Projecte Fi de Carrera

Enginyeria tècnica de Telecomunicació

MODELADO, SIMULACIÓN Y CONTROL DIGITAL DE AUTOGIROS

Daniel López García

Director: Asier Ibeas Hernández

Departament de Telecomunicació i d'Enginyeria de Sistemes

Escola d'Enginyeria
Universitat Autònoma de Barcelona (UAB)

Juliol 2013

El sotasignant, Asier Ibeas Hernández, Professor de l'Escola d'Enginyeria (EE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el projecte presentat en aquesta memòria de Projecte Fi de Carrera ha estat realitzat sota la seva direcció per l'alumne Daniel López García.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 28 de maig de 2013.

Signatura: Asier Ibeas Hernández

LICENCIA

(Creative Commons)

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

Se considera aeronave todo vehículo capaz de navegar por el aire y se dividen en dos categorías:

- Aerostatos: son más livianos que el aire y se sustentan debido al impulso estático del aire.
- Aerodinos: son más pesados que el aire y son capaces de generar sustentación.

La sustentación de los aerodinos puede generarse por alas fijas, como en el caso del avión, o por alas giratorias, como el helicóptero. El autogiro es un aerodino que está considerado entre el avión y el helicóptero por sus características, pese a que su apariencia es mucho más cercana a la del helicóptero como podemos observar en la figura 1. Como el avión, cuenta con un motor de propulsión, que lo hace avanzar horizontalmente, y con ala no conectada al motor, que es quien se encarga de la sustentación. La diferencia es que en el avión se trata de un ala fija y en el autogiro de ala rotatoria que gira a causa del viento que la atraviesa (no actuada). Los helicópteros también tienen ala rotatoria pero en este caso sí que está conectada al motor (actuada) y por tanto se encarga tanto de la propulsión como de la sustentación.

Figura 1. Autogiro en pleno vuelo.

Los aerodinos viajan habitualmente con un piloto a bordo, pero también podemos encontrarnos con los llamados vehículos aéreos no tripulados (VANT), UAV por sus siglas en inglés (*Unmanned Aerial Vechicle*), o sistema aéreo no tripulado, UAS (*Unmanned Aerial System*). Estos vehículos pueden ser pilotados remotamente desde

una estación terrestre o estar dotados de un sistema de control autónomo para seguir una trayectoria específica. Algunos de estos vehículos no tripulados, llamados OUAV (*Optional Unmanned Aerial Vechicle*), pueden permitir que, además, viaje un piloto a bordo.

La mayoría de UAV están basados en aviones y helicópteros. En cuanto a los autogiros, no hay presente en el mercado ningún UAV de este tipo ni investigaciones al respecto. El desarrollo de un vehículo no tripulado basado en autogiro sería, por tanto, algo innovador e interesante ya que presentaría algunas ventajas tanto económicas como funcionales respecto a otros tipos de vehículos.

En la figura 2 encontramos el esquema que engloba el desarrollo completo de un UAV. Este proyecto se centrará únicamente en los tres primeros niveles y se basará en el caso concreto de un autogiro que permita viajar a un piloto a bordo.

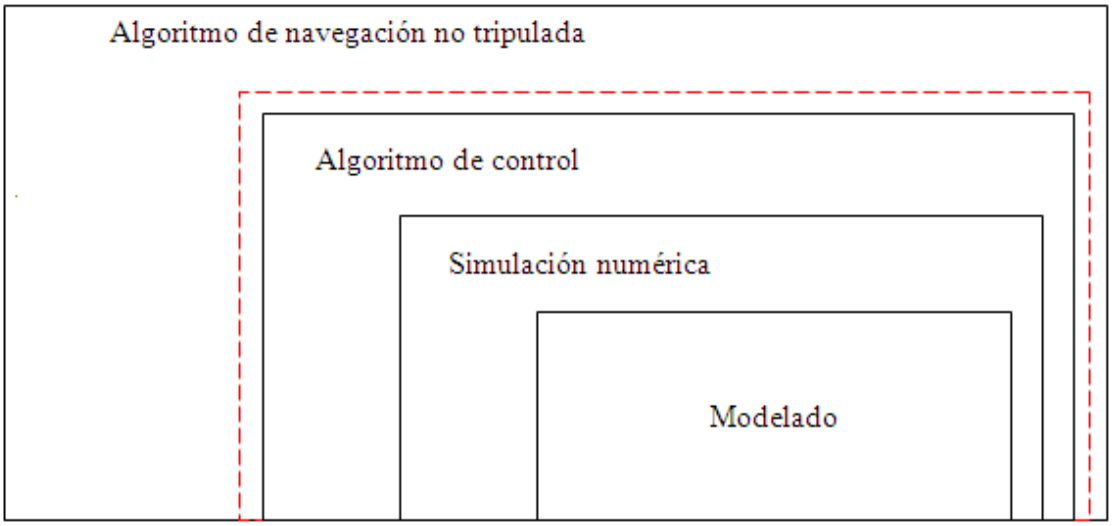


Figura 2. Esquema de desarrollo de un UAV.

1.1 Estructura del documento

El documento que sigue está estructurado de la siguiente forma:

- El capítulo 2 analiza la viabilidad del proyecto que se va a realizar.
- El capítulo 3 define el modelado del sistema.
- El capítulo 4 simula numéricamente el comportamiento.
- El capítulo 5 describe el control del vehículo.
- El capítulo 6 estudia la implementación del sistema en un microcrontrolador.
- El capítulo 7 recoge, finalmente, los aspectos más destacados del trabajo.

2. Estudio de Viabilidad

En este capítulo se analizará la viabilidad del desarrollo de un autogiro como vehículo aéreo no tripulado. Para ello se estudiarán los objetivos del proyecto y las ventajas y desventajas que nos va a suponer contar con este tipo concreto de aeronave para llevar a cabo las misiones que suelen realizar los UAV. A continuación, se hará un repaso del estado actual de las aeronaves no tripuladas. Finalmente, se realizará la planificación estimada del proyecto a partir de las tareas que se deben realizar y de los recursos disponibles.

2.1 Objetivos del proyecto

Los objetivos que se deberán cumplir durante la realización del proyecto son los siguientes:

- **Modelado:** definir el modelo conceptual en base a las ecuaciones dinámicas que describen el comportamiento del vehículo.
- **Simulación:** simulación numérica de su dinámica, mediante el cálculo científico, con la ayuda de un entorno de trabajo adecuado como Matlab.
- **Control:** diseño de un algoritmo sencillo que cumpla con ciertas especificaciones y permita a la aeronave comportarse según los parámetros de referencia fijados de antemano por un operador.
- **Implementación:** mediante un software como MPLAB, desarrollar el código de un microcontrolador.

2.2 Ventajas e inconvenientes del autogiro

El autogiro ofrece una serie de ventajas respecto a otras aeronaves. Entre estas ventajas destacan:

- Vuelo a velocidades muy bajas. Dependiendo de la potencia y la carga del motor puede llegar a tener una velocidad mínima inferior a 15 Km/h.
- Tolera ráfagas de viento, superiores a los 80 Km/h, que no permiten volar a ciertos aviones.
- Gran maniobrabilidad. Es la aeronave más ágil del mercado.
- Posibilidad de aterrizar y despegar en espacios reducidos. El aterrizaje se realiza de forma vertical y la carrera de despegue es muy corta, aproximadamente de 20 metros con condiciones favorables.
- Inmunidad ante las turbulencias y las corrientes térmicas producidas por efectos del calor.

- Seguridad ante un fallo del motor. En caso de avería del motor en vuelo, el vehículo planea y desciende lentamente sin perder el control en ningún momento.
- No entra en pérdida y por lo tanto no puede entrar en barrena.
- Facilidad de pilotaje. Su control es muy intuitivo y no requiere una instrucción tan exhaustiva como la que se necesita para manejar un helicóptero.
- Precio inicial y costos de mantenimiento muy inferiores a los de aviones y helicópteros.

A pesar de todas estas ventajas tenemos que tener en cuenta que el autogiro tiene las siguientes limitaciones:

- No alcanza velocidades tan altas como las del avión.
- No es capaz de sustentarse en un punto fijo (*hovering* en inglés) como el helicóptero.

2.2.1. Aplicaciones

Su principal aplicación sería llevar a cabo misiones que resulten peligrosas o pesadas para el piloto como, por ejemplo:

- Labores de extinción de incendios. Puede ayudar a conocer la extensión y ubicación exacta del fuego en montañas y bosques.
- Reconocimiento de campos de batalla. Esta aeronave está muy indicada para este tipo de misiones debido a su alta defensa por su gran maniobrabilidad.
- Tareas de búsqueda y rescate. Si se equipa con una cámara infrarroja puede ser utilizado para localizar a personas desaparecidas.
- Detección de icebergs en alta mar. Los barcos que tengan el espacio suficiente para transportarlo pueden utilizarlo para la predicción de icebergs en un área próxima. Sería de gran utilidad en el caso de icebergs pequeños e indetectables por satélites y radares pero que pueden tener serias consecuencias en caso de impacto.
- Reconocimiento de riesgo químico. Contando con los sensores adecuados podría evaluar el estado de zonas con ambiente de alta toxicidad química o radiactiva.
- Vigilancia. Se puede encargar de misiones de observación en zonas de acceso restringido y fronteras.

2.3 Estado del arte

El origen y desarrollo de este tipo de vehículos fue, como en la mayoría de avances tecnológicos, para fines militares. Los primeros UAV aparecieron tras finalizar la Primera Guerra Mundial y se utilizaron durante la Segunda Guerra Mundial. En los últimos años han adquirido una gran importancia dada su gran versatilidad para llevar a cabo tanto misiones militares como civiles.

Hasta la fecha se han fabricado prototipos aislados de UAV basados en girodinos, en 2003 y 2006, y autogiros, en 1963, pero desde entonces no se ha llevado a cabo ninguna investigación para el desarrollo de vehículos de este tipo. En el mercado actual la mayoría UAV están basados en aviones y helicópteros. A modo de ejemplo tenemos la tabla 2, donde se muestran los últimos modelos desarrollados por las empresas más importantes en el campo de los UAV.

EMPRESA	NOMBRE DEL MODELO	TIPO DE VEHÍCULO
Boeing	ScanEagle	Avión
Boeing	A160 Hummingbird	Helicóptero
Boeing	X-50A Dragonfly	Avión
Thales	Watchkeeper WK450	Avión
Kamov	MBVK-137	Helicóptero
INTA	ALO	Avión
Northrop Grumman	RQ-4 Global Hawk	Avión
Northrop Grumman	MQ-8 Fire Scout	Helicóptero
Lockheed	RQ-3 DarkStar	Avión
Lockheed	RQ-170 Sentinel	Avión
General Atomics Aeronautical Systems	MQ-1C Gray Eagle UAS	Avión
General Atomics Aeronautical Systems	Avenger	Avión
BAE Systems	Skylynx II	Avión
BAE Systems	HERTI	Avión
ZALA Aero	ZALA 421-12	Avión
ZALA Aero	ZALA 421-06	Helicóptero
CASSIDIAN	ATLANTE	Avión
Tekplus	Centauro	Helicóptero
INDRA	Pelícano	Helicóptero

Tabla 1. Lista de vehículos aéreos no tripulados.

2.4 Organización del proyecto

Antes de comenzar con el proyecto identificaremos cuales serán las tareas que se van a realizar y se estimaran los plazos correspondientes para cada una de ellas en función de los recursos disponibles. Además, se van a prever los posibles contratiempos con los que nos podemos encontrar para cumplir los plazos. De este modo podremos realizar un seguimiento del proyecto y sabremos en todo momento si se está trabajando con retraso o adelanto.

2.4.2. Recursos y tareas a realizar

Para la realización del proyecto será necesario contar con un ordenador con Matlab y MPLAB instalados y conexión a Internet, además de la propia dedicación personal. Con estos requisitos se podrán llevar a cabo todas las tareas necesarias, que están descritas en la tabla 2.

DESCRIPCIÓN DE LA ACTIVIDAD	TAREA	TAREA PRECEDENTE	TIEMPO ESTIMADO (HORAS)
Búsqueda bibliográfica	1	Ninguna	5
Estudio de viabilidad	2	1	10
Estudio y entendimiento de las ecuaciones dinámicas	3	Ninguna	55
Simulación	4	3	25
Entendimiento del control	5	4	55
Diseño del algoritmo	6	5	75
Pruebas del algoritmo	7	6	15
Implementación	8	7	30

Tabla 2. Listado de tareas del proyecto.

2.4.3. Planificación

Una vez hemos definido cuáles son las tareas que vamos a realizar y el tiempo que va a consumir cada una de ellas podemos planificar la duración total del proyecto y representar gráficamente el tiempo dedicado mediante el diagrama de Gantt de la figura 3.

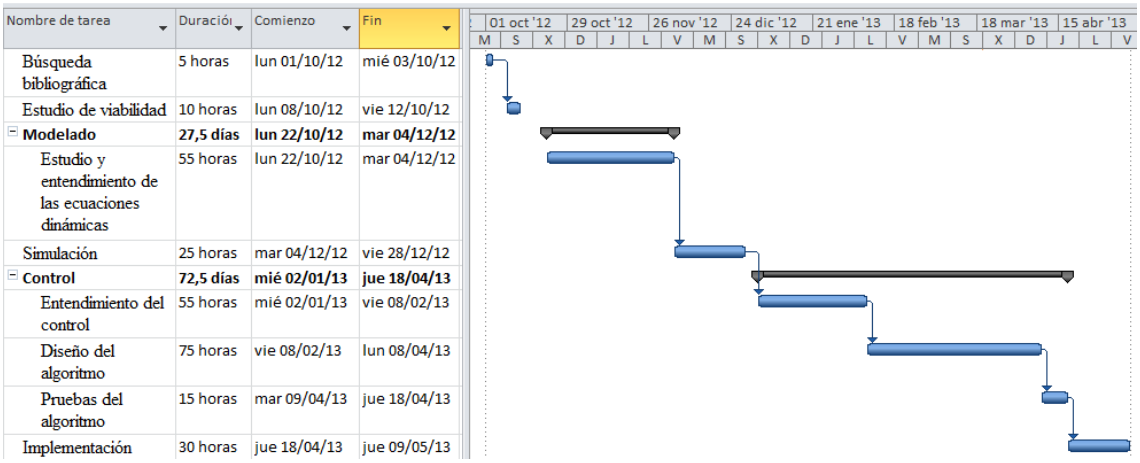


Figura 3. Diagrama de Gantt del proyecto.

2.4.4. Análisis de riesgo y plan de contingencia

Para asegurar la finalización a tiempo de este proyecto se realizará un plan de contingencia para minimizar el impacto de los contratiempos que puedan surgir durante el periodo de trabajo. Los riesgos que pueden comprometer su finalización están detallados en la tabla 3 junto a la probabilidad de que ocurra cada imprevisto y las contramedidas más adecuadas para cada uno de ellos.

RIESGO	PROBABILIDAD	CONTRAMEDIDA
Problemas de salud	Media	Trabajar con cierto adelanto respecto al calendario planificado
Pérdida de datos del trabajo realizado	Media	Realizar una copia de seguridad de todo el proyecto de forma rutinaria
No tener acceso a Matlab o MPLAB	Baja	Contar con software libre alternativo que tenga las mismas prestaciones
Encontrar trabajo	Baja	Reservar días de la semana para el proyecto para seguir al mismo ritmo
Avería del ordenador	Media	Contar con un segundo equipo en todo momento

Tabla 3. Plan de contingencia.

2.5 Conclusiones

El motivo de este estudio ha sido analizar la viabilidad del desarrollo de un autogiro no tripulado. Una vez consideradas las ventajas e inconvenientes de esta aeronave podemos afirmar que podría llevar a cabo todas las misiones civiles y militares que realizan los helicópteros, salvo aquellas en las que se precisa el vuelo estacionario, a un coste mucho menor. Además, podría realizar tareas que no sean viables para un avión, como las que requieran vuelo a baja velocidad o aterrizaje y despegue en espacios reducidos. A estos dos aspectos debemos añadir que no existe actualmente este producto en el mercado ni se estén realizando investigaciones al respecto, lo que aumenta el interés de este proyecto. Así pues, podemos concluir que sí que tendría sentido el desarrollo de un vehículo no tripulado de este tipo.

3. Modelado

El objetivo de este capítulo es estudiar el movimiento del autogiro, y para ello se debe obtener el modelo dinámico.

3.1 Análisis de la dinámica

La dinámica relaciona las fuerzas que actúan en un cuerpo y el movimiento que en él se genera. Así, el modelo dinámico del autogiro tiene como objetivo conocer la relación entre su movimiento y las fuerzas implicadas en el mismo. Este modelo relaciona matemáticamente:

1. La localización del vehículo definida por las coordenadas de posición y sus derivadas: velocidad y aceleración.
2. Las fuerzas y pares aplicados en el cuerpo.
3. Los parámetros físicos del autogiro, como dimensión, masa e inercias.

El modelo dinámico de un mecanismo se puede desarrollar a partir de las ecuaciones Newton-Euler. Estas ecuaciones tienen en cuenta la segunda ley de Newton, que plantea el equilibrio de fuerzas, y el teorema de rotación de Euler [1]:

$$\sum F = m\dot{v}\,, \qquad \sum \tau = I\dot{\omega} + \omega \times (I\,\omega) \tag{1}$$

Este método cuenta con una gran eficiencia computacional, pero presenta ecuaciones poco estructuradas y resulta muy complejo cuando estamos tratando con un mecanismo con varios grados de libertad. En nuestro caso, tenemos un vehículo con seis grados de libertad (arriba/abajo, derecha/izquierda y adelante/atrás) y, por lo tanto, debemos buscar una alternativa.

3.1.1. Obtención del modelo dinámico mediante la formulación de Lagrange

El método más adecuado a nuestras necesidades es la formulación lagrangiana, que se basa en fundamentos energéticos. Para poder escribir las ecuaciones dinámicas mediante este método comenzaremos definiendo la Lagrangiana, L, como la diferencia entre la energía cinética y la energía potencial del sistema [2-3]:

$$L(q, \dot{q}) = T - \mathrm{U} \tag{2}$$

Donde:

- L es la función lagrangiana.
- T es la energía cinética.
- U es la energía potencial.
- q es la coordenada generalizada.
- $\dot{q}$ es la velocidad generalizada.

Un cuerpo rígido en el espacio tiene las siguientes coordenadas generalizadas:

$$q = [\ \boldsymbol{\varepsilon}\ \eta\ ] = [\ x\ y\ z\ \ \Phi\ \theta\ \Psi\ ]^T \tag{3}$$

Donde:

- $\boldsymbol{\varepsilon}$ = [x, y, z] es la posición del centro de masa del autogiro.
- $\eta = [\Phi, \theta, \Psi]$ son los ángulos de Tait-Bryan: inclinación o cabeceo (pitch), alabeo (roll) y guiñada o deriva (yaw).

El Lagrangiano para el autogiro, separando la energía cinética en traslacional y rotacional, viene dado por:

$$L(q, \dot{q}) = T_{tras} + T_{rot} - U \tag{4}$$

Aplicando la definición de energías:

$$L(q, \dot{q}) = \frac{m}{2}\ \dot{\boldsymbol{\varepsilon}}^T\ \dot{\boldsymbol{\varepsilon}} + \frac{1}{2}\ \omega^T J\ \omega - mgz \tag{5}$$

Donde:

- m es la masa total del vehículo.
- $\dot{\boldsymbol{\varepsilon}}$ es el vector de velocidad lineal de longitud 3.
- ω es el vector de velocidad angular de longitud 3.
- J es el tensor de inercia de tamaño 3x3.
- g es la aceleración gravitacional.
- z es la altura del centro de masa del vehículo.

