Universidad Complutense de Madrid y Universidad Nacional de Educación a Distancia

Máster en Ingeniería de Sistemas y Control



Simulador Interactivo de la Plataforma CrazyFlie 2.0.

Blanca de la Parra de Toro

Directores: Dr. María Guinaldo Losada y Dr. Rafael Socas Gutiérrez

Curso 2019-2020

Convocatoria Junio

Máster en Ingeniería de Sistemas y Control



Simulador Interactivo de la Plataforma CrazyFlie 2.0

Proyecto Tipo A: Proyecto específico propuesto por un profesor

Blanca de la Parra de Toro

Directores: Dr. María Guinaldo Losada y Dr. Rafael Socas Gutiérrez

Espacio reservado para la hoja de calificaciones (que será introducida por la secretaria del Máster después de la defensa del mismo).

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado



Blanca de la Parra de Toro

Dedicatoria

A mi marido David, por ser siempre mi apoyo incondicional, y darme fuerzas en las largas noches de trabajo. A mi hija Sofía, mi luz, y a mis primas Isabel y Cristina, para que esta tesis les demuestre que todos los sueños se pueden alcanzar con trabajo, esfuerzo y perseverancia y les motive a alcanzar los suyos.

A mis tíos Javier y Nati, por ser para mí como mis segundos padres y apoyarme y valorarme siempre incondicionalmente.

A mi nona, que me vio empezar este proyecto, y que aunque no ha podido verlo terminar, seguro que estará sonriendo con sus labios rojos.

Por último, a la Universidad Complutense de Madrid y a la UNED, y a mis tutores, con mención especial a mi tutor Rafael Socas, por su apoyo y motivación y por brindarme la oportunidad de poder especializarme en el mundo del control y ayudarme a cumplir un sueño.

Blanca de la Parra de Toro

Resumen

En este apartado se incluye un resumen del proyecto.

Resumen

Este proyecto se desarrolla un laboratorio de simulación para drones utilizando la plataforma hardware CrazyFlie 2.0. Se trata de un entorno o *framework* diseñado e implementado de forma modular, pensado para que pueda seguir evolucionando con el tiempo y permitir al usuario, no sólo simular el CrazyFlie 2.0, sino otras muchas plataformas de vuelo.

Este simulador o laboratorio virtual permite al usuario interaccionar con el hardware del CrazyFlie 2.0 de tres formas distintas. Por un lado, le permite analizar los datos de un vuelo realizado con anterioridad, pudiendo representar los diferentes parámetros del sistema a partir de los datos de vuelo. Por otro lado, es posible conectar directamente el hardware, realizar un vuelo y grabarlo dentro del laboratorio, para su posterior análisis. Por último, para facilitar el diseño de controladores en la plataforma CrazyFlie 2.0, u otras en el futuro, este entorno permite al usuario simular el vuelo del CrazyFlie 2.0 de forma interactiva. Además, se pueden cambiar las ganancias de los controladores usados dentro del laboratorio, y lanzar de nuevo la simulación con unos simples clics de ratón.

El sistema ofrece además un Planificador de Misión, una herramienta sencilla e intuitiva para el diseño de trayectorias a seguir por el dron, pudiendo simularlas en fases posteriores. Además, tiene la capacidad de poder estimar la autonomía de la batería para la trayectoria diseñada, lo que permite conocer de antemano si el CrazyFlie 2.0 será capaz o no de realizar toda la misión diseñada.

Adicionalmente, este laboratorio incluye dos tipos de controladores pre-diseñados (PID periódicos y PID basados en eventos) y dos modelos del CrazyFlie 2.0 (lineal y no lineal), que permiten a los usuarios menos experimentado o a estudiantes de ingeniería de control, familiarizarse rápidamente con la plataforma CrazyFlie 2.0 (con otras en el futuro) e incluso poder probar ciertos controladores sencillos en el modo de simulación interactivo.

El programa posee una ventana principal que consta de la representación 3D del CrazyFlie 2.0, un *Attitude Indicator* (*AI*), varias cintas de registro o *strip charts* y un conjunto de alarmas. Una vez se inicia la simulación, el AI representará la actitud del CrazyFlie 2.0 en cada momento, la trayectoria 3D dentro de la ventana 3D, así como las variables principales que caracterizan el vuelo representadas en los *strip* charts, evolucionando con el tiempo. Además, el diseño de esta ventana principal es completamente configurable por el usuario, quedando guardada una vez se cierre la aplicación.

Por último, este laboratorio permite al usuario simular el CrazyFlie 2.0 de forma ideal, sin gasto de batería o sin limitaciones en los parámetros de vuelo o en los errores de los controladores, así como también habilita la simulación realista, teniendo en cuenta limitaciones del hardware, limitaciones o el uso de la batería, etc.

Palabras Clave

CrazyFlie 2.0, laboratorio virtual, simulador, simulador de vuelo, simulación 3D, simulador interactivo, simulador en tiempo real, planificación de misión, diseño de misión, control, PID, PID por eventos, seguimiento de trayectoria, cuadricóptero, nano cuadricóptero, nano helicópteros quadrotor.

CrazyFlie 2.0, *virtual lab, simulator, flight simulator, 3D simulator, interactive simulator, realtime simulator, mission planning, mission planner, control, PID, Events PID, tracking, quadcopter, quadrotor, nano quadcopter, nano quadrotor helicopter.*

Índice

1. Introducción 18
2. Descripción de la Plataforma CrazyFlie 2.0 22
2.1. Modelo matemático 22
2.2. Hardware 25
2.3. Software 25
3. Simulador Interactivo: Modos de Simulación 28
3.1. Modo de simulación *Offline* 28
3.2. Modo de simulación *Real-Time* 29
3.2.1. Intérprete 29
3.2.2. Interfaz de usuario personalizada 30
3.2.3. Guardado 30
3.2.4. Simulación 31
3.3. Modo *Hardware In the Loop (HIL)* 31
4. Simulador interactivo: descripción de los controles principales 34
4.1. Planificador de misión o *Mission Planner* 34
4.1.1. Estimación de gasto de batería 37
4.2. Ventana de simulación 3D 40
4.3. Indicador de actitud o *Attitude Indicator (AI)* 41
4.4. Gráficas dinámicas *o strip charts* 42
4.5. Alarmas 43
4.6. Control de tiempo 44
4.7. Controles de la simulación 45
4.8. Control de batería y valor en Voltios 45
4.9. Fichero de *settings* del programa 46
4.10. Fichero de *log* o de registro de errores del programa 46
5. Simulador Interactivo: Controladores 48
5.1. Controlador PID periódico 49
5.2. Controlador PID basado en eventos 50
6. Características adicionales del simulador 54
6.1. Escalabilidad 54

6.2. Seguridad y protección de datos 55
6.3. Operabilidad 55
6.4. Ventajas con respecto a otros simuladores del mercado 56
7. Simulaciones y Resultados de la Experimentación 58
7.1. Simulación *Offline* 58
7.2. Segunda Simulación *Offline* 61
7.3. Simulación *Real-Time* 65
7.4. Segunda Simulación *Real-Time* 66
8. Conclusiones y líneas futuras 70
8.1. Líneas de trabajo futuras 70
9. Referencias y Bibliografía 74
10. Abreviaturas y Acrónimos 78
A. Anexo A – Guía de usuario 82
A.1. Introducción 82
A.2. Guía de usuario 82
B. Anexo B - Limitaciones Físicas del CrazyFlie 2.0 y Limitaciones a los Errores del Controlador 117
B.1. Introducción 117
B.2. Limitaciones físicas al vuelo 117
B.3. Limitaciones en los Errores de los Controladores 118
B.4. Valores por defecto de las Limitaciones Físicas 118
C. Anexo C – Modelo Dinámico del CrazyFlie 2.0 121
C.1. Introducción 121
C.2. Hipótesis 121
C.3. Definiciones preliminares 121
C.3.1. Sistema de referencia 121
C.3.2. Variables 122
C.3.3. Ecuaciones dinámicas 122
C.3.3.1. Modelo No Lineal 123
C.3.3.2. Modelo Lineal 123
C.3.4. Ecuaciones del movimiento vertical 124
C.3.5. Ecuaciones del movimiento direccional 124
C.3.6. Ecuaciones del movimiento lateral 125
C.3.7. Ecuaciones del movimiento Longitudinal 125
C.3.8. Punto de *trim* o de equilibrio del CrazyFlie 2.0 125

D. Anexo D - Estadísticas del laboratorio virtual 127
D.1. Introducción 127
D.2. Estadísticas sobre las líneas de código 127
D.2.1. Tiempos de ejecución 127

Índice de Figuras

Figura 1 Representación de los ejes cuerpo del CrazyFlie 2.0. 22
Figura 2 Representación de los ejes cuerpo y los ejes inerciales, así como de distintos parámetros para el CrazyFlie 2.0. 23
Figura 3 Diálogo para la selección de los diferentes modos de simulación. 28
Figura 4 Secuencia de pasos en el modo *Real-Time.* 29
Figura 5 Configuración inicial al lanzar el Planificador de Misión en el laboratorio. 35
Figura 6 Trayectoria tipo *spline* diseñada con el Planificador de Misión. 36
Figura 7 Trayectoria tipo 'recta' diseñada con el Planificador de Misión. 36
Figura 8 Vista vertical del montaje del banco de ensayo para determinar la vida útil de la batería del CrazyFlie 2.0. 37
Figura 9 Vista desde atrás del montaje del banco de ensayo para determinar la vida útil de la batería del CrazyFlie 2.0. 38
Figura 10 Representación de los datos de ensayo para obtener el modelo de gasto de batería. 39
Figura 11 Ventana 3D del czSimulator. 40
Figura 12 AI incluido en la ventana principal del simulador. 41
Figura 13 Explicación de los ángulos del AI. 41
Figura 14 Ejemplo de cinta de registro o *strip chart.* 43
Figura 15 Alarmas de entrada en pérdida, batería baja y altitud demasiado alta. 44
Figura 16 Alarmas para la velocidad de rotación de los rotores. 44
Figura 17 Control de tiempo en el modo de simulación *Offline*. 45
Figura 18 Controles de la simulación en el modo de simulación *Offline*. 45
Figura 19 Representación del nivel de batería restante. 45
Figura 20 Arquitectura del control en el hardware CrazyFlie 2.0. 48
Figura 21 Estructura típica de un controlador PID. 49
Figura 22 Estructura de un controlador general basado en eventos. 50
Figura 23 Estructura de un controlador PID basado en eventos. 50
Figura 24 Lógica del detector de eventos de un controlador PID basado en eventos. 51
Figura 25 Diagrama del patrón de diseño Modelo Vista Controlador. 54
Figura 26 Estructura de la distribución del programa. 55
Figura 27 Ventana principal del czSimulator antes de iniciar una simulación *Offline*. 58
Figura 28 Trayectoria realizada por el CrazyFlie 2.0 en la simulación. 59
Figura 29 Módulo de la velocidad en la simulación. 59
Figura 30 Componente vertical del vector velocidad en la simulación. 60
Figura 31 Evolución de la altura en la simulación. 60
Figura 32 Planificador de misión con la trayectoria que se ha realizado en la simulación. 61
Figura 33 Planificador de misión con la trayectoria que se ha realizado en la segunda simulación. 61
Figura 34 Trayectoria realizada por el CrazyFlie 2.0 en la simulación. 62
Figura 35 Módulo de la velocidad en la simulación. 62
Figura 36 Componente vertical del vector velocidad en la simulación. 63

Figura 37 Evolución de la altura en la simulación. 63
Figura 38 Evolución de la coordenada X con el tiempo. 64
Figura 39 Error en la coordenada X del sistema, con respecto a la trayectoria deseada o consigna. 64
Figura 40 Evolución de la altura en la simulación. 65
Figura 41 Módulo de la velocidad en la simulación. 66
Figura 42 Trayectoria realizada por el CrazyFlie 2.0 en la simulación. 67
Figura 43 Módulo de la velocidad en la simulación. 67
Figura 44 Evolución de la altura con el tiempo. 68
Figura 45 Ventana de *settings* para la aplicación de límites físicos al vuelo. 117
Figura 46 Ventana de *settings* para la aplicación de límites a los errores de los controladores. 118
Figura 47 Representación de los ejes cuerpo del CrazyFlie 2.0. 121

Índice de Tablas

Tabla 1 con características de los módulos adicionales del CrazyFlie 2.0. 25
Tabla 2 Tiempo de duración de la batería para diversos % de empuje. 38
Tabla 3 Tabla comparativa de dos simuladores comerciales con el entorno desarrollado (czSimulator). 56
Tabla 4 Abreviaturas usadas en la memoria, la guía de usuario y el simulador czSimulator. 78
Tabla 5 Estadísticas de las líneas de código del simulador. 127
Tabla 6 Tiempos de ejecución de los principales procesos del simulador. 128

Capítulo 1
Introducción

En este capítulo se presentan los objetivos del presente proyecto y se compara con el actual estado del arte y otros trabajos existentes.

1. Introducción

En los últimos años, la industria de los drones o *UAV (Unmanned Aerial Vehicle* o Vehículos Aéreos no Tripulados) no ha hecho más que crecer (en torno a un 30% los últimos años), hasta tal punto que se espera que en 2021 genere unos ingresos cercanos a 12.000 millones de dólares [1]. Igualmente, el número de patentes en este sector ha crecido exponencialmente y el valor de las inversiones se sitúa en los últimos tres años entre 19 y 22 billones de dólares ([2] y [3]).

Es un mercado no sólo ocupado por grandes compañías de defensa o aviación, sino también por un gran número de empresas privadas y *startups*. Sus aplicaciones se extienden a lo largo de sectores tan dispares como el ocio y la cultura hasta la aviación y defensa, pasando por la agricultura, vigilancia, sector de la energía y la construcción. En este entorno de rápido crecimiento y alta inversión no es de extrañar que gran número de personas privadas, compañías e instituciones quieran llevar a cabo sus propias investigaciones.

Dentro del mundo de los drones, los cuadricópteros son de los vehículos más extendidos, siendo el CrazyFlie 2.0 una de las plataformas hardware más utilizadas. Ello se debe, entre otros factores, a su fácil manejo y su corta curva de aprendizaje gracias a todos los medios disponibles [4]. Se trata de una plataforma barata, ligera, de software abierto, con código en C, que muchas universidades y centros de investigación han decidido utilizar para desarrollo e investigación de nuevas técnicas de identificación, control, *tracking* o posicionamiento.

Por ello, el objetivo del proyecto es presentar un laboratorio virtual que facilite un entorno de trabajo abierto, intuitivo y sencillo al investigador, donde poder probar diferentes modelos dinámicos del CrazyFlie 2.0, generar las trayectorias a seguir (incluyendo limitaciones), probar diferentes controladores, e incluso reproducir simulaciones externas realizadas previamente con Matlab®. Este laboratorio proporciona una herramienta modular, para facilitar la tarea de sintonizar los controladores para el CrazyFlie 2.0 o de otras plataformas en el futuro, y hacerse una idea del comportamiento que éstas tendrán en la realidad.

Este laboratorio está diseñado, no sólo para el investigador experimentado, si no para el estudiante de ingeniería de control que desea ahondar un poco más en ciertos aspectos de la disciplina, usando para ello la plataforma CrazyFlie 2.0. Para ello, no sólo se permite la configuración de nuevos tipos de controladores, si no que se proporcionan diversos modelos ya predeterminados y ajustados para que el usuario pueda simular y "jugar" con la herramienta en pocos minutos.

El desarrollo de *frameworks* o plataformas donde facilitar el desarrollo de leyes de control con CrazyFlie no es nuevo, y se ha llevado a cabo en simuladores como el CrazyS de la Universidad de Sannio en Benevento [10] y [12], o la universidad de Poznan [11], que ha desarrollado el simulador 4*FLY*. Para una comparación en detalle entre el simulador desarrollado y los simuladores profesionales mencionados, se refiere al lector al apartado 6.4 de esta memoria.

Con respecto a las diferentes técnicas de control, debido a su precio económico y a la facilidad de manejo, el CrazyFlie 2.0 también se ha utilizado extensivamente en trabajos como [13] en el que se compara el diseño clásico de un PID con un controlador LQ. En [14]-[15], en los que se diseña un controlador LQR, en [16], en el que se diseña y despliega un controlador PID Fuzzy para CrazyFlie 2.0 o en [17], en el que se estudia la generación de trayectorias con *B-Splines* o *splines* naturales, y se aplica una técnica de control por backstepping.

Con respecto a otros trabajos, el simulador que se presenta en este proyecto tiene la ventaja de haberse realizado en lenguaje Python, utilizando un patrón de diseño de software conocido como *Model View Controller* o MVC, en el que el código está implementado de forma modular y, por tanto, es fácilmente escalable. Permite la interacción directa, tanto con el CrazyFlie 2.0 como con Matlab®, por lo que proporciona al usuario todo lo necesario para familiarizarse con el vuelo del CrazyFlie 2.0. Además, facilita el análisis de datos de vuelo y el estudio y optimización de controladores para CrazyFlie 2.0, todo ello dentro de un único entorno.

Otra ventaja que presenta este sistema es la posibilidad de definir las trayectorias a seguir por el dron, a través de un módulo de diseño de misión o Mission Planner, que permite al usuario modelar también limitaciones de vuelo del CrazyFlie 2.0.

Adicionalmente, todos los ficheros de configuración pueden ser exportados a Matlab®, permitiendo de esta manera usar todo el potencial de esta herramienta para el diseño y *tunning* de las distintas estrategias de control, pudiendo después leer ese fichero de vuelta en el simulador, permitiendo al usuario visualizar de forma clara el vuelo de la plataforma y solucionar rápidamente los problemas observados.

A continuación, se presenta en más detalle el hardware y software que componen esta plataforma.

Capítulo 2
Descripción de la Plataforma CrazyFlie 2.0

En este capítulo se describe la plataforma de vuelo CrazyFlie 2.0 utilizada en este proyecto, incluyendo el software y hardware que lo componen.

2. Descripción de la Plataforma CrazyFlie 2.0

2.1.Modelo matemático

El programa czSimulator (CZS) posee dos modelos de CrazyFlie 2.0: uno lineal y otro no lineal.

En el modo de simulación *Real-Time*, se puede elegir qué tipo de modelo se desea utilizar y comprobar rápidamente los efectos que esta decisión tiene en la simulación.

En la Figura 1 se muestran los ejes que suelen utilizarse normalmente para definir el movimiento del CrazyFlie 2.0. En este caso, se han usado estos mismos ejes cuerpo.

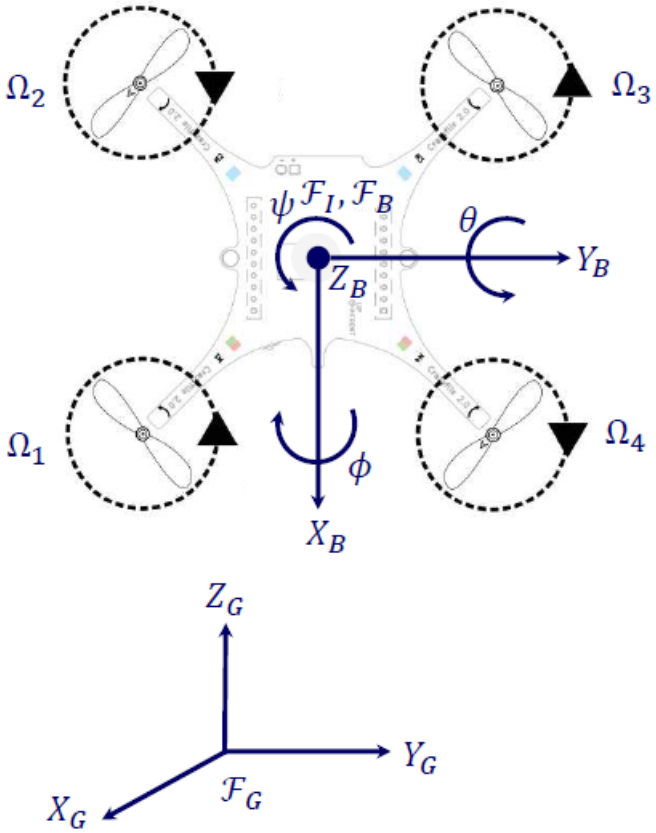


Figura 1 Representación de los ejes cuerpo del CrazyFlie 2.0.

Según la teoría del *quadrotor* o cuatrirotor, las velocidades angulares de rotación de los motores, las fuerzas de empuje de cada motor, etc. se definen como se detalla en la Figura 2, incluida a continuación:

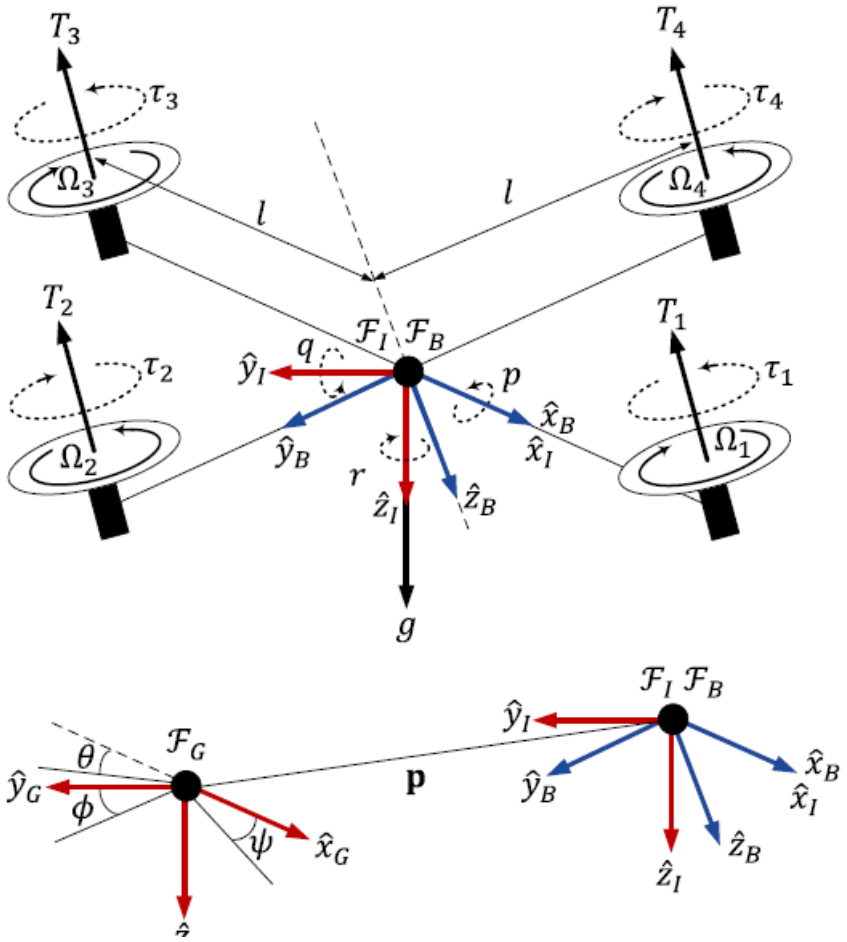


Figura 2 Representación de los ejes cuerpo y los ejes inerciales, así como de distintos parámetros para el CrazyFlie 2.0.

En forma matricial, las ecuaciones dinámicas del modelo no lineal del hardware CrazyFlie 2.0 pueden expresarse como sigue en las siguientes ecuaciones.

Las ecuaciones dinámicas de las aceleraciones angulares son:

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{u_2}{I_{xx}} & q \cdot r \cdot \frac{I_{yy} - I_{zz}}{I_{xx}} \\ \frac{u_3}{I_{yy}} & p \cdot r \cdot \frac{I_{zz} - I_{xx}}{I_{yy}} \\ \frac{u_4}{I_{zz}} & p \cdot q \cdot \frac{I_{xx} - I_{yy}}{I_{zz}} \end{pmatrix} \tag{1.1}$$

Las ecuaciones dinámicas de los Ángulos de Euler son:

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}=\begin{pmatrix}1 & T_\theta\cdot S_\phi & T_\theta\cdot C_\phi\\ 0 & C_\phi & -S_\phi\\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta}\end{pmatrix}\cdot\begin{pmatrix}p\\ q\\ r\end{pmatrix} \tag{1.2}$$

Donde,

- $(p \quad q \quad r)^T$: son las velocidades angulares del sistema
- $(I_{xx} \quad I_{yy} \quad I_{zz})^T$: son los momentos de inercia del sistema
- $(\phi \quad \theta \quad \psi)^T$: representa la orientación del sistema (esto es, de los ejes cuerpo con respecto a los ejes inerciales)
- $(u_2 \quad u_3 \quad u_4)^T$: representa los momentos que provocan las rotaciones del sistema
- C_Φ : representa el coseno de Φ
- S_Φ : representa el seno de Φ
- T_Φ : representa la tangente del ángulo Φ

Con las ecuaciones (1.1) y (1.2) se pueden obtener las velocidades lineales del sistema y la posición, como se detalla en las siguientes ecuaciones:

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix}=\frac{1}{m}\cdot\begin{pmatrix}0\\ 0\\ u_1\end{pmatrix}+g\cdot\begin{pmatrix}-S_\theta\\ C_\theta S_\phi\\ C_\theta C_\phi\end{pmatrix}\cdot\begin{pmatrix}qw-rv\\ ru-pw\\ pv-qu\end{pmatrix} \tag{1.3}$$

La posición de la aeronave en el espacio es:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{pmatrix}=\begin{pmatrix}C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi\\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{pmatrix}\cdot\begin{pmatrix}u\\ v\\ w\end{pmatrix} \tag{1.4}$$

Donde,

- $(x \quad y \quad z)^T$: es la posición del sistema
- $(u \quad v \quad w)^T$: es la velocidad lineal del sistema
- m : es la masa del sistema

Para más información acerca de los modelos dinámicos del CrazyFlie 2.0 implementados en el simulador, consúltese el Anexo C.

2.2.Hardware

Como ya se ha dicho, el CrazyFlie 2.0 es una plataforma de vuelo, de ala rotatoria, ligera y de software abierto.

Consta de un cuadricóptero o *quadrotor*, una batería, una radio, y un altímetro de alta precisión. La radio sirve de interfaz para poder conectar y controlar el CrazyFlie 2.0 desde el ordenador.

Existen otros posibles hárdwares, como el *buzzer deck*, el sistema de posicionamiento *locomotion*, *bigQuad deck*, entre otros, que se pueden añadir a esta plataforma, según sea el objetivo del estudio. El diseño mecánico del CrazyFlie 2.0 permite montar estas tarjetas "extra" encima o debajo del bloque principal, haciéndolo extremadamente flexible.

Los principales componentes del CrazyFlie 2.0 y sus características se detallan en la siguiente tabla [5].

Nombre	Peso [gr]	Consumo [mA]	Posición de montaje
LED-ring	3.3	0-700	Abajo
Qi charger	5	N/A	Abajo
Buzzer	1.8	10	Arriba/Abajo
Big quad	3.8	N/A	Arriba/Abajo
UWB LPS	3.3	160	Arriba/Abajo
Micro-SD	1.7	~30	Arriba/Abajo
Z-Ranger	1.3	~15	Abajo
Flow	1.6	~40	Abajo
Obstacle Avoidance	N/A	~0.3	Arriba
Multi-ranger	N/A	~90	Arriba
Mocap marker deck	N/A	0	Arriba
Z-Ranger v2	1.3	~15	Abajo
Flow v2	1.6	~40	Abajo
Lighthouse-4	2.7	~40	Arriba

Tabla 1 con características de los módulos adicionales del CrazyFlie 2.0.

