al final de esta sección, los estudiantes deberán obtener una serie de gráficos que representen la respuesta del sistema bajo diferentes combinaciones de parámetros PID y comprender el impacto de cada parámetro en el control de altura del dron en condiciones ideales.

Parte 2: simulación del modelo simplificado en condiciones no ideales

En la segunda parte, los estudiantes extenderán el modelo simplificado para incluir efectos de no idealidad, tales como la resistencia al aire y el retardo en la dinámica de los motores. Estos factores agregan realismo al modelo, permitiendo que la simulación refleje de manera más precisa las condiciones del mundo. Los gráficos generados en esta parte deben permitir a los estudiantes comparar y analizar las diferencias en el rendimiento del sistema bajo condiciones ideales y no ideales, entendiendo cómo los efectos del mundo real pueden alterar el desempeño de un controlador PID diseñado en simulación.

Parte 3: experimentación física con el dron Crazyflie

La última parte del laboratorio consiste en trasladar los parámetros de control utilizados en la simulación al dron Crazyflie físico, empleando el sistema Robotat para capturar y analizar la respuesta del dron en tiempo real. En esta sección se busca obtener datos experimentales que reflejen el comportamiento real del dron bajo diferentes configuraciones de PID. Al finalizar, los estudiantes tendrán una visión integral de cómo los parámetros del controlador PID afectan el desempeño del dron en simulaciones y en experimentación física, reforzando la comprensión de las limitaciones y capacidades del modelo de control de altura en aplicaciones de la vida real.

9.1.5. Resultados esperados

Para cada sección del laboratorio, se obtienen gráficos que ilustran cómo el sistema responde ante diferentes valores del controlador PID. En cada caso se probaron combinaciones de valores de KP, KI y KD, y se observaron las distintas respuestas. Algunas presentaron un alto sobreimpulso, otras poca estabilización y algunas presentaron respuestas estables y controladas.

Figura 33. Resultados de la simulación del modelo simplificado de dron en condiciones ideales.

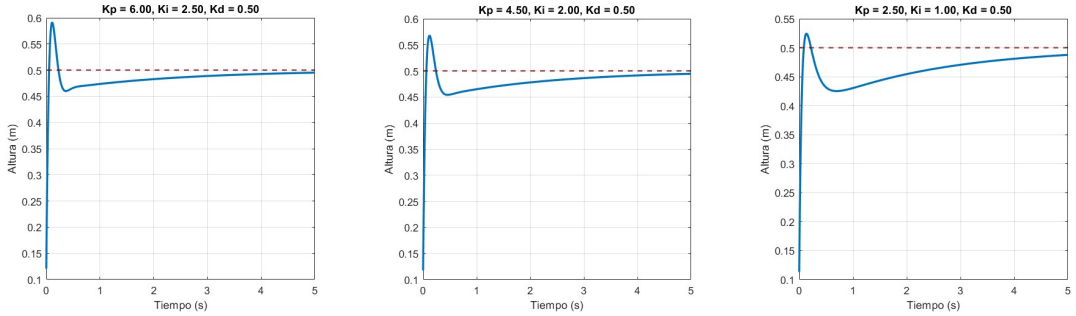


Figura 34. Resultados de la simulación del modelo simplificado de dron en condiciones no ideales.

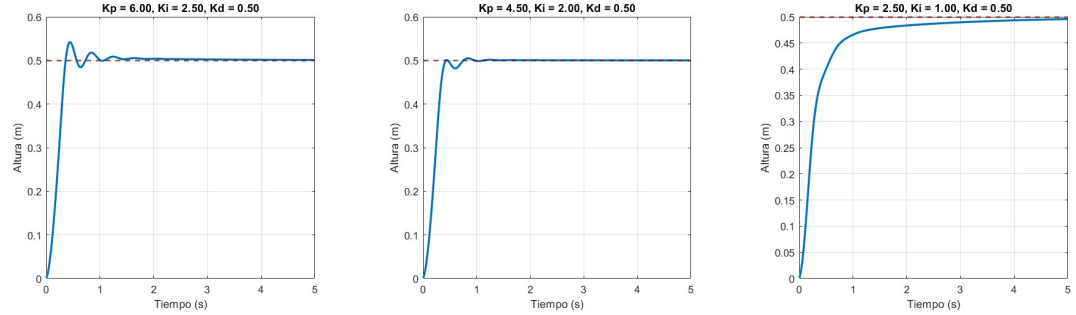
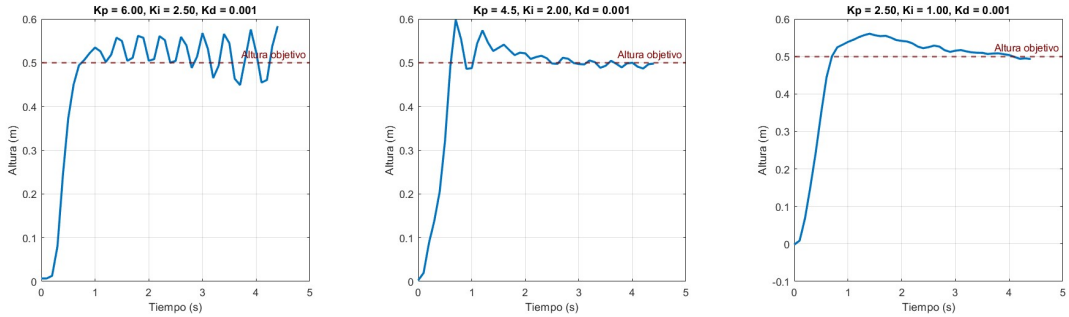


Figura 35. Resultados de la experimentación física con el dron Crazyflie.



Estos resultados evidencian cómo las condiciones específicas del sistema afectan la elección y ajuste del controlador PID. Las simulaciones con un modelo simplificado y las pruebas con el dron físico muestran comportamientos diferentes, lo que resalta las limitaciones de los modelos teóricos cuando se aplican a sistemas reales. Mientras que el modelo simulado ofrece un punto de partida para estudiar la dinámica del dron, la experimentación física permite observar complejidades adicionales, donde los ajustes en el controlador PID deben considerar factores no idealizados para lograr una estabilidad.

9.1.6. Observaciones importantes

Antes de iniciar el laboratorio, es fundamental que los estudiantes revisen el manual de usuario del Crazyflie 2.1 para familiarizarse con su funcionamiento, así como con la instalación y configuración de las herramientas de software necesarias. Este manual está incluido en los anexos de este documento y sirve como una referencia esencial para el desarrollo de las actividades.

También se debe tener en consideración las condiciones del entorno para que la placa de expansión Flow Deck funcione de forma óptima. Como se menciona en el primer capítulo, el entorno de experimentación debe tener la iluminación suficiente para que la sombra del dron no sea visible y afecte el funcionamiento del sensor de flujo óptico. Asimismo, el entorno de experimentación debe tener una superficie antireflectiva para no perjudicar la estimación de altura con el sensor infrarrojo de la placa Flow Deck.

Por último, durante los experimentos realizados se detectó un problema con el dron pues este comenzaba a disminuir su altura gradualmente durante el vuelo. El motivo de esto fue que la batería del dron había perdido capacidad, sin embargo, al sustituirla por una batería en buen estado solucionó dicho problema. En caso surga dicho problema durante el laboratorio, únicamente debe retirar la batería en uso y sustituirla por una nueva.

9.2. Guía de laboratorio 2: generación y seguimiento de trayectorias con Crazyflie 2.1

Este laboratorio se sitúa como una pieza clave dentro del curso de Robótica, proporcionando a los estudiantes una experiencia práctica que vincula los fundamentos teóricos de la navegación y planificación de trayectorias con aplicaciones en sistemas robóticos reales. A través del uso del dron Crazyflie 2.1 y el sistema Robotat, los estudiantes aplicarán conceptos esenciales de la robótica, como la generación y seguimiento de trayectorias en entornos con obstáculos, fortaleciendo sus habilidades en algoritmos de interpolación y control de movimiento autónomo.

9.2.1. Descripción

En este laboratorio los estudiantes se familiarizarán con el dron Crazyflie 2.1 y el sistema de captura de movimiento Robotat para generar y seguir trayectorias a través de rutas con obstáculos. Los estudiantes desarrollarán algoritmos de interpolación para crear rutas basadas en la posición de los obstáculos y utilizarán las funciones de alto nivel para controlar al dron durante el seguimiento de la trayectoria generada.

9.2.2. Objetivos

Este laboratorio tiene como propósito integrar conceptos teóricos de planificación y navegación con aplicaciones prácticas en sistemas robóticos reales. Dentro del curso de Robótica, permite a los estudiantes desarrollar competencias en la generación de trayectorias y su seguimiento utilizando el dron Crazyflie 2.1. Los detalles específicos de los objetivos están ampliados en la guía incluida en los anexos.