El vector de la velocidad angular *ω* respecto a los ejes de coordenadas se relaciona con las velocidades generalizadas ή utilizando una relación cinemática [4]:

$$\dot{\eta}\ =\ W_\eta^{-1}\omega\ , \tag{6}$$

donde:

$$W_\eta = \begin{bmatrix} -\sin\theta & 0 & 1 \\ \cos\theta \sin\Psi & \cos\Psi & 0 \\ \cos\theta \cos\Psi & -\sin\Psi & 0 \end{bmatrix} \tag{7}$$

Definiendo:

$$\mathrm{M} \;=\; \mathrm{M}(\eta) = \mathrm{W}_{\eta}^{\mathrm{T}}\, \mathrm{J}\, W_{\eta}\,, \tag{8}$$

tal que:

$$\mathrm{T}_{\mathrm{rot}} \;=\; \frac{1}{2}\; \dot{\eta}^{T}\; M\; \dot{\eta} \tag{9}$$

Donde M es el tensor de inercia para la energía cinética total rotacional del autogiro, expresada en coordenadas generalizadas.

Una vez definida la Lagrangiana podemos escribir el modelo dinámico completo del autogiro a partir de las llamadas ecuaciones de Euler-Lagrange:

$$\frac{d}{dt}\frac{\partial \mathrm{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathrm{L}}{\partial \mathrm{q}} = \begin{bmatrix} F_{\boldsymbol{\varepsilon}} \\ \tau \end{bmatrix} \tag{10}$$

Donde F_{ε} es la fuerza traslacional aplicada al vehículo y τ representa los momentos de alabeo, cabeceo y guiñada. Dado que el Lagrangiano no contiene términos en la energía cinética que combinen $\dot{\boldsymbol{\varepsilon}}$ con $\dot{\eta}$, las ecuaciones de Euler-Lagrange se pueden dividir en la dinámica traslacional y rotacional.

La ecuación para el movimiento de traslación es:

$$\frac{d}{dt}\frac{\partial \mathrm{L}_{\mathrm{tras}}}{\partial \dot{\boldsymbol{\varepsilon}}} - \frac{\partial \mathrm{L}_{\mathrm{tras}}}{\partial \boldsymbol{\varepsilon}} = F_{\boldsymbol{\varepsilon}}\;, \tag{11}$$

de donde se obtiene:

$$m\ddot{\boldsymbol{\varepsilon}} + mgE_{z} = F_{\boldsymbol{\varepsilon}} \tag{12}$$

Reescribiendo (12) en función de las componentes traslacionales del vector de estados tenemos:

$$\begin{cases} \ddot{x} = \dfrac{1}{m}\,(cos\Psi \mathrm{sin}\theta \mathrm{cos}\Phi + sin\Psi \mathrm{sin}\Phi)\; Q + \dfrac{1}{m}\; cos\Psi\; \mathrm{U1} + \dfrac{A_x}{m} \\ \ddot{y} = \dfrac{1}{m}(sin\Psi \mathrm{sin}\theta \mathrm{cos}\Phi - cos\Psi \mathrm{sin}\Phi)\; Q + \dfrac{1}{m}\; sin\Psi\; \mathrm{U1} + \dfrac{A_y}{m} \\ \ddot{z} = -g + \dfrac{1}{m}(cos\theta cos\Phi)\; Q + \dfrac{A_z}{m} \end{cases} \tag{13}$$

Donde A_{x}, A_{y} y A_{z} son fuerzas aerodinámicas que actúan sobre la aeronave en cada eje, U1 es la fuerza de entrada de control aplicada, que proporciona la propulsión horizontal al vehículo, y Q es el par rotor. En la mayoría de aeronaves de ala giratoria se considera constante la velocidad del rotor debido a que está conectado a un motor controlado por

realimentación. En el caso del autogiro, el rotor gira a una velocidad determinada por la cantidad de aire que lo atraviesa y, por tanto, es necesario tener en cuenta este grado de libertad. La ecuación se describe:

$$j\,\dot{\Omega} = -Q\ , \tag{14}$$

donde Ω es la velocidad de rotación de las palas y j es la inercia de la rotación del rotor, que se aproxima a partir de la inercia de aleteo de la pala I_β :

$$j = N_b I_\beta\ , \tag{15}$$

donde N_b es el número de palas.

Para el movimiento rotacional se escribe:

$$\frac{d}{dt}\frac{\partial \mathrm{L_{rot}}}{\partial \dot{\eta}} - \frac{\partial \mathrm{L_{rot}}}{\partial \eta} = \tau\ , \tag{16}$$

o:

$$\frac{d}{dt}\left[\dot{\eta}^T\, M \frac{\partial \dot{\eta}}{\partial \dot{\eta}}\right] - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T\, M\, \dot{\eta}) = \tau\ , \tag{17}$$

derivando, se tiene:

$$M\ddot{\eta} + \dot{M}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T\, M\, \dot{\eta}) = \tau \tag{18}$$

Definiendo el vector de Coriolis como:

$$\bar{V}(\eta, \dot{\eta}) = \dot{M}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}\left(\dot{\eta}^T\, M\, \dot{\eta}\right), \tag{19}$$

sustituyendo:

$$M\ddot{\eta} + \bar{V}(\eta, \dot{\eta}) = \tau\ , \tag{20}$$

y puesto que $\bar{V}(\eta, \dot{\eta})$ puede escribirse como:

$$\bar{V}(\eta, \dot{\eta}) = (\dot{M} - \frac{1}{2}\frac{\partial}{\partial \eta}\left(\dot{\eta}^T\, M\,\right))\dot{\eta} = C(\eta, \dot{\eta})\dot{\eta} \tag{21}$$

Donde $C(\eta, \dot{\eta})$ son los términos de Coriolis que contienen efectos centrífugos y giroscópicos relacionados con η.

Reescribiendo el modelo rotacional de la dinámica:

$$M(\eta)\,\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} = \tau \tag{22}$$

$Con\ \tau$:

$$\tau = \begin{bmatrix} \tau_{\Phi} \\ \tau_{\theta} \\ \tau_{\Psi} \end{bmatrix}, \tag{23}$$

donde:

$$\tau_{\Phi} = -I_{xx}(\ddot{\Phi}\, sin\theta + \dot{\Phi}\dot{\theta}\, cos\theta - \ddot{\Psi} sin^2\theta - 2\dot{\Psi}\dot{\theta} sin\theta\, cos\theta)$$
$$+ I_{yy}(\ddot{\theta}\, cos\theta\, sin\Phi\, cos\Phi - \dot{\theta}^2\, sin\theta\, sin\Phi cos\Phi - \dot{\theta}\dot{\Phi} cos\theta sin^2\Phi$$
$$+\dot{\theta}\dot{\Phi} cos\theta cos^2\Phi + \ddot{\Psi} cos^2\theta sin^2\Phi$$
$$-2\dot{\Psi}\dot{\theta}\, sin\theta\, cos\theta sin^2\Phi + 2\dot{\Psi}\dot{\Phi}\, cos^2\theta\, sin\Phi\, cos\Phi) + I_{zz}(\ddot{\Psi} cos^2\theta cos^2\Phi$$
$$-2\dot{\Psi}\dot{\theta}\, sin\theta\, cos\theta cos^2\Phi - 2\dot{\Psi}\dot{\Phi}\, cos^2\, \theta sin\Phi\, cos\Phi - \ddot{\theta}\, cos\theta\, sin\Phi\, cos\Phi$$
$$+\dot{\theta}^2\, sin\theta\, sin\Phi cos\Phi + \dot{\theta}\dot{\Phi} cos\theta sin^2\Phi - \dot{\theta}\dot{\Phi} cos\theta cos^2\Phi)$$

$$\tau_{\theta} = I_{xx}(\dot{\Psi}\dot{\theta}\, cos\theta - \dot{\Psi}^2\, sin\theta cos\theta) + I_{yy}(\ddot{\theta} cos^2\Phi - 2\dot{\theta}\dot{\Phi}\, sin\Phi\, cos\Phi$$
$$+\ddot{\Psi}\, cos\theta sin\Phi\, cos\Phi + \dot{\Psi}\dot{\theta}\, cos\theta cos^2\Phi - \dot{\Psi}\dot{\theta}\, cos\theta sin^2\Phi + \dot{\Psi}^2\, sin\theta cos\theta sin^2\Phi)$$
$$-I_{zz}(\ddot{\Psi}\, cos\theta sin\Phi\, cos\Phi - \dot{\Psi}^2\, sin\theta cos\theta cos^2\Phi - \dot{\Psi}\dot{\Phi}\, cos\theta sin^2\Phi$$
$$+\dot{\Psi}\dot{\Phi}\, cos\theta cos^2\Phi - \ddot{\theta} sin^2\Phi - 2\dot{\theta}\dot{\Phi}\, sin\Phi\, cos\Phi)$$

$$\tau_{\Psi} = I_{xx}\left(\ddot{\theta} - \ddot{\Psi} sin\theta - \dot{\Psi}\dot{\theta} cos\theta\right) - I_{yy}(-\dot{\theta}^2\, sin\Phi\, cos\Phi - \dot{\Psi}\dot{\theta}\, cos\theta sin^2\Phi$$
$$+\dot{\Psi}\dot{\theta}\, cos\theta cos^2\Phi + \dot{\Psi}^2 cos^2\theta\, sin\Phi\, cos\Phi) - I_{zz}(-\dot{\Psi}^2 cos^2\theta\, sin\Phi\, cos\Phi$$
$$+\dot{\Psi}\dot{\theta}\, cos\theta sin^2\Phi - \dot{\Psi}\dot{\theta}\, cos\theta cos^2\Phi + \dot{\theta}^2\, sin\Phi\, cos\Phi)$$

$M(\eta)$:

$$\begin{bmatrix} I_{xx} & 0 & -I_{xx}\, sin\theta \\ 0 & I_{yy} cos^2\Phi + I_{zz} sin^2\Phi & \left(I_{yy} - I_{zz}\right) \cos\Phi \sin\Phi \cos\theta \\ -I_{xx}\, sin\theta & \left(I_{yy} - I_{zz}\right) \cos\Phi \sin\Phi \cos\theta & I_{xx} sin^2\theta + I_{yy} sin^2\Phi cos^2\theta + I_{zz} cos^2\Phi cos^2\theta \end{bmatrix} \tag{24}$$

Siendo Ixx, Iyy y Izz los momentos de inercias de alabeo, cabeceo y guiñada respectivamente.

Y $C(\eta, \dot{\eta})$:

$$C(\eta, \dot{\eta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{25}$$

donde:

$c_{11} = 0$

$c_{12} = \left(I_{yy} - I_{zz}\right)(\dot{\theta}\cos\Phi\sin\Phi + \dot{\Psi}\sin^2\Phi\cos\theta) + \left(I_{zz} - I_{yy}\right)\dot{\Psi}\cos^2\Phi\cos\theta - I_{xx}\dot{\Psi}\cos\theta$

$c_{13} = \left(I_{zz} - I_{yy}\right)\dot{\Psi}\cos\Phi\sin\Phi\cos^2\theta$

$c_{21} = \left(I_{zz} - I_{yy}\right)(\dot{\theta}\cos\Phi\sin\Phi + \dot{\Psi}\sin^2\Phi\cos\theta) + \left(I_{yy} - I_{zz}\right)\dot{\Psi}\cos^2\Phi\cos\theta - I_{xx}\dot{\Psi}\cos\theta$

$c_{22} = \left(I_{zz} - I_{yy}\right)\dot{\Phi}\cos\Phi\sin\Phi$

$c_{23} = -I_{xx}\dot{\Psi}\sin\theta\cos\theta + Iyy\,\dot{\Psi}\sin^2\Phi\cos\theta\sin\theta + Izz\dot{\Psi}\cos^2\Phi\sin\theta\cos\theta$

$c_{31} = \left(I_{yy} - I_{zz}\right)\dot{\Psi}\cos^2\theta\sin\Phi\cos\Phi - I_{xx}\dot{\theta}\cos\theta$

$c_{32} = \left(I_{zz} - I_{yy}\right)(\dot{\theta}\cos\Phi\sin\Phi\sin\theta + \dot{\Phi}\sin^2\Phi\cos\theta) + \left(I_{yy} - I_{zz}\right)\dot{\theta}\cos^2\Phi\cos\theta$

$+ I_{xx}\dot{\Psi}\sin\theta\cos\theta - Iyy\,\dot{\Psi}\sin^2\Phi\sin\theta\cos\theta - Izz\dot{\Psi}\cos^2\Phi\sin\theta\cos\theta$

$c_{33} = \left(I_{yy} - I_{zz}\right)\dot{\Phi}\cos\Phi\sin\Phi\cos^2\theta - Iyy\dot{\theta}\sin^2\Phi\cos\theta\sin\theta - Izz\,\dot{\theta}\cos^2\Phi\cos\theta\sin\theta$

$+ I_{xx}\,\dot{\theta}\cos\theta\sin\theta$

Reescribiendo (20) en función de las componentes rotacionales del vector de estados tenemos:

$$\begin{cases} \ddot{\Phi} = \dfrac{(I_{yy} - I_{zz})}{I_{xx}}\,\dot{\theta}\dot{\Psi} - \dfrac{J_R\,\Omega}{I_{xx}}\,\dot{\theta} + \dfrac{l}{I_{xx}}U2 + \dfrac{A_p}{I_{xx}} \\ \ddot{\theta} = \dfrac{(I_{zz} - I_{xx})}{I_{yy}}\,\dot{\Phi}\dot{\Psi} + \dfrac{J_R\,\Omega}{I_{yy}}\,\dot{\Phi} + \dfrac{l}{I_{yy}}U3 + \dfrac{A_q}{I_{yy}} \\ \ddot{\Psi} = \dfrac{(I_{xx} - I_{yy})}{I_{zz}}\,\dot{\Phi}\dot{\theta} + \dfrac{1}{I_{zz}}U4 + \dfrac{A_r}{I_{zz}} \end{cases} \tag{26}$$

Donde Ap, Aq y Ar son fuerzas aerodinámicas que actúan sobre la aeronave, J_R el momento de inercia del rotor sobre su eje y U2, U3 y U4 las fuerzas de entrada de control aplicadas.

3.2 Conclusiones del capítulo

En este capítulo se ha estudiado el modelo dinámico que describe el movimiento del autogiro a partir de la formulación de Lagrange. Para su obtención se ha tenido en cuenta la relación existente entre las fuerzas implicadas, los parámetros físicos de la aeronave y el movimiento generado. De esta manera, se han definido las ecuaciones referentes al desplazamiento traslacional y rotacional.

La obtención de estas ecuaciones nos permitirá simular el vuelo del vehículo mediante software y observar su respuesta ante diferentes fuerzas de control aplicadas y la posible presencia de fuerzas aerodinámicas externas.

4. Simulación

En este capítulo vamos a realizar la simulación numérica del vehículo en un entorno de cálculo científico como Matlab para poder comprobar su comportamiento en funcionamiento.

4.1 Ecuaciones

Las ecuaciones que se van a procesar son las obtenidas en (13) y (26). La resolución de este sistema nos va a permitir realizar un seguimiento de las posiciones y velocidades tanto traslacionales como rotacionales. Hay varios métodos que permiten analizar ecuaciones diferenciales de este tipo, en nuestro caso optamos por los métodos de Runge-Kutta, concretamente el de cuarto orden, que se basa en la aproximación de soluciones de ecuaciones diferenciales ordinarias de primer orden.

4.1.1. Método de Runge-Kutta de cuarto orden

Para poder utilizar este método necesitamos que las ecuaciones diferenciales sean de primer orden, así que necesitamos añadir ecuaciones adicionales al sistema para que cumpla estos requisitos:

$$
\begin{cases}
\frac{d\dot{x}}{dt} = \frac{1}{m}\left(cos\Psi \sin\theta \cos\Phi + sin\Psi \sin\Phi\right) Q + \frac{1}{m}\, cos\Psi\, U1 + \frac{A_x}{m} \\
\frac{d\dot{y}}{dt} = \frac{1}{m}\left(sin\Psi \sin\theta \cos\Phi - cos\Psi \sin\Phi\right) Q + \frac{1}{m}\, sin\Psi\, U1 + \frac{A_y}{m} \\
\frac{d\dot{z}}{dt} = -g + \frac{1}{m}\left(cos\theta cos\Phi\right) Q + \frac{A_x}{m} \\
\frac{dx}{dt} = \dot{x} \\
\frac{dy}{dt} = \dot{y} \\
\frac{dz}{dt} = \dot{z} \\
\frac{d\dot{\Phi}}{dt} = \frac{(I_{yy} - I_{zz})}{I_{xx}}\, \dot{\theta}\dot{\Psi} - \frac{J_R\, \Omega}{I_{xx}}\, \dot{\theta} + \frac{l}{I_{xx}} U2 + \frac{A_p}{I_{xx}} \\
\frac{d\dot{\theta}}{dt} = \frac{(I_{zz} - I_{xx})}{I_{yy}}\, \dot{\Phi}\dot{\Psi} + \frac{J_R\, \Omega}{I_{yy}}\, \dot{\Phi} + \frac{l}{I_{yy}} U3 + \frac{A_q}{I_{yy}} \\
\frac{d\dot{\Psi}}{dt} = \frac{\left(I_{xx} - I_{yy}\right)}{I_{zz}}\, \dot{\Phi}\dot{\theta} + \frac{1}{I_{zz}} U4 + \frac{A_r}{I_{zz}} \\
\frac{d\Phi}{dt} = \dot{\Phi} + \dot{\theta} \sin\Phi \tan\theta + \dot{\Psi} \cos\Phi \tan\theta \\
\frac{d\theta}{dt} = \dot{\theta} \cos\Phi - \dot{\Psi} \sin\Phi \\
\frac{d\Psi}{dt} = \dot{\theta} \sin\Phi \sec\theta + \dot{\Psi} \cos\Phi \sec\theta
\end{cases}
\tag{27}
$$

Ahora que ya tenemos las ecuaciones a resolver, definimos un problema de valor inicial:

$$x' = f(t,x), \qquad x(x_0) = x_0 \tag{28}$$

Entonces el método Runge-Kutta de cuarto orden para el problema está dado por la ecuación [5]:

$$x_{i+1} = x_i + \frac{1}{6}(\text{k1} + 2\text{k2} + 2\text{k3} + \text{k4})\,\text{h} \tag{29}$$

Donde:

k1= $f(t_i, x_i)$

k2= $f(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k1\ h)$

k3= $f(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k2\ h)$

k4= $f(t_i + h, x_i + k3\ h)$

h= t_{i+1} - t_i

El código que se va a encargar de realizar esta tarea en Matlab es el que se muestra en la figura 4.

```
tspan=0:0.05:60;
N=length(tspan);
w=x0;
for i=1:N-1
h=tspan(i+1)-tspan(i);
t=tspan(i);
K1=h*feval(@sistemadeecuaciones,t,w);
K2=h*feval(@sistemadeecuaciones,t+h/2,w+K1/2);
K3=h*feval(@sistemadeecuaciones,t+h/2,w+K2/2);
K4=h*feval(@sistemadeecuaciones,t+h,w+K3);
w=w+(K1+2*K2+2*K3+K4)/6;
x(:,i+1)=w;
end
```

Figura 4. Código del método Runge-Kutta de cuarto orden.

De esta manera se va a realizar un bucle de N repeticiones, que es la longitud del vector de tiempo *tspan* que hemos generado. En el primer ciclo se va a evaluar el sistema de doce ecuaciones, que está almacenado en un archivo llamado *sistemadeecuaciones*, para las condiciones iniciales definidas en x_0. Para cada ciclo posterior se va a evaluar el sistema en el instante de tiempo correspondiente a partir del estado anterior. Al mismo tiempo, se irá generando la matriz de datos *x* de dimensiones 12xN a partir de la cual se podrán graficar los resultados.