2.3.Software

Posee un *firmware* escrito en lenguaje C que permite la reprogramación del dispositivo, lo que dota al usuario de la posibilidad de poder desarrollar su controlador dentro del laboratorio presentado, y probarlo después en el hardware del CrazyFlie 2.0 a través del cambio de los conocidos como controladores *Off-board*. Además, existe una amplia comunidad de desarrolladores contribuyendo a la mejora del *firmware* existente (2020.02 en el momento de escribir esta memoria).

Este cuadricóptero puede ser controlado, tanto desde un dispositivo móvil como desde el ordenador, a través del cliente "CrazyFlie PC Client", desarrollado por Bitcraze. Si se usa un móvil, lo único que se requiere es Android o iOS, que soporte una conexión Bluetooth de baja energía (BLE) y en el que se haya instalado la aplicación "CrazyFlie Client".

Por otro lado, si se prefiere un ordenador, lo único que se necesita es disponer de un mando de consola (para el control de la plataforma) y haber instalado la aplicación cliente de CrazyFlie en el ordenador [6]. Esta aplicación, junto con la radio, permite conectar el CrazyFlie 2.0 al ordenador y poder observar y controlar el vuelo en la ventana de usuario.

El laboratorio presentado permite también al usuario conectar el hardware del CrazyFlie 2.0 con el programa, de forma que el usuario pueda visualizar en tiempo real distintos parámetros de vuelo del CrazyFlie 2.0, así como el consumo de batería o la comanda a los motores, entre otras características.

Si se desea profundizar en el desarrollo de código para CrazyFlie, se recomienda visitar la página de los creadores de la plataforma (2011), fundadores de la empresa Bitcraze [7]. En ella se pueden encontrar, desde el repositorio de código de GitHub [8] a proyectos y tutoriales.

Capítulo 3 Simulador Interactivo: Modos de Simulación

En este capítulo se describen los tres modos de simulación implementados en el laboratorio: Offline, Real-Time y HIL.

3. Simulador Interactivo: Modos de Simulación

En esta sección de la memoria se describen los tres principales modos de simulación que se han implementado en el simulador.

Como se ha comentado con anterioridad, estos tres modos son:

- *Offline*
- *Real-Time*
- *Hardware-In-The-Loop (HIL)*

Cuando se inicia el programa, se visualiza el siguiente cuadro de diálogo, a través del cual se puede seleccionar el modo de simulación:

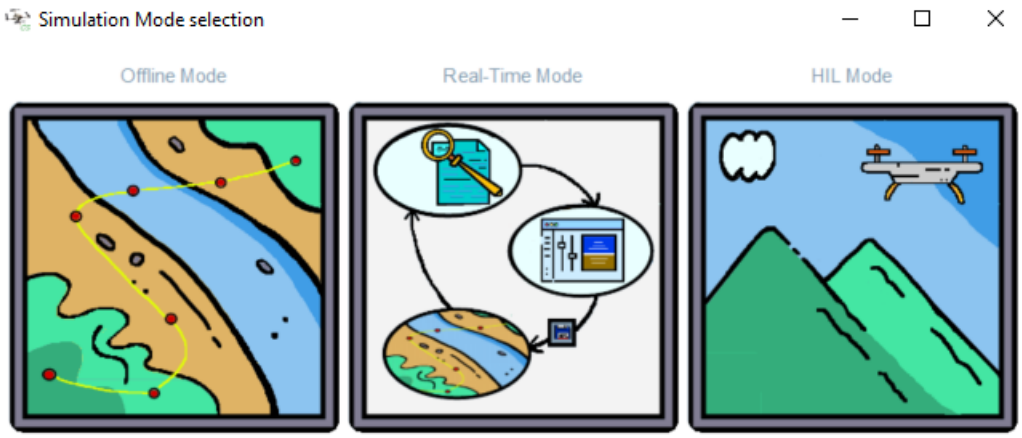


Figura 3 Diálogo para la selección de los diferentes modos de simulación.

Comentar que los logos han sido diseñados por la autora de este proyecto, y contienen referencias a la infancia de la misma. A continuación se detalla cada uno de estos modos.

3.1.Modo de simulación *Offline*

En el modo *Offline*, el programa pedirá al usuario un fichero *.mat de Matlab® en el que se esperan una serie de vectores, procedentes de una simulación realizada previamente en Matlab®. Una vez que el usuario seleccione el fichero, deberá "conectar" estas variables con los nombres de las variables reconocidos por el programa. El entorno dispone de la funcionalidad de realizar esta conexión entre nombres de variables de forma automática, para facilitar la tarea al usuario.

Es importante que todas las señales en el fichero de Matlab® **sean escalares**, puesto que los *arrays* estructurados, tal y como se dice en [25], no están soportados, y por tanto la librería de Python usada para leer el fichero de Matlab®, no es capaz de convertir estas estructuras a los vectores de datos requeridos por el simulador.

Hecho esto, la simulación podrá iniciarse, y todas las gráficas, el AI, la ventana 3D, etc. empezarán a representar el vuelo del CrazyFlie 2.0 leído del fichero. A su vez, se dispone de una serie de botones para el control de la simulación, que le permiten pausarla, pararla o reiniciarla cuando sea necesario.

En este modo de simulación el usuario puede elegir aplicar dos tipos de limitaciones, limitaciones físicas al vuelo del CrazyFlie y limitaciones a los errores del controlador. Para mayor información sobre las limitaciones (físicas y a los controladores, refiérase el lector al Anexo B de esta memoria. Además, se puede consultar cómo aplicar las diversas limitaciones, acudiendo a la Guía de Usuario, incluida en el Anexo A de esta memoria.

La clave de este modo de simulación es que permite visualizar muy rápidamente una simulación ya hecha en Matlab®. Si se quiere poder realizar realimentación a la misma, debe acudirse al modo *Real-Time*.

3.2.Modo de simulación *Real-Time*

En el modo de simulación *Real-Time*, se permite modificar un diseño de controlador realizado en Matlab® con nuevos valores de las ganancias elegidos en este laboratorio virtual, ejecutar de nuevo la simulación de Matlab® (desde el propio simulador) y representar todas las variables del vuelo para analizar las prestaciones del nuevo controlador diseñado.

En el siguiente diagrama se muestra la secuencia de pasos que el simulador llevará a cabo en este modo:

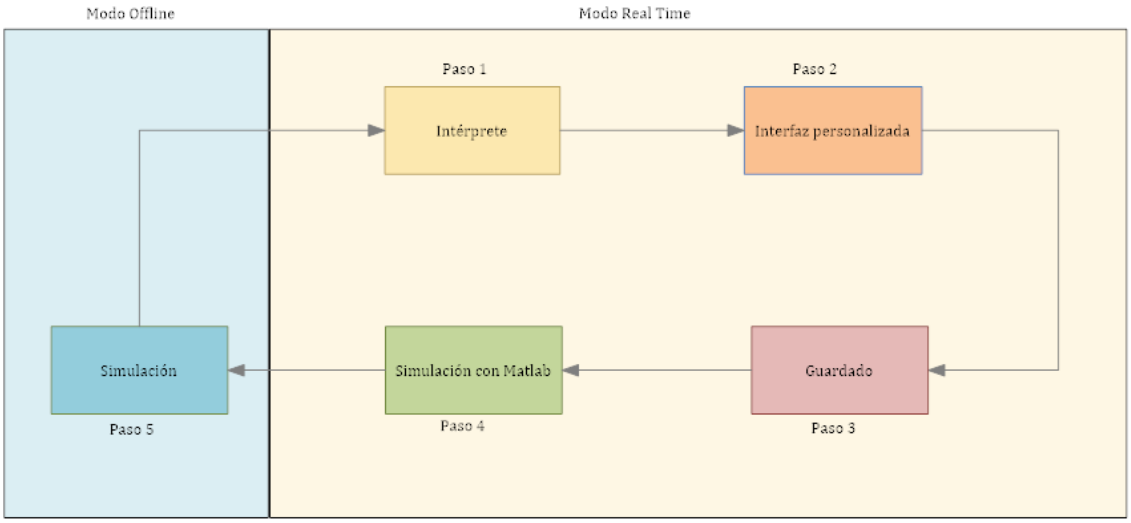


Figura 4 Secuencia de pasos en el modo *Real-Time.*

A continuación se describen cada uno de los pasos en más detalle:

3.2.1. Intérprete

Este módulo del laboratorio virtual es capaz de leer un fichero de Simulink® y extraer todas las ganancias de controladores encontradas en el mismo (de determinados tipos reconocidos por el simulador). Una limitación, por el momento, para este intérprete es que se ha asumido que todos los controladores a interpretar estarán dentro de un bloque en Simulink® llamado "*Off-board Controllers*". Esta limitación es fácilmente eliminable, y el intérprete puede ampliarse tanto como se desee. Esta generalización se plantea como una nueva prestación para futuras versiones del simulador. Esta lista de ganancias que se ha

leído le servirá al programa para personalizar la interfaz de usuario (*Graphical User Interface* o GUI) como se mostrará en el siguiente paso.

En este momento, el usuario podrá modificar las ganancias como desee, y una vez que el diseño cumpla con los requisitos deseados, podrá ejecutar una simulación del controlador en Matlab®. En el caso de no disponer de un fichero de simulación, el programa generará uno automáticamente, en el que además se puede incluir una trayectoria específica a seguir. Esto se explicará más adelante en el apartado sobre el Planificador de Misión, consultar el apartado 4.

El intérprete reconoce todo tipo de controladores mientras el bloque incluya la palabra clave *Controller* en el nombre del bloque o sistema en Simulink®, aunque sólo presentará en la interfaz de usuario, aquellos puertos de entrada del controlador cuyo valor sea constante.

3.2.2. Interfaz de usuario personalizada

Después de que el programa lea el fichero de Simulink® proporcionado por el usuario, éste creará una interfaz de usuario personalizada, basada en ese fichero de Simulink® leído. Esta interfaz incluirá una serie de *sliders* o barras de control, una por cada ganancia de controlador interpretado en el fichero, que serán agrupadas por controlador y página, bloque o sistema de Simulink®.

Como ya se ha dicho, se podrán modificar los valores de cada una de las ganancias de los controladores, con el objetivo de poder simular con las nuevas ganancias muy rápidamente.

La principal ventaja de este método es que el usuario puede simular un controlador diseñado con Matlab®/Simulink® y modificar sus ganancias, sin necesidad de poseer una licencia de estos programas. Esto permite que no sólo experimentados profesionales utilicen esta herramienta si no que proporciona un laboratorio virtual para estudiantes que quieren experimentar con controladores que puedan aplicar a este tipo de robots aéreos.

Además, pensando en los usuarios menos experimentados, czSimulator incluye dos tipos de controladores por defecto, controladores PID y controladores PID basados en eventos para aplicar al CrazyFlie 2.0. Ambas arquitecturas de controlares pueden usarse en este modo de simulación, de forma que no se necesite estar en posesión de ningún otro controlador ya diseñado. Para saber cómo seleccionar el uso de estos controladores por defecto, refiérase a la Guía de usuario, incluida en el Anexo A de esta memoria.

3.2.3. Guardado

En este paso, laboratorio virtual czSimulator guardará "de vuelta" todo el modelo de Simulink® interpretado, sustituyendo los antiguos valores de las ganancias por los nuevos valores configurados por el usuario.

3.2.4. Simulación

Por último, una vez que el usuario ha modificado las ganancias de los controladores puede volver a simularlo en el czSimulator.

Una vez el programa termine de realizar la simulación del controlador en Matlab®, el programa iniciará el modo *Offline*, leyendo por defecto el fichero generado por Matlab® en la nueva simulación. El usuario podrá elegir la asociación de variables que desee e iniciar la visualización de la nueva simulación.

En este momento, todas las cintas de registro (o gráficas dinámicas) comenzarán a representar las variables, así como la ventana 3D y el AI la posición del dron. Si el resultado obtenido no es el esperado, puede volverse al modo *Real-Time*, para modificar las ganancias, guardar el modelo de Simulink® y realizar la simulación, repitiendo de esta forma los pasos 2, 3 y 4 de la Figura 4. Para más detalles sobre cómo llevar a cabo todo el proceso, consultar la Guía de usuario en el Anexo A de esta memoria.

3.3. Modo *Hardware In the Loop (HIL)*

Este modo permite conectar un CrazyFlie 2.0 real al simulador. El hardware CrazyFlie debe estar encendido y la radio Crazyradio PA USB *dongle* debe estar conectada al ordenador antes de iniciar el laboratorio virtual. Si el programa no encuentra la radio conectada, se dispone de un botón para forzar al simulador a volver a escanear los dispositivos.

Una vez que el CrazyFlie 2.0 esté conectado al simulador, tanto la ventana 3D como el AI reflejarán el comportamiento del mismo. Además, la gran ventaja de este laboratorio virtual es que permite al usuario guardar el vuelo realizado con el CrazyFlie 2.0 real. El fichero será guardado en formato de un fichero *.mat de Matlab®, que puede leerse de nuevo en el modo *Offline* del programa, para poder así analizar el comportamiento del CrazyFlie 2.0, tanto cuando es manejado por un piloto como por el controlador que esté implementado en ese momento en el dron. Para una información más detallada, consúltese la Guía de usuario en el Anexo A de esta memoria.

Capítulo 4 Simulador Interactivo: Descripción de los Controles Principales

En este capítulo se detallan todos los controles incluidos en la ventana principal del simulador.

4. Simulador interactivo: descripción de los controles principales

Para proporcionar un entorno lo más sencillo posible de utilizar, el simulador está dotado de un Interfaz de Usuario o *User Interface* gráfico.

Esta interfaz gráfica está compuesta principalmente por una ventana desde la que se puede acceder a todas las funcionalidades del programa, que a su vez se compone de diferentes menús, botones y cintas de registro o *strip charts*, que permiten el manejo del programa y la visualización de los parámetros principales que caracterizan el vuelo de la plataforma CrazyFlie 2.0.

Las principales características del simulador son:

- Interfaz de usuario sencilla y configurable
- Tres modos de simulación
  - Modo *Offline*
  - Modo *Real-Time*
  - Modo *Hardware-In-The-Loop* (HIL)
- Capacidad de definición de trayectorias
- Simulación 3D
- Registro de datos de vuelo
- Aviso de cercanía a límites definidos por el usuario
- Capacidad de acelerar el tiempo
- Estimación de la vida útil de la batería
- Registro y manipulación de errores
- Capacidad de reconfiguración de la ventana principal y registro de la misma

Todas estas capacidades se traducen en una serie de controles principales que se encuentran en la ventana principal del simulador, y de ficheros y que se detallan en la siguiente enumeración:

- Módulo de planificación de misión o *Mission planner*
- Ventana de simulación 3D
- Ventana de *Attitude Indicator* o AI
- Cintas de registro o s*trip charts* con las variables principales del CrazyFlie 2.0
- Alarmas
- "Acelerador" de tiempo
- Simulador de consumo de batería
- Grabadora de vuelo real del CrazyFlie 2.0
- Fichero de *settings*
- Fichero de registro de errores o *log*

En las siguientes secciones se describen en detalle cada uno de los puntos enumerados.

4.1.Planificador de misión o *Mission Planner*

Este módulo del programa permite al usuario diseñar de una forma gráfica e intuitiva la trayectoria a seguir por el CrazyFlie 2.0. Una vez diseñada, se puede exportar a un fichero de Matlab®, para poder utilizarla en una simulación, o para utilizarla en el modo *Real-*

Time del programa. Si se dispusiera en el entorno real del sistema *loco positioning system*, incluso podría comandarse al CrazyFlie 2.0 el seguimiento de esta trayectoria diseñada.

En la siguiente figura se muestra la configuración inicial del Planificador de Misión:

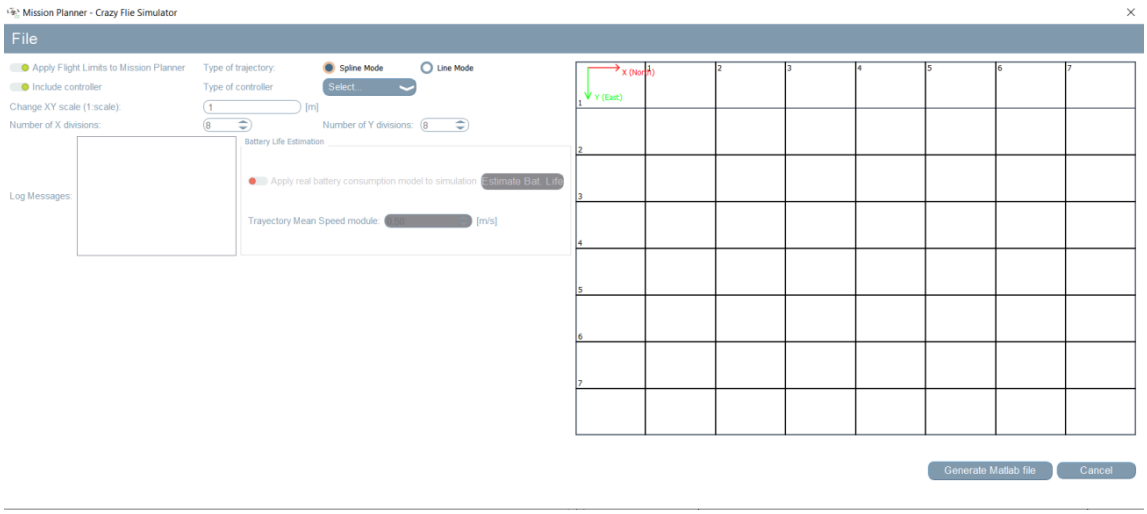


Figura 5 Configuración inicial al lanzar el Planificador de Misión en el laboratorio.

La trayectoria se configura seleccionando una serie de puntos de control o *waypoints*. La configuración de estos puntos de control incluye ciertas limitaciones físicas del CrazyFlie 2.0, que se explican en detalle en el Anexo B de esta memoria. Se pueden añadir tantos puntos de control en el mapa como desee, y configurar la altura de cada uno, teniendo en cuenta las limitaciones. Además, puede configurar tanto el tamaño del mapa utilizado como su escala.

Adicionalmente, es posible seleccionar qué tipo de trayectoria quiere que el programa genere para pasar por todos los puntos de control. En la versión actual, este laboratorio virtual dispone de dos opciones implementadas: líneas rectas y *splines* cúbicas o naturales, pero éste es uno de los aspectos ampliables en futuras versiones del laboratorio.

En las siguientes figuras se muestran unos ejemplos de los dos tipos de trayectorias soportados:

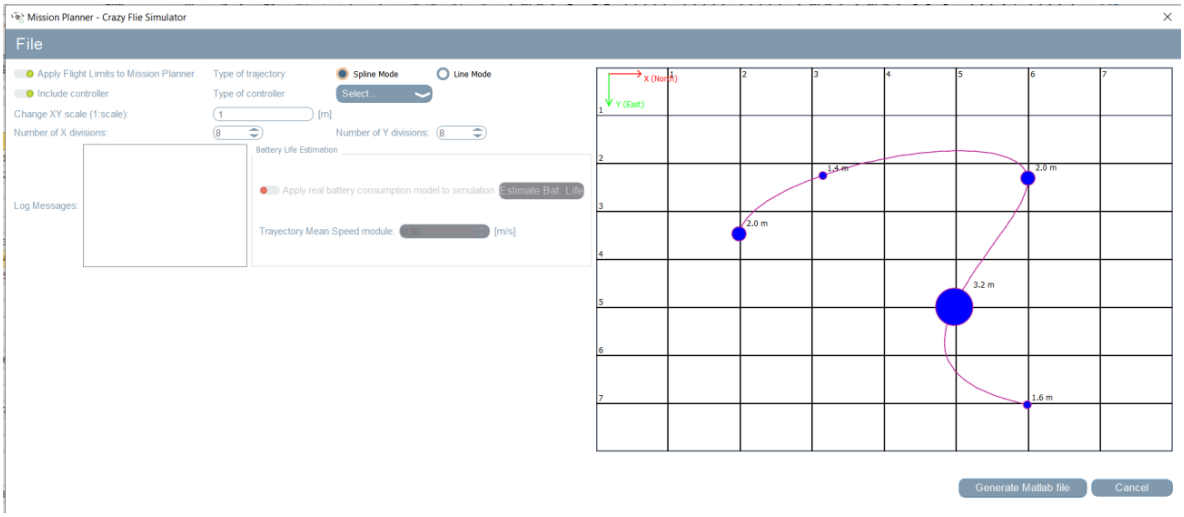


Figura 6 Trayectoria tipo *spline* diseñada con el Planificador de Misión.

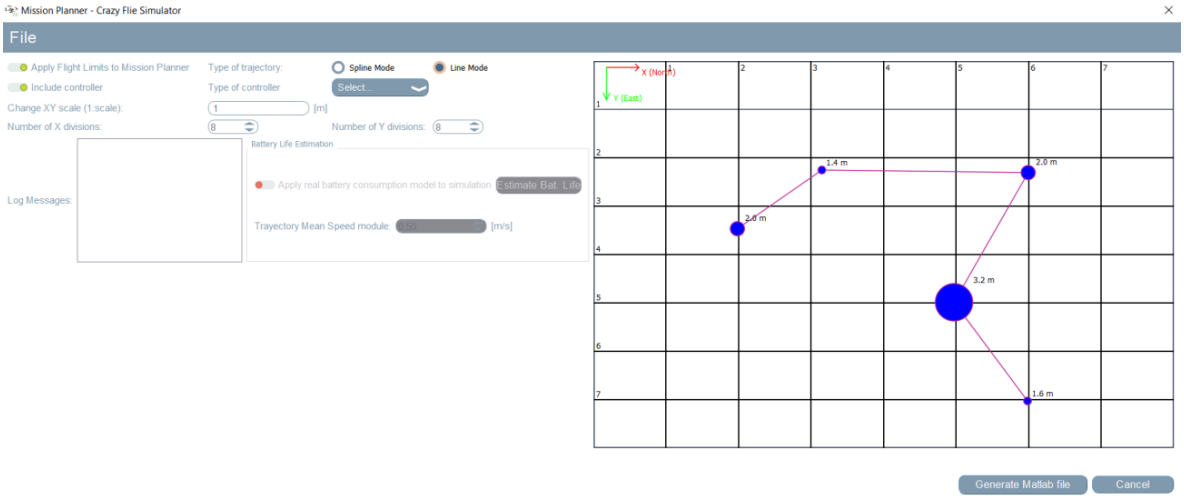


Figura 7 Trayectoria tipo 'recta' diseñada con el Planificador de Misión.

Una vez que se confirma al programa que se ha terminado de configurar la trayectoria, éste generará una *spline* natural o cúbica o una recta (según se haya configurado) que pasará por todos los puntos de control y que será la trayectoria real que seguirá el CrazyFlie 2.0.

A la hora de exportar la trayectoria a Matlab®, el usuario puede configurar el número de elementos de los vectores posición exportados por el programa. Para calcular este número, czSimulator se basa en dos parámetros, el tiempo total de simulación y el incremento de tiempo.

La ecuación utilizada para el cálculo del número de puntos es la siguiente:

$$npts = \frac{T_m}{dt} + 1 \qquad (1.5)$$

Dónde:

- T_m : es el tiempo total de simulación
- dt : es el incremento de tiempo en la simulación

Por ejemplo, si se configura un tiempo de simulación de 10s y un incremento de tiempo de 0.1s se obtendrán vectores de trayectoria con 101 puntos. Esto proporciona al usuario una herramienta para llegar a un punto de equilibrio entre *performance* y precisión en la simulación que quiera realizar en Matlab®. Para más información sobre cómo configurar la trayectoria a seguir por el CrazyFlie 2.0, y sobre todas las opciones de esta herramienta, consúltese el Anexo A de esta memoria.

4.1.1. Estimación de gasto de batería

Se ha implementado un modelo estimado de gasto de batería del CrazyFlie 2.0. Para la obtención del modelo, primero se realizaron una serie de experimentos en un "banco de ensayo" para el CrazyFlie 2.0, de forma que se limitaba el porcentaje disponible de empuje para los rotores (a través de la herramienta CrazyFlie PC Client).

En las Figura 8 y Figura 9 puede apreciarse el montaje llevado a cabo para disponer de un banco de ensayos estático y poder ensayar el CrazyFlie 2.0 a los diferentes perfiles de empuje para determinar la vida útil de la batería.

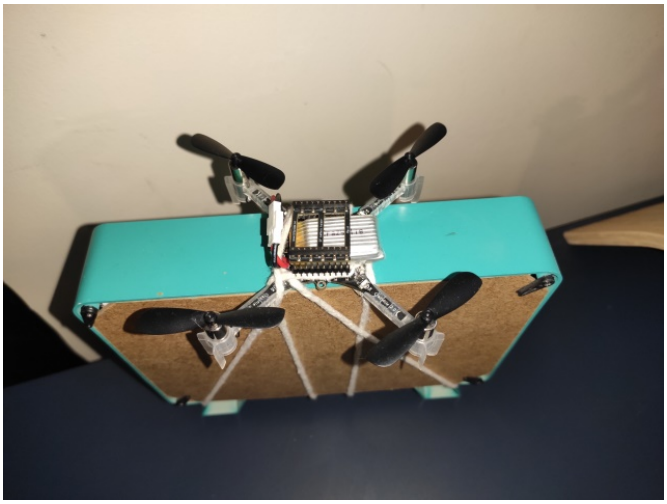


Figura 8 Vista vertical del montaje del banco de ensayo para determinar la vida útil de la batería del CrazyFlie 2.0.

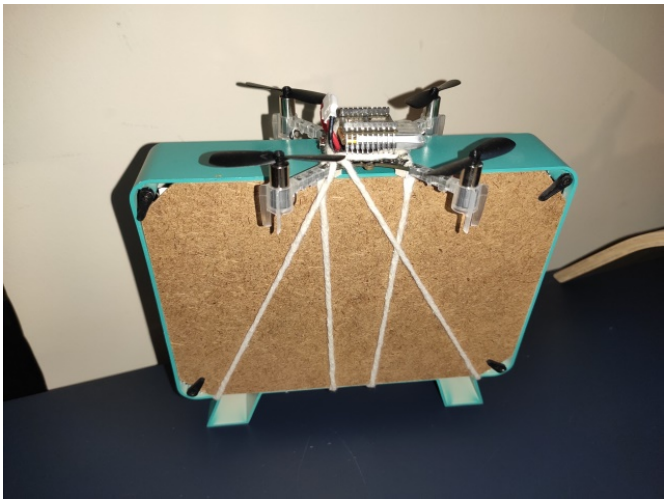


Figura 9 Vista desde atrás del montaje del banco de ensayo para determinar la vida útil de la batería del CrazyFlie 2.0.

En la siguiente tabla se presentan los resultados obtenidos en el ensayo:

Empuje	Duración de la batería [minutos]
10	12.2067
20	3.7267
30	4.1533
40	1.5483
50	3.8417
60	5.5667
70	6.4883
80	0.2592
90	3.315
100	3.1883

Tabla 2 Tiempo de duración de la batería para diversos % de empuje.