9.2.3. Material y equipo necesario

El laboratorio requiere un conjunto de herramientas y equipos esenciales, incluyendo el dron Crazyflie 2.1, el sistema Robotat, y *software* como Matlab y Python. La lista completa y las configuraciones necesarias se encuentran descritas en la guía ubicada en los anexos.

9.2.4. Procedimiento

Este laboratorio se divide en cuatro partes principales: preparación del entorno con las herramientas de *software*, captura de información de la pista con obstáculos, generación de la trayectoria y finalmente el vuelo del dron siguiendo la trayectoria. A continuación, se describen algunos detalles de las distintas secciones de guía de laboratorio:

Parte 1: configuración del entorno

Se añaden las carpetas con funciones específicas de Crazyflie y Robotat al entorno de Matlab, lo que facilita el acceso a las herramientas de *software* de control y captura de movimiento necesarias para el laboratorio.

Parte 2: captura de información

Se establece conexión con el sistema de captura de movimiento Robotat para obtener la pose (posición y orientación) de los puntos de despegue, aterrizaje y obstáculos. Esta información es clave para la generación de la trayectoria, asegurando que el dron pueda navegar evitando los obstáculos detectados.

Parte 3: generación de trayectoria

Se definen puntos clave de la trayectoria, incluyendo un punto inicial de despegue y uno final de aterrizaje. Alrededor de cada obstáculo, se generan puntos previos y posteriores para garantizar que el dron mantenga una distancia segura al atravesar la ruta.

Se emplea interpolación lineal (mediante interp1 en Matlab) para suavizar la trayectoria, facilitando así el seguimiento del dron sin movimientos bruscos o difíciles de realizar.

Utilizando gráficos en 3D se visualiza la trayectoria completa en el entorno de obstáculos. Esto permite observar cómo el dron debería navegar en el espacio definido, incluyendo el despegue, el paso alrededor de los obstáculos, y el aterrizaje.

Parte 4: seguimiento de la trayectoria

Una vez que la trayectoria ha sido generada y visualizada, se emplean funciones de alto nivel para que el dron siga la ruta de forma autónoma. Esta sección del código inicia la conexión con el dron, realiza un despegue a la altura deseada, y sigue la trayectoria a una velocidad especificada. Finalmente, se realiza el aterrizaje una vez completada la trayectoria.

9.2.5. Resultados esperados

En los resultados obtenidos, se muestran dos gráficas que representan las trayectorias generadas para la pista de obstáculos de la Figura 38 utilizando diferentes métodos de interpolación. La primera gráfica, mostrada en la Figura 36, utiliza interpolación lineal, donde los segmentos entre puntos clave son rectos, lo cual produce una trayectoria menos suave y con cambios abruptos en la dirección del movimiento. La segunda gráfica, correspondiente a la Figura 37, muestra la trayectoria generada con interpolación spline, un método que utiliza funciones polinómicas para conectar los puntos de control de manera continua y suave, minimizando cambios bruscos en la dirección y asegurando una transición fluida tanto en

posición como en la derivada de la trayectoria. Al comparar ambas gráficas, se observa claramente que la interpolación spline ofrece una transición más suave, eliminando los cambios bruscos y proporcionando una trayectoria más natural para el seguimiento del dron.

Figura 36. Resultado de trayectoria generada con método de interpolación *linear*.

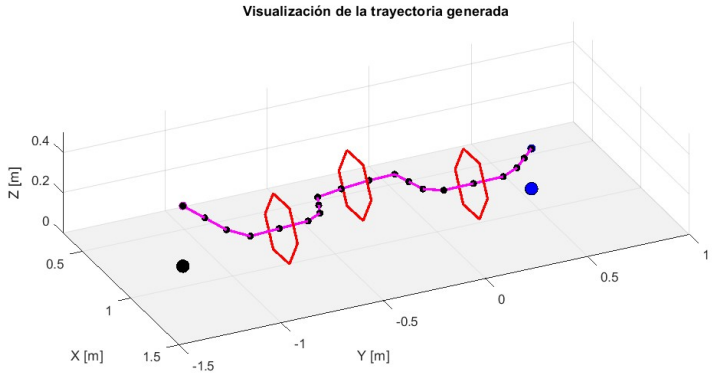
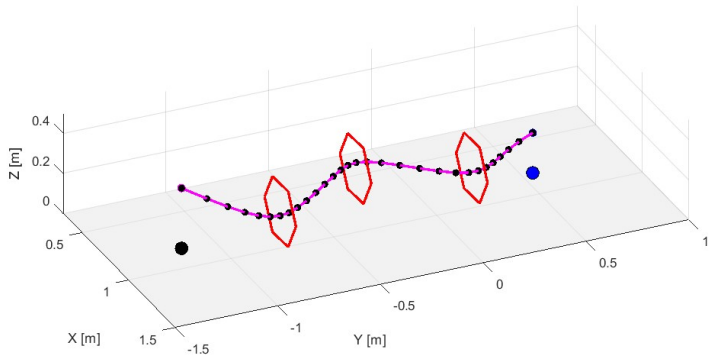


Figura 37. Resultado de trayectoria generada con método de interpolación *spline*.



Debido a esta ventaja en la suavidad de la trayectoria, se seleccionó la interpolación spline como la mejor opción para el seguimiento con el dron Crazyflie. Esta suavidad permitió que el dron realizara el recorrido con mayor estabilidad y precisión, sin cambios repentinos que pudieran afectar su control. En la prueba de seguimiento, el dron logró completar la trayectoria spline de manera exitosa, manteniendo una buena precisión en su desplazamiento y siguiendo la ruta definida a través del entorno con obstáculos. Este resultado confirmó que la interpolación spline es adecuada para lograr un control más estable en aplicaciones de seguimiento de trayectorias.

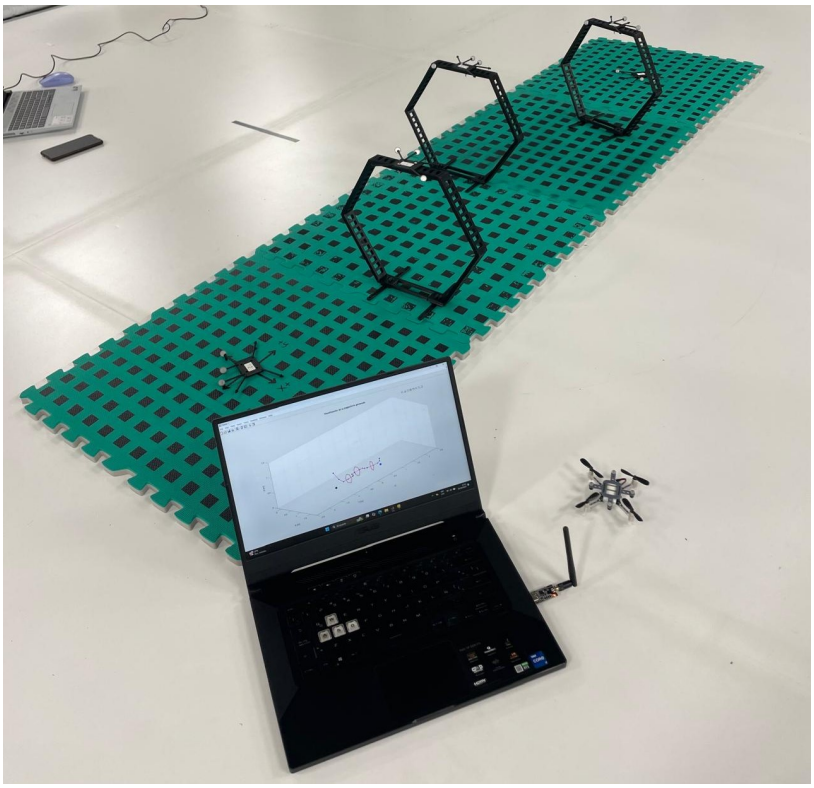
9.2.6. Observaciones importantes

Antes de iniciar el laboratorio, es fundamental que los estudiantes revisen el manual de usuario del Crazyflie 2.1 para familiarizarse con su funcionamiento, así como con la instalación y configuración de las herramientas de software necesarias. Este manual está incluido en los anexos de este documento y sirve como una referencia esencial para el desarrollo de las actividades.