4.2 Diagrama de la simulación

Observando las ecuaciones que describen la dinámica del autogiro, podemos comprobar que el movimiento del cuerpo es un movimiento de rototraslación en el espacio, es decir, su dinámica se puede representar como la combinación de un movimiento traslacional de su centro de masa y una rotación alrededor del eje que pasa por el centro de masa. Así, el sistema se puede dividir, tal como muestra la figura 5, en dos subsistemas: el subsistema de traslación, que depende de la entrada de control U1 y de los ángulos de orientación del vehículo, y el subsistema de rotación, que está afectado por las entradas de control U2, U3 y U4.

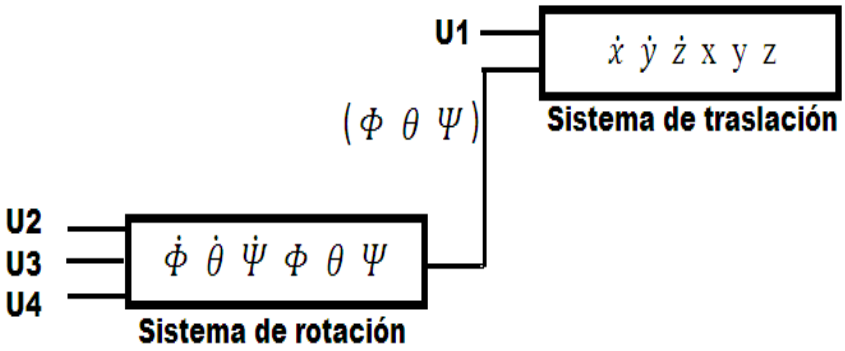


Figura 5. Diagrama de la simulación.

4.3 Resultados de la simulación

Para realizar la simulación se ha considerado que se trata de un modelo de autogiro VPM M16 [6]. Los parámetros utilizados son: m = 450 kg, l = 4.16 m, g = 9.81 m/s^2, J_R = 46 kg/ m^2, I_{xx} = 195 kg/m^2, I_{yy} = 637 kg/m^2 y I_{zz} = 4425 kg/m^2.

Se realizan dos simulaciones para poder evaluar el procedimiento propuesto. En la primera simulación se considera que el autogiro comienza en la posición (x,y,z) = (0, 0, 300) m y con orientación $(\Phi, \theta, \Psi) = (\frac{\pi}{8}, \frac{\pi}{8}, \frac{\pi}{4})$ rad. En t = 25s se deja de aplicar potencia al motor propulsor del vehículo, y en t = 30s se generan dos perturbaciones externas, que son escalones en los momentos aerodinámicos A_x y A_y, de 50 Nm cada una. Los resultados de esta simulación se presentan en las figura 6, donde se muestra la altura y la posición respecto al horizonte al finalizar la simulación utilizando el programa diseñado por Élodie Roux [7], y en las figuras 7 y 8.

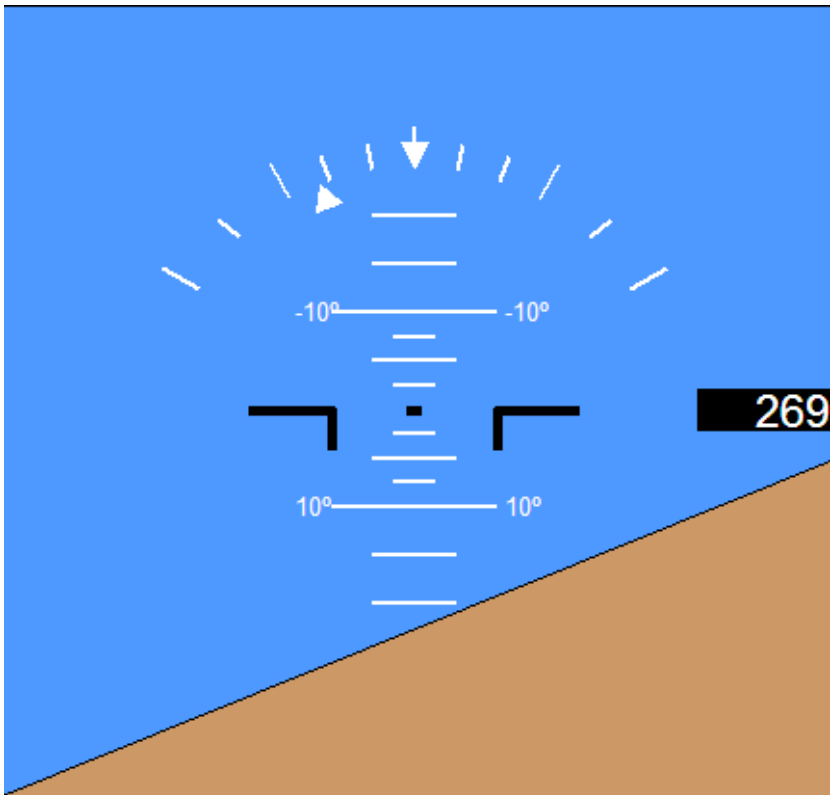


Figura 6. Altura y posición respecto al horizonte tras la primera simulación.

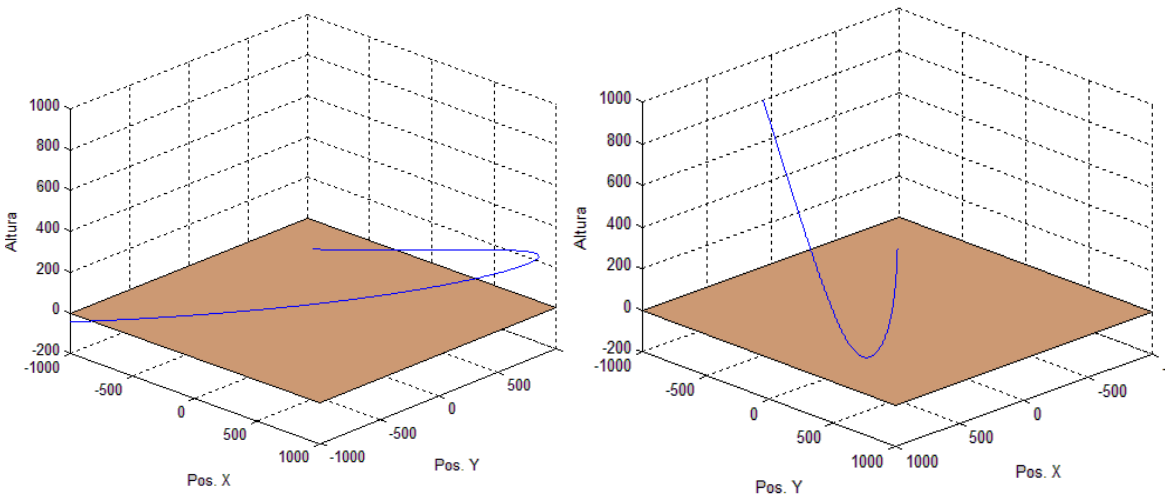


Figura 7. Trayectoria xyz durante la primera simulación desde dos ángulos de vista.

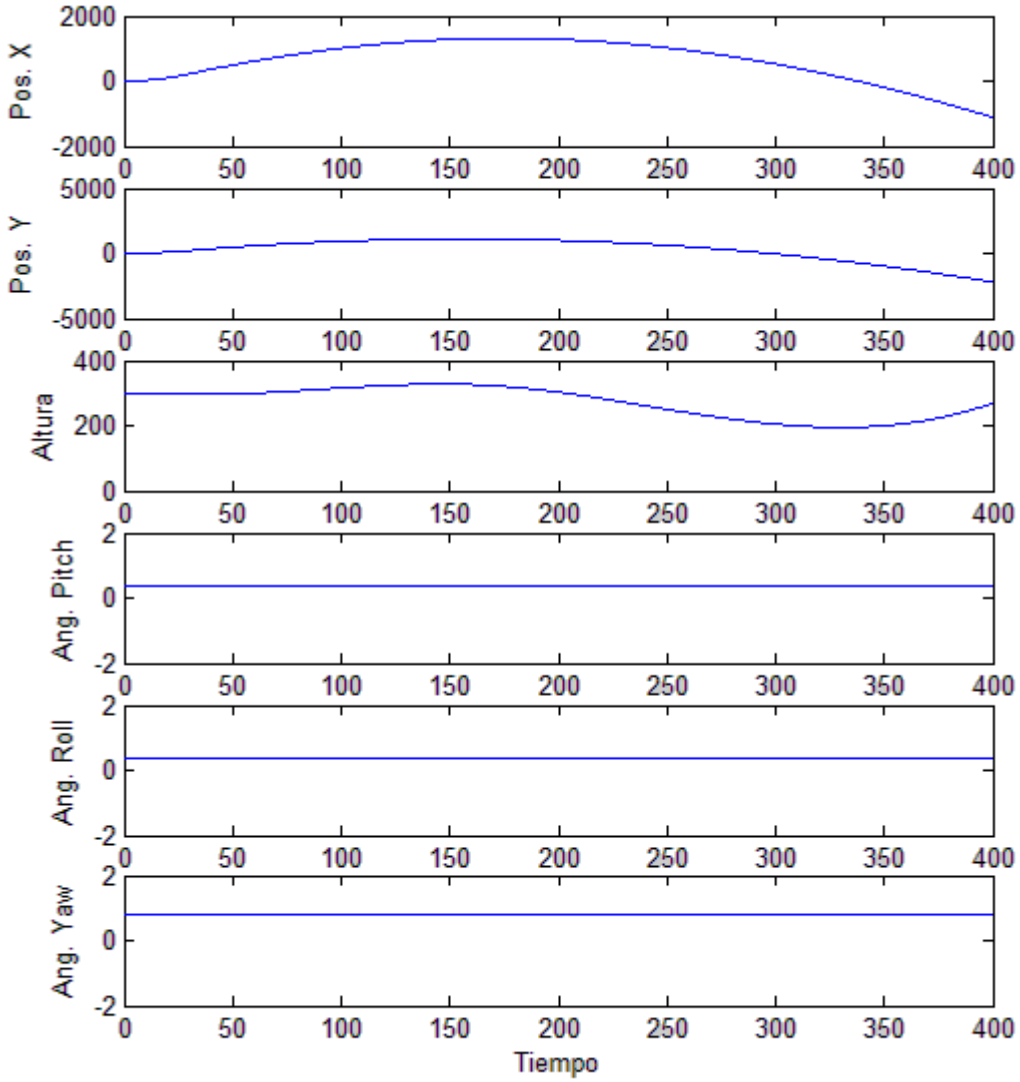


Figura 8. Evolución temporal de la posición y de la orientación durante la primera simulación.

A partir de las gráficas obtenidas tras la simulación se observa que el vehículo se comporta de la manera esperada, avanzando en primer lugar con el ángulo de deriva definido. Tras dejar de transmitir potencia al motor y la aparición de las perturbaciones continúa avanzando pero siendo su velocidad cada vez menor hasta provocar que comience a retroceder. Este descenso de velocidad traslacional afecta a la altura del autogiro, que decae momentáneamente para volver a ascender de nuevo cuando adquiere la velocidad necesaria.

En la segunda simulación se considera que el autogiro comienza en el origen de coordenadas, es decir, en la posición (x,y,z) = (0, 0, 0) m y con orientación (Φ, θ, Ψ) = (0, 0, 0) rad. En este caso no se van a considerar perturbaciones externas, simplemente se va a aplicar potencia al motor propulsor y se va a mantener una velocidad angular respecto al ángulo de deriva de $\frac{\pi}{50}$ rad/s. Los resultados de esta simulación se muestran en las figuras 9, 10 y 11.

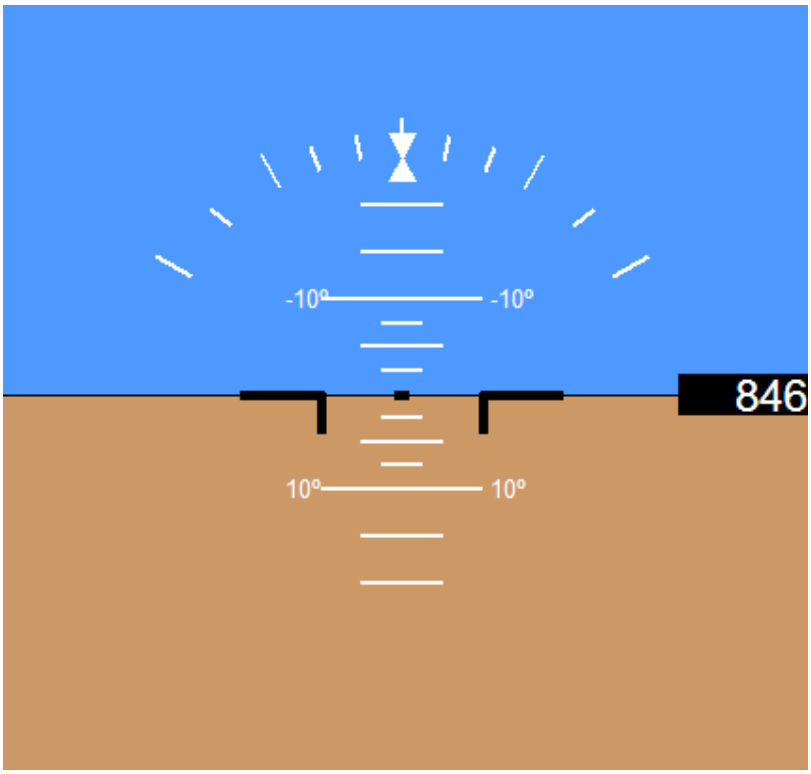


Figura 9. Altura y posición respecto al horizonte tras la segunda simulación.

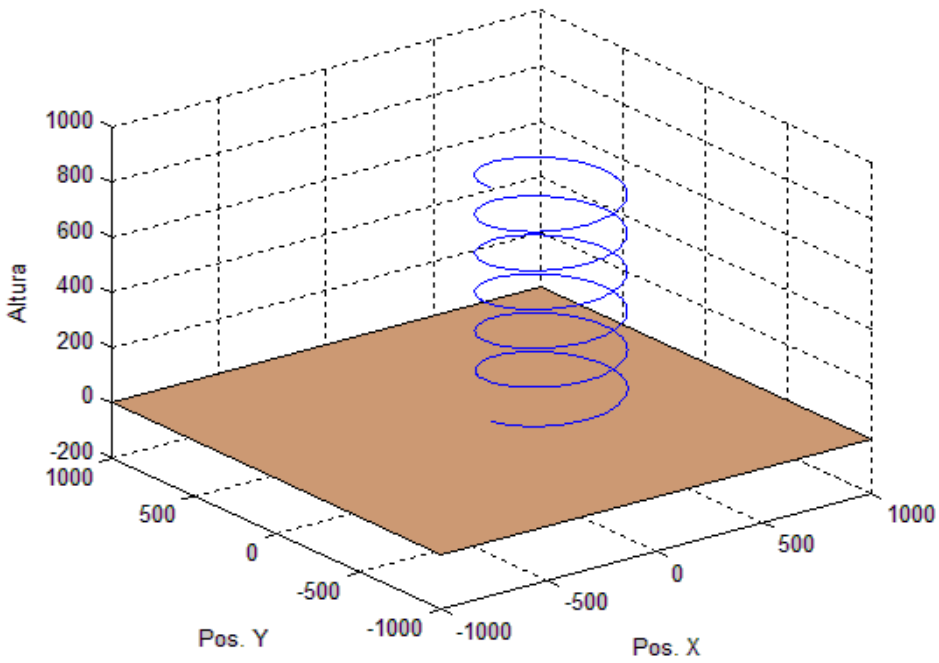


Figura 10. Trayectoria xyz durante la segunda simulación.

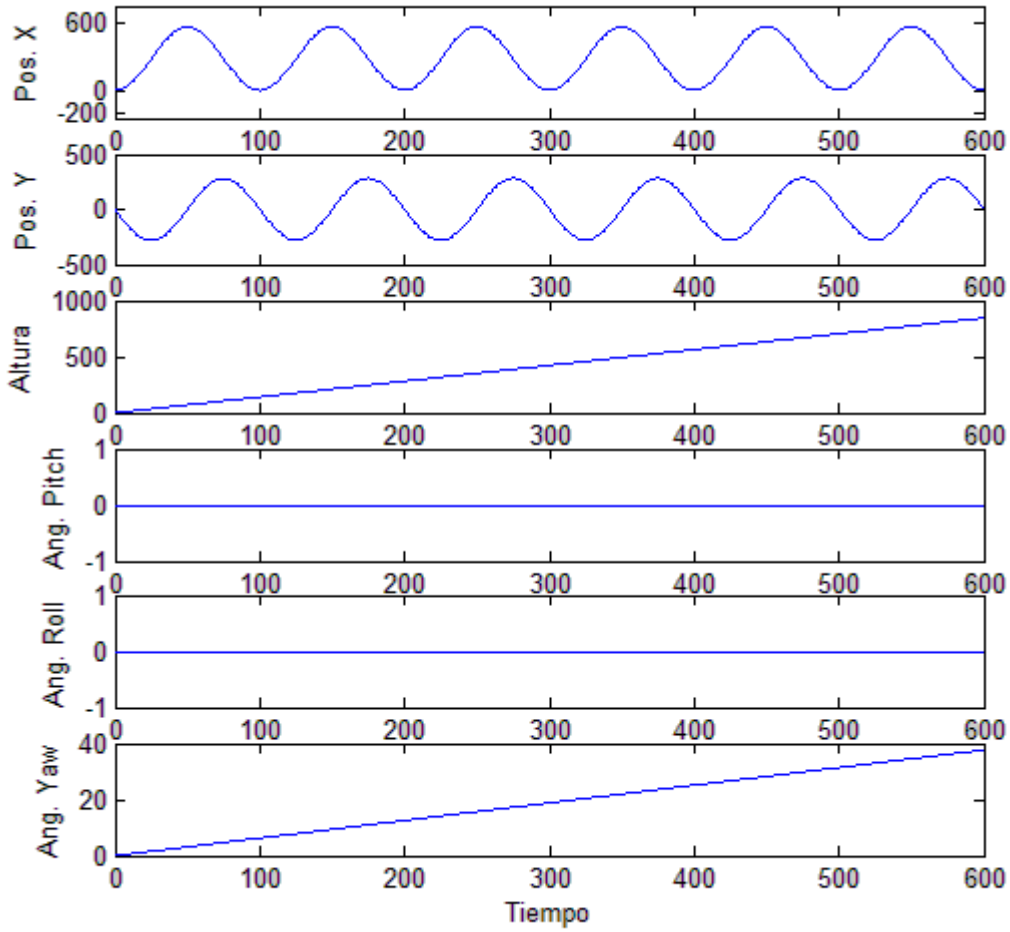


Figura 11. Evolución temporal de la posición y de la orientación durante la segunda simulación.

Las gráficas resultantes de esta simulación demuestran que la trayectoria es la prevista. El vehículo asciende lentamente pero a velocidad constante debido a que la velocidad de avance es la misma durante todo el trayecto y, dado que se mantiene también una velocidad de giro constante, dibuja una figura helicoidal en el espacio.

4.4 Conclusiones del capítulo

En este capítulo se ha propuesto el método de Runge-Kutta de cuarto orden como solución a las ecuaciones diferenciales que describen la dinámica del autogiro. De este modo, se ha podido evaluar la posición, orientación y velocidad del vehículo en cada instante. Se han realizado dos simulaciones de vuelo y se ha comprobado mediante las gráficas resultantes que el comportamiento de la aeronave es el esperado.

Una vez obtenido y probado el simulador de vuelo en el que se definen las fuerzas de entrada manualmente, el siguiente objetivo será modificar el programa de tal manera que sea el propio sistema quién determine el valor de estas entradas para cumplir con los parámetros de referencia prefijados.