Si se representan estos datos, se obtiene la siguiente figura:

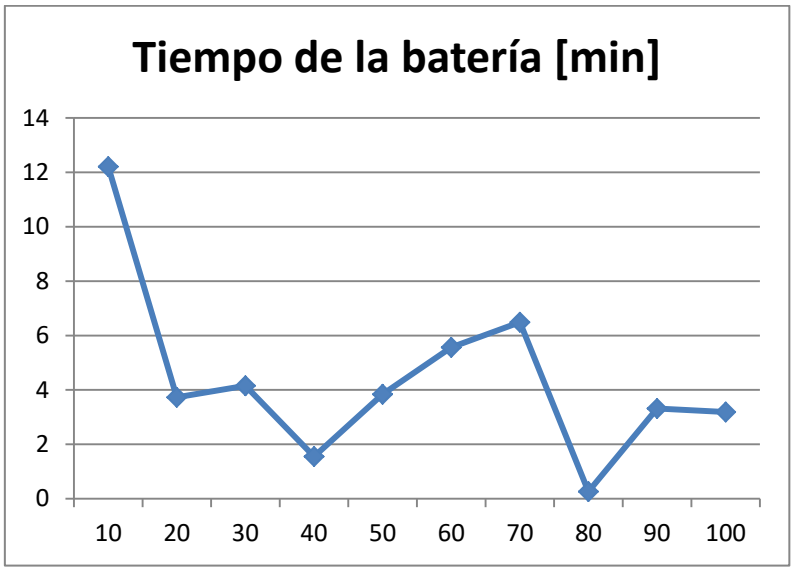


Figura 10 Representación de los datos de ensayo para obtener el modelo de gasto de batería.

Como puede observarse en la Figura 10, se han obtenido datos con mucha dispersión. Para confirmar los ensayos y que la vida de la batería obtenida no se debiera a agotamiento de la misma, ciertos ensayos fueron repetidos, obteniendo similares resultados.

En vista de la población de datos, se decidió aproximar un modelo por mínimos cuadrados (y no una curva polinómica que pasase por todos los datos de ensayo, como se pensó en un principio.

Aplicando esta técnica, el modelo obtenido es el siguiente:

$$t_{flight} = a + b \cdot T_{\%} + c \cdot T_{\%}^3 \tag{1.6}$$

Dónde:

- t_{flight} : es el tiempo de vida de la batería en minutos
- $T_{\%}$: es el porcentaje de empuje (10%, 20%, etc.)
- $a,\ b\ y\ c$:son los valores de los coeficientes de la aproximación

$$a = 9.373917745 \tag{1.7}$$

$$b = -0.13048371 \tag{1.8}$$

$$c = 7.37885 \cdot 10^{-6} \tag{1.9}$$

Este modelo permite implementar una estimación de la autonomía particularizada para las trayectorias que el usuario diseñe en el módulo de Planificación de Misión. Lo que se ha hecho es, estimar el porcentaje de empuje en función de las distintas alturas de los puntos de control de la trayectoria. Además, se ha tenido en cuenta una velocidad lineal media de

recorrido de la trayectoria, que el usuario puede configurar. Con estos datos, el programa puede estimar la autonomía de la batería según la misión y avisar al usuario de si el CrazyFlie 2.0 será capaz de realizarla antes del gasto completo de la batería o no.

Además, puede elegirse simular la trayectoria sin tener en cuenta el gasto de batería (modo de simulación ideal) o, por el contrario, se puede configurar que el simulador pare la simulación en el caso de que la batería se gaste. Para más detalles de cómo configurar estos controles y estimar el gasto de batería previo a la simulación de la misma, acúdase a la Guía de usuario, incluida en el Anexo A de esta memoria.

4.2.Ventana de simulación 3D

Se ha incluido esta pequeña ventana 3D en la que se presenta un modelo 3D del CrazyFlie 2.0, que se moverá siguiendo los vectores de posición y actitud en cada uno de los modos de simulación del czSimulator.

Con el ratón, se puede cambiar la perspectiva de la cámara usada en la vista 3D, así como ampliar o alejar la vista. El programa permite también hacer *auto-zoom* de esta ventana para poder volver a la perspectiva original y centrar la imagen en el CrazyFlie 2.0, a través de un botón de *reset*. Adicionalmente, esta ventana representa la trayectoria que el CrazyFlie 2.0 está siguiendo en 3D e incluye unos ejes para que el usuario pueda interpretar la trayectoria analizando esta ventana en conjunción con las cintras de registro o *strip charts*.

En la Figura 11 se muestra una imagen de esta ventana en una típica simulación.



Figura 11 Ventana 3D del czSimulator.

Para más detalles sobre los controles concretos de esta ventana, consultar el Anexo A de esta memoria.

4.3.Indicador de actitud o *Attitude Indicator (AI)*

Este componente sirve para representar la actitud del CrazyFlie 2.0 durante el vuelo. Reflejará la actitud del CrazyFlie 2.0 en cada instante de la simulación, en los modos *Offline o Real*-Time, o del vuelo, en modo *HIL*.

En la Figura 12 se presenta el estado inicial del AI antes de iniciar la simulación con el czSimulator.

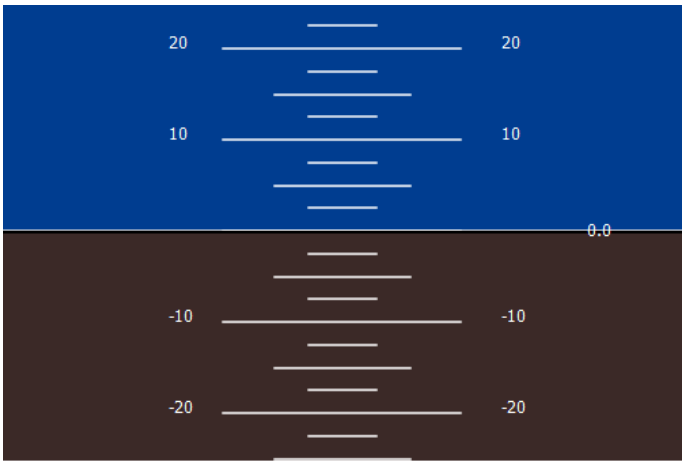


Figura 12 AI incluido en la ventana principal del simulador.

En la siguiente figura, se explica cómo interpretar el AI y poder ver de cada uno de los ángulos que se presentan en este control.

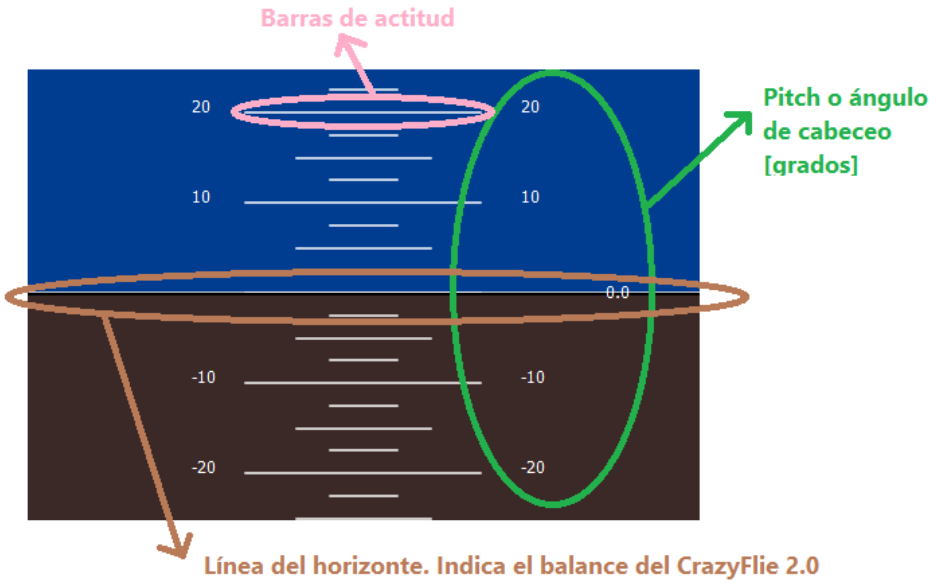


Figura 13 Explicación de los ángulos del AI.

4.4.Gráficas dinámicas *o strip charts*

Son un tipo especial de gráfica que representa de forma dinámica una sola variable o varias. En castellano *strip chart* podría traducirse como una cinta de registro (como las típicas de un electrocardiograma, por ejemplo), pues lo que hace es mostrar una ventana de tiempo centrada en el instante presente, por lo que los datos que se pueden visualizar van cambiando, "moviéndose" hacia la izquierda y dejando de verse los datos más antiguos. Los datos más actuales, por tanto, son los que van "apareciendo" a la derecha de este control.

Las variables representadas en los *strip charts* en el simulador son las siguientes:

- Trayectoria

Se representa la coordenada X en el eje horizontal y la coordenada Y en el eje vertical, ambas en metros.

- Position
  - Coordenada X: Se representa el tiempo en segundos en el eje horizontal y la coordenada X de la posición en metros en el vertical.
  - Coordenada Y: Se representa el tiempo en segundos en el eje horizontal y la coordenada Y en metros en el eje vertical.
  - Coordenada Z o Altura: Se representa el tiempo en segundos en el eje horizontal y la altura o coordenada Z en metros en el eje vertical.
- Velocidad
  - Módulo de la velocidad: Se representa el tiempo en segundos en el eje horizontal y el módulo de la velocidad en metros por segundo en el eje vertical.
  - Coordenada X de la velocidad o variable u: Se representa el tiempo en segundos en el eje horizontal y la componente X de la velocidad en metros por segundo en el eje vertical.
  - Coordenada Y de la velocidad o variable v: Se representa el tiempo en segundos en el eje horizontal y la componente Y de la velocidad en metros por segundo en el eje vertical.
  - Coordenada Z de la velocidad o variable w: Se representa el tiempo en segundos en el eje horizontal y la componente Z de la velocidad en metros por segundo en el eje vertical.
- Ángulos de Euler
  - Ángulo de cabeceo o *theta*: Se representa el tiempo en segundos en el eje horizontal y el ángulo de cabeceo en grados en el eje vertical.
  - Ángulo de balanceo o *Phi*: Se representa el tiempo en segundos en el eje horizontal y el ángulo de balanceo en grados en el eje vertical.
  - Ángulo de guiñada o *Psi:* Se representa el tiempo en segundos en el eje horizontal y el ángulo de guiñada en grados en el eje vertical.
- Errores del controlador
  - Coordenada X: Se representa el tiempo en segundos en el eje horizontal y el error entre la coordenada X de la consigna y la coordenada X de la posición actual, en metros en el eje vertical.

- Coordenada Y: Se representa el tiempo en segundos en el eje horizontal y el error entre la coordenada Y de la consigna y la coordenada Y de la posición actual, en metros en el eje vertical.
- Coordenada Z: Se representa el tiempo en segundos en el eje horizontal y el error entre la coordenada Z de la consigna y la coordenada Z de la posición actual, en metros en el eje vertical.

- Consignas de los controladores:
  - PWM1, PWM2, PWM3 y PWM4: comandas de los controladores *On-board* a los rotores del CrazyFlie 2.0 (actuadores)
  - Ω_r , ϕ_r , θ_r y r_c : comandas de los controladores *Off-board*

En la siguiente figura, se muestra una cinta de registro o *strip chart* como ejemplo de todas las mencionadas anteriormente:

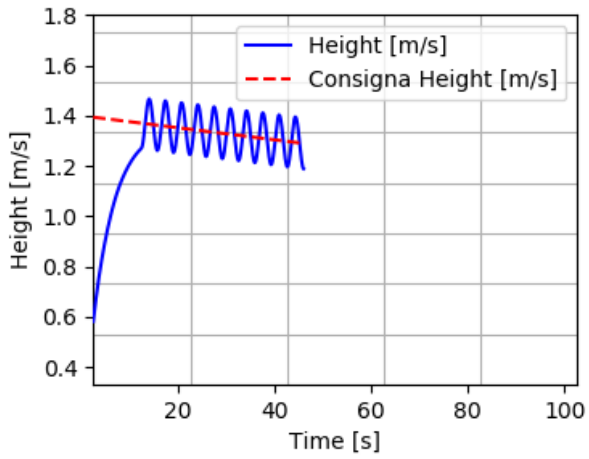


Figura 14 Ejemplo de cinta de registro o *strip chart.*

4.5.Alarmas

Las alarmas son controles visuales que se han incluido en la ventana principal para alertar al usuario cuando el CrazyFlie 2.0 está llegando a algún límite físico de aceleración, giro de los rotores, altura límite para perder comunicación, etc.

Las alarmas incluidas son:

– *Battery low (*Batería baja)

Esta alarma estará en color naranja si el nivel de batería es inferior al 25%, y será de color rojo si el nivel de batería es inferior al 10% del total.

Simulará el gasto de batería normal en los modos *Offline* y *Real-Time* y será el nivel real de batería del CrazyFlie 2.0, en el modo *HIL*, en el que el equipo está conectado al programa.

– *Stall* (Entrada en pérdida)

Representa cuando el CrazyFlie 2.0, en el modo de simulación *Offline* o en el modo HIL, está cerca de entrar en pérdida. En el modo HIL es especialmente importante pues permite

al usuario reaccionar e incrementar la velocidad de giro de los rotores. Si las limitaciones físicas están activadas para ser aplicadas a la simulación, una vez que el CrazyFlie 2.0 entre en pérdida, la simulación se parará.

– *High Altitude* (Altitud demasiado alta)

Representa cuando el CrazyFlie 2.0, en el modo de simulación *Offline* o en el modo HIL, está cerca de encontrarse por encima de un determinado umbral de altitud, con el consiguiente riesgo de pérdida de la señal de radio. En el modo HIL es especialmente importante pues permite al usuario reaccionar y reducirla.

Si las limitaciones físicas están activadas para ser aplicadas a la simulación, una vez que el CrazyFlie 2.0 sobrepase este umbral de altura, la simulación se parará. Este límite de altitud se ha fijado igual a 1km. En la Figura 15 se incluye una representación de las tres alarmas anteriores:



Figura 15 Alarmas de entrada en pérdida, batería baja y altitud demasiado alta.

– 4 alarmas para las velocidades de los rotores

Estas alarmas muestran al usuario cuando los rotores del CrazyFlie 2.0 están girando a una velocidad de rotación baja, normal, media o alta a través de un código de colores. Los rotores están numerados del 1 al 4, tal y como se muestra en la Figura 47.

Los niveles de estas alarmas son, obtenidos de [18], son:

- Velocidad muy baja (azul): por debajo de las 5000 rpm.
- Velocidad normal (verde): entre las 5000 rpm y las 45000 rpm.
- Velocidad media (naranja o ámbar): entre las 45000 rpm y las 65000 rpm.
- Velocidad alta (rojo): superior a 65000 rpm.

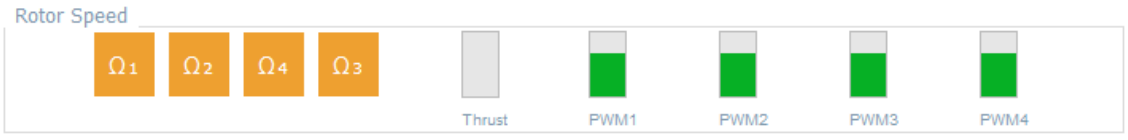


Figura 16 Alarmas para la velocidad de rotación de los rotores.

4.6.Control de tiempo

Este control le permite al usuario poder "acelerar" el tiempo, de manera que pueda simular un vuelo de 30 minutos, por ejemplo, en pocos minutos. Este control sólo estará disponible en el modo de simulación *Offline,* e indirectamente en el modo *Real-Time*, cuando el usuario simule el nuevo controlador.

La aceleración del tiempo se hace de forma que un golpe de reloj en el ordenador corresponda a varias muestras del vector de tiempo del vuelo. De esta forma, una

simulación larga que se haya realizado en Matlab® podrá visualizarse con el simulador mucho más rápidamente. Se puede llegar a acelerar el tiempo hasta cinco veces más rápido, en escalones de uno en uno, como puede verse en la siguiente imagen del control.

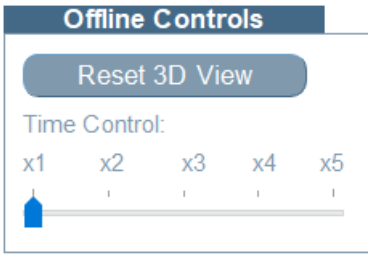


Figura 17 Control de tiempo en el modo de simulación *Offline*.

4.7.Controles de la simulación

Para facilitar el análisis del vuelo en el modo *Offline*, este laboratorio virtual posee unos controles de simulación, como si de un vídeo se tratara. En la siguiente imagen se muestra la configuración de estos controles:

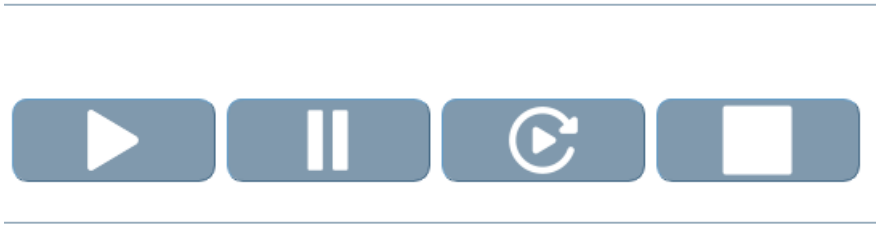


Figura 18 Controles de la simulación en el modo de simulación *Offline*.

De izquierda a derecha, los botones representados en la anterior figura representan:

- *Play*: inicia la presentación del vuelo
- *Pause:* pausa la presentación del vuelo. Si el usuario vuelve a pulsar el botón de *Play*, el programa continuará donde se había pausado
- *Replay*: sirve para volver a ver la simulación desde el instante inicial
- *Stop*: para completamente la simulación

Para más detalles sobre estos controles de puede acudir al Anexo A, donde se recoge la Guía de usuario del laboratorio.

4.8.Control de batería y valor en Voltios

En la siguiente imagen se muestra el control que presenta la batería restante del CrazyFlie 2.0 y su valor en Voltios.

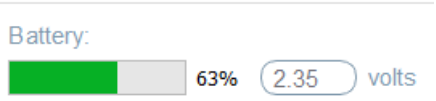


Figura 19 Representación del nivel de batería restante

Para el control visual que muestra el porcentaje restante de batería (parte izquierda de la Figura 19), lo que se ha hecho es asumir que el 100% de la batería tendría una vida útil de siete minutos, como se especifica en [28]. Además, se ha asumido que el voltaje nominal de la batería es de 3.7V (información obtenida de [7]). Con estos datos, se puede estimar el nivel restante de batería, a partir del tiempo restante calculado con la ecuación (1.6).

En el modo *Offline*, para calcular el porcentaje de empuje se necesita conocer el empuje real, para lo que se necesita que la señal F_z del programa tenga asociada alguna variable del fichero de simulación de Matlab® leído. Si éste no es el caso, se reflejará en el fichero de *log* del programa que no se ha podido calcular el nivel de la batería, y por tanto se deshabilitarán todos los controles visuales relacionados (barra de progreso de la batería y alarma de batería baja).

4.9.Fichero de *settings* del programa

Una característica importante del laboratorio, y diferenciable con el resto de laboratorios virtuales existentes en el mercado, es que permite al usuario cambiar y personalizar el diseño de la ventana principal del programa. El usuario puede decidir qué cintas de registro quiere ver (si sólo la posición, o posición y trayectoria, etc.) con un volumen elevado de combinaciones. Además, puede elegir ver o no las pestañas con los valores de los principales parámetros del vuelo.

En la ventana de preferencias se puede seleccionar además la aplicación o no de distintas limitaciones físicas al vuelo o la aplicación de limitaciones a los controladores, así como distintas características del modo de simulación *Real-Time*.

Cada vez que se cierre el programa, czSimulator generará un nuevo fichero de *settings*, donde quedarán guardadas todas estas preferencias configuradas por el usuario, de forma que la próxima vez que se lance el laboratorio, todo quede configurado como haya decidido el usuario. Para más información acerca de todas las opciones y preferencias disponibles, consúltese la Guía de usuario, incluida en el Anexo A de esta memoria.

4.10. Fichero de *log* o de registro de errores del programa

El programa incluye la generación de un fichero de *log* o de registro en el que se guardan todas las operaciones importantes llevadas a cabo por el programa y por el usuario. Si algo va mal, este fichero puede utilizarse para saber qué operación ha fallado, o si algo no se ha ejecutado como se esperaba. Este fichero se genera automáticamente cada vez que el usuario ejecuta el programa y será guardado por defecto, en la carpeta que incluya la distribución del programa.

Capítulo 5 Simulador Interactivo: Controladores

En este capítulo se detallan las arquitecturas de control incluidas en el simulador como controladores pre determinados.

5. Simulador Interactivo: Controladores

En este capítulo se explican en detalle las estructuras de control implementadas como controladores por defecto, para ser usadas en el modo de simulación *Real-Time.* El motivo de incluir estos controladores en este modo de simulación ha sido facilitar el acceso de estudiantes de ingeniería de control a estructuras de control muy conocidas, perfectamente válidas para el control de un dron como el CrazyFlie 2.0 real.

La estructura de control en el CrazyFlie se muestra en la Figura 20.

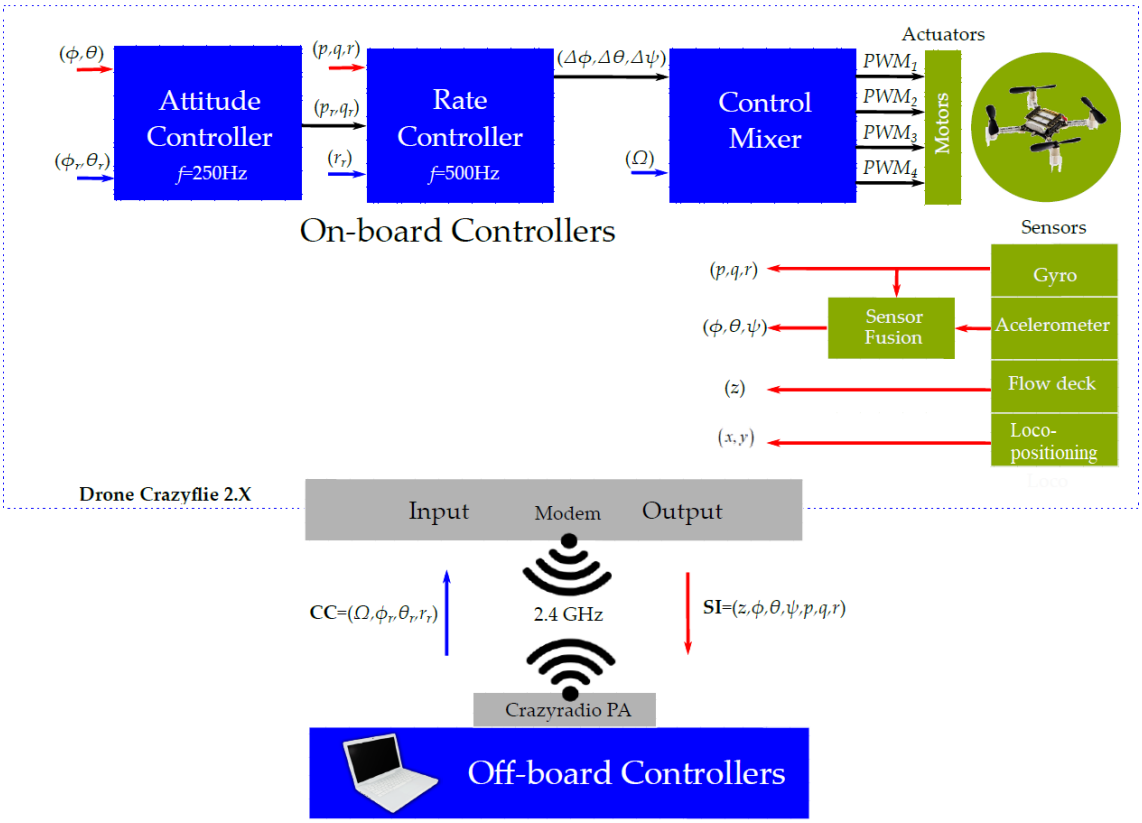


Figura 20 Arquitectura del control en el hardware CrazyFlie 2.0.

El *firmware* del CrazyFlie 2.0 está dividido en controladores *On-board* y controladores *Off-board*. El primer grupo contiene una serie de controladores internos al CrazyFlie 2.0, capaces de estabilizar el sistema a partir de $(\Omega, \phi_r, \theta_r, r_r)$. Esto se consigue gracias a tres controladores en cascada, como puede verse en la Figura 20. Se trata de un controlador de actitud, un controlador de las velocidades angulares y un *Control mixer* para obtener las señales PWM a ser comandadas a cada motor del CrazyFlie 2.0. Externos al sistema, se encuentran los controladores *Off-board*. Estos controladores son los que vamos a modificar desde el simulador, y los que se interpretan en el modo de simulación *Real-Time*. En el caso de los controladores por defecto incluidos en el programa, y presentados en detalle en las siguientes secciones, los controladores *Off*-board se componen de tres controladores PID, uno para la posición X e Y, otro para el control de la fuerza de empuje y otro para el control en guiñada, ψ .

5.1. Controlador PID periódico

El controlador de tipo PID (Proporcional-Integral-Derivativo) periódico tiene, típicamente, la estructura que se muestra en la Figura 21.

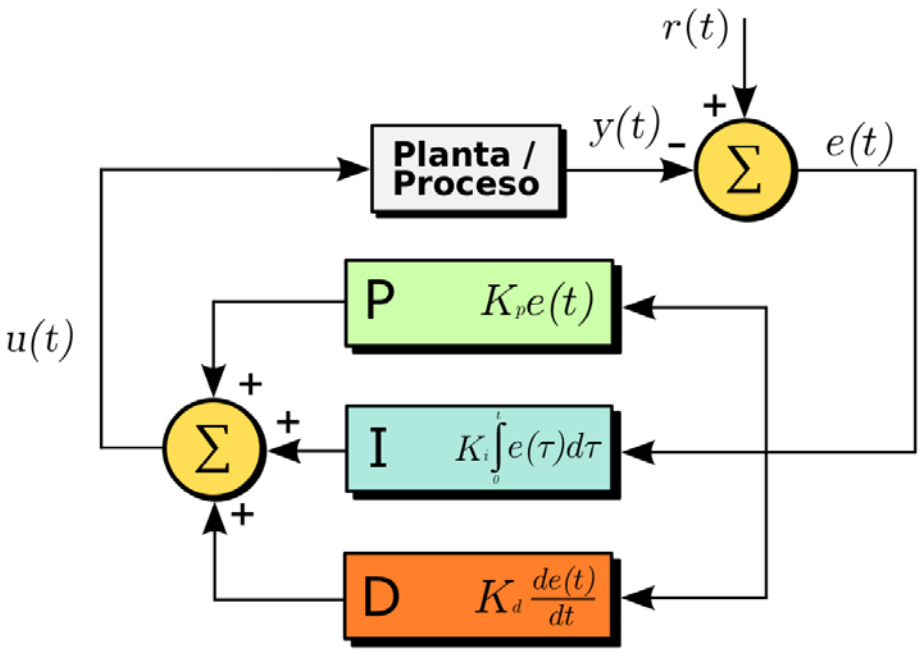


Figura 21 Estructura típica de un controlador PID.

Se compone de tres bloques principales

- Bloque proporcional: Hace que, con el tiempo, la señal de error se aproxime a cero.
- Bloque integral: Su función es corregir el error en estado estacionario, provocado por aquellas perturbaciones exteriores al sistema que el control proporcional no es capaz de compensar.
- Bloque derivativo: Ayuda a disminuir el rebasamiento u *overshooting* de la señal al inicio del proceso de control y disminuye el error cuando éste cambia de signo.

En suma, la salida del controlador PID periódico puede definirse como:

$$y(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) \cdot d\tau + K_d \cdot \frac{de}{dt} \qquad (1.10)$$

Cada una de las ganancias K_p, K_i y K_d son las que se podrán modificar en el programa en el modo de simulación *Real-Time*.

El PID periódico integrado en el programa habilita al usuario a poder simular distintas trayectorias y analizar cómo el CrazyFlie 2.0 sigue cada una de ellas, en función de las ganancias configuradas para este controlador PID.