También se debe tener en consideración las condiciones del entorno para que la placa de expansión Flow Deck funcione de forma óptima. Como se menciona en el primer capítulo, el entorno de experimentación debe tener la iluminación suficiente para que la sombra del dron no sea visible y afecte el funcionamiento del sensor de flujo óptico. Asimismo, el entorno de experimentación debe tener una superficie antireflectiva para no perjudicar la estimación de altura con el sensor infrarrojo de la placa Flow Deck.

Por último, durante los experimentos realizados se detectó un problema con el dron, pues este comenzaba a disminuir su altura gradualmente durante el vuelo. El motivo de esto fue que la batería del dron había perdido capacidad, sin embargo, al sustituirla por una batería en buen estado solucionó dicho problema. En caso surga dicho problema durante el laboratorio, únicamente debe retirar la batería en uso y sustituirla por una nueva.

Figura 38. Pista de obstáculos establecida para pruebas del laboratorio.



CAPÍTULO 10

Conclusiones

- Se realizó la integración exitosa de la placa de expansión Flow Deck en el dron Crazyflie 2.1, validando su funcionamiento mediante pruebas de vuelo que resaltaron la importancia de utilizar superficies opacas texturizadas con patrones y de baja reflectividad. Estas características garantizaron un óptimo desempeño de los sensores de la placa, permitiendo un vuelo y posicionamiento estable.
- Se desarrollaron herramientas de *software* empleando la biblioteca cflib en Python, adaptadas para su integración en Matlab. Esto permitió crear un entorno versátil para el control individual del dron Crazyflie 2.1, abarcando desde experimentos básicos de despegue hasta seguimientos de trayectorias complejas.
- Se desarrolló un manual de usuario para el dron Crazyflie 2.1 con la placa de expansión Flow Deck integrada, proporcionando una guía detallada sobre el ensamble, configuración y manejo del dron.
- Se desarrolló una guía de laboratorio diseñada para el curso de Sistemas de Control 1, enfocada en el ajuste y el análisis de controladores PID aplicados al dron Crazyflie 2.1 con la placa Flow Deck. Los estudiantes aprenderán cómo las constantes PID afectan el control de altura del dron en un entorno práctico.
- Se desarrolló una guía de laboratorio diseñada para el curso de Robótica 1, centrada en la generación y el seguimiento de trayectorias con el dron Crazyflie 2.1 y la placa Flow Deck en una pista con obstáculos. Los estudiantes desarrollarán habilidades de interpolación, planificación de trayectorias y navegación con el dron.

CAPÍTULO 11

Recomendaciones

- Se recomienda adquirir otras placas de expansión para ampliar las capacidades del dron Crazyflie. En particular, la placa de expansión MultiRanger expandiría las capacidades del dron para detectar obstáculos, permitiendo que evite obstáculos de manera autónnoma.
- Se recomienda instalar una red de protección alrededor de la mesa de trabajo del ecosistema Robotat con el fin de evitar accidentes que dañen a los usuarios, los drones o el equipo de laboratorio.
- Se sugiere modificar el protocolo de comunicación del ecosistema Robotat con el fin de implementar una fusión de sensores más eficiente, al habilitar la posibilidad de desarrollar una rutina de posicionamiento absoluto mediante una función *callback* basada en eventos.
- Se sugiere adquirir un conjunto amplio de repuestos para los drones Crazyflie, en especial, repuestos de hélices y motores, ya que suelen dañarse con facilidad durante los experimentos y, en las prácticas de laboratorio, podrían ser necesarios.

CAPÍTULO 12

Referencias

[1] F. Sanabria, «Diseño e implementación de una plataforma de pruebas para sistemas de control para el dron Crazyflie 2.0,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[2] C. Perafan, «Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[3] D. Otzoy, «Uso del cuadricóptero Crazyflie dentro del ecosistema Robotat para la generación de trayectorias y la evasión de obstáculos,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[4] J. Gordillo, «Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un enjambre de drones Crazyflie dentro de un ecosistema robótico basado en captura de movimiento,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[5] J. Avila, «Adaptación del sistema de drones Crazyswarm al ecosistema Robotat,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[6] B. Garrido, «Levantamiento de una plataforma de pruebas para sistemas multidrones con OptiTrack y Crazyswarm,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[7] C. Chadehumbe y J. Sjöberg, «Autonomous flight of the micro drone Crazyflie 2.1 through an obstacle course,» Independent thesis basic level (degree of Bachelor), Universidad Uppsala, 2020.

[8] Bitcraze, *Crazyflie 2.1*, Acceso: 12 de mayo de 2024, 2024. dirección: `https://www.bitcraze.io/products/crazyflie-2-1/`.

[9] Bitcraze, *Sistema de coordenadas de Crazyflie 2.1*, Acceso: 12 de mayo de 2024, 2024. dirección: `https://www.bitcraze.io/documentation/system/platform/cf2-coordinate-system`.

[10] Bitcraze, *Crazyradio*, Acceso: 12 de mayo de 2024, 2024. dirección: `https://www.bitcraze.io/products/crazyradio-pa/`.

[11] Bitcraze, *Flow Deck v2*, Acceso: 12 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/products/flow-deck-v2/.

[12] Bitcraze, *Estimación de estado con la placa de expansión Flow Deck*, Acceso: 12 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/#flowdeck-measurement-model.

[13] M. Persson, *Visual Odometry in Principle and Practice*, 1.ª ed. Linköping University, 2022, ISBN: 978-91-7929-168-6.

[14] Bitcraze, *Flujo óptico*, Acceso: 12 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/2017/11/optical-flow/.

[15] P. I. Inc., *PMW3901MB-TXQT: Optical Motion Tracking Chip*, Acceso: 23 de mayo de 2024, 2024. dirección: https://wiki.bitcraze.io/_media/projects:crazyflie2:expansionboards:pot0189-pmw3901mb-txqt-ds-r1.00-200317_20170331160807_public.pdf.

[16] S. Electronics, *VL53L1X: Time-of-Flight (ToF) ranging sensor based on ST's FlightSense technology*, Acceso: 23 de mayo de 2024, 2024. dirección: https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html#overview.

[17] Bitcraze, *The Crazyflie Python API explanation*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/user-guides/python_api/.

[18] Bitcraze, *Bootloader module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/bootloader/.

[19] Bitcraze, *CPX module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/cpx/.

[20] Bitcraze, *Crazyflie module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/crazyflie/.

[21] Bitcraze, *CRTP module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/crtp/.

[22] Bitcraze, *Drivers module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/drivers/.

[23] Bitcraze, *Localization module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/localization/.

[24] Bitcraze, *Positioning module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/positioning/.

[25] Bitcraze, *Utils module*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/utils/.

[26] Bitcraze, *CRTP - Communication with the Crazyflie*, Acceso: 25 de mayo de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/crtp/.

[27] Bitcraze, *Getting started with the Crazyflie 2.X*, Acceso: 04 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/.

[28] Bitcraze, *Crazyclient Installation Instructions*, Acceso: 04 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/installation/install/.

[29] Bitcraze, *Crazyflie Firmware Update*, Acceso: 04 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/userguides/userguide_client/#firmware-upgrade.

[30] Bitcraze, *Installing USB driver on Windows*, Acceso: 05 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/repository/crazyradio-firmware/master/building/usbwindows/.

[31] Bitcraze, *Propeller Balancing Tutorial*, Acceso: 05 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/tutorials/propeller-balancing/.

[32] Bitcraze, *Getting started with expansion decks*, Acceso: 05 de junio de 2024, 2024. dirección: https://www.bitcraze.io/documentation/tutorials/getting-started-with-expansion-decks/.

[33] Python, *Python 3.11.0*, Acceso: 05 de junio de 2024, 2024. dirección: https://www.python.org/downloads/release/python-3110/.

[34] Matlab, *Versions of Python Compatible with MATLAB Products by Release*, Acceso: 06 de junio de 2024, 2024. dirección: https://la.mathworks.com/support/requirements/python-compatibility.html.

[35] Bitcraze, *Crazyflie simulation repository*, Acceso: 08 de octubre de 2024, 2023. dirección: https://github.com/bitcraze/crazyflie-simulation/tree/main.

CAPÍTULO 13

Anexos

13.1. Repositorio del proyecto en GitHub

Puede acceder al repositorio del proyecto en GiHub por medio del siguiente enlace: `https://github.com/PabloCaal/Herramientas_de_software_crazyflie.git`

13.2. Manual de usuario para Crazyflie 2.1 con la placa de expansión Flow Deck incorporada

Este manual tiene como objetivo principal guiar al usuario en la exploración y uso del dron Crazyflie 2.1 equipado con la placa de expansión Flow Deck incorporada. Se busca que el usuario comprenda el funcionamiento básico, pueda validar el ensamble y aprenda a operar el dron desde su ordenador.