5. Sistema de control

El control automático de vehículos aéreos representa un problema complejo de control debido a la naturaleza no lineal del sistema que se desea controlar. Cuando se desean controlar otros sistemas que presentan una variación lenta respecto al tiempo, como por ejemplo la temperatura de una habitación, se suele aproximar el sistema para linealizarlo alrededor de un punto. En nuestro caso, el movimiento del vehículo viene descrito por un modelo no lineal y los parámetros a controlar varían rápidamente. Por lo tanto, una aproximación demasiado simple de nuestro sistema provocaría un comportamiento no predecible. Las no linealidades se deben a factores inerciales, aerodinámicos y a la relación entre sus distintos grados de libertad.

El objetivo de este capítulo es analizar el problema de control del comportamiento del autogiro con ciertos parámetros de referencia asignados. Este control se realizará mediante la linealización por realimentación de estados, basándose en la cancelación de las no linealidades del sistema.

5.1 Control basado en la linealización por realimentación

La linealización por realimentación es una estrategia de control que linealiza la dinámica, o una parte de ella, de un sistema no lineal sin realizar aproximaciones. De este modo se evita el riesgo de hacer una aproximación suficientemente simple como para tener un comportamiento impredecible. Este método cancela las no linealidades del sistema con el objetivo de linealizar su dinámica, y, una vez conseguido, permitirá aplicar técnicas de control lineal.

El primer paso es definir el error de seguimiento como la diferencia entre el valor de referencia y el valor real de la variable a controlar [8]:

$$z = x_d - x \tag{30}$$

La derivada de la variable que se está controlando vendrá dada por:

$$\dot{x} = \dot{x}_d + \alpha\, z \tag{31}$$

Sustituyendo (30) en (31) se obtiene:

$$\dot{x} = \dot{x}_d + \alpha\, (x_d - x) \tag{32}$$

Con $\alpha > 0$.

A continuación se presenta un ejemplo sencillo para mostrar el procedimiento de obtención de la señal de control para un caso concreto. Suponiendo que se desea controlar la derivada de una variable s mediante una señal de control u y que está descrita por la ecuación:

$$\ddot{s} = 4\dot{s} - 5m - u \tag{33}$$

Se procede a aislar la señal de control:

$$u = 4\dot{s} - 5m - \ddot{s} \tag{34}$$

Y sustituyendo la segunda derivada de s por la forma obtenida en la ecuación (32) tenemos:

$$u = 4\dot{s} - 5m - \ddot{s}_d + \alpha\,(\dot{s}_d - \dot{s}) \tag{35}$$

5.1.1. Control del sistema de rotación

En primer lugar se va a realizar el control de los ángulos de orientación. La ecuación que hace referencia al ángulo de inclinación es:

$$\ddot{\Phi} = \frac{(\mathrm{I_{yy}} - \mathrm{I_{zz}})}{\mathrm{I_{xx}}}\,\dot{\theta}\dot{\Psi} - \frac{J_R\,\Omega}{\mathrm{I_{xx}}}\,\dot{\theta} + \frac{l}{\mathrm{I_{xx}}}\mathrm{U2} + \frac{A_p}{\mathrm{I_{xx}}} \tag{36}$$

Aislando la entrada de control U2:

$$U2 = \frac{\mathrm{I_{xx}}}{l}\left(\frac{(\mathrm{I_{zz}} - \mathrm{I_{yy}})}{\mathrm{I_{xx}}}\,\dot{\theta}\dot{\Psi} + \frac{J_R\,\Omega}{\mathrm{I_{xx}}}\,\dot{\theta} - \frac{A_p}{\mathrm{I_{xx}}} + \ddot{\Phi}\right) \tag{37}$$

Sustituyendo la derivada de la variable que se desea controlar por la forma obtenida en la ecuación (32) se tiene:

$$U2 = \frac{\mathrm{I_{xx}}}{l}\left(\frac{(\mathrm{I_{zz}} - \mathrm{I_{yy}})}{\mathrm{I_{xx}}}\,\dot{\theta}\,\dot{\Psi} + \frac{J_R\,\Omega}{\mathrm{I_{xx}}}\,\dot{\theta} - \frac{A_p}{\mathrm{I_{xx}}} + \ddot{\Phi}_d + \alpha_\Phi\left(\dot{\Phi}_d - \dot{\Phi}\right)\right) \tag{38}$$

Siguiendo los mismos pasos se obtienen las señales de control para U3 y U4, que afectan a los ángulos de alabeo y deriva:

$$U3 = \frac{\mathrm{I_{yy}}}{l}\left(\frac{(\mathrm{I_{xx}} - \mathrm{I_{zz}})}{\mathrm{I_{yy}}}\,\dot{\Phi}\,\dot{\Psi} - \frac{J_R\,\Omega}{\mathrm{I_{yy}}}\,\dot{\Phi} - \frac{A_q}{\mathrm{I_{yy}}} + \ddot{\theta}_d + \alpha_\theta\left(\dot{\theta}_d - \dot{\theta}\right)\right) \tag{39}$$

$$U4 = \mathrm{I_{zz}}\left(\frac{(\mathrm{I_{yy}} - \mathrm{I_{xx}})}{\mathrm{I_{zz}}}\,\dot{\theta}\dot{\Phi} - \frac{A_r}{\mathrm{I_{zz}}} + \ddot{\Psi}_d + \alpha_\Psi\left(\dot{\Psi}_d - \dot{\Psi}\right)\right) \tag{40}$$

5.1.2. Control del sistema de traslación

Para el control traslacional hay que tener en cuenta que, a diferencia del resto de aeronaves de ala rotatoria, el desplazamiento respecto al eje z depende de las velocidades a lo largo de los ejes x-y, que a su vez dependen de la propulsión generada por la señal de control U1. Así pues, no se puede forzar una velocidad horizontal y otra vertical por separado, ya que son dependientes entre sí. Por lo tanto, serán necesarias dos señales de control para gestionar el movimiento traslacional.

Si el objetivo es viajar a una velocidad de referencia vertical u horizontal sin importar la velocidad en el eje contrario, basta con utilizar la señal de control correspondiente. En caso de querer respetar una velocidad de referencia tanto horizontal como vertical, se debe alternar entre las dos señales en cada iteración, asumiendo que su comportamiento no será tan exacto como en el caso anterior.

La señal de control de la velocidad en el eje z se obtiene mediante el mismo procedimiento que para las señales de control del movimiento rotacional:

$$U1 = \frac{m^2}{cos\theta cos\Phi}\left(g - \frac{A_z}{m} + \ddot{z}_d + \alpha_z\left(\dot{z}_d - \dot{z}\right)\right) \tag{41}$$

Para obtener la señal de control horizontal se comienza definiendo la aceleración horizontal como:

$$\dot{xy} = \sqrt{\ddot{x}^2 + \ddot{y}^2} \tag{42}$$

Siendo $\dot{xy}$ el parámetro a controlar, se puede sustituir:

$$\sqrt{\ddot{x}^2 + \ddot{y}^2} = \dot{xy}_d + \alpha_{xy}\left(\dot{xy}_d - \dot{xy}\right) \tag{43}$$

Finalmente, sustituyendo los valores de las aceleraciones:

$$U1 = -cos\Psi\, A_x - sin\Psi\, A_y + \frac{\sqrt{(2cos\Psi\, A_x + 2sin\Psi\, A_y)^2 - 4((m(\dot{xy}_d + \alpha_{xy}\,(\dot{xy}_d - \dot{xy})))^2 - {A_x}^2 - {A_y}^2)}}{2} \tag{44}$$

5.1.3. Resultados de la simulación

Antes de comprobar el funcionamiento del control global, se van a realizar pruebas independientes para cada una de las señales de control propuestas. En primer lugar, se analiza el control de la velocidad en el eje xy. La velocidad que se quiere tener es de 15 m/s, la aceleración de referencia se considera nula tanto en esta como en las siguientes pruebas, con α_{xy} = 10 y sin tener en cuenta perturbaciones externas. El resultado de esta simulación se muestra en la figura 12.

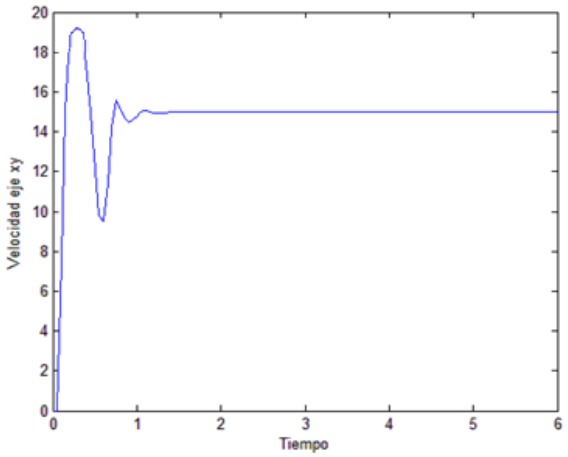


Figura 12. Evolución temporal de la velocidad en el eje xy cuando se aplica el control con linealización por realimentación.

Como se observa en la gráfica, tras un tiempo de asentamiento, la velocidad de avance es la deseada y, por tanto, el comportamiento es el adecuado. A continuación, se realiza el control de la velocidad vertical. Se desea una velocidad de ascenso de 1 m/s, considerando $\alpha_z = 10$ y sin presencia de perturbaciones. Esta simulación se muestra en la figura 13.

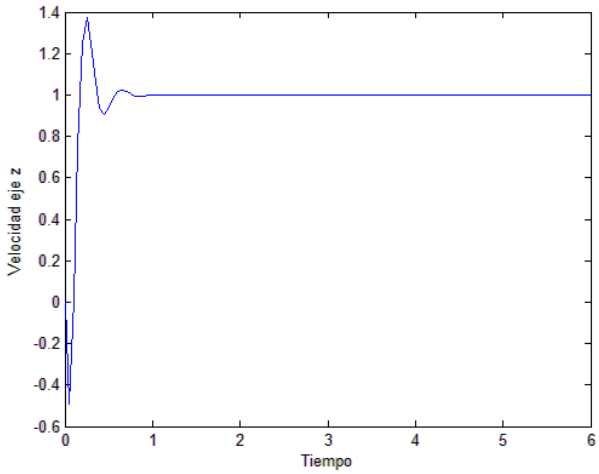


Figura 13. Evolución temporal de la velocidad en el eje z cuando se aplica el control con linealización por realimentación.

Como en el caso anterior, se observa que la velocidad es la deseada tras un tiempo de asentamiento. El siguiente paso es probar las señales de control que afectan al sistema de rotación. La velocidad deseada para cada uno de los ángulos es de 0.1 rad/s, con $\alpha_{\Phi} = \alpha_{\theta} = \alpha_{\Psi} = 10$ y sin ningún tipo de perturbación. Los resultados de estas simulaciones se muestran en las figuras 14, 15 y 16.

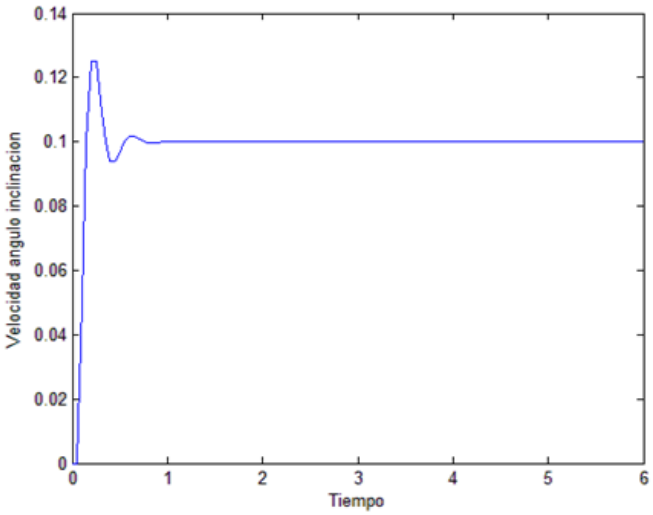


Figura 14. Evolución temporal de la velocidad de inclinación cuando se aplica el control con linealización por realimentación.

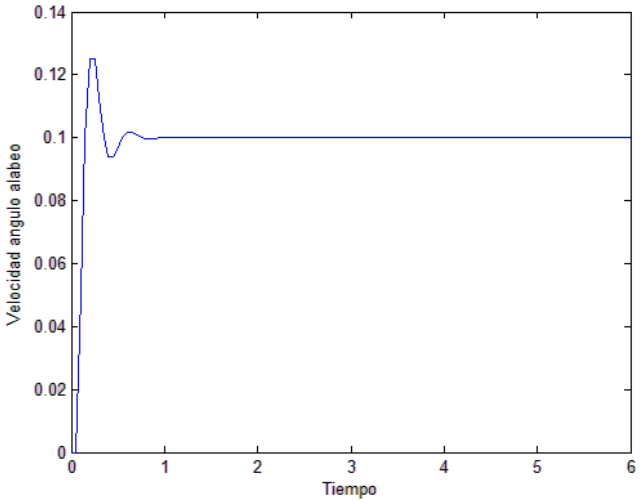


Figura 15. Evolución temporal de la velocidad de alabeo cuando se aplica el control con linealización por realimentación.

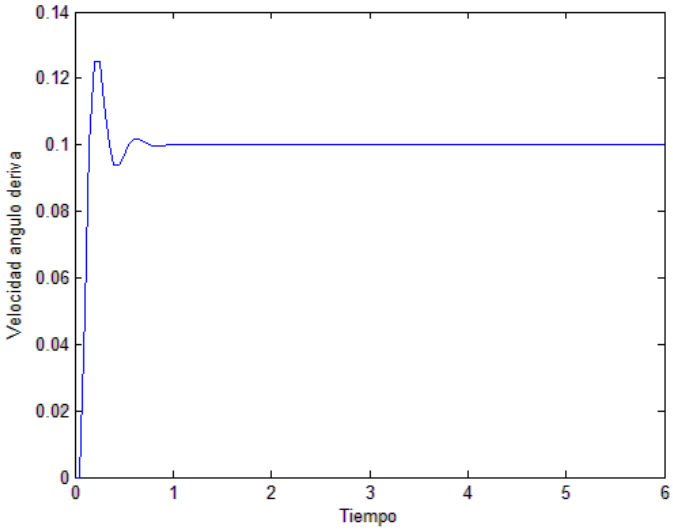


Figura 16. Evolución temporal de la velocidad de deriva cuando se aplica el control con linealización por realimentación.

Tras analizar todas las gráficas obtenidas tras las simulaciones, se puede afirmar que el controlador funciona correctamente con los valores α asignados. En todos los casos presenta un tiempo de asentamiento determinado y, posteriormente, alcanza la velocidad de referencia que ha sido fijada previamente por el usuario.

Una vez que se ha comprobado que el funcionamiento del controlador es el adecuado, se procede a realizar una prueba con presencia de perturbaciones. Para esta prueba se va a considerar nuevamente que la velocidad deseada de avance en el eje xy es de 15 m/s, con α_{xy}=10 y se va a introducir una perturbación externa, es decir, un escalón aerodinámicos en A_x de 900 Nm en t = 4s. El resultado de esta simulación se muestra en la figura 17.

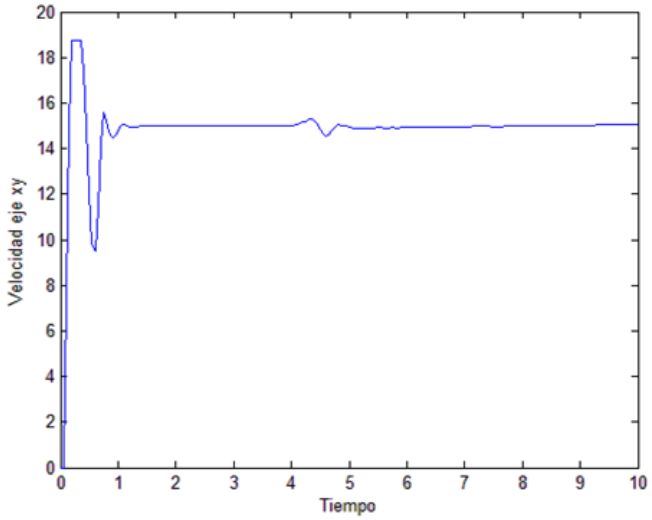


Figura 17. Evolución temporal de la velocidad en el eje xy cuando se aplica el control con linealización por realimentación con perturbaciones externas.

En esta gráfica se observa que, igual que en los casos anteriores, tras un tiempo de asentamiento se adquiere la velocidad de referencia. Más adelante, cuando se introduce la perturbación, se observa una variación de la velocidad durante un cierto tiempo. Finalmente, el controlador rechaza esta perturbación y se vuelve a avanzar a la velocidad deseada.

La siguiente prueba del controlador tratará el caso en el que se quiere ascender a una velocidad determinada a la vez que se desea avanzar a otra velocidad también fijada previamente. Estas velocidades serán de 1 m/s en el eje vertical y de 5 m/s para el avance horizontal, partiendo de la posición (x,y,z) = (0, 0, 300) m y con orientación (Φ, θ, Ψ) = (0, 0, $\frac{\pi}{4}$) rad. El valor tanto de α_z como de α_{xy} será de 10 y no se tendrán en cuenta perturbaciones externas. Los resultados de esta simulación se muestran en las figuras 18, 19 y 20.

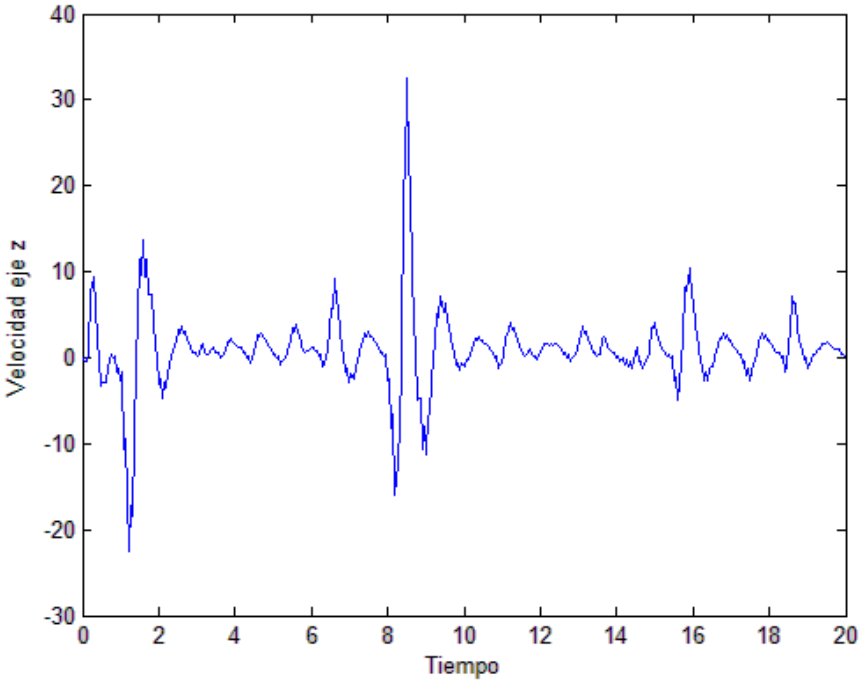


Figura 18. Evolución temporal de la velocidad en el eje z cuando se aplica el control con linealización por realimentación.