Para más información sobre los PID periódicos puede acudirse, entre otras referencias, a [29], de donde se obtuvo la idea de usar la metáfora de la hoja de cálculo y, por tanto, poder interaccionar rápidamente con los controladores e implementar el modo *Real-Time*.

En [30], se describen ciertas técnicas de diseño de controladores PID. Más específicamente para CrazyFlie 2.0, puede encontrarse información sobre los controladores PID periódicos implementados como controladores *On-board* en [31].

5.2. Controlador PID basado en eventos

En la Figura 22 se muestra la estructura general de un controlador basado en eventos [34]-[36]. En el caso de un controlador PID basado en eventos, su estructura suele ser como la que se muestra en la Figura 23, esto es, se compone de un controlador PID en el que se ha incluido un detector de eventos. Este detector de eventos será quien decida cuándo se aplicará la acción de control. Para ello, estará constantemente comprobando si un determinado evento ha ocurrido o no. Si la respuesta es afirmativa, se activará el control, y el controlador PID actuará para estabilizar el sistema. Si, por el contrario, la respuesta es negativa, el sistema quedará en lazo abierto. La lógica de este detector de eventos se muestra también en la Figura 23. La definición del evento puede estar considerada en base a diferentes variables del sistema. En el caso de los controladores PID diseñados para el simulador, un evento se define como la condición por la cual el error (diferencia entre la salida del sistema y la señal de referencia) supera un cierto umbral.

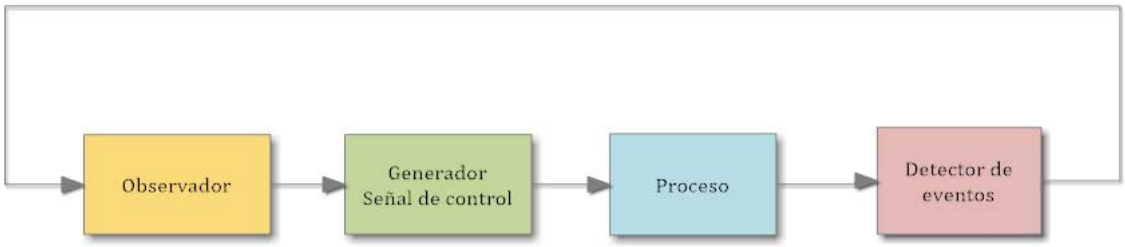


Figura 22 Estructura de un controlador general basado en eventos.

En la Figura 23 se muestra un poco más en detalle la estructura de un controlador PID basando en eventos.

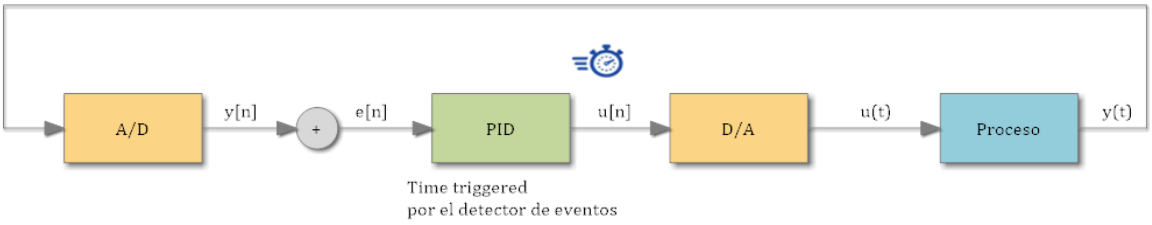


Figura 23 Estructura de un controlador PID basado en eventos.

En la Figura 23, el bloque "A/D" representa la conversión de señales analógicas a digitales, y el bloque "D/A" representa la conversión de señales digitales a analógicas. Esto es, se pasa de señales continuas en el tiempo a señales muestreadas, o viceversa. A continuación, en la Figura 24 se presenta en detalle la estructura del controlador PID con el detector de eventos.

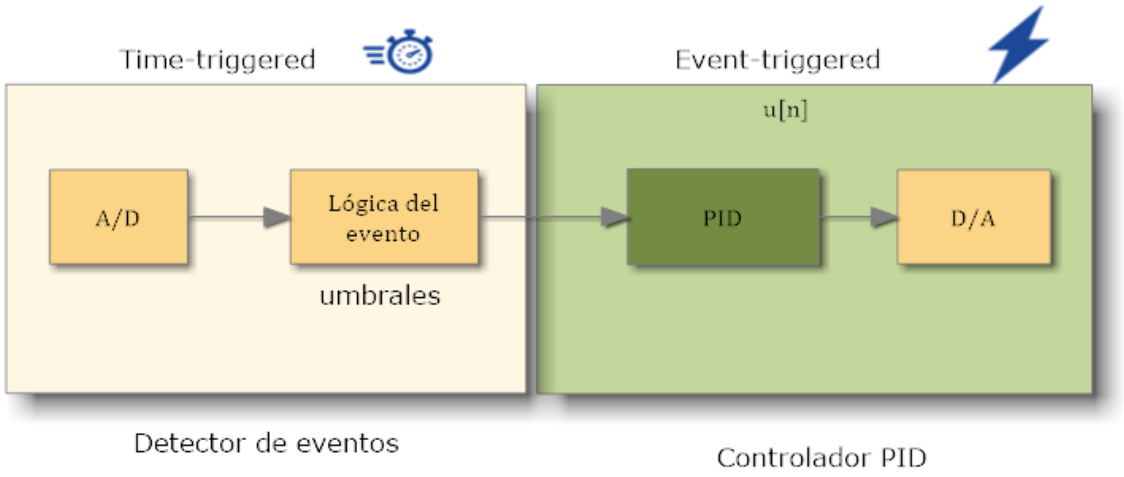


Figura 24 Lógica del detector de eventos de un controlador PID basado en eventos.

Una gran ventaja del simulador presentado es que el usuario puede jugar con estos valores umbrales para cada uno de los PID en el modo de simulación *Real*-Time y comprobar si se producen ciclos límite en diferentes variables de vuelo.

Una dificultad de los controladores PID basados en eventos es que pueden aparecer ciclos límites, que son oscilaciones autosostenidas del sistema. Los ciclos límite ocurren cuando el sistema tiene como solución en el plano de fase una trayectoria cerrada, a la que otras trayectorias no cerradas tienden en espiral. Para comprobar la existencia de ciclos límites, se debe tener en cuenta el armónico de orden superior de la función de transferencia del sistema, como se detalla en [32]. En otras ocasiones, ciertas estrategias de análisis del control, como el muestreo por cruce, pueden llevar a tener ciclos límites en el sistema [33].

Otra causa de la aparición de los ciclos límites es la elección de los valores umbral, que hacen que el detector de eventos active o no el control. Típicamente, estos valores son constantes y deben elegirse con especial cuidado. En [33] se revisan diferentes metodologías para la generación de eventos, y en [39] se detalla una estrategia para optimizar los umbrales de activación del evento. Para analizar la aparición de ciclos límites pueden utilizarse métodos estocásticos y realizar numerosas simulaciones para estudiar el comportamiento del sistema [34].

A pesar de esta dificultad, los controladores PID basados en eventos tienen una ventaja, como se describe en [32], y es que alargan la vida del actuador y reducen el gasto de energía debido al control, pues éste sólo es aplicado en ciertos intervalos de tiempo. La amplitud del ciclo límite depende de las condiciones iniciales del sistema Para más información sobre los ciclos límites puede acudirse a [31] para un resumen conceptual y a [29] y [30] para un estudio y aplicación más detallado.

Capítulo 6
Características Adicionales del Simulador

En este capítulo se detallan todas las características principales del laboratorio virtual desarrollado y sus elementos diferenciadores.

6. Características adicionales del simulador

6.1. Escalabilidad

El código del programa utiliza para su implementación el patrón de diseño MVC o Modelo Vista Controlador. Según este patrón, la implementación de las clases se divide en tres capas, como las que se muestran en el siguiente diagrama:

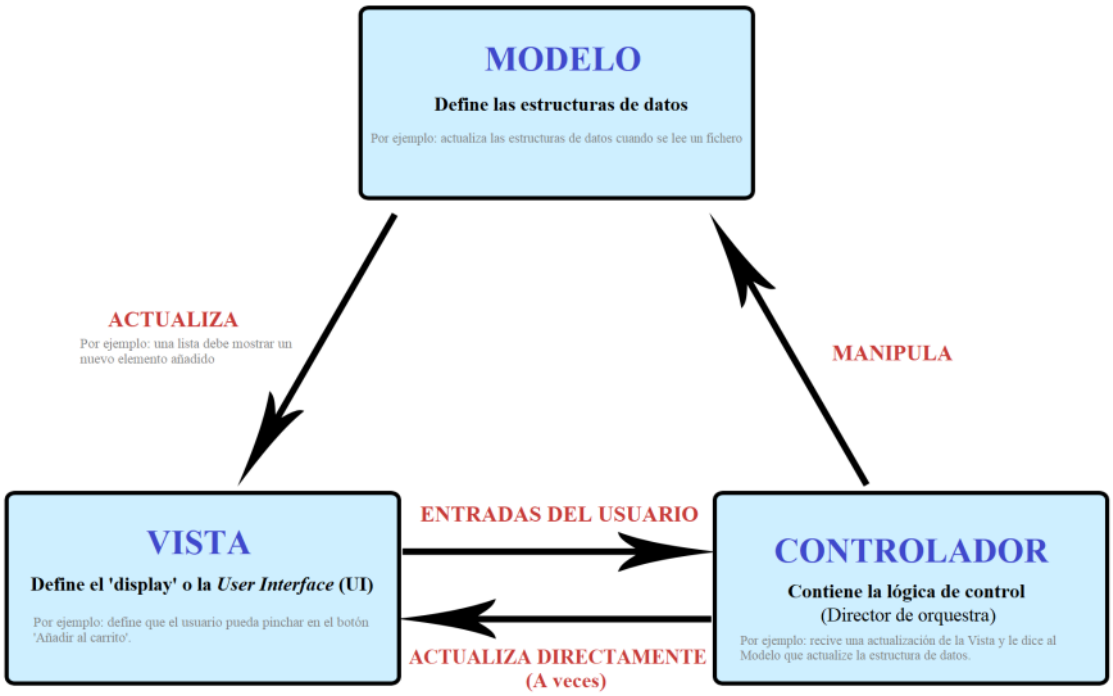


Figura 25 Diagrama del patrón de diseño Modelo Vista Controlador.

A continuación se explica muy brevemente cada una de las capas:

- Modelo: Define y controla todas las estructuras de datos, manipulación de ficheros (lectura y guardado), etc.
- Vista: Controla la interacción con el usuario, recibe información tanto del Modelo como del Controlador y puede enviar información al Controlador.
- Controlador: Ejerce el papel de "director de la orquesta". Controla cómo se comporta la Vista y, comanda todas las acciones del Modelo para actualizar las estructuras de datos. Es el cerebro del programa.

En el laboratorio virtual czSimulator, este patrón de diseño se traduce en tres carpetas distintas en las que se estructura el código fuente. En cada una de ellas se encuentra la clase principal: czView, czModel o czController, que hace el papel de cada una de las capas explicadas y las coordina. El hecho de haber aplicado este patrón al código hace que sea muy fácilmente entendible por cualquier investigador con una experiencia media en programación, de manera que se pueda escalar a otras plataformas, otros controladores por defecto u otros motores de simulación distintos de Matlab® de una forma muy sencilla.

Además, todo el código está comentado y el programa incluye una Guía de usuario del programador, accesible para aquellos usuarios avanzados que posean la contraseña requerida. Para terminar, añadir que el lenguaje utilizado para el desarrollo es *Python*, lenguaje cada vez más utilizado y extendido en la comunidad científica, e ideal para desarrollar y reutilizar librerías especializadas de simulación, como las que se han utilizado y desarrollado para el CrazyFlie 2.0.

6.2. Seguridad y protección de datos

Este simulador interactivo cuenta con varias capas de protección. La primera de ellas se encuentra en el código fuente, el cual no se proporciona con la distribución del programa. En su lugar, se proporciona un fichero ejecutable y una carpeta de dependencias. Esta carpeta de 'dependencias' contiene, tanto el código fuente del programa (encriptado), como todas las librerías de *Python* necesarias. Esto hace que el programa sea auto-contenido y el usuario no necesite instalar ningún tipo de programa adicional.

En la siguiente figura se muestra la estructura de distribución del programa:

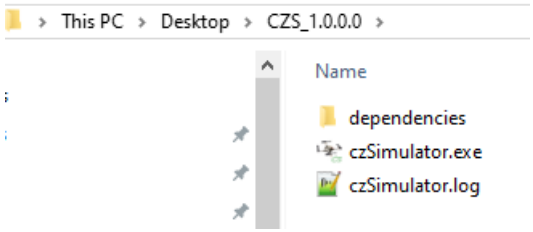


Figura 26 Estructura de la distribución del programa.

Otra capa de seguridad se incluye en el propio programa, en el modo *Real-Time*. En este modo el usuario puede leer un fichero de control MDL de Matlab®, pero el código de este fichero no se muestra en ningún momento, ni se guarda en el código del programa. El fichero sólo es leído e interpretado, borrando todo dato del mismo una vez que el usuario lo guarda con las nuevas ganancias de los controladores y lanza la simulación en el programa.

6.3. Operabilidad

El simulador requiere Windows 7, 8.1 o 10. No necesita ningún otro *software*. El paquete en el que se distribuye el ejecutable contiene auto-contenidas todas las librerías de *Python* necesarias para poder ejecutar el programa. Además, para el futuro desarrollador, se han generado ficheros que generan de forma automática todos los *scripts* de los diálogos y ventanas cada vez que cambian, así como de los recursos utilizados (imágenes, iconos, etc.).

Se ha desarrollado también un *script* que genera la carpeta de distribución de forma automática, y un último *script* que permite instalar el mismo entorno de compilación de *Python*, eliminando así el infierno de obtener las mimas versiones de librerías usadas para esta versión de programa. Todo esto hace que la operabilidad del programa sea muy alta, permitiendo a nuevos desarrolladores ampliar el laboratorio virtual a otras plataformas, o con más capacidades de simulación, de una forma sencilla y muy directa.

6.4. Ventajas con respecto a otros simuladores del mercado

Una de las principales ventajas del laboratorio virtual presentado es que permite al usuario realizar diferentes tipos de simulaciones de manera muy rápida, con una única herramienta, y con muy pocos *clics*. Además es muy intuitivo y con una interfaz de usuario muy sencilla de aprender. En la Tabla 1 se presenta una comparación entre diversas funcionalidades de czSimulator, el simulador desarrollado para este proyecto, y dos de los simuladores más conocidos para CrazyFlie 2.0.

Específicamente para el simulador *crazyS* se ha encontrado muy poca información, aunque se ha decidido incluirlo en la siguiente tabla, pues es de los pocos simuladores especializados en CrazyFlie 2.0.

Característica	4FLY	CrazyS	czSimulator
Interfaz gráfica	✗	✗	✓
Interacción con OpenGL	✓	✓	✓
Modo Manual	✓	✗	✓
Representación gráfica de las variables del vuelo	✓	✗	✓
Posibilidad de interaccionar con las cintas de registro	✗	✗	✓
Configuración de la Ventana Principal	✗	✗	✓
Guardado de *settings* del usuario	✗	✗	✓
Modo de simulación *Offline*	✓	✗	✓
Modo de simulación *Real-Time*	✗	✗	✓
Modo de simulación HIL	✗	✗	✓
Módulo de Planificación de Misión	✗	✗	✓
Programa auto-incluido	✗	✗	✓
Capacidad para modificar el modelo dinámico del CrazyFlie 2.0	✗	✓	✓

Tabla 3 Tabla comparativa de dos simuladores comerciales con el entorno desarrollado (czSimulator).

Como puede comprobarse, el número de capacidades de las que dispone el laboratorio virtual desarrollado, comparado con el número de capacidades de los "competidores" es bastante superior. czSimulator permite mucha más interacción y personalización por parte del usuario.

Capítulo 7
Simulaciones y Resultados de la Experimentación

En este capítulo se incluyen diferentes simulaciones realizadas con el laboratorio virtual para familiarizar al usuario con los resultados que puede esperar del simulador.

7. Simulaciones y Resultados de la Experimentación

En esta sección se incluyen diversas simulaciones llevadas a cabo con el laboratorio virtual.

7.1. Simulación *Offline*

En este apartado se incluye un ejemplo de una simulación *Offline* realizada con el programa. El controlador usado para obtener el fichero de Matlab® es un PID y el modelo usado del CrazyFlie 2.0 es el modelo no lineal detallado en el Anexo C de esta memoria. Una vez leído el fichero y asociados los nombres de las variables, el aspecto de la ventana es el que se muestra en la siguiente figura:

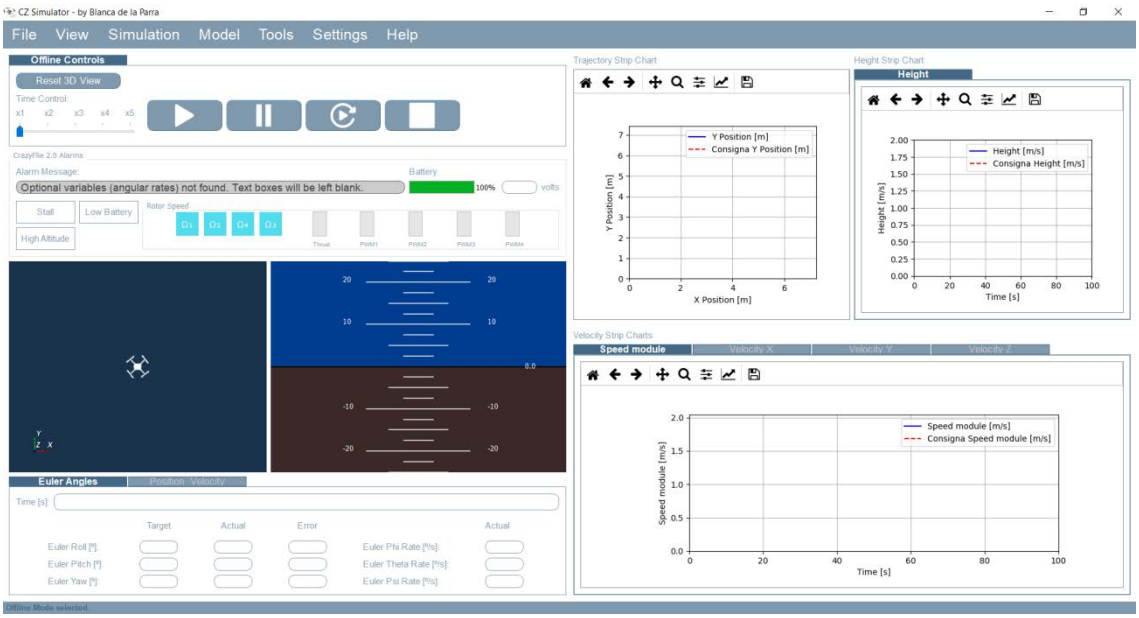


Figura 27 Ventana principal del czSimulator antes de iniciar una simulación *Offline*.

Una vez iniciada la simulación, se pueden ir observando las distintas variables. Por ejemplo, en la Figura 28 se muestra la trayectoria que está realizando el CrazyFlie 2.0.

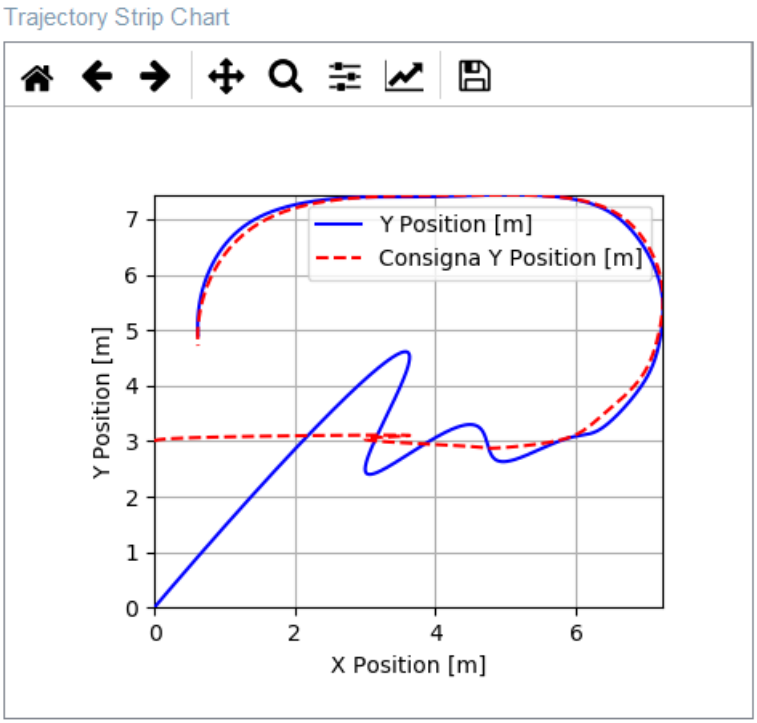


Figura 28 Trayectoria realizada por el CrazyFlie 2.0 en la simulación.

Como otro ejemplo, en las siguientes dos imágenes, Figura 29 y Figura 30, se pueden ver el módulo de la velocidad y la componente vertical del vector velocidad.

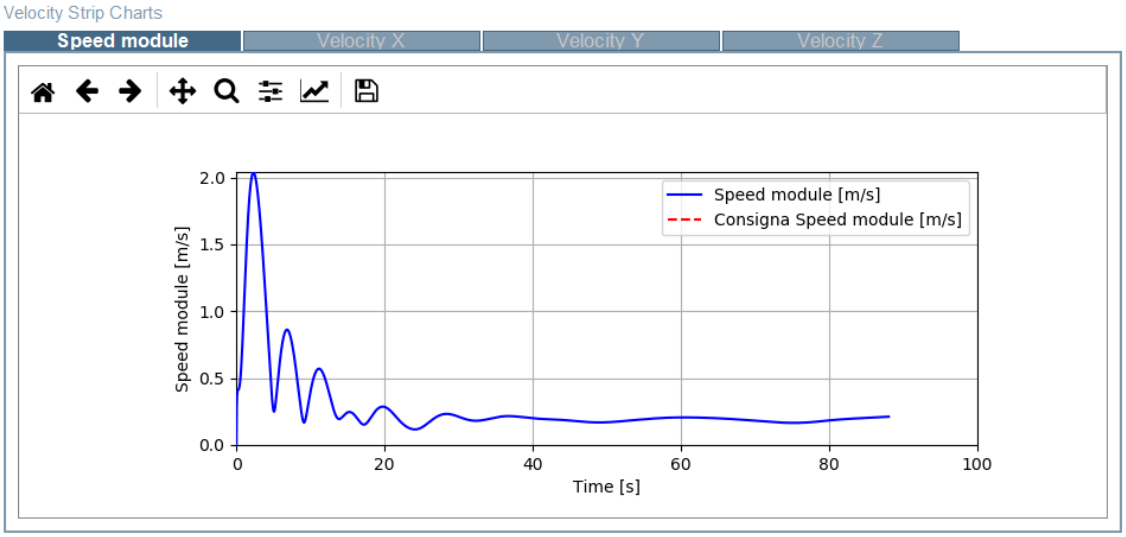


Figura 29 Módulo de la velocidad en la simulación.

Puede observarse en la Figura 29 que el módulo de la velocidad primero crece mucho, de 0.0m/s a 2.0m/s y después va disminuyendo hasta que se "estabiliza" más o menos a 0.25m/s. Este comportamiento cuadra con lo que podemos ver en la Figura 28, en la que inicialmente el CrazyFlie está tratando de alcanzar la trayectoria de referencia, y una vez lo consigue, va disminuyendo la velocidad, va requiriendo menos empuje y se va estabilizando.

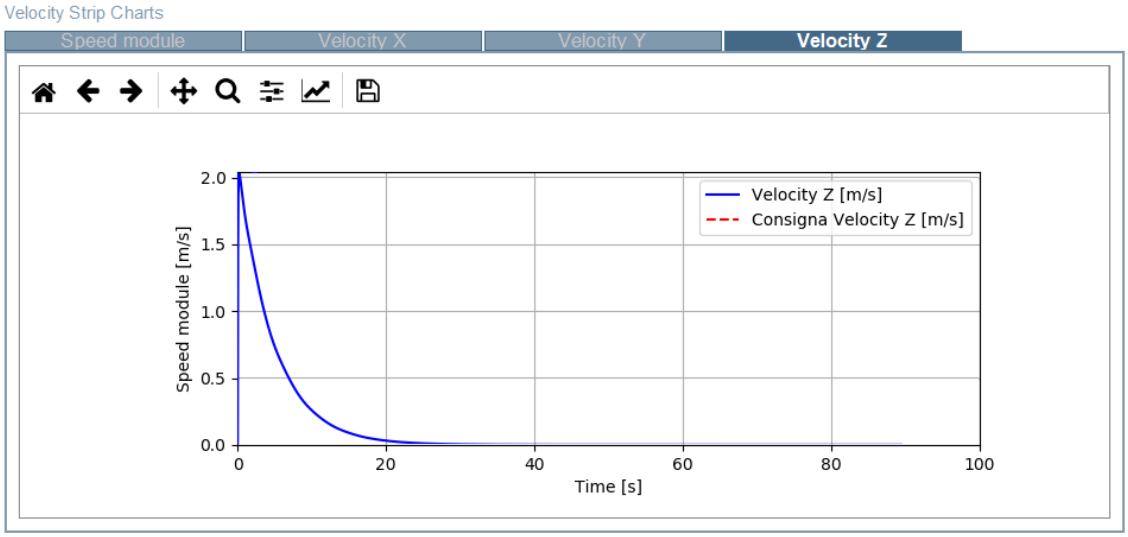


Figura 30 Componente vertical del vector velocidad en la simulación.

Adicionalmente, podemos observar la evolución de la altura en la siguiente figura:

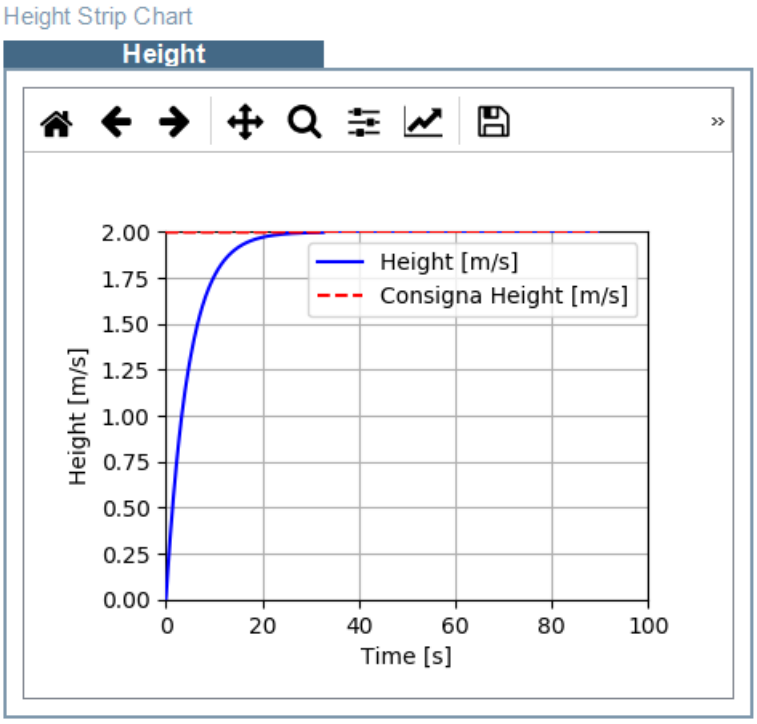


Figura 31 Evolución de la altura en la simulación.