Puede acceder al documento del manual de usuario por medio del siguiente enlace: `https://uvggt.sharepoint.com/:b:/s/Test399/Ef-4VzTXSd5Oj4fbcY4QnhOBhfVLU9aJvIyRXwldkylW9A?e=4V2yKz`

13.2.1. Material y equipo

A continuación, se presenta el listado de los materiales y equipos necesarios para seguir los pasos de este manual:

- Dron Crazyflie 2.1 ensamblado
- Placa de expansión Flow Deck
- Dispositivo Crazyradio PA

- Ordenador con Windows 10/11

13.2.2. Consideraciones importantes

Antes de comenzar, asegúrese de que el equipo esté completo y en buen estado, ya que podría haber sufrido daños durante el uso previo, almacenamiento o transporte. Preste especial atención a las hélices del dron, debido que son propensas a sufrir daños, lo cual puede afectar el funcionamiento del dron. En caso de detectar algún defecto en el dron o sus componentes, por favor reportarlo inmediatamente con su catedrático o auxiliar de laboratorio.

13.2.3. Verificación del ensable y secuencia de inicio

Familiarización con el dron Crazyflie

Crazyflie es una plataforma robótica de desarrollo aéreo de código abierto desarrollada por Bitcraze. Se trata de un micro cuadricóptero por su tamaño compacto y sus cuatro motores con hélices. El control del dron se realiza mediante radiofrecuencia, usando un ordenador con sistema operativo Windows y el dispositivo Crazyradio como antena de comunicación.

En esta sección, verificará que el dron Crazyflie esté correctamente ensamblado y que todos sus componentes se encuentren en buenas condiciones. La Figura 39 presenta una imagen del dron ensamblado como referencia visual.

Figura 39. Dron Crazyflie 2.1 ensamblado.

Secuencia de inicio

Después de validar el ensamble del dron y el estado de sus componentes, proceda a encenderlo para verificar el funcionamiento del firmware. Para ello, siga estos pasos:

1. Verifique que la batería se encuentre conectada con la placa base del dron.

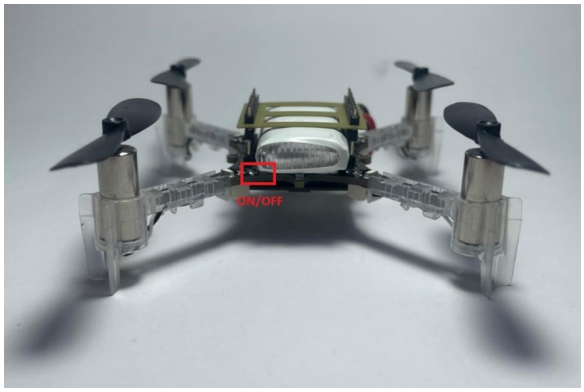
2. Coloque el dron sobre una superficie rígida en la que las hélices puedan circular libremente.

3. Presione el botón de encendido que se encuentra a un costado de la placa base del dron. En la Figura 40 se muestra la ubicación de dicho botón.

4. Observe la secuencia de inicio y consulte el resultado.

5. Apague el dron Crazyflie.

Figura 40. Botón de encendido y apagado en dron Crazyflie 2.1.



Al encenderse, el Crazyflie ejecutará una secuencia de inicio que verifica el estado del hardware y calibra los sensores, cuyo resultado se evidencia con una secuencia de encendido de los LED:

- **Encendido y todo está bien**: los LED azules (2 y 3) están completamente iluminados y el LED delantero derecho (1) parpadea en rojo dos veces por segundo.
- **Encendido y todo está bien, pero los sensores aún no están calibrados**: los LED azules (2 y 3) están completamente iluminados y el LED delantero derecho (1) parpadea en rojo con un intervalo de 2 segundos. Coloque el Crazyflie 2.1 en una superficie nivelada y manténgalo absolutamente quieto para calibrar.
- **Radio conectada**: el LED delantero izquierdo (4) parpadea en rojo y/o verde.
- **Batería baja**: el LED delantero derecho (1) está completamente iluminado en rojo. Es hora de aterrizar y recargar la batería.
- **Carga**: el LED azul trasero izquierdo (3) parpadea mientras que el LED azul trasero derecho (2) está encendido.
- **Modo de cargador de arranque**: los LED azules (2 y 3) en la parte posterior parpadean aproximadamente una vez por segundo.
- **Error de prueba**: el LED delantero derecho (1) parpadea repetidamente cinco pulsos rojos con una pausa más larga entre grupos.

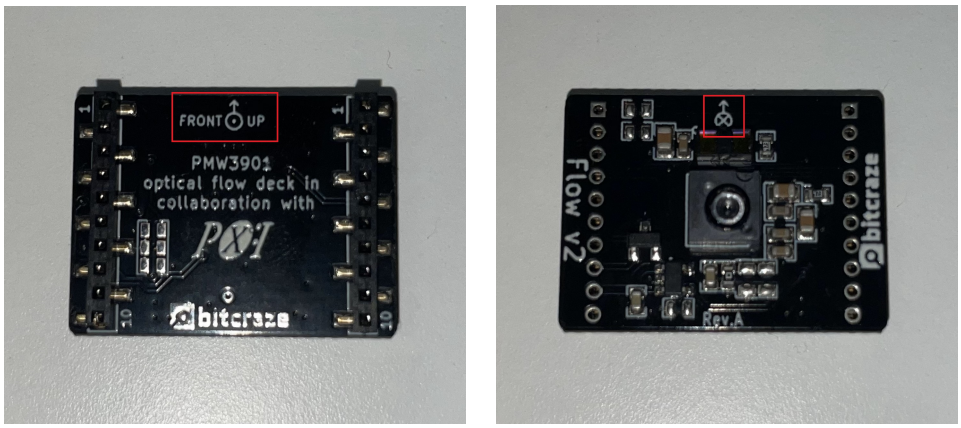
Verifique que el resultado obtenido sea el correspondiente a “Encendido y todo está bien”.

13.2.4. Instalación de la placa de expansión Flow Deck

Instalación

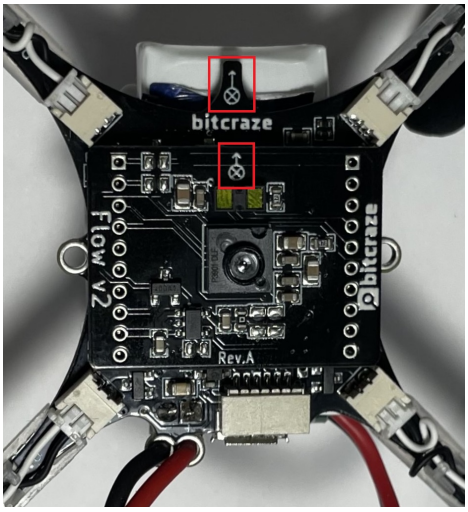
A continuación, instalará la placa de expansión Flow Deck sobre el dron Crazyflie 2.1. Es importante que antes de colocarlo verifique que el dron se encuentra apagado, debido a que colocar la placa con el dron encendido podría provocar daños irreversibles en el hardware de la placa.

Figura 41. Vista frontal y trasera de la placa de expansión Flow Deck.



Al instalar la placa en el dron, o si ya se encuentra instalada, verifique que la dirección de instalación es la correcta. Esto puede hacerlo al verificar que el símbolo de dirección (resaltado en la Figura 41) coincide con la dirección del dron Crazyflie, indicado con el mismo símbolo en su placa base. El resultado debería ser idéntico al mostrado en la Figura 42.

Figura 42. Placa Flow Deck instalada sobre el dron Crazyflie.



Una vez instalada la placa, habrá concluido con la validación del hardware.

13.2.5. Instalación de librería y dependencias

Instalación de Python 3.7 – 3.11

Para controlar el dron desde su ordenador, deberá tener instalada una versión compatible de Python (entre 3.7 y 3.11). Si no tiene Python instalado, siga estos pasos:

- Descargue la versión de Python adecuada para su sistema operativo desde la página oficial (se recomienda instalar la versión 3.11.0).
- Durante la instalación, asegúrese de seleccionar la opción .ªgregar Python a la ruta del sistema"para facilitar el uso desde la línea de comandos.
- Finalice la instalación y verifique que Python esté correctamente instalado ejecutando el siguiente comando en el terminal: python –version

Instalación de librería Crazyflie en Python

Una vez que Python esté instalado, puede instalar la librería de Crazyflie:

- Abra una ventana de comandos o terminal.
- Ejecute el siguiente comando para instalar la librería Crazyflie: pip install cflib

Esto instalará todas las dependencias necesarias para poder programar y controlar el dron desde Python.