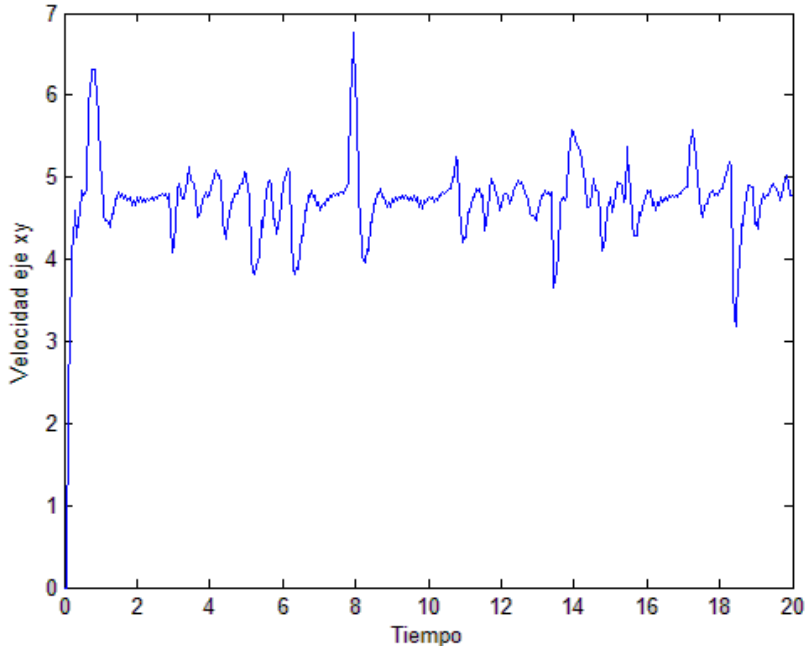


Figura 19. Evolución temporal de la velocidad en el eje xy cuando se aplica el control con linealización por realimentación.

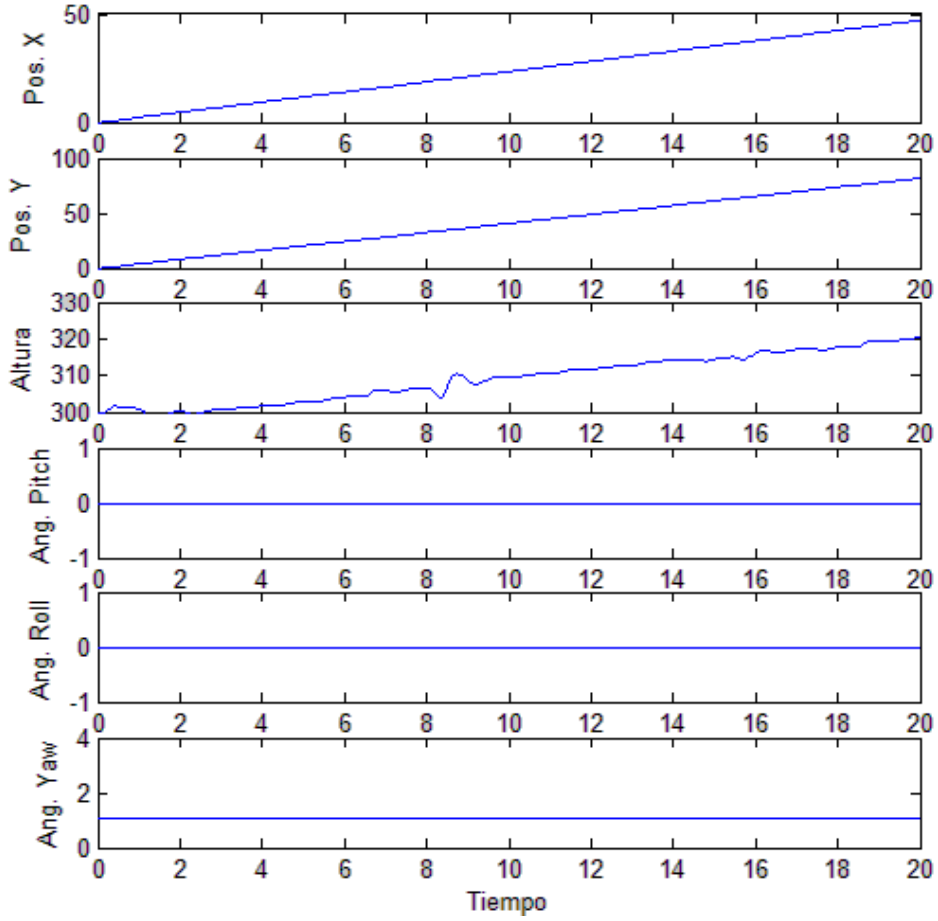


Figura 20. Evolución temporal de la posición y de la orientación cuando se aplica el control con linealización por realimentación.

Las gráficas de esta simulación muestran que las velocidades son mucho más irregulares, como era previsible, que en los casos anteriores. No obstante, el resultado, como se observa en la figura 20, es bastante preciso.

Finalmente, se va a probar el seguimiento de trayectorias de referencia ya conocidas. Se realizarán tres simulaciones para comprobar que el vehículo cumple con las previsiones. En la primera simulación se va a realizar un cambio de altura, partiendo de la posición (x,y,z) = (0, 0, 300) m y con orientación (Φ, θ, Ψ) = (0, 0, $\frac{\pi}{4}$) rad, volará a una altura constante. Entre t = 5s comenzará el ascenso a una velocidad de 1 m/s para volver a mantenerse a la misma altura a partir de t = 10 s. Se utiliza α_z = 10 y no se introducen perturbaciones. Esta simulación se presenta en las figuras 21 y 22.

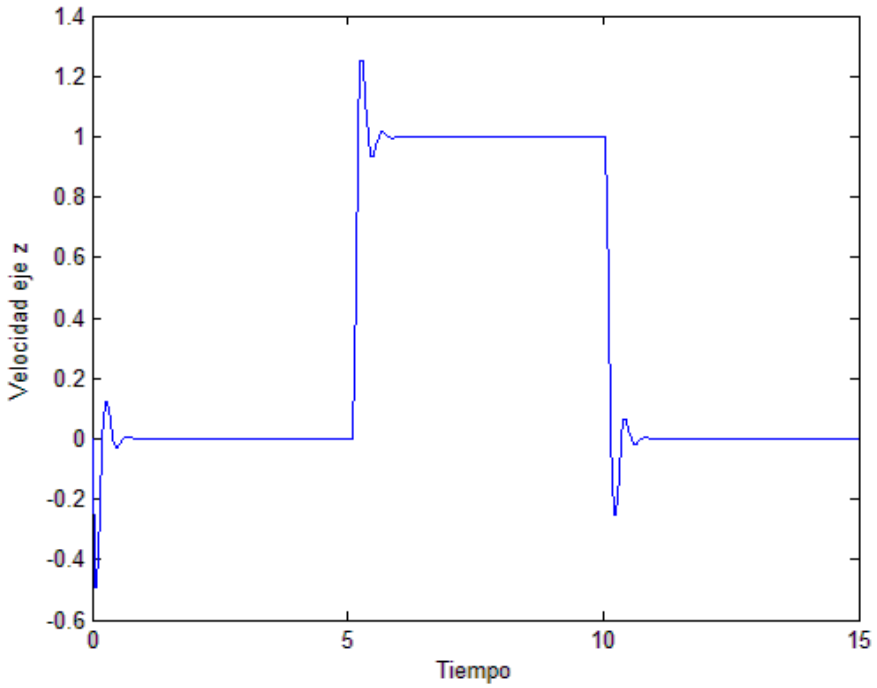


Figura 21. Evolución temporal de la velocidad en el eje z para el seguimiento de trayectoria de la primera simulación cuando se aplica el control con linealización por realimentación.

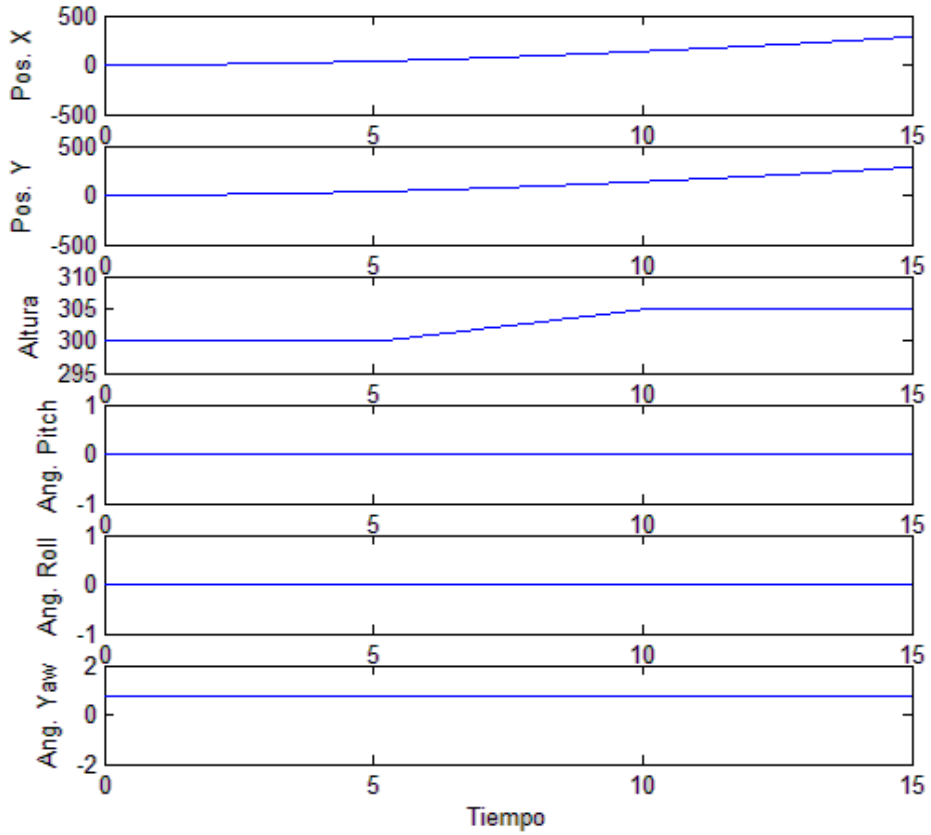


Figura 22. Evolución temporal de la posición y de la orientación para el seguimiento de trayectoria de la primera simulación cuando se aplica el control con linealización por realimentación.

La gráfica de la figura 21 muestra cómo la velocidad de ascenso es nula durante los primeros cinco segundos, pasa a ser de 1 m/s en t = 5s tras el tiempo de asentamiento necesario y se vuelve a anular a partir de los 10 segundos de vuelo. En la figura 22 vemos que el ascenso ha sido el esperado, es decir, 5 metros en 5 segundos.

En la siguiente simulación se va a considerar que el origen es la posición (x,y,z) = (-800, -800, 300) m y su orientación (Φ, θ, Ψ) = (0, 0, 0) rad. El objetivo es que, volando a una altura constante, una vez alcanzada la posición (x,y,z) = (-200, -800, 300) realice un cambio de sentido. Para conseguirlo, se va a utilizar una velocidad de referencia en la velocidad de deriva de $\frac{2\pi}{25}$ rad/s hasta alcanzar un ángulo de deriva, $\Psi = \pi$ rad. Los valores de α_z y de α_ψ serán de 10 y no se tendrán en cuenta perturbaciones externas. Los resultados se muestran en la figuras 23.

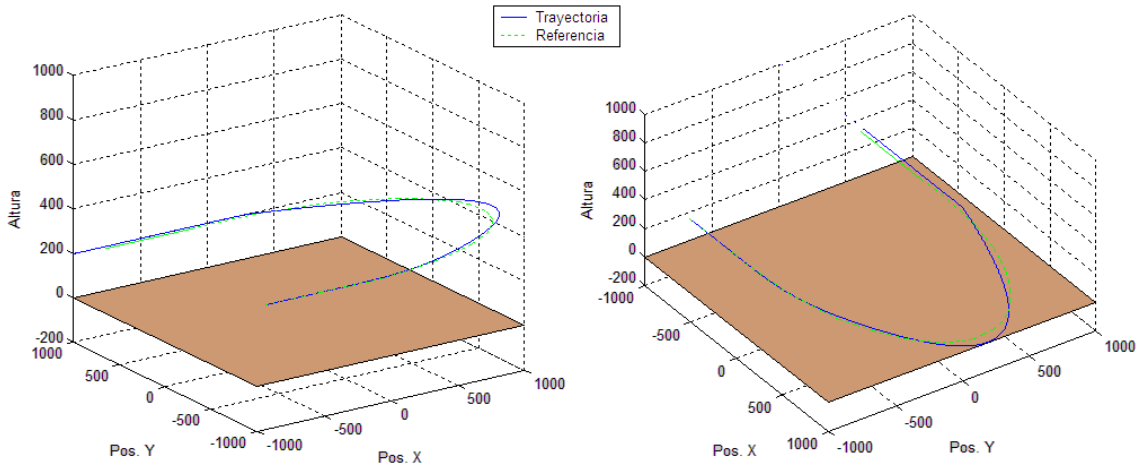


Figura 23. Trayectoria xyz durante el seguimiento de trayectoria de la segunda simulación cuando se aplica el control con linealización por realimentación desde dos ángulos de vista.

En la gráfica obtenida se observa que el controlador ha trabajado correctamente ya que la trayectoria es la esperada. En primer lugar avanza sobre el eje x, cuando alcanza el punto de referencia, comienza a variar el ángulo de deriva hasta alcanzar los π rad para retroceder a lo largo del mismo eje.

En la última simulación vamos a considerar que partimos del origen de coordenadas y de orientación, es decir: (x,y,z) = (0, 0, 0) m y (Φ, θ, Ψ) = (0, 0, 0) rad. El objetivo es que el autogiro ascienda a una velocidad de referencia $\dot{z}_d$ = 1.4 m/s dibujando una figura helicoidal en el espacio. Para que se comporte de esta manera, el controlador deberá mantener una velocidad de avance $\dot{xy}$ de 15 m/s y una velocidad del ángulo de deriva $\dot{\Psi}$ de $\frac{2\pi}{25}$ rad/s. Se cuenta con valores de $\alpha_z = \alpha_\psi = \alpha_{xy}$ = 10 y sin presencia de perturbaciones. Los resultados quedan reflejados en la figura 24.

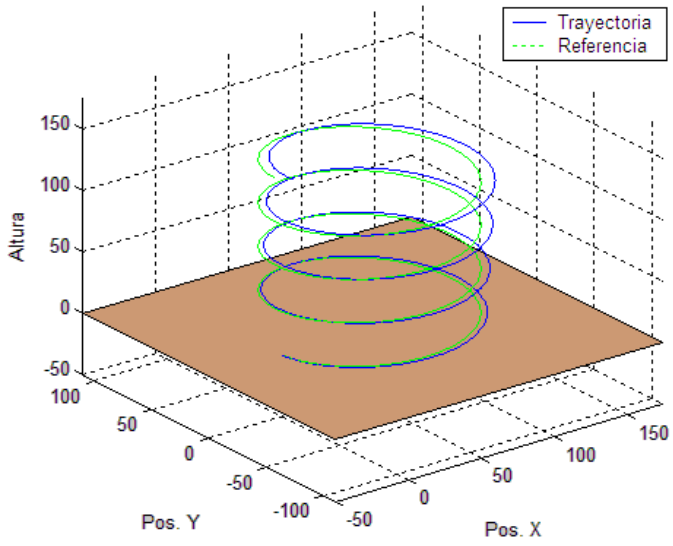


Figura 24. Trayectoria xyz durante el seguimiento de trayectoria de la tercera simulación cuando se aplica el control con linealización por realimentación.

Como se observa en la gráfica, la trayectoria del vehículo dibuja una figura helicoidal en el espacio, demostrando que el controlador cumple con el objetivo de mantener las velocidades de referencia fijadas de tal manera que la trayectoria realizada sea la deseada.

5.2 Conclusiones del capítulo

En este capítulo se ha diseñado el sistema de control del autogiro. La estrategia utilizada ha sido la de linealización por realimentación. Se han definido las dos señales de control correspondientes al sistema de traslación y las tres señales referentes al sistema de rotación, siendo la velocidad en todos los casos el parámetro de referencia. Se han realizado simulaciones para cada una de las señales de control de manera independiente y tres simulaciones en las que se aplican varias señales de control al mismo tiempo y se pretende seguir una trayectoria ya conocida. Se ha comprobado que tanto en las simulaciones individuales como en el seguimiento de trayectoria el sistema de control ha funcionado correctamente y que la aeronave se ha comportado de la manera esperada.

La incorporación del controlador al sistema da por finalizada la realización del simulador y nos permitirá desarrollar el código necesario para su implementación digital.

6. Implementación

Este capítulo está dedicado a la implementación del sistema. Para dicha implementación contamos con varias opciones que se analizan a continuación:

- **FPGA:** este dispositivo contiene bloques de lógica totalmente configurables y es reprogramable. Son bastante rápidos y permiten configurar cualquier sistema pero su rendimiento no sería el óptimo ya que la lógica estaría desordenada.
- **DSP:** está diseñado para el tratamiento digital de las señales. Ofrece una gran velocidad de proceso. Este bloque no se basta por sí solo para trabajar, debe de conectarse a otros bloques como el de la memoria y el de la entrada y salida.
- **Microcontrolador PIC:** en este chip están integrados todos los bloques comentados. El tiempo de desarrollo es más pequeño ya que tiene un diseño más simple, pese a que no es capaz de trabajar a velocidades tan grandes.
- **Microcontrolador dsPIC:** se puede considerar como un microcontrolador PIC que cuenta con DSP. Así, la velocidad será muy superior, asumiendo que el costo y la complejidad también aumentarán.

Debido a que nuestro sistema de control necesita trabajar a tiempo real y no puede permitirse fallos, ya que es un sistema crítico y, en caso de cometer un error, puede causar un daño grave, la opción que más se ajusta a nuestras necesidades es la del microcontrolador dsPIC.

6.1 Elección del dsPIC

El siguiente paso es escoger un modelo que cumpla con los requisitos para implementar el sistema. Los principales requisitos que debe cumplir nuestro chip son: contar con un conversor analógico-digital con un mínimo de 14 pines de entrada y un módulo PWM (*Pulse Width Modulation*) con un mínimo de 4 pines de salida para generar las salidas analógicas de nuestro sistema. Para escoger un modelo que se adapte a nuestras necesidades accedemos a la página web del fabricante, Microchip [9], donde se muestran las características de cada uno de ellos.

La familia dsPIC30 ofrece un gran rendimiento a un precio competitivo. Dentro de esta familia encontramos varios modelos cuyas diferencias principales se observan en la tabla 4.

Product	Volume Pricing	Program Memory KBytes	RAM	Code-Guard Security	EEPROM Data Memory	Internal Oscillator	Motor Control PWM Channels	Analog Peripherals	I/O Pins
dsPIC30F5016	$5.59	66	2048	Basic	1024	7.37 MHz, 512 kHz	8	16 x 10-bit @ 1000 (ksps) 1-A/D	68
dsPIC30F6011A	$6.89	132	6144	Advanced	2048	7.37 MHz, 512 kHz		16 x 12-bit @ 200 (ksps) 1-A/D	52
dsPIC30F6013A	$7.14	132	6144	Advanced	2048	7.37 MHz, 512 kHz		16 x 12-bit @ 200 (ksps) 0-D/A 0 x 0-bit @ 0 (ksps) 1-A/D	68
dsPIC30F6010A	$7.01	144	8192	Advanced	4096	7.37 MHz, 512 kHz	8	16 x 10-bit @ 1000 (ksps) 1-A/D	68
dsPIC30F6012A	$6.96	144	8192	Advanced	4096	7.37 MHz, 512 kHz		16 x 12-bit @ 200 (ksps) 1-A/D	52
dsPIC30F6014A	$7.25	144	8192	Advanced	4096	7.37 MHz, 512 kHz		16 x 12-bit @ 200 (ksps) 1-A/D	68
dsPIC30F6015	$7.18	144	8192	Advanced	4096	7.37 MHz, 512 kHz	8	16 x 10-bit @ 1000 (ksps) 1-A/D	52

Tabla 4. Comparativa entre dsPICs de la familia dsPIC30.

