
Adaptación del sistema de drones Crazyswarm al ecosistema Robotat

Julio Andrés Avila García-Salas



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Adaptación del sistema de drones Crazyswarm al ecosistema
Robotat**

Trabajo de graduación presentado por Julio Andrés Avila García-Salas
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

(f) _____
Ing. Miguel Zea

Tribunal Examinador:

(f) _____
Ing. Miguel Zea

(f) _____
MSc. Carlos Esquit

(f) _____
Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, de de 2023.

El ámbito ~~aeroespacial~~ y los sistemas autónomos han sido por años mis áreas de interés, lo que me motivó a lograr una integración entre el laboratorio de robótica y los drones Crazyflie, para que futuras promociones de las carreras de Ingeniería Mecatrónica, Electrónica y Biomédica puedan utilizar dichos drones en sus cursos de la carrera. De esta forma, se estará revolucionando en el campo del control de drones y la robótica de enjambre en la Universidad del Valle de Guatemala y en el país.

Este proyecto fue asesorado por el Msc. Miguel Zea, a quien agradezco por darme la oportunidad de trabajar en el desarrollo del Robotat, por su asesoramiento y apoyo a lo largo del año y por motivarme a dedicarme a la investigación de sistemas autónomos. Junto a él, agradezco al PhD. Luis Rivera y al Ing. Kurt Kellner por su apoyo y retroalimentación en este proceso y por sus enseñanzas a lo largo de mi carrera.

Agradezco especialmente a mis padres y mi hermana por apoyar mi carrera todos estos años, motivarme a alcanzar la excelencia en todos los ámbitos y acompañarme en todos los aspectos de mi vida, llevándome a la culminación de mi carrera.

Prefacio	III
Lista de figuras	VI
Lista de cuadros	VII
Resumen	VIII
Abstract	IX
1. Introducción	1
2. Antecedentes	2
3. Justificación	5
4. Objetivos	6
5. Alcance	7
6. Marco teórico	8
7. Infraestructura de Crazyswarm	16
7.1. Ensamblaje y verificación de funcionamiento	16
7.1.1. Pruebas de vuelo en Android	18
7.2. Máquina virtual y cliente de Crazyflie	18
7.3. Pruebas iniciales de vuelo en Python	19
7.3.1. Códigos de control en Python	19
7.3.2. Prueba de vuelo inicial y verificación	20
7.4. Consideraciones físicas del modelo dinámico	20
7.5. Instalación de Crazyswarm y sus dependencias	22
7.5.1. MOCAP4ROS2	22
7.6. Inicialización de Crazyswarm y pruebas iniciales	22
7.6.1. Implementación de trayectorias	24

8. Integración de Crazyswarm con Robotat	28
8.1. Comunicación en Robotat	28
8.1.1. Creación del servidor Crazyswarm	29
8.2. Códigos de comunicación por TCP en Matlab	31
8.3. Diseño de trayectorias en Matlab	32
8.4. Integración de Crazyswarm con otros agentes autónomos	34
8.5. Pruebas Finales y validación	34
9. Conclusiones	38
10. Recomendaciones	39
11. Bibliografía	40
12. Anexos	42

Lista de figuras

1.	Ecosistema Robotat	3
2.	Investigación en ACT	4
3.	Investigación en McGill	4
4.	Crazyflie 2.1.	9
5.	Filtro de Kalman extendido.	10
6.	Arquitectura de Control.	10
7.	Cámara de captura OptiTrack.	13
8.	Estructura de control de Crazyswarm.	14
9.	Maquina Virtual de Crazyflie.	15
10.	Piezas del Crazyflie 2.1.	17
11.	Motor ensamblado en su base.	17
12.	Motores conectados a la placa.	18
13.	Crazyflie ensamblado.	18
14.	Aplicación de control en Android.	19
15.	Cliente de Bitcraze	19
16.	Descompensación de vuelo.	21
17.	Intefaz de RViz	22
18.	Espacio de trabajo en OptiTrack.	23
19.	Marcador reflectivo instalado sobre un Crazyflie	24
20.	Prueba con 3 drones	25
21.	Prueba con 3 drones	25
22.	Prueba con 3 drones	26
23.	Prueba con 3 drones	26
24.	Arquitectura de comunicaciones	29
25.	Servidor en funcionamiento	31
26.	Trayectoria parabólica	33
27.	Trayectoria experimental 1	35
28.	Validación de un Elipsoide inclinado	36
29.	Derivadas del movimiento en 3 ejes.	37

Lista de cuadros

1.	Estructura del paquete de datos	29
2.	Funciones de Matlab	32
3.	Resultados de posiciones	35
4.	Resultados de posiciones	36

Resumen

denominado

La  Universidad del Valle de Guatemala cuenta con un laboratorio de Robótica ~~Robotat~~ Robotat, el cual cuenta con un sistema de captura de movimiento y se trabajan líneas de investigación relacionadas con sistemas de control y robótica, además de ser el espacio donde se realizan las prácticas de laboratorio de los cursos de robótica. Está diseñado para realizar pruebas con agentes autónomos, entre ellos robots humanoides, robots móviles con ruedas, brazos robóticos y drones.

Entre los drones a utilizar se encuentran los drones Crazyflie 2.1, los cuales son de tamaño pequeño y pueden emplearse en conjunto para controlar enjambres de drones. Se realizó este control mediante el sistema Crazyswarm, el cual funciona a través de ROS2 en Linux. Con el objetivo de que este sistema de drones pueda utilizarse en prácticas de laboratorio y otras líneas de investigación en la Universidad, se adaptó la infraestructura de este sistema al ecosistema Robotat a través de los paquetes y funciones de ROS2. Esto se realizó mediante la obtención de paquetes de información generados por los algoritmos de control y el sistema de captura de movimiento del Robotat, logrando una integración entre los drones y el laboratorio.



Abstract ✓



Swarm robotics is one of the newest concepts known in mobile robotics and control systems, being one of the most important research areas nowadays. On the other hand, drones have become a popular device because of their practical design and controlling methods and are useful for multiple purposes, like research, photography and even toys.

The following thesis uses the Robotat environment at Universidad del Valle de Guatemala and its capability of using a group of multiple nanocopters for swarm controlling, being a great opportunity for the students to learn how to control these drones.

CAPÍTULO 1

Introducción

La robótica de enjambre es un concepto relativamente nuevo en el ámbito de la robótica y los sistemas de control, siendo uno de los temas de investigación más populares hoy en día. Por otro lado, los drones se han convertido en un dispositivo popular debido a su diseño y métodos de control prácticos y son útiles para múltiples propósitos, como investigación, fotografía hasta juguetes y entretenimiento.

La siguiente tesis utiliza el ecosistema Robotat en la Universidad del Valle de Guatemala y su capacidad de implementar un grupo de múltiples nanocópteros para control de enjambre, siendo una gran oportunidad para que los estudiantes aprendan a controlar dichos drones.

CAPÍTULO 2

Antecedentes

Los drones Crazyflie se han utilizado en líneas de investigación relacionadas con sistemas de control y robótica de enjambre debido a su extensa documentación y versatilidad para desarrollo de algoritmos. Para poder realizar un control eficiente de múltiples agentes, es necesario implementar un sistema que sea capaz de obtener y procesar información acerca del comportamiento de los agentes en tiempo real.

Crazyflie 2.1

Los drones a trabajar durante esta línea de investigación son los Crazyflies 2.1, los cuales son cuadricópteros de tamaño pequeño, que pueden ser controlados mediante un sistema de control de enjambre llamado Crazyswarm, el cual es utilizado mediante ROS en Ubuntu.

Anteriormente, se trabajó con estos drones en una línea de investigación pero no en forma de enjambre, sino con un drone únicamente. Se desarrollaron herramientas para el uso individual de estos drones. En primer lugar, se comprobó la compatibilidad de comunicación entre un drone y Python para el análisis de datos, estos se guardan en bruto y se recomendó graficarlos posteriormente mediante Matlab. Se desarrolló una interfaz gráfica para prácticas de laboratorio, en la cual, se pudo observar el comportamiento del drone ante las propiedades de control determinadas, en este caso, un controlador PID y sus respectivas constantes. Cabe destacar que para poder hacer esta herramienta fue necesario plantear un modelo matemático a través del identificador de sistemas de Matlab. Como resultado de estas herramientas, se llegaron a planificar 2 prácticas de laboratorio para los cursos de Sistemas de Control. [1]

Ecosistema Robotat

OJO: poner [ref] antes
del punto. Ej: Control[1].

El objetivo principal es utilizar este enjambre de drones en el ecosistema Robotat, el cual es un entorno tecnológico ubicado en el laboratorio de Robótica CIT-116 de la Universidad del Valle de Guatemala. [2] Se implementó un sistema de captura de movimiento mediante



Figura 1: Ecosistema Robotat.

[2]

un set de 6 cámaras de la marca OptiTrack, las cuales están conectadas a un switch que se comunica con una computadora mediante UDP, también fue implementado un protocolo MQTT para establecer una comunicación Wi-Fi. La información recopilada es procesada mediante un algoritmo de Python y luego es enviada a un router que genera una red Wi-Fi local, a la cual es posible conectar diferentes dispositivos y obtener datos como posiciones lineales y angulares. Las cámaras están colocadas de tal forma que rodean una tarima hecha de concreto y acero, como se muestra en la Figura 1.

Crazyswarm y ROS2

ROS es un sistema operativo para sistemas robóticos, el cual provee funciones especiales que facilitan el análisis y control en proyectos de robótica. Este suele trabajarse en sistemas basados en Linux como Ubuntu. A través de ROS2 puede trabajarse Crazyswarm, el cual es un sistema para controlar enjambres de drones Crazyflie 2.1, además se cuenta con un cliente llamado Bitcraze, el cual está basado en Python y permite la comunicación entre el servidor y los drones mediante telecomunicación por radiofrecuencia, este es un sistema independiente de ROS. El entorno Crazyswarm está siendo reemplazado por la versión 2, a la cual se le han estado trabajando mejoras.

El posicionamiento de los drones se ha realizado de múltiples maneras en distintas líneas de investigación, entre la documentación oficial de Crazyswarm puede encontrarse información sobre pruebas realizadas en sistemas de captura de movimiento, tales como OptiTrack. Para poder realizar experimentos con OptiTrack es necesario implementar un módulo con marcadores reflectivos y este debe estar diseñado a la medida para que el drone pueda ser detectado por las cámaras. Otros métodos utilizados para el posicionamiento de drones es mediante un dispositivo Kinect de la marca Microsoft, utilizado en consolas Xbox. [3]

Entre los centros de investigación que han utilizado Crazyswarm en forma de “enjambre”, está el Laboratorio ACT de la *University of Southern California*, donde se implementó un control de 49 Crazyflies con un sistema de captura de movimiento, creando rutinas de trayectorias como en la Figura 2. Por otro lado, la Escuela en Ciencias de la Computación de la Universidad de McGill ha orientado el control de Crazyswarm para un solo drone de



Figura 2: Investigación en ACT [ref].

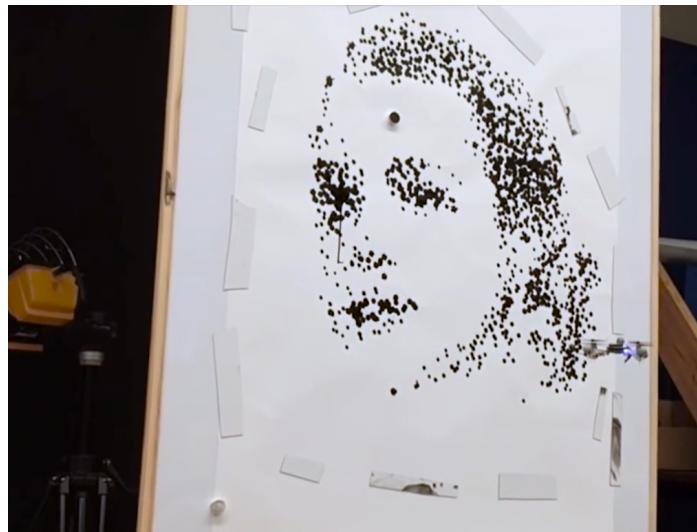


Figura 3: Investigación en McGill [ref].

modo que pueda usarse el posicionamiento para realizar pinturas, como se ve en la Figura 3.