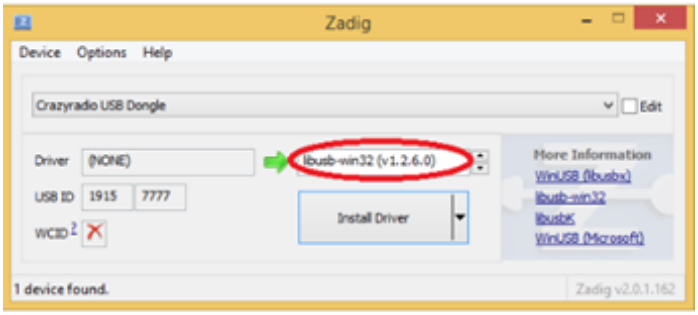
Instalación de controladores USB

- Descargue el programa Zadzig desde este enlace: Zadzig
- Conecte el dispositivo Crazyradio a un puerto USB de su ordenador.
- Abra Zadzig y verá la ventana mostrada en la Figura 43 con el dispositivo detectado automáticamente.
- Presione el botón Instalar controlador y espere a que finalice el proceso.

Descarga del código de prueba

Para verificar el correcto funcionamiento del dron y la placa Flow Deck, descargue el código de prueba de este enlace: Archivos de prueba

Figura 43. Instalación de controladores USB mediante Zadzig.



Ejecución de la prueba de conexión

Siga los siguientes pasos:

- Abra el código descargado en su entorno de desarrollo preferido.
- Conecte el dispositivo Crazyradio al puerto USB de su ordenador.
- Encienda el Crazyflie 2.1.
- Ejecute el código de prueba de conexión y observe el resultado.

Hasta este punto ha logrado configurar al dron Crazyflie correctamente para su uso y ha instalado las dependencias de software básicas.

13.2.6. Paquete de herramientas de software Crazyflie

Descarga de paquete de herramientas

Para controlar al dron Crazyflie de forma fluida y, desde el entorno de Matlab, se ha desarrollado un conjunto de funciones para interactuar con el Crazyflie de forma básica. Estas funciones permiten conectarse al dron, leer variables, modificar parámetros internos y ejecutar movimientos.

Enlace de descarga: Herramientas de software Crazyflie

Consideraciones importantes

- **Dependencias**: asegúrese de que el archivo crazyflie_python_commands.py esté en la misma carpeta que las funciones de MATLAB para que el código funcione correctamente. Si desea mover el archivo, actualice la ruta en crazyflie_connect.m.
- **Compatibilidad**: revise las versiones de MATLAB y Python para asegurar compatibilidad con cflib y evite conflictos entre las dependencias.

Uso del paquete de herramientas

Una vez descargado el paquete, descomprima los archivos en la carpeta de su elección y asegúrese de que están todos los archivos dentro de la misma carpeta para facilitar su uso en Matlab. A continuación, se explica cómo utilizar las funciones principales del paquete:

El paquete de herramientas proporciona una serie de funciones en MATLAB que permiten al usuario conectar y controlar el dron Crazyflie a través de scripts en Python. A continuación, se describe cada función:

Funciones de conexión y desconexión

- crazyflie_connect.m
  - **Descripción**: establece la conexión entre MATLAB y el Crazyflie, utilizando un identificador único.
  - **Parámetros**: drone_number (número entero de 1 a 12 que representa el dron a conectar).
  - **Retorno**: objeto SyncCrazyflie, representando la conexión.
  - **Errores**: error si drone_number no es válido o si falla la conexión.
- crazyflie_disconnect.m
  - **Descripción**: termina la conexión activa con el Crazyflie.
  - **Parámetros**: scf (objeto de conexión SyncCrazyflie).
  - **Retorno**: no retorna valor.

Funciones de lectura y configuración

- crazyflie_detect_flow_deck.m
  - **Descripción**: verifica la conexión del accesorio Flow Deck al Crazyflie.
  - **Parámetros**: scf (objeto SyncCrazyflie).
  - **Retorno**: ninguno.
- crazyflie_get_pid_values.m
  - **Descripción**: obtiene valores PID (proporcional, integral, derivativo) para los ejes X, Y y Z.
  - **Parámetros**: scf.
  - **Retorno**: estructura pid_values con valores (X, Y, Z) y valores PID [Kp, Ki, Kd].
- crazyflie_get_pid_x/y/z.m
  - **Descripción**: obtiene valores PID específicos para el eje (X, Y o Z).

- **Parámetros**: scf.
- **Retorno**: estructura con valores P, I y D para el eje.

- crazyflie_get_pose.m
  - **Descripción**: obtiene la pose actual del Crazyflie (posición y orientación).
  - **Parámetros**: scf.
  - **Retorno**: vector pose con coordenadas y orientación.
- crazyflie_set_pid_values.m
  - **Descripción**: configura valores PID en los tres ejes.
  - **Parámetros**: scf, p_gains, i_gains, d_gains.
  - **Retorno**: ninguno.
- crazyflie_set_pid_x/y/z.m
  - **Descripción**: configura valores PID en un eje específico (X, Y o Z).
  - **Parámetros**: scf, P, I, D.
  - **Retorno**: ninguno.
- crazyflie_set_pose.m
  - **Descripción**: establece una pose absoluta (posición y orientación).
  - **Parámetros**: scf, x, y, z, qx, qy, qz, qw (cuaterniones).
  - **Retorno**: ninguno.
- crazyflie_set_position.m
  - **Descripción**: configura la posición absoluta en el espacio.
  - **Parámetros**: scf, x, y, z.
  - **Retorno**: ninguno.

Funciones de control de movimiento

- crazyflie_takeoff.m
  - **Descripción**: inicia el despegue hasta una altura en un tiempo determinado.
  - **Parámetros**: scf, height (altura en metros, opcional), duration (duración en segundos, opcional).
  - **Retorno**: ninguno.
- crazyflie_land.m
  - **Descripción**: inicia el aterrizaje hasta una altura en un tiempo determinado.
  - **Parámetros**: scf, height (opcional), duration (opcional).
  - **Retorno**: ninguno.

- crazyflie_move_to_position.m
  - **Descripción**: mueve el Crazyflie a una posición específica.
  - **Parámetros**: scf, x, y, z, velocity.
  - **Retorno**: ninguno.
- crazyflie_goto_robotat.m
  - **Descripción**: mueve el Crazyflie en el espacio del sistema Robotat.
  - **Parámetros**: scf, x, y, z, velocity, tcp_obj, agent_id.
  - **Retorno**: ninguno.

13.2.7. Ejemplo básico de uso

A continuación, se muestra un ejemplo básico para el uso de estas funciones:

Figura 44. Ejemplo simple de flujo de trabajo en Matlab con funciones Crazyflie.

```
% Conectarse con un dron Crazyflie específico
dron_id = 1;
crazyflie_1 = crazyflie_connect(dron_id);

% Detectar si la placa Flow Deck está conectada
if crazyflie_detect_flow_deck(crazyflie_1)
    disp("Placa Flow Deck detectada.")
else
    disp("Placa Flow Deck no detectada.")
end

% Ejecturar el despegue con valores predeterminados
crazyflie_takeoff(crazyflie_1);

% Moverse a la posición (0.5, 0.0, 0.5) a una velocidad de 1 m/s
crazyflie_move_to_position(0.5, 0.0, 0.5, 1.0);

% Ejecutar el aterrizaje del dron con valores predeterminados
crazyflie_land(crazyflie_1);

% Finalizar la conexión
crazyflie_disconnect(crazyflie_1);
```

13.2.8. Referencias del manual de usuario

- Bitcraze. (2024). *Getting started with the Crazyflie 2.1.* Recuperado de `https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/`
- Bitcraze. (2024). *Getting started with expansion decks.* Recuperado de `https://www.bitcraze.io/documentation/tutorials/getting-started-with-expansion-decks/`
- Bitcraze. (2024). *Installing USB driver on Windows.* Recuperado de `https://www.bitcraze.io/documentation/repository/crazyradio-firmware/master/building/usbwindows/`

13.3. Laboratorio de sistemas de control: análisis de control de altura del dron Crazyflie 2.1

Este laboratorio utiliza el dron Crazyflie 2.1 como planta de estudio para implementar y ajustar controladores PID que permitan controlar su altura de vuelo. Los estudiantes realizarán una comparativa entre los resultados de las simulaciones y la experimentación real con el fin de observar las diferencias entre el comportamiento simulado del modelo simplificado y el comportamietno del dron físico.