Tras analizar los datos de cada modelo, se observa que hay tres que cumplen los requisitos que se pedían: dsPIC30F5016, dsPIC30F6010A y dsPIC30F6015. Estos tres chips cuentan con un conversor analógico-digital de 16 pines de entrada y con 8 pines de salida para las señales analógicas generadas. Para nuestro sistema se va a optar por el dsPICF6010A ya que cuenta con más del doble de memoria de programa que el dsPICF5016 y con 16 pines extra de entrada/salida respecto al dsPIC30F6015, además de ser más económico que éste último.

El dsPIC30F6010A cuenta con un total de 80 pines, como se observa en la figura 25. De ellos, prestaremos especial atención a los de entrada del conversor analógico-digital (AN0 a AN15) y a los 8 pines del módulo PWM (PWM1L a PWM4H), de los cuales cuatro serán utilizados para las salidas analógicas.

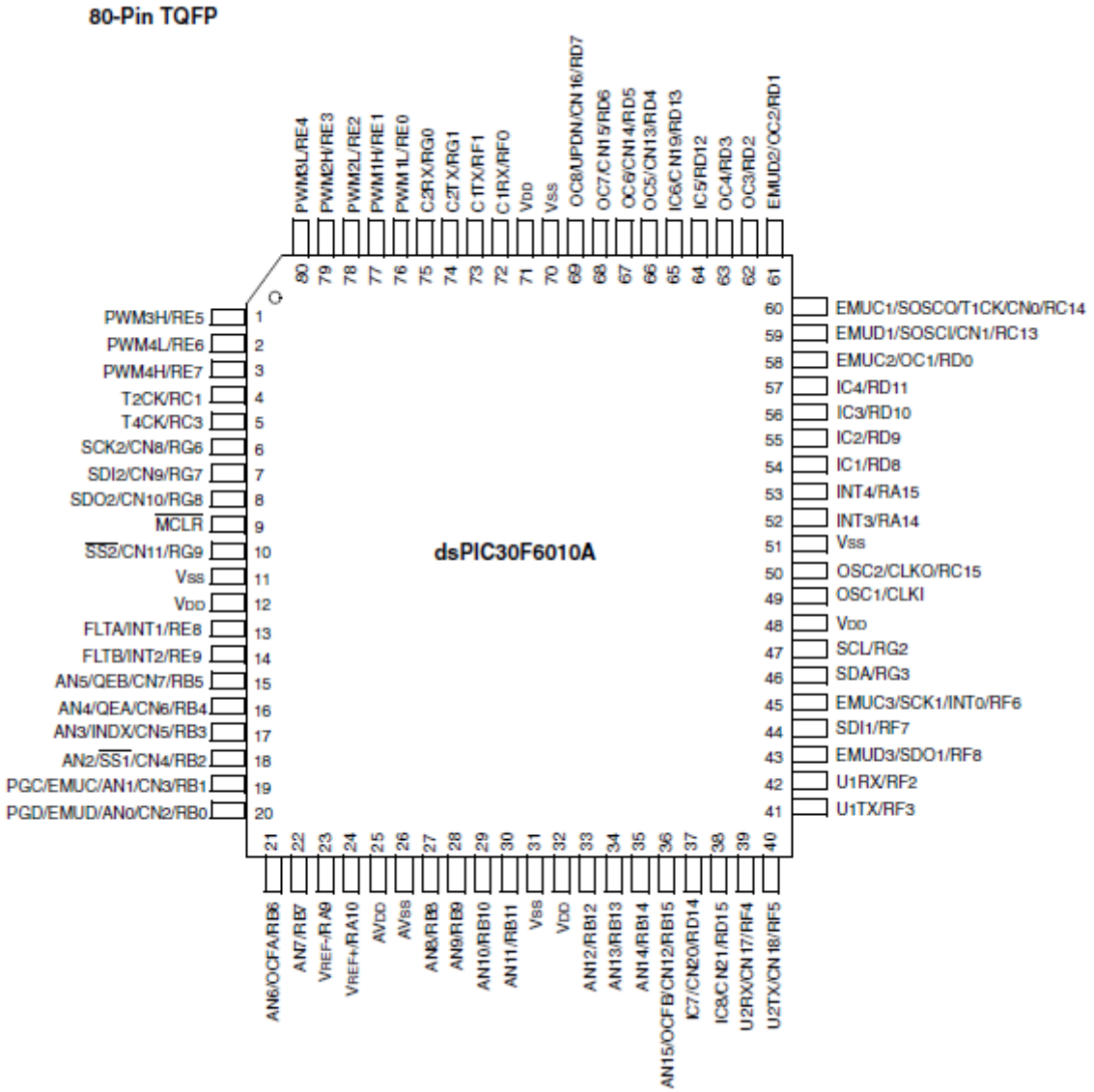


Figura 25. Diagrama de pines del dsPIC30F6010A.

6.2 Programación del dsPIC30F6010A

En este apartado se va a describir el procedimiento a seguir para programar el dsPIC, configurando los distintos módulos según convenga a nuestras necesidades y escribiendo las instrucciones necesarias para ejecutar el sistema.

6.2.1. Programación del temporizador

En primer lugar se va a programar el temporizador. Su función será generar una interrupción cada vez que queramos leer las entradas y generar las señales de control de salida. Utilizaremos uno de los cinco temporizadores de los que está provisto nuestro dsPIC, el TIMER1. Para configurar este temporizador se utiliza el registro de 16 bits T1CON. En la figura 26 podemos ver el significado de cada uno de estos bits.

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMR1	0100	Timer1 Register															
PR1	0102	Period Register 1															
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—

Figura 26. Registros del temporizador 1.

Los bits del registro T1CON que necesitamos modificar son TON, que deberá activarse para poner en funcionamiento el temporizador, TCKPS, para seleccionar el prescaler necesario y TCS. El bit TCS, que por defecto está desactivado, sirve para seleccionar el reloj del temporizador, pudiendo ser el reloj interno del dsPIC (cuando TCS = 0) o un oscilador externo (TCS = 1).

En el registro PR1, que consta de 16 bits, se escribe el valor que queremos que alcance el temporizador para generar la interrupción. En nuestro caso calcularemos los ciclos necesarios para que se genere una interrupción cada 50 ms:

$$Número\ de\ ciclos = \frac{F_{osc}}{4 \cdot Prescaler} \cdot Tiempo \tag{45}$$

Sustituyendo por los valores adecuados, con un prescaler de valor 8 (TCKPS = 01) y oscilador local, obtenemos:

$$\frac{7370 \cdot\ 10^3\ Hz}{4 \cdot 8} \cdot 0.05\ s = 11515.6\ ciclos \tag{46}$$

Por lo tanto, el valor que debemos escribir en el registro PR1 es 11516. Ahora se debe activar el bit T1IE, situado en el bit 3 del registro IEC0, para habilitar las interrupciones de este temporizador. Además, cada vez que se genere una interrupción, se deberá desactivar el bit T1IF, situado en el bit 3 del registro IFS0, ya que se activa automáticamente cuando ocurre la interrupción.

6.2.2. Programación del conversor analógico-digital

El dsPIC30F6010A cuenta con un conversor analógico-digital interno. Éste consta de un máximo de 16 pines de entrada, como se observa en la figura 27. El valor convertido se guarda en un búfer de sólo lectura.

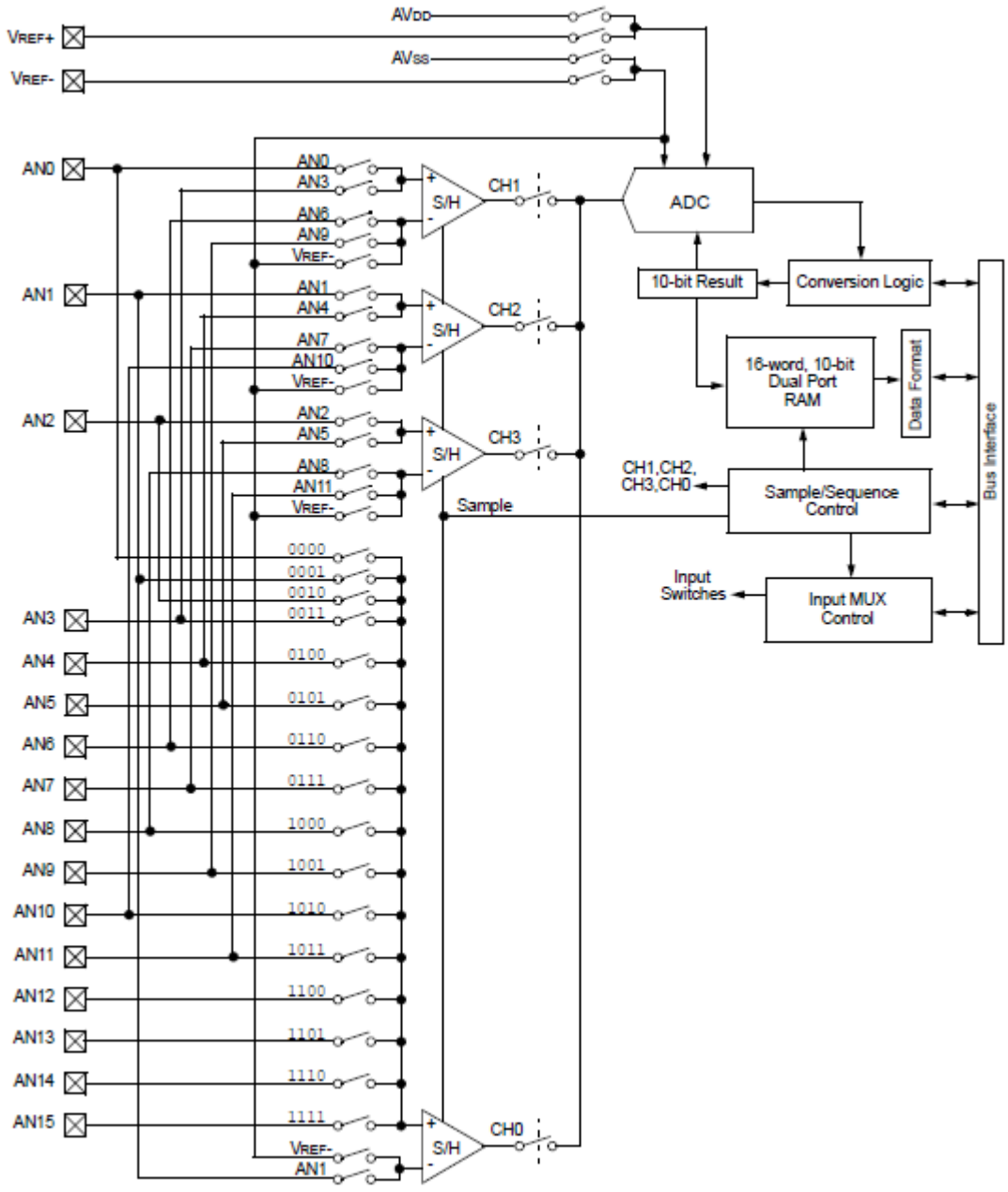
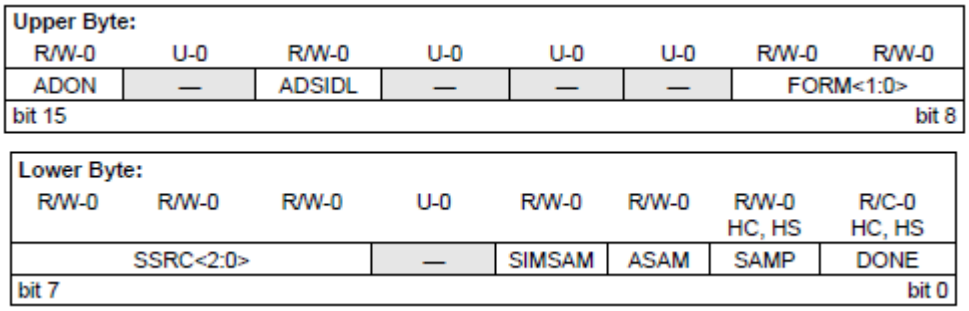


Figura 27. Diagrama de bloques del conversor analógico-digital del dsPIC30F6010A.

Antes de configurar el conversor tenemos que saber cómo se van a utilizar los pines de entrada. En nuestro caso, los primeros 14 pines estarán conectados a sensores que nos darán la información referente a las velocidades del vehículo (horizontal, vertical y de rotación respecto a cada eje y al eje del rotor), su orientación (ángulos de cabeceo, alabeo y deriva) y las velocidades de referencia que tiene que seguir. Los dos siguientes pines se utilizarán como entrada digital, en este caso el conversor no convertirá su valor, para determinar si el control de propulsión será de avance horizontal, vertical o alterno.

Una vez definido el uso que se le va a dar a cada pin se procede a configurar el conversor de la manera pertinente. En primer lugar se van a activar las interrupciones del conversor activando el bit ADIE, situado en el bit 11 del registro IEC0, y a continuación se modifican los registros ADCON1, ADCON2 y ADCON3. El primero de ellos está formado por los 16 bits que muestra la figura 28.



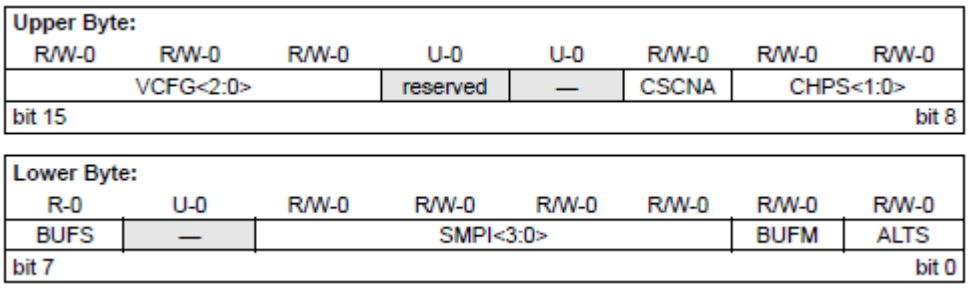
Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	—	—	—	FORM<1:0>	
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0 HC, HS	R/C-0 HC, HS
SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE
bit 7							bit 0

Figura 28. Registro ADCON1 del conversor.

El bit ADON será necesario activarlo para encender el conversor. Los bits FORM definen el formato en que la conversión se guarda en el búfer, que en nuestro caso será fraccional con signo (FORM = 11, DOUT = sddd dddd dd00 0000). Los bits SSRC determinan la manera en que se inician y se detienen las conversiones, en nuestro caso optamos por la conversión automática (SSRC = 111), ignorando los valores de ASAM y SAMP. Por último, el bit DONE, de sólo lectura, determina si las conversión ha finalizado o no.

Los 16 bits que forman el registro ADCON2 son los que aparecen en la figura 29.



Upper Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>			reserved	—	CSCNA	CHPS<1:0>	
bit 15							bit 8

Lower Byte:							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							bit 0

Figura 29. Registro ADCON2 del conversor.

Los bits VCFG se utilizan para seleccionar los valores VrefH y VrefL del conversor. En nuestro caso vamos a utilizar los valores de AV_{DD} y AV_{SS} (VCFG = 000). El bit CSCNA se debe activar para habilitar la lectura de los pines de entrada. Los bits CHPS se utilizan para seleccionar el pin que se va a convertir, pero en nuestro caso, como se verá más adelante, se va a optar por un muestreo secuencial automático y, por lo tanto, estos bits serán ignorados. Los bits SMPI determinan el número de conversiones que tiene que realizar el conversor antes de generar la interrupción, en nuestro caso 14 (SMPI = 1110).

El siguiente registro, ADCON3, cuenta con los 16 bits descritos en la figura 30.

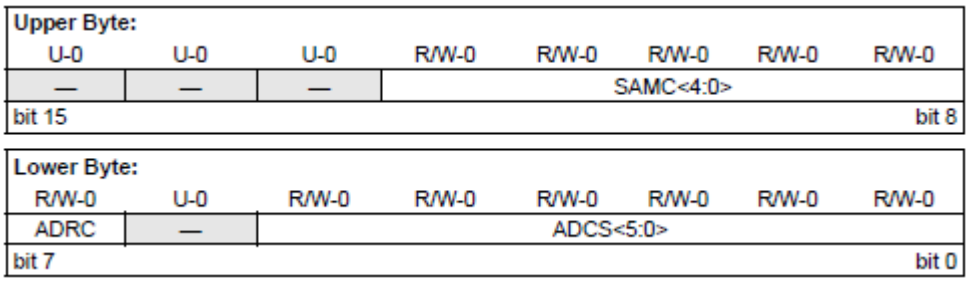


Figura 30. Registro ADCON3 del conversor.

En este registro vamos a utilizar los bits SAMC y ADCS para determinar el tiempo de muestreo. En nuestro caso vamos a tomar ADCS = 000000, SAMC = 00001 y vamos a comprobar que cumpla la condición de T_{ad} mínimo de la tabla 5. Para ello calculamos:

$$T_{ad} = \frac{T_{cy}}{2} \tag{47}$$

Donde T_{cy} es el tiempo de un ciclo:

$$T_{cy} = \frac{4}{F_{osc}} = \frac{4}{7370 \cdot 10^3\ Hz} = 543\ ns \tag{48}$$

Sustituyendo:

$$T_{ad} = \frac{543\ ns}{2} = 271.37\ ns\ > 153.85\ ns \tag{49}$$

Finalmente:

$$T_{sample} = T_{ad} \cdot SAMC = 271.37\ ns \tag{50}$$

Siendo el tiempo total de muestreo y conversión de:

$$T_{tot} = T_{sample} + T_{conv} = 271.37\ ns + 12 \cdot 271.37\ ns = 3.53\ \mu s \tag{51}$$

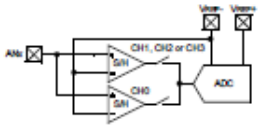
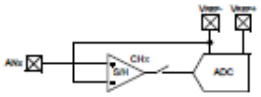
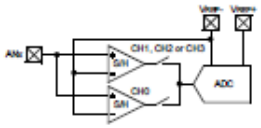
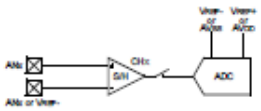
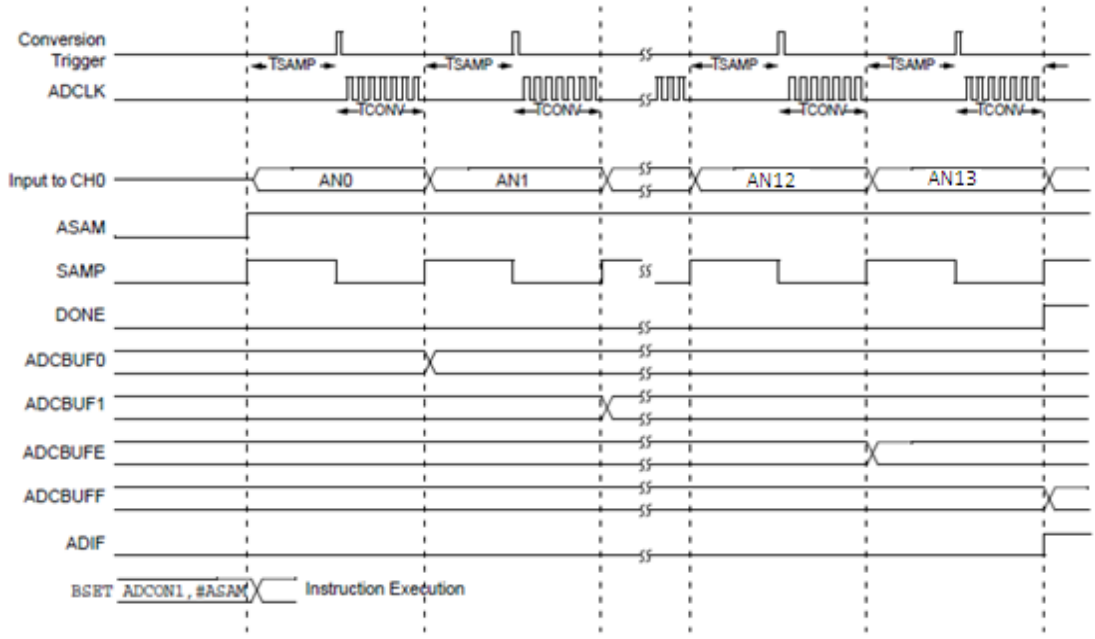
A/D Speed	TAD Minimum	Sampling Time Min	R_S Max	VDD	Temperature	A/D Channels Configuration
Up to 1 Msps	83.33 ns	12 TAD	500 Ω	4.5V to 5.5V	-40°C to +85°C	
Up to 750 ksps	95.24 ns	2 TAD	500 Ω	4.5V to 5.5V	-40°C to +85°C	
Up to 600 ksps	138.89 ns	12 TAD	500 Ω	3.0V to 5.5V	-40°C to +125°C	
Up to 500 ksps	153.85 ns	1 TAD	5.0 kΩ	4.5V to 5.5V	-40°C to +125°C	
Up to 300 ksps	256.41 ns	1 TAD	5.0 kΩ	3.0V to 5.5V	-40°C to +125°C	

Tabla 5. Ratio de conversión del conversor.