Se ve que la altura aumenta de 0 metros a 2 metros en 20 segundos y después se mantiene constante.

Si ahora analizamos cuál es la trayectoria que se ha diseñado en el Planificador de Misión o *Mission Planner* podemos confirmar que, efectivamente, es una trayectoria que se realiza a una altura constante de 2 metros.

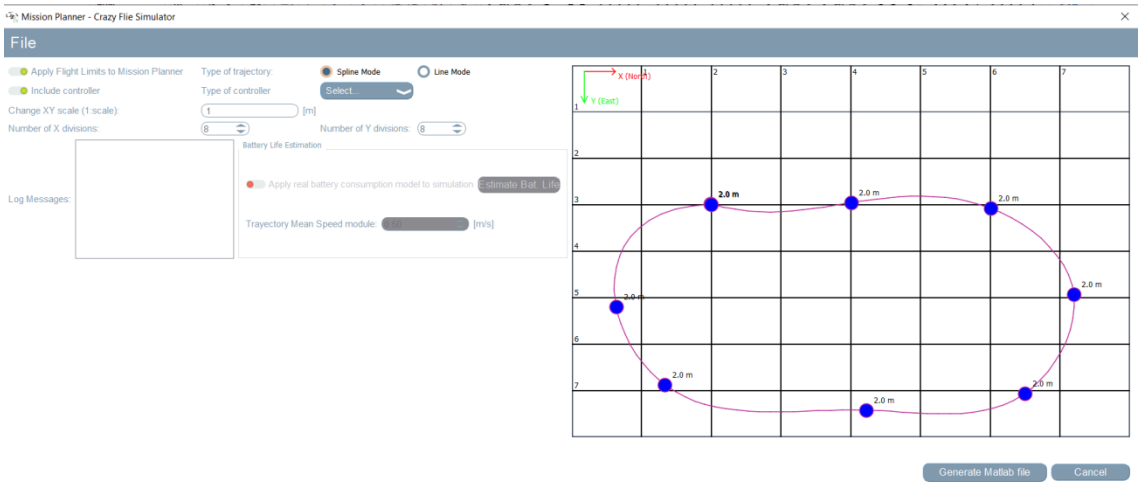


Figura 32 Planificador de misión con la trayectoria que se ha realizado en la simulación.

Como puede comprobarse, este simulador virtual sirve para poder analizar en detalle qué es lo que el CrazyFlie 2.0 está haciendo en la simulación, poder analizar cada una de las variables más importantes en el vuelo y tomar acciones a la hora de rediseñar el controlador, basándose en el comportamiento observado del CrazyFlie 2.0.

7.2. Segunda Simulación *Offline*

En esta otra simulación se ha decidido diseñar una trayectoria en la que la altura no es constante, si no que va cambiando de un punto de control a otro. En la Figura 33 pueden verse cada uno de los puntos de control seleccionados y la altura de cada uno de ellos.

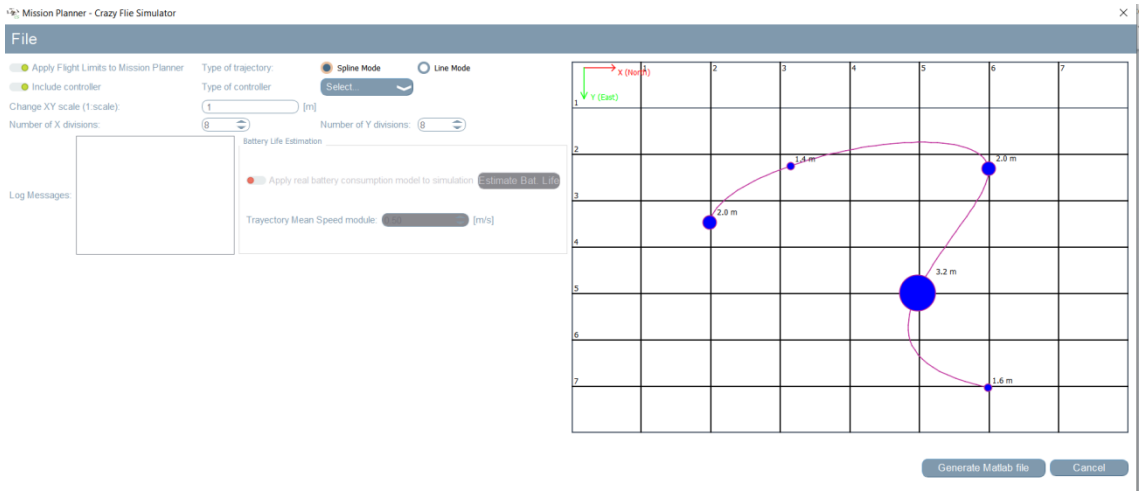


Figura 33 Planificador de misión con la trayectoria que se ha realizado en la segunda simulación.

Una vez iniciada la simulación, se pueden ir observando las distintas variables. A continuación se van a presentar las distintas cintras de registro de esta simulación. Por ejemplo, en la Figura 34 se muestra la trayectoria que está realizando el CrazyFlie 2.0.

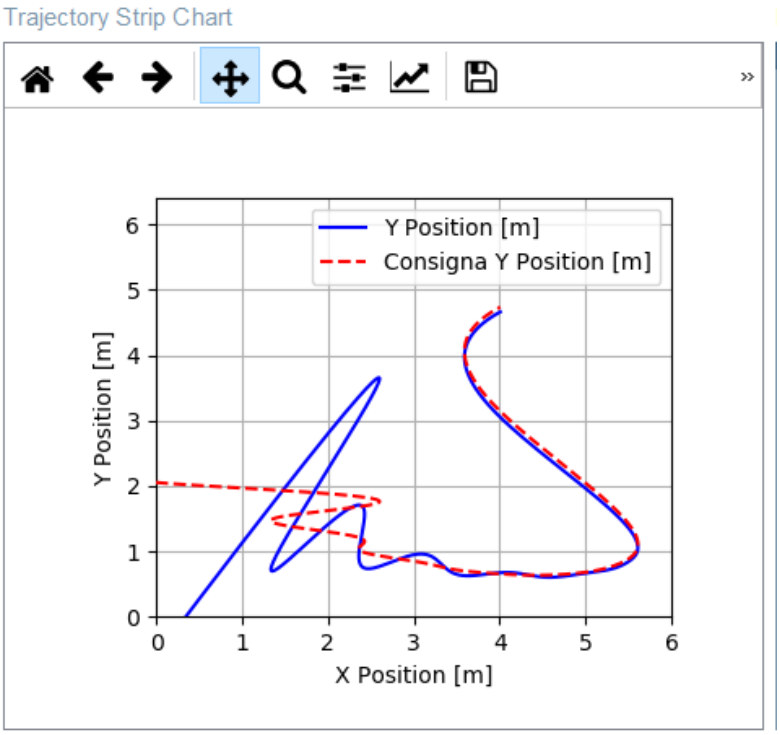


Figura 34 Trayectoria realizada por el CrazyFlie 2.0 en la simulación.

Como otro ejemplo, en las siguientes dos imágenes, Figura 35 y Figura 36 se pueden ver el módulo de la velocidad y la componente vertical del vector velocidad:

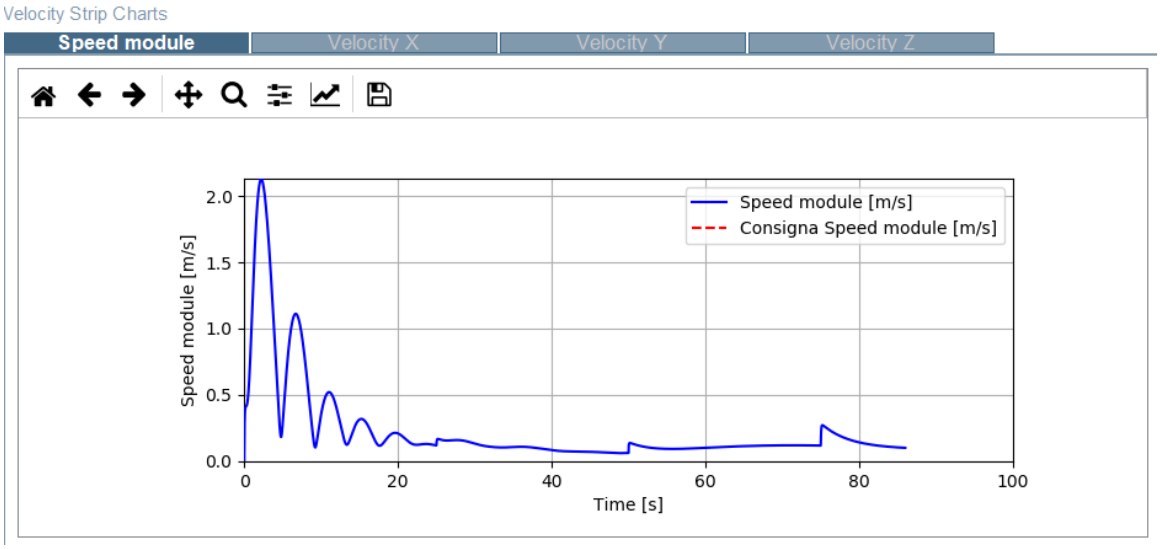


Figura 35 Módulo de la velocidad en la simulación.

Puede observarse en la Figura 35 que el módulo de la velocidad primero crece mucho, de 0.0m/s a 2.0m/s y después va disminuyendo hasta que se "estabiliza" más o menos a 0.25m/s. Después, si la altura tiene que crecer, el módulo de la velocidad también aumenta, como puede comprobarse en la Figura 37 alrededor del instante 50s.

Este comportamiento cuadra con lo que podemos ver en la Figura 34, en la que inicialmente el CrazyFlie está tratando de alcanzar la trayectoria de referencia, y una vez lo consigue, va disminuyendo la velocidad, va requiriendo menos empuje y se va estabilizando.

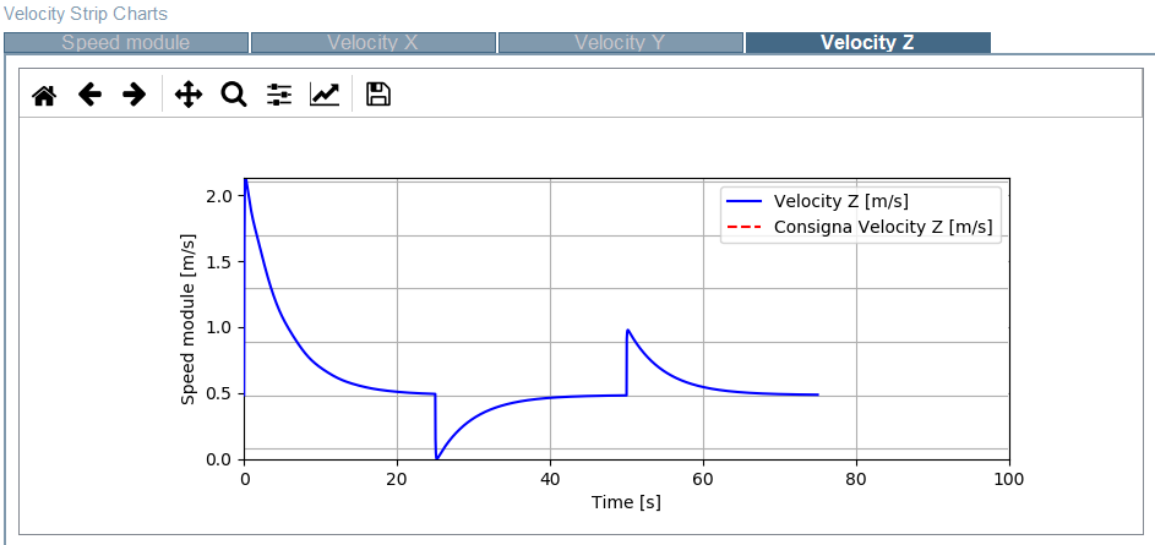


Figura 36 Componente vertical del vector velocidad en la simulación.

Adicionalmente, podemos observar la evolución de la altura en la Figura 37:

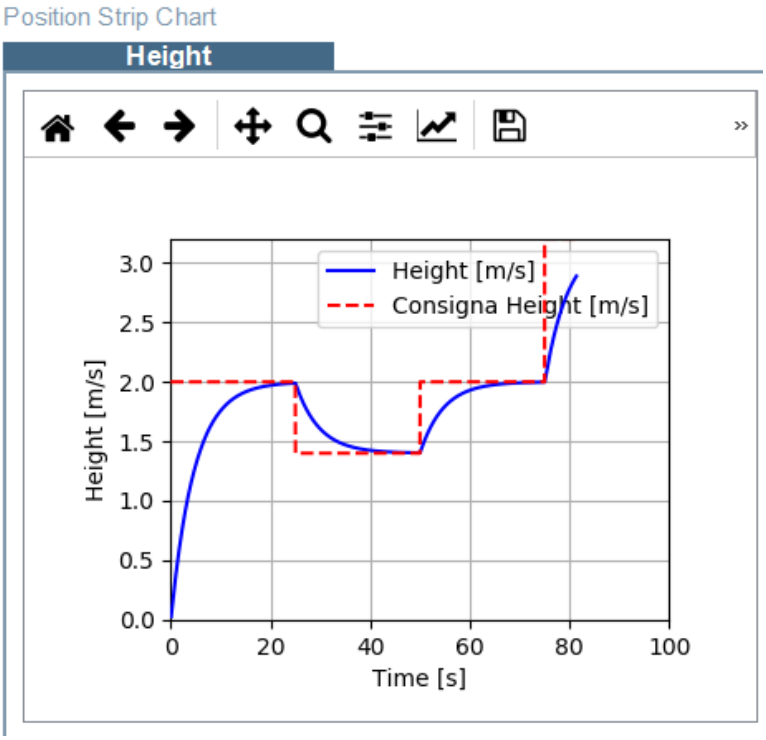


Figura 37 Evolución de la altura en la simulación.

Se ve que la altura aumenta de 0 metros a 2 metros en poco más de 20 segundos, después disminuye a 1.4 metros (altura del segundo punto de control en la trayectoria), vuelve a

aumentar a 2 metros y después vuelve a aumentar hasta 3.2 metros, siguiendo lo que se ha diseñado en el Planificador de misión (Figura 34).

Finalmente, en la Figura 38, puede observarse la evolución de la coordenada X con el tiempo.

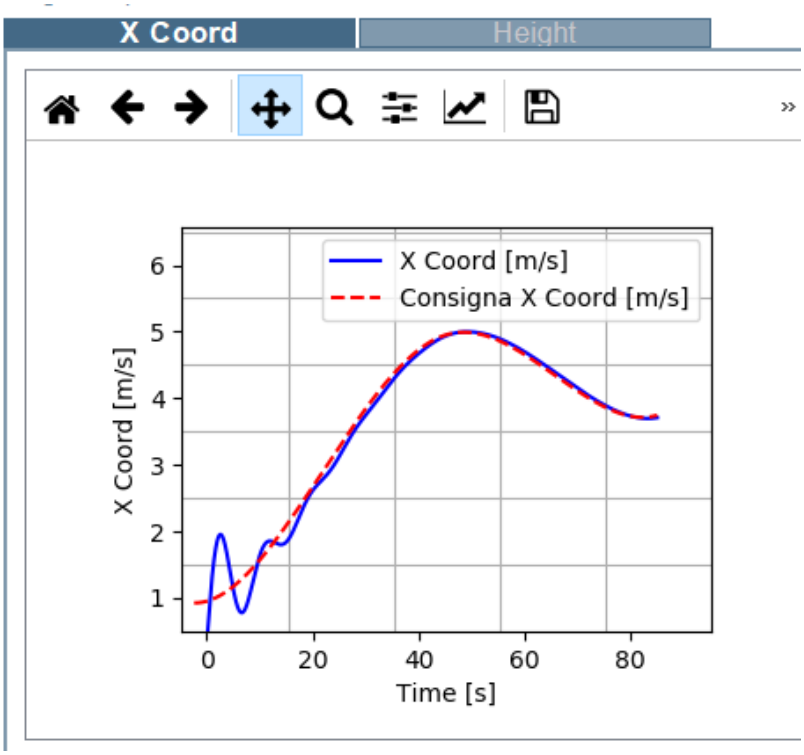


Figura 38 Evolución de la coordenada X con el tiempo.

Tanto en la Figura 34, como en la Figura 38 se ve que hay un pequeño *overshoot* al principio de la simulación, pero después ya el controlador "captura" la trayectoria de referencia y el error es muy pequeño, tanto en la trayectoria seguida como en la coordenada X del CrazyFlie 2.0.

Para analizar este error, y su evolución con el tiempo, no se tiene más que añadir la cinta de registro para el 'Error en X', obteniendo la curva incluida en la siguiente figura.

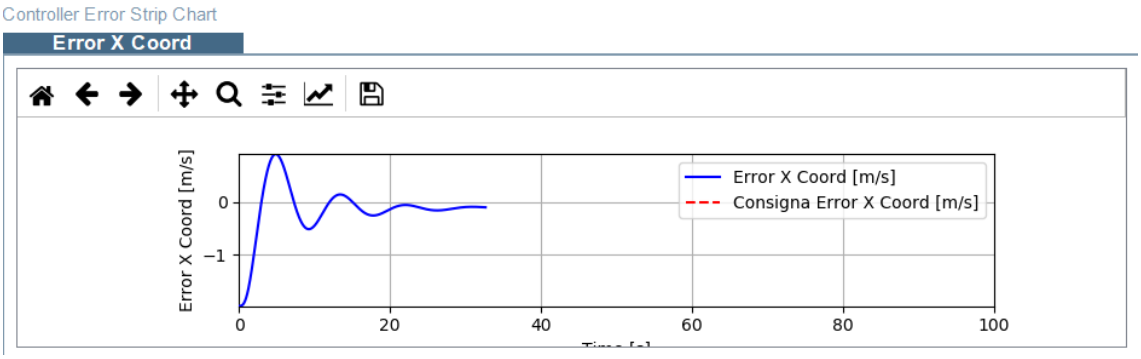


Figura 39 Error en la coordenada X del sistema, con respecto a la trayectoria deseada o consigna.

Puede verse que el error oscila entre 1 y -1 al principio pero el sistema rápidamente se estabiliza, tendiendo el error a cero.

7.3. Simulación *Real-Time*

En esta sección se incluyen los resultados obtenidos de una simulación *Real-Time*, en la que se partió del controlador y los resultados obtenidos en la primera simulación *Offline* incluida en este capítulo, y se modificaron los umbrales de los controladores para obtener ciclos límite. Por ejemplo, en la Figura 40 se muestra la evolución de la altura con el tiempo. Se comprueba que existe ciclo límite pues la altura oscila alrededor de la altura de consigna (2 metros), con una amplitud de 2 metros.

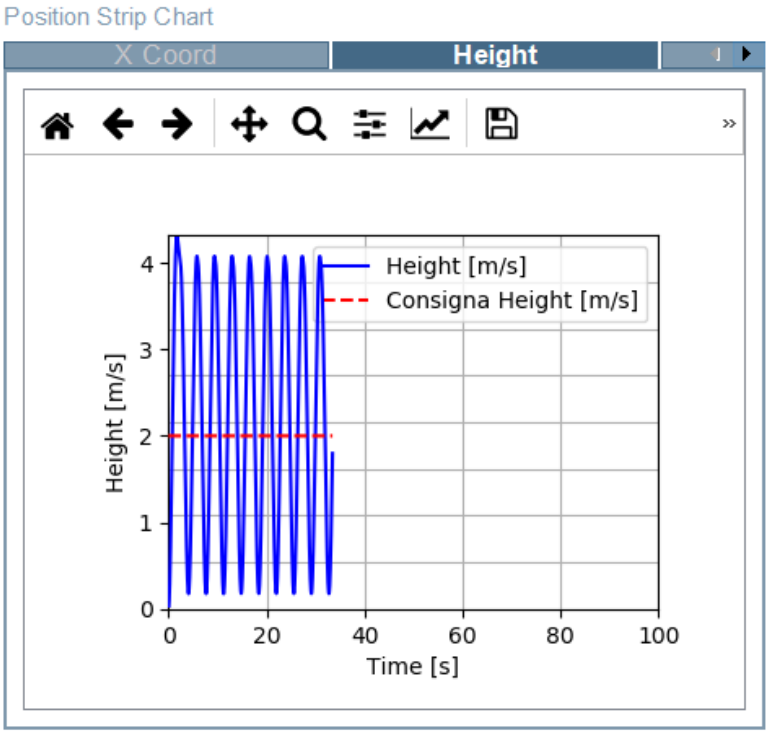


Figura 40 Evolución de la altura en la simulación.

Como otro ejemplo, en la Figura 41 se presenta el módulo de la velocidad:

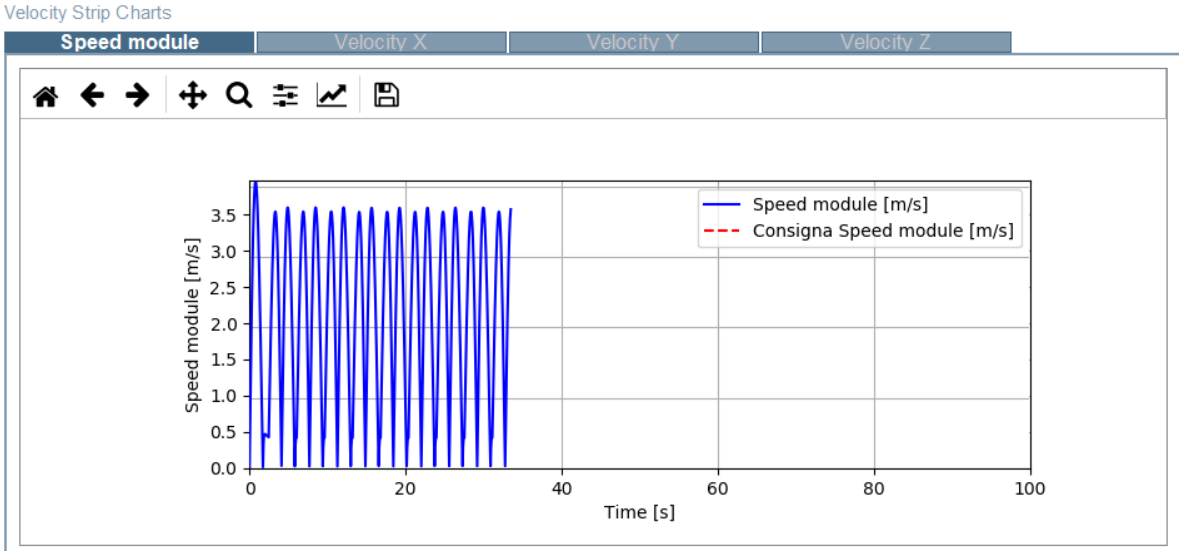


Figura 41 Módulo de la velocidad en la simulación.

Puede observarse que también oscila con una amplitud de 3.5 m/s. Estas oscilaciones son tan grandes que el sistema se inestabiliza y no se ha conseguido poder representar la trayectoria.

Para poder visualizar mejor los efectos del ciclo límite, se presenta una última simulación en la sección siguiente.

7.4. Segunda Simulación *Real-Time*

Se ha realizado una segunda simulación *Real-Time*, siguiendo la misma trayectoria que se presentó en la Figura 42 pero cambiando los valores umbral para el controlador en altura, para que sea distinto de cero y se produzca un ciclo límite pequeño, que no llegue a desestabilizar el sistema. Por ejemplo, en la Figura 42se muestra la trayectoria, para confirmar que el sistema consigue realizar la trayectoria.

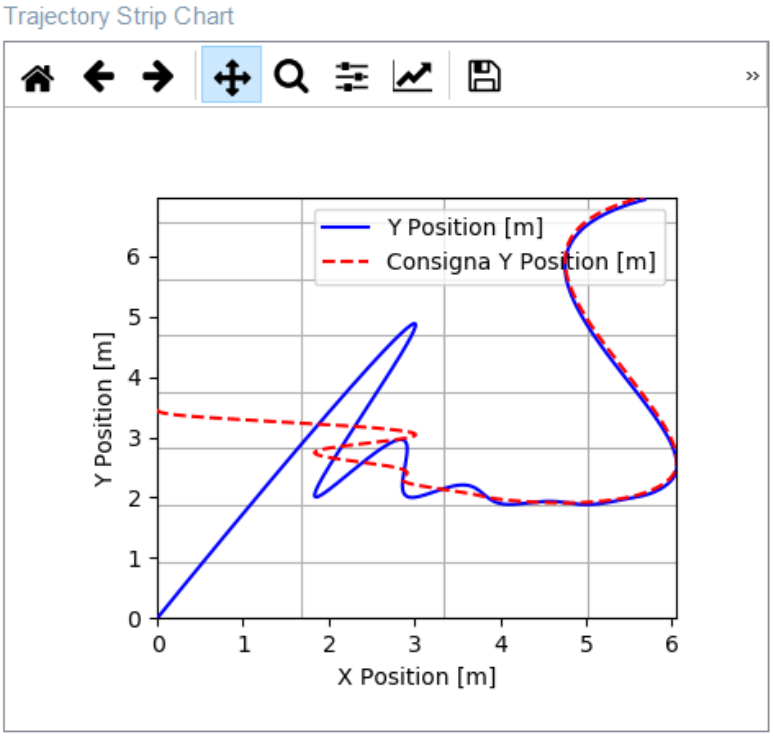


Figura 42 Trayectoria realizada por el CrazyFlie 2.0 en la simulación.

Como otro ejemplo, en la Figura 43 se presenta el módulo de la velocidad:

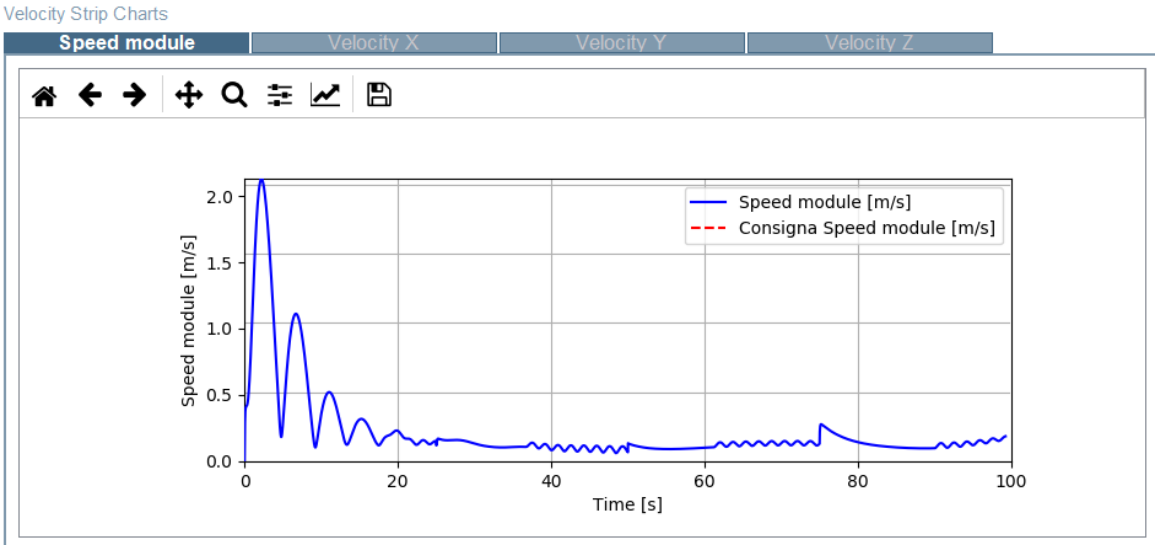


Figura 43 Módulo de la velocidad en la simulación.