Puede acceder a la guía de laboratorio por medio del siguiente enlace: `https://uvggt.sharepoint.com/:b:/s/Test399/EQ2bpNqLs-pJutDHlR2BFnoBrTgIYXakJv6iFK3mYgw6rg?e=ssH93G`

13.3.1. Objetivos

- Comprender el comportamiento dinámico del dron Crazyflie al emplear un controlador PID en la altura de vuelo.
- Observar el efecto de los parámetros K_P, K_I y K_D en la respuesta del sistema, tanto en simulaciones como en experimentos físicos.
- Comparar los resultados entre un modelo simplificado en distintas condiciones y el comportamiento real del dron, identificando diferencias y limitaciones del modelo.
- Desarrollar habilidades de implementación y ajuste de controladores PID en entornos de simulación en Matlab y experimentaciones prácticas.

13.3.2. Material y equipo necesario

A continuación, se presenta el listado de materiales y equipo necesarios para seguir los pasos de este manual:

- Dron Crazyflie 2.1 con la placa de expansión Flow Deck integrada.
- Dispositivo Crazyradio.
- Ordenador con Windows 10/11 con Matlab y Pyhton instalados.
- Paquete de herramientas de software descargado.
- Sistema de Captura de Movimiento del ecosistema Robotat del laboratorio CIT-116.

13.3.3. Procedimiento

Primera parte: simulación del modelo simplificado en condiciones ideales

En esta sección, deberá realizar una simulación en Matlab del comportamiento del modelo simplificado de un dron, controlado por un controlador PID en condiciones ideales. Deberá seguir los pasos presentados a continuación para lograr la simulación, obtener respuestas para distintas versiones del controlador y, con base en estas, responder un conjunto de preguntas.

Modelo simplificado del dron

Se considerará un modelo simplificado que describe la dinámica vertical de vuelo del dron. Asumimos una dinámica de segundo orden con la siguiente ecuación:

$$m \cdot \ddot{h} = u - mg$$

Donde:

- h es la altura del dron.
- u es la entrada de control (fuerza generada por los motores).
- g es la gravedad.
- m es la masa del dron.

Implementaremos un controlador PID para calcular el valor de entrada de control en función del error de altura:

$$u = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

Donde K_p, K_i y K_d son las ganancias del controlador.

Instrucciones

1. **Definición de parámetros:** defina variables para almacenar los valores de los parámetros de la simulación correspondientes a todas las cantidades involucradas, como las características del sistema o los parámetros del controlador.

   Parámetros mínimos para incluir:

   - **Parámetros del sistema**
     - Masa del dron
     - Gravedad

- **Parámetros del controlador PID**
  - K_p: ganancia proporcional
  - K_i: ganancia integral
  - K_d: ganancia derivativa
  - Variables para almacenar el error
- **Otros parámetros de simulación**
  - Altura objetivo
  - Tiempo de simulación
  - Variables de estado inicial
  - Vectores para almacenar resultados

1. **Bucle de simulación:** establezca una estructura cíclica como bucle de simulación, en ella actualizará la altura del dron en cada paso del tiempo, realizará el cálculo del error, la actualización del controlador PID y la simulación de la dinámica del dron.

   - En cada paso de tiempo, calcule el error de altura, así como los términos de control integral y derivativo.
   - Con los parámetros K_p, K_i y K_d, implemente la ecuación del controlador PID para la entrada del sistema.
   - Calcule la aceleración, velocidad y altura del dron en función del control y las condiciones ideales (sin resistencias adicionales).
   - Recuerde que con la ecuación del modelo puede determinar la aceleración de cada iteración y con ésta y el diferencial de tiempo, puede determinar velocidad y altura.
   - Guarde los valores obtenidos de altura, velocidad, señal de control, y error en los vectores definidos.

2. **Gráficas:** genere una figura con tres gráficos: altura, error, y señal de control a lo largo del tiempo. Observe la estabilidad de la altura deseada y la respuesta de control.

3. **Simulación variando las ganancias del controlador PID:** al llegar a este punto ya debe tener un algoritmo completo y funcional de simulación del modelo simplificado del dron (*pregunte a su catedrático de laboratorio para confirmar que sea así*).

   Utilice su algoritmo de simulación para obtener las **tres gráficas de resultados para cada combinación de ganancias del controlador PID** que se presentan en la siguiente tabla:

   Cuadro 1. Combinaciones de ganancias PID para los experimentos.

Prueba	K_p	K_i	K_d
1	2.50	0.00	0.00
2	2.50	1.00	0.05
3	4.50	2.00	0.05
4	6.00	2.50	0.05

   Luego de obtener las gráficas para cada combinación de ganancias, analice todos los resultados y responda:

- ¿Cómo afectan los parámetros K_p, K_i y K_d a la respuesta del sistema? Comente sobre la velocidad de respuesta, sobreimpulso y estabilidad.
- ¿Qué parámetros del controlador producen una respuesta más suave y cuál tiende a generar oscilaciones?
- ¿Cómo influye la integral del error en el comportamiento del dron durante el despegue?

Segunda parte: simulación de modelo simplificado en condiciones no ideales

En esta sección, nuevamente realizará una simulación del modelo simplificado del dron con la diferencia de que lo hará para un entorno con condiciones no ideales (más realistas). Esta sección extiende el modelo anterior para incluir condiciones de no idealidad (resistencia al aire y retardo en motores).

Instrucciones

1. **Definición de parámetros de no idealidad:**
   - Cree una copia de su algoritmo anterior y agregue los efectos de resistencia al aire (coeficiente c_{drag}) y retardo en la dinámica de los motores (τ) a su modelo.
   - Recuerde crear las variables necesarias para incluir dichos efectos.
     - **Resistencia al aire:**
       - Coeficiente c_{drag}: utilice un valor de 0.15.
     - **Retardo en la dinámica de los motores:**
       - Constante de tiempo de dinámica τ: utilice un valor de 0.1.
       - Defina los límites para la señal de control:
         - Límite máximo: $2.5 * \text{masa} * \text{gravedad}$.
         - Límite inferior: 0.
2. **Bucle de simulación:**
   - Adicione al cálculo de la señal de control, incluya la ecuación de la dinámica de los motores y saturación de la señal:

$$u = u_{prev} + \left(\frac{u_{\text{controlador}} - u_{prev}}{\tau} \right) \cdot dt$$

   - Luego utilice las funciones de MATLAB max y min para limitar la señal de control, lo que figurará la saturación de los motores.
   - Modifique el cálculo de la aceleración incluyendo resistencia al aire:

$$a = \frac{u - mg - c_{drag} v}{m}$$

     - a: es la aceleración del dron.
     - v: es la velocidad del dron.

- u: es la entrada de control (fuerza generada por los motores).
- g: es la gravedad.
- m: es la masa del dron.
- c_{drag}: es el coeficiente de resistencia al aire.

3. **Gráficas:** grafique nuevamente altura, error, y señal de control. Nuevamente realice la simulación con el listado de combinaciones de parámetros PID de la parte anterior y genere las gráficas **por cada combinación** y compare los resultados respondiendo las preguntas:
   - ¿Cómo afectan la resistencia del aire y la dinámica de los motores al desempeño del dron?
   - ¿Qué diferencias observa al comparar las condiciones ideales con las no ideales?

Tercera parte: experimentación física con el dron

Instrucciones

1. **Organización de archivos y herramientas de software:** revise la organización de sus archivos y verifique que pueda ejecutar todas las funciones, tanto del Robotat como de Crazyflie. Estos archivos se encuentran en carpetas propias, por ello, en Matlab se utiliza el comando addpath para añadir las carpetas al entorno de Matlab.
2. **Conectar con el Sistema Robotat:** inicializa la conexión con Robotat y configure el identificador del marcador del dron.
3. **Configuración del Timer para capturar los datos con Robotat:** esta sección se le otorgará para que únicamente tenga que ejecutarla, aún así no está de más que analice el código. Esta sección genera variables para utilizar una función Timer Callback que se utilizará para capturar la posición del dron durante los experimentos.
4. **Configuración del controlador PID:** defina las ganancias PID del controlador en el dron:
   - K_p: ganancia proporcional, utilice un valor inicial de 2.50.
   - K_i: ganancia integral, utilice un valor inicial de 0.50.
   - K_d: ganancia derivativa, utilice un valor inicial de 0.00.

Cuadro 2. Combinación de ganancias para experimentos físicos con dron Crazyflie.

Prueba	K_p	K_i	K_d
1	2.50	0.00	0.00
2	2.50	1.00	0.01
3	4.50	2.00	0.05
4	6.00	2.50	0.05

5. **Secuencia de Vuelo y Captura de Datos:** en esta sección deberá realizar los experimentos de vuelo con modificación en el controlador PID del eje Z del dron Crazyflie. Para ello, solo debe seguir las indicaciones a continuación:

- Coloque el dron Crazyflie sobre la plataforma del ecosistema Robotat en el punto indicado de despegue.
- Encienda el dron y conecte el dispositivo Crazyradio.
- Ejecute la secuencia del experimento dado (no modifique ninguna parte de este algoritmo).
- Al terminar el vuelo, analice y exporte las gráficas de los resultados de posición capturados con el sistema de captura de movimiento del Robotat.

Evaluación

A más tardar una semana después de la sesión (el día de inicio del próximo laboratorio), deberán subir un reporte a Canvas (**archivo.pdf**; *uno por pareja de laboratorio*).

El reporte deberá incluir:

1. **Identificación**: sus nombres, carnés, nombre del curso, sección de laboratorio (11, 12, 21 o 22), número y título del laboratorio, fecha.
2. **Resultados**: en esta sección deben incluir todas las funciones, gráficas y demás resultados obtenidos. También deben incluir las respuestas a las preguntas planteadas en la guía (breves, al punto). **Asegúrense de incluir todo lo requerido en esta guía. Se verificará que esté todo lo indicado en color azul.**

Nota: usen la numeración de la guía para organizar sus resultados. Por ejemplo, la figura con las gráficas pedida en el inciso 3 de la Segunda Parte debería estar bajo el numeral 3 de una sección titulada "Segunda Parte". Si lo prefieren, pueden numerar las secciones así: 2.3, 2.4, ..., 3.2, etc. *Los incisos de la guía que no requieran resultados (nada en azul) no necesitan aparecer en el reporte* (por ejemplo, los de la Primera Parte, o el inciso 1 de la Segunda Parte).

Asegúrense de numerar y titular todas las figuras/gráficas (ej.: Figura 1. Gráficas de entrada y salida del circuito, versus número de muestra).

Cuadro 3. Distribución de la evaluación del laboratorio.

Criterio	Porcentaje
Asistencia y trabajo en el laboratorio	20 %
Reporte	80 %

13.4. Laboratorio de robótica: generación y seguimiento de trayectorias a través de una ruta con obstáculos

En este laboratorio los estudiantes se familiarizarán con el dron Crazyflie 2.1 y el sistema de captura de movimiento Robotat para generar y seguir trayectorias a través de rutas con obstáculos. Los estudiantes desarrollarán algoritmos de interpolación para crear rutas basadas en la posición de los obstáculos y utilizarán las funciones de alto nivel para controlar al dron durante el seguimiento de la trayectoria generada.

Puede acceder a la guía de laboratorio por medio del siguiente enlace: `https://uvggt.sharepoint.com/:b:/s/Test399/ETe2_zEQtvtPsw0xRuCYuPMBWN934RPaWAr7UqGZkSkU6w?e=CGYCzc`

13.4.1. Objetivos

- Familiarizarse con el dron Crazyflie 2.1 y su uso dentro del ecosistema Robotat.
- Utilizar el sistema de captura de movimiento Robotat para obtener las poses de los obstáculos y utilizarlas para generar trayectorias mediante distintos métodos de interpolación.
- Utilizar funciones de alto nivel para ejecutar el seguimiento de las trayectorias generadas empleando al dron Crazyflie 2.1.

13.4.2. Material y equipo necesario

A continuación, se presenta el listado de materiales y equipo necesarios para seguir los pasos de este manual:

- Dron Crazyflie 2.1 con la placa de expansión Flow Deck integrada.
- Dispositivo Crazyradio.
- Ordenador con Windows 10/11 con Matlab y Pyhton instalados.
- Paquete de herramientas de *software* descargado.
- Sistema de Captura de Movimiento del ecosistema Robotat del laboratorio CIT-116.

13.4.3. Prelaboratorio

Previo a asistir a la sesión de laboratorio designada, deberá leer el manual de usuario del Crazyflie 2.1. Debe procurar que entiende el funcionamiento básico del dron e instalar en su ordenador todas las dependencias de *software* presentadas. Esto se debe a que el tiempo de laboratorio está dado exclusivamente para realizar experimentos con el controlador de posición del dron

13.4.4. Procedimiento

Dado el stock limitado (10) de drones Crazyflie 2.1 que se encuentran actualmente habilitados, esta práctica se trabajará en parejas. Se estará verificando que **ambos integrantes asistan y participen** en el desarrollo de la práctica.

En esta práctica usted tendrá como tarea principal desarrollar algoritmos de generación de trayectorias en un entorno dado utilizando información extraída mediante el sistema de captura de movimiento Robotat. Posteriormente, tendrá que utilizar funciones de alto nivel para controlar al dron Crazyflie 2.1 y realizar el seguimiento físico de las trayectorias previamente generadas. Estos drones poseen instalada una base de *markers* para detectar mediante el sistema de captura de movimiento, la cual también define el número/nombre del robot.

La idea de la práctica será utilizar a los agentes Crazyflie 2.1 en conjunto del sistema de captura de movimiento para controlar la pose del dron durante el seguimiento de la trayectoria generada a través de la ruta de obstáculos establecida. Todo esto mediante el uso de funciones de alto nivel desde MATLAB.

1. **Preste atención** a las indicaciones que dará el catedrático al inicio del laboratorio (en caso este llegue tarde, esta guía contiene la misma información).
2. **Descargue de Canvas** el archivo mt3006lab2.zip y extraiga sus contenidos dentro de una carpeta en una ubicación de su preferencia. Dentro de esta carpeta encontrará las rutinas para interactuar con el sistema de captura de movimiento, al igual que las funciones de alto nivel para controlar al Crazyflie. Adicionalmente, incluye el script base laboratorio2.m. Este último contiene:
    - Código básico de conexión (y desconexión) al Robotat para obtener información del sistema de captura de movimiento.
    - Código básico de conexión (y desconexión) al dron Crazyflie 2.1 seleccionado y ejemplo de funciones de control de vuelo de alto nivel.
    - Una sección en donde debe definirse la posición de los obstáculos, la idea es emplear el sistema de captura para obtener la pose (x, y, z, yaw) dentro de las variables. Además, deberá obtener la pose (x, y) de los puntos de despegue y aterrizaje, y con toda esta información extraída, generar trayectorias con distintos métodos de interpolación.
    - Finalmente, deberá generar una visualización tridimensional de los obstáculos, puntos de despegue, aterrizaje y la trayectoria atravesando correctamente la ruta.
    - Una sección en donde deberá ejecutar el seguimiento de las trayectorias generadas utilizando las funciones de alto nivel del Crazyflie.
3. **Valide el algoritmo de generación de trayectorias** modificando la posición de los obstáculos y realizando nuevamente la generación y seguimiento con el Crazyflie. Cuando esté satisfecho con sus resultados, muéstrelos al catedrático de laboratorio.

Para esta práctica se empleará la siguiente rúbrica de evaluación **individual**:

- **60 %:** por el correcto funcionamiento del algoritmo de generación de trayectoria.
- **40 %:** por el correcto funcionamiento del algoritmo de seguimiento de trayectoria.

El porcentaje de nota resultante se verá multiplicado por un factor entre 0 % y 100 % que consistirá en el reporte que estará evaluado el catedrático de laboratorio, de asistencia y trabajo **individual** durante todas las sesiones de trabajo de la práctica.

Nota: recuerde que entregas tardías representan penalización del 25 % por semana.

13.5. Funciones de control desarrolladas en Python

13.5.1. Función connect

Código 13.1. Función en Python para establecer la conexión con Crazyflie.

```
def connect(uri):
try:
scf = SyncCrazyflie(uri, cf=Crazyflie(rw_cache='./cache'))
scf.open_link()
print(f"Connection to Crazyflie established successfully.")
sys.stdout.flush()
return scf
except Exception as e:
if 'Cannot find a Crazyradio Dongle' in str(e):
print(f"Error: Crazyradio Dongle not found. Ensure the dongle is
    connected properly.")
elif 'Connection refused' in str(e):
print(f"Error: Connection to Crazyflie was refused. Check if the
    Crazyflie is powered on and in range.")
else:
print(f"General error occurred while trying to connect to Crazyflie.
    Error details: {str(e)}")
```

13.5.2. Función disconnect

Código 13.2. Función en Python para cerrar la conexión con Crazyflie.

```
def disconnect(scf):
try:
if scf:
scf.close_link()
print(f"Successfully disconnected from Crazyflie.")
else:
print(f"Error: Invalid SyncCrazyflie object. No connection to close.")
except Exception as e:
print(f"Error: An issue occurred while disconnecting from Crazyflie.
    Error details: {str(e)}")
```

13.5.3. Función get_pose

Código 13.3. Función en Python para obtener la pose actual del Crazyflie.

```
def get_pose(scf):
try:
# Set up the log configuration to get position and orientation data
pose_log_config = LogConfig(name='Pose', period_in_ms=100)
pose_log_config.add_variable('stateEstimate.x', 'float')
pose_log_config.add_variable('stateEstimate.y', 'float')
pose_log_config.add_variable('stateEstimate.z', 'float')
pose_log_config.add_variable('stateEstimate.roll', 'float')
pose_log_config.add_variable('stateEstimate.pitch', 'float')
pose_log_config.add_variable('stateEstimate.yaw', 'float')
pose = {'x': 0.0, 'y': 0.0, 'z': 0.0, 'roll': 0.0, 'pitch': 0.0, 'yaw':
    0.0}
new_data = Event()

def pose_callback(timestamp, data, logconf):
pose['x'] = data['stateEstimate.x']
pose['y'] = data['stateEstimate.y']
pose['z'] = data['stateEstimate.z']
pose['roll'] = data['stateEstimate.roll']
pose['pitch'] = data['stateEstimate.pitch']
pose['yaw'] = data['stateEstimate.yaw']
new_data.set()

pose_log_config.data_received_cb.add_callback(pose_callback)

try:
existing_configs = scf.cf.log.log_blocks
for config in existing_configs:
if config.name == 'Pose':
config.stop()
config.delete()

except AttributeError:
pass

scf.cf.log.add_config(pose_log_config)
pose_log_config.start()
new_data.wait()
pose_log_config.stop()
print(f"Pose retrieved successfully")
print(f"x: {pose['x']:.2f}, y: {pose['y']:.2f}, z: {pose['z']:.2f},
    roll: {pose['roll']:.2f}, pitch: {pose['pitch']:.2f}, yaw: {pose['
    yaw']:.2f}")
return [pose['x'], pose['y'], pose['z'], pose['roll'], pose['pitch'],
    pose['yaw']]

except Exception as e:
print(f"ERROR: An error occurred while retrieving the pose: {str(e)}")
```

13.5.4. Función get_pid_values

Código 13.4. Función en Python para obtener los valores de todos los PID de posición del Crazyflie.

```
def get_pid_values(scf):
try:
pid_values = {
  'X': [
  float(scf.cf.param.get_value('posCtlPid.xKp')),
  float(scf.cf.param.get_value('posCtlPid.xKi')),
  float(scf.cf.param.get_value('posCtlPid.xKd'))],
  'Y': [
  float(scf.cf.param.get_value('posCtlPid.yKp')),
  float(scf.cf.param.get_value('posCtlPid.yKi')),
  float(scf.cf.param.get_value('posCtlPid.yKd'))],
  'Z': [
  float(scf.cf.param.get_value('posCtlPid.zKp')),
  float(scf.cf.param.get_value('posCtlPid.zKi')),
  float(scf.cf.param.get_value('posCtlPid.zKd'))]
}
print("PID values for X axis: P={:.2f}, I={:.2f}, D={:.2f}".format(
    pid_values['X'][0], pid_values['X'][1], pid_values['X'][2]))
print("PID values for Y axis: P={:.2f}, I={:.2f}, D={:.2f}".format(
    pid_values['Y'][0],  pid_values['Y'][1], pid_values['Y'][2]))
print("PID values for Z axis: P={:.2f}, I={:.2f}, D={:.2f}".format(
    pid_values['Z'][0],  pid_values['Z'][1], pid_values['Z'][2]))
return pid_values

except Exception as e:
print(f"ERROR: An error occurred during the pid values lecture: {str(e)
    }")
```

13.5.5. Función get_pid_x

Código 13.5. Función en Python para obtener un PID de posición específico del Crazyflie.

```
def get_pid_x(scf):
try:
pid_x = {
  'P': float(scf.cf.param.get_value('posCtlPid.xKp')),
  'I': float(scf.cf.param.get_value('posCtlPid.xKi')),
  'D': float(scf.cf.param.get_value('posCtlPid.xKd'))
}
print("PID values for X axis: P = {:.2f}, I = {:.2f}, D = {:.2f}".
    format(pid_x['P'], pid_x['I'],  pid_x['D']))
return pid_x

except Exception as e:
print(f"ERROR: An error occurred while retrieving the PID values for X
    axis: {str(e)}")
```

13.5.6. Función detect_flow_deck

Código 13.6. Función en Python para detectar la placa Flow Deck en Crazyflie.

```
def detect_flow_deck(scf):
try:
flow_deck_detected = scf.cf.param.get_value('deck.bcFlow2')
if flow_deck_detected == '1':
print(f"Flow Deck detected successfully.")
return 1
else:
print(f"Flow Deck not detected. Please verify that it is installed
    properly.")
return 0

except Exception as e:
print(f"Error: An issue occurred while detecting the Flow Deck. Error
    details: {str(e)}")
```

13.5.7. Función set_position

Código 13.7. Función en Python para actualizar la posición del estimador del Crazyflie.

```
def set_position(scf, x, y, z):
try:
if not all(isinstance(coord, (int, float)) for coord in [x, y, z]):
print(f"ERROR: Input values invalids.")
scf.cf.extpos.send_extpos(x, y, z)
time.sleep(0.01)
print(f"Absolute position successfully set.")

except Exception as e:
print(f"ERROR: An error occurred during the position update: {str(e)}")
```

13.5.8. Función set_pid_x

Código 13.8. Función en Python para configurar a un PID de posición específico del Crazyflie.

```
def set_pid_x(scf, P, I, D):
try:
scf.cf.param.set_value('posCtlPid.xKp', P)
scf.cf.param.set_value('posCtlPid.xKi', I)
scf.cf.param.set_value('posCtlPid.xKd', D)
print(f"Successful PID modification.")

except Exception as e:
print(f"ERROR: An error occurred during the PID modification: {str(e)}"
    )
```

13.5.9. Función set_pid_values

Código 13.9. Función en Python para configurar todos los PID de posición del Crazyflie.

```
def set_pid_values(scf, p_gains, i_gains, d_gains):
try:
scf.cf.param.set_value('posCtlPid.xKp', p_gains['X'])
scf.cf.param.set_value('posCtlPid.xKi', i_gains['X'])
scf.cf.param.set_value('posCtlPid.xKd', d_gains['X'])
scf.cf.param.set_value('posCtlPid.yKp', p_gains['Y'])
scf.cf.param.set_value('posCtlPid.yKi', i_gains['Y'])
scf.cf.param.set_value('posCtlPid.yKd', d_gains['Y'])
scf.cf.param.set_value('posCtlPid.zKp', p_gains['Z'])
scf.cf.param.set_value('posCtlPid.zKi', i_gains['Z'])
scf.cf.param.set_value('posCtlPid.zKd', d_gains['Z'])
print(f"Successful PID modification.")

except Exception as e:
print(f"ERROR: An error occurred during the PID modification: {str(e)}"
    )
```

13.5.10. Función takeoff

Código 13.10. Función en Python para despegar al Crazyflie.

```
def takeoff(scf, height = 0.3, duration = 1.0):
try:
position = get_pose(scf)
current_z = position[2]
if current_z > 0.1:
print(f"The Crazyflie was already in the air.")
return 0
commander = HighLevelCommander(scf.cf)
commander.takeoff(absolute_height_m=height, duration_s=duration)
time.sleep(duration)
print(f"Takeoff completed successfully")

except Exception as e:
print(f"ERROR: An error occurred during takeoff: {str(e)}")
```

13.5.11. Función land

Código 13.11. Función en Python para aterrizar al Crazyflie.

```
def land(scf, height = 0.0, duration = 2.0):
try:
position = get_pose(scf)
current_z = position[2]
if current_z <= 0.1:
print(f"The Crazyflie was already on the ground.")
return 0
commander = HighLevelCommander(scf.cf)
commander.land(absolute_height_m=height, duration_s=duration)
time.sleep(duration)
commander.stop()
print(f"Landing completed successfully.")

except Exception as e:
print(f"ERROR: An error occurred during landing: {str(e)}")
```

13.5.12. Función move_to_position

Código 13.12. Función en Python para enviar a una posición específica al Crazyflie.

```
def move_to_position(scf, x, y, z, velocity = 1.0):
try:
commander = scf.cf.high_level_commander
current_position = get_pose(scf)
current_x, current_y, current_z = current_position[0], current_position
    [1], current_position[2]
distance = ((x - current_x)**2 + (y - current_y)**2 + (z - current_z)
    **2)**0.5
duration = distance / velocity
commander.go_to(x, y, z, yaw=0.0, duration_s=duration)
time.sleep(duration)
print(f"Position command completed successfully")

except Exception as e:
print(f"ERROR: An error occurred during moving to position: {str(e)}")
```