El siguiente registro a configurar es ADPFG, de 16 bits. Cada uno de sus bits determina si los pines conectados al conversor son entradas analógicas o entradas y salidas digitales. En nuestro caso, los dos bits más significativos de ADPFG estarán activados, ya que los utilizaremos como datos de entrada digitales, y el resto de bits desactivados, que es el valor que determina que se trata de una entrada analógica. Además, será necesario activar el bit TRIS correspondiente a cada uno de los 16 pines, que se encuentran en el puerto B del dsPIC, para indicar que la dirección de todos los pines es de entrada.

Por último, el registro ADCSSL, también consta de 16 bits y es el encargado de determinar qué entradas del conversor se van a muestrear secuencialmente de manera automática. Cada uno de sus bits hace referencia a una de las entradas del conversor. Los bits activados formarán parte de la secuencia de muestreo y los demás serán ignorados. En nuestro caso se desactivarán los 2 bits más significativos y se activarán los 14 restantes. En la figura 31 se muestra de manera gráfica cómo se va a realizar esta secuencia y dónde se va a guardar el valor de cada conversión.



Buffer	Buffer @ Primera Interrupción	Buffer @ Segunda Interrupción
ADCBUF0	AN0 sample 1	AN0 sample 17
ADCBUF1	AN1 sample 2	AN1 sample 18
ADCBUF2	AN2 sample 3	AN2 sample 19
ADCBUF3	AN3 sample 4	AN3 sample 20
ADCBUF4	AN4 sample 5	AN4 sample 21
ADCBUF5	AN5 sample 6	AN5 sample 22
ADCBUF6	AN6 sample 7	AN6 sample 23
ADCBUF7	AN7 sample 8	AN7 sample 24
ADCBUF8	AN8 sample 9	AN8 sample 25
ADCBUF9	AN9 sample 10	AN9 sample 26
ADCBUFA	AN10 sample 11	AN10 sample 27
ADCBUFB	AN11 sample 12	AN11 sample 28
ADCBUFC	AN12 sample 13	AN12 sample 29
ADCBUFD	AN13 sample 14	AN13 sample 30

Figura 31. Muestreo secuencial automático del conversor analógico-digital.

Cada vez que el temporizador del dsPIC genere una interrupción se iniciará el muestreo secuencial del conversor. Se esperará hasta que el bit ADIF se active, que será cuando se hayan terminado las 14 conversiones, y se convertirá el valor de cada búfer para asignárselo a la variable que corresponda en cada caso (velocidad, ángulo, etc.). Para ello se comprobará el bit que hace referencia al signo de la magnitud y se asignará un valor máximo a cada variable medida, más un factor de seguridad del 20%, de modo que se escalen los valores de tensión de entrada. Una vez asignados estos valores se procede a calcular las 4 señales de control del sistema y se desactiva el bit ADIF, situado en el bit 11 del registro IFS0, ya que se activa automáticamente cada vez que finalizan las 14 conversiones.

6.2.3. Programación del módulo PWM

El último paso que queda por hacer es generar las salidas analógicas del dsPIC a partir de las señales digitales de control calculadas por nuestro sistema. Para hacerlo vamos a utilizar el módulo PWM que tiene incorporado el dsPIC30F6010A.

Éste módulo genera una señal de pulsos con una frecuencia base fijada. La duración del pulso es proporcional a la amplitud de la señal original, es decir, según el valor de la variable de salida el ciclo de trabajo variará entre 0 y 100 %. En la figura 32 se muestra una señal típica PWM con diferentes ciclos de trabajo.

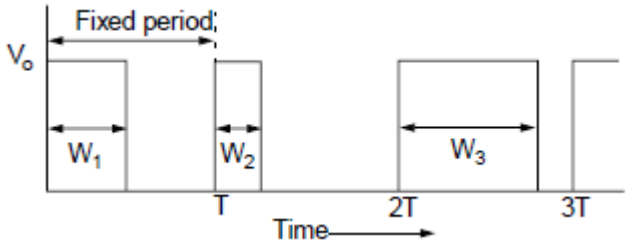


Figura 32. Señal típica PWM.

Esta señal presenta un gran pico en la frecuencia 1/T. Esto es ruido que debe ser eliminado conectando los pines de salida del módulo PWM que estemos utilizando a un filtro paso bajo como muestra la figura 33.

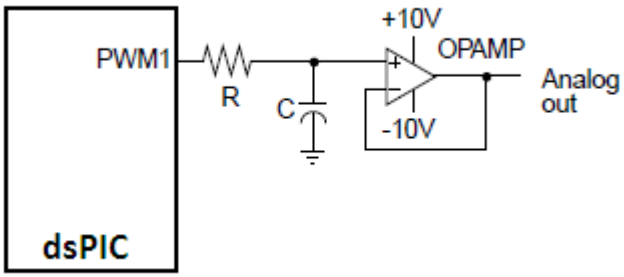


Figura 33. Filtro paso bajo conectado al canal 1 del módulo PWM.

Para utilizar el módulo PWM en nuestro dsPIC comenzaremos habilitando sus interrupciones activando el bit PWMIE, situado en el bit 7 del registro IEC2 y poniendo en funcionamiento el módulo activando el bit PTEN, situado en el bit 15 del registro PTCON. Este registro también contiene los bits PTCKPS, bits 2 y 3, que determinan el prescaler de su temporizador, PTMR. Utilizando PCKPS = 01, que es un prescaler 4, calculamos el número de ciclos necesarios para interrumpir cada 50 ms con la ecuación (45), que resulta en 23032 ciclos. Este valor lo guardamos en el registro PTPER de tal manera que cada vez que el contador del PWM alcance el valor de PTPER genere una interrupción y vuelva a comenzar a contar desde cero.

Si volvemos a la figura 25, donde se mostraban todos los pines de nuestro dsPIC, observamos que los pines de salida del módulo PWM son: PWM1L, PWM1H, PWM2L, PWM2H, PWM3L, PWM3H, PWM4L y PWM4H. Por defecto estos pines son complementarios por parejas, pero para nuestro caso vamos a utilizarlos independientemente. Para ello debemos activar los bits PMOD1, PMOD2, PMOD3 y PMOD4. Finalmente, definimos 4 pines de salida PWM activando los bits PEN1L, PEN2L, PEN3L y PEN4L.

Cada vez que se genera una interrupción, se actualiza el valor de los registros PDC1, PDC2, PDC3 y PDC4, que variaran entre 0 y PTPER. Estos registros definen cuanto tiempo estará activo el bit de salida de cada canal, que se mantendrán hasta el momento en que el registro PTMR supere el valor del registro correspondiente. Este funcionamiento se observa más claramente en la figura 34.

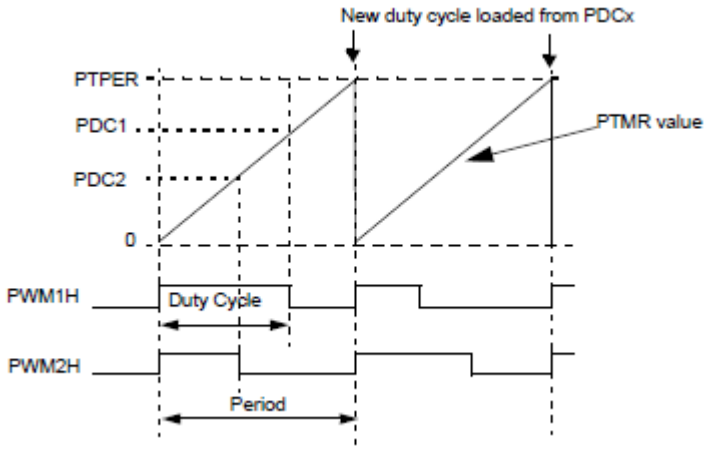


Figura 34. Generación del ciclo de trabajo del módulo PWM.

Para asignar el valor correspondiente a cada registro PDC se utiliza la expresión:

$$PDC = \frac{PTPER}{2} \cdot \left(1 + \frac{Valor\ de\ la\ variable}{Valor\ máximo\ de\ la\ variable}\right) \tag{52}$$

De tal modo, un ciclo de trabajo del 50% equivale al valor 0 de la variable. Ciclos de trabajo inferiores al 50% representan los valores negativos de la variable y los superiores a los positivos. Por último, tras cada interrupción se deberá desactivar el bit PWMIF, situado en el bit 7 del registro IFS2, que se activa cada vez que el registro PTMR toma el valor guardado en PTPER.

6.3 Simulación

Una vez configurados el conversor analógico-digital, el temporizador y el módulo PWM, vamos a realizar una simulación para comprobar que todo funciona correctamente y que las señales generadas son las deseadas. Para esta simulación, se considera un control de velocidad horizontal, PTPER = 23032 y se escriben en un archivo de texto los valores de las señales que queramos cargar en el búfer del conversor. De esta manera, se le asignará el valor correspondiente a cada variable tal como se muestra en la figura 35.

Address	Symbol Name	Value
0876	vxy	11.02192
0872	vz	0.05964775
086E	pitch	0.1721379
086A	roll	-0.1291034
0866	yaw	1.045123
0862	vpitch	0.08848533
085E	vroll	0.1327280
085A	vyaw	0.2875773
0856	ohm	29.79248
0852	vxyd	14.95832
084E	vzd	0.0000000
084A	vpitchd	0.1659100
0846	vrolld	0.0000000
0842	vyawd	0.3133855

Figura 35. Valor de todas las variables del sistema.

Tras la asignación de valores se procede a calcular las cuatro señales de control necesarias. Los resultados se presentan en la figura 36.

Address	Symbol Name	Value
083E	U2	120.0976
083A	U3	-261.8133
0836	U4	1147.203
0832	U1	17713.81

Figura 36. Señales de control calculadas.

Para asegurar que el valor que toman las señales de control son los correctos, se realiza una simulación en Matlab bajo las mismas condiciones. El valor que se asigna a las señales de control en este caso se muestra en la figura 37.

	1	2	3	4
1	17713.80	120.11	-261.82	1147.20

Figura 37. Señales de control calculadas en Matlab.

Comparando los resultados de las figuras 36 y 37 se confirma que el código implementado realiza su función de manera adecuada. Por último, a partir de las señales de control obtenidas, se calcula el número de ciclos que va a estar activo cada canal de salida del módulo PWM.

Address	Hex	Decimal	Symbol Name
01D6	x34F3	13555	PDC1
01D8	x2D86	11654	PDC2
01DA	x2BCE	11214	PDC3
01DC	x2EB4	11956	PDC4

Figura 38. Valor de los registros PDC.

Como se observa en la figura 38, la cantidad de ciclos que se mantiene activa cada salida es proporcional al valor de la señal que se desea representar. El tercer canal de salida, tal y como estaba previsto, es el único que estará activo menos de PTPER/2 ciclos, puesto que está generando una señal de valor negativo.

6.4 Conclusiones del capítulo

En este capítulo se han analizado varias alternativas para la implementación y se ha concluido que la opción más adecuada es un microcontrolador dsPIC. Entre todos los modelos disponibles se ha escogido el dsPIC30F6010A ya que cumple con todos los requisitos. Se ha programado el temporizador y el conversor analógico-digital del dsPIC, así como el módulo PWM, que es el encargado de generar las salidas analógicas del sistema. Finalmente, se ha realizado una simulación para comprobar que las señales de control y de salida se han generado correctamente.

7. Conclusiones

Tras concluir este trabajo hemos visto cuales son los principales pasos para el desarrollo de un vehículo aéreo no tripulado, concretamente para el caso de un autogiro. El objetivo principal era diseñar el algoritmo de control de la aeronave, a fin de implementarlo en un microcontrolador.

En primer lugar, se ha estudiado el movimiento del autogiro. Se ha justificado el uso de la formulación de Lagrange para obtener el modelo dinámico, teniendo en cuenta las fuerzas implicadas, los parámetros físicos y el movimiento generado. A partir de este modelo se han definido las ecuaciones que describen el comportamiento del vehículo en funcionamiento.

A continuación, se ha realizado la simulación numérica en Matlab. Para ello, se ha utilizado el método de Runge-Kutta de cuarto orden como solución a las ecuaciones diferenciales obtenidas. Así, se ha podido evaluar la posición, orientación y velocidad del autogiro en varias situaciones a partir de fuerzas de entrada definidas manualmente.

Seguidamente, se ha diseñado el algoritmo de control mediante la estrategia de linealización por realimentación. De este modo, es el propio sistema el que genera las señales de control necesarias para seguir las velocidades de referencia fijadas con antelación. Se han realizado varias simulaciones y se ha comprobado que el sistema de control funciona correctamente.

Por último, se ha justificado la elección de un microcontrolador dsPIC para la implementación del sistema. Se han configurado y programado los bloques necesarios y, finalmente, se ha simulado la generación de las señales de salida del microcontrolador a partir de valores cargados en la entrada del dispositivo.

El apartado que ha presentado mayor dificultad ha sido la obtención y solución de las ecuaciones dinámicas del autogiro. Han sido necesarias gran cantidad pruebas y una dedicación mayor de la esperada. Esto ha provocado que se haya producido, durante esta etapa, una ligera desviación respecto a la planificación temporal definida previamente al inicio del trabajo. En cuanto al resto de apartados, el tiempo dedicado se ha ajustado en gran medida al que se había previsto.

Bibliografía

[1]

Maria Dina Vivarelli. "Development of spinor descriptions of rotational mechanics from Euler's rigid body displacement theorem". Celestial Mechanics and Dynamical Astronomy, p. 193-207. Editorial Springer. 1984.
Teorema de rotación de Euler y segunda ley de Newton.

[2]

D.A. Wells. "Lagrangian Dynamics". Schaum Outline Series. 1967.
Uso de la formulación de Lagrange para obtener las ecuaciones de movimiento.

[3]

P. Castillo, R. Lozano, A. Dzul. "Modelling and Control of Mini-Flying Machines". Editorial Springer. 2005.
Obtención del modelo dinámico mediante las ecuaciones de Euler-Lagrange.

[4]

Goldstein, H. "Classical Mechanics". Editorial Addison-Wesley. 1980.
Energía cinética y momento cinético del movimiento.

[5]

Robert L. Devaney, Morris W. Hirsch, Stephen Smale. "Differential Equations, Dynamical Systems and an Introduction to Chaos". Academic Press. 2002.
Definición del método Runge-Kutta de cuarto orden.

[6]

Safety Regulation Group - Civil Aviation Authority. "The Aerodynamics of Gyroplanes", Appendix 2. The Stationery Office. 2010.
Parámetros del autogiro VPM M16.

[7]

http://elodieroux.com/EditionsElodieRouxBonus_Matlab.html
Programa diseñado por Élodie Roux que muestra gráficamente la altura y la posición respecto al horizonte a partir de la entrada, que son la altura y los ángulos de alabeo y cabeceo.

[8]

Frederic P Miller, Agnes F Vandome, John McBrewster. "Feedback Linearization: Nonlinear system, Diffeomorphism, Smooth function, Lie derivative". 2012.
Obtención de la señal de control mediante la linealización por realimentación.

[9]

http://www.microchip.com/pagehandler/en-us/family/16bit/
Página web de Microchip, fabricante de dsPICs.

Anexo A: Código del simulador en Matlab

Script Principal:

```
clear all;
variables;
global m l g Ixx Iyy Izz Nb Ib Ax Ay Az Ap Aq Ar U1 U2 U3 U4 JR ohm Q

%parametros de la simulacion:
Ax=0;
Ay=0;
Az=0;
Ap=0;
Aq=0;
Ar=0;
U1=0;
U2=0;
U3=0;
U4=0;
alfaxy=10
alfaz=10
alfapitch=10
alfaroll=10
alfayaw=10
%---------------------------
x0=[0 0 0 0 0 0 0 0 0 0 0 0];%velocidad-posición-velocidad angular-posición angular
tspan=0:0.05:10;
N=length(tspan);
n=length(x0);
ohm=1;
Q=1;
x0=reshape(x0,n,1);
x=[x0 zeros(n,N-1)];
w=x0;

for i=1:N-1
h=tspan(i+1)-tspan(i);
t=tspan(i);
vxyd(i)=0
aceleracionxyd(i)=0
vzd(i)=0
aceleracionzd(i)=0
vpitchd(i)=0
aceleracionpitchd(i)=0
vrolld(i)=0
aceleracionrolld(i)=0
vyawd(i)=0
aceleracionyawd(i)=0
K1=h*feval(@sistemadeecuaciones,t,w);
K2=h*feval(@sistemadeecuaciones,t+h/2,w+K1/2);
K3=h*feval(@sistemadeecuaciones,t+h/2,w+K2/2);
K4=h*feval(@sistemadeecuaciones,t+h,w+K3);
w=w+(K1+2*K2+2*K3+K4)/6;
x(:,i+1)=w;
vxy(i)=sqrt((x(1,i))^2+(x(2,i))^2)
B=(2*cos(x(12,i))*Ax+2*sin(x(12,i))*Ay)
C=-((m*(aceleracionxyd(i)+alfaxy*(vxyd(i)-
vxy(i))))*abs((m*(aceleracionxyd(i)+alfaxy*(vxyd(i)-vxy(i)))))-Ax*abs(Ax)-
Ay*abs(Ay))
U1=((m^2)/(cos(x(10,i))*cos(x(11,i))))*(g-(Az/m)+aceleracionzd(i)+alfaz*(vzd(i)-
(x(3,i))))
if(mod(i,2)~=0)
U1=(-B+sqrt(B^2-4*C))/2
end
a1=((1/m)*((cos(x(12,i)))*sin(x(10,i))*cos(x(11,i))+(sin(x(12,i)))*sin(x(11,i)))*Q+
(1/m)*abs(cos(x(12,i)))*U1*(Nb*Ib)+(Ax/m))
a2=((1/m)*((sin(x(12,i)))*sin(x(10,i))*cos(x(11,i))-
(cos(x(12,i)))*sin(x(11,i)))*Q+(1/m)*abs(sin(x(12,i)))*U1*(Nb*Ib)+(Ay/m))
```

```
Q=(a1)*abs(a1)+(a2)*abs(a2)
ohm= Nb*Ib* vxy(i)
U2=(Ixx/l)*(((Izz-Iyy)*x(8,i)*x(9,i)/Ixx)+(JR*ohm*x(8,i)/Ixx)-
(Ap/Ixx)+aceleracionpitchd(i)+alfapitch*(vpitchd(i)-(x(7,i))))
U3=(Iyy/l)*(((Ixx-Izz)*x(7,i)*x(9,i)/Iyy)-(JR*ohm*x(7,i)/Iyy)-
(Aq/Iyy)+aceleracionrolld(i)+alfaroll*(vrolld(i)-(x(8,i))))
U4=(Izz)*(((Iyy-Ixx)*x(7,i)*x(8,i)/Izz)-
(Ar/Izz)+aceleracionyawd(i)+alfayaw*(vyawd(i)-(x(9,i))))
end

x=x';
t=tspan;

subplot(6,1,1)
plot(t,x(:,4))
ylabel('Pos. X');
subplot(6,1,2)
plot(t,x(:,5))
ylabel('Pos. Y');
subplot(6,1,3)
plot(t,x(:,6))
axis([0 15 290 310 ])
ylabel('Altura');
subplot(6,1,4)
plot(t,x(:,10))
ylabel('Ang. Pitch');
subplot(6,1,5)
plot(t,x(:,11))
ylabel('Ang. Roll');
subplot(6,1,6)
plot(t,x(:,12))
ylabel('Ang. Yaw');
xlabel('Tiempo');
figure(2)
view(3)
patch([-1000 1000 1000 -1000],[1000 1000 -1000 -1000],[0 0 0 0],'FaceColor',[0.8
0.6 0.45])
grid on
hold on
axis([-1000 1000 -1000 1000 -200 1000])
plot3(x(:,4),x(:,5),x(:,6))
xlabel('Pos. X')
ylabel('Pos. Y')
zlabel('Altura')
figure(3)
SimuladorGrafico
   pitch=x(N,10);
   roll=x(N,11);
   altura=x(N,6);
displaymodifie=functionDisplay(pitch,roll,altura)
plot(t,U1t)
figure(4)
plot(t,x(:,1))
figure(5)
plot(t,U1t)
figure(6)
plot(t,vxy)
xlabel('Tiempo')
ylabel('Velocidad eje xy')
figure(7)
plot(t,x(:,3))
xlabel('Tiempo')
ylabel('Velocidad eje z')
```