Se observa en la anterior gráfica un pequeño ciclo límite, que coincide con el ciclo límite que puede observarse en la altura, incluida en la Figura 44.

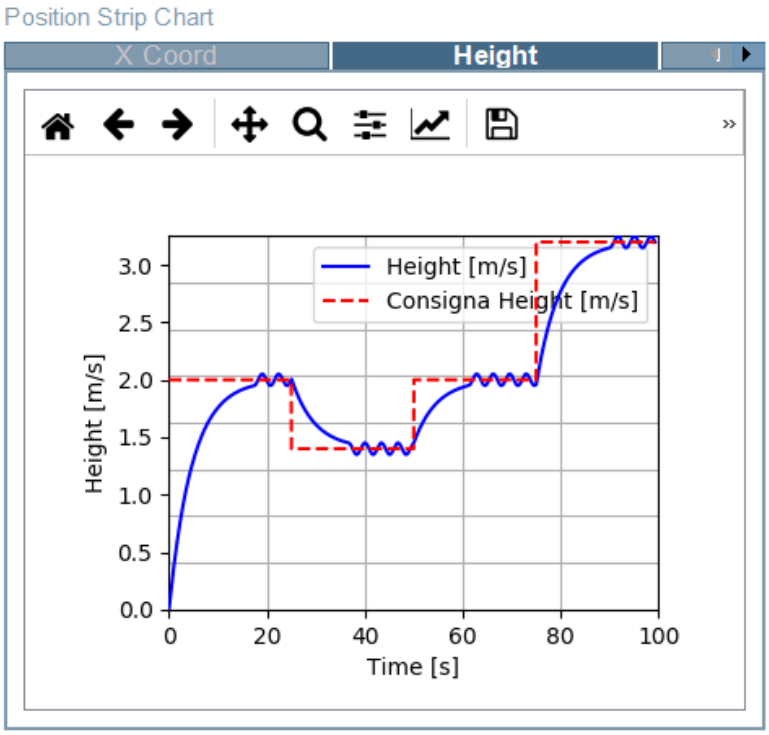


Figura 44 Evolución de la altura con el tiempo.

En la imagen anterior puede observarse muy bien cuándo se produce el ciclo límite. Ocurre cada vez que el sistema alcanza la trayectoria deseada y el error se hace muy pequeño. Mientras el error entre la altura actual del sistema y la consigna es suficientemente grande, el controlador aplica el PID diseñado y el error se va haciendo cada vez más pequeño. Sin embargo, cuando se está muy cerca de la consigna, el detector de eventos del PID basado en eventos detecta el evento a intervalos, haciendo que el sistema entre y salga de un lazo abierto, dando lugar al pequeño ciclo límite que se observa.

Para más información sobre los ciclos límite puede acudirse a la sección 5.2 de esta memoria.

Capítulo 8
Conclusiones y Líneas futuras

En este capítulo se detallan los logros alcanzados y las posibles líneas futuras de trabajo, como puede ser dotar al laboratorio de mayor funcionalidad y generalizarlo para nuevos tipos de controladores.

8. Conclusiones y líneas futuras

Se ha presentado un entorno de simulación o laboratorio virtual que permite simular y analizar el vuelo del hardware CrazyFlie 2.0 de forma muy rápida y eficiente. La razón principal del desarrollo de este proyecto fue la falta de un simulador interactivo con el que poder modificar el modelo del CrazyFlie 2.0 o sus controladores, a la vez que poder simular su comportamiento de forma ágil e intuitiva.

El simulador desarrollado ha sido probado extensivamente y se ha utilizado con éxito para analizar diversas trayectorias, diseñadas a su vez con el propio entorno. Esto ha permitido comprobar que este simulador permite examinar la dinámica del vuelo del CrazyFlie 2.0 en detalle.

Además, se han realizado también diversas simulaciones a partir de la modificación de modelos de control ya existentes, utilizando el modo de simulación *Real-Time*, y se ha comprobado que pueden modificarse las ganancias dentro del programa, llegando incluso a convertir un controlador PID en PID basado en eventos al cambiar los valores umbrales, y provocando así un ciclo límite que pudo visualizarse y analizarse en el simulador.

El hecho de poder guardar un vuelo realizado por el hardware CrazyFlie 2.0, conectado al programa, también se ha probado ser de gran interés, pues permite analizar cuantas veces sean necesarias, un vuelo ya realizado, que de otra forma no se podría recuperar. Esto supone una gran ventaja con respecto a los otros simuladores de CrazyFlie 2.0 existentes.

Finalmente, se ha comprobado la utilidad del módulo de estimación de la vida útil de la batería, pues permite saber con antelación si el hardware CrazyFlie 2.0 podrá o no realizar un determinado vuelo o trayectoria, o si por el contrario, se agotará la batería antes de terminar la misión de vuelo.

Un logro importante del entorno desarrollado es que sienta las bases de un futuros entornos de simulación más complejos, pues se ha implementado de forma muy modular. Esto permitirá investigar nuevos modelos dinámicos del hardware CrazyFlie 2.0, otros tipos de controladores, otras plataformas de vuelo, etc. Todo ello simplemente con añadir muy pocas líneas de código al laboratorio actual presentado.

8.1. Líneas de trabajo futuras

El mercado de los drones es un mercado en pleno apogeo y expansión. Esto quiere decir que pueden surgir, y surgirán, nuevas plataformas de vuelo similares al CrazyFlie 2.0, además de las ya existentes.

Para satisfacer esta rápida evolución y cambio, la estructura del simulador permite la adición de estas nuevas plataformas al modo de simulación *HIL* de forma muy rápida y con necesidad de muy pocas líneas de código. Esto haría posible, no sólo la grabación del vuelo del CrazyFlie 2.0, sino de otras plataformas como el PixHawk, Raspberri Pi, ArduPilot, etc.

Otras posibles mejoras a realizar al simulador serían:

- Poder añadir ruido a la simulación.
- Implementar el *firmware* de CrazyFlie 2.0 (y de la nueva versión salida al mercado CrazyFlie 2.1) y poder actualizar el controlador *On-board* de la plataforma con controladores diseñados en este entorno de simulación.
- Habilitar la adición de *plugins*, de forma que no tenga que ser el desarrollador quien implemente los controladores por defecto o el soporte de la plataforma de vuelo, si no que fuera el usuario quien lo desarrollara e implementara.
- Implementar más posibilidades en el diseño de trayectorias y no sólo líneas rectas o *splines* cúbicas.
- A nivel de programación, se querría mejorar el *performance* de la versión ejecutable del programa. Para ello se ha pensado en desarrollar un *thread* o hilo independiente, exclusivo para la interfaz de usuario, pues ésta tiene ya cierta entidad y hace al programa responder con cierta latencia.

Capítulo 9
Referencias y Bibliografía

En este capítulo se incluyen todas las referencias y la bibliografía utilizada para el proyecto.

9. Referencias y Bibliografía

1. Zaballa, N., (2017). La Industria de los drones. *Revista Forbes*. Recuperado desde: https://forbes.es/empresas/10287/la-industria-de-los-drones-en-5-anos/.
2. McClendon, A., (2020). Visión general estratégica y alcance de la Cámara de Consumo del *Drones Market*. Recuperado desde: https://www.torretriangular.com/camara-de-consumo-de-los-drones-ingresos-del-mercado-predicciones-de-crecimiento-y-jugadores-lideres-2020-2029/.
3. Jiménez, M. (2016). Drones, una industria creciente pero fragmentada. *Revista CincoDías.* Recuperado desde: https://cincodias.elpais.com/cincodias/2016/12/05/tecnologia/1480965319_898663.html.
4. Bitcraze. Recuperado de: https://www.bitcraze.io/crazyflie-2/.
5. Bitcraze. Recuperado de: https://wiki.bitcraze.io/projects:crazyflie2:expansionboards:index#compatibility_matrixes.
6. Bitcraze. Recuperado de: https://www.bitcraze.io/getting-started-with-the-crazyflie-2-0/.
7. Bitcraze. Recuperado de: https://www.bitcraze.io/.
8. Bitcraze, GitHub: (https://github.com/bitcraze). Accedido el 28.02.2020.
9. Socas, R., Dormido, R., Guinaldo, M., Dormido,S., (2019). A Control Engineering Framework for quadrotors.
10. CrazyFlie Simulator. Project aimed to modeling, developing and integrating the Crazyflie 2.0 nano-quadcopter in the physics based simulation environment Gazebo. Recuperado de: https://github.com/gsilano/CrazyS.
11. 4FLY Simulator. Institute of Control, Robotics and Information Engineering of the Poznan University of Technology. Recuperado de: http://wojciech.giernacki.pracownik.put.poznan.pl/page/2.
12. Silano, G., Lanelli, L., (2019). CrazyS: A Software-in-the-Loop Simulation Platform for the CrazyFlie 2.0 Nano-Quadcopter. *Robot Operating Systems (ROS),* volume 831. Recuperado desde: https://link.springer.com/chapter/10.1007/978-3-030-20190-6_4
13. Bouabdallah, S., Noth, A., & Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. *In Intelligent Robots and Systems*, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on (Vol. 3, pp. 2451-2456). IEEE.

14. Brown, L., Lindberg, G., Lilja, A., and Jeber, A., (2018). Design and Verification of Controller for Bitcraze CrazyFlie 2.0. *Gregslabaugh publications, Group 15.*
15. Landry, B., (2014). Planning and Control for Quadrotor Flight throught Cluttered Environments. *Master Thesis submitted to the Department of Electrical Engineering and Computer Science.*
16. Candan, F., Beke, A., Kumbasar, T., (2018). Design and Deployment of Fuzzy PID Controllers to the nano quadcopter CrazyFlie 2.0. *Conference INISTA 2018, at Greece*. DOI:10.1109/INISTA.2018.8466284.
17. Ngoc Thinh Nguyen, Ionela Prodan, Laurent Lefèvre. Flat trajectory design and tracking with saturation guarantees: a nano-drone application. International Journal of Control, Taylor & Francis, 2018, 10.1080/00207179.2018.1502474. hal-02074337.
18. Bitcraze. Recuperado de: https://www.bitcraze.io/2015/02/measuring-propeller-rpm-part-3/
19. Bitcraze. Recuperado de: https://wiki.bitcraze.io/_media/projects:crazyflie:hardware:fullriver_671723hs25c_wiki.png?cache=
20. Bitcraze. Recuperado de: https://wiki.bitcraze.io/misc:investigations:thrust
21. Recuperado de: http://wiki.mikrokopter.de/FlugZeit#Strom_ermitteln
22. Bitcraze. Recuperado de: https://store.bitcraze.io/products/240mah-lipo-battery-including-500ma-usb-charger
23. Bitcraze. Recuperado de: https://www.bitcraze.io/2015/02/measuring-propeller-rpm-part-3/
24. Bitcraze. Recuperado de: https://www.bitcraze.io/support/f-a-q/
25. Mathworks. Recuperado de: https://uk.mathworks.com/help/matlab/matlab_external/handle-data-returned-from-matlab-to-python.html
26. Farnell. Recuperado de: http://www.farnell.com/datasheets/2775765.pdf
27. Avnet. Recuperado de: https://www.avnet.com/shop/emea/products/invensense/mpu-9250307445734563529379 2?CMP=EMEA_Octopart_inventoryfeed_VSE
28. Bitcraze. Recuperado de: https://www.bitcraze.io/support/f-a-q/
29. Guzman, J. L., Astrom, K. J., Dormido, S., Hagglund, T., Berenguel, M., & Piguet, Y. (2008). Interactive learning modules for PID control [Lecture Notes]. *IEEE Control Systems Magazine*, *28*(5), 118-134.

30. Guzmán, J. L., Hägglund, T., Åström, K. J., Dormido, S., Berenguel, M., & Piguet, Y. (2014). Understanding PID design through interactive tools. *IFAC Proceedings Volumes*, *47*(3), 12243-12248.
31. Poirier, M. F., Stancy, A., Codres, E., Martines, M., (2018). *Control of a quadcopter CrazyFlie*. Univertisy of Manchester.
32. Sánchez J., Guinaldo M., Dormido S. y Visioli A., (2020). Validity of continuous tuning rules in event'based PI controllers using symmetric send-on-delta sampling: An experimental approach. *Computers and Chemical Engineering*.
33. Dormido S., Sánchez J., Kofman E., (2008). Muestreo, Control y Comunicación basados en eventos. *Revista Iberoamericana de Automática e Informática Industrial.* ISSN: 1697-7912. Vol. 5, Núm. 1.
34. Cervin A., Aström, Johan K., (2007). On Limit Cycles in Event-Based Control Systems. *46th IEE Conference on Decision and Control.*
35. Dormido S. Educación en Automática: nuevos retos. *Departamento de Informática y Automática, UNED.*
36. Sánchez, J., Visioli, A., & Dormido, S. (2012). Event-based PID control. In *PID control in the Third Millennium* (pp. 495-526). Springer, London.
37. Sánchez, J., Visioli, A., & Dormido, S. (2011). A two-degree-of-freedom PI controller based on events. *Journal of Process Control*, *21*(4), 639-651.
38. Pawlowski, A., Merigo, L., Guzmán, J. L., Dormido, S & J., Visioli, A. (2018). Two-degree-of-freedom control scheme for depth of hypnosis in anesthesia. *3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID* .
39. Socas, R., Dormido, S., Dormido, R. (2016). *Optimal Threshold Setting for Event-Based Control Strategies*. IEEE Access 2017, Vol. 5, 2880-2893.

Capítulo 10
Abreviaturas y Acrónimos

En este capítulo se incluyen todas las abreviaturas y acrónimos utilizados.

10. Abreviaturas y Acrónimos

Las abreviaturas y acrónimos usados se detallan en la siguiente tabla:

Nombre	Significado
AI	*Attitude Indicator*
BLE	*Bluetooth Low Energy*
CG	Centro de Gravedad
GUI	*Graphical User Interface*
HIL	*Hardware In the Loop*
IMU	*Inertial Measurement Unit*
MVC	*Model View Controller*
NED	Ejes *North-East-Down*

Tabla 4 Abreviaturas usadas en la memoria, la guía de usuario y el simulador czSimulator.

Anexos

En los siguientes anexos se incluye toda aquella información complementaria que se considera útil para el lector pero que no está directamente relacionada con la presentación del laboratorio virtual desarrollado.

Anexo A
Guía de Usuario

En este anexo se incluye la descripción del manejo del programa y cómo realizar todos los procesos descritos en capítulos anteriores.

A. Anexo A – Guía de usuario

A.1. Introducción

En este anexo se incluye la Guía de Usuario incluida en el simulador también.

A.2. Guía de usuario

En las siguientes páginas puede encontrarse la Guía de usuario para el czSimulator, el simulador modular e interactivo del CrazyFlie 2.0 que se ha desarrollado.

Máster en Ingeniería de Sistemas y Control



czSimulator – CrazyFlie Simulator

Autor: Blanca de la Parra de Toro

1

Simulador Interactivo de la Plataforma CrazyFlie 2.0



Guía de Usuario

Autor: Blanca de la Parra de Toro

2

Índice

1. Instalación y ejecución
2. Ventana de selección del Modo de Simulación
3. Modo *Offline*
4. Modo *Real-Time*
5. Modo HIL
6. *Settings* del programa

3

01.Instalación y Ejecución

4

01. Instalación y Ejecución

Simulador Interactivo de la Plataforma CrazyFlie 2.0

Instalación y Ejecución

Para instalar el programa basta con seguir los siguientes pasos:

1) Desempaquetar el *.zip que se ha recibido
2) Hacer doble-*click* en el fichero con nombre "czSimulator.exe"
3) Aparecerá la ventana de 'Logo' del simulador y, pasado un tiempo, la ventana de selección del Modo de Simulación

5

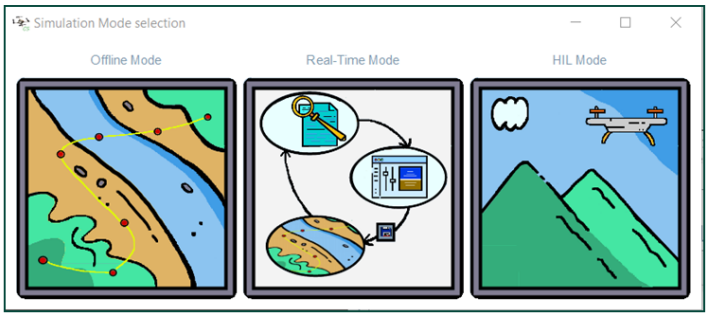
02. Ventana de selección del Modo de Simulación

6

02. Ventana de Selección del Modo de Simulación

Selección del Modo de Simulación

Una vez ejecutado el programa, después de la Ventana de 'Logo', aparecerá la ventana de selección del Modo de Simulación. Permite elegir entre: Modo *Offline*, *Real-Time* o HIL.



7

03. Modo *Offline*

8

3. Modo *Offline*

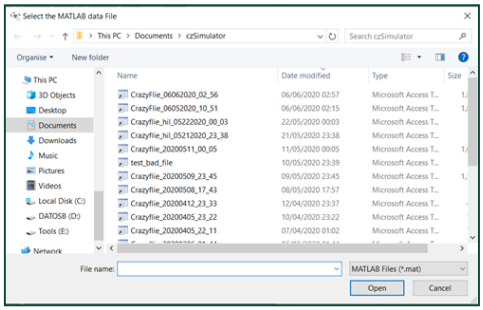
3.1 Selección del fichero *.mat
3.2 Asociación de nombres de variables
3.3 Descripción de la Ventana Principal
3.3.1 Menú Principal
3.3.2 Controles de la simulación
3.3.3 Alarmas
3.3.4 Ventana 3D
3.3.5 *Attitude Indicator* o Indicador de Actitud
3.3.6 Pestañas de parámetros
3.3.7 Gráficas
3.6 Planificador de Misión

9

03.
Modo *Offline*
*
Selección del fichero *.mat

Selección del fichero *.mat

Cuando se seleccione el Modo Offline, se mostrará una ventana de Explorer para la selección de un fichero de simulación, *.mat.



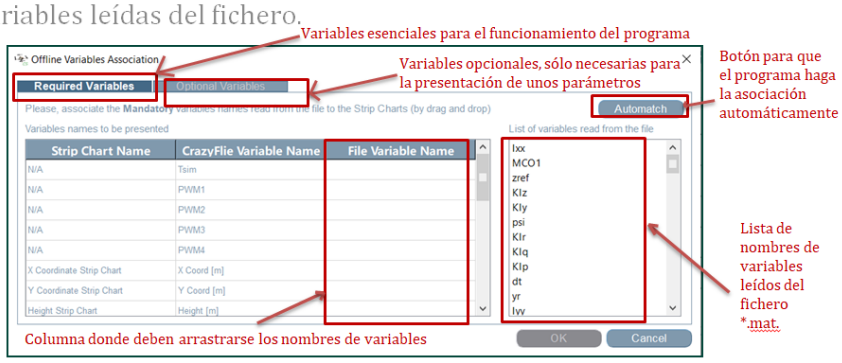
Seleccione el fichero a simular y pulse el botón «Open».

10

03.
Modo *Offline*
*
Asociación de nombres de variables

Asociación de nombres de variables

Cuando se seleccione el fichero, se mostrará el diálogo de asociación o *match* de los nombres de variables que espera el programa con las variables leídas del fichero.



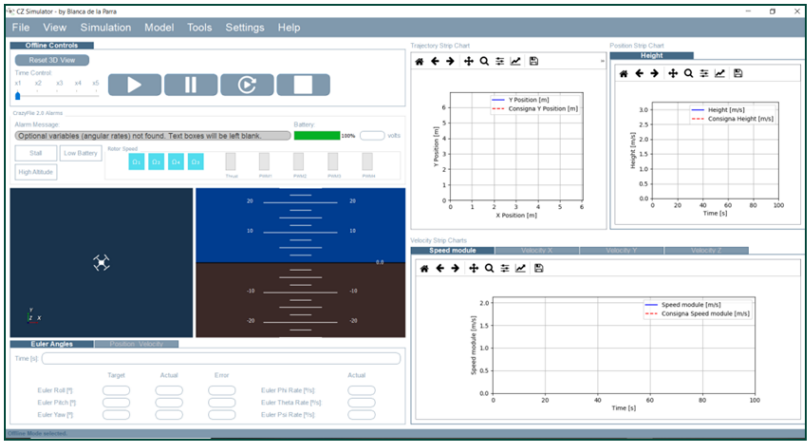
Una vez asociados los nombres desde la lista a la tabla, hacer *click* en «Ok».
Si se ha asociado el nombre incorrecto, basta con arrastrarlo de nuevo a la lista.

11

03.
Modo *Offline*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal

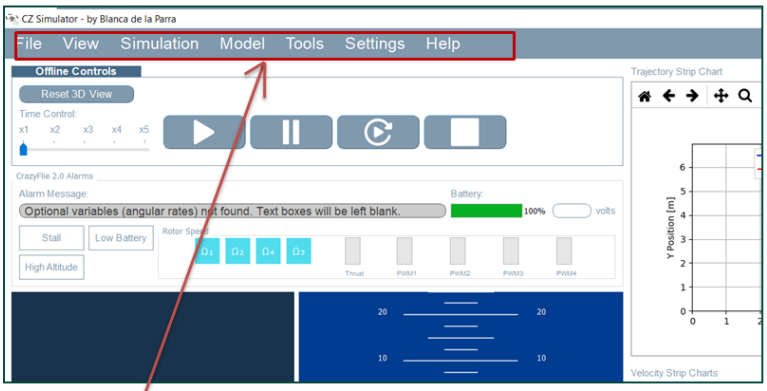
Una vez se asocien las variables, la Ventana Principal tendrá el siguiente aspecto:



12

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Principal



El menú principal se compone de distintos sub-menús (*File, View, etc.*)

Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (1/12).

13

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú File

- **Save template:** para una futura versión. Permitiría al usuario guardar el fichero *.mat leído y así simplificar el proceso.
- **Exit:** para poder salir del programa.



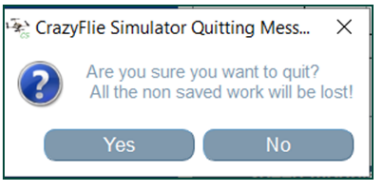
Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (2/12).

14

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú File > Exit

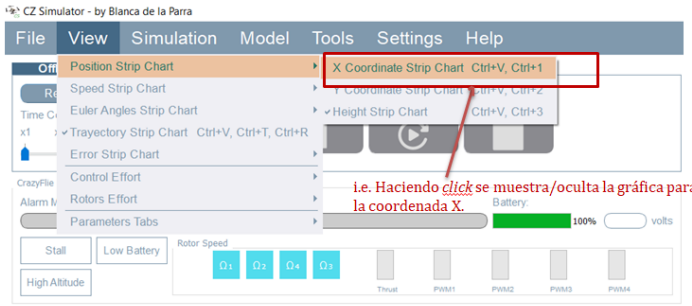
Se preguntará al usuario si realmente quiere salir del programa a través de un diálogo. Si se hace *click* en «Ok», se cerrará el programa.



03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú View

Cada gráfica tiene una opción *chequeable* en este menu. Si la opción tiene un *tick*, la gráfica será visible. Sino, no.



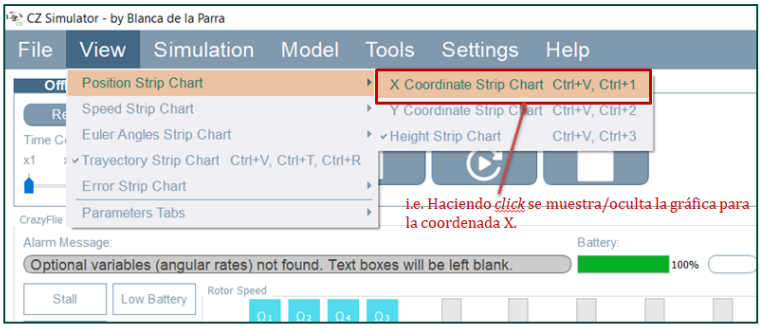
La configuración de este menú se guarda en las *settings*, y será leída de vuelta cada vez que se inicie el programa.

***IMPORTANTE:** Si se esconden todas las gráficas de un grupo, después deben añadirse en order (de arriba a abajo según aparecen en el menú).

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú View

Cada gráfica tiene una opción *chequeable* en este menu. Si la opción tiene un *tick*, la gráfica será visible. Sino, no.



La configuración de este menú se guarda en las *settings*, y será leída de vuelta cada vez que se inicie el programa.

Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (5/12).

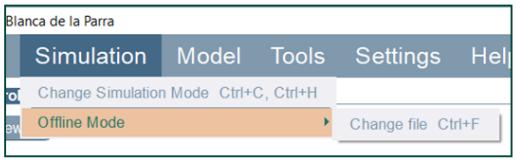
17

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Simulation

Cambiará según el Modo de Simulación.

- **Change Simulation Mode:** permite cambiar de un modo de simulación a otro.
- ***Offline Mode > Change file*:** permite cambiar el fichero *.mat leído.



Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (6/12).

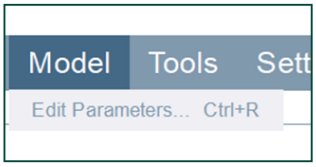
18

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Model

Cambiará según el Modo de Simulación.

- **Edit Parameters...:** permite cambiar los valores de los distintos parámetros del CrazyFlie 2.0. Éstos se usarán, por ejemplo, en el fichero que genere el Planificador de Misión.

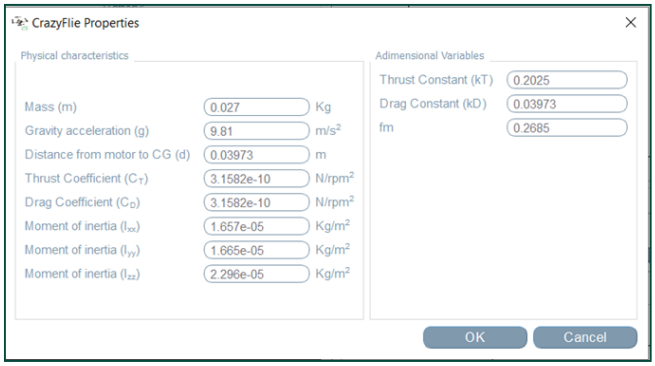


Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (7/12).

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Model > Edit Parameters...

Se presentará el diálogo de edición de parámetros del CrazyFlie 2.0. Si se hace *click* en «Ok», el programa guardará los nuevos valores.

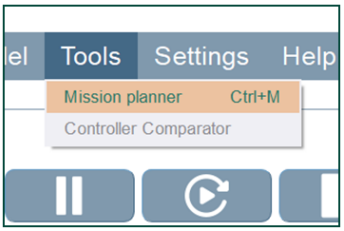


Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (8/12).

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Tools

- ***Mission Planner* o Planificador de Misión:** inicia el Planificador de Misión.
- ***Controller* Comparator:** para una futura versión. Permitirá comparar la salida de dos controladores.



Para el Planificador de Misión, referirse a la sección dedicada al mismo.

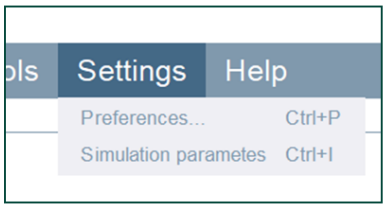
Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (9/12).

21

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Settings

- ***Preferences*:** cambia los *settings* o preferencias del programa.
- ***Simulation parameters*:** permite cambiar los parámetros de la simulación.



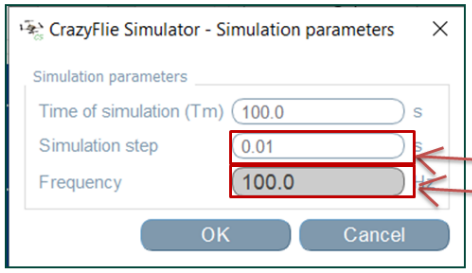
Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (10/12).

22

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Settings > Simulation parameters

Inicia el diálogo de cambio de los principales parámetros de la simulación.



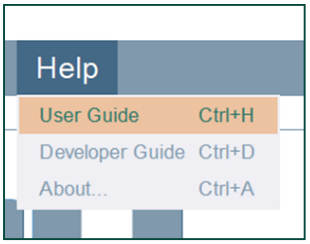
Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (11/12).

23

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Menú Principal

Menú Help

- ***User Guide***: inicia esta guía en PDF.
- ***Developer Guide***: guía de usuario pensada para desarrolladores (formato HTML de página web). Necesita credenciales.
- **About**: *información sobre el autor del programa y fecha de publicación del programa.*

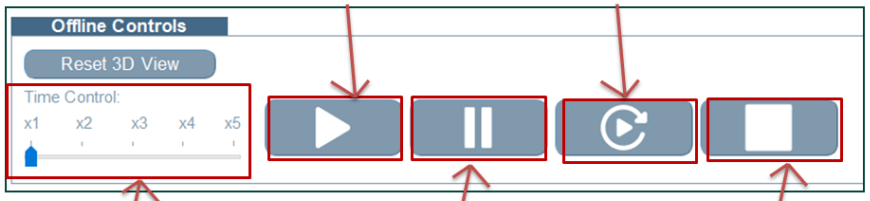


Modo *Offline*: Descripción de la Ventana Principal . Menú Principal (12/12).

24

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Controles de la simulación

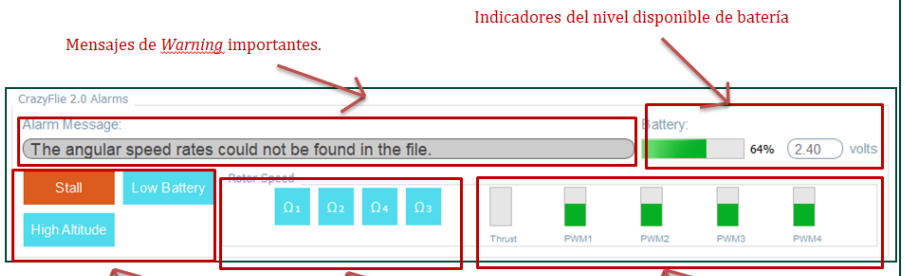
Controles de la simulación



25

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Alarmas

Alarmas



26

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Ventana 3D

Ventana 3D

Muestra el CrazyFlie 2.0 en 3D recorriendo la trayectoria comandada en el fichero *.mat.

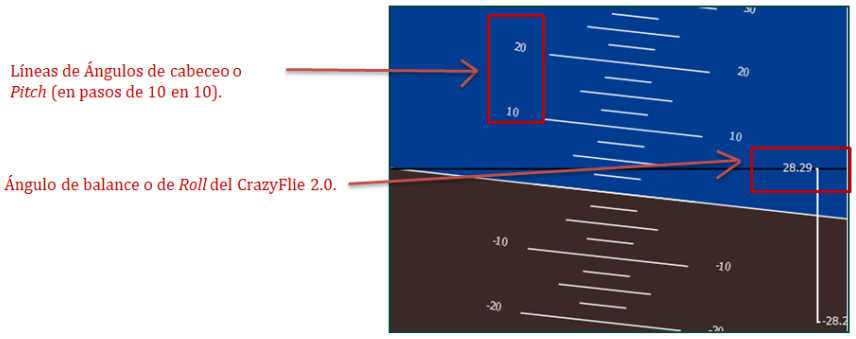


27

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Indicador de Actitud

Indicador de Actitud o *Attitud Indicator* (AI)

Indica los Ángulos de Euler actuales del CrazyFlie 2.0, según va recorriendo la trayectoria comandada en el fichero *.mat.



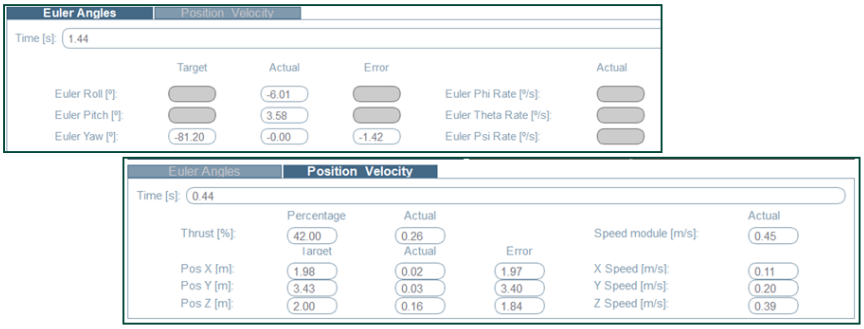
28

03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Pestañas de parámetros

Pestañas de parámetros

Está separada en dos pestañas:

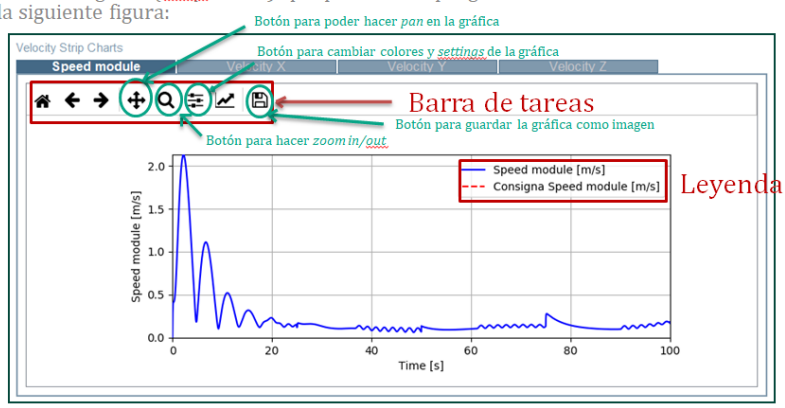
- **Ángulos de Euler**: contiene campos de texto con los valores de los tres ángulos de Euler actuales, los valores de consigna de la trayectoria deseada y los errores. Si se han leído del fichero, también mostrará las aceleraciones angulares .
- **Posición y Velocidad**: contiene campos de texto con los valores de la posición actual, la posición deseada y el error. Además, incluye los valores del módulo de velocidad y las componentes de la velocidad actual y del empuje, tanto en valor actual como en porcentaje con respecto al empuje máximo.



03.
Modo *Offline*
*
Descripción de la Ventana Principal
*
Gráficas

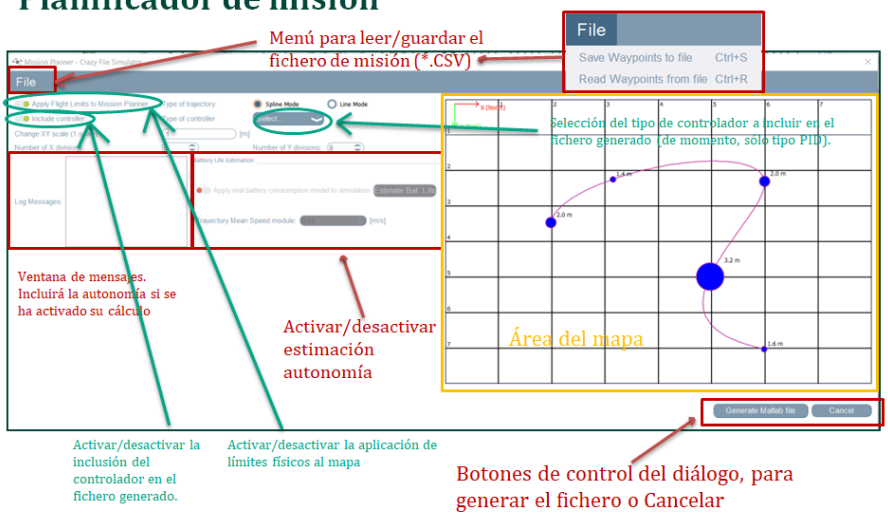
Gráficas

Cada uno de los elementos del menú View puede mostrarse/ocultarse. Todas las gráficas o cintas de registro *(strip charts)* que presenta el programa tienen la estructura mostrada en la siguiente figura:



03.
Modo *Offline*
*
Planificador de misión

Planificador de misión



Modo *Offline*: Planificador de Misión (1/2).

03.
Modo *Offline*
*
Planificador de misión

Planificador de misión



Modo *Offline*: Planificador de Misión (2/2).

04. Modo *Real-Time*

33

4. Modo *Real-Time*

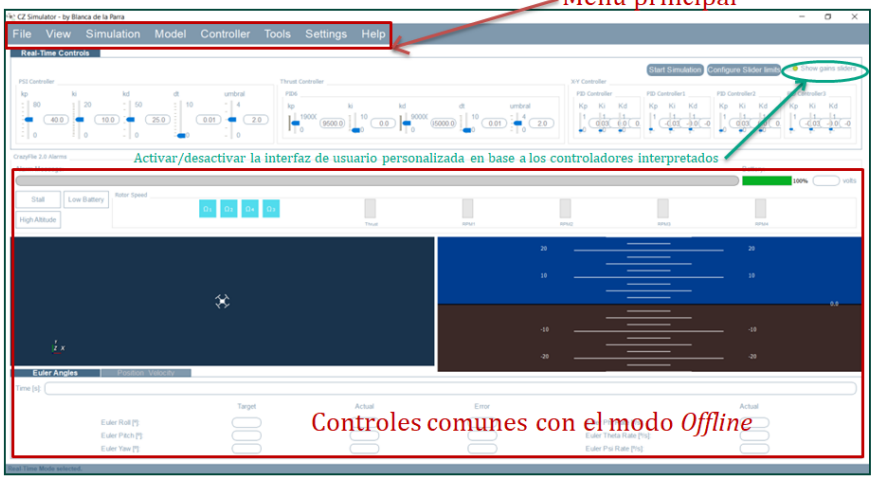
4.1 Descripción de la Ventana Principal
4.1.1 Configuración de los límites de los *slider*
4.1.2 Menú Principal
4.2 Lanzamiento de una nueva simulación
4.3 Simulación del resultado de la simulación

34

04.
Modo *Real-Time*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal

Si se lee uno de los controladores por defecto:



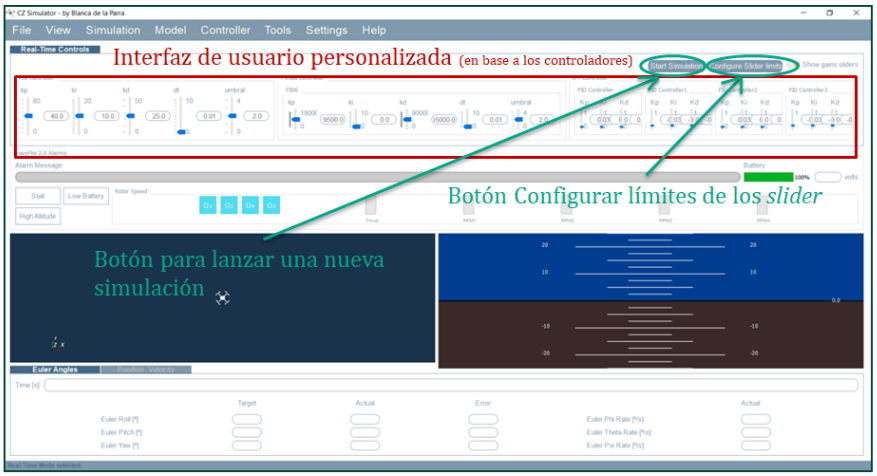
Modo *Real-Time*: Descripción de la Ventana Principal (1/5).

35

04.
Modo *Real-Time*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal

Si se lee uno de los controladores por defecto:



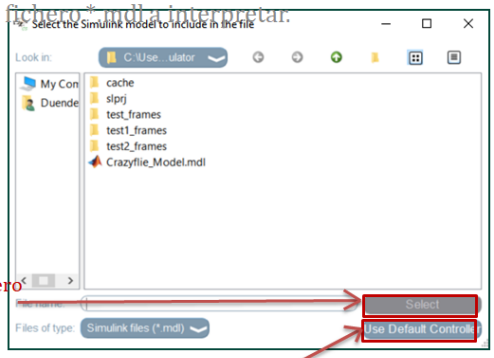
Modo *Real-Time*: Descripción de la Ventana Principal (2/5).

36

04.
Modo *Real-Time*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal

Si no se lee uno de los controladores por defecto, aparecerá una ventana de Explorer, para seleccionar el fichero *.mdl a interpretar.



Permite seleccionar cualquier fichero *.mdl para ser interpretado.

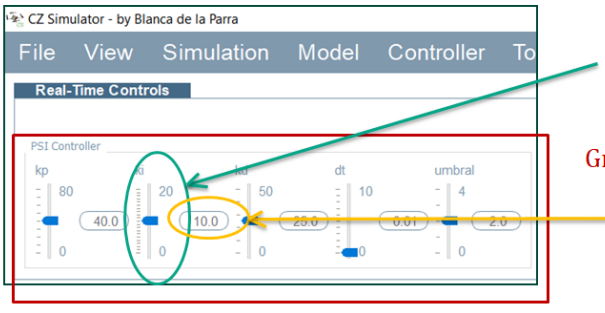
El programa leerá el controlador por defecto seleccionado en la ventana de *Settings*

Modo *Real-Time*: Descripción de la Ventana Principal (3/5).

04.
Modo *Real-Time*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal

Cada controlador leído del fichero *.mdl se incluirá en un grupo. Cada ganancia modificable será representada en la interfaz con un *slider*.



Slider de una ganancia

Grupo de control

Valor actual de la ganancia

Los controladores leídos que no tengan entradas configurables (constantes) se incluirán en una lista de controladores "no modificables".

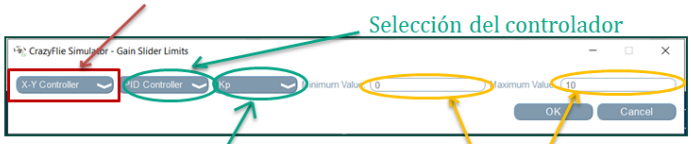
Modo *Real-Time*: Descripción de la Ventana Principal (4/5).

04.
Modo *Real-Time*
*
Descripción de la Ventana Principal

Descripción de la Ventana Principal: Configuración de los límites de los *Sliders*

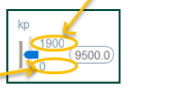
Cada controlador leído del fichero *.mdl se incluirá en un grupo. Cada ganancia modificable será representada, dentro de su grupo, con un *slider*.

Selección del grupo de controladores (página de Simulink®)



Selección de la ganancia

Valores mínimo y máximo del *Slider*.
Una vez se haga *click* en «Ok» se actualizarán los valores:



Modo *Real-Time*: Descripción de la Ventana Principal (5/5).

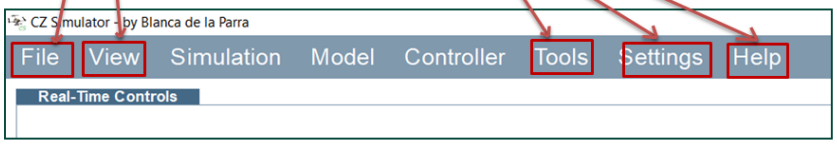
39

04.
Modo *Real-Time*
*
Menú Principal

Menú Principal

El Menú Principal del Modo *Real-Time* se muestra en la siguiente figura.

Menús comunes al modo *Offline* (consultar Modo *Offline*)



Modo *Real-Time*: Menú Principal (1/4).

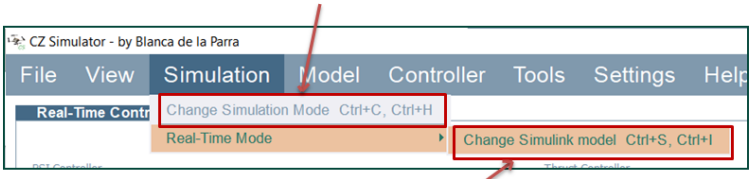
40

04.
Modo *Real-Time*
*
Menú Principal

Menú Simulation

Permite cambiar el Modo de Simulación o el fichero *.mdl interpretado.

Cambia el Modo de Simulación



Permite cambiar el fichero *.mdl leído.
Abrirá un diálogo de Explorer como en la slide 37, donde se podrá elegir el fichero, o usar un controlador por defecto.

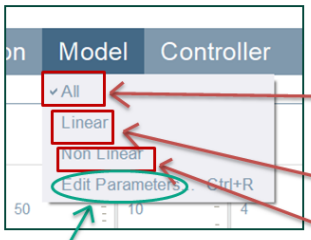
Modo *Real-Time*: Menú Principal (2/4).

41

04.
Modo *Real-Time*
*
Menú Principal

Menú Model

Permite filtrar por modelo los controladores por defecto disponibles en la ventana de *Settings*.



En *Settings* aparecerán para seleccionar todos los controladores por defecto sin filtro por tipo de modelo.

En *Settings* sólo aparecerán los controladores por defecto con modelo lineal.

En *Settings* sólo aparecerán los controladores por defecto con modelo no lineal.

Permite editar los parámetros del CrazyFlie 2.0

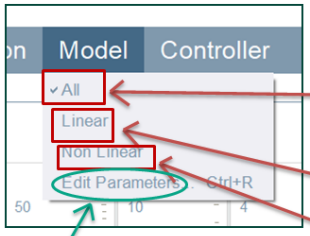
Modo *Real-Time*: Menú Principal (3/4).

42

04.
Modo *Real-Time*
*
Menú Principal

Menú Model: opción «Edit Parameters...»

Muestra un diálogo para poder modificar los principales parámetros del CrazyFlie 2.0.



En *Settings* aparecerán para seleccionar todos los controladores por defecto sin filtro por tipo de modelo.

En *Settings* sólo aparecerán los controladores por defecto con modelo lineal.

En *Settings* sólo aparecerán los controladores por defecto con modelo no lineal.

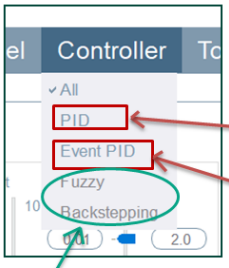
Permite editar los parámetros del CrazyFlie 2.0

Modo *Real-Time*: Menú Principal (3/4).

04.
Modo *Real-Time*
*
Menú Principal

Menú Controller

El Menú Principal del Modo *Real-Time* se muestra en la siguiente figura.



En *Settings* aparecerán para seleccionar todos los controladores por defecto que utilicen PID.

En *Settings* aparecerán para seleccionar todos los controladores por defecto que utilicen PID basado en eventos.

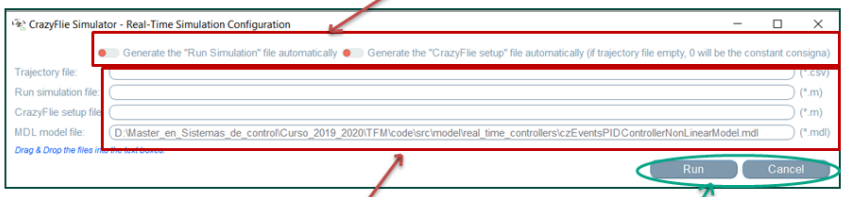
En desarrollo

Modo *Real-Time*: Menú Principal (4/4).

04.
Modo *Real-Time*
*
Lanzamiento de una nueva simulación

Lanzamiento de una nueva simulación

Una vez que se hayan modificado las ganancias de los controladores como se desee, basta con hacer *click* en «Start Simulation» para lanzar una nueva simulación. El siguiente diálogo aparecerá:



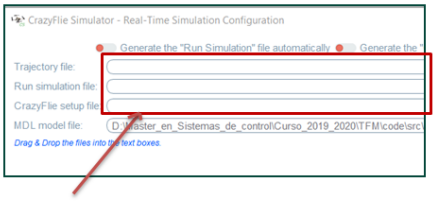
Modo *Real-Time*: Lanzamiento de una nueva simulación (1/3).

45

04.
Modo *Real-Time*
*
Lanzamiento de una nueva simulación

Lanzamiento de una nueva simulación

- **Trajectory file:** fichero *.CSV generado por el Planificador de Misión con la trayectoria a simular.
- **Run simulacion file:** Fichero que ejecuta la simulación en Matlab®. Consultar el fichero generado automáticamente por el programa para más información.
- **CrazyFlie setup file:** fichero donde se definen todos los parámetros del CrazyFlie 2.0. Consultar el fichero generado automáticamente por el programa para más información.
- **MDL model file:** nuevo fichero *.mdl generado a partir de las nuevas ganancias generadas por el usuario.



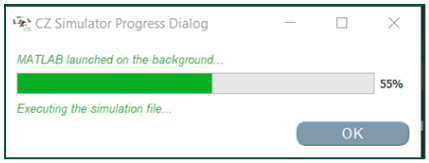
Modo *Real-Time*: Lanzamiento de una nueva simulación (2/3).

46

04.
Modo *Real-Time*
*
Lanzamiento de una nueva simulación

Lanzamiento de una nueva simulación

Cuando se lance la simulación, el programa tardará un tiempo* en responder mientras se lanza Matlab®en el *background* y se simula usándolo como *engine* de simulación. Para indicarlo, el programa mostrará una barra de progreso:



*En tests realizados, este tiempo suele ser inferior a dos minutos.

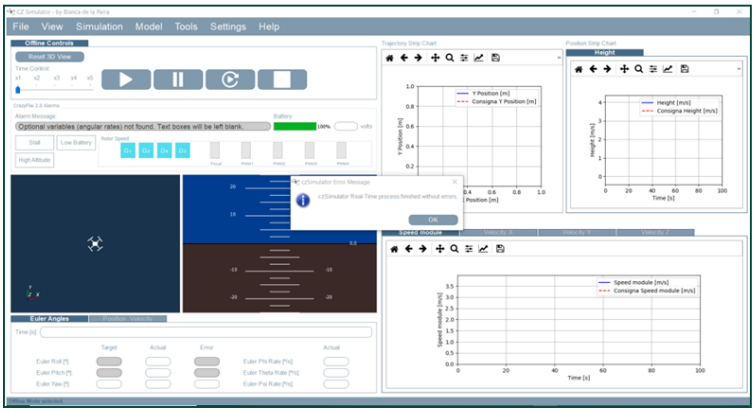
Modo *Real-Time*: Lanzamiento de una nueva simulación (3/3).

47

04.
Modo *Real-Time*
*
Simulación del resultado de la simulación

Simulación del resultado de la simulación

Una vez Matlab® finalice, el programa mostrará el modo de simulación *Offline*, habiendo leído por defecto la salida de la simulación. El usuario tendrá que hacer *match* de las variables y ya podrá analizar el vuelo como en el modo *Offline*.



48

05. Modo HIL

49

5. Modo HIL

5.1 Descripción de la Ventana Principal
5.1.1 Menú Principal

50

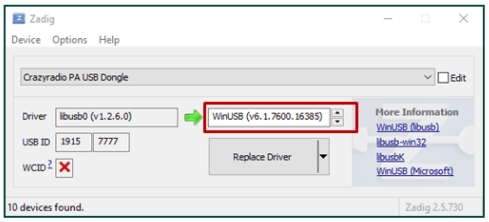
06. Ventana Principal * Modo HIL * Instalación de los drivers del CrazyFlie radio

Instalación de los drivers del CrazyFlie radio

Para poder conectar el CrazyFlie al programa, se debe instalar los drivers del CrazyFlie radio.

Para ello:

1) Conectar la radio del CrazyFlie (CrazyFlie radio)
2) Ejecutar el programa Zadig (incluido con la distribución del simulador) o disponible en: https://zadig.akeo.ie/
3) Seleccionar como dispositivo «Crazyradio PA USB Dongle» e instalar el driver «**WinUSB**», como se muestra en la figura



Modo HIL: Instalación de los drivers del CrazyFlie radio (1/2).

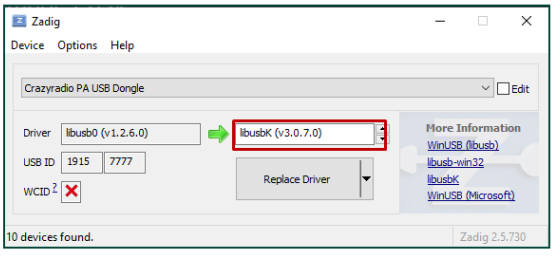
51

06. Ventana Principal * Modo HIL * Instalación de los drivers del CrazyFlie radio

Instalación de los drivers del CrazyFlie radio

Si los problemas para conectar el dispositivo al simulador persisten, seguir los siguientes pasos:

1) Conectar la radio del CrazyFlie (CrazyFlie radio)
2) Ejecutar el programa Zadig
3) Seleccionar como dispositivo «Crazyradio PA USB Dongle» e instalar el driver «**libusbK**», como se muestra en la figura



Modo HIL: Instalación de los drivers del CrazyFlie radio (2/2).

52

06. Ventana Principal * Modo HIL * Descripción de la Ventana Principal

Descripción de la Ventana Principal

IMPORTANTE: El CrayFlie 2.0 debe estar conectado antes de iniciar el programa.

Una vez se seleccione el Modo HIL, el aspecto del programa será:



Sólo se describirán los controles específicos de este modo. Los demás están descritos en el Modo *Offline* (alarmas, pestañas de parámetros, ventana 3D, Indicador de Actitud y gráficas).

Modo HIL: Descripción de la Ventana Principal (1/3).

06. Ventana Principal * Modo HIL * Descripción de la Ventana Principal

Descripción de la Ventana Principal

Una vez se seleccione el Modo HIL, el aspecto del programa será:



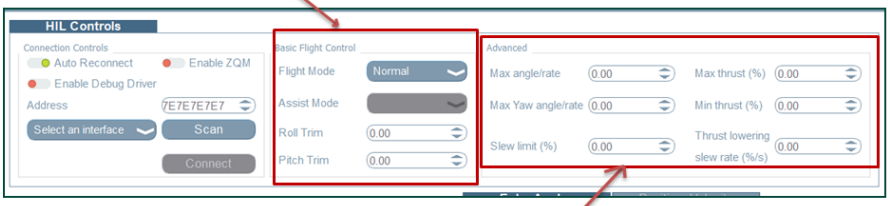
Modo HIL: Descripción de la Ventana Principal (2/3).

06. Ventana Principal * Modo HIL * Descripción de la Ventana Principal

Descripción de la Ventana Principal

Una vez se seleccione el Modo HIL, el aspecto del programa será:

Establece tipo de vuelo del CrazyFlie 2.0 (como el CrazyFlie PC Client)



Establece límites al vuelo del CrazyFlie 2.0 (si el modo avanzado está activado).

Modo HIL: Descripción de la Ventana Principal (3/3).

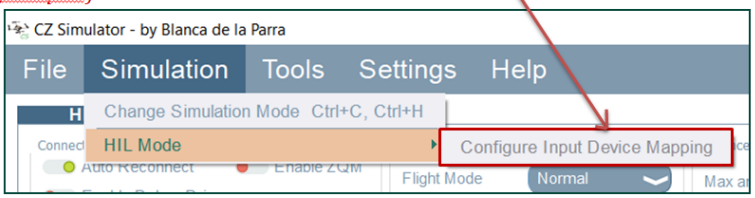
55

06. Ventana Principal * Modo HIL * Menú Principal

Menú Principal

Una vez se seleccione el Modo HIL, el aspecto del programa será:

Inicia el diálogo de calibración del control del CrazyFlie 2.0 (con un *gamepad*). En desarrollo.



Modo HIL: Descripción de la Ventana Principal . Menú Principal (1/2).

56

06. *Settings* del programa

57

6. *Settings* del programa

6.1 Descripción de la Ventana de Preferencias
6.1.1 Pestaña *«General Configuration»*
6.1.2 Pestaña *«Flight Limitations»*
6.1.3 Pestaña *«Controller Limitations»*
6.1.4 Pestaña *«Real-Time Settings»*

58

06. Ventana Principal * *Settings* del programa * Descripción de la Ventana de Preferencias

Descripción de la Ventana de *Preferences*

Permite cambiar ciertas preferencias del programa.



Diferentes pestañas de configuración

Guarda un nuevo fichero de *Settings* y las aplica al programa

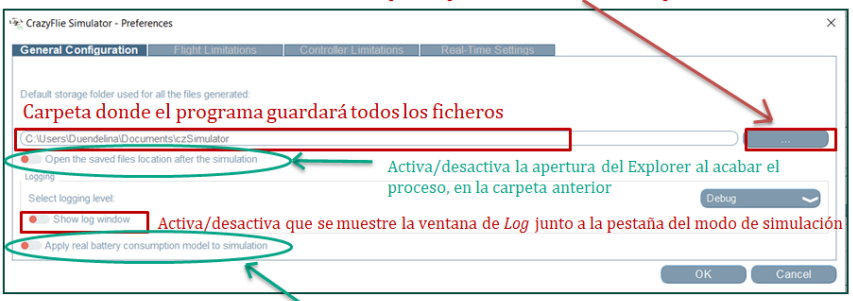
***Settings del programa*: Ventana de preferencias (1/5).**

06. Ventana Principal * *Settings* del programa * Descripción de la Ventana de Preferencias

Pestaña de «General Configuration»

Permite cambiar ciertas preferencias del programa.

Abrirá una ventana de Explorer para seleccionar otra carpeta



Activa/desactiva que se pare la simulación en caso de que el CrazyFlie 2.0 se quede sin batería (según la estimación de la autonomía).

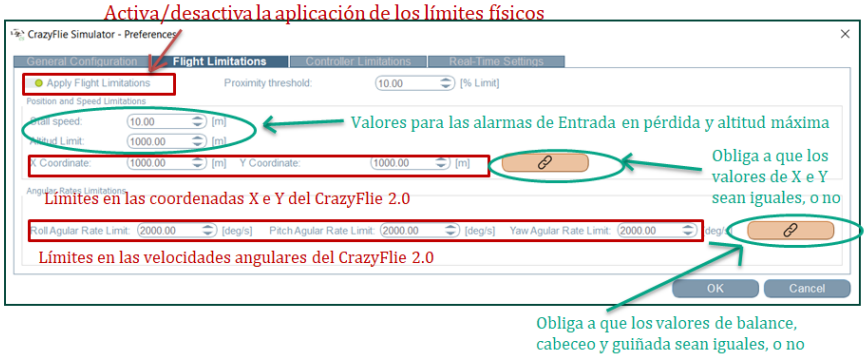
***Settings del programa*: Ventana de preferencias (2/5).**

06. Ventana Principal * *Settings* del programa * Descripción de la Ventana de Preferencias

Simulador Interactivo de la Plataforma CrazyFlie 2.0

Pestaña de «Flight Limitations»

Permite aplicar límites físicos al vuelo del CrazyFlie 2.0. Si están activados, el programa se parará en caso de superen estos límites.



***Settings del programa*: Ventana de preferencias (3/5).**

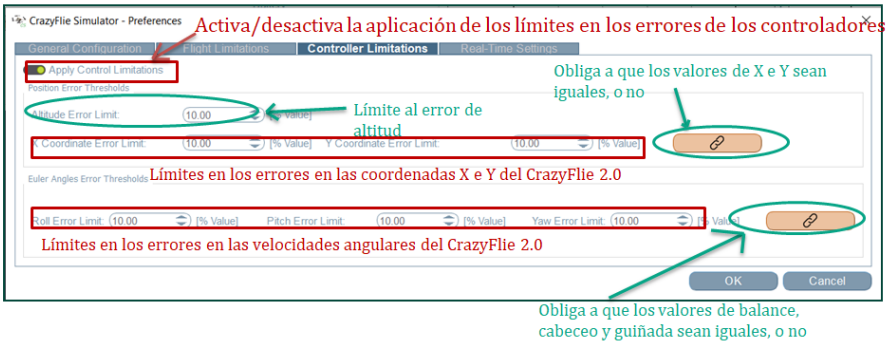
61

06. Ventana Principal * *Settings* del programa * Descripción de la Ventana de Preferencias

Simulador Interactivo de la Plataforma CrazyFlie 2.0

Pestaña de «Controller Limitations»

Permite aplicar límites a los errores usados para el control del vuelo del CrazyFlie 2.0. Si están activados, el programa se parará en caso de superen estos límites.



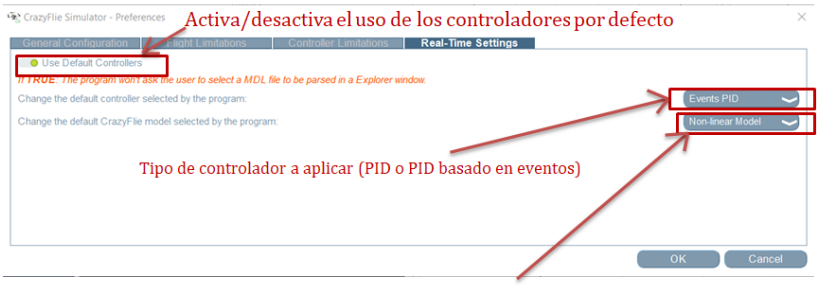
***Settings del programa*: Ventana de preferencias (4/5).**

62

06. Ventana Principal
*
Settings del programa
*
Descripción de la Ventana de Preferencias

Pestaña de «Real-Time Settings»

Permite configurar si el modo *Real-Time* debe usar un controlador por defecto (fichero *.mdl) y definir qué controlador por defecto elegir.



***Settings del programa*: Ventana de preferencias (5/5).**

63

FIN

64

Anexo B Limitaciones Físicas del CrazyFlie 2.0 y Limitaciones en los Errores del Controlador

En este anexo se describen todas las limitaciones que el usuario puede aplicar en el simulador.

B. Anexo B - Limitaciones Físicas del CrazyFlie 2.0 y Limitaciones a los Errores del Controlador

B.1. Introducción

En este anexo se presentan los dos tipos de limitaciones que se pueden aplicar en el simulador a través de la ventana de *settings*.

B.2. Limitaciones físicas al vuelo

En esta sección se detallan las limitaciones físicas que se pueden aplicar en el simulador:

- Valor de la velocidad de entrada en pérdida o *stall*: Este valor afectará a la alarma que muestra si el CrazyFlie 2.0 está cercano a la pérdida.
- Límite de altitud: Este valor afectará a la alarma que informa al usuario del estado del CrazyFlie 2.0 con respecto a esta altitud límite.
- Límites en las coordenadas X e Y: Estos límites afectarán al Planificador de Misión, pues se limitará el tamaño del mapa para no permitir diseñar una trayectoria que supere estos límites.
- Límites a las velocidades angulares: Si se activa la aplicación de estos límites físicos a la simulación del vuelo del CrazyFlie 2.0, el simulador parará la simulación cada vez que se sobrepase uno de estos límites. Esto permite saber con exactitud si se alcanzan ciertos valores de ciertas variables del vuelo, sin necesidad de tener que estar pendiente de las gráficas y de cómo evolucionan con el tiempo.

En la Figura 45 se muestra el diálogo del simulador para configurar los límites físicos a aplicar al vuelo del CrazyFlie 2.0.

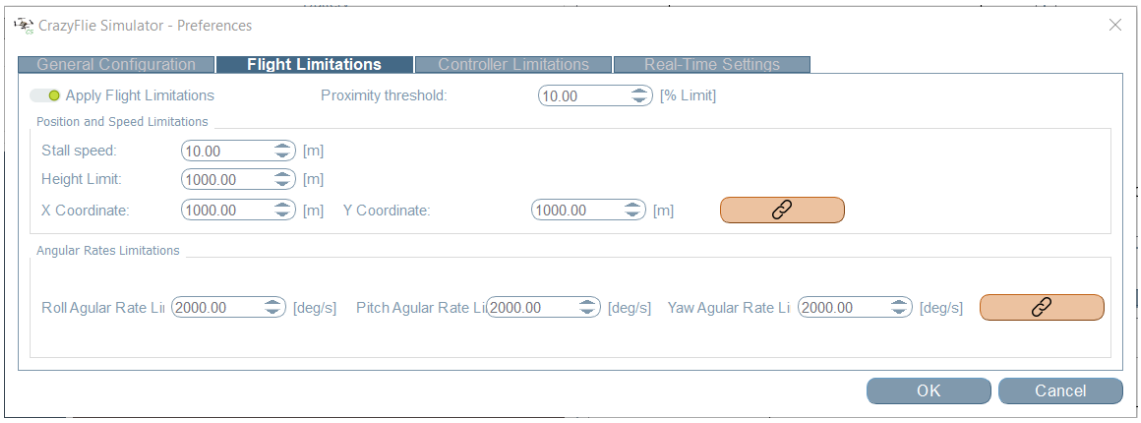


Figura 45 Ventana de *settings* para la aplicación de límites físicos al vuelo.

Como ya se ha indicado, si alguna de las limitaciones anteriores se alcanza o supera, el simulador se parará automáticamente y se iluminarán las alarmas pertinentes, para indicar al usuario por qué se ha abortado la simulación. Para más detalles sobre cómo desactivar o cambiar los valores de estas limitaciones, consultar el Anexo A de esta memoria.

B.3. Limitaciones en los Errores de los Controladores

El simulador permite también la aplicación de otro tipo de limitaciones, que consisten en restringir los errores en posición y actitud del CrazyFlie 2.0. De esta forma, se puede detectar rápidamente si un controlador empieza a volver inestable. En la Figura 46 se muestran todas las opciones de las que dispone el simulador para aplicar límites a los errores de los controladores.

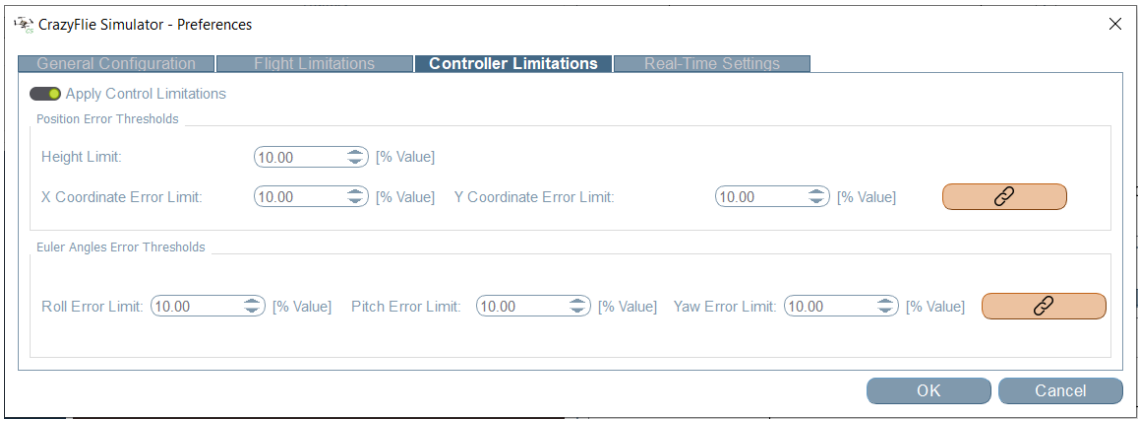


Figura 46 Ventana de *settings* para la aplicación de límites a los errores de los controladores.

Como puede comprobarse en la anterior figura, pueden limitarse:

- El error en altura
- El error en posición de las coordenadas X e Y
- El error en los tres Ángulos de Euler, ϕ, θ y ψ

Las limitaciones que se han incluido son las siguientes:

B.4. Valores por defecto de las Limitaciones Físicas

Por defecto en el programa, las limitaciones físicas al vuelo estarán deshabilitadas. Sin embargo, si son activadas por el usuario, los valores por defecto que se han aplicado a cada una de las limitaciones son:

- Altura debe ser menor de 1km: Para que la radio tenga alcance suficiente, la altura debe ser menor de 1km. Este límite se ha obtenido de [26].
- Delta de posición en X e Y menor de 1km: Para que la radio del CrazyFlie 2.0 tenga alcance suficiente, se guarda el punto inicial en el que se inicia el vuelo y se calcula que el diferencial de posición en la coordenada X o Y no sea mayor de 1km. Este límite se ha obtenido de [26].
- Aceleración angular: Este límite se ha obtenido a partir de la máxima aceleración angular que los giróscopos incluidos en la *Inertial Measurement* Unit (IMU) montada en el CrazyFlie 2.0, la MPU-9250, pueden medir. Este valor es igual a ±250º/s.

Si alguna de las limitaciones anteriores se alcanza o supera, el simulador se parará automáticamente y se iluminarán las alarmas pertinentes, para indicar al usuario por qué se ha abortado la simulación. Si las limitaciones están activadas, no sólo se parará la

simulación si no que, cuando el valor actual esté cerca del límite, el cuadro de texto de las pestañas de parámetros cambiará de estilo para indicar que se está alcanzando el límite de esa variable. La definición de ese "cerca" es la siguiente:

$$\underbrace{\alpha}_{\text{limit}} - \underbrace{\beta}_{actual\ value} \leq \frac{10}{100} \cdot \alpha \tag{1.11}$$

Esto es, la diferencia entre el valor actual y el límite es inferior al 10% del valor del límite. Este parámetro (10%) también se puede cambiar en la ventada de *settings*. Igualmente, en la ventana de preferencias el usuario podrá desactivar la aplicación de estas limitaciones al vuelo, aunque estará activada por defecto. Igualmente, los valores de las limitaciones se pueden modificar en este diálogo. Para más detalles sobre cómo desactivar o cambiar los valores de estas limitaciones, consultar el Anexo A de esta memoria.

Anexo C
Modelo Dinámico del CrazyFlie 2.0

En este anexo se incluyen todas las ecuaciones dinámicas del CrazyFlie 2.0 y los modelos utilizados en el simulador.

C. Anexo C – Modelo Dinámico del CrazyFlie 2.0

C.1.Introducción

En este anexo se presentan las principales ecuaciones dinámicas demo modelo del CrazyFlie 2.0 y cómo se han obtenido, definiendo para ello los sistemas de referencia usados y las hipótesis aplicadas.

C.2.Hipótesis

A continuación se incluyen las hipótesis realizadas para poder formular las ecuaciones dinámicas del CrazyFlie 2.0.

- Cuerpo rígido
- Cuerpo simétrico
- Masa del CrazyFlie 2.0 constante
- Las fuerzas de empuje no dependen de la geometría (son simétricas)

C.3.Definiciones preliminares

C.3.1. Sistema de referencia

En este apartado se recoge la definición de los ejes cuerpo del CrazyFlie 2.0, que se usarán para expresar las ecuaciones dinámicas. En la siguiente figura se muestran los 4 rotores del CrazyFlie 2.0, cómo se han numerado y cómo se han tomado los ejes cuerpo con respecto a esa numeración.

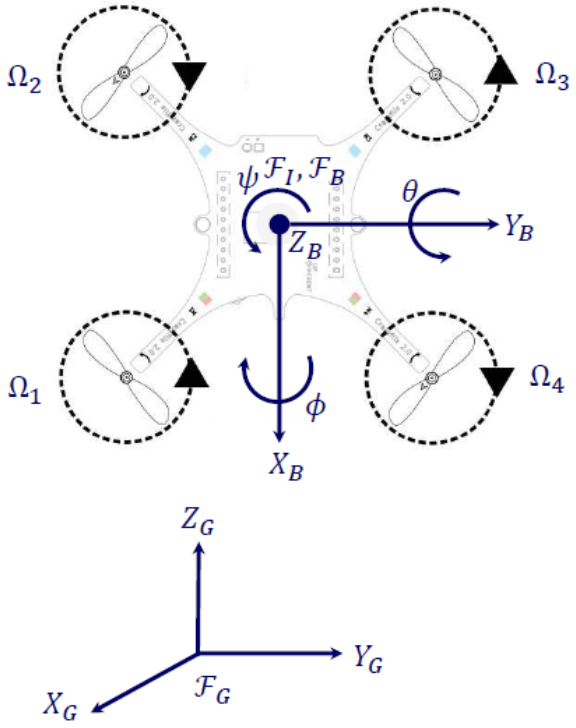


Figura 47 Representación de los ejes cuerpo del CrazyFlie 2.0.

Como puede verse en la anterior figura, los rotores están numerados de 1 a 4. El eje X cuerpo (X_B) se encuentra entre los motores 1 y 4 y el eje Y cuerpo (Y_B) entre los motores 4 y 3. El eje Z cuerpo (Z_B) forma un triedro a derechas con los anteriores. Estos tres ejes componen el sistema de referencia de ejes cuerpo (F_B).

También se utilizarán los ejes inerciales (F_I), cuyo origen se situará en el Centro de Gravedad (CG) del CrazyFlie 2.0, y cuyos ejes X, Y, Z siguen el criterio NED o ENU (*North-East-Down*), por el que cada uno de ellos se alinea con el Norte, Este y 'Abajo', respectivamente, de un plano tangencial a la superficie de la Tierra, con punto de tangencia la posición del CrazyFlie 2.0.

C.3.2. Variables

En esta sección se incluye la definición de las variables que luego se usarán en las ecuaciones dinámicas.

F_z	Fuerza de ascenso del CrazyFlie 2.0, en ejes cuerpo.
$\begin{pmatrix} M_x & M_y & M_z \end{pmatrix}^T$	Componentes de los momentos de torsión en cada uno de los ejes cuerpo.
d	Distancia del motor al Centro de Gravedad (CG).
C_T	Coeficiente de Empuje (adimensional).
C_D	Coeficiente de Resistencia Aerodinámica (adimensional).
$\Omega_i \ \ (i=1,..,4)$	Velocidad angular de rotación de cada uno de los motores.
Ω_e	Velocidad angular de rotación de equilibrio de los rotores.
$\begin{pmatrix} p & q & r \end{pmatrix}^T$	Velocidad angular del CrazyFlie 2.0, expresada en ejes cuerpo.
$\begin{pmatrix} \phi & \theta & \psi \end{pmatrix}^T$	Ángulos de Euler, también conocidos como ángulos de Tait Bryan de los ejes cuerpo del CrazyFlie 2.0 con respecto a los ejes inerciales.
$\begin{pmatrix} u & v & w \end{pmatrix}^T$	Componentes de la velocidad del CrazyFlie 2.0 con respecto al plano inercial.
$\begin{pmatrix} x & y & z \end{pmatrix}^T$	Posición del CrazyFlie 2.0 con respecto a los ejes inerciales.

C.3.3. Ecuaciones dinámicas

Teniendo en cuenta las hipótesis anteriores y los sistemas de referencia descritos, se pueden formular las ecuaciones dinámicas que se recogen en las siguientes secciones.

C.3.3.1. Modelo No Lineal

El vector de estado del CrazyFlie 2.0 se define como:

$$X = \begin{pmatrix} x & y & z & \phi & \theta & \psi & p & q & r \end{pmatrix}^T \tag{1.1}$$

Las ecuaciones de estado del CrazyFlie 2.0 son:

$$\ddot{x} = \frac{F_z}{m} \cdot \left(C_\psi S_\theta C\phi + S_\psi S_\phi \right) \tag{1.2}$$

$$\ddot{y} = \frac{F_z}{m} \cdot \left(S_\psi S_\theta S_\phi + C_\psi S_\phi \right) \tag{1.3}$$

$$\ddot{z} = \frac{F_z}{m} \cdot \left(C_\theta C_\phi \right) - g \tag{1.4}$$

$$\dot{\phi} = p + T_\theta S_\phi q + T_\theta C_\phi r \tag{1.5}$$

$$\dot{\theta} = C_\phi q - S_\phi r \tag{1.6}$$

$$\dot{\psi} = \frac{S_\phi}{C_\theta} q + \frac{C_\phi}{C_\theta} r \tag{1.7}$$

$$\dot{p} = qr \cdot \frac{\left(I_{yy} - I_{zz} \right)}{I_{xx}} + \frac{M_x}{I_{xx}} \tag{1.8}$$

$$\dot{q} = rp \cdot \frac{\left(I_{zz} - I_{xx} \right)}{I_{yy}} + \frac{M_y}{I_{yy}} \tag{1.9}$$

$$\dot{r} = pq \cdot \frac{\left(I_{yy} - I_{xx} \right)}{I_{zz}} + \frac{M_z}{I_{zz}} \tag{1.10}$$

En las anteriores ecuaciones:

$$C_\Phi = \cos\left(\Phi \right) \tag{1.11}$$

$$S_\Phi = \sin\left(\Phi \right) \tag{1.12}$$

C.3.3.2. Modelo Lineal

Las ecuaciones expresadas en el apartado anterior se pueden simplificar, asumiendo que el sistema está en equilibrio, y por tanto, asumiendo lo siguiente:

$$\phi, \theta, \psi \approx 0 \tag{1.13}$$

Se asume que los ángulos de Euler son siempre ángulos pequeños, pudiendo simplificar:

$$\cos(\Phi) \approx 1 \tag{1.14}$$

$$\sin(\Phi) \approx 0 \tag{1.15}$$

Siendo Φ cualquiera de los ángulos mencionados.

Realizando estas sustituciones en las ecuaciones (1.2)-(1.10), éstas resultan:

$$\ddot{x} = 0 \rightarrow \dot{x} = u \tag{1.16}$$

$$\ddot{y} = 0 \rightarrow \dot{y} = v \tag{1.17}$$

$$\ddot{z} = \frac{F_z}{m} - g \tag{1.18}$$

$$\dot{\phi} = p \tag{1.19}$$

$$\dot{\theta} = q \tag{1.20}$$

$$\dot{\psi} = r \tag{1.21}$$

$$\dot{p} = qr \cdot \frac{(I_{yy} - I_{zz})}{I_{xx}} + \frac{M_x}{I_{xx}} \tag{1.22}$$

$$\dot{q} = rp \cdot \frac{(I_{zz} - I_{xx})}{I_{yy}} + \frac{M_y}{I_{yy}} \tag{1.23}$$

$$\dot{r} = pq \cdot \frac{(I_{yy} - I_{xx})}{I_{zz}} + \frac{M_z}{I_{zz}} \tag{1.24}$$

Esto permite desacoplar las dinámicas de los diferentes modos de vuelo.

C.3.4. Ecuaciones del movimiento vertical

Las ecuaciones del movimiento vertical del CrazyFlie 2.0, con el modelo lineal, son:

$$\begin{bmatrix} \dot{w} \\ \dot{z} \end{bmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \cdot \begin{bmatrix} w \\ z \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix} \cdot F_z \tag{1.25}$$

C.3.5. Ecuaciones del movimiento direccional

Las ecuaciones del movimiento direccional del CrazyFlie 2.0, con el modelo lineal, son:

$$\begin{bmatrix} \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \cdot \begin{bmatrix} r \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{zz}} \\ 0 \end{bmatrix} \cdot M_z \tag{1.26}$$

C.3.6. Ecuaciones del movimiento lateral

Las ecuaciones del movimiento lateral del CrazyFlie 2.0, con el modelo lineal, son:

$$\begin{bmatrix} \dot{p} \\ \dot{\phi} \\ \dot{v} \\ \dot{y} \end{bmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} p \\ \phi \\ v \\ y \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot M_x \tag{1.27}$$

C.3.7. Ecuaciones del movimiento Longitudinal

Las ecuaciones del movimiento longitudinal del CrazyFlie 2.0, con el modelo lineal, son:

$$\begin{bmatrix} \dot{q} \\ \dot{\theta} \\ \dot{u} \\ \dot{x} \end{bmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} q \\ \theta \\ u \\ x \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{yy}} \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot M_{yy} \tag{1.28}$$

C.3.8. Punto de *trim* o de equilibrio del CrazyFlie 2.0

Para linealizar el modelo no lineal del CrazyFlie 2.0, como ya se ha dicho, se ha usado un punto de equilibrio. En este punto de equilibrio o *trim*, la fuerza de empuje es igual al peso. Igualando la ecuación (1.18) a cero, resulta:

$$\frac{F_z}{m} - g = 0 \rightarrow F_z = mg = C_T \cdot \left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \tag{1.29}$$

Asumiendo que todos los motores girasen a la misma velocidad de giro, la velocidad angular de equilibrio Ω_e resultaría:

$$mg = 4C_T \cdot \Omega_e^2 \rightarrow \boxed{\Omega_e = \sqrt{\frac{m \cdot g}{4 \cdot C_T}}} \tag{1.30}$$

Anexo D
Estadísticas del Laboratorio Virtual

En este anexo se incluyen algunas estadísticas con respecto al simulador desarrollado.

D. Anexo D - Estadísticas del laboratorio virtual

D.1. Introducción

En este anexo se quiere incluir toda la información relacionada con el código del simulador y unas estadísticas que proporcionen al lector una visión del trabajo llevado a cabo para obtener el simulador presentado en este proyecto. Para ello, se detallará el número de clases, funciones y líneas de código implementadas, así como ciertos tiempos de ejecución.

D.2. Estadísticas sobre las líneas de código

En la Tabla 5 se incluyen las estadísticas para el número de clases, funciones y líneas de código implementadas.

Elemento medido	Valor de la medida estadística
Número de ficheros de código	38
Número de ficheros gráficos	17
Número de clases implementadas	42
Número de funciones implementadas	429
Número de líneas de código	14.754
Número de líneas de código para la interfaz gráfica	3.200
Número de líneas de código totales	**17.954**

Tabla 5 Estadísticas de las líneas de código del simulador.

Como ya se describió en 6.1, el patrón de diseño usado para la implementación del código es MVC. La capa Vista contribuye al número de líneas totales con un 25.5% aproximadamente y con un 24.5% de las funciones implementadas. Por su parte, la capa Modelo contribuye con un 26.7% al número de líneas de código totales y con un 10% de las funciones implementadas. Estos números dan una idea del alto peso que tiene en el simulador toda la interacción con el usuario y el gran número de capacidades de este tipo del que se ha dotado al programa.

D.2.1. Tiempos de ejecución

En esta sección del anexo quiere ponerse de manifiesto el buen hacer o alto *performance* del simulador presentado en este proyecto. Para ello se muestran en la Tabla 6 los tiempos de ejecución de los principales procesos que el simulador lleva a cabo.

Proceso	Tiempo de ejecución [s]
Simulación de un fichero en Modo *Offline*	12.76*
Lectura de un fichero *mat (de 2MB)	0.03
Simulación *Offline* de 100s	127.36
Interpretación de un fichero *.mdl en el modo *Real-Time* (con un sistema de 3 capas en Simulink®)	0.07
Simulación de un fichero *.mdl con un controlador PID basado en eventos y	52.93

una simulación de 100s	
Escaneo de los puestos disponibles del hardware CrazyFlie 2.0	1.34
Conexión con el hardware CrazyFlie 2.0	0.03

Tabla 6 Tiempos de ejecución de los principales procesos del simulador.

*NOTA: hay que tener en cuenta que el tiempo medio de reacción del ser humano a un estímulo visual (la aparición de un diálogo, en este caso) es de 0.25s.

Lo más reseñable de la tabla anterior es que el bucle completo de la simulación *Real-Time* puede realizarse en poco menos de un minuto. Esto proporciona al investigador o estudiante una herramienta ideal para poder probar y analizar en simulación múltiples diseños de controlador realmente rápido.