CAPÍTULO 3

Justificación

Los drones son dispositivos que han ganado popularidad en los últimos años por una característica especial, ya que al modelarlos como un sistema dinámico se presenta una dinámica no lineal, pero pueden ser controlados de forma satisfactoria con un controlador lineal, siendo este un control relativamente sencillo. Esto convierte a los drones en un excelente método de aprendizaje para diseñar y analizar modelos y algoritmos de control. Debido a esto, trabajar físicamente con un drone o inclusive, con un conjunto de estos, es una alternativa para poner en práctica los aprendizajes adquiridos a lo largo de una carrera universitaria relacionada con electrónica y para realizar investigación relacionada con sistemas de control clásico, moderno y robótica de enjambre. Para poder realizar esto, es necesario contar con un medio que le provea a los algoritmos de control la información necesaria para trabajar, incluyendo, pero no limitándose, a las posiciones espaciales y orientaciones de los respectivos drones. Debido a que se cuenta con un entorno que es capaz de capturar esta información física, el camino a tomar es la creación de un intermediario que permita la comunicación entre el sistema de captura y los algoritmos de control para drones.

El ecosistema Robotat fue creado como un entorno tecnológico que sirviera como un lugar de aprendizaje de sistemas robóticos y para desarrollar líneas de investigación relacionadas con sistemas de control y robótica. Para culminar el desarrollo de este entorno, es necesario integrar a todos los agentes autónomos que formarán parte de él, siendo los drones Crazyflie una parte importante del ecosistema.

CAPÍTULO 4

Objetivos ✓

Objetivo general

Adaptar el sistema de control de drones Crazyswarm al ecosistema Robotat mediante ROS2 y sus respectivos paquetes y funciones.

Objetivos específicos

- Emplear la información recibida por las cámaras de captura OptiTrack en el ecosistema Robotat para definir algoritmos de control del sistema de drones.
- Crear algoritmos que conviertan información de origen TCP/UDP a radio frecuencia para establecer una comunicación directa con Crazyswarm.
- Levantar la infraestructura de Crazyswarm y desarrollar pruebas iniciales de vuelo.

CAPÍTULO 5

Alcance



La meta principal de este proyecto es permitir que cualquier estudiante tenga la oportunidad de utilizar este sistema de drones y desarrollar nuevos métodos de control y planificación, esto sin la necesidad de que tengan conocimiento sobre control, uso de Ubuntu o la dinámica de un drone. También se espera que la infraestructura de Crazyswarm esté lo suficientemente desarrollada para que futuros estudiantes puedan desarrollar aplicaciones.

Este trabajo se enfocará en una integración directa entre Crazyswarm y el ecosistema Robotat, de forma que las herramientas ya existentes en dicho espacio puedan aprovecharse y adaptarse a los drones.

CAPÍTULO 6

Marco teórico ✓

Crazyflie

Los drones Crazyflie son micro drones de dimensiones significativamente pequeñas (92x92x29 mm) y una masa de 27 g, cuentan con cuatro motores con un diseño simétrico (Figura 4).

El control directo de estos drones se realiza mediante radio frecuencia, específicamente a una frecuencia de 2.4 GHz, la cual pertenece a la banda de radios ISM. Cuenta con un amplificador de radio de 20 dB, con el cual se puede trabajar en un radio de hasta 1 km. Entre sus características eléctricas, se encuentra un microcontrolador STM32F405, el cual corre a 168 MHz. Este microcontrolador presenta una gran ventaja al trabajar a una velocidad alta ya que se puede garantizar un procesamiento de señales y una ejecución de algoritmos eficientes. Cuenta con un cargador LiPo con modos de 100 mA hasta 980 mA, el tiempo de vuelo estimado con la batería cargada es de 7 minutos, con un tiempo de carga de 40 minutos. Según Bitcraze, es recomendado que cualquier carga adicional que soporte el drone no supere los 15 g.

¿ [ref] ?

Control y estimación de estado

La IMU del Crazyflie 2.1 cuenta con un acelerómetro/giroscopio de 3 de ejes BMI080 y un sensor de presión BMP388. El control del drone se basa en un modelo  sistema dinámico. El modelado de sistemas dinámicos es utilizado en sistemas de control robótica para diseñar un controlador basado en la naturaleza del sistema, obteniendo un modelo matemático de este. Las variables de estado son valores determinados de un sistema dinámico, el cual puede ser un sistema físico, mecánico, eléctrico, etc. En el caso de un drone cuadricóptero, las variables de estado son los ángulos de rotación alrededor de sus ejes (*roll, pitch, yaw*) su posición en el espacio y cómo estas variables cambian en el tiempo (sus derivadas). Para el caso de Crazyflie, las variables fundamentales a controlar son dichos ángulos y la altitud de este. Se utilizan estimaciones de estado para convertir las señales de los sensores en variables de estado y así poder realizar el respectivo control.

¿ [ref] ?

añadir
a
glosario



Figura 4: Crazyflie 2.1.
[4]

Filtro de Kalman

Debido a que se utilizan métodos externos para determinar el estado de los drones, siendo un sistema de captura de movimiento para este caso, es necesario combinar las mediciones internas del drone con las externas por lo que se utiliza el filtro de Kalman como método para fusión de sensores y estimador/observador de estado. Los observadores de estado son algoritmos que utilizan el modelo matemático del sistema y mediciones del mismo para dar una estimación de las variables de estado. El filtro de Kalman es un observador de estado que toma en cuenta el ruido generado por perturbaciones en los actuadores o sensores del sistema, modelándolos como ruido blanco no correlacionado.[5] Además, puede utilizarse para fusión de sensores, permitiendo estimar el estado del sistema a través de múltiples mediciones.

Para pruebas de vuelo en lazo abierto puede utilizarse el filtro de Kalman Complementario, es liviano en cuanto a uso computacional y eficiente para posiciones angulares. Este filtro toma como valores de entrada las mediciones del giroscopio para los ángulos de Euler y el ToF (*Time of Flight*) para estimar la altura, sin embargo esta variación se recomienda únicamente para control manual. La otra variante es un filtro de Kalman Extendido como un filtro recursivo que estima el estado actual del drone basado en las mediciones internas y externas y el modelo matemático del mismo. Esta variación permite basar el estado tanto en las mediciones del giroscopio como en el FlowDeck de Bitcraze, el método de posicionamiento por LightHouse de Bitcraze o en un método de captura de movimiento especializado, de esta forma pueden obtenerse los ángulos de Euler, la posición espacial en los 3 ejes rectangulares y sus velocidades, como se ve en la Figura 5.[6]

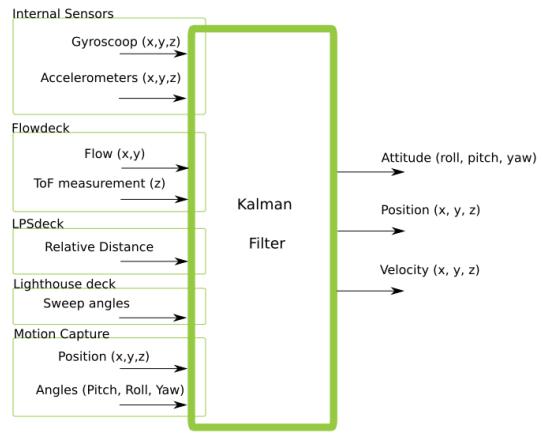


Figura 5: Filtro de Kalman Extendido.
[6]

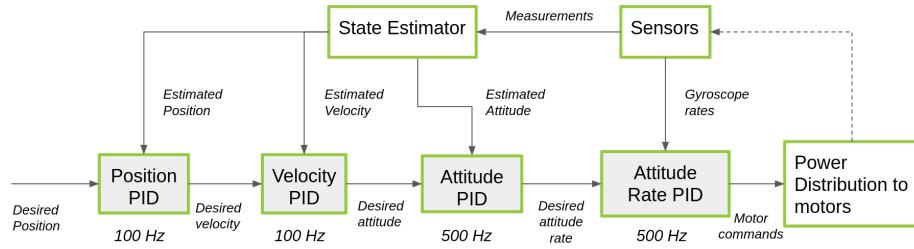


Figura 6: Arquitectura de Control.
[6]

Método de control

Los drones Crazyflie cuentan con 3 tipos de controladores: PID, INDI y Mellinger. El controlador INDI (Incremental Nonlinear Dynamic Inversion) fue diseñado especialmente para sistemas aereos y permite un control eficiente sin necesidad de contar con un modelo detallado del sistema, utilizando las mediciones de las aceleraciones para generar estimaciones y sustituir al modelo [7]. El método de control de los drones por defecto se realiza mediante una arquitectura de controladores PID en cascada, donde la entrada de control es la posición y la velocidad esperada, que luego determina ángulos de pitch y roll deseados, seguido de una velocidad angular y finalmente determina el thrust de los motores, como se ve en la Figura 6.

Sistemas de radio frecuencia

La comunicación por Radio Frecuencia (RF) es una tecnología para las telecomunicaciones que se da por un dispositivo que crea ondas electromagnéticas a altas frecuencias, donde las ondas se propagan en el espacio alcanzando a otro dispositivo receptor que procese la

información a la misma frecuencia. Según la frecuencia de las ondas, las comunicación puede clasificarse en

\itemsep 0em

- VLF: 3 kHz - 30 kHz ,
- LF: 30 kHz - 300 kHz ,
- HG: 3 MHz - 30 MHz ,
- VHF: 30 MHz - 300 MHz .
- UHF: 300 MHz - 3 GHz .
- SHF: 3 GHz - 30 GHz .
- EHF: 30 GHz - 300 GHz .

Las antenas convierten la información digital en señales eléctricas y luego en ondas electromagnéticas, o viceversa en el caso de la recepción de información. La ventaja que presenta este tipo de comunicación es que el dispositivo que procese tanto la información enviada como recibida puede ser un microcontrolador, siempre que este tenga la capacidad de conectarse a una antena y funcione a una frecuencia lo suficientemente alta como para procesar la señal de radio y digitalizar la información.

[ref] ?

Multithreading

Para alcanzar una ejecución de funciones eficiente, muchos microcontroladores emplean interrupciones para lograr que instrucciones específicas se ejecuten con prioridad mientras se realiza otra operación, logrando una noción de paralelismo en el funcionamiento. En el caso de computadoras este paralelismo se logra mediante el procesamiento de “hilos” de forma simultanea, de esta forma un algoritmo puede ejecutar distintas funciones paralelamente sin inconvenientes de bloqueo por la ejecución de otras instrucciones, aumentando el rendimiento del núcleo sin tener que alterar la frecuencia del reloj del procesador. [8]

Protocolos de redes

Para crear una comunicación eficiente se debe trabajar en conjunto con protocolos de comunicación por Redes, como TCP, el cual divide la información en paquetes y garantiza que estos lleguen correctamente a su destino. El protocolo TCP, que significa *Transmission Control Protocol*, es un protocolo de 3 vías donde primero se envía una solicitud inicial (*SYN*) desde el origen, luego el destino envía un paquete de confirmación (*SYN-ACK*) y finalmente el origen envía un último paquete (*ACK*) seguido de la información a transmitir. Este trabaja en conjunto con el protocolo IP, el cual se encarga únicamente asegurar una conexión entre el servidor de origen y destino mediante el sistema de direcciones de Internet. La ventaja de trabajar con este tipo de protocolos de comunicación es la versatilidad en la creación de herramientas con múltiples lenguajes de programación, como un servidor que permite la conexión de otros dispositivos conectados a la misma red en forma de clientes,

permitiendo una comunicación directa entre los clientes conectados a dicho servidor. El método más utilizado para la creación de servidores es Python, ya que cuenta con librerías especializadas para la comunicación TCP, además permite trabajar con *multi threading*, lo que eficientiza la comunicación al permitir que múltiples clientes pidan o envíen datos al servidor simultáneamente.[9]

El *User Datagram Protocol*, o UDP, es un protocolo de comunicación de Internet que, a diferencia del TCP, permite una comunicación rápida al no requerir que se establezca formalmente una conexión antes de iniciar con las transmisión de datos, sin embargo, esto también provoca que los paquetes puedan perderse en tránsito. [10]

Formato JSON

Para la estructura de los datos transferidos entre los distintos medios de comunicación, puede implementarse el formato *JavaScript Object Notation* (JSON), el cual es ligero computacionalmente y de fácil lectura y escritura. Adicionalmente, puede aplicarse a otros lenguajes de programación, como C, C++, Python, etc. En el caso particular de Matlab, la codificación de información en formato JSON puede realizarse mediante la creación de una estructura, con atributos nombrados como la estructura de datos que se desea manejar en un paquete, para posteriormente codificar dicha estructura a JSON y enviarlo mediante algún protocolo de comunicación especificado. Gracias a que los nombres de los atributos pueden ser especificados por el usuario, la codificación y decodificación de datos puede realizarse en distintos lenguajes de programación, permitiendo una comunicación más eficiente entre dispositivos.

¿*[ref]?*

Sistemas de captura de movimiento

La captura de movimiento es una tecnología que ha ganado popularidad los últimos años al emplearse en aplicaciones como animación, videojuegos o investigación de sistemas mecánicos. Este proceso se da mediante cámaras de luz infrarroja, la cual se apunta hacia el sistema o conjunto de sistemas a capturar, los cuales deben contar con algún material que refleje la luz. ~~En la Universidad del Valle de Guatemala~~ las cámaras de la marca Optitrack, las cuales son utilizadas en el Robotat en la Universidad del Valle de Guatemala (Figura 7), cuentan con un set de emisores de luz infrarroja y utiliza pequeñas bolas cubiertas de material reflejante, conocidas como marcadores. Este tipo de captura de movimiento es conocido como óptica pasiva, ya que los marcadores reflejan la luz en lugar de generarla. [2]

esta
correcta
la palabra?

Los marcadores pueden usarse de forma individual para rastrear únicamente posiciones, o bien pueden trabajarse en grupos de 3 con una forma específica para analizar también la orientación de los cuerpos. El rastreo de cuerpos rígidos se basa en la geometría de estos, ya que para que el sistema pueda diferenciar a cada cuerpo, los marcadores deben estar en posiciones distintas de forma permanente para que siempre lo detecte de la misma forma.



Figura 7: Cámara de captura OptiTrack.

[11]

Ecosistema Robotat

Para transferir los datos obtenidos por el sistema de captura en el Robotat, las cámaras OptiTrack están conectadas a un Switch que se comunica con un servidor mediante un protocolo UDP. La información es procesada y enviada mediante un servidor de Python que posteriormente se comunica con un router Wi-Fi, el cual permite la comunicación entre dispositivos externos y agentes autónomos en el Robotat. Esta comunicación entre el router y los dispositivos se logra mediante un protocolo TCP, de esta forma puede accederse directamente a la información mediante la decodificación de un documento de tipo JSON.

¿Tie? ?

Cliente de Crazyflie y ROS2

El *Robotics Operating System 2*, o ROS2 por sus siglas en inglés, es una plataforma complementaria al sistema operativo del ordenador que proporciona herramientas para desarrollo de sistemas robóticos. Esta plataforma funciona mediante nodos, los cuales son unidades independientes que envían y reciben información. Uno de los principales objetivos de ROS es permitir una comunicación flexible entre nodos. Esto funciona mediante el concepto de publicación y suscripción, donde un nodo puede suscribirse a otro y así recibirá los datos publicados, datos que posteriormente pueden ser procesados para alguna tarea.

ROS2 permite que se puedan trabajar plataformas ya desarrolladas, tales como Crazyswarm. Crazyswarm es la infraestructura necesaria para trabajar con conjuntos de drones Crazyflie, esta utiliza la estructura de los nodos en ROS para establecer una comunicación entre el sistema de captura de movimiento a utilizar y los drones. ROS funciona con nodos, creando un servidor de Crazyflie que establece la comunicación directa entre los drones y la interfaz utilizada, que en este caso es ROS2 junto con la API de Python como se ve en la Figura 8.

El control de los drones se divide en cuatro capas independientes. La capa física transmite los paquetes de información desde y hacia el drone, donde puede implementarse una antena de radiofrecuencia para esto. El *link* implementa los canales para los paquetes, abstrayendo el medio físico e implementando un canal de transmisión y otro de recepción entre el Crazyflie. El CRTP, acrónimo de *Crazy Realtime Protocol*, implementa la información del puerto y el canal para crear la ruta del paquete a varios subsistemas. Finalmente, los subsistemas

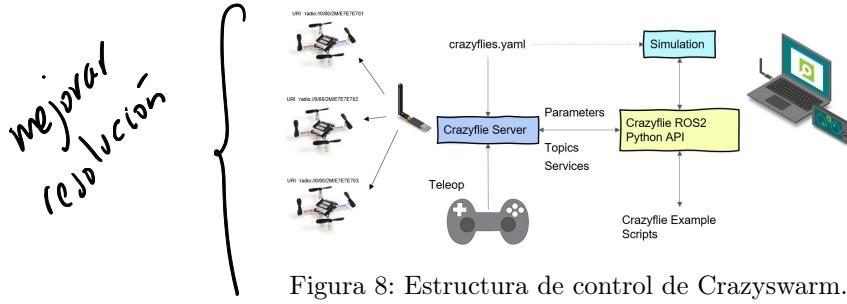


Figura 8: Estructura de control de Crazyswarm.

[12]

implementan las funcionalidades del Crazyflie que pueden ser controladas, habiendo sólo un puerto por subsistema.

El protocolo CRTP fue diseñado para permitir una priorización de paquetes y volver más eficiente la comunicación con el drone, permitiendo enviar una trayectoria mientras el drone se controla en tiempo real, siempre que el puerto de la trayectoria sea de mayor prioridad. Cada paquete CRTP contiene un puerto (4 bits), un canal (2 bits), y una carga de hasta 31 bytes. Para crear la conexión del protocolo se debe habilitar el *link* USB usando un paquete de control USB, luego el *link* Radio mantiene dos contadores de paquete para garantizar que no habrá pérdida de paquetes. Finalmente, el subsistema *log* mantiene un estado de todos los bloques *log* y continua enviando información aún si se pierde el *link*[13]

Otro protocolo de comunicación para Crazyflie es el CPX (*Crazyflie Packet eXchange*), el cual fue diseñado para funcionar a bajo nivel y que se empleen otros protocolos por encima. Su función es solucionar el problema de distribuir paquetes a través de múltiples microcontroladores ya que este problema no existía cuando se diseñó el protocolo CRTP, por lo que es un protocolo complementario. Debido a que se utiliza entre microcontroladores, puede enviarse de múltiples formas, como Wi-Fi/TCP, SPI o UART dependiendo de los microcontroladores a comunicar y los módulos adicionales que se implementen en el drone. Este protocolo permite el manejo de paquetes grandes, donde paquetes por debajo de 30 bytes se entregan en un bloque y paquetes más grandes pueden separarse en bloques y un identificador especifica cual es el bloque final.

Para establecer la comunicación de Crazyswarm, el servidor de Crazyflie se conecta con múltiples drones mediante una o más antenas de radiofrecuencia. Se cuenta con dos *backends* a elegir, el “cpp” basado en la capa más baja y el “cflib”, que funciona con Python a un nivel más alto. Este puede manejar aspectos de comunicación de bajo nivel, como recibir los parámetros del Crazyflie y convertirlos a parámetros de ROS2 para crear los parámetros de Crazyflie basado en la entrada.

Crazyflie cuenta con una máquina virtual, la cual funciona mediante Ubuntu y esta contiene todas las librerías para controlar los drones mediante una antena de radiofrecuencia USB. Estas librerías pueden visualizarse y editarse mediante Microsoft Visual Studio, que ya cuenta instalados los compiladores de C++ y Python. Desde Visual Studio, pueden iniciarse los códigos y, si la antena está conectada y un Crazyflie está encendido, pueden realizarse pruebas de movimiento y obtención de datos del drone. La máquina virtual también cuenta con el cliente de Crazyflie, desde el cual puede controlarse al Crazyflie mediante un control externo y es posible visualizar los ángulos de rotación del drone en tiempo real. En caso

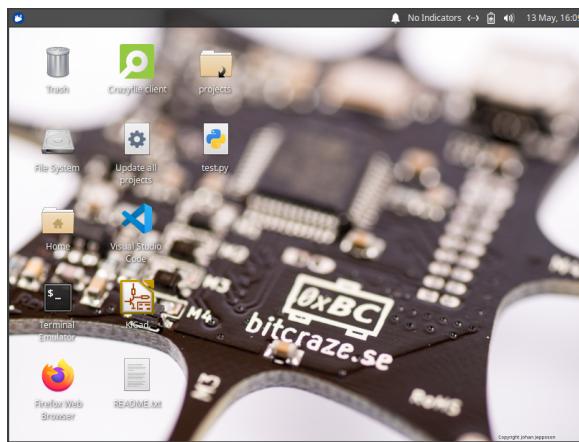


Figura 9: Maquina Virtual de Crazyflie.

de que no se cuente con un control externo, puede utilizarse la aplicación para Android de Crazyflie y controlar al drone mediante *Bluetooth*.

Para poder realizar las pruebas del Crazyflie es necesario configurar el puerto de la antena durante la instalación de la máquina virtual, lo cual puede realizarse fácilmente mediante Oracle. También se deben configurar todos los dispositivos que se planeen utilizar en la máquina virtual, de lo contrario no serán reconocidos. Una vez instalada la máquina virtual se debe abrir una terminal e instalar ciertas librerías mediante el gestor de paquetes pip3. Las librerías son:

`\i\em\exp 0cm`

- Matplotlib.
- Pandas.
- Canvas.
- pyinstaller.

Finalmente, es necesario actualizar las librerías de los controladores. Esto se logra mediante la opción incluida en el escritorio de la máquina virtual, como se ve en la Figura 9. Sin embargo, se presenta un problema al actualizar las librerías ya que luego de hacerlo deja de funcionar el cliente para controlar el drone con un control externo, por lo que si se desea trabajar con las librerías de Python y con el cliente se recomienda instalar dos máquinas virtuales y que una de estas no se actualice.

CAPÍTULO 7

Infraestructura de Crazyswarm ✓

La implementación involucra

██████████ de Crazyswarm █████ el funcionamiento de múltiples funciones y estructuras de datos en ROS2, lo cual representa una infraestructura de control y comunicaciones compleja. Antes de llevar a cabo el levantamiento de dicha infraestructura es necesario verificar el funcionamiento de cada uno de los Crazyflies a utilizar, para posteriormente incluirlos en Crazyswarm. A continuación se presentan las pruebas realizadas para comprender el funcionamiento de los drones por sí solos y verificar que estos no presenten problemas en su vuelo, seguido de las pruebas █████ la inicialización de Crazyswarm y su funcionamiento con múltiples Crazyflies. █████ realizadas durante

7.1. Ensamblaje y verificación de funcionamiento

El empaque del Crazyflie 2.1 cuenta con el drone desensamblado como se aprecia en la Figura 10. Incluye:

- Placa con microcontrolador.
- 6 bases para motores .
- 5 motores.
- 10 hélices para motores .
- 4 pines para anclar módulos .
- 1 cable micro USB.
- 1 batería de litio con pegamento .



Figura 10: Piezas del Crazyflie 2.1.



Figura 11: Motor ensamblado en su base.

- 1 PCB para sostener la batería.

Además, se debe contar con una antena **Crazydongle** para poder controlar a un Crazyflie o varios. *¿Cómo? Profundizar ya que esto toma relevancia más adelante.*

El primer paso es colocar los motores. Para esto, se recomienda enrollar los cables de cada motor para evitar que estos se enreden con las hélices, luego debe insertarse en los soportes y posicionar los cables de acuerdo a las guías de plástico de los soportes, quedando como en la Figura 11. Una vez que todos los motores estén montados en sus bases, deben colocarse en la placa principal. Estos se insertan hasta encajar en la placa, para posteriormente conectarlos

Como en log Figura 12

Finalmente, deben colocarse las hélices de forma cruzada para evitar una rotación natural alrededor del eje z. La forma en que se colocaron fue la siguiente: las hélices con la marca A fueron colocadas en la esquina inferior izquierda y en la esquina superior derecha, mientras que las hélices con la marca B se colocaron en las otras dos esquinas. La batería puede colocarse con los pines entre la placa y los **headers**. Luego del ensamblaje debe verse como en la Figura 13.

*¿Qué es? o añadir
al glosario*

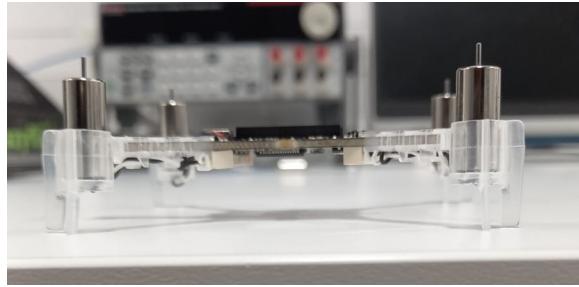


Figura 12: Motores conectados a la placa.



Figura 13: Crazyflie ensamblado.

7.1.1. Pruebas de vuelo en Android

Bitcraze cuenta con una aplicación compatible con Android que permite una conexión con un Crazyflie mediante Bluetooth, esta aplicación cuenta con dos pads que se asemejan a un *joystick* que controlan al drone. El pad izquierdo aumenta el thrust de los motores y el derecho controla los ángulos de roll y pitch (Figura 14). [14]

El principal problema que se presenta al volar el Crazyflie mediante la aplicación es que los drones presentan una descompensación en la distribución de thrust, ya que al elevar el Crazyflie este se ~~mueve~~ hacia la derecha naturalmente.
~~movía~~

7.2. Máquina virtual y cliente de Crazyflie

Bitcraze proporciona una máquina virtual en formato ISO descargable, esta puede utilizarse en cualquier programa que corra máquinas virtuales. Esta máquina funciona basada en Ubuntu y cuenta con las herramientas necesarias para realizar pruebas iniciales de vuelo y desarrollar códigos de control para un Crazyflie (Figura 15).

El cliente de Bitcraze permite modificar la dirección de los Crazyflies, esto es de vital



Figura 14: Aplicación de control en Android.

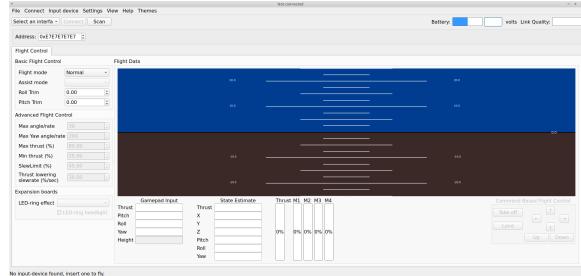


Figura 15: Cliente de Bitcraze

importancia ya que al momento de realizar un control de múltiples drones estos deben tener distintas direcciones. Este cliente también permite actualizar el *firmware* de cada uno de los Crazyflies, para asegurar que funcionen con los parámetros más recientes documentados por Bitcraze. Una vez que se ha configurado el *firmware* y la dirección deseada, pueden realizarse pruebas de estimación de estado y control con un mando externo.

glosario o definir / especificar

provistas

7.3. Pruebas iniciales de vuelo en Python

Los códigos de Python incluidos en la máquina virtual funcionan con las librerías ~~providas~~ por la misma.

7.3.1. Códigos de control en Python

*definir o colocar
en glosario*

El proyecto cuenta con una serie de códigos de control, estos códigos reciben atributos que suelen ser ángulos de rotación, tasas de rotación (ambos en grados), posiciones espaciales (en metros) o porcentaje de *thrust* en forma de *PWM*, el cual va de 0x0000 a 0xFFFF. La principal funcionalidad utilizada de Bitcraze consiste en una estructura de funciones llamada *commander*. Para las pruebas de vuelo, se hizo uso especial de la función *send_setpoint()*, la cual recibe como parámetros los ángulos de *roll*, *pitch*, *yawrate* y el *thrust* deseado. Dichos parámetros son concatenados en un objeto que corresponde a la *DATA* del paquete, que posteriormente es enviado al puerto determinado automáticamente por la función.

El primer código probado fue el de *ramp.py*, el cual enciende los motores con un *thrust* mínimo durante unos segundos. Este código fue modificado para aumentar el empuje de los motores y analizar cómo podía elevarse al Crazyflie.

7.3.2. Prueba de vuelo inicial y verificación

Para elevar el drone, se implementó un código basado en la primera (Ecuaciones 1 y 2) y segunda ley de Newton (Ecuaciones 3 a 5), donde se le dio un thrust suficiente para sacar al Crazyflie de su equilibrio estático y acelerarlo verticalmente hacia arriba.

$$\sum F_y = 0 , \quad (1)$$

$$T = mg , \quad (2)$$

$$\sum F_y = ma_y , \quad (3)$$

$$T - mg = ma_y , \quad (4)$$

$$T = m(g + a_y) . \quad (5)$$

De esta forma, se encontró que una forma experimental de encontrar la fuerza equivalente de los motores en función del thrust mediante la primera Ley de Newton, como se ve en la ~~E~~cación 6, esto claramente asumiendo una relación lineal.

$$F(PWM) = \frac{mg}{44500} PWM . \quad (6)$$

Primero, se aplicó un PWM de 47500, lo cual representa un 72 % del valor máximo permitido por los motores, este nivel en los motores se mantiene durante 350 ms hasta que alcanza una altura aproximada de medio metro. Luego de esto se llevó a un valor de 44500, aproximadamente el 68 % del valor máximo, este valor es suficiente como para que los motores generen un empuje equivalente al peso del drone. A pesar de la eficiencia de este método para elevar al drone, la descompensación mencionada en los motores no permite una estabilidad estática planar, ya que los Crazyflies tienden a moverse a la derecha y hacia el frente, esto puede explicarse debido a errores en la calibración del giroscopio, provocando que la IMU tome ángulos de pitch y roll distintos de cero como valores de equilibrio. 16

7.4. Consideraciones físicas del modelo dinámico

Al realizar las pruebas de vuelo con los códigos de Python, se notó una falta de estabilidad por parte del drone, ya que al dar un thrust constante el Crazyflie se eleva pero tiende a moverse hacia la derecha y adelante. Esto continuó sucediendo a pesar de establecer ángulos de roll y pitch iguales a cero, lo cual ~~mostró~~ que el control de los Crazyflies sin la utilización de un sistema de captura es en lazo abierto. El modelo dinámico de un drone se establece basado en las leyes de Newton, para el análisis de torques se puede notar que no



Figura 16: Descompensación de vuelo.

se toma en cuenta un torque provocado por el peso del drone alrededor de los ejes x y y , como se ve en la Ecuación 8.

$$\sum \tau = I\ddot{\eta} , \quad (7)$$

$$I\dot{v} + v \times (Iv) + \Gamma = \tau , \quad (8)$$

Donde I representa a la matriz de inercia, v es el vector de velocidades angulares, Γ el vector de fuerzas giroscópicas y τ los torques externos.[15]

Debido a esto, cualquier variación en cuanto a la ubicación del centro de masa altera la ecuación fundamental y por ende el modelo del sistema dinámico. Cada vez que se hacía un cambio de batería se debía procurar que esta quedara lo más centrada posible para que la alteración al modelo fuera lo más pequeña posible, lo cual representa una notable fuente de error para las pruebas en lazo abierto. Además se tuvo que tomar en cuenta que cada batería utilizada tenía una masa distinta, por lo que cada vez que se hacía un cambio de batería era necesario alterar tanto el *thrust* de arranque como el estable.

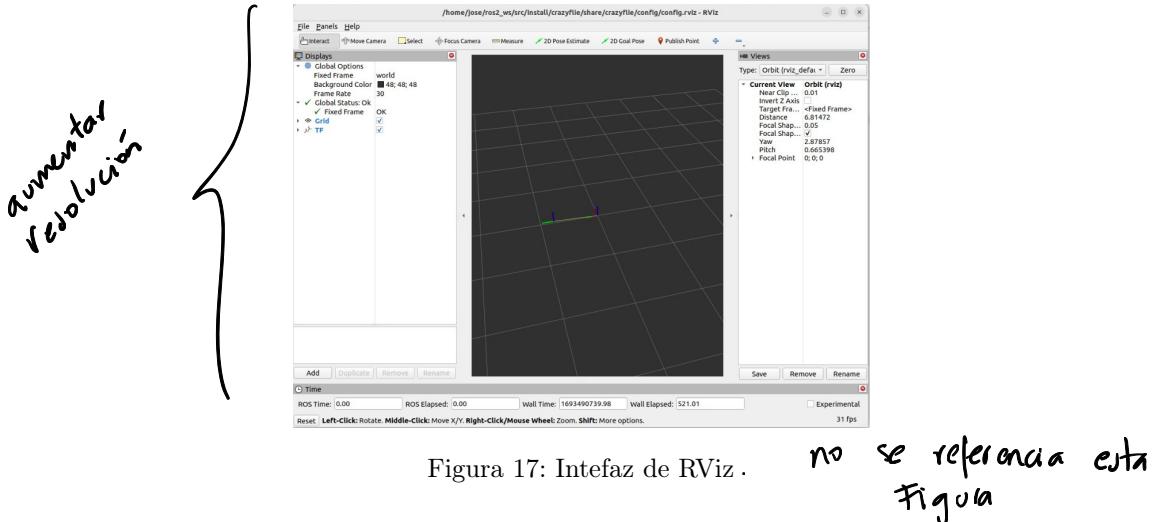


Figura 17: Interfaz de RViz.

7.5. Instalación de Crazyswarm y sus dependencias

ROS2 funciona en Ubuntu, se recomienda usar la versión más reciente y que esté sea LTS (requerido por ROS2), la versión de ROS2 utilizada fue la *Humble Hawksbill*. La instalación de ROS2 y Crazyswarm se realizó en una USB que permite iniciar Ubuntu en una computadora como sistema operativo nativo, lo que da flexibilidad de trabajar en distintos ordenadores pero a su vez limita las capacidades del sistema.

La instalación de Crazyswarm consiste en la preparación del espacio de trabajo de ROS2 y la inclusión de los repositorios de Github especificados por la documentación de Crazyswarm.

glosario

7.5.1. MOCAP4ROS2

Pueden utilizarse distintos métodos de captura de movimiento para obtener las posiciones y orientaciones de los drones a utilizar, independientemente del método utilizado, se requiere de un intermediario entre dicho sistema y el sistema de control, el cual está definido por Crazyswarm en este caso. MOCAP4ROS2 es una herramienta que recibe la transmisión de datos de un sistema de captura y los envía a un nodo específico, cabe resaltar que los datos transmitidos ya deben estar procesados, por lo que se requiere de otro ordenador que procese los datos en bruto mediante un programa especializado.[16]

7.6. Inicialización de Crazyswarm y pruebas iniciales

Comandos de iniciación

Una vez que se haya instalado Crazyswarm exitosamente, se puede proceder a las pruebas de control. Los archivos de configuración deben ser modificados para definir las características del entorno en el que se trabajará, como la cantidad de drones a utilizar y las direcciones de cada uno, sus posiciones iniciales y el método de captura de movimiento.

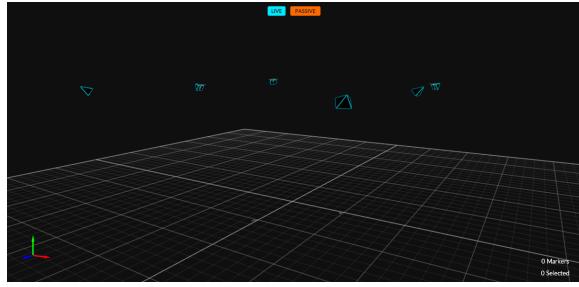


Figura 18: Espacio de trabajo en OptiTrack.

Luego de realizar la configuración de los drones, se deben inicializar los entornos de ROS2 en las carpetas raíz (*src*) donde se instalaron Crazyswarm y las dependencias de MOCAP4ROS2, posteriormente las librerías deben activarse y configurar la captura de datos de OptiTrack. Las librerías de MOCAP4ROS2 permiten seleccionar si se desea trabajar únicamente con *single markers* o con cuerpos rígidos que ya se hayan creado en Motive. Si la configuración y activación se hizo correctamente, se debería de apreciar la recepción de datos en una terminal de Ubuntu mediante el tópico de ROS2 que se encarga de procesar dichos datos, mostrando las coordenadas de los marcadores presentes en el Robotat.