Variables.m:

```
global m l g Ixx Iyy Izz JR Nb
m=450;
l=4.160;
g=9.81;
Ixx=195;
Iyy=637;
Izz=4425;
JR=51.6;
Ib=46;
Nb=2;
```

Sistemadeecuaciones.m:

```
function xprim = sistemadeecuaciones(t,x);
global m l g Ixx Iyy Izz Ax Ay Az Ap Aq Ar U1 U2 U3 U4 JR ohm Q
xprim=
       [(1/m)*(cos(x(12))*sin(x(10))*cos(x(11))+sin(x(12))*sin(x(11)))*Q+
       +(1/m)*cos(x(12))*U1+(Ax/m);
       (1/m)*(sin(x(12))*sin(x(10))*cos(x(11))-
       cos(x(12))*sin(x(11)))*Q+(1/m)*sin(x(12))*U1+(Ay/m);
       (1/m)*(cos(x(10))*cos(x(11)))*Q+(Az/m)-g;
       x(1);
       x(2);
       x(3);
       ((Iyy-Izz)*x(8)*x(9)/Ixx)-(JR*ohm*x(8)/Ixx)+(l*U2/Ixx)+(Ap/Ixx);
       ((Izz-Ixx)*x(7)*x(9)/Iyy)+(JR*ohm*x(7)/Iyy)+(l*U3/Iyy)+(Aq/Iyy);
       ((Ixx-Iyy)*x(8)*x(7)/Izz)+(1*U4/Izz)+(Ar/Izz);
       x(7)+x(8)*(sin(x(11)))*(tan(x(10)))+x(9)*(cos(x(11)))*(tan(x(10)));
       x(8)*(cos(x(11)))-x(9)*(sin(x(11)));
       x(8)*(sin(x(11)))*(sec(x(10)))+x(9)*(cos(x(11)))*(sec(x(10)));];
```

SimuladorGrafico.m:

```
set(gca,'Xtick',[])
set(gca,'Ytick',[])
set(gca,'Position',[0.05 0 0.77 1])
Display.Hauteur=1;
Display.Largeur=Display.Hauteur;
       r1=3*Display.Largeur/10;
       r2=r1+Display.Largeur/30;
   r3=r1+Display.Largeur/20;
       Display.PasAssiette=r1*180/(pi*25);

   axis([-Display.Largeur/2 Display.Largeur/2 -Display.Hauteur/2
Display.Hauteur/2])
Display.Ciel=patch([-1;-1;1;1]*Display.Largeur/2,[-1;1;1;-
1]*Display.Largeur/2,'b');
set(Display.Ciel,'FaceColor',[0.3 0.6 1])
Display.Terre=patch([-1;-1;1;1]*Display.Largeur/2,[0;-1;-
1;0]*Display.Largeur/2,'r');
set(Display.Terre,'FaceColor',[0.8 0.6 0.4])

petitebarre=(1/2)*Display.Largeur/10;
moyennebarre=Display.Largeur/10;
grandebarre=2*Display.Largeur/10;
RefTerreX=[petitebarre moyennebarre petitebarre grandebarre moyennebarre
moyennebarre]/2;
RefTerreX=[-RefTerreX;RefTerreX];
RefTerreX=[RefTerreX RefTerreX];
RefTerreY=[2.5:2.5:10 15 20]*Display.PasAssiette*pi/180;
RefTerreY=[RefTerreY;RefTerreY];
RefTerreY=[RefTerreY -RefTerreY];
Display.PitchBar=line(RefTerreX,RefTerreY);
```

```
set(Display.PitchBar,'Color','w')
set(Display.PitchBar,'LineWidth',2)
TextX=[1.1 1.1 -1.45 -1.45]*grandebarre/2;
TextY=[1 -1 1 -1]*10*pi/180*Display.PasAssiette;
TextT={'-10°';'10°';'-10°';'10°'};
Display.PitchBartext=text(TextX,TextY,TextT);
set(Display.PitchBartext,'Color','w')
clear petitebarre & moyennebarre & grandebarre & RefTerreX & RefTerreY & TextX &
TextY & TextT

RoseX=[r1*cos(pi/6) r1*cos(pi/4) r1*cos(pi/3) r1*cos(7*pi/18) r1*cos(4*pi/9);
               r3*cos(pi/6) r2*cos(pi/4) r3*cos(pi/3) r2*cos(7*pi/18)
r2*cos(4*pi/9)];
RoseY=[r1*sin(pi/6) r1*sin(pi/4) r1*sin(pi/3) r1*sin(7*pi/18) r1*sin(4*pi/9);
               r3*sin(pi/6) r2*sin(pi/4) r3*sin(pi/3) r2*sin(7*pi/18)
r2*sin(4*pi/9)];
RoseX=[RoseX [0;0] -RoseX];
RoseY=[RoseY [r1;r3] RoseY];
Display.Rosasse=line(RoseX,RoseY)
set(Display.Rosasse,'Color','w')
set(Display.Rosasse,'LineWidth',2)
LargeurF=(1/3)*(Display.Largeur/10);
Display.RosasseFleche=patch([0 1 -1]*LargeurF/2,r1-LargeurF*sin(60*pi/180)*[0 1
1],'w');
set(Display.RosasseFleche,'EdgeColor','w')
Display.RefRosasseFleche=[get(Display.RosasseFleche,'Xdata')';get(Display.RosasseFl
eche,'Ydata')'];
Display.RosasseFlecheFixe=patch([0 1 -1]*LargeurF/2,r1+LargeurF*sin(60*pi/180)*[0 1
1],'w');
set(Display.RosasseFlecheFixe,'EdgeColor','w')
clear RoseX & RoseY
clear r1 & r2 & r3 & LargeurF
Display.AltitudeBox=patch(-
[0;1.6*Display.Largeur/10;1.6*Display.Largeur/10;0]+Display.Largeur/2,[1;1;-1;-
1]*Display.Largeur/40,'k')
Display.AltitudeValue=text(-
1.6*Display.Largeur/10+Display.Largeur/2+5*Display.Largeur/1000,-
25*Display.Largeur/10000,'00000')
set(Display.AltitudeValue,'Color','w')
set(Display.AltitudeValue,'Fontsize',18)
Display.AileDroite=line([Display.Largeur/10 Display.Largeur/10
2*Display.Largeur/10],[-0.5*Display.Hauteur/10 0 0]);
Display.AileGauche=line([-Display.Largeur/10 -Display.Largeur/10 -
2*Display.Largeur/10],[-0.5*Display.Hauteur/10 0 0]);
Display.Cockpit=line([-0.1*Display.Largeur/10 0.1*Display.Largeur/10],[0 0])
set(Display.AileDroite,'LineWidth',4)
set(Display.AileGauche,'LineWidth',4)
set(Display.Cockpit,'LineWidth',4)
set(Display.AileDroite,'Color','k')
set(Display.AileGauche,'Color','k')
set(Display.Cockpit,'Color','k')

global Display
```

functionDisplay.m:

```
function             displaymodifie=functionDisplay(pitch,roll,altura)
theta=pitch
phi=roll
altitude=altura

global Display
       theta=functionRameneAnglepi(theta);
       phi=functionRameneAnglepi(phi);
       h=round(altitude);
if(h/10000<1)
   if(h/1000<1)
      if(h/100<1)
         if(h/10<1)
            set(Display.AltitudeValue,'String',['        ' num2str(round(h))])
         else
              set(Display.AltitudeValue,'String',['      ' num2str(round(h))])
             end
          else
          set(Display.AltitudeValue,'String',['    ' num2str(round(h))])
              end
       else
       set(Display.AltitudeValue,'String',['  ' num2str(round(h))])
   end
else
   set(Display.AltitudeValue,'String',[num2str(round(h))])

end
Mrot=[cos(phi)       -sin(phi);
          sin(phi)   cos(phi)];
Mtr=Mrot*[0;Display.PasAssiette];
Pos=Mrot*[Display.RefRosasseFleche];
set(Display.RosasseFleche,'Xdata',Pos(1,:))
set(Display.RosasseFleche,'Ydata',Pos(2,:))
clear Pos
if(phi<pi/2 & phi>-pi/2)==1
   set(Display.Terre,'Xdata',Display.Largeur/2*[1 1 -1 -1])
   set(Display.Terre,'Ydata',[Display.Largeur/2*tan(phi)-
(theta*180/pi)/41.5*Display.Hauteur/2 -Display.Hauteur/2 -Display.Hauteur/2 -
Display.Largeur/2*tan(phi)-(theta*180/pi)/41.5*Display.Hauteur/2])
elseif(phi==-pi/2)
   set(Display.Terre,'Xdata',Display.Largeur/2*[0 -1 -1 0])
   set(Display.Terre,'Ydata',Display.Hauteur/2*[-1 -1 1 1])
elseif(phi==pi/2)
   set(Display.Terre,'Xdata',Display.Largeur/2*[0 1 1 0])
   set(Display.Terre,'Ydata',Display.Hauteur/2*[1 1 -1 -1])
else
       set(Display.Terre,'Xdata',Display.Largeur/2*[1 1 -1 -1])
       set(Display.Terre,'Ydata',[Display.Largeur/2*tan(phi)-
(theta*180/pi)/41.5*Display.Hauteur/2 Display.Hauteur/2 Display.Hauteur/2 -
Display.Largeur/2*tan(phi)-(theta*180/pi)/41.5*Display.Hauteur/2])
end
```

Anexo B: Código C de la implementación del dsPIC

```
#include "p30fxxxx.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.141516
#define m 450
#define l 4.16
#define g 9.81
#define Ixx 195
#define Iyy 637
#define Izz 4425
#define JR 51.6
#define vmax 44.7
#define vzmax 3.175
#define vrotacmax 9.42
#define vohmmax 39.77
#define U1max 100000
#define U2max 10000
#define U3max 10000
#define U4max 30000
#define alfaxy 10
#define alfaz 10
#define alfapitch 10
#define alfaroll 10
#define alfayaw 10
float ConvertBuffer(float);

float U1;
float U2;
float U3;
float U4;
int inter = 0;
int s = 1;

void ini_ADC( void )
{
IPC2bits.ADIP = 0x0003; // Nivel de prioridad del conversor = 3.
IFS0bits.ADIF = 0 ;     // Desactivar el bit de flag del conversor.
IEC0bits.ADIE = 1;      // Habilitar interrupciones del conversor.
ADPCFG = 0xC000;        // Los 14 primeros pines son entradas analogicas. Los otros
                        // dos son pines digitales de entrada/salida.
TRISB = 0xFFFF;         // Se activa el bit TRIS de los 16 pines como entrada.
ADCSSL = 0x3FFF;        // Se activan los bits correspondientes a los 14 pines
                        // para formar parte de la secuencia del conversor.
ADCON1 = 0x03E0;        // SSRC bit = 111 para que el Trigger que detiene el
                        // muestreo e inicia la conversión sea automático.
                        // FORM bit = 11 formato fraccional y con signo
ADCON1bits.ASAM = 1;    // Comenzar el muestreo tras finalizar una conversión.
ADCON2 = 0x0434;        // SMPI bit = 1110 para interrumpir tras 14 muestreos.
                        // CSCNA bit = 1 para activar la lectura de los pines.
                        // VrefH y VrefL Son AVdd y AVss, Buffer de 16 bits.
ADCON3 = 0x0100;        // ADCS bit = 000000, SAMC bit = 00001
ADCON1bits.ADON = 1;    // ADC ON.
}

void ini_T1( void )
{
IPC0bits.T1IP = 0x0002; // Nivel de prioridad del temporizador = 2.
IFS0bits.T1IF = 0 ;     // Desactivar el bit de flag del temporizador.
IEC0bits.T1IE = 1 ;     // Habilitar interrupciones del Timer1
PR1 = 11516;            // Periodo de 11516 ciclos para interumpir cada 0.05 seg.
T1CON = 0x8010;         // Registro de control de 16 bits del Timer1:
                        // TON=1, PRESCALER=01 (8), TCS=0.
}

void __attribute__((interrupt, shadow, auto_psv)) _T1Interrupt()
{
while (!IFS0bits.ADIF); // Esperar la lectura de los 15 pines y active el flag.
```

```
float vxy=ConvertBuffer(ADCBUF0)*vmax*1.2;  // Convierte y escala el valor
float vz=ConvertBuffer(ADCBUF1)*vzmax*1.2;
float pitch=ConvertBuffer(ADCBUF2)*PI;
float roll=ConvertBuffer(ADCBUF3)*PI;
float yaw=ConvertBuffer(ADCBUF4)*PI;
float vpitch=ConvertBuffer(ADCBUF5)*vrotacmax*1.2;
float vroll=ConvertBuffer(ADCBUF6)*vrotacmax*1.2;
float vyaw=ConvertBuffer(ADCBUF7)*vrotacmax*1.2;
float ohm=ConvertBuffer(ADCBUF8)*vohmmax*1.2;
float vxyd=ConvertBuffer(ADCBUF9)*vmax;
float vzd=ConvertBuffer(ADCBUFA)*vzmax;
float vpitchd=ConvertBuffer(ADCBUFB)*vrotacmax;
float vrolld=ConvertBuffer(ADCBUFC)*vrotacmax;
float vyawd=ConvertBuffer(ADCBUFD)*vrotacmax;

if((vxyd-vxy)>0)
       {
              s = 1;
       }

else
       {
       s = -1;
       }
if(PORTBbits.RB15==0 && PORTBbits.RB14==0 //Bits RB15 y RB14 determinan si el
                                          // control de U1 será de avance
                                          // horizontal, vertical o alterno.
{
U1 =(sqrt(4*(pow(m*(alfaxy*(vxyd-vxy)),2))))/2;
}
else if(PORTBbits.RB15==0 && PORTBbits.RB14==1)
{
U1 =((m*m)/cos(pitch)*cos(roll))*(g+alfaz*(vzd-vz));
}
else
{
if (inter==0) {
U1 =(sqrt(4*(pow(m*(alfaxy*(vxyd-vxy)),2))))/2;
inter = 1;
}
else {
U1 =((m*m)/cos(pitch)*cos(roll))*(g+alfaz*(vzd-vz));
inter = 0;
}
}
U2 =(Ixx/l)*(((Izz-Iyy)*vroll*vyaw/Ixx)+(JR*ohm*vroll/Ixx)+alfapitch*(vpitchd-vpitch));
U3 =(Iyy/l)*(((Ixx-Izz)*vpitch*vyaw/Iyy)-(JR*ohm*vpitch/Iyy)+alfaroll*(vrolld-vroll));
U4 =(Izz)*(((Iyy-Ixx)*vpitch*vroll/Izz)+alfayaw*(vyawd-vyaw));

IFS0 = IFS0 & 0xF7F7;  // Se desactivan los bits de flag del Timer1 y del CAD
}
void ini_pwm( void )
{
IPC9bits.PWMIP = 0x0001; // Nivel de prioridad del conversor = 1.
IFS2bits.PWMIF = 0 ;     // Desactivar el bit de flag del PWM.
IEC2bits.PWMIE = 1 ;     // Habilitar interrupciones del PWM.
PTCONbits.PTCKPS = 1;    // Prescaler = 01 (4)
PTPER = 23032;           // Periodo de 0.05 segundos.
PWMCON1bits.PMOD1 = 1 ;  // Uso independente del PWM
PWMCON1bits.PMOD2 = 1 ;
PWMCON1bits.PMOD3 = 1 ;
PWMCON1bits.PMOD4 = 1 ;
PWMCON1bits.PEN1L = 0 ;  // Los pines Low usados como salidas del PWM.
PWMCON1bits.PEN2L = 0 ;
PWMCON1bits.PEN3L = 0 ;
PWMCON1bits.PEN4L = 0 ;
PTCONbits.PTEN = 1;      // PWM ON.
}
void __attribute__((__interrupt__,__no_auto_psv__)) _PWMInterrupt(void)
{
PDC1 = (PTPER/2)*(1+(U1/U1max));  // Valor del Duty Cycle. 50% -> U1=0.
PDC2 = (PTPER/2)*(1+(U2/U2max));
```

```
PDC3 = (PTPER/2)*(1+(U3/U3max));
PDC4 = (PTPER/2)*(1+(U4/U4max));
IFS2bits.PWMIF = 0 ;              // Desactivar el bit de flag del PWM.
}
float ConvertBuffer(float buf)
{
buf = buf/64;            // Se eliminan los 6 ceros insignificantes de la derecha.
if (buf > 511){          // El bit más significativo nos indica el signo.
buf = (buf-512)/511;     // Se elimina el valor de ese bit.
return buf;
}

else {
buf = -buf/511;
return buf;
}
}

int main()

{
ini_T1();     // Inicializa el temporizador 1.
ini_ADC();    // Inicializa el conversor analógico digital.
ini_pwm();    // Inicializa el modulo PWM.
while(1);     // Bucle infinito.
}
```

Datosbufer.txt:

```
0x0069
0x0008
0x001C
0x0215
0x00AA
0x0004
0x0006
0x000D
0x013F
0x00AB
0x0000
0x0009
0x0000
0x0011
```

Resum:

Aquest projecte consisteix en dissenyar l'algoritme de control d'un autogir no tripulat. La seva aplicació principal es d'ur a terme tasques rutinàries o perilloses per al pilot com, per exemple, extinció d'incendis, avaluació de risc químic o vigilància d'indrets d'accés restringit.

Es realitza un estudi del moviment del vehicle per a obtenir el seu model dinàmic. A partir de les equacions que descriuen el seu moviment, es realitza una simulació numèrica del vehicle. S'incorpora el controlador dissenyat i s'avalua el seu funcionament. Finalment, s'implementa el sistema en un microcontrolador.

Resumen:

Este proyecto consiste en diseñar el algoritmo de control de un autogiro no tripulado. Su aplicación principal es llevar a cabo tareas rutinarias o peligrosas para el piloto como, por ejemplo, extