
Diseño e implementación de una plataforma de pruebas para sistemas de control para el dron *Crazyflie 2.0*

Francis José Sanabria Salazar



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Diseño e implementación de una plataforma de pruebas para
sistemas de control para el dron *Crazyflie 2.0***

Trabajo de graduación presentado por Francis José Sanabria Salazar
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

Vo.Bo.:

(f) _____
Ing. Pablo Mazariegos

Tribunal Examinador:

(f) _____
Ing. Pablo Mazariegos

(f) _____
MSc. Carlos Esquit

(f) _____
Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, 5 de junio de 2022.

Prefacio

Mi pasión por el mundo aeronáutico y aeroespacial, así como mi interés por explorar las ciencias de la aviación, especialmente la Aviónica y Sistemas de Control, me llevaron a desarrollar el presente trabajo de graduación. Considerando que la educación es un pilar para el desarrollo, trabajar este proyecto como un método de aprendizaje para estudiantes de la Universidad del Valle de Guatemala en los cursos de Sistemas de Control 1 y 2, justifican que haya tomado la decisión de escoger la plataforma de pruebas como proyecto, dado que combinan dos de mis intereses en el mundo profesional.

El presente trabajo fue realizado bajo la supervisión del Ingeniero Miguel Zea, con la ayuda del Ingeniero Luis Rivera, a quienes expreso mi total agradecimiento por el apoyo brindado en el desarrollo del proyecto y por compartir sus conocimientos de manera desinteresada.

Agradezco a mis padres por haberme apoyado en mi vida académica, por su esfuerzo y amor al no dejarme solo a lo largo de mi carrera universitaria, quienes confiaron en mí y forjaron a la persona que ahora soy. Quiero agradecer al Centro Universitario Ciudad Vieja por apoyarme y formarme como un profesional, especialmente durante el esfuerzo realizado durante la pandemia del COVID-19.

Prefacio	III
Lista de figuras	VIII
Lista de cuadros	IX
Resumen	X
Abstract	XI
1. Introducción	1
2. Antecedentes	2
3. Justificación	5
4. Objetivos	6
4.1. Objetivo general	6
4.2. Objetivos específicos	6
5. Alcance	7
6. Marco teórico	8
7. El Crazyflie 2.0 y su máquina virtual	21
7.1. Descripción del Crazyflie 2.0 y validación de funcionamiento	21
7.2. Instalación y configuración de la máquina virtual	25
7.2.1. Máquina virtual Bircraze	26
7.2.2. Exportar la máquina virtual	28
7.3. Uso del Crazyflie Client	28
7.3.1. Conexión del Dron a la aplicación	29
7.3.2. Modificaciones iniciales al dron	30
7.4. Programación del Crazyflie	32
7.4.1. Compilar el código	33

7.4.2. Flashear el Crazyflie 2.0	34
7.5. La API de Python	34
7.5.1. Estructura de la biblioteca	34
7.5.2. Parámetros	35
7.5.3. Utilidades	36
7.6. La API del Crazyflie 2.0	37
7.6.1. Marco de registro y parámetros	37
7.6.2. CRTP - Comunicación con Crazyflie	38
7.6.3. Sensores para control	38
7.6.4. Controladores en el Crazyflie	39
7.6.5. Marco del Comandante	41
8. Descripción de los experimentos y las pruebas	42
8.1. Uso de la plataforma física	42
8.1.1. Conexión del codificador rotacional	43
8.2. Conexión de Arduino y Python	44
8.3. Conexión de Crazyflie y Python	44
8.3.1. Lectura de parámetros	44
8.3.2. Movimiento del dron a través de Python	45
8.4. Interfaz gráfica en Python	45
8.4.1. Gráfica del codificador rotacional	46
8.4.2. Guardar datos de corrida	46
8.4.3. Explicación de la interfaz desarrollada	47
8.4.4. Uso de Matlab para graficar la Respuesta al Escalón	48
8.5. Identificación de sistemas para determinar la planta del <i>Crazyflie 2.0</i> y su plataforma en función de transferencia	49
8.6. Identificación de sistemas para determinar la planta del <i>Crazyflie 2.0</i> y su plataforma en espacio de estados	52
8.7. Configuración de un controlador PID mediante su planteamiento en LQR . .	53
9. Guías de laboratorio desarrolladas	56
9.1. Manual de usuario e instalación de herramientas	56
9.1.1. Objetivos:	56
9.1.2. Descripción	57
9.1.3. Práctica Piloto	57
9.2. Laboratorio 1 - Control PID	59
9.2.1. Herramientas	59
9.2.2. Objetivos del laboratorio	59
9.2.3. Desarrollo del laboratorio	59
9.2.4. Práctica Piloto	60
9.3. Laboratorio 2 - Control LQR	61
9.3.1. Objetivos del laboratorio	61
9.3.2. Desarrollo del laboratorio	61
9.3.3. Herramientas	61
9.3.4. Práctica Piloto	62
10. Conclusiones	63

11. Recomendaciones	64
12. Bibliografía	65
13. Anexos	67
13.1. Manual de Recuperación de la Máquina Virtual	67
13.2. Manual de uso del <i>Crazyflie 2.0</i>	67
13.3. Guía de Laboratorio 1	67
13.4. Guía de Laboratorio 2	67

Lista de figuras

1.	Anillo experimental para control de orientación de un cuadricóptero [4].	4
2.	Plataforma física creada por Martínez para el Crazyflie 2.0 [5]	4
3.	Cuadricóptero Simple [6].	8
4.	Configuración en cruz (izquierda). Configuración en equis (derecha) [7].	8
5.	Enumerado de los 4 motores de un cuadricóptero en configuración equis (x) [7].	9
6.	Ángulos de euler en un cuadricóptero [7].	10
7.	Crazyflie 2.0 [1].	10
8.	Marco de referencia en el Crazyflie con el marco de referencia inercial [10]. . .	12
9.	Resultados de linealización de un cuadricóptero [11]	17
10.	Árbol de máquinas virtuales [13].	19
11.	Codificado rotacional óptico [15].	20
12.	Componentes del <i>Crazyflie 2.0</i>	23
13.	Armado de motores del <i>Crazyflie 2.0</i>	23
14.	Colocado de motores del <i>Crazyflie 2.0</i>	24
15.	Dron completamente armado.	24
16.	Conexión del Crazyflie al celular.	25
17.	Prueba de vuelo.	25
18.	Antena de Radio <i>Crazyradio PA</i> [17].	29
19.	Escaneo de Crazyflie.	29
20.	Conectar dron al <i>Crazyflie Client</i>	30
21.	Iniciar pestaña de <i>Bootloader</i>	30
22.	Configuración de <i>firmware</i>	31
23.	Programando el <i>Crazyflie</i>	31
24.	Configurando la comunicación del <i>Crazyflie</i>	32
25.	Carpeta del <i>Firmware</i> del <i>Crazyflie 2.0</i>	33
26.	Terminal agregada a <i>Visual Studio Code</i>	33
27.	Compilando el código del <i>Firmware</i> del <i>Crazyflie 2.0</i>	34
28.	Flasheo del código del <i>Firmware</i> del <i>Crazyflie 2.0</i>	34
29.	Flujo de información de sensores en el <i>Crazyflie</i> [21].	39
30.	Controlador PID en el <i>Crazyflie 2.0</i>	40

31. Conexión de codificador rotacional al Arduino	43
32. Puerto serial del Arduino con el ángulo dado	43
33. Conexión serial entre Python y Arduino	44
34. Conexión del dron a Python	44
35. Lectura de parámetros del <i>Crazyflie 2.0</i>	45
36. Control de orientación del dron con Python.	45
37. Gráfica del ángulo de cabeceo respecto al número de muestras	46
38. Gráfica del ángulo de cabeceo guardado como figura.	47
39. Interfaz gráfica	48
40. Gráfica de una corrida del ángulo de cabeceo.	49
41. Parámetros de rendimiento de la respuesta al escalón de un corrida del ángulo de cabeceo.	49
42. Resultado de la identificación de sistemas con 4 polos y 4 ceros.	50
43. Planteamiento del sistema completo.	50
44. Resultado de la identificación de sistemas para orden 1 (en azul) y 2 (en verde).	52
45. Estudiantes realizando la prueba piloto para el manual de instalación de herramientas.	58
46. Estudiantes realizando la prueba piloto para la guía de laboratorio 1.	61

Lista de cuadros

1. Variables involucradas en la dinámica del dron	12
2. Parámetros físicos del Crazyflie 2.0 [11].	15

Resumen

En el presente proyecto se trabajó con una plataforma para la experimentación de control de actitud para el *Crazyflie 2.0*; donde se comenzó realizando una guía básica de uso para el dron, en esta guía se describe la manera que se debe de configurar el dron para poder volarlo, así como la descripción de los pasos necesarios y los recursos a instalar para poder modificar el *firmware* del dron; la guía desarrollada también describe la instalación y configuración de una máquina virtual que contiene empaquetados todos los programas y recursos necesarios para poder trabajar con el *Crazyflie 2.0*.

Por otra parte, se describió la API de una librería en Python que permite controlar el dron a través de la computadora con funciones de alto nivel; se utilizó dicha librería para poder desarrollar una interfaz gráfica que permitía controlar el dron de manera sencilla, así como enviarle parámetros de configuración y modificar los controladores de vuelo para el ángulo de cabeceo del dron; dicha interfaz se le agregó la funcionalidad de visualizar los datos leídos del codificador rotacional de la plataforma física y guardarlos para posteriormente ser procesados según se requiera.

Se desarrollaron 2 guías de laboratorio; la primer guía de laboratorio se basó en el uso del controlador PID diseñada para el curso de Sistemas de Control 1 donde se utilizó la interfaz gráfica para modificar los parámetros del controlador PID y observar las distintas respuestas. La segunda guía de laboratorio se basó en el uso del controlador LQR diseñada para el curso de Sistemas de Control 2, a su vez también se utilizó la interfaz gráfica pero se le realizaron modificaciones para el controlador. Por otra parte, se describió la API del *Firmware* del dron y se cargó el nuevo controlador al microcontrolador del *Crazyflie 2.0*.

Finalmente, se realizó una prueba piloto con el manual de instalación y con las guías de laboratorio, en el cual se contó con el apoyo de estudiantes de los cursos de Sistemas de Control 1 y de Sistemas de Control 2, quienes realizaron la guía exitosamente en el tiempo de laboratorio estipulado para ambos cursos.

Abstract

This is an abstract of the study developed under the

CAPÍTULO 1

Introducción

En la Universidad del Valle de Guatemala la carrera de Ingeniería Mecatrónica está orientada a sistemas de control y robótica, cursos que en sus laboratorios se han trabajado con sistemas bastante sencillos, fue por ello que el Ing. Eléctrico Gabriel Martínez en su proyecto de graduación “Plataforma de pruebas y ajustes de sistemas de control para vehículos multirrotores” desarrolló una plataforma para hacer pruebas de algoritmos de control de actitud para el dron *Crazyflie 2.0*, los cuales fueron adquiridos por la Universidad del Valle de Guatemala. El proyecto presentado en este trabajo de graduación no es más que la continuación de la implementación de dicha plataforma de pruebas, presentando la implementación en software de la plataforma física así como una guía de uso del dron y un conjunto de guías de laboratorio donde se use la plataforma.

En este trabajo se presenta el proceso llevado a cabo para desarrollar e implementar una aplicación en *software* de la plataforma física en un ambiente de pruebas que permita diseñar, configurar y estudiar sistemas de control de orientación para el *Crazyflie 2.0* en los laboratorios para los cursos de Sistemas de Control 1 y 2, utilizando el dron como un recurso pedagógico para los laboratorios de los cursos antes mencionados. El desarrollo de este proyecto se divide en 3 partes principales: familiarizarse con el *Crazyflie 2.0* para aprender a volarlo, configurarlo y realizar las modificaciones a su *firmware*; desarrollar una interfaz gráfica que permita controlar el dron, modificar su controlador de orientación en un grado de libertad, y obtener datos del codificador rotacional disponible en la plataforma física. Finalmente, desarrollar un conjunto de guías de laboratorio para poder modificar y estudiar los sistemas de control de orientación del dron.

Cada una de las partes son evaluadas por separado. Primero se realiza un manual de operación del dron y una guía para la instalación de la máquina virtual que empaqueta los programas y herramientas para poder configurar el *Crazyflie 2.0*. En segundo lugar se realiza una interfaz gráfica en Python y finalmente, se realiza una prueba piloto para las guías de laboratorio a utilizar en los cursos de Sistemas de Control 1 y 2.

CAPÍTULO 2

Antecedentes

El Crazyflie 2.0 es un cuadricóptero de código abierto utilizado por muchas universidades para investigación y educación de los diferentes sistemas que lo componen [1]. El sistema de control de orientación ha sido un área de interés para los investigadores, utilizando el dron como plataforma para robótica e ingeniería de control [2], el trabajo de investigación en estas áreas se ha realizado a través de simulaciones mediante herramientas de *software* comerciales como MATLAB/Simulink o ROS previo a su implementación física [3]. Una vez realizada la simulación, se hace necesaria la implementación física en un ambiente controlado y seguro, por lo que se hace preciso el uso de una plataforma que permita su implementación física para la experimentación real [4].

A continuación se describe una serie de trabajos realizados en el área de control de cuadricópteros, incluyendo una plataforma desarrollada en la Universidad del Valle de Guatemala.

Crazyflie 2.0: Plataforma de código abierto

El Crazyflie 2.0 es una plataforma de desarrollo de vuelo de código abierto, muy versátil, con un peso de 27 gramos, esta plataforma ha sido de utilidad para realizar investigación por varias universidades del mundo por las características mencionadas con anterioridad. Las principales áreas exploradas son [1]:

- **Sensores:** La plataforma cuenta con varios sensores, a los cuales puede aplicarse procesamiento de señales para optimizar el rendimiento de los mismos obteniendo señales menos ruidosas.
- **Fusión de sensores:** La plataforma cuenta con distintos sensores, tales como el giroscopio y los acelerómetros, a los que se puede realizar fusión de sensores, por lo que se pueden estudiar algoritmos como el Filtro de Kalman para realizar estos procedimientos.

- **Teoría de control:** La plataforma cuenta con varios lazos de control distintos, para poderse controlar el alabeo, cabeceo y guiñada del Crazyflie 2.0. Adicionalmente, puede controlar el sistema de posicionamiento externo, permitiendo experimentar con distintos controladores y evaluar su desempeño en el dron.
- **Autonomía a bordo:** Implementación de una aplicación que permite el control externo sobre el firmware Crazyflie.

Dron Crazyflie 2.0 como plataforma de investigación y educación en robótica e ingeniería de control

Se utilizó el *Crazyflie 2.0* como una plataforma experimental de código abierto para la investigación y educación en robótica y sistemas de control. En la investigación se demuestra el potencial de su amplio uso para investigadores y estudiantes. Entre ellos destaca la adquisición de datos de medición de vuelo y la posibilidad de uso de plataformas como *MATLAB/Simulink* y *Python* que permita implementar un modelo matemático de la dinámica de Crazyflie 2.0 para simulaciones previas a la implementación física. Finalmente, los autores usaron un dispositivo apuntador (posicionador) ayudado de un sistema de visión de computador para poder controlar el dron [2].

Gazebo - ROS - Marco de Simulink para control de desplazamiento y seguimiento de trayectoria de Crazyflie 2.0

Nithya & Rashmi [3] realizaron una investigación en el área de desarrollo e implementación de algoritmos de control para drones mediante el uso de una plataforma de simulación. Para ello hicieron el uso de MATLAB y Simulink para el control de posición y seguimiento de trayectorias del modelo Gazebo del Crazyflie 2.0. Trabajaron en Gazebo y MATLAB/-Simulink y la interconexión de comandos entre ambos entornos. Se utilizó un controlador PID para lograr el control de vuelo estacionario y el seguimiento de la trayectoria del dron Crazyflie 2.0.

Aunque la plataforma se utilizó para probar un algoritmo de control de movimiento para el Crazyflie 2.0, el enfoque seguido de la investigación es el desarrollo y prueba de algoritmos control para cualquier dron compatible con ROS .

Control de 3 grados de libertad del modelo de un cuadricóptero

En el 2007 Kim, Stol & Veckman [4] crearon una plataforma que permiten control de 3 grados de libertad para el movimiento rotacional del dron de 4 motores, eliminando por completo el movimiento traslacional del mismo. El rango de operación de la plataforma correspondía a $\pm 40^\circ$ para el caso de alabeo y cabeceo, y de $\pm 180^\circ$ para la guiñada. Esta plataforma contaba con codificadores rotacionales que permitían leer de forma exacta la velocidad y posición rotacional de cada eje de movimiento, así como un *dSpace control board* utilizado para la adquisición de datos y para producir las señales de control PWM para los motores. El objetivo principal de esta plataforma era para el estudio, diseño y prueba

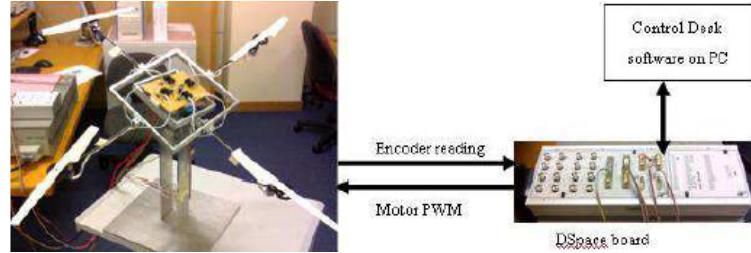


Figura 1: Anillo experimental para control de orientación de un cuadricóptero [4].

de controladores de orientación para un cuadricóptero. La Figura 1 muestra la plataforma experimental fabricada.

Plataforma física creada en la Universidad del Valle de Guatemala

Martínez [5] realizó el diseño e implementación física de una plataforma para el dron Crazyflie 2.0 con fines de ser utilizada para diseñar, configurar y evaluar el desempeño de algoritmos de control de actitud para dicho cuadricóptero.

En este trabajo se realizó un diseño simplificado de suspensión cardán de un grado de libertad compuesta de una plataforma central que permite el movimiento de cabeceo o alabeo (dependiendo de cómo se encuentre orientado el cuadricóptero). El eje de rotación de dicha plataforma posee un codificador rotacional que permite observar el comportamiento del dron y así evaluar el desempeño del controlador para el movimiento de cabeceo o alabeo. Sin embargo, entre las limitaciones de este proyecto fue que no se trabajó con la aplicación de la computadora, tampoco se trabajó con el *firmware* del dron, la implementación del codificador rotacional no se llevó a cabo [5].