Una vez que se inicia Crazyswarm, se inicializa MOCAP4ROS2 junto con el programa RViz, el cual funciona como una interfaz gráfica para visualizaciones 3D. Si tanto el sistema de captura de movimiento como el sistema operativo funcionan correctamente y los Crazyflies a utilizar están al alcance de las cámaras, RViz debería mostrar la posición y la orientación en tiempo real de los drones. Una forma de verificar que existe una comunicación entre Crazyswarm y los Crazyflies es verificar que todos los drones tengan una luz amarilla intermitente, evidenciando una transferencia de datos.

Calibración del sistema de captura

MOCAP4ROS2 requiere que los datos sean procesados anteriormente, por lo que debe utilizarse el programa Motive. Motive permite la calibración de cámaras y definir el origen (0,0,0) en el espacio a trabajar. Las cámaras se encienden mediante un comando por Python, luego de abrir Motive deben calibrarse las cámaras a través de la vara de calibración hasta tener una cantidad de muestras aceptable, luego se emplea la escuadra que define el origen. Si se hizo correctamente debe verse como en la Figura 18. Una vez que la calibración se hizo de manera exitosa, puede iniciarse la transmisión de datos a través de una red local especificada, en el caso del Robotat la dirección IP es la 192.168.50.200.

Pruebas físicas

Para llevar a cabo pruebas con los drones, fue necesario colocar un marcador del sistema de captura sobre cada uno. Como método inicial se colocaron los marcadores entre la batería y el PCB que la sostiene, por lo que se mantiene relativamente firme sobre el Crazyflie, como se observa en la Figura 19. Cabe destacar que este es un método poco eficiente ya que al no estar sujeto sobre el drone, este tiende a moverse debido a que las vibraciones de

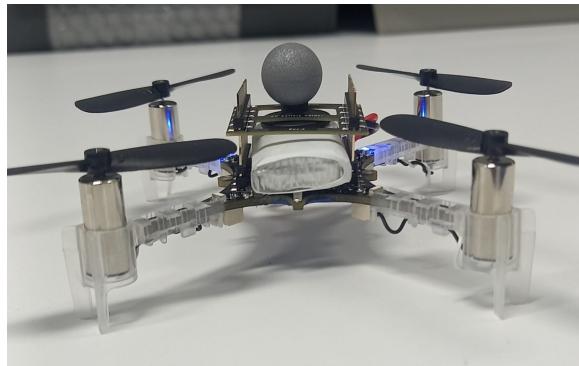


Figura 19: Marcador reflectivo instalado sobre un Crazyflie

los motores generan de igual forma una vibración en el marcador, provocando que los pines que sostienen la PCB sobre la batería comiencen a moverse al punto en que el marcador se afloja, esto a su vez induce aún más vibraciones en el drone, afectando su rendimiento.

Luego de inicializar Crazyswarm en ROS, pueden ejecutarse códigos de prueba. El código “hello-world.py” toma a un Crazyflie al azar de los que se encuentren conectados y dentro del sistema de captura y lo eleva 1 metro durante unos segundos. Se realizó una prueba de estabilidad al perturbar al drone una vez que este alcanzaba la altura especificada y este corregía el error inmediatamente con un *Overshoot* o suficientemente bajo como para no ser notable a simple vista, este rechazo a perturbaciones funciona con la posición en los 3 ejes.

Luego de analizar las funciones provistas por Crazyswarm, se realizó una prueba con la función *goto*, a la cual se le envía un vector de posición absoluta con respecto al sistema de captura de movimiento. La altura especificada en la función se alcanzó y mantuvo un estado de equilibrio en el tiempo establecido, también se le dio perturbaciones externas empujando al Crazyflie en todas las direcciones y aún así corrigió su posición, verificando el funcionamiento del controlador.

Una vez que se han verificado las pruebas de control con un solo drone, puede procederse a las pruebas con múltiples Crazyflies, se recomienda agregarlos uno por uno para asegurar que cada uno funcione correctamente. Cabe destacar que las pruebas con múltiples Crazyflies requieren que las *direcciones* ya hayan sido modificadas, la librería de Crazyswarm cuenta con una función que detecta a todos los Crazyflies encendidos cercanos a la antena y crea un vector que permite indexar a los drones y referirse a cada uno como la posición de dicho vector, de esta forma se puede controlar al Crazyflie número 1 refiriéndose a él como *cf[0]*. Luego de verificar el funcionamiento de múltiples drones logrando que estos se eleven a cierta altura durante unos segundos, puede procederse al diseño de rutinas.

7.6.1. Implementación de trayectorias

Las primeras pruebas con trayectorias se realizaron utilizando el controlador PID de los Crazyflies, donde la primera trayectoria de prueba fue un movimiento armónico simple. Los resultados pueden apreciarse ~~de~~ las Figuras 20 a 23.

Debido a que este es un controlador para estabilización y no para rastreo de referencias,

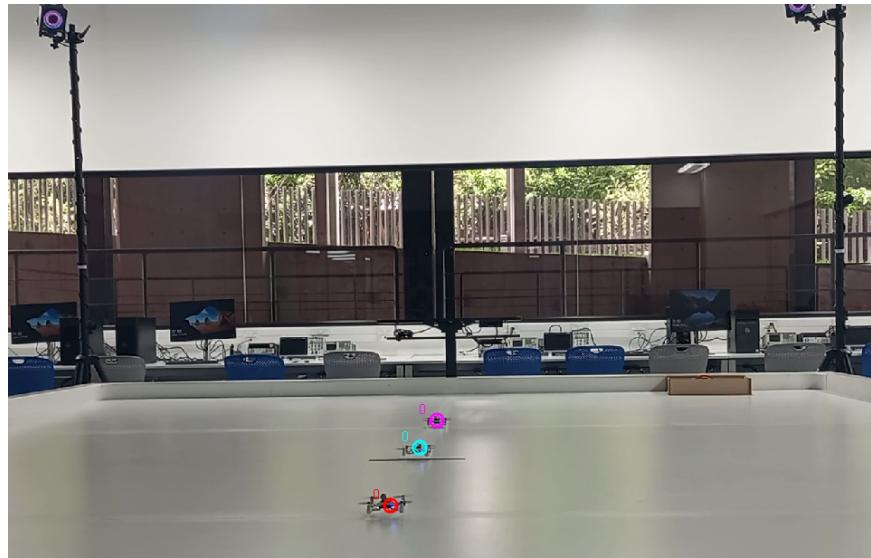


Figura 20: Prueba con 3 drones .

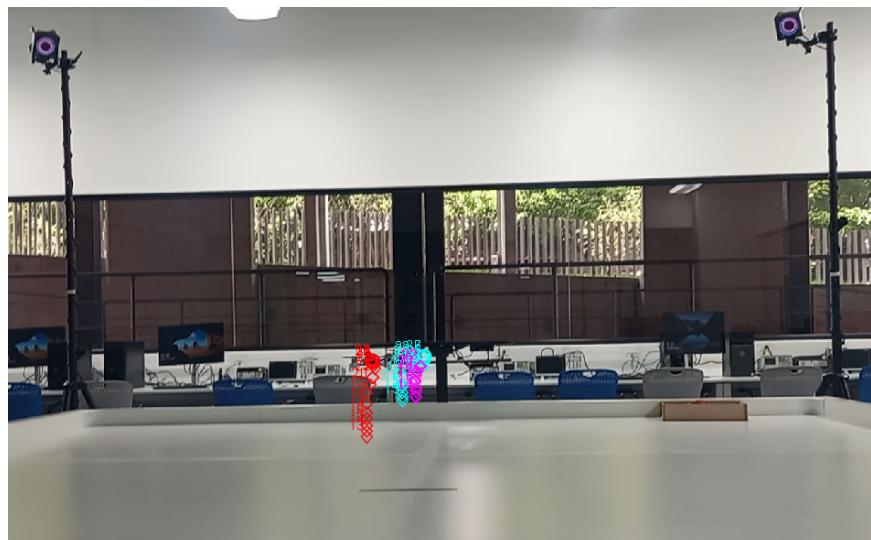


Figura 21: Prueba con 3 drones .

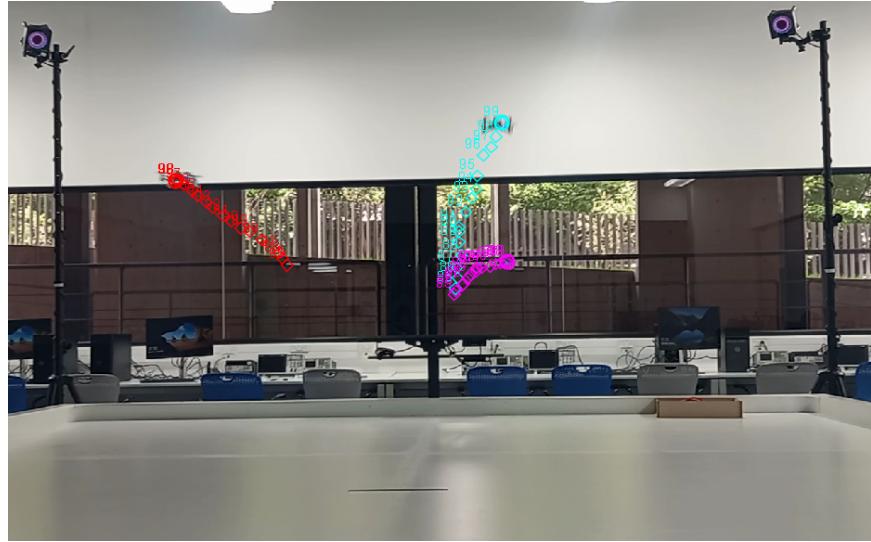


Figura 22: Prueba con 3 drones



Figura 23: Prueba con 3 drones

¿Cómo se ve esto en la figura?

puede llegar a apreciarse un movimiento pausado durante las trayectorias debido a que en promedio cada segundo se actualiza el punto espacial de referencia para el controlador. Dependiendo de la velocidad de la trayectoria, o la frecuencia del movimiento en caso de que sea periódico, puede alcanzarse un movimiento más fluido en los Crazyflies. Entre las observaciones realizadas durante estas pruebas, se determinó que Crazyswarm no tiene un bloqueo en la ejecución de sus comandos en caso de que los drones no alcancen las posiciones establecidas, por lo que si uno de los Crazyflies llegara a quedarse sin batería durante una rutina y cayera, esto no afectaría en las trayectorias de los demás, ya que el ordenador sólo envía el paquete de datos con la posición recopilada por el sistema de captura y las posiciones deseadas, ~~y~~ el lazo de control se cierra internamente en el Crazyflie.

Por el momento está decente el capítulo, pero falta presentar resultados más tangibles empleando las mediciones del OptiTrack. Muchas de las cosas que se mencionan con respecto del drift del drone por el peso pueden medirse y mostrarse con data de Motive, para que sea más contundente que sólo observaciones. También te recomendaría releer los párrafos luego de escribirlos, dado que en algunos casos le falta claridad a la redacción y da la impresión que estás dando muchas vueltas para llegar al punto. Igualmente pueden presentarse también los datos que publica ROS2 en la terminal o bien en RViz, ya que el capítulo sí está escueto en resultados tangibles.

CAPÍTULO 8

el ecosistema Integración de Crazyswarm con Robotat ✓

Como se ha mencionado a lo largo de este trabajo, la meta principal es alcanzar una fusión exitosa entre la comunicación interna del ecosistema Robotat y el sistema de control de Crazyswarm, por lo que dicha fusión debe realizarse mediante los métodos de comunicación ya establecidos y adaptar la infraestructura de Crazyswarm. Para esto, se creó una variante del servidor de comunicación del Robotat especialmente para establecer una comunicación con Crazyswarm, de esta forma es posible que cualquier computadora con la capacidad de crear clientes TCP pueda comunicarse con los drones sin necesidad de utilizar Ubuntu.

8.1. Comunicación en Robotat

Como concepto inicial, se pretendía utilizar el *toolbox* de Matlab que permite la comunicación directa con los tópicos de ROS2, sin embargo esto presentó problemas debido a que los puertos de las computadoras del Robotat están bloqueados por motivos de seguridad. Por otro lado, este método requería que todos los estudiantes que quisieran desarrollar pruebas con Crazyswarm debían comprar dicho *toolbox*, lo cual no es factible. Debido a esto, se buscó otra alternativa para establecer una comunicación entre Crazyswarm y Matlab, por lo que se procedió a trabajar en un paquete especializado en ROS con un nodo que envíe y reciba datos mediante una comunicación TCP. Sin embargo, la comunicación entre Matlab y ROS requería de un servidor, el cual podría realizar las mismas tareas que dicho nodo, por lo que finalmente se decidió implementar únicamente el servidor y que este ejecutara las funciones de Crazyswarm internamente basado en los comandos recibidos. La estructura final implementada en el Robotat se muestra en la Figura 24.

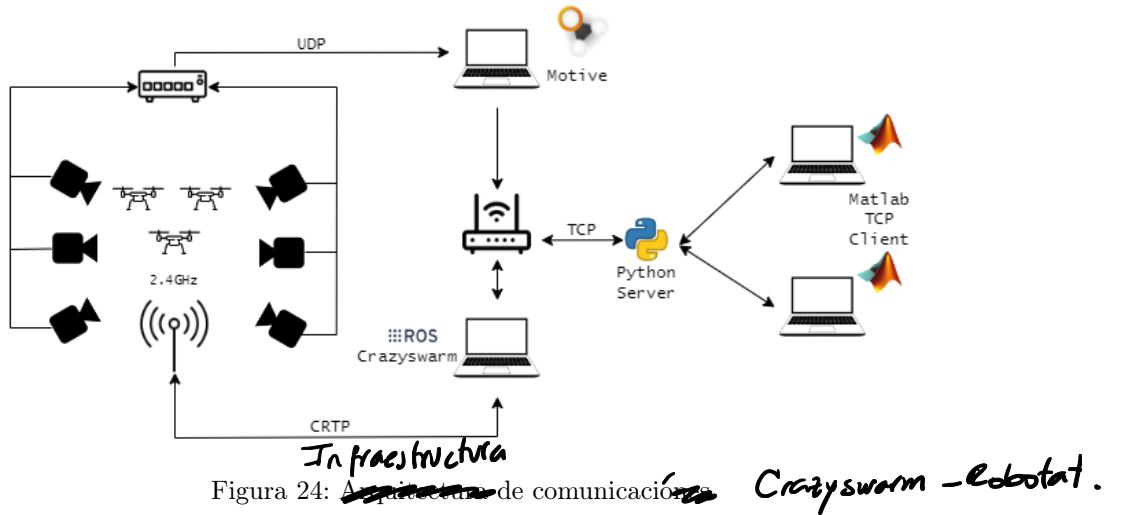


Figura 24: Arquitectura de comunicación.

Crazyswarm - Robotat.

BYTES	src	len	pty	pld
	1	1	1	8520
	Origen del paquete	Tamaño del array	Tipo de paquete	Concatenación de las trayectorias
	205-Matlab	Cantidad de CF o largo de la trayectoria	0-127 DATA 128-255 CMD	3 matrices de posiciones espaciales

Cuadro 1: Estructura del paquete de datos

*para la comunicación
Crazyswarm - Robotat.*

8.1.1. Creación del servidor Crazyswarm

Los dispositivos a conectar son

- Ordenador que corre Crazyswarm .
- Ordenador que corre OptiTrack .
- Ordenadores que deseen comunicarse con Crazyswarm .
- Crazyflies a ~~modo~~ ^{modo}.

Los usuarios del Robotat pueden conectarse al router que genera la red Wi-Fi Robotat mediante el servidor con el mismo nombre y pueden realizar dicha conexión mediante antenas externas. Se replicó este concepto creando un servidor en Python usando la dirección IP 192.168.50.170 y el puerto 64183, a diferencia del servidor original del Robotat que tiene una dirección 192.168.50.200, este cambio se debe a que ya se está utilizando este intermediario para enviar datos del sistema de captura de movimiento desde OptiTrack a Crazyswarm, por lo que podría haber un problema de saturación en el puerto. El servidor recibe información codificada en formato JSON, la estructura de los datos se presenta en el Cuadro 1.

Wi-Fi

Las instrucciones base de control utilizan las funciones de Crazyswarm dentro del ciclo del servidor, enviando puntos mediante la función *goto* para posiciones específicas o para

trayectorias, que envía a la cantidad de Crazyflies especificada los puntos de un trayectoria generada previamente.[17] Para llevar a cabo este servidor, se importaron las librerías mostradas a continuación.

- `thread`
- `socket`
- `json`
- `numpy`
- `threading`

El servidor se conecta a la dirección IP y puerto especificados comenzando a recibir datos, el paquete se lee y decodifica siguiendo el formato JSON, luego los datos se procesan mediante un match (equivalente a switch en Python), el cual funciona como condicional en función de los comandos (CMD) recibidos. Luego de clasificar el comando, se procede a leer el set de datos (pld) y, dependiendo del comando, se re agrupan los datos. Esto se debe a que la decodificación funciona mejor si las matrices generadas en Matlab se envían como arreglos de una sola dimensión, haciendo necesario que dicho arreglo vuelva a agruparse como matriz. Además, el pld también se aprovecha para enviar datos como el número de Crazyflies a utilizar o en el caso de un *goto*, la dirección del Crazyflie a utilizar. Una vez que se extraen los datos del pld, estos se reorganizan de forma que se crean 3 arreglos en Python y se ponen en forma de matriz según el número de Crazyflies a utilizar y la cantidad de puntos en la trayectoria. De esta forma, el servidor se inicia como un código de rutinas normal de Crazyswarm y este se ejecuta directamente como comando de ROS2 luego de inicializar Crazyswarm.

Una vez que se verificó el funcionamiento de la comunicación TCP, se procedió a evaluar la eficiencia de la comunicación. El objetivo inicial era generar las trayectorias para múltiples drones en Matlab y enviar los puntos en forma de arrays. El problema que presentó este concepto es la cantidad de datos a enviar, ya que para generar las trayectorias se genera un punto cada 0.1 segundos, por lo que una trayectoria de 10 segundos representa 100 posiciones espaciales por Crazyflie, resultando en 3,000 datos de tipo *double* si se desea trabajar con 10 Crazyflies. Se redujo el periodo de duración de cada punto de las trayectorias a 1 segundo para evitar este problema, por lo que una trayectoria de 10 segundos con 10 Crazyflies genera 300 datos. Este cambio no genera inconvenientes con la implementación de las trayectorias en Crazyswarm, ya que esperar 1 segundo entre puntos no representa un problema de continuidad para los drones, siendo visualmente eficiente. Para asegurar que el paquete de datos puede ser procesado por el servidor, se definió adentro de este que cada paquete tiene un largo de 8120 bytes, dando espacio a que se trabajen con trayectorias de hasta 10 Crazyflies con una duración de 20 segundos.

Luego de múltiples pruebas con distintas rutinas de vuelo, se observó un problema recurrente que se da cada vez que desde Matlab se envía un comando que genere un error en Crazyswarm, por ejemplo enviar un comando para un Crazyflie que no existe o que no fue inicializado. El problema que se presenta cuando se da este tipo de error es que la conexión entre Matlab y el servidor puede perderse, lo que implica volver a crear el cliente, esto puede

```
jose@jose-VirtualBox:~/ros2_ws/src$ ros2 run crazyflie_examples hello_world
Connected to: 192.168.50.200:62096
Crazyflie server is listening on port 64183...
[[ 1.11937969  0.51029564 -0.56795187 -1.12402704 -0.64667694  0.42522496
  1.10617699  0.770115 -0.27398717 -1.0661868  0.8781392  0.11726553
  1.00485688  0.96858744  0.04188318 -0.92341473 -1.0396494  -0.20003521
  0.82349044  1.08990277 ]]

[[ -1.11937969 -0.51029564  0.56795187  1.12402704  0.64667694 -0.42522496
  -1.10617699 -0.770115  0.27398717  1.0661868  0.8781392 -0.11726553
  -1.00485688 -0.96858744 -0.04188318  0.92341473  1.0396494  0.20003521
  -0.82349044 -1.08990277 ]]

[[ 0.14975012  1.33681104  1.29481405  0.06237099 -1.22741567 -1.38872202
  -0.27324376 -1.09345356  1.45483472  0.47864754 -0.93760597 -1.49182888
  -0.6744712   0.7629922  1.49896468  0.8567953  -0.57310713 -1.47609751
  -1.02197065  0.37175131 ]]

[[ -0.14975012 -1.33681104 -1.29481405 -0.06237099  1.22741567  1.38872202
  0.27324376 -1.09345356 -1.45483472 -0.47864754  0.93760597  1.49182888
  0.6744712   -0.7629922 -1.49896468 -0.8567953  0.57310713  1.47609751
  1.02197065 -0.37175131 ]]
```

Figura 25: Servidor en funcionamiento

volverse problemático ya que el servidor no indica si se perdió la conexión, si se vuelve a crear un cliente sin que la conexión se haya perdido puede generar errores en el servidor, obligando al usuario a reiniciarlo y por ende todas las conexiones en Matlab deben reiniciarse de igual forma. Reiniciar la conexión no presenta problemas a gran escala ni interfiere con el funcionamiento de Crazyswarm, sin embargo es necesario esperar un minuto o dos para volver a iniciar el servidor, ya que luego de apagarlo este aun detecta que el puerto está en uso. Otra alternativa es cerrar la terminal de Ubuntu y abrir una nueva para volver a iniciar el servidor, tomando en cuenta que debe de volver a activarse la configuración de ROS.

8.2. Códigos de comunicación por TCP en Matlab

Matlab provee una serie de herramientas que permite la comunicación por TCP con otros dispositivos conectados a la misma red mediante la función `tcpclient()`, la cual requiere de una dirección IP y un puerto, dicha función crea una variable de tipo `tcpclient`.^[18] Una vez que se ha creado el cliente TCP en el espacio de trabajo, puede procederse a enviar y recibir datos del servidor mediante las funciones `write` y `read`, tomando en consideración que antes de escribir al cliente, se debe crear la estructura del paquete con el formato mencionado anteriormente. Luego debe codificarse en formato JSON y convertirse en datos tipo `uint8`. Cabe destacar que, como se mencionó anteriormente, la comunicación se vuelve más eficiente cuando las matrices se envían como arreglos, por lo que las matrices de posición espacial se convirtieron en vectores y fueron concatenados en el `pld`. Cada función creada utiliza parámetros distintos y los manipula según la necesidad, pero todos presentan una estructura similar a la siguiente

```
s.len = length(x);
s.cmd = 179;
s.pld = [N; x(:); y(:); z(:)];
write(tcp_obj, uint8(jsonencode(s)));
```

donde `N` representa el número de Crazyflies a utilizar y `x, y` y `z` son las matrices con las posiciones espaciales respectivas. Debido a que es preferible enviar las matrices como un arreglo en forma de vector de una sola dimensión con los datos concatenados, se envían las matrices en forma de vector. De esta forma 3 matrices de 10×10 se convierten en un vector con 300 dimensiones. Para alcanzar una integración eficiente y accesible para todos los usuarios del Robotat, se diseñaron funciones en Matlab que únicamente requieren de

Funciones de Matlab		
cmd	Función	Descripción
-	crazyswarm_connect	Crea el objeto TCP para conectar el cliente al servidor.
178	crazyswarm_goto	Envía a un Crazyflie especificado a una posición espacial.
179	crazyswarm_traj	Envía las trayectorias discretizadas para los Crazyflies deseados.
180	crazyswarm_land	Aterriza al mismo tiempo a todos los Crazyflies especificados.
190	crazyswarm_test	Verifica la comunicación con el servidor Crazyswarm.
-	crazyswarm_disconnect	Elimina el objeto TCP y corta la comunicación con el servidor.

Cuadro 2: Funciones de Matlab

la especificación del o de los Crazyflies a utilizar y los distintos puntos de las trayectorias. Tomando esto en cuenta, dichas funciones cuentan con programación defensiva para evitar percances, como que se envíe un comando a un Crazyflie que no existe, lo que podría causar fallas en Crazyswarm; o como que se envíen posiciones espaciales que sobrepasen los límites del Robotat, lo cual puede incidir en daños a la Crazyflies, al equipo de laboratorio o a los usuarios. Las funciones creadas ~~██████████~~ se muestran en el Cuadro 2.

Todas las funciones reciben como parámetro un objeto de tipo *tcpclient* de Matlab, a excepción de *crazyswarm_connect()* que genera dicho objeto.

8.3. Diseño de trayectorias en Matlab

Una de las ventajas más notables de utilizar Matlab como cliente principal, es la versatilidad que el programa presenta en el desarrollo de modelos matemáticos. El objetivo principal del algoritmo generador de trayectorias es que sea eficiente con cualquier cantidad de drones requerida. Para esto, se realizó un código que genera distintas trayectorias con las

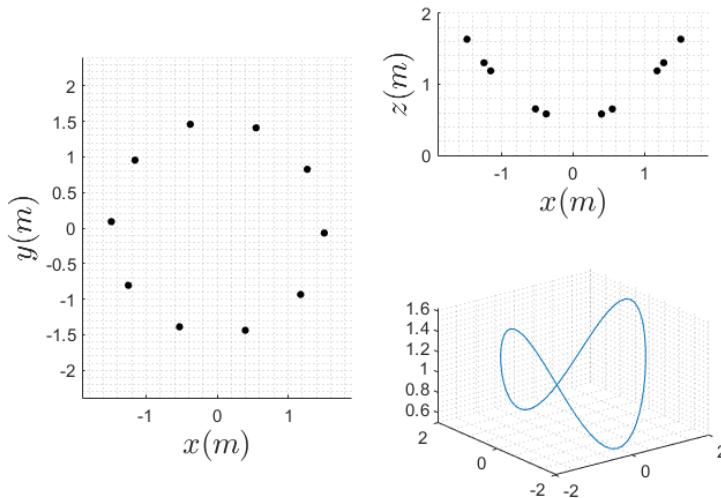


Figura 26: Trayectoria parabólica

consideraciones del usuario, cada sección del código genera una trayectoria distinta a la que se le pueden modificar parámetros como duración, radio y/o amplitud de oscilación, altura máxima y cantidad de drones.

Todas las trayectorias se basaron en el modelo de un oscilador armónico y una parametrización polar para obtener una parametrización estándar. En general, trabajan con el modelo de la Ecuaciones 9 y 10.[19] donde n es un vector de tiempo discretizado con el período de muestreo seleccionado, N es la cantidad de drones a utilizar y k es un número entero entre 0 y $N-1$ para crear el desfase simétrico entre drones.

modo matemática \$ \$

$$x_k[n] = R \cos\left(n + \frac{2\pi k}{N}\right) \quad \text{\textbackslash left(\textbackslash right)} \quad (9)$$

$$y_k[n] = R \sin\left(n + \frac{2\pi k}{N}\right) \quad (10)$$

El resto de trayectorias se basaron en este mismo concepto variando la parametrización en función de cada geometría deseada, por ejemplo, una trayectoria circular en el plano XY y una parábola en los otros planos se haría como en las Ecuaciones 9 y 10 junto la Ecuación 11, donde A y h representan las alturas máxima y mínima respectivamente.

modo mak

$$z_k[n] = Ax[n]^2 + h, \quad (11)$$

Este movimiento se ve como en la Figura 26 cuando se utilizan 10 Crazyflies.

8.4. Integración de Crazyswarm con otros agentes autónomos

Debido a que la integración con el Robotat se realizó mediante un servidor local, este puede utilizarse también para manejar y redirigir la información recopilada por el sistema de captura de movimiento. OptiTrack proporciona un servidor llamado NatNetClient, el cual funciona de igual forma en Python mediante un protocolo TCP y permite obtener directamente la posición y la orientación de un cuerpo rígido, donde la orientación puede proporcionarse en forma de ángulos de Euler con una secuencia especificada o de cuaternión unitario. Las funciones de este cliente pueden importarse en forma de librería al servidor creado de Crazyswarm, permitiendo trabajar en conjunto con los Crazyflies y con el sistema de captura, de forma que otros agentes autónomos que requieran de un posicionamiento en tiempo real pueden implementarse en el Robotat junto con Crazyswarm.

Para realizar esta integración, fue necesario configurar el nodo en ROS2 que se encarga de publicar las mediciones del sistema de captura, de forma que tanto MOCAP4ROS2 como Crazyswarm fueran capaces de recibir y procesar el estado de marcadores en forma de cuerpos rígidos y no solo como *single markers*, como se estuvo trabajando desde un inicio. Una vez que se permitió el procesamiento de estos datos y que se incluyeron las funciones de NatNetClient en el servidor, se importaron las funciones existentes del Robotat para acceder en tiempo real a la posición y a la orientación de los cuerpos rígidos detectados por las cámaras. Cabe destacar la importancia de la utilización de *multi-threading* para evitar un bloqueo en el funcionamiento del servidor mientras se realizan otras operaciones, de esta forma es posible solicitar el estado de un marcador al sistema de captura mientras se realiza una trayectoria con los Crazyflies sin que el servidor entre en conflicto.

8.5. Pruebas finales y validación

Para validar los resultados obtenidos en la experimentación de la integración, se pensó en utilizar una función propia de Crazyswarm para obtener la posición de los Crazyflies en tiempo real y verificar que las trayectorias seguidas fueran similares a las rutinas creadas en Matlab. Sin embargo, este tipo de funciones que accede al estimador de los drones aún son propias de Crazyswarm 1. ~~1.0, 1.1, 1.2, 1.3~~ no tienen soporte en Crazyswarm 2, por lo que se recurrió a utilizar las herramientas de Motive para grabar los datos de las trayectorias.

Motive permite registrar datos de posiciones y orientaciones para los marcadores y cuerpos rígidos que se encuentren al alcance de las cámaras, pudiendo exportar la información en un archivo de tipo csv. Para validar los resultados y comprobar que los Crazyflies realizaban trayectorias similares a la diseñadas en Matlab, primero se compararon las trayectorias con el paraboloide elíptico. Luego de filtrar los datos y graficarlos se llegó a un resultado como el de la Figura 27. Como puede apreciarse, la trayectoria seguida es visualmente idéntica a la diseñada en Matlab (Figura 26), para validar los datos de forma cuantitativa se obtuvieron las distancias mínimas y máximas en los ejes en la trayectoria diseñada y en la real. Como puede observarse en el Cuadro 3, los errores en el posicionamiento de los extremos de la trayectoria no superan el 8.5 %, resultando en un error promedio del 3.85 %.

También se realizó la validación para la trayectoria conocida como elipsoide inclinado,

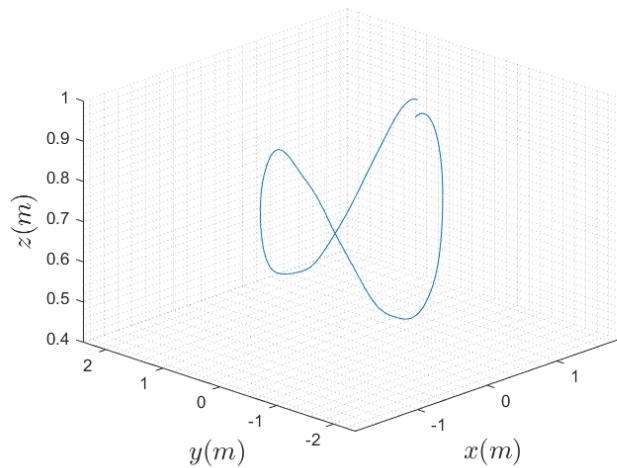
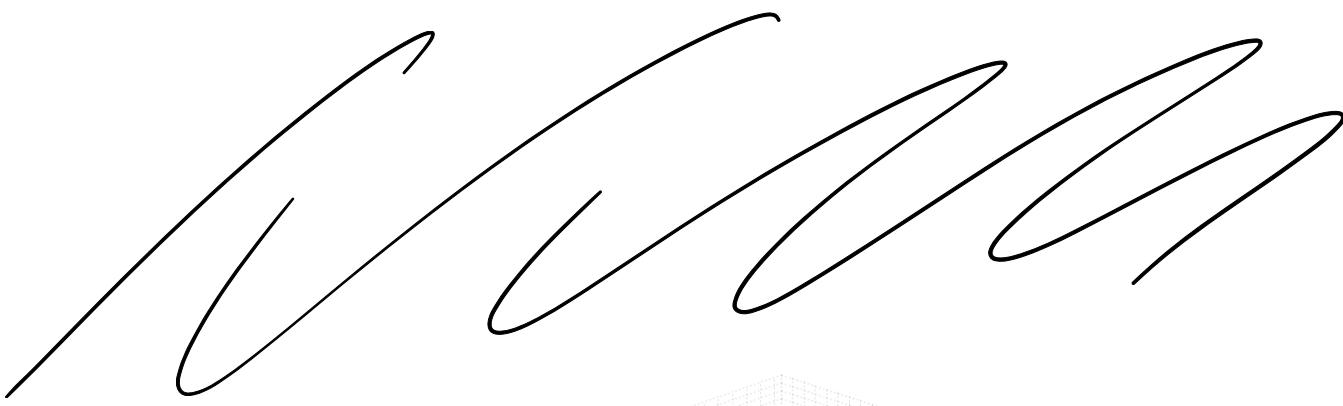
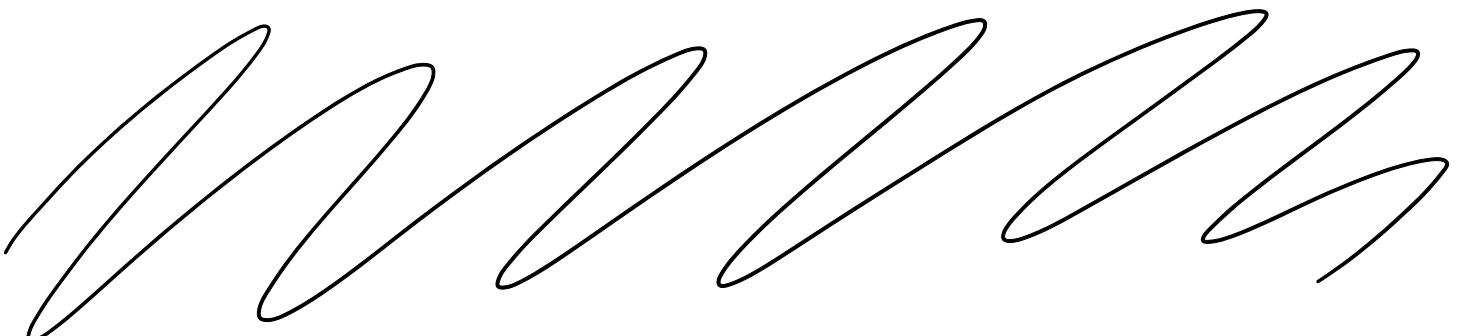


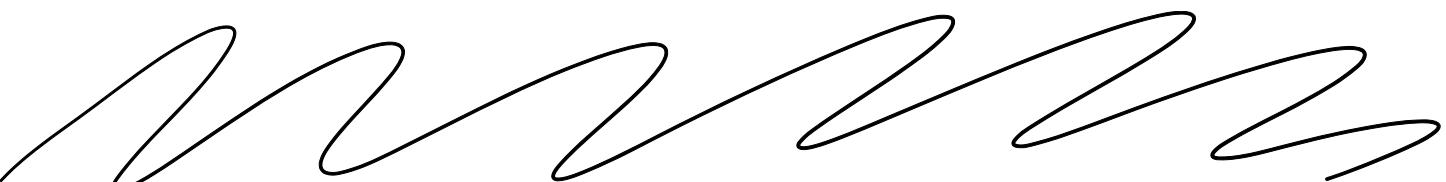
Figura 27: Trayectoria experimental 1.



Porcentajes de Error				
		Teórica(m)	Real(m)	Error(%)
Z	Máximo	0.9991	0.9374	6.18
	Mínimo	0.5297	0.4993	5.73
X	Máximo	0.9950	0.9897	0.53
	Mínimo	-0.9991	-1.0096	1.05
Y	Máximo	0.9699	0.9825	1.30
	Mínimo	-0.9258	-1.0028	8.32

¿MSE para las trayectorias reales vs las teóricas?

Cuadro 3: Resultados de posiciones ¿De qué?



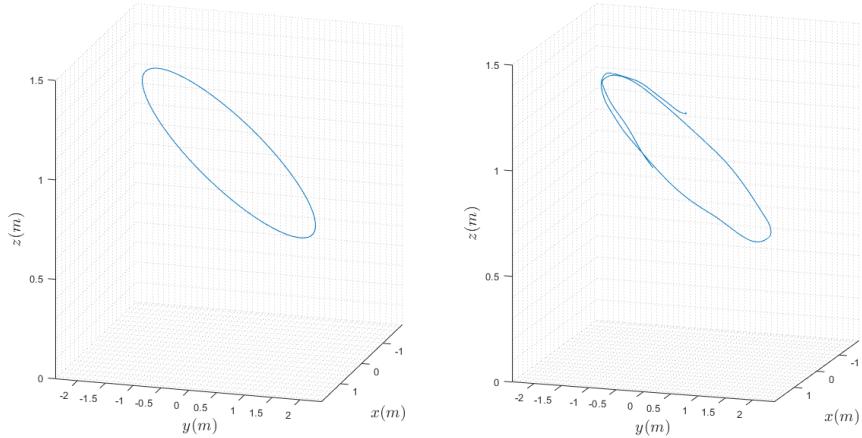


Figura 28: Validación de un ellipsoide inclinado .

Porcentajes de Error				
		Teórica(m)	Real(m)	Error(%)
Z	Máximo	1.3750	1.3250	3.63
	Mínimo	0.6252	0.6084	2.67
X	Máximo	1.1248	1.1177	0.64
	Mínimo	-1.1247	-1.1139	0.96
Y	Máximo	1.4993	1.4994	0.00
	Mínimo	-1.4928	-1.4999	0.47

Cuadro 4: Resultados de posiciones ¿ De qué ?

¿MSE?

de la misma forma en que se hizo con la trayectoria anterior. Los resultados se muestran en la Figura 28 y en el Cuadro 4. Como puede apreciarse en los porcentajes de error, esta trayectoria presentó un error máximo 2 veces menor al máximo de la trayectoria anterior y un error mínimo de 0 %, resultando en un error promedio de 1.3950 %. Esto se debe a que el tipo de trayectoria presenta una continuidad suave y sus cambios con respecto al tiempo en los 3 ejes no presentan máximos o valores ascendentes de forma simultánea. A diferencia de dicha trayectoria, el paraboloide hiperbólico sí presenta valores ascendentes y máximos en los mismos instantes de tiempo, provocando movimientos y giros más agresivos como se aprecia en la Figura 29, donde se grafican las derivadas de las posiciones con respecto al tiempo y puede observarse que entre la medición 100 y 200 se presenta una velocidad en x con magnitud máxima y velocidades en y y z ascendentes. De esta forma, puede afirmarse que trayectorias suaves y continuas presentarán mayor eficiencia y fidelidad a los datos teóricos.

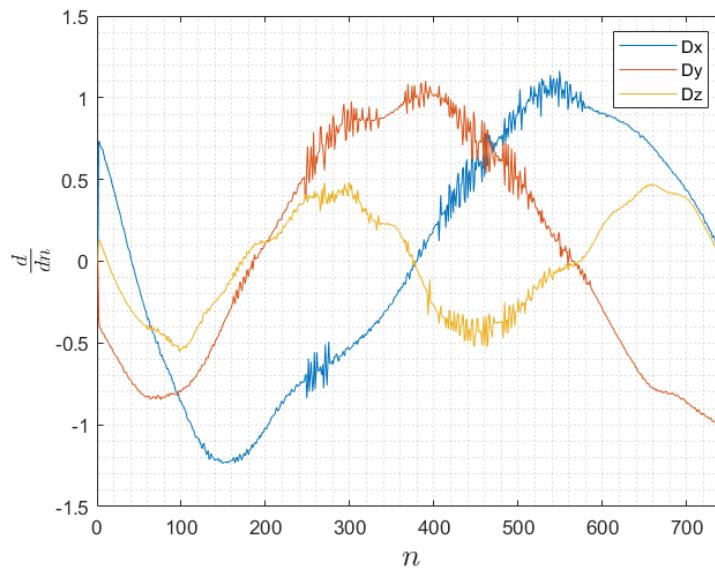
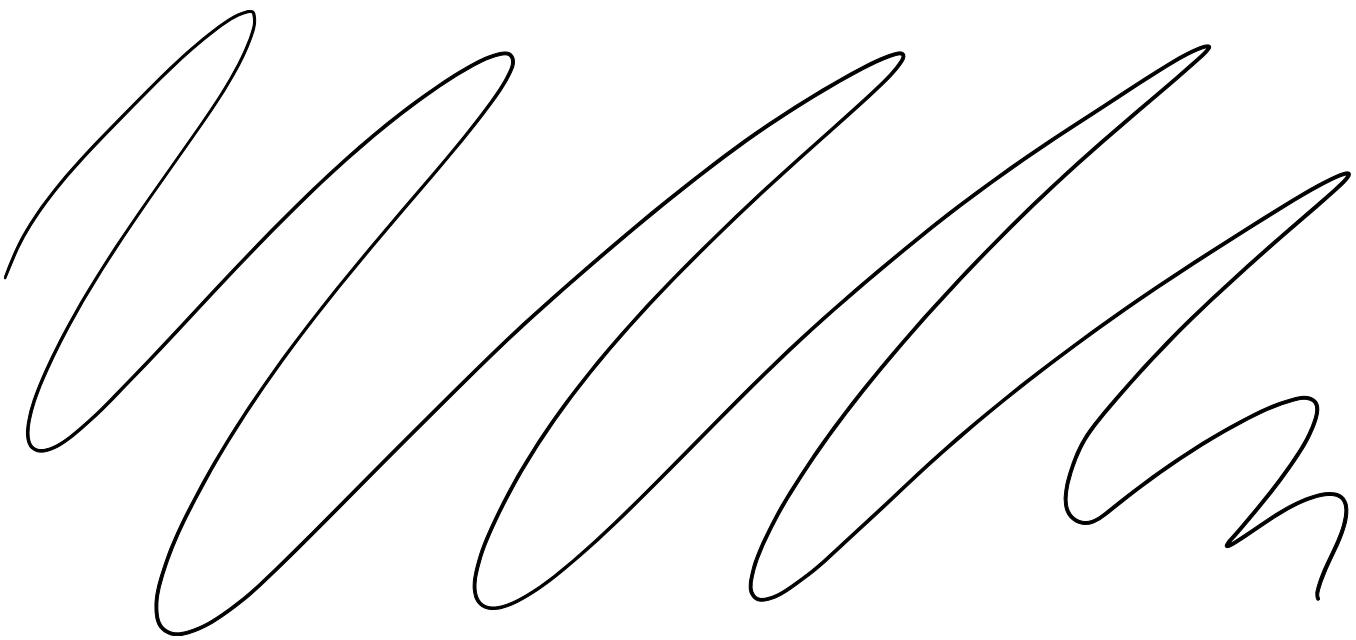


Figura 29: Derivadas del movimiento en 3 ejes.

¿cuál?

Similar al capítulo anterior, lo que tenés actualmente está decente para esta primera entrega pero sí deja mucho que desear en comparación a los resultados prácticos que han obtenido. Para las trayectorias aún falta hacer el cálculo del error cuadrático dado que evaluar sólo el punto inicial y final de la trayectoria no presenta una perspectiva completa. Adicionalmente, muchos resultados que se presentan como simples observaciones deben validarse con data. Por ejemplo, cuestiones como el tiempo de muestreo entre puntos para las trayectorias y los posibles errores de comunicación Matlab-Python pueden presentarse con datos tangibles de pruebas realizadas con distintos períodos de muestreo y viendo el listado de errores que despliega Python con paquetes errados. La estructura del código del servidor tampoco queda muy clara haciendo sólo una descripción de texto, por lo que valdría la pena agregar diagramas de bloque o de flujo que dejen claro qué era lo que ya tenía por defecto el Robotat y en dónde es que se está haciendo la integración con el Crazyswarm.

CAPÍTULO 9

Conclusiones

- El filtro de Kalman extendido presenta una excelente integración entre las mediciones de la IMU de los Crazyflies y las mediciones realizadas por el sistema de captura de movimiento, a pesar de la diferencia de tiempos de trabajo debido a que el origen de las mediciones es distinto. La estabilidad observada durante las pruebas en lazo cerrado demuestran una estabilidad total tanto en rutinas estacionarias de regularización de una posición espacial, como en la interpolación de puntos en una trayectoria para un rastreo de referencias.
- Se levantó la infraestructura de Crazyswarm, implementando las cámaras de captura de movimiento OptiTrack y utilizando el método de *single marker* para determinar la posición espacial de los drones Crazyflie.
- Se alcanzó una integración exitosa entre el sistema Crazyswarm y el Robotat a través de un servidor que permite la comunicación entre comandos de Matlab y el sistema de control de los Crazyflies.
- El servidor desarrollado permite el desarrollo de proyectos relacionados con sistemas de control y robótica de enjambre, permitiendo expandir las capacidades del laboratorio Robotat y los campos de estudio.

CAPÍTULO 10

Recomendaciones

- Se recomienda trabajar Crazyswarm en un ordenador que cuente con una partición de Ubuntu en lugar de una máquina virtual o desde una memoria externa, ya que esto limita las capacidades del sistema operativo y puede volver lenta la ejecución de Crazyswarm.
- Por cuestiones de seguridad de los usuarios del Robotat, se recomienda que durante la etapa de desarrollo e investigación del funcionamiento del sistema operativo, se coloque una red o manta alrededor del espacio de trabajo del Robotat, para evitar accidentes que dañen a los usuarios, a los drones o al equipo del laboratorio.
- Se recomienda expandir las funcionalidades del servidor Crazyswarm desarrollado para que la comunicación sea más eficiente y con ello se facilite el desarrollo de aplicaciones, junto con la expansión en la gama de sistemas autónomos en el Robotat.

CAPÍTULO 11

Bibliografía

- [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] “Crazyswarm Documentation.” (), dirección: <https://crazyswarm.readthedocs.io/en/latest/>. (Accedido el: 21/4/2023).
- [4] “Crazyflie 2.1 | Bitcraze.” (), dirección: <https://www.bitcraze.io/products/crazyflie-2-1/>. (Accedido el: 13/5/2023).
- [5] “Kalman Filter.” (), dirección: <https://www.sciencedirect.com/topics/earth-and-planetary-sciences/kalman-filter>. (Accedido el: 18/8/2023).
- [6] “State estimation.” (), dirección: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/. (Accedido el: 20/5/2023).
- [7] G. EwoudJ.J.Smeur QipingChu, “Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles,” *Journal of Guidance, Control, and Dynamics: devoted to the technology of dynamics and control*, pág. 1, 2016.
- [8] “Multithreading, más potencia para los procesadores.” (), dirección: <https://www.ionos.es/digitalguide/servidores/know-how/explicacion-del-multithreading/>. (Accedido el: 9/9/2023).
- [9] “Socket Programming in Python.” (), dirección: <https://realpython.com/python-sockets/>. (Accedido el: 18/8/2023).
- [10] “¿Qué es el UDP?” (), dirección: <https://www.cloudflare.com/es-es/learning/ddos/glossary/user-datagram-protocol-udp/>. (Accedido el: 13/5/2023).
- [11] “Primex-40-Specs.” (), dirección: <https://optitrack.com/cameras/primex-41/specs.html>. (Accedido el: 21/4/2023).
- [12] “Overview-Crazyswarm2.” (), dirección: <https://imrclab.github.io/crazyswarm2/overview.html>. (Accedido el: 22/4/2023).

- [13] “CRTP-Communication with the Crazyflie.” (), dirección: <https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/crtpp/>. (Accedido el: 6/5/2023).
- [14] “Crazyflie Android client.” (), dirección: <https://www.bitcraze.io/documentation/repository/crazyflie-android-client/master/>. (Accedido el: 18/8/2023).
- [15] M. C. Falconi. R, “Dynamic Model and Control of an Over-actuated Quadrotor UAV,” *International Federation of Automatic Control*, pág. 1, 2012.
- [16] “MOCAP4ROS2.” (), dirección: <https://mocap4ros2-project.github.io/index.html>. (Accedido el: 18/8/2023).
- [17] “Python API Reference.” (), dirección: <https://crazyswarm.readthedocs.io/en/latest/api.html>. (Accedido el: 18/8/2023).
- [18] “Comunicación TCP/IP.” (), dirección: <https://la.mathworks.com/help/matlab/tcpip-communication.html>. (Accedido el: 18/8/2023).
- [19] S. J, *Cálculo de varias variables. 7ma edición. Trascendentes tempranas*. Cengage Learning, 2012.

CAPÍTULO 12

Anexos

Puede colocarse el enlace al repositorio de código aquí.