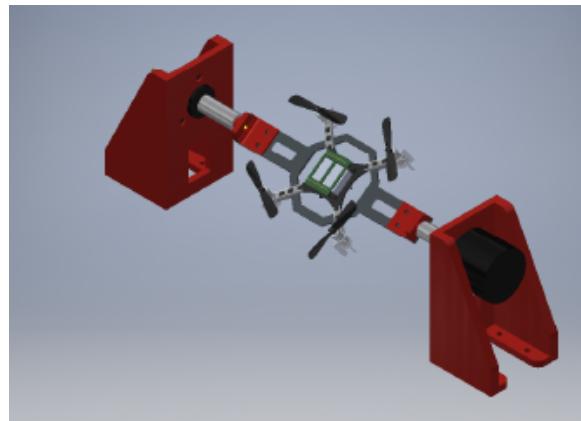


Figura 2: Plataforma física creada por Martínez para el Crazyflie 2.0 [5]

CAPÍTULO 3

Justificación

En distintas universidades alrededor del mundo y centros de investigación, existe un estudio tangible de sistemas de control en drones a través de plataformas de experimentación. El uso de estas plataformas permite hacer estudios más realistas que las simulaciones sin comprometer la seguridad de los investigadores y protegiendo la integridad del dron, siendo esta una considerable ventaja sobre las simulaciones, que no consideran todos los factores que se encuentran involucrados durante el vuelo.

El Crazyflie 2.0 es uno de los drones más utilizados para la investigación, dada su viabilidad y fácil desarrollo. Sin embargo, no existe una plataforma de pruebas comercialmente accesible. Es por ello que Martínez [5] en su trabajo trató de solventar este problema, quedando atrás la implementación en software de la plataforma y la implementación y prueba de controladores de orientación para el Crazyflie 2.0.

A través de este proyecto se plantea la continuación de la plataforma fabricada por Martínez, aprendiendo a configurar el *firmware* del dron para poder implementar distintos controladores de orientación en el mismo. También, se llevará a cabo el desarrollo de guías de prueba apoyadas por la creación de una aplicación en *Python* que permita obtener resultados importantes sobre las respuestas de los controladores implementados. Esto con el fin de poder utilizar la plataforma en los laboratorios de Sistemas de Control 1 y 2 con la ayuda de una interfaz gráfica realizada en *Python* que resultará de gran ayuda para la elaboración de las guías de laboratorio en la Universidad del Valle de Guatemala.

Es importante destacar que, a futuro los resultados de este proyecto contribuirán al aprendizaje de los cursos de sistemas de control mediante la experimentación con sistemas físicos a través de una plataforma segura y fácil de usar. El repositorio de guías, en conjunto con la interfaz gráfica en *Python*, asegurarán que el estudiante pueda adquirir las competencias básicas de los cursos de Sistemas de Control 1 (control clásico) y Sistemas de Control 2 (control moderno), a través de la creación e implementación de los controladores planteados en ambos cursos.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar e implementar una aplicación de *software* que permita utilizar una plataforma física creada para el diseño, configuración y estudio de sistemas de control de orientación para el dron Crazyflie 2.0.

4.2. Objetivos específicos

- Realizar un manual de operación que permita configurar y utilizar correctamente la plataforma y el dron Crazyflie 2.0.
- Desarrollar e implementar una interfaz gráfica en *Python* para poder realizar las pruebas de control de la plataforma para el Crazyflie 2.0, creada en la fase anterior del proyecto.
- Desarrollar un repositorio de experimentos que permita configurar y estudiar los sistemas de control de orientación del dron Crazyflie 2.0 en la plataforma de trabajo que pueda emplearse dentro de los laboratorios de los cursos de Sistemas de Control 1 y 2.

CAPÍTULO 5

Alcance

El alcance de este proyecto se define en la implementación en software de la plataforma fabricada por Martínez [5] para el dron *Crazyflie 2.0* trabajada en la Universidad del Valle de Guatemala. Dentro de esta implementación se propone la realización de 2 guías de laboratorio donde se utilice el *Crazyflie 2.0* como sistema de estudio (una planta, hágase de sistemas de control) para los cursos de Sistemas de Control 1 y Sistemas de Control 2 impartidos en la Universidad del Valle de Guatemala.

Se realiza una guía de laboratorio para el curso de Sistemas de Control 1, donde se trabajará con la modificación del control PID del ángulo de cabeceo del dron. Esta guía de laboratorio tendrá como herramienta una interfaz gráfica en Python la cual recopilará los datos del codificador rotacional que posee la plataforma y controlará al dron.

La siguiente guía de laboratorio se realiza para el curso de Sistemas de Control 2, donde se trabaja con el diseño e implementación de un controlador LQR para el control de actitud, prestándole especial atención al ángulo de cabeceo, dado que es el único que será posible medir dado que la plataforma física solo permite el movimiento de un grado de libertad.

Finalmente se propone realizar un manual de usuario del *Crazyflie 2.0* para aprender a utilizarlo, volarlo, y a su vez, que indique cómo se debe de modificar su *firmware* a través de la configuración de una máquina virtual previamente creada.

Entre las limitaciones más importantes que afectó al desarrollo del proyecto fue las medidas sanitarias debidas a la pandemia del Covid-19, lo cual complicó y alentó el trabajo de laboratorio; esto tuvo un gran impacto para el desarrollo de las pruebas físicas y la obtención de los resultados.

CAPÍTULO 6

Marco teórico

Generalidades de un cuadricóptero

Un cuadricóptero es un helicóptero de geometría simple que se compone de 4 rotores controlados de forma independiente, montados en un marco rígido. La Figura 3 muestra uno de ellos, en donde se observa que el motor 1 y 3 rotan en sentido contrario a las agujas del reloj, mientras que el motor 2 y 4 giran en sentido de las agujas del reloj [6].

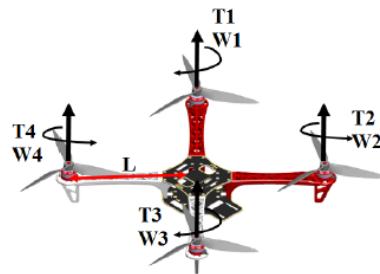


Figura 3: Cuadricóptero Simple [6].

Existen 2 configuraciones básicas para el caso del cuadricóptero, las cuales pueden observarse en la Figura 4.

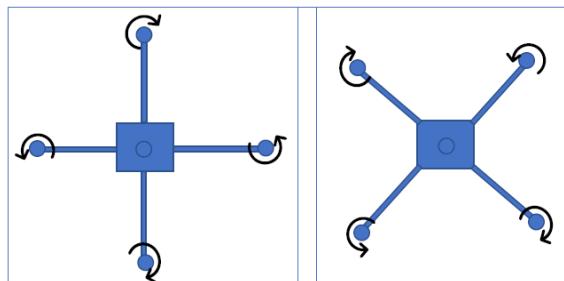


Figura 4: Configuración en cruz (izquierda). Configuración en equis (derecha) [7].

Ángulos de orientación en un cuadricóptero

Los ángulos de Euler son una forma intuitiva de poder representar la orientación en un dron, dado que su significado físico es bastante claro y muchos algoritmos de control los usan ampliamente. Según la teoría de Euler, la rotación de un cuerpo rígido alrededor de un punto fijo se puede considerar como varias rotaciones del punto fijo. Esto se puede lograr mediante 3 rotaciones elementales, donde cada eje de rotación es uno de los ejes de coordenadas del marco de coordenadas de rotación y cada ángulo de rotación es uno de los ángulos de Euler [8].

Para poder definir cada uno de los ángulos de orientación en el dron, es necesario definir los motores para la configuración en equis (x) del dron, para ello se hace referencia a la Figura. 5.

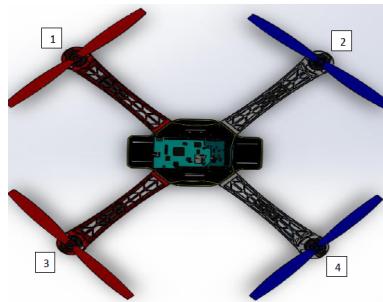


Figura 5: Enumerado de los 4 motores de un cuadricóptero en configuración equis (x) [7].

- **Ángulo de alabeo ϕ (roll):** Este movimiento es proporcionado por el aumento (o disminución) de velocidad de los motores 1 y 3, y por la disminución (o aumento) de velocidad de los motores de la derecha. Esto hace que se produzca un par de torsión con respecto al eje x que hace girar el cuadricóptero como se observa en la Figura 6a. [7]
- **Ángulo de cabeceo θ (pitch):** Este movimiento es muy similar al alabeo y se genera mediante el aumento (o disminución) la velocidad de los motores 1 y 2, y por la disminución (o aumento) de los motores de 3 y 4. Esto hace que se produzca un par de torsión con respecto al eje y que hace girar el cuadricóptero como se observa en la Figura 6b. [7]
- **Ángulo de guiñada Ψ (yaw):** Este movimiento es proporcionado por el aumento (o disminución) velocidad de los motores 1 o 2 y por la disminución (o aumento) de los motores 3 o 4. Esto conduce a un par de torsión con respecto al eje z que hace girar el cuadricóptero como se observa en la Figura 6c. El movimiento de guiñada se genera gracias a que las hélices de izquierda y derecha giran en sentido horario mientras que las hélices delantero-trasero giran en sentido antihorario [7].



Figura 6: Ángulos de euler en un cuadricóptero [7].

Crazyflie 2.0

El Crazyflie 2.0, Figura 7, es el segundo dron en miniatura de código abierto desarrollado por Bitcraze AB, el cual es utilizado para investigación por su versatilidad, bajo costo y soporte para desarrollo. Este dron ligero de 4 rotores, está equipado con radio de baja latencia y de largo alcance, así como Bluetooth LE para poder comunicarse. El dron también está equipado por múltiples sensores, en lo que se incluye la IMU MPU-9250 con un giroscopio, acelerómetro y magnetómetro de 3 ejes, y un barómetro (LPS25H).

El Crazyflie cuenta con varias plataformas de expansión con un peso máximo recomendado de 15 gramos, que permiten proporcionarle mayor funcionalidad, como sensores. Con la batería completamente cargada puede proporcionar un tiempo de vuelo de 7 minutos, y ésta batería dura 40 minutos en recargarse.

A continuación se presentan algunos detalles técnicos de la plataforma [9]:

Características:

- Diseño duradero.



Figura 7: Crazyflie 2.0 [1].

- Fácil de montar y no requiere de soldadura.
- Actualizaciones de firmware inalámbricas.
- Carga a bordo a través de uUSB estándar.

Especificaciones Mecánicas:

- Masa de despegue de 27g.
- Tamaño: 92 mm x 92 mm x 92 mm (medidas de rotor a rotor).

Especificaciones de radio:

- Radio de banda ISM de 2.4 GHz.
- Amplificador de radio de 20 dBm probado a más de 1 km de distancia con Crazyradio PA.
- Compatibilidad con Bluetooth Low Energy con clientes iOS y Android disponible.
- Radio compatible con versiones anteriores de Crazyflie Nano y Crazyradio originales.

Microcontroladores:

- MCU de aplicación principal STM32F405 (Cortex-M4, 168MHz, 192kb SRAM, flash de 1Mb).
- MCU de administración de energía y radio nRF51822 (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash).
- Conector USB.
- Cargador LiPo a bordo con modos 100mA, 500mA y 980mA disponibles.
- Capacidad USB OTG parcial.
- EEPROM de 8 KB.

Modelado matemático del *Crazyflie 2.0*

Para simplificar el análisis y la determinación del modelo matemático sin alejarse del modelo físico real [7], se asume lo siguiente:

- El cuadricóptero es un cuerpo rígido.
- La masa y los momentos de inercia son constantes.
- El centro geométrico y el centro de gravedad del cuadricóptero es el mismo.

Previo a analizar en análisis matemático se muestra una tabla que facilitará la lectura de variables y ayudará al lector comprender el desarrollo matemático del sistema.

Vector	Variable	Descripción
ξ	x	Posición X del dron respecto al marco de referencia inercial
	y	Posición Y del dron respecto al marco de referencia inercial
	z	Posición Z del dron respecto al marco de referencia inercial
η	ϕ	Ángulo de balanceo
	θ	Ángulo de cabeceo
	ψ	Ángulo de guiñada
V^B	u	Velocidad lineal en X del dron respecto al marco de referencia inercial
	v	Velocidad lineal en Y del dron respecto al marco de referencia inercial
	w	Velocidad lineal en Z del dron respecto al marco de referencia inercial
ω^B	p	Velocidad de balanceo del dron respecto al marco de referencia inercial
	q	Velocidad de cabeceo del dron respecto al marco de referencia inercial
	r	Velocidad de guiñada del dron respecto al marco de referencia inercial

Cuadro 1: Variables involucradas en la dinámica del dron

El análisis matemático, en su mayoría, fue obtenido del trabajo previo de Silino & Iannelli [10] y de Guerra [7]; ambos trabajos aportando al desarrollo del modelado del Crazyflie 2.0.

Como primer paso se debe de definir el marco de referencia Inercial O_{FI} (que corresponde al marco físico) y el marco de referencia del dron, el cual está fijo al cuerpo rígido del dron tal como se muestra en la Figura 8.

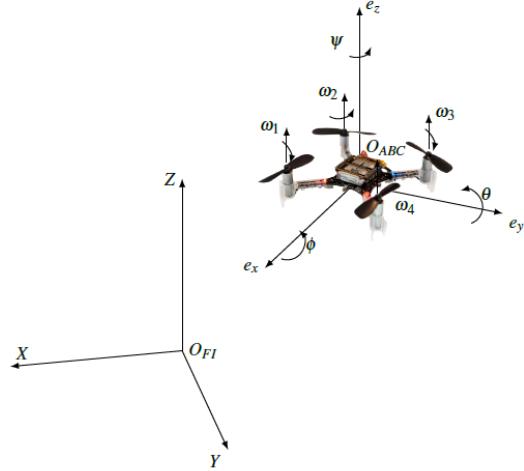


Figura 8: Marco de referencia en el Crazyflie con el marco de referencia inercial [10].

La posición absoluta del dron se define en el marco de referencia Inercial O_{FI} mediante ξ , mientras que la posición angular con los ángulos de Euler mediante η . El vector q contiene tanto a la posición lineal como angular.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix}. \quad (1)$$

El origen del sistema de referencia del dron está situado en el centro de masa del mismo. En este sistema las velocidades lineales se determinan mediante V^B [m/s] y las angulares mediante ω^B [rad/s]. Entonces, se tiene un vector de velocidades el cual se expresa mediante \vec{v} .

$$\vec{v} = [V^B \ \ \omega^B]^T = [u \ \ v \ \ w \ \ p \ \ q \ \ r]^T. \quad (2)$$

La matriz de rotación del sistema de referencia respecto al sistema de referencia inercial es, según [7].

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi C_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi C_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}. \quad (3)$$

En donde $C_x = \cos(x)$, $S_x = \sin(x)$ y R es una matriz ortogonal, es decir $R^{-1} = R^T$, donde esta inversa es la matriz de rotación desde el sistema de referencia inercial al sistema de referencia del cuadricóptero.

Es posible mapear las velocidades angulares desde el sistema de referencia inercial al marco de referencia del cuadricóptero, mediante la matriz de transformación denotada por W_η .

$$\vec{v} = W_\eta \dot{\eta} \Rightarrow \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \quad (4)$$

O bien, obtener las velocidades angulares desde el marco de referencia del cuadricóptero al marco de referencia inercial.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi/C_\theta & C_\phi/C_\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (5)$$

Ecuaciones de Newton - Euler

Una vez definidas ambas velocidades en ambos marcos de referencia se procede a determinar los actuadores del dron y su relación con la dinámica del sistema. Según [10] la fuerza de empuje que genera el cuadricóptero se define mediante:

$$F_z = C_T (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2). \quad (6)$$

A su vez, los momentos de torsión. según [10]. que puede generar el dron se define como:

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} C_T d (-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ C_T d (-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \\ \sqrt{2} C_M (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \quad (7)$$

Donde d es la distancia desde el centro de masa del dron al eje del rotor, ω_i es la velocidad angular del i -ésimo motor, C_T y C_M son las constantes de empuje y momento del rotor respectivamente.

Ecuación de Newton para la posición

En este caso es conveniente definir las ecuaciones de Newton desde el marco de referencia inercial, puesto que resulta más fácil plantearlo, de ello se obtiene la siguiente ecuación.

$$m \ddot{\xi} = G + R T_B, \quad (8)$$

donde m es la masa del cuadricóptero, G es el fuerza de la gravedad, y T_B es la fuerza total de empuje de los 4 rotores, las cuales se definen mediante:

$$T_B = \begin{bmatrix} 0 \\ 0 \\ F_z \end{bmatrix}, \quad (9)$$

$$G = \begin{bmatrix} 0 \\ 0 \\ -g \cdot m \end{bmatrix}. \quad (10)$$

Operando la ecuación se obtiene la siguiente expresión:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{F_z}{m} \begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi C_\phi \\ S_\psi S_\theta C_\phi - C_\psi S_\phi \\ C_\psi S_\phi \end{bmatrix}. \quad (11)$$

Ecuación de Euler para la orientación:

Primero es necesario determinar las inercias del cuadricóptero y, dado que el Crazyflie 2.0 es simétrico, la matriz de representa mediante:

$$I_{dron} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \quad (12)$$

Para este sistema, en [7], se considera la siguiente ecuación, que describe la dinámica de orientación del dron, planteada desde el marco de referencia del cuadricóptero.

$$I\dot{v} + v \times (Iv) + \Gamma = \tau. \quad (13)$$

Donde $I\dot{v}$ es la aceleración angular de la inercia, $v \times (Iv)$ son las fuerzas centrípetas, $+\Gamma$ las fuerzas giroscópicas y τ el torque externo. Resolviendo para \dot{v} se obtiene la siguiente ecuación, descrita en [7].

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz}) qr/I_{xx} \\ (I_{zz} - I_{xx}) qr/I_{yy} \\ (I_{xx} - I_{yy}) qr/I_{zz} \end{bmatrix} - I_r \begin{bmatrix} \frac{q}{I_{xx}} \\ -\frac{p}{I_{yy}} \\ 0 \end{bmatrix} w_r + \begin{bmatrix} \frac{\tau_\phi}{I_{xx}} \\ \frac{\tau_\theta}{I_{yy}} \\ \frac{\tau_\psi}{I_{zz}} \end{bmatrix}. \quad (14)$$

Con las ecuaciones de Newton-Euler se describe por completo dinámica del dron, lo cual será de utilidad para poder realizar las simulaciones del sistema.

Parámetros físicos:

Los parámetros físicos fueron obtenidos gracias a un estudio realizado en [11], lo cual facilitó el modelado del sistema.

Parámetros	Descripción	Valor
m	Masa total	$0.27[Kg]$
d	Largo del brazo	$39.73 \times 10^{-3}[m]$
r	Radio de rotor	$23.1348 \times 10^{-3}[m]$
I_{xx}	Principal Momento de Inercia alrededor del eje x	$1.395 \times 10^{-5}[Kg \times m^2]$
I_{yy}	Principal Momento de Inercia alrededor del eje y	$1.436 \times 10^{-5}[Kg \times m^2]$
I_{zz}	Principal Momento de Inercia alrededor del eje z	$2.173 \times 10^{-5}[Kg \times m^2]$
K_T	Coeficiente de empuje adimensional	0.2025
K_D	Coeficiente de empuje adimensional	0.11

Cuadro 2: Parámetros físicos del Crazyflie 2.0 [11].

Con estos parámetros ya obtenidos se procede a calcular el coeficiente de empuje C_T y el coeficiente de Torque C_D , estos mediante las fórmulas establecidas en [11]. Antes se define la densidad del aire correspondiendo a $\rho = 1.255[Kg/m^3]$.

$$C_T = k_T \rho (2r)^4 / 3600 = 3.1582 \times 10^{-10}[N/rpm^2]. \quad (15)$$

$$C_D = k_D \rho (2r)^5 / 3600 = 7.9379 \times 10^{-12}[Nm/rpm^2]. \quad (16)$$

Linealización y representación en el espacio de estados:

La representación de un sistema lineal en espacio de estados está dado por el siguiente sistema de ecuaciones, segúin [11]:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) \end{cases} \quad (17)$$

En general, esta ecuación describe la evolución de un sistema lineal variante en el tiempo, donde $x(t)$ es el vector de estados, $y(t)$ es el vector de salida y $u(t)$ es el vector de entrada. Según [11], para la representación del espacio de estados de un cuadricóptero, es convencional considerar la siguiente realización lineal invariante en el tiempo del sistema, lo que significa que las matrices A, B, C y D son estáticas y no cambian con el tiempo:

$$\begin{cases} \Delta\dot{x} = A\Delta x + B\Delta u \\ \Delta y = C\Delta x + D\Delta u \end{cases} \quad (18)$$

Donde el prefijo Δ significa que el vector es el resultado de un proceso de linealización. Para poder linealizar al sistema, debe hacerse alrededor de un punto, este punto será donde el sistema deja de evolucionar en el tiempo y esto se logra en el vuelo estacionario a cierta altura. A este hecho se le conoce como estado de equilibrio y se representa mediante:

$$x_e = [x_e \ y_e \ z_e \ \psi_e \ \theta_e \ \phi_e \ u_e \ v_e \ w_e \ r_e \ q_e \ p_e]^T. \quad (19)$$

En términos de cálculo de linealización, la posición lineal y el ángulo de guiñada del cuadricóptero son indiferentes, por lo que se consideran como constantes arbitrarias. En caso de los ángulos de balanceo y cabeceo, deben ser cero para que el cuadricóptero mantenga la posición estacionaria, al igual que cualquier velocidad lineal o angular, por lo que el vector de estado de equilibrio resulta ser:

$$x_e = [x_e \ y_e \ z_e \ \psi_e \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T. \quad (20)$$

En estado de equilibrio, la única fuerza que está actuando sobre el cuadricóptero es la fuerza de la gravedad, y para que el cuadricóptero mantenga este estado, el empuje debe de ser igual a la gravedad, por lo que se tiene:

$$C_T (\omega_{e1}^2 + \omega_{e2}^2 + \omega_{e3}^2 + \omega_{e4}^2) = mg. \quad (21)$$

Bajo la hipótesis de que el cuadricóptero es perfectamente simétrico, las aspas giran a la misma velocidad para mantener al cuadricóptero en equilibrio. Al final se obtiene la siguiente ecuación:

$$\omega_e = \sqrt{\frac{mg}{4C_T}} = 16073[\text{rpm}]. \quad (22)$$

Dado que los motores son las entradas, se obtiene el siguiente vector de entradas:

$$u_e = [\omega_e \ \omega_e \ \omega_e \ \omega_e]^T. \quad (23)$$

Luego de realizar un proceso de linealización, según [11], los resultados se observan en la Figura 9

$$\begin{bmatrix}
\Delta \dot{x} \\
\Delta \dot{y} \\
\Delta \dot{z} \\
\Delta \dot{\psi} \\
\Delta \dot{\theta} \\
\Delta \dot{\phi} \\
\Delta \dot{u} \\
\Delta \dot{v} \\
\Delta \dot{w} \\
\Delta \dot{r} \\
\Delta \dot{q} \\
\Delta \dot{p}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta z \\
\Delta \psi \\
\Delta \theta \\
\Delta \phi \\
\Delta u \\
\Delta v \\
\Delta w \\
\Delta r \\
\Delta q \\
\Delta p
\end{bmatrix} + \omega_e \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
2C_T/m & 2C_T/m & 2C_T/m & 2C_T/m \\
-2C_D/I_{zz} & 2C_D/I_{zz} & -2C_D/I_{zz} & 2C_D/I_{zz} \\
-\sqrt{2}dC_T/I_{yy} & \sqrt{2}dC_T/I_{yy} & \sqrt{2}dC_T/I_{yy} & -\sqrt{2}dC_T/I_{yy} \\
-\sqrt{2}dC_T/I_{xx} & -\sqrt{2}dC_T/I_{xx} & \sqrt{2}dC_T/I_{xx} & \sqrt{2}dC_T/I_{xx}
\end{bmatrix} \begin{bmatrix}
\Delta \omega_1 \\
\Delta \omega_2 \\
\Delta \omega_3 \\
\Delta \omega_4
\end{bmatrix}$$

Figura 9: Resultados de linealización de un cuadricóptero [11]

Desacoplamiento del movimiento

En la realización del espacio de estados del modelo lineal del cuadricóptero, las cuatro entradas del sistema actúan directamente en solo cuatro estados del sistema tal como se observa a continuación:

$$\begin{bmatrix}
F_z \\
M_x \\
M_y \\
M_z
\end{bmatrix} = 2\omega_e \begin{bmatrix}
C_T & C_T & C_T & C_T \\
-dC_T/\sqrt{2} & -dC_T/\sqrt{2} & dC_T/\sqrt{2} & dC_T/\sqrt{2} \\
-dC_T/\sqrt{2} & dC_T/\sqrt{2} & dC_T/\sqrt{2} & -dC_T/\sqrt{2} \\
-C_D & C_D & -C_D & C_D
\end{bmatrix} \begin{bmatrix}
\Delta \omega_1 \\
\Delta \omega_2 \\
\Delta \omega_3 \\
\Delta \omega_4
\end{bmatrix}. \quad (24)$$

Se puede definir una matriz de transformación entre las fuerzas que actúan sobre el cuerpo del cuadricóptero y la velocidad angular de los motores:

$$\Gamma = 2\omega_e \begin{bmatrix}
C_T & C_T & C_T & C_T \\
-dC_T/\sqrt{2} & -dC_T/\sqrt{2} & dC_T/\sqrt{2} & dC_T/\sqrt{2} \\
-dC_T/\sqrt{2} & dC_T/\sqrt{2} & dC_T/\sqrt{2} & -dC_T/\sqrt{2} \\
-C_D & C_D & -C_D & C_D
\end{bmatrix}. \quad (25)$$

Puesto que la Matriz Γ es invertible,

$$\mathbf{\Gamma}^{-1} = \frac{1}{2\omega_e} \begin{bmatrix}
1/(4C_T) & -\sqrt{2}/(4dC_T) & -\sqrt{2}/(4dC_T) & -1/(4C_D) \\
1/(4C_T) & -\sqrt{2}/(4dC_T) & \sqrt{2}/(4dC_T) & 1/(4C_D) \\
1/(4C_T) & \sqrt{2}/(4dC_T) & \sqrt{2}/(4dC_T) & -1/(4C_D) \\
1/(4C_T) & \sqrt{2}/(4dC_T) & -\sqrt{2}/(4dC_T) & 1/(4C_D)
\end{bmatrix}. \quad (26)$$

Es posible encontrar las velocidades angulares de cada motor en relación a las fuerzas y momentos deseados, por lo que se tiene:

$$\begin{bmatrix}
\Delta \omega_1 \\
\Delta \omega_2 \\
\Delta \omega_3 \\
\Delta \omega_4
\end{bmatrix} = \frac{1}{2\omega_n} \begin{bmatrix}
1/(4C_T) & -\sqrt{2}/(4dC_T) & -\sqrt{2}/(4dC_T) & -1/(4C_D) \\
1/(4C_T) & -\sqrt{2}/(4dC_T) & \sqrt{2}/(4dC_T) & 1/(4C_D) \\
1/(4C_T) & \sqrt{2}/(4dC_T) & \sqrt{2}/(4dC_T) & -1/(4C_D) \\
1/(4C_T) & \sqrt{2}/(4dC_T) & -\sqrt{2}/(4dC_T) & 1/(4C_D)
\end{bmatrix} \begin{bmatrix}
F_z \\
M_x \\
M_y \\
M_z
\end{bmatrix} \quad (27)$$

La ecuación anterior dicta cómo cada motor contribuye a cada una de las fuerzas que actúan sobre el cuerpo del cuadricóptero. Este estudio confirma que las fuerzas verticales, laterales, longitudinales y direccionales (guiñada) actúan de forma independiente entre sí en el modelo matemático y, por lo tanto, la dinámica del cuadricóptero se desacopla y se puede estudiar como subsistemas.

Subsistema vertical: Describe la dinámica de los movimientos ascendentes del robot (los cambios en la distancia z). Se represente mediante la siguiente ecuación.

$$\begin{bmatrix} \Delta\dot{w} \\ \Delta\dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta z \end{bmatrix} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} \Delta F_z. \quad (28)$$

Subsistema direccional: El ángulo de guiñada y su velocidad dictan la dinámica del cuadricóptero en el plano XY, mediante la siguiente ecuación en el espacio de estados.

$$\begin{bmatrix} \Delta\dot{r} \\ \Delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta r \\ \Delta\psi \end{bmatrix} + \begin{bmatrix} 1/I_{zz} \\ 0 \end{bmatrix} \Delta M_z. \quad (29)$$

Subsistema lateral: La dinámica lateral gobierna el movimiento de cabeceo del cuadricóptero, así como su posición Y en el marco inercial:

$$\begin{bmatrix} \Delta\dot{p} \\ \Delta\dot{\phi} \\ \Delta\dot{v} \\ \Delta\dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta\phi \\ \Delta v \\ \Delta y \end{bmatrix} + \begin{bmatrix} 1/I_{xx} \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta M_x. \quad (30)$$

Subsistema longitudinal: Rige el movimiento alrededor del eje X del marco fijo del cuadricóptero, y su posición X y velocidad en el marco inercial.

$$\begin{bmatrix} \Delta\dot{q} \\ \Delta\dot{\theta} \\ \Delta\dot{u} \\ \Delta\dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta\theta \\ \Delta u \\ \Delta x \end{bmatrix} + \begin{bmatrix} 1/I_{yy} \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta M_x. \quad (31)$$

Caracterización de motores: En la sección anterior se determinó que la entrada del sistema dinámico en espacio de estados corresponde a velocidades angulares de los motores dadas en revoluciones por minuto [RPM], pero explorando el firmware del Crazyflie 2.0 se determinó que ésta no es la entrada real del sistema, sino un PWM de 19 bits correspondiendo a un valor de 0 a 65535. En [12], se determinó la relación existente entre el PWM y las RPM de cada motor lo cual se presenta en la siguiente ecuación:

$$RPM = 0.2685PWM + 4070.3 \quad (32)$$

La Ecuación 32 es específicamente para el dron *Crazyflie 2.0* y fue necesaria obtenerla debido a que no se usaron motores sin escobillas. Dentro de las suposiciones para el experimento realizado por [12] se considera un tiempo de respuesta rápido, a manera de simplificar el sistema, sabiendo que esto no es cierto pero es una buena aproximación.

Máquina virtual Bitcraze

Una máquina virtual, tal como se indica en [13] se encarga de emular a un ordenador completo, de manera que puede ejecutar otro sistema operativo en su interior. Esta tiene su propio disco duro, memoria, tarjeta gráfica y demás componentes de hardware solo que virtuales. Sin embargo, a pesar de ello, la máquina virtual tiene reservados recursos del anfitrión (ordenador donde la máquina virtual está instalada). En la Figura 10 se muestra el diagrama de funcionamiento de una máquina virtual.

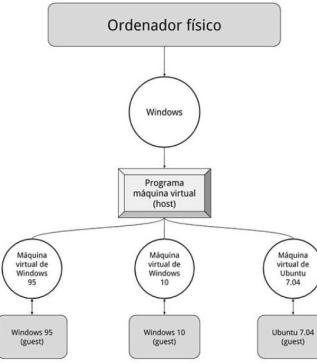


Figura 10: Árbol de máquinas virtuales [13].

A manera de facilitar el desarrollo de proyectos utilizando el Crazyflie 2.0, Bitcraze ha creado y preconfigurado una máquina virtual que contiene los requisitos previos necesarios para la creación y uso de proyectos en Bitcraze. Esto evita instalar dependencias y configurar el entorno de desarrollo [14].

Codificador rotacional óptico

El codificador rotacional es un sensor de movimiento que genera una respuesta al movimiento, esta respuesta en forma de una señal PWM. Brinda información acerca de la posición y velocidad angular, así como el sentido de rotación. Existen dos tipos de codificadores ampliamente utilizados: los incrementales y los absolutos. Los incrementales se utilizan cuando solo se requiere la posición relativa o en el caso de que el precio sea un problema; mientras que los absolutos dan la posición específica del eje dado que contienen información binaria en cada posición angular [15].

Los codificadores rotacionales ópticos son uno de más utilizados (atrás de los magnéticos), poseen una LED IR que brilla a través de un disco de codificación con líneas que interrumpen el haz de luz del fotosensor, y de esta manera se crea la señal que un microcontrolador debe de interpretar [15]. Su funcionamiento puede observarse en la Figura 11.

El codificador rotacional utilizado en la plataforma creada por Martínez [5] es un LPD3806-600BM-G5-24C, el cual es de tipo incremental con 600 pulsos por revolución y una frecuencia de respuesta de 0 a 20KHz; su voltaje de alimentación es de 5 a 24 voltios.

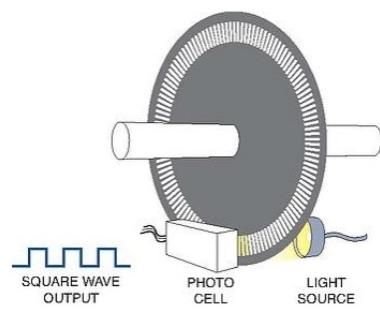


Figura 11: Codificado rotacional óptico [15].

CAPÍTULO 7

El Crazyflie 2.0 y su máquina virtual

El presente capítulo incluye el procedimiento que se tuvo que realizar para aprender a utilizar el Crazyflie 2.0, lo que correspondía a verificar la integridad del mismo dado que había sido utilizado con anterioridad. Además, incluye una breve descripción de las pruebas de vuelo realizadas; la instalación, uso y reparación de errores de la máquina virtual utilizada, así como los paquetes que fueron necesarios instalar para poder utilizar el dron; las pruebas realizadas con la máquina virtual y el uso del Crazyflie Client como aplicación que valida el correcto funcionamiento de la antena de radio y que permitió descargarle la actualización del *firmware* del dron necesaria para posteriormente poder modificarlo a través de *Visual Studio Code*, que es la plataforma de programación utilizada en la última máquina virtual trabajada.

7.1. Descripción del Crazyflie 2.0 y validación de funcionamiento

Antes de la experimentación fue necesario conocer las características del dron, leyendo la documentación del mismo e interpretando la información proveniente de cada luz LED, datos que se encuentran en el manual de operación del dron disponible en el Anexo 13.2. Se verificó que contenía todos los componentes mencionados en su descripción y que su aspecto físico no mostraba ninguna señal de daño. Por otra parte, dado que fue armado previamente, se verificó que fuese armado correctamente.

Verificación de integridad del dron construido

Luego de conocer el dron y determinar que estuviese completo, lo primero que se realizó fue verificar que el dron funcionara correctamente. En este paso era importante validar cada componente, dado que el Crazyflie había sido utilizado previamente y había estado almacenado por 2 años. Para ello fue necesario realizar los pasos que se describen en la sección de testeo de la página de *Bitcraze* [16].

Lo que se verificó fue lo siguiente:

- **Encendido y todo está bien:** Se conectó la batería y se presionó el botón de encendido. Se verificó que los LED azules (2 y 3) están completamente iluminados y el LED frontal derecho (1) parpadea en rojo dos veces por segundo, lo cual significa que pasó la prueba. Este estado se pudo ver luego de que los sensores se calibraran dentro de la rutina del dron.
- **Encendido y todo está bien, pero los sensores aún no están calibrados:** Los LED azules (2 y 3) están completamente iluminados y el LED frontal derecho (1) parpadea en rojo con un intervalo de 2 segundos. Para ello se colocó el dron en una superficie nivelada y se dejó quieto completamente hasta que se pudo observar la secuencia anterior.
- **Batería baja:** No solo se verificó el dron a carga completa sino en descarga. En este caso se el LED frontal derecho (1) está completamente iluminado en rojo.
- **Cargando:** Para cargar el dron hay que colocarlo en modo de cargador de arranque, presionando el botón de encendido por aproximadamente 3 segundos hasta observar que los LED azules (2 y 3) en la parte posterior parpadean aproximadamente una vez por segundo. Seguidamente se conecta el cable micro-USB hasta observar que el LED azul trasero izquierdo (3) parpadea mientras que el LED azul trasero derecho (2) está encendido.

Armando un nuevo dron

A pesar de ya tener un dron construido y funcional, también se decidió armar otro a manera de tener una guía en caso fuese necesario hacer uso de todos los disponibles en la Universidad del Valle de Guatemala. Los componentes disponibles para armar el *Crazyflie 2.0* se muestran en la Figura 12.

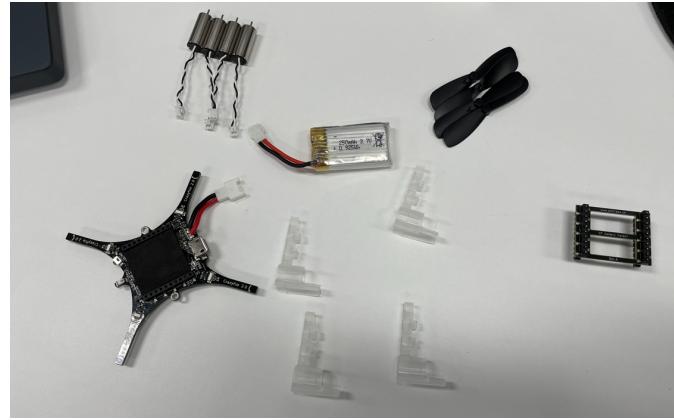


Figura 12: Componentes del *Crazyflie 2.0*.

Lo primero que se realizó fue verificar que la placa del dron no estuviera dañada o presentara algún error, para ello se conectó el cable micro USB y se comprobó la rutina de LEDs de **Encendido y todo está bien** que se describió con anterioridad.

Seguidamente se armaron los motores, introduciéndolos sobre su base de plástico y trenzando sus dos cables para que se encontraran bien asegurados, tal como se muestra en la Figura 13.

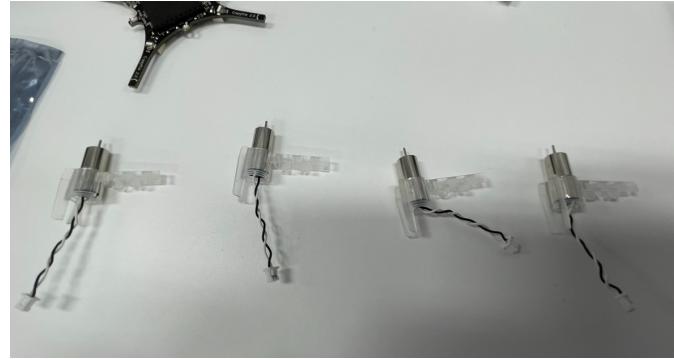


Figura 13: Armado de motores del *Crazyflie 2.0*.

Luego, se colocaron los motores sobre la estructura del dron y se conectaron a sus respectivos puertos, tal como se muestra en la Figura 14.

Luego de conectar correctamente cada motor se colocaron las aspas, dos de ellas que giraran en sentido del reloj, las cuales tienen impresa una letra “A” y el sentido de giro se observa en la estructura, y dos de ellas que giran en contra del sentido de las agujas del reloj, denotadas con la letra “B”. A su vez, se conectó la batería al dron. El resultado final del dron armado se observa en la Figura 15.



Figura 14: Colocado de motores del *Crazyflie 2.0*.



Figura 15: Dron completamente armado.

Prueba de vuelo

Verificada ya la integridad física del dron y que las indicaciones fueran las correctas se procedió a realizar pruebas de vuelo. Estas pruebas fueron realizadas utilizando la aplicación para celular proporcionada por la compañía *Bitcrze* denominada *Crazyflie 2.0* que se encuentra disponible para *iOS* y *Android*. Esta aplicación se conecta mediante *Bluetooth* al *Crazyflie*, con ella se estuvieron realizando pruebas de vuelo para verificar que también era capaz de volar previo a la experimentación del mismo, lo que determinaba un correcto funcionamiento del *firmware* del dron y del módulo de conexión *Bluetooth*.



Figura 16: Conexión del Crazyflie al celular.

Una vez instalada la aplicación se conectó al celular presionando el botón de “*Connect*” que se observa en la Figura 16, y a partir de ello se procedió a volar el dron. Esto se tuvo que realizar en un campo al aire libre con viento calmo, para evitar accidentes y evitar que el *Crazyflie* se dañara ya que ante cualquier caída éste aterrizaría sobre grama, tal como se observa en la Figura 17.



Figura 17: Prueba de vuelo.

7.2. Instalación y configuración de la máquina virtual

Una vez familiarizado físicamente con el *Crazyflie* 2.0 y la forma de volar utilizando un teléfono inteligente, además de verificar que esté funcionando correctamente; se procedió a estudiar el *firmware* de este mediante la documentación presentada por *Bitcraze*. Para ello, fue necesario la instalación de una máquina virtual, la cual contiene todas las herramientas necesarias para poder cargarle modificaciones a la programación del dron.

Para poder instalar la máquina virtual fue necesario instalar una aplicación que permita la virtualización, siendo esta **Virtualbox** la cual se instaló desde el sitio oficial de la aplicación. Para ello se siguieron los pasos presentados en el Anexo 13.2.

7.2.1. Máquina virtual Bircraze

Una vez instalado el virtualizador, se procedió a descargar la máquina virtual correspondiente para poder trabajar el *software* del *Crazyflie*. Se descargaron dos máquinas virtuales, una lanzada en el 2018 y la siguiente fue lanzada mientras se trabajaba con el dron en abril del 2021. Dado que se trabajó con ambas se hablará de ellas, las cuales fueron descargadas del repositorio del *Github* de Bitcraze para la máquina virtual.

Bitcraze VM 2018.12

Se realizó el procedimiento para instalar la máquina virtual del 2018, la cual presentaba varios problemas con el *Crazyflie Client* y con el repositorio de Repositorio de Github que contenía todos los códigos de Bitcraze, entre problemas con falta de paquetes.

La versión 2018.12 utilizaba Eclipse como editor de texto y ya incluía pre instalado los paquetes para poder compilar y flashear el firmware del Crazyflie y también el compilador de C. A su vez ya incluía los proyectos para los demás microcontroladores utilizados en conjunto con el dron.

Uno de los problemas era que no se clonaba bien el repositorio de *Github* y el mismo no se encontraba sincronizado ni actualizado, para ello a través de la consola de la máquina virtual se tuvieron que correr los siguientes comandos:

```
git submodule init
git submodule update
git submodule sync
git submodule update
```

Código 7.1: Inicialización de repositorio de **Github** de *Bitcraze*

A la hora de iniciar el *Crazyflie Client* tampoco lo hacía, por ello fue necesario ejecutar las siguientes líneas de código.

Además los scripts de Python no podían ejecutarse desde Eclipse, por lo que era necesario utilizar la consola para poder correrlos, lo que implicaba buscar el archivo y luego escribir la línea de comandos que lo ejecutaba, lo cual se consideraba un proceso ineficiente a la hora de hacer modificaciones.

Bitcraze VM 2021.04

Revisando el repositorio de *GitHub* de la máquina virtual, se determinó la existencia de una nueva versión de Máquina Virtual, la cual traía consigo cambios considerables. Uno de ellos era la actualización de Ubuntu a la versión 20.01, por otra parte, se arreglaron los problemas anteriores a la inicialización de *Crazyflie Client* y el repositorio de GitHub de código, se eliminó Eclipse y se empezó a utilizar *Visual Studio Code* como ambiente de desarrollo, lo cual facilitó a gran medida la edición de código por la familiarización con el editor de texto y la gran disponibilidad de paquetes de código abierto desarrollados por la comunidad.

Se procedió a descargar esta nueva máquina virtual y agregarla al virtualizador, tal como se describe en la guía de instalación de la máquina virtual. Uno de las tareas que se realizó era validar lo trabajado con la versión anterior, esto implicó realizar las siguientes tareas.

- Actualizar los códigos presentados por Bitcraze a través de su comando en el escritorio.
- Iniciar el *Bitcraze Client* y verificar que la aplicación abra correctamente.
- Iniciar VSCode y verificar que se encuentren todos los códigos disponibles.
- Conectar el Crazyflie PA (la antena de radio USB) y verificar que funcione correctamente.

Contenido de la máquina virtual

Es importante identificar las herramientas que posee la máquina virtual, a manera de aprovecharlos y utilizarlos de la mejor manera en el trabajo desarrollado.

Paquetes y compiladores: Debido a que todos los controladores están programados en C, la máquina virtual viene con el compilador de C capaz de compilar los códigos para los microcontroladores utilizados en la plataforma, además cuenta con todos los paquetes necesarios para poder flashear dichos controladores.

Repositorio de Github con firmware de los controladores: Dentro de la máquina virtual viene clonado un repositorio de Github de los 2 controladores que contiene el dron, así como el controlador de la antena de radio para conectar el dron a la computadora. Estos códigos se encuentran con todas las librerías necesarias para cada controlador, los cuales se actualizan con tan solo presionar el ícono de sincronización que se encuentra en el escritorio, con nombre “*Update all project*”.

Aplicación Crazyflie Client: Esta aplicación es muy similar a la aplicación de Crazyflie 2.0 disponible para iOS, solo que presenta una interfaz más compleja con muchas más capacidades, de las cuales se hablará a detalle en la sección destinada para ello.

Python 3.7 configurado y su librería de Crazyflie: Debido a que la aplicación de Crazyflie Client fue realizada en Python, Bitcraze pone a disposición una librería capaz de controlar el Crazyflie con Python 3.7. Además brindan la documentación necesaria de la API para poder utilizar la librería y realizar nuevas aplicaciones.

Visual Studio Code: Para el desarrollo de código, la máquina virtual utiliza el editor de texto de Visual Studio Code. Este ya consta de extensiones previamente instaladas, de las cuales se incluye la de C/C++, Python 3.7, Cortex-Debug, Jupyter y Github, las cuales son de gran utilidad para el desarrollo de código.

Modificaciones de la máquina virtual

Se le realizaron ciertas modificaciones a la máquina virtual, esto para poder utilizar el codificador rotacional, entre otras configuraciones necesarias para el desarrollo de la interfaz

gráfica propuesta.

Aplicaciones adicionales:

Arduino: Debido a que la plataforma física consta de un Codificador Rotacional, se utilizó un Arduino Uno para leer los datos (el cual se programó utilizando Arduino desde la máquina virtual). Era necesario identificar el puerto que Arduino reconocía para el microcontrolador (Arduino Uno), y así poder habilitar el puerto serial dándole permisos para la comunicación serial mediante la consola de Ubuntu.

Librerías de Python: Se instalaron algunas librerías de Python necesarias para la aplicación que se desarrolló. Estas fueron instaladas mediante el gestor de paquetes **pip3** de Python que se encontraba previamente instalado. Estas fueron las siguientes:

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

7.2.2. Exportar la máquina virtual

Dado que se le realizaron modificaciones a la máquina virtual, y para evitar que cada estudiante tuviese que realizar dichas modificaciones, se exportó la máquina virtual configurada. El proceso para ello, consistió en los siguientes pasos.

1. Asegurarse de que la máquina virtual estuviese apagada.
2. Ingresar a la pestaña de archivos de VirtualBox (*file* en inglés).
3. Seleccionar la máquina virtual que se desea exportar.
4. Darle la dirección donde se desea guardar.
5. Presionar la tecla “Next” y luego presionar “Export”.

Con esto se tiene un archivo tipo **ova** el cual se puede importar a otra computadora como se describe en el manual de uso incluido en el Anexo 13.2.

7.3. Uso del Crazyflie Client

De manera similar a la aplicación *Crazyflie 2.0* disponible para celular, Crazyflie Client es una aplicación para computadora que permite interactuar con el dron, donde utiliza una antena de radio denominada *Crazyradio PA* la cual se muestra en la Figura 18.



Figura 18: Antena de Radio *Crazyradio PA* [17].

Se inicializó la aplicación, la cual se encontraba en el escritorio de la máquina virtual, se verificó que el dispositivo de radio estuviese conectado al puerto de la máquina virtual y no a la computadora anfitriona. Una vez comprobada la conexión de la antena y la correcta inicialización de la máquina virtual se procedió a conectar el dron.

7.3.1. Conexión del Dron a la aplicación

Lo primero que se debía hacer era poder escanear la dirección de radio asignada por defecto al dron, para ello se presionó la tecla de “Scan” en la esquina superior izquierda. Una vez encontrada la interfaz de radio utilizada, se seleccionó en la pestaña de “Select Interface”, para finalmente presionar la tecla de “Connect”.

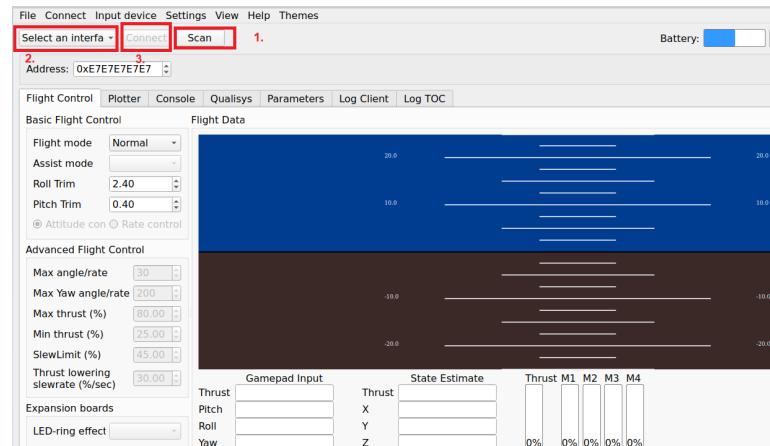


Figura 19: Escaneo de Crazyflie.

Una vez conectado se habilitará el botón de “Disconnect” y aparecerán datos como el voltaje de la batería y el horizonte artificial se habilitará indicando de que ya está conectado correctamente. Por otra parte, en el Crazyflie la luz LED m3 se prenderá de color verde

indicando que el dron se encuentra transmitiendo datos.

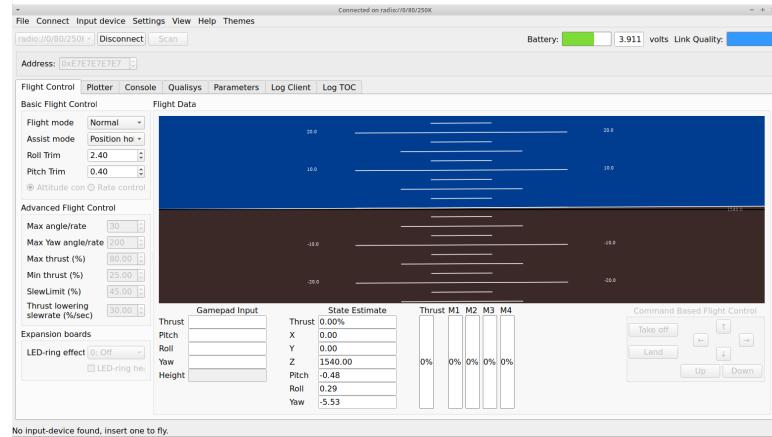


Figura 20: Conectar dron al Crazyflie Client.

7.3.2. Modificaciones iniciales al dron

Como se mencionó con anterioridad, los drones habían estado mucho tiempo guardados, por lo que fue necesario realizar una actualización a su firmware al más reciente. Para ello se utilizó el Crazyflie Client, realizando lo siguiente.

Lo primero en realizarse es moverse a la ventana de “Bootloader” que se encuentra en la pestaña de “Connect” y seleccionando la opción de “Bootloader”, tal como se muestra en la Figura 21.

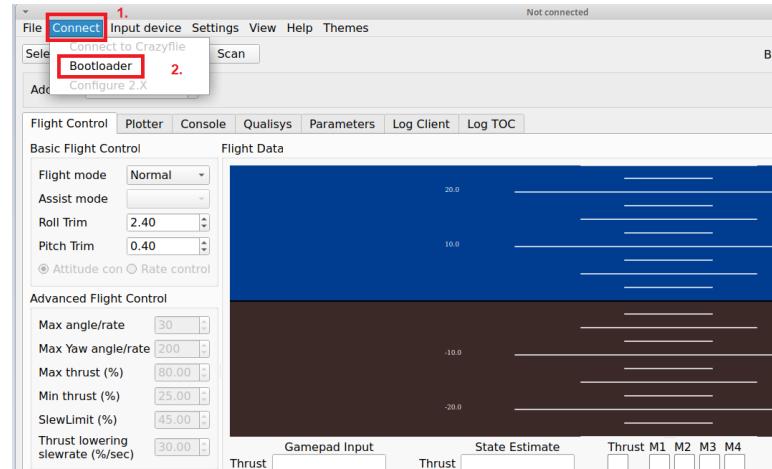


Figura 21: Iniciar pestaña de Bootloader.

Una vez en esta pestaña, se colocó al dron en modo “Bootloader”. El procedimiento para ello consistió en presionar el botón de encendido por 3 segundos cuando el dron se encuentra apagado, luego de ello empezarán a parpadear las dos LEDs azules del mismo indicando que ya se encuentra en dicho modo.

Colocado el dron en modo “Bootloader” y, estando en la pestaña correspondiente, se colocó sobre la pestaña de “Cold Boot”. Seguidamente se inicializó en modo “initiate bootloader cold boot”, se verificó que se estuviese conectado en “Status”, se buscó la versión más reciente de firmware para el dron y se procedió a programarlo, tal como se describe en la Figura 22.

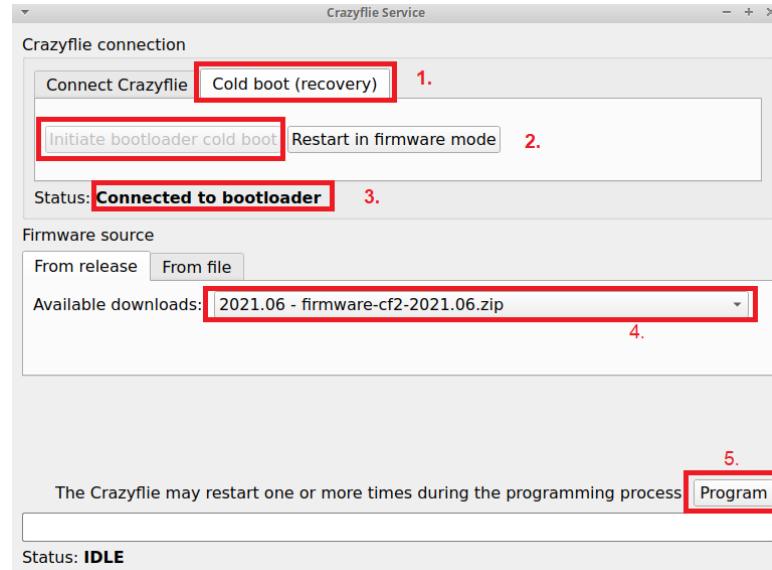


Figura 22: Configuración de *firmware*

Una vez presionado el botón de “Program” se procedió a la actualización del *firmware* del dron, esto duró aproximadamente 2 minutos y mientras el proceso ocurría la información de la pantalla se veía tal como se muestra en la Figura 23.

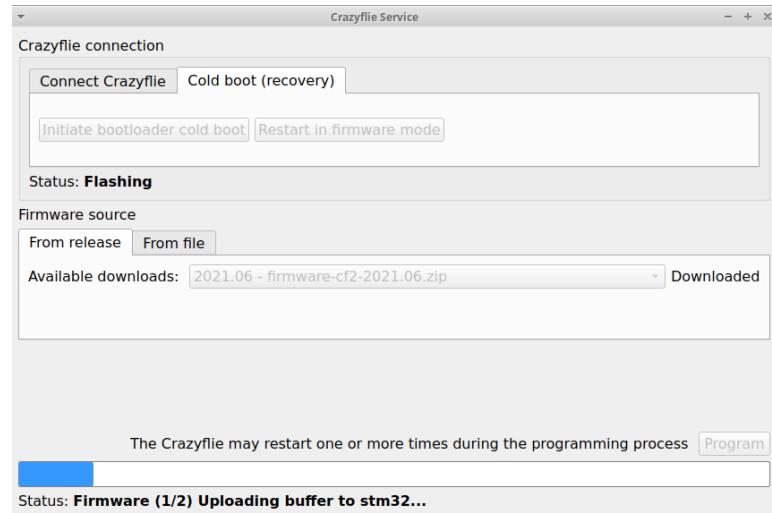


Figura 23: Programando el Crazyflie

Debido a que se busca trabajar con más de un dron a la vez y la forma de conectar es a través de una dirección de radio a un canal definido, es necesario asignarle un canal a cada

Crazyflie que se estará utilizando durante cada práctica de laboratorio. A continuación se muestra el procedimiento de cómo se realizó el cambio de dirección para el dron.

Para ello lo primero en realizar fue conectar el dron, como se realizó con anterioridad. Una vez conectado se seleccionó la pestaña “Connect” y dentro de las opciones se seleccionó la opción de “Configure 2.X” para obtener una nueva pestaña, en esta nueva pestaña lo único que se cambió es el canal de comunicación denominado “Radio channel” tal como se muestra en la Figura 24.

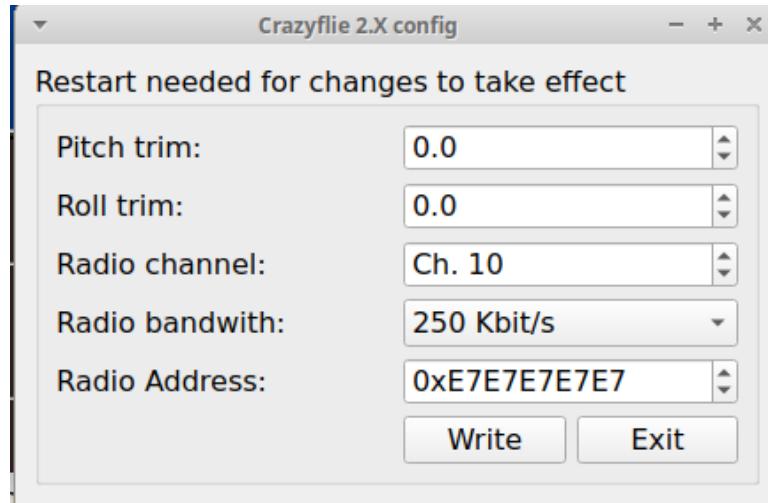


Figura 24: Configurando la comunicación del Crazyflie.

Estas fueron las configuraciones necesarias realizadas al dron, ya con ello fue posible programar el firmware de este en base a las necesidades y requerimientos planteados.

7.4. Programación del Crazyflie

Para programar el Crazyflie se utilizó *Visual Studio Code*, para ello lo primero que se debe de realizar es abrir la carpeta donde se encuentra todo el código del firmware del Crazyflie mediante la opción “Open Folder” y navegar a la dirección en la máquina virtual donde se encontraba el proyecto del Firmware, la cual corresponde a:

```
/home/bitcraze/projects/crazyflie/firmware/
```

Una vez trasladados en esa dirección se observa todos los códigos correspondientes al controlador STM 32 F405, siendo este el controlador principal del dron, tal como se observa en la Figura 25.

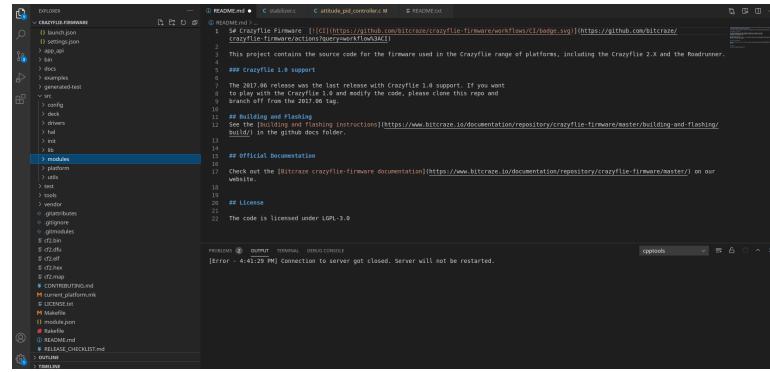


Figura 25: Carpeta del Firmware del Crazyflie 2.0.

Una vez en la carpeta actual, se realizó una prueba para verificar que efectivamente se estuviese cargando el código al microcontrolador. Para ello, dentro del proyecto se navegó a la dirección **src/drivers/interface/led.h** y se realizó lo siguiente:

Se buscó la línea de código:

```
#define SYS_LED LED_RED_R
```

Y se cambió por:

```
#define SYS_LED LED_GREEN_R
```

Luego se guardó el proyecto presionando el botón de “Guardar”.

7.4.1. Compilar el código

Luego de haber modificado y guardado el código fuente, el siguiente paso consistió en compilarlo. Esto para poder cargárselo al microcontrolador del Crazyflie, para ello se presionó la pestaña de “Terminal” y luego “New Terminal” para agregar una terminal a *Visual Studio Code*. Aparecerá una pestaña similar a lo que se muestra en la Figura 26.

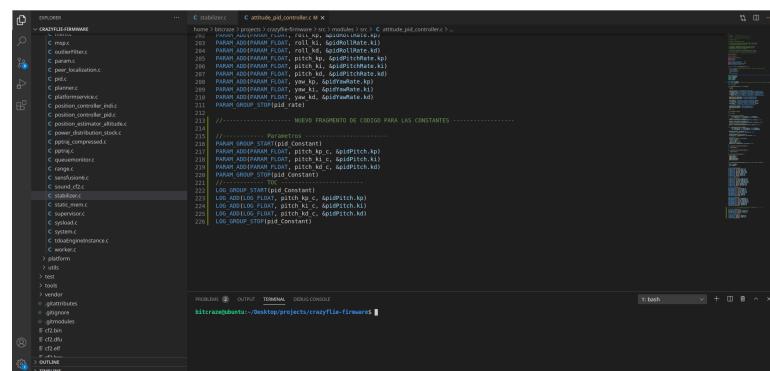
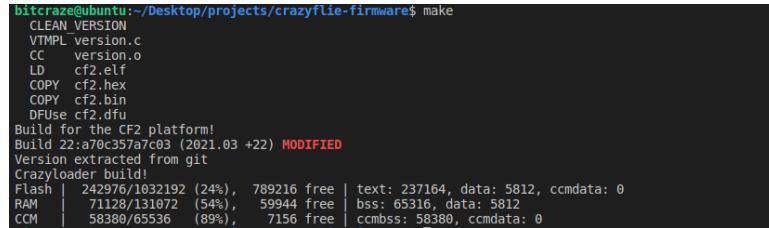


Figura 26: Terminal agregada a *Visual Studio Code*

Una vez abierta la terminal, para compilar el código únicamente se escribió “make”, se presionó “Enter” y empezó a compilar el código, tal como se muestra en la Figura 27.



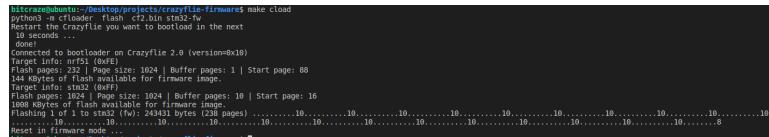
```
bitcraze@ubuntu:~/Desktop/projects/crazyflie-firmware$ make
  CLEAN VERSION
  VTMPL_version.c
  CC  version.o
  LD  cf2.elf
  COPY cf2.hex
  COPY cf2.bin
  DFUse cf2.dfu
Build for the CF2 platform!
Build 22:a70c357a7c03 (2021.03 +22) MODIFIED
Version extracted from git
Crazyloader build!
Flash | 242976/1032192 (24%), 789216 free | text: 237164, data: 5812, ccmdata: 0
RAM  | 71128/131072 (54%), 59944 free | bss: 65316, data: 5812
CCM | 58380/65536 (89%), 7156 free | ccmrss: 58380, ccmdata: 0
```

Figura 27: Compilando el código del *Firmware* del *Crazyflie 2.0*.

7.4.2. Flashear el Crazyflie 2.0

Una vez compilado el código fue necesario poder grabarlo en la memoria del controlador del Crazyflie 2.0. Para ello se preparó el dron, colocándolo en modo *Bootloader* (procedimiento que se realizó con anterioridad y que consiste en presionar por 3 segundos el botón de encendido del dron), hasta que las luces LED azules se vuelvan intermitentes.

Una vez preparado, se verificó que la antena de radio Crazyradio PA esté conectada a la máquina virtual y luego se escribió en la consola “make cload” donde comenzó el proceso de *flasheo*, tal como se muestra en la Figura 28.



```
bitcraze@ubuntu:~/Desktop/projects/crazyflie-firmware$ make cload
python3 -m cloader flash cf2.bin stm32fw
Format the crazyflie you want to bootload in the next
10 seconds
done
Connected to bootloader on crazyflie 2.0 (version=0x10)
Target Info: nrf51 (0x0FE)
Flash pages: 1024 | Page size: 1024 | Buffer pages: 1 | Start page: 0
144 Kbytes of flash available for firmware image.
Flashing page 0 to start (0x0): 0x4001 bytes (0 pages)
Flashing page 1 to start (0x10): 0x4001 bytes (0 pages)
Flashing page 2 to start (0x20): 0x4001 bytes (0 pages)
Flashing page 3 to start (0x30): 0x4001 bytes (0 pages)
Flashing page 4 to start (0x40): 0x4001 bytes (0 pages)
Flashing page 5 to start (0x50): 0x4001 bytes (0 pages)
Flashing page 6 to start (0x60): 0x4001 bytes (0 pages)
Flashing page 7 to start (0x70): 0x4001 bytes (0 pages)
Flashing page 8 to start (0x80): 0x4001 bytes (0 pages)
Flashing page 9 to start (0x90): 0x4001 bytes (0 pages)
Flashing page 10 to start (0xA0): 0x4001 bytes (0 pages)
Flashing page 11 to start (0xB0): 0x4001 bytes (0 pages)
Flashing page 12 to start (0xC0): 0x4001 bytes (0 pages)
Flashing page 13 to start (0xD0): 0x4001 bytes (0 pages)
Flashing page 14 to start (0xE0): 0x4001 bytes (0 pages)
Flashing page 15 to start (0xF0): 0x4001 bytes (0 pages)
Reset in Firmware mode ...
```

Figura 28: Flasheo del código del *Firmware* del *Crazyflie 2.0*

7.5. La API de Python

Para poder usar y controlar fácilmente el *Crazyflie 2.0*, el equipo de *Bitcraze* realizó una librería en Python que brinda funciones de alto nivel para poder implementarlas por el usuario. A continuación se muestra de forma general cómo utilizar la librería y la API que implementa [18].

7.5.1. Estructura de la biblioteca

La biblioteca es asíncrona y se basa en devoluciones de llamada por eventos. Sin embargo, existen algunas funciones como *open_link* que regresarán inmediatamente, mientras otras como *connected* se llamarán cuando se abra el enlace de conexión. A pesar de ello, existen algunas funciones síncronas para clases seleccionadas, de las cuales se hablarán más adelante.

Identificador Uniforme de Recursos (URI)

Los enlaces de comunicación se identifican mediante un URI, actualmente estos se utilizan únicamente para radio y depuración, por ejemplo:

- **radio: // 0/10 / 250K**: Esta es una interfaz de radio, tipo USB número 0, que utiliza el canal de radio 10 y la velocidad de radio es de 250Kb por segundo.
- **debug: // 0/1** Es una interfaz de depuración, que utiliza el id 0 y el canal 1.

Variables y registro

La librería permite la creación de configuraciones de registro que se utilizan para registrar variables de interés desde el *firmware* del dron. Cada configuración contiene una cantidad de variables que deben de registrarse, así como un periodo de tiempo de la frecuencia con la que los datos deben de enviarse al host. Al agregarse la configuración al firmware del dron, este enviará automáticamente los datos cada periodo establecido.

7.5.2. Parámetros

La librería de Python también admite la lectura y escritura de parámetros en tiempo de ejecución del *firmware* destinado para usarse en datos del *firmware* que no cambian continuamente, estos pueden cambiar cuando se configuran desde el *host* o durante el inicio. Estos deben de usarse de la siguiente manera:

- Registrar devoluciones de llamada actualizadas de parámetros desde cualquier momento en su aplicación.
- Conectarse al Crazyflie.
- Solicitar actualización para todos los parámetros.
- La biblioteca llamará a todas las devoluciones de llamada registradas.
- El *host* puede escribir parámetros que se enviarán al *firmware*.

Nombres de variables y parámetros

Todos los nombres de parámetros y variables de registro utilizan la misma estructura que corresponde a: **grupo.nombre**. Para este procedimiento existe un límite de 28 caracteres. El grupo debe usarse para agrupar a grupos lógicos, por ejemplo, para el grupo Estabilizador, los nombres de parámetros puede ser Cabeceo, Balanceo y Guiñada. Es decir:

- Estabilizador.Cabeceo.

- Estabilizador.Balanceo.
- Estabilizador.Guiñada.

7.5.3. Utilidades

Iniciar controladores de enlace

Antes de poder utilizar la biblioteca, los controladores de enlace deben de iniciarse , esto se realizó mediante la siguiente función:

```
init_drivers()
```

Conectarse al *Crazyflie 2.0*

Lo primero que se debe de realizar es encontrar al *Crazyflie 2.0* al cual se debe de conectarse lo cual se observa en el Código 7.2, esto se realiza indicándole a la librería que escanee todas las interfaces de radio disponible (por el momento se incluye radio y depuración).

```
cflib.crtplib.init_drivers()
available = cflib.crtplib.scan_interfaces()
for i in available:
    print "Interface with URL[%s] found and name/comment[%s]" %
        (i[0], i[1])
```

Código 7.2: Escaneo de radio para conexión al *Crazyflie 2.0*

Lo que sigue a continuación es poder conectar el dron, una vez identificado. Para ello se escribe el código descrito en la Lista 7.3.

```
crazyflie = Crazyflie()
crazyflie.connected.add_callback(crazyflie_connected)
crazyflie.open_link("radio://0/10/250K")
```

Código 7.3: Conectarse al *Crazyflie 2.0*

Y para poder terminar el conexión, se debe de cerrar la conexión del enlace, lo cual se realiza mediante la siguiente línea de código, donde se llama a una función de la librería, referenciándose al objeto crazyflie.

```
crazyflie.close_link()
```

Mandarle posiciones al *Crazyflie 2.0*

Mediante la librería es posible indicarle al dron en qué posición se desea que se coloque en cierta posición específica a cierto potencia en los motores. Se realiza mediante la siguiente función.

```
send_setpoint(roll, pitch, yaw, thrust)
```

Esta función pide que se le indique un ángulo de alabeo, cabeceo y guiñada, a cierta potencia especificada por un número entre 10001 (casi sin potencia) a 60000 (máxima potencia). Este comando debe de actualizarse cada 500 ms como máximo dado que después de este tiempo si no se actualiza el dron lo hará automáticamente donde todos los campos de la función sean cero. Y si después de 2 segundos aún no se aplica la función, el *Crazyflie 2.0* cortará la energía a los motores.

7.6. La API del *Crazyflie 2.0*

En la sección anterior se estableció la API de Python, la cual es utilizada para controlar el dron y obtener información del mismo. Sin embargo, dado que también se realizan modificaciones al *firmware* es importante describir de forma general la API del *Crazyflie 2.0* en sí, la cual se describe a continuación.

7.6.1. Marco de registro y parámetros

Los parámetros y marcos de registro son utilizados para poder registrar datos del *Crazyflie* y establecer variables de interés en el tiempo de ejecución [19].

Tabla de contenido (TOC)

Las variables que están disponibles para el marco de registro / parámetros se deciden en tiempo de compilación para el firmware *Crazyflie*. El uso de variables de macros en C puede estar disponible para el marco [19]. Un ejemplo de uso de estos parámetros son los ángulos de orientación del dron, los cuales se pueden leer desde la librería de Python, sin embargo, primero es necesario definirlos en el *firmware*, esto se realiza de la siguiente manera:

```
LOG_GROUP_START(stabilizer)
LOG_ADD(LOG_FLOAT, roll, &eulerRollActual)
LOG_ADD(LOG_FLOAT, pitch, &eulerPitchActual)
LOG_ADD(LOG_FLOAT, yaw, &eulerYawActual)
LOG_ADD(LOG_UINT16, thrust, &actuatorThrust)
LOG_GROUP_STOP(stabilizer)
```

Código 7.4: Agregar registros al *firmware*

Durante la compilación se crea una tabla de contenido (TOC) que contiene todas las variables disponibles junto con el tipo y las restricciones de acceso. Hay una tabla de contenido para cada marco, una para el registro y otra para los parámetros. Cuando el cliente se conecte, descargará el TOC para saber qué variables se pueden utilizar [19].

Parámetros

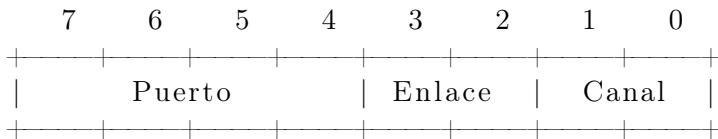
Usando el marco de parámetros es posible leer y escribir variables en tiempo de ejecución, pero se debe tener en cuenta lo siguiente:

- No hay protección de subprocessos en lectura / escritura. Dado que la arquitectura es de 32 bits y el parámetro más grande que puede tener es de 32 bits, es seguro escribir una variable.
- Se debe utilizar el marco de parámetros para leer las variables que se establecen durante la puesta en marcha, si estas cambian durante el tiempo de ejecución, se debe utilizar el marco de registro.
- La lectura y escritura de un parámetro se puede realizar en cualquier momento siempre y cuando el dron esté conectado.

7.6.2. CRTP - Comunicación con Crazyflie

El *Crazyflie* tiene su propio protocolo de comunicación de lato nivel, denominado CRTP (*Crazy RealTime Protocol*) el cual es un protocolo simple que utiliza varios puertos, con comunicación bidireccional, los medios de transmisión de datos puede ser a través de USB o mediante el Crazyradio [20].

Cada paquete tiene un encabezado de 1 byte y puede transportar hasta 31 bytes de carga útil de datos. El campo de encabezado tiene el siguiente diseño:



Donde:

- **Puerto:** Se utiliza para identificar la funcionalidad o tarea asociada al mensaje.
- **Enlace:** Es una palabra reservada pero que aún no está en uso.
- **Canal:** SE utiliza para identificar la subtarea/funcionalidad.

CRTP está diseñado para ser sin estado, por lo que no es necesario ningún procedimiento de establecimiento de comunicación. Se puede enviar cualquier comando en cualquier momento, pero para algunos comandos de registro / parámetros, es necesario descargar el TOC (tabla de contenido) para que el host pueda enviar la información correcta [20]..

7.6.3. Sensores para control

El *Crazyflie* está equipado con una serie de sensores, que le permiten determinar su posición o orientación; de forma generalizada se describe el camino que siguen los datos

desde su obtención mediante sensores hasta llegar a los motores para controlar al dron, los cuales, tienen un efecto en la manera que vuela el crzayflie y a su vez la forma en que los sensores toman los datos [21].

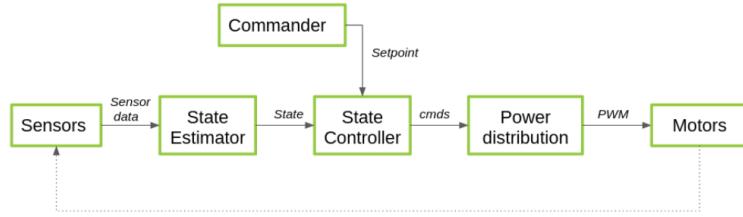


Figura 29: Flujo de información de sensores en el *Crazyflie* [21].

Como se observa en la Figura 29, la información es obtenida por los sensores, luego es procesada por un Filtro de Kalman (siendo este el estimador de estados), la información de estados es necesaria para el controlador de estados (siendo la medición actual) y se compara con la referencia, que son los datos enviados por el comandante (el estado que queremos para el dron), luego esta información es trabajada por los sensores y la manda a la distribución de potencia, que es encargado de mandar la señal en forma de un PWM a los motores, los cuales generan un cambio en el estado del dron.

7.6.4. Controladores en el Crazyflie

Como se describe en la sección 7.6.3, una vez que el estimador haya calculado el estado actual tanto de la posición, orientación y sus velocidades; le corresponde a los controladores mantener al dron en un punto de ajuste definido por el usuario. Para realizar esto, el *firmware* del Crazyflie implementa 3 etapas de control, las cuales son:

- Tasa de orientación.
- Orientación absoluta.
- Posición o velocidad actual.

Para ello, el dron cuenta con 3 controladores implementados.

- Proporcional Integral Derivativo (PID).
- Inversión dinámica incremental no lineal (INDI).
- Mellinger.

Para los experimentos descritos en el Capítulo 8, se utilizó un controlador PID, es por ello, que únicamente se explica dicho controlador y en la forma que fue modificado.

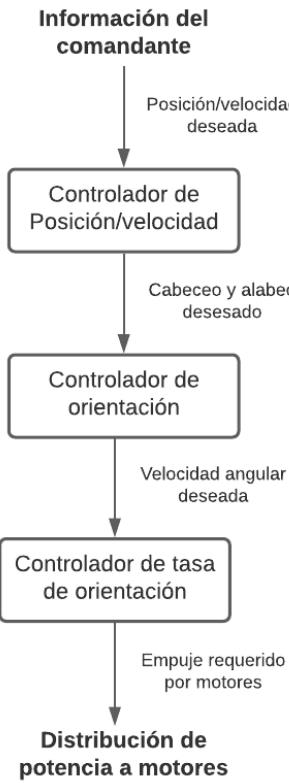


Figura 30: Controlador PID en el *Crazyflie 2.0*

Controlador PID en cascada

Como se describe en la Figura 30, el PID implementado se trata de uno en cascada. Por lo tanto, se envía la posición para los puntos de ajuste de posición deseados al controlador de posición PID. Estos dan como resultado los ángulos de cabeceo y balanceo deseados, que se envían directamente al controlador PID de actitud. Estos determinan las tasas de ángulo deseadas que se envían al controlador de tasa de ángulo [22].

Controlador PID para la tasa de orientación

El controlador PID de tasa de actitud es el que controla directamente la tasa de actitud. Resuelve casi directamente las tasas del giroscopio (primero mediante un poco de filtrado) y toma el error entre la tasa de actitud deseada como entrada. Esto genera los comandos que se envían directamente a la distribución de potencia de motores. El lazo de control funciona a 500 Hz [22].

Controlador PID para la orientación

El controlador PID de actitud absoluta es el lazo exterior del controlador de actitud. Esto toma la actitud estimada del estimador de estado y toma el error del punto de ajuste de actitud deseado para controlar la actitud del *Crazyflie* [22]. .

Controlador PID para posición o velocidad

Corresponde a la capa externa de control, este controlador recibe directamente los puntos de ajuste deseados, los cuales son enviados directamente por el usuario, es por ello que el lazo de control funciona a una frecuencia más pequeña, siendo de 100 Hz [22]. .

Controlador de interés

La combinación de estos controladores permiten al *Crazyflie* mantenerse en equilibrio; sin embargo, no se trabajaron con todos, dado que se tiene como interés modificar el controlador que influye directamente en el ángulo de cabeceo del dron; siendo este el controlador de orientación, el cual corresponde a la segunda etapa de control del sistema, como se observa en la Figura 30.

7.6.5. Marco del Comandante

El marco del comandante, resulta ser de mucha importancia para la aplicación realizada, dado que es el encargado de darle los puntos de ajuste a los controladores, para que estos se encarguen de llevar al dron a posición y orientación deseada.

Los puntos de ajuste se pueden establecer directamente, esto a través de la librería proporcionada en Python o mediante el módulo de comandante de alto nivel, el cual se controla en el interior del *Crazyflie* o remotamente mediante la librería de Python [23].

CAPÍTULO 8

Descripción de los experimentos y las pruebas

El presente capítulo describe todas las pruebas y experimentos realizados necesarios previo a la elaboración de las guías de laboratorio. En primer lugar, se programó un Arduino Uno el cual fue encargado de leer las señales provenientes del codificador rotacional; procesarlas y luego mandar el ángulo por comunicación serial a la computadora. Seguidamente se realizaron pruebas de comunicación con el Arduino Uno y Python.

Al verificar el correcto funcionamiento de la comunicación serial, se realizaron pruebas de comunicación en conexión y desconexión de la librería de Python y el Crazyflie 2.0;. Se realizaron experimentos de envío de datos del Crazyflie a la computadora. Posteriormente se realizaron pruebas de colocar el dron en cierta posición mediante comandos en la computadora.

Con los experimentos realizados, se procedieron a realizar pruebas con la interfaz gráfica, en la cual se unificaron todas las pruebas anteriores utilizando una ventana sencilla y fácil de manejar.

Los códigos desarrollados en estos experimentos se encuentran en un repositorio de *GitHub* debidamente comentado, el cual se encuentra clonado en la máquina virtual para poder revisarse o modificarse a futuro si fuese necesario.

8.1. Uso de la plataforma física

Lo primero que se realizó con la plataforma fue verificar el estado de la misma. Para ello se revisaron todas las juntas y se verificó que el dron estuviese sujetado correctamente, en el cual se determinó la orientación del mismo sabiendo que el grado de libertad disponible en la plataforma permitía movimientos en cabeceo del dron, por lo que los experimentos que

se realizaron giraron en torno a dicho movimiento.

8.1.1. Conexión del codificador rotacional

El medio de comunicación entre la plataforma física y la interfaz gráfica que se realizó es el codificador rotacional, el cual es encargado de medir el grado de libertad de la plataforma. Para poder extraer la información del mismo se utilizó un Arduino Uno, el cual se conectó tal como se muestra en la Figura 31. Ya con ello se procedió a programar el Arduino.

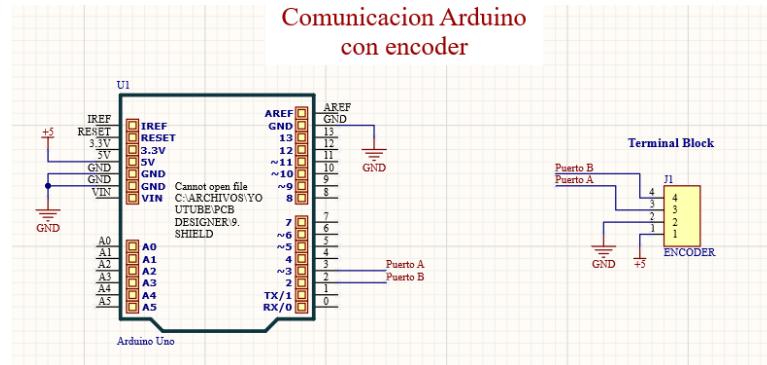


Figura 31: Conexión de codificador rotacional al Arduino

Pruebas con el codificador rotacional y el Arduino

Se realizó un primer programa en el cual se utilizó una librería para codificadores rotacionales incrementales disponible para Arduino, se implementó la librería y se multiplicó el factor correspondiente para el codificador rotacional teniendo este un valor de 0.15. En el código se habilitó el puerto serial para que le mandara los datos a la consola de Arduino y que imprima el ángulo donde se verificó que efectivamente estuviese funcionando, como se observa en la Figura 32.

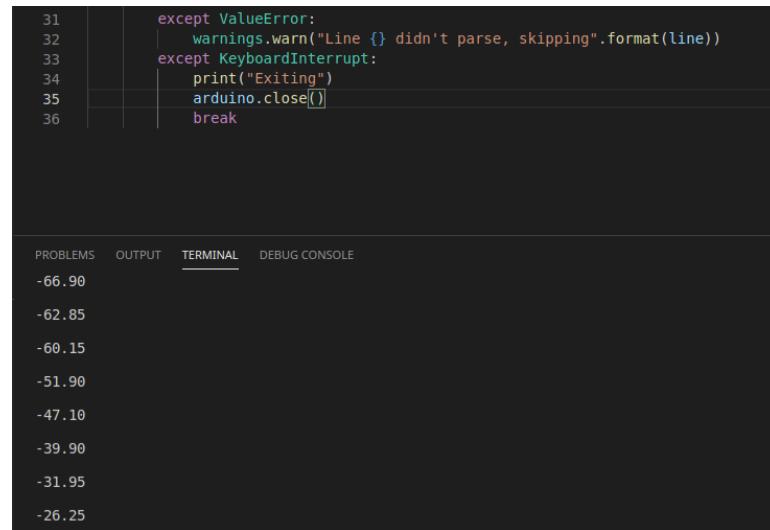


Figura 32: Puerto serial del Arduino con el ángulo dado

8.2. Conexión de Arduino y Python

Como se describió en la sección de la máquina virtual *Visual Studio Code* contiene Python instalado en la consola con todas las librerías, entre ellas *pyserial*. La siguiente prueba que se realizó fue conectar el Arduino con Python para verificar si imprimía los datos en la consola tal como lo se hacía en Arduino.

Como paso importante consistió en habilitar el puerto serial y llamarlo en Python, se procedió a la lectura de datos y luego a imprimirlas en la consola, donde se verificó el correcto funcionamiento de la comunicación serial tal como se observa en la Figura 33.



The image shows a screenshot of the Visual Studio Code interface. On the left, a code editor displays a Python script with the following code:

```
31     except ValueError:
32         warnings.warn("Line {} didn't parse, skipping".format(line))
33     except KeyboardInterrupt:
34         print("Exiting")
35         arduino.close()
36         break
```

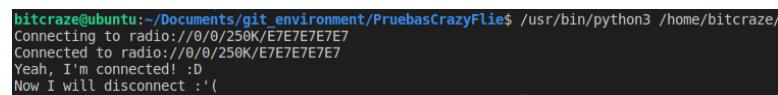
On the right, a terminal window shows the output of the script, which is a list of numerical values:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
-66.90
-62.85
-60.15
-51.90
-47.10
-39.90
-31.95
-26.25
```

Figura 33: Conexión serial entre Python y Arduino

8.3. Conexión de Crazyflie y Python

Teniendo en consideración la documentación presentada en la descripción de la API en su sección correspondiente, se realizó un código en Python para poder conectar el Crazyflie a la computadora a través del *CrazyRadio PA*, en esta prueba solo se verificó la conexión correcta al dron tal como se muestra en la Figura 34.



```
bitcraze@ubuntu:~/Documents/git_environment/PruebasCrazyFlie$ /usr/bin/python3 /home/bitcraze/
Connecting to radio://0/0/250K/E7E7E7E7E7
Connected to radio://0/0/250K/E7E7E7E7E7
Yeah, I'm connected! :D
Now I will disconnect :('
```

Figura 34: Conexión del dron a Python

8.3.1. Lectura de parámetros

Como se observó con anterioridad, se logró conectar correctamente el dron a Python, la siguiente etapa consistió en extraer información del *Crazyflie 2.0* a través de los parámetros

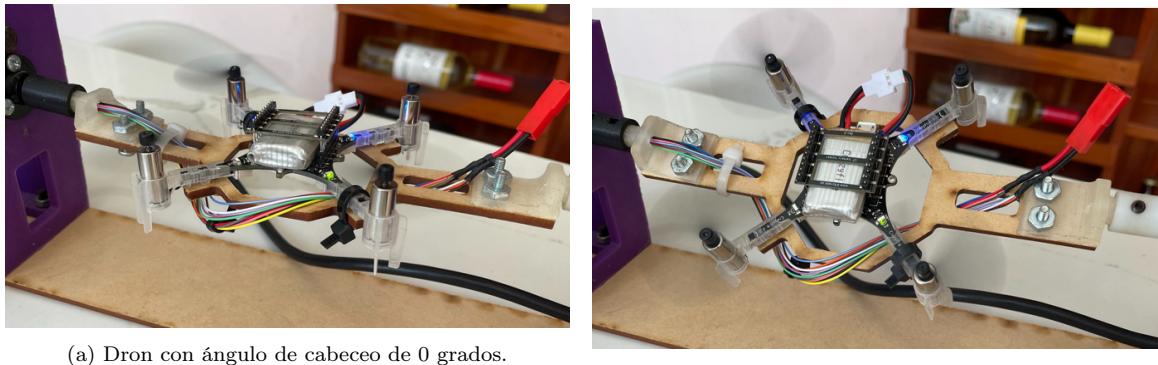
previamente definidos en el *firmware* del dron, para ello se realizó un código que permitía la lectura de parámetros: ángulos de alabeo, banqueo y cabeceo. Esto se observó que funcionó correctamente tal como se observa en la Figura 35

```
bitcraze@ubuntu:~/Documents/git_environment/PruebasCrazyFly$ /usr/bin/python3 /home/bitcraze
Connecting to radio://0/0/250K/E7E7E7E7E7
Connected to radio://0/0/250K/E7E7E7E7E7
[907432][Stabilizer]: stabilizer.roll: 2.433 stabilizer.pitch: 3.175 stabilizer.yaw: -28.646
[907532][Stabilizer]: stabilizer.roll: 2.429 stabilizer.pitch: 3.096 stabilizer.yaw: -28.647
[907632][Stabilizer]: stabilizer.roll: 2.423 stabilizer.pitch: 3.121 stabilizer.yaw: -28.648
[907732][Stabilizer]: stabilizer.roll: 2.424 stabilizer.pitch: 3.085 stabilizer.yaw: -28.656
[907832][Stabilizer]: stabilizer.roll: 2.420 stabilizer.pitch: 3.024 stabilizer.yaw: -28.658
[907932][Stabilizer]: stabilizer.roll: 2.419 stabilizer.pitch: 3.012 stabilizer.yaw: -28.660
[908032][Stabilizer]: stabilizer.roll: 2.418 stabilizer.pitch: 3.007 stabilizer.yaw: -28.662
[908132][Stabilizer]: stabilizer.roll: 2.420 stabilizer.pitch: 2.963 stabilizer.yaw: -28.662
[908232][Stabilizer]: stabilizer.roll: 2.465 stabilizer.pitch: -0.754 stabilizer.yaw: -28.701
[908332][Stabilizer]: stabilizer.roll: 2.510 stabilizer.pitch: -4.584 stabilizer.yaw: -28.735
[908432][Stabilizer]: stabilizer.roll: 2.551 stabilizer.pitch: -8.584 stabilizer.yaw: -28.764
```

Figura 35: Lectura de parámetros del *Crazyflie 2.0*

8.3.2. Movimiento del dron a través de Python

El siguiente paso de los experimentos realizados consistió en indicarle a través de Python a qué posición en el grado de libertad debía moverse el *Crazyflie 2.0*. Para ello se utilizó nuevamente la librería disponible, donde ya existía una función específica, por lo que se programó un código de prueba encargado de realizar dicha función. Las Figuras 36a y 36b muestran el detalle de su implementación.



(a) Dron con ángulo de cabeceo de 0 grados.

(b) Dron con ángulo de cabeceo de 30 grados.

Figura 36: Control de orientación del dron con Python.

8.4. Interfaz gráfica en Python

Luego de la experimentación del dron y su plataforma con cada uno de los componentes del mismo, fue necesario unificar toda la programación realizada en una sola interfaz, donde se incluyeron todas las funcionalidades de la librería que serían necesarias para poder trabajar con las guías de laboratorio.

8.4.1. Gráfica del codificador rotacional

Se consideró de importancia graficar en tiempo real la posición angular del dron (ángulo de cabeceo) para que mientras se realicen las pruebas se pudiese observar el comportamiento tanto en el dron como en la gráfica por la información del codificador rotacional, lo cual se logró utilizando las librerías de *Matplotlib* de Python. El resultado se observa en la Figura 37.

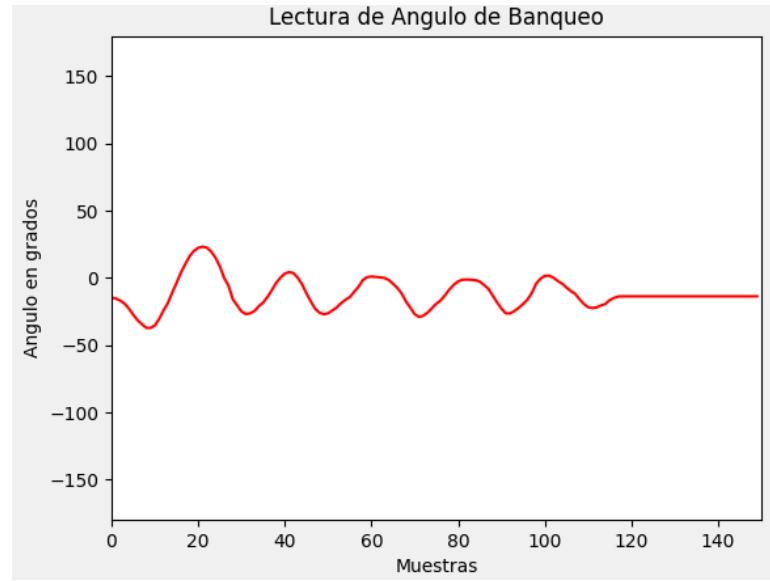


Figura 37: Gráfica del ángulo de cabeceo respecto al número de muestras

8.4.2. Guardar datos de corrida

Dado que los datos serán utilizados con fines didácticos para prácticas de laboratorio se considera importante saber guardarlos, para ello se propuso guardar un gráfico y guardar los datos en bruto. El gráfico se guarda como imagen en formato PNG y los datos en bruto se guardan en un formato .cvs el cual se podrá importar a *MATLAB* para futuro procesamiento. En la Figura 38 se muestra la gráfica guardada de una corrida.

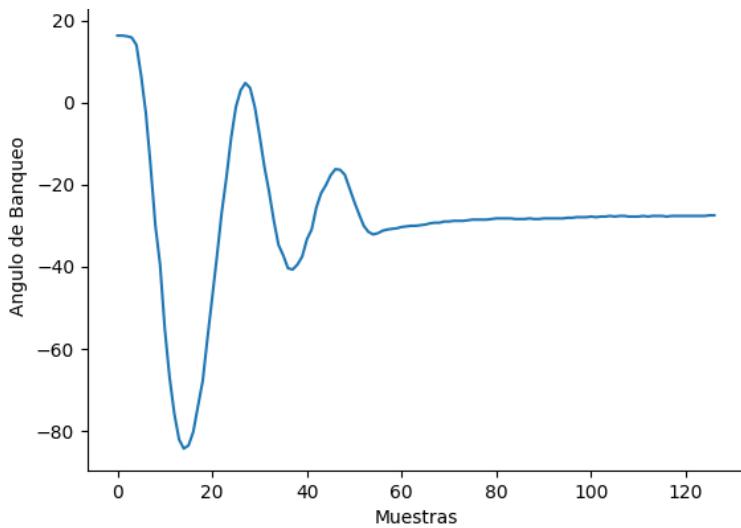


Figura 38: Gráfica del ángulo de cabeceo guardado como figura.

8.4.3. Explicación de la interfaz desarrollada

En la Figura 39 se puede observar la interfaz que se realizó, esta fue hecha pensando en la primera práctica de laboratorio, cuya descripción se encuentra en la Sección 9.2. A continuación se detalla la interfaz creada, así como la funcionalidad que se le fue agregada a cada botón.

- **Recuadro A:** En este recuadro se observa el gráfico en tiempo real del ángulo medido por el codificador rotacional, que para el caso por defecto al cual se encuentra el dron corresponde al ángulo de cabeceo.
- **Recuadro B:** En este recuadro se conecta y desconecta el dron, tal como los primeros dos botones indican. El botón de “Tarar” define un cero local para la programación del codificador rotacional. Dado que es de tipo incremental, se debe de establecer un cero al inicio y este coincida con el cero global.
- **Recuadro C:** El tercer recuadro corresponde a la orientación que el usuario desea colocar al dron, donde se inicia una rutina al presionar el botón que se encuentra en dicho recuadro. Para que la rutina se realice es evidente que el dron debe de estar conectado.
- **Recuadro D:** Este es el recuadro donde el usuario configura las constantes que deseé para poder configurar el controlador PID de orientación para el ángulo de cabeceo. Estas constantes se envían al presionar el botón de “Enviar Constantes”, mientras que el botón de “Reset Constantes” configura el PID a las constantes por defecto. El botón de “Guardar Corrida” guarda los datos de la última rutina que realizó el dron, tanto los datos en bruto como una figura de la corrida.

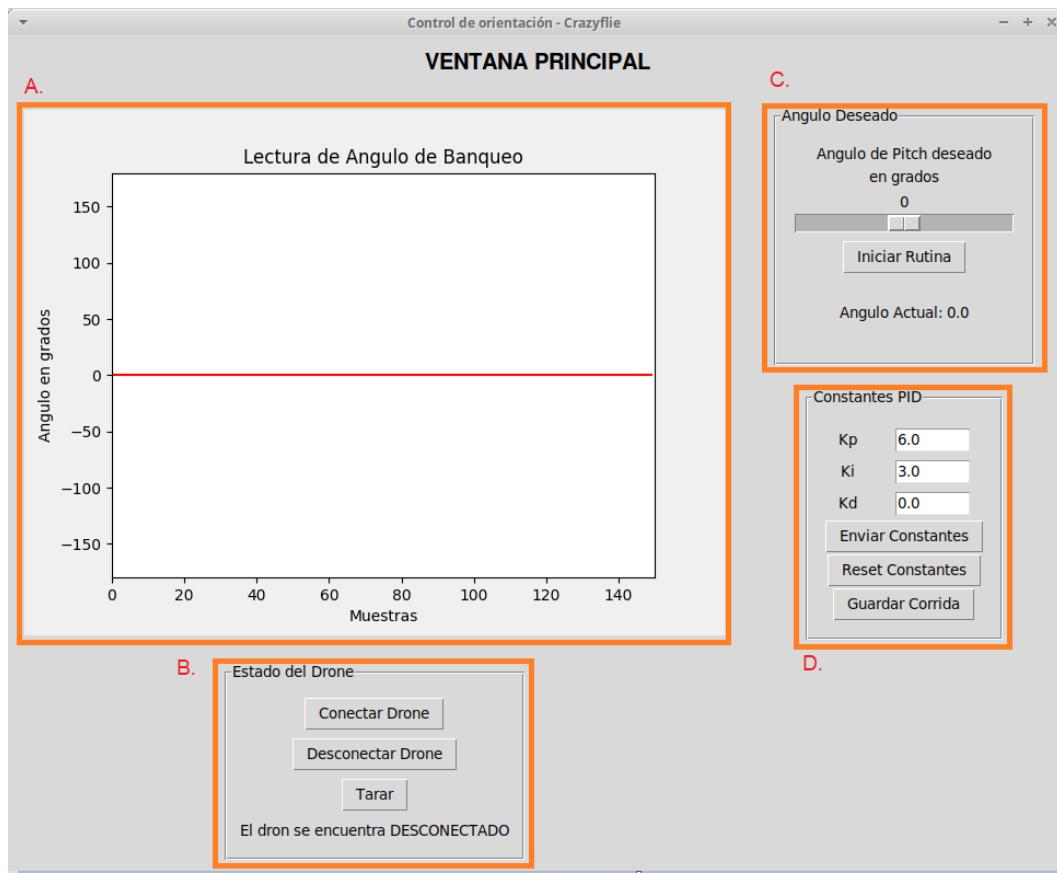


Figura 39: Interfaz gráfica

8.4.4. Uso de Matlab para graficar la Respuesta al Escalón

Los datos extraídos por el codificador rotacional serán analizados para poder obtener su respuesta al escalón, en lo que corresponde al análisis, se obtiene tanto la forma gráfica de la respuesta así como los parámetros de rendimiento, los cuales son de utilidad para el desarrollo de las prácticas de laboratorio.

Lo primero que se realizó es importar los datos a Matlab, luego se aplicó un pre-procesamiento para obtener los datos de interés, seguidamente se realizó la gráfica respectiva. En la Figura 40 se muestra un ejemplo de una corrida graficada.

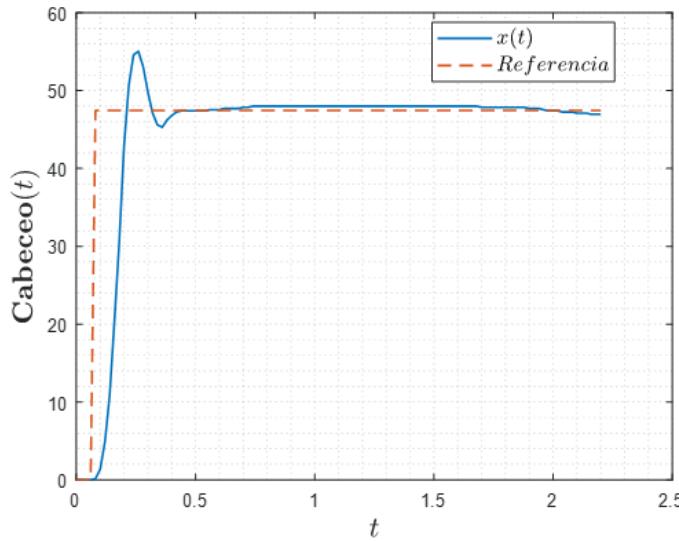


Figura 40: Gráfica de una corrida del ángulo de cabeceo.

Seguidamente, se procedió a obtener los parámetros de rendimiento de la respuesta al escalón graficada, para ello se utilizó una función de Matlab previamente definida, denominada `stepinfo()`, los resultados de aplicar dicha función se observan en la Figura 41.

```
struct with fields:

    RiseTime: 0.0808
    SettlingTime: 0.3900
    SettlingMin: 42.9001
    SettlingMax: 55.0501
    Overshoot: 16.0168
    Undershoot: 0
    Peak: 55.0501
    PeakTime: 0.2600
```

Figura 41: Parámetros de rendimiento de la respuesta al escalón de un corrida del ángulo de cabeceo.

8.5. Identificación de sistemas para determinar la planta del *Crazyflie 2.0* y su plataforma en función de transferencia

Dada la complejidad de la planta, no resultó práctico el poder realizar un procedimiento analítico para poder determinar el modelo matemático del sistema representado en función de transferencia, el cual será de utilidad para el desarrollo de las guías de laboratorio. Para poder determinar dicho modelo matemático se utilizó la herramienta de MATLAB denominada **Identificación de sistemas**.

La identificación de sistemas necesita tanto la entrada del sistema como la salida; con dichos datos proporcionados y indicándole la cantidad de polos y ceros, el algoritmo utiliza optimización para construir la función de transferencia que mejor se aadecue a los datos proporcionados.

Los datos de entrada y salida que se utilizaron fueron los mismos que se observan en la gráfica de la Figura 40, se realizaron pruebas utilizando distintas cantidades de polos y ceros, para los cuales el algoritmo obtuvo distintas funciones de transferencia; sin embargo el mejor resultado se obtuvo utilizando una función de transferencia con 4 polos y 4 ceros. Los resultados se observan en la Figura 46

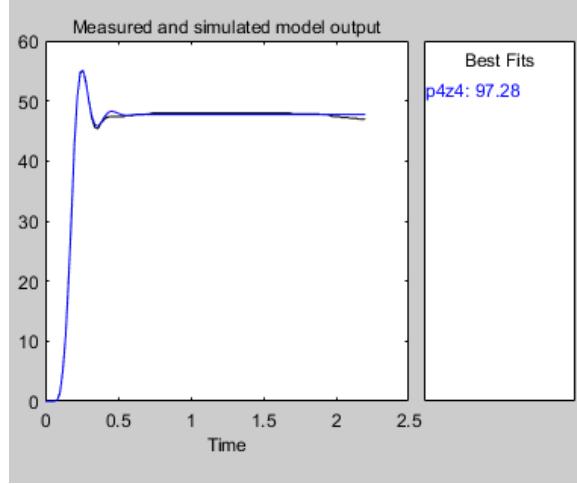


Figura 42: Resultado de la identificación de sistemas con 4 polos y 4 ceros.

La función de transferencia resultante es:

$$H(s) = \frac{4.211 \times 10^{15}s^4 - 4.232 \times 10^{18}s^3 + 6.596 \times 10^{20}s^2 - 2.321 \times 10^{22}s + 1.972 \times 10^{24}}{1.153 \times 10^{18}s^4 + 1.004 \times 10^{20}s^3 + 4.839 \times 10^{21}s^2 + 1.242 \times 10^{23}s + 1.96 \times 10^{24}}$$

La cual corresponde a la función de transferencia del sistema completo, es decir, la planta y el controlador, con retroalimentación negativa. Lo que interesa en este caso particular es poder obtener la función de transferencia de la planta, dado que ya se conoce la del controlador.

La arquitectura que se plantea para obtener la planta se observa en la Figura 43, interesa obtener únicamente la función de transferencia $G(s)$ dado que ya se conoce $PID(s)$ y la función de transferencia del sistema completo $H(s)$.

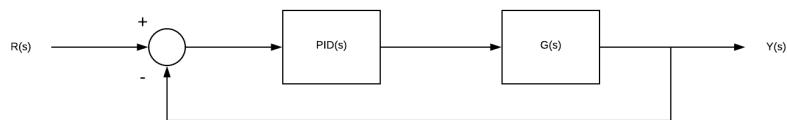


Figura 43: Planteamiento del sistema completo.

De álgebra de bloques se obtiene que la función de transferencia está dada por:

$$H(s) = \frac{P(s)}{Q(s)} = \frac{PID(s)G(s)}{1 + PID(s)G(s)} = \frac{Y(s)}{R(s)} \quad (33)$$

De la ecuación anterior se despeja para $G(s)$, por lo que se obtiene la expresión:

$$H(s) = \frac{P(s)}{Q(s)} = \frac{PID(s)G(s)}{1 + PID(s)G(s)} = \frac{Y(s)}{R(s)} \quad (34)$$

Despejando para $G(s)$, la ecuación correspondiente es:

$$G(s) = \frac{P(s)}{Q(s)PID(s)} \left(1 - \frac{P(s)}{Q(s)}\right)^{-1} \quad (35)$$

Teniendo $H(s) = P(s)/Q(s)$, se definió $PID(s)$.

Para el sistema resultante del proceso de identificación de sistemas, la función de transferencia correspondiente al $PID(s)$ ya estaba previamente definida, esto quiere decir que el proceso de identificación de los datos ya se encontraba embebida dicha función de transferencia, la cual tiene la siguiente forma:

$$PID(s) = \frac{K_d s^2 + K_p s + K_i}{s} \quad (36)$$

Para el cual se consideró:

- $K_p = 12.0$
- $K_i = 6.00$
- $K_d = 0.01$

Por lo que se tiene:

$$PID(s) = \frac{s^2 + 1200s + 600}{100s} \quad (37)$$

Teniendo todas las funciones de transferencia definidas, estas se ingresaron al paquete de matemática simbólica de MATLAB para que realizara la operatoria correspondiente, finalmente se obtuvo la función de transferencia de la planta del Crazyflie, la cual es la que se desea analizar.

$$G(s) = \frac{4.21 \times 10^{15} s^5 - 4.23 \times 10^{18} s^4 + 6.6 \times 10^{20} s^3 - 2.3 \times 10^{22} s^2 + 1.97 \times 10^{24} s}{1.153 \times 10^{18} s^4 + 1.004 \times 10^{20} s^3 + 4.839 \times 10^{21} s^2 + 1.242 \times 10^{23} s + 1.96 \times 10^{24}}$$

8.6. Identificación de sistemas para determinar la planta del *Crazyflie 2.0* y su plataforma en espacio de estados

De manera análoga a la identificación de sistemas en función de transferencia presentada en la Sección 8.5 se realizó la identificación de sistemas en espacio de estados. Se utilizó Matlab nuevamente, en la cual se realizaron distintas pruebas con diferentes órdenes para el espacio de estados, siendo la mejor estimación para un sistema de orden 2, los resultados se evidencian se evidencian en la Figura 44.

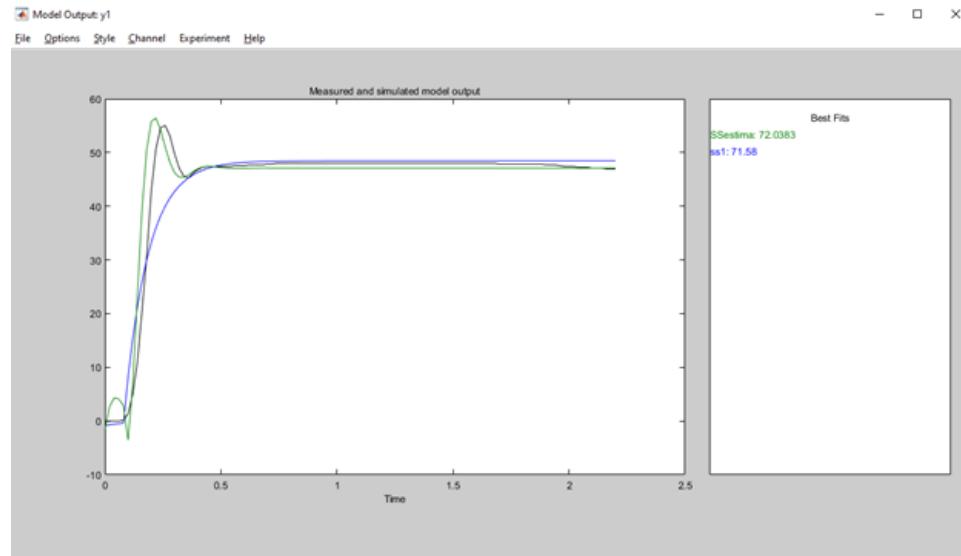


Figura 44: Resultado de la identificación de sistemas para orden 1 (en azul) y 2 (en verde).

Luego, se obtener los datos de identificación de sistemas se realizó el siguiente procedimiento.

1. Se pasó el sistema de espacio de estados a función de transferencia (comando *ss2tf* en Matlab).
2. Se separó dicha función de transferencia en numerador $P(s)$ y denominador $Q(s)$.
3. Se obtuvo la función de transferencia del controlador PID utilizado para la obtención de datos.
4. Se realizaron los cálculos para obtener la planta del dron y la plataforma únicamente.
5. Con la función de transferencia resultante se pasó el sistema a espacio de estados nuevamente (comando *tf2ss* en Matlab).

De lo que se obtuvo el siguiente sistema en espacio de estados:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du}\end{aligned}\tag{38}$$

Donde A,B,C y D son matrices, las cuales se describen a continuación.

$$A = \begin{bmatrix} -27.5235 & -23.7152 \\ 37.6820 & -1.1594 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.1171 \\ 1.0795 \end{bmatrix}$$

$$C = \begin{bmatrix} -21.1973 & 5.6194 \end{bmatrix}$$

$$D = 0$$

8.7. Configuración de un controlador PID mediante su plantamiento en LQR

Para desarrollar el segundo laboratorio, se planteó un controlador PID en espacio de estados, el cual resulta ser un controlador LQR; esto para que Matlab sea capaz de resolver el problema de optimización y a partir de él lograr obtener las constantes PID óptimas.

Partiendo de que se tiene un sistema en espacio de estados, el cual se define mediante

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du} \end{aligned} \tag{39}$$

El PID en tiempo continuo se expresa mediante:

$$u = K_p e + K_I \int_0^t e(\tau) d\tau + K_D \dot{e} \tag{40}$$

$$e = r - y \tag{41}$$

$$u = K_p e + K_I \int_0^t e(\tau) d\tau + K_D C \dot{x} \tag{42}$$

Donde

Asumiendo que la referencia es constante (caso que aplica para el sistema planteado). El controlador puede reescribirse de la siguiente manera:

$$u = K_P r - K_P C x + K_I z + K_D C \dot{x} \tag{43}$$

Donse se selecciona z tal que $\dot{z} = r - y = r - Cx$

$$u = K_P r - K_P C x + K_I z - K_D C A x - K_D C B u \quad (44)$$

Dado que $K_D C B u = 0$, entonces:

$$u = K_P r - K_P C x + K_I z - K_D C A x \quad (45)$$

Se considera a

$$\bar{x} = \begin{bmatrix} x \\ z \end{bmatrix}$$

Entonces se agrupa la ecuación mediante

$$u = -[(K_P C + K_D C A) \ K_I] \begin{bmatrix} x \\ z \end{bmatrix} + K_P r \quad (46)$$

Luego de operar las matrices C y A, definidas para un sistema de segundo orden, se determina que la matriz \bar{K} resultante es de 1 fila y 3 columnas.

Se puede definir un nuevo sistema aumentado, el cual corresponde a:

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r \quad (47)$$

El cual corresponde a un sistema aumentado, que se resume en:

$$\dot{\bar{x}} = \bar{A} \bar{x} + \bar{B} u + M r \quad (48)$$

Por lo que puede re- escribirse el sistema como:

$$u = -\bar{K} \bar{x} + K_P r \quad (49)$$

Observando la Ecuación 46, se determina que la constante K_I se puede extraer directamente de la matriz de constantes \bar{K} correspondiente a la tercera columna, sin embargo, se debe de efectuar una última operación para determinar K_P y K_D dado que no se encuentran de forma explícita, para ello se aplica la siguiente igualdad.

$$K_P * C + K_D * C * A = \bar{K}[1 : 2] \quad (50)$$

Cabe mencionar que ASFASFASFD

Seguidamente se aplica un *solver* de sistema de ecuaciones, para que pudiese determinar dichas constantes, mediante el siguiente comando en *MATLAB*.

```
Variables=solve ( Matrix == K_bar(1,1:2) ,{K_P,K_D});
```

Código 8.1: Solución de constantes del controlador

Con el Código 8.1 se obtienen las dos constantes faltantes del sistema. Estas constantes son las que se deben de ingresar en el controlador PID, a partir de la determinación de las mismas en variable de estado.

CAPÍTULO 9

Guías de laboratorio desarrolladas

En el presente capítulo se abordará la parte pedagógica de la plataforma de pruebas. Se iniciará describiendo el manual de usuario e instalación de herramientas para el *Crazyflie 2.0* el cual se considera como el pre-laboratorio. En esta etapa se le dará la introducción de la plataforma al estudiante y la familiarización con las herramientas que estará utilizando. Seguidamente se describirán los laboratorios 1 y 2 diseñados para los cursos de Sistemas de Control 1 y Sistemas de Control 2, respectivamente, haciendo una descripción de los objetivos y las pruebas piloto realizadas.

9.1. Manual de usuario e instalación de herramientas

Antes de llevar a cabo los laboratorios, el estudiante deberá familiarizarse con el dron, con la plataforma y tener las herramientas necesarias para modificar la plataforma, así como una computadora en la cual que tenga al menos 10 GB de memoria disponible, para ello se recomienda utilizar las computadoras disponibles en la universidad y tener derechos de administrador.

9.1.1. Objetivos:

- Familiarizarse con el dron y la forma de utilizarlo.
- Familiarizarse con la plataforma y cómo conectarla.
- Instalar la máquina virtual que incluye todas las herramientas para utilizar el *Crazyflie*.
- Aprender a manejar los datos desde la máquina virtual a la computadora que actúa como anfitrión.

9.1.2. Descripción

El manual de usuario inicia con la descripción de las pruebas que se deben de realizar para verificar que el dron se encuentra en buen estado y que ningún componente represente un problema para el desarrollo de la guía. Luego de las pruebas, se da una breve descripción del dron, la forma en que funciona y la orientación del mismo. En esta parte se detalla la secuencia de luces y el significado de cada una de ellas.

Seguidamente se da una descripción de como poder volar el dron con un teléfono, paso que puede obviarse por razones didácticas. Dicha prueba solo será necesaria si los drones han estado almacenados por mucho tiempo.

Dada la breve introducción al dron, cómo encenderlo e interpretar la secuencia de luces, se da una breve descripción de una máquina virtual y los pasos para instalar tanto el virtualizador como la máquina virtual realizada, y la manera que se deben de conectar los puertos a dicha máquina.

9.1.3. Práctica Piloto

Diseño experimental

Para realizar esta prueba se simuló un laboratorio del curso de Sistemas de Control 1 con una pareja de estudiantes, esto se llevó a cabo dentro de las instalaciones destinadas para dicho laboratorio y con el establecido para el mismo, correspondiendo a dos horas y 25 minutos (145 minutos).

A los alumnos se les brindó el material previo a la sesión de laboratorio, lo cual correspondía a tener la guía de instalación y guía de uso del *Crazyflie* descargadas y leídas, así como el archivo de la máquina virtual descargado.

El día de la práctica, se les asignó una computadora del laboratorio con los derechos de administrador, dicha computadora ya contenía la máquina virtual descargada (para ahorrar tiempo dado que era un archivo pesado) dentro de los archivos de la computadora. Por otra parte, se tenía la plataforma y el *Crazyflie* para verificar las conexiones luego de la instalación de instalación de la máquina virtual.

Con el material brindado, se dio inicio a la guía, donde se cronometró el tiempo desde el cual los estudiantes ingresaron al laboratorio, hasta finalizar la práctica. La idea de la práctica consistió en minimizar en lo posible la asistencia por parte del asistente de laboratorio, y tomar nota en donde se tuvieron complicaciones en la práctica.

Materiales y equipo

- Archivos de laboratorio (Manual de uso del *Crazyflie* y guía de instalación)
- Computadora con al menos 10 GB de espacio de disco duro y con acceso de administrador y 2 puertos USB libres.

- Plataforma de pruebas para el *Crazyflie*.
- *Crazyflie* con su respectiva antena de radio.

Resultados

La prueba piloto resultó ser exitosa, entre los resultados más importantes obtenidos se puede mencionar que la guía se logró resolver dentro del tiempo establecido, los alumnos cumplieron con los objetivos planteados sin la ayuda del auxiliar de laboratorio. Sin embargo, se tuvieron algunas sugerencias, las cuales se mencionan a continuación:

- Se identificaron 3 errores de ortografía, los cuales se corrigieron.
- Se entraron dos errores en redacción.
- Fue necesario definir el nombre del archivo y colocar correctamente el path donde se guardaran los datos.
- La guía de uso del *Crazyflie* y el manual de instalación de la máquina virtual se encontraban en un solo documento, se sugirió trabajarlos en documentos separados.
- Se sugirió indicar en la guía que debían de descargar previamente el archivo de la máquina virtual, dado el tamaño del mismo, lo cual podría presentar un problema intentar descargarlo en el tiempo de laboratorio.
- Se sugirió tener una memoria USB con la máquina virtual descargada, en caso algún estudiante no hubiese podido descargar la máquina virtual.
- Finalmente, se sugirió instalar el virtualizador y la máquina virtual en todas las computadoras del laboratorio de robótica.

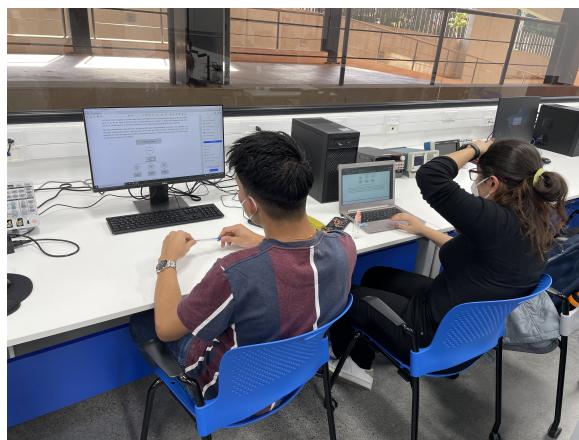


Figura 45: Estudiantes realizando la prueba piloto para el manual de instalación de herramientas.

La prueba piloto se realizó en 1 hora con 21 minutos desde que se inició con la p

9.2. Laboratorio 1 - Control PID

La primera práctica de laboratorio fue diseñada para el curso de Sistemas de Control 1, impartido en la Universidad del Valle de Guatemala. Durante esta práctica se estará trabajando con el controlador PID, siendo este uno de los controladores más versátiles y más utilizados en control clásico. Anteriormente para esta práctica se utilizaba un circuito correspondiente a un filtro pasa bajas de segundo orden, pero se consideró de importancia poder experimentar con un sistema más complejo que no correspondiera a un filtro. Fue por ello que se utilizó el controlador de orientación del ángulo de cabeceo del *Crazyflie 2.0* para realizar dicho laboratorio.

9.2.1. Herramientas

Para este laboratorio es imperativo que, a parte de la máquina virtual y de la plataforma, la computadora cuente con Matlab, dado que será necesario para ajustar el PID en la última parte del laboratorio.

Es importante que el estudiante esté familiarizado con la plataforma, que ya haya leído la guía de usuario y completado cada uno de los pasos que se describen en la guía; por otra parte, que cuente con las herramientas necesarias y que las sepa utilizar previo al uso de la guía de laboratorio.

9.2.2. Objetivos del laboratorio

Un paso importante fue poder identificar los objetivos del laboratorio. Dado que fue sustituido, es importante que las competencias aún se sigan cumpliendo en base a los objetivos planteados. Para esta práctica, los objetivos fueron:

- Estudiar los controlados PI y PID.
- Utilizar Matlab/Simulink para ajustar los parámetros del controlador de orientación del ángulo de cabeceo.
- Experimentar con el Crazyflie variando las constantes de control en base a la tabla de comportamientos de ajuste de constantes.
- Fuente de voltaje configurable a 3.7 voltios con 3 amperios.

9.2.3. Desarrollo del laboratorio

El primer laboratorio fue desarrollado en 3 partes, en la primera parte se planteó experimentar con las constantes del controlador para que el estudiante pudiera identificar el comportamiento de variar cada constante. En la segunda parte, el estudiante realizará un ajuste manual (empírico) de las constantes, dicho en otras palabras, a prueba y error en base al comportamiento que identificó en la primera parte. En la tercera y última parte, el

estudiante utilizará el modelo de la planta disponible en MATLAB y utilizará la herramienta para ajustes de PID de MATLAB (denominada *PID tunner*) para ajustar el PID al valor requerido en la práctica.

9.2.4. Práctica Piloto

Diseño experimental

Para realizar esta prueba se simuló un laboratorio del curso de Sistemas de Control 1 con una pareja de estudiantes, esto se llevó a cabo dentro de las instalaciones destinadas para dicho laboratorio y con el establecido para el mismo, correspondiendo a dos horas y 25 minutos (145 minutos).

A los alumnos se les brindó el material previo a la sesión de laboratorio, lo cual correspondía a tener la guía de instalación y guía de uso del *Crazyflie* descargadas y leídas, así como el archivo de la máquina virtual descargado.

El día de la práctica, se les asignó una computadora con la máquina virtual instalada y previamente probada, además de la plataforma y el *Crazyflie* con su respectiva antena de radio. Se explicó la guía de laboratorio y se dieron indicaciones generales de la misma, se dio una breve explicación del uso de la máquina virtual y se explicaron las herramientas que se estarían utilizando en la misma.

De igual manera, se cronometró el tiempo desde que inició el laboratorio hasta que finalizaron todas las pruebas; se trató de minimizar la participación del asistente de laboratorio dejando las instrucciones muy bien detalladas, y haciendo una explicación clara de la misma.

Materiales y equipo

- Guía de laboratorio y manual de uso de la máquina virtual.
- Computadora con la máquina virtual de *Bitcraze* instalada y con *Matlab*.
- Archivos de *Matlab* para resolver la guía.
- Plataforma de pruebas para el *Crazyflie*.
- *Crazyflie* con su respectiva antena de radio.
- Fuente de voltaje.

Resultados

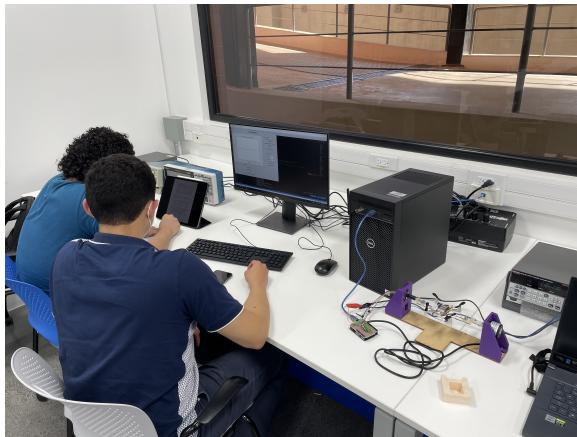


Figura 46: Estudiantes realizando la prueba piloto para la guía de laboratorio 1.

9.3. Laboratorio 2 - Control LQR

9.3.1. Objetivos del laboratorio

Para esta práctica, los objetivos estuvieron relacionados al curso de Sistemas de Control 2, los cuales se detallan a continuación.

- Estudiar el comportamiento del controlador LQR.
- Utilizar el Crazyflie 2.0 como planta de estudio para variar los parámetros del controlador PID mediante su planteamiento en LQR.
- Utilizar Matlab para ajustar los parámetros de un controlador LQR para el Crazyflie 2.0.

9.3.2. Desarrollo del laboratorio

El segundo laboratorio se desarrolló en dos partes, la primera parte consistió en una explicación de la formulación del PID en espacio de estados, definiéndolo mediante un controlador LQR, tal como se explicó en la Sección 8.7. Y la segunda parte consistió en la configuración de las constantes óptimas, que se obtienen por definición del LQR.

9.3.3. Herramientas

Las herramientas para el laboratorio 2 son las mismas detalladas en el laboratorio 1; es decir, la máquina virtual, la plataforma y *Matlab*.

9.3.4. Práctica Piloto

Diseño experimental

Para realizar esta prueba se simuló un laboratorio del curso de Sistemas de Control 2 con una pareja de estudiantes de dicho curso, esto se llevó a cabo dentro de las instalaciones destinadas para dicho laboratorio y con el establecido para el mismo, correspondiendo a dos horas y 25 minutos (145 minutos).

A los estudiantes se les proporcionó el material una semana antes de la sesión de laboratorio para que pudieran leerlo con anticipación y tener claro lo que estarían realizando en la práctica, este material correspondía a tener la guía de laboratorio, un manual de uso de la máquina virtual y los archivos de *Matlab* que se estarían utilizando en la práctica.

El día de la práctica, se les asignó una computadora con la máquina virtual instalada y previamente probada, además de la plataforma y el *Crazyflie* con su respectiva antena de radio. Se explicó la guía de laboratorio y se dieron indicaciones generales de la misma, se dio una breve explicación del uso de la máquina virtual en base a la guía presentada y se explicaron las herramientas que se estarían utilizando en la misma.

Materiales y equipo

Resultados

CAPÍTULO 10

Conclusiones

- Se desarrolló e implementó un aplicación en software que permite utilizar la plataforma física creada en la fase previa, en la cual se pudo diseñar, configurar y probar distintos controladores de orientación para el dron *Crazyflie 2.0*, así como se exploró y aprendió a utilizar y configurar el dron para el cual estaba diseñada la plataforma. En el proceso, se determinó que existe una documentación completa sobre el uso del dron por parte de la empresa *Bitcraze*, lo cual evidencia que es una plataforma que puede ser utilizada para investigación, desarrollo y educación.
- Se realizó un manual de operación en el cual se describe el funcionamiento del dron, además de una guía de como encenderlo y conectarlo a un teléfono inteligente. En dicho manual también se describen los pasos a seguir para instalar la máquina virtual que permite configurar el *Crazyflie 2.0* así como realizarle modificaciones a su *firmware*. Por otra parte, también se realizó un manual de recuperación de la máquina virtual, como un documento de respaldo en caso se pierda la información disponible.
- Se desarrolló una interfaz gráfica en Python la cual permite controlar el *Crazyflie 2.0* haciendo que éste se mueva según el ángulo de cabeceo que el usuario le ingrese dentro de un límite definido en la aplicación. A su vez, se logró modificar el controlador de orientación del dron a través de la interfaz, así como le lectura del ángulo de cabeceo a través del codificador rotacional disponible en la plataforma física.
- Se realizó una guía de laboratorio diseñada para el curso de Sistemas de Control 1, donde se utilizó el *Crazyflie 2.0* y la interfaz gráfica desarrollada para poder configurar, modificar y estudiar el controlador PID del ángulo de cabeceo del dron.
- Se desarrolló una guía de laboratorio para el curso de Sistemas de Control 2, en el cual se utilizó la misma interfaz diseñada para la guía de laboratorio 1, sin embargo el planteamiento del control fue en espacio de estados, siendo este un controlador LQR.

CAPÍTULO 11

Recomendaciones

- Se recomienda migrar del *Crazyflie 2.0* al *Crazyflie 2.1* puesto que el primero ya se encuentra descontinuado y a la hora que se requieran más drones será un problema el poder obtenerlos. La empresa actualmente brinda soporte para ambos drones, sin embargo llegará un momento donde solo se tenga soporte para la versión 2.1.
- Se recomienda modificar la plataforma física a materiales más livianos y rígidos, debido al crecimiento exponencial de la variedad de materiales disponibles para la impresión 3D es imperativo buscar una solución que permita obtener una plataforma más liviana lo cual le brindaría mayor agilidad al dron.
- Implementar el *Crazyflie* en el sistema de captura de movimiento instalado en la universidad para poder utilizar explorar el controlador de altura del dron; o bien, realizar aplicaciones de robótica *swarm* con los drones disponibles en la universidad.

CAPÍTULO 12

Bibliografía

- [1] C. 2.0, *Crazyflie 2.0*, 2021. **url:** <https://store.bitcraze.io/products/crazyflie-2-0>.
- [2] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński **and** P. Kozierski, «Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering,» **in** *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, IEEE, 2017, **pages** 37–42.
- [3] M. Nithya **and** M. Rashmi, «Gazebo-ROS-Simulink Framework for Hover Control and Trajectory Tracking of Crazyflie 2.0,» **in** *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, IEEE, 2019, **pages** 649–653.
- [4] T. S. Kim, K. Stol **and** V. Kecman, «Control of 3 DOF quadrotor model,» **in** *Robot Motion and Control 2007*, Springer, 2007, **pages** 29–38.
- [5] G. Martínez, «Plataforma de pruebas y ajustes de sistemas de control para vehículos multirrotores,» phdthesis, Universidad del Valle de Guatemala, 2019.
- [6] A. Tagay, A. Omar **and** M. H. Ali, «Development of control algorithm for a quadcopter,» *Procedia Computer Science*, **jourvol** 179, **pages** 242–251, 2021, 5th International Conference on Computer Science and Computational Intelligence 2020, ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2021.01.003>. **url:** <https://www.sciencedirect.com/science/article/pii/S1877050921000041>.
- [7] J. D. Guerra Quiñones, «Control de actitud de un cuadricóptero,» phdthesis, 2018.
- [8] Q. Quan, *Introduction to multicopter design and control*. Springer, Singapore, 2017. DOI: https://doi.org/10.1007/978-981-10-3382-7_4.
- [9] Bitcraze, *Crazyflie 2.0*, 2021. **url:** <https://www.bitcraze.io/products/crazyflie-2-1/>.
- [10] G. Silano **and** L. Iannelli, «CrazyS: A Software-in-the-Loop Simulation Platform for the Crazyflie 2.0 Nano-Quadcopter,» **in** *Robot Operating System (ROS): The Complete Reference (Volume 4)*, A. Koubaa, **editor**. Cham: Springer International Publishing, 2020, **pages** 81–115, ISBN: 978-3-030-20190-6. DOI: [10.1007/978-3-030-20190-6_4](https://doi.org/10.1007/978-3-030-20190-6_4). **url:** https://doi.org/10.1007/978-3-030-20190-6_4.

- [11] C. Luis **and** J. L. Ny, «Design of a trajectory tracking controller for a nanoquadcopter,» *arXiv preprint arXiv:1608.05786*, 2016.
- [12] G. P. Subramanian, «Nonlinear control strategies for quadrotors and CubeSats,» 2015.
- [13] I. Ramírez, *Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas*, january 2020. **url:** <https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas>.
- [14] Bitcraze, *Bitcraze Wiki*. **url:** <https://wiki.bitcraze.io/projects:virtualmachine:index>.
- [15] A. A. Kadhum **and** M. M. Abdulhussein, «Implementation dc motor as servomotor by using arduino and optical rotary encoder,» *Materials Today: Proceedings*, 2021, ISSN: 2214-7853. DOI: <https://doi.org/10.1016/j.matpr.2021.03.576>. **url:** <https://www.sciencedirect.com/science/article/pii/S2214785321026663>.
- [16] Bitcraze, *Getting started with the Crazyflie 2.X*, 2021. **url:** <https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/#flying>.
- [17] ——, *Crazyradio PA*, 2021. **url:** <https://www.bitcraze.io/products/crazyradio-pa/>.
- [18] ——, *La API de Crazyflie Python*, 2021. **url:** https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/user-guides/python_api/.
- [19] ——, *Logging and parameter frameworks*, 2021. **url:** <https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.06/userguides/logparam/>.
- [20] ——, *CRTP - Communication with the Crazyflie*, 2021. **url:** <https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.06/functional-areas/crtp/>.
- [21] ——, *Sensor to control*, 2021. **url:** https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.06/functional-areas/sensor_to_control/.
- [22] ——, *Controllers in the Crazyflie*, 2021. **url:** <https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.06/functional-areas/controllers/>.
- [23] ——, *The Commander Framework*, 2021. **url:** https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.06/functional-areas/commanders_setpoints/.

CAPÍTULO 13

Anexos

13.1. Manual de Recuperación de la Máquina Virtual

13.2. Manual de uso del *Crazyflie 2.0*

13.3. Guía de Laboratorio 1

13.4. Guía de Laboratorio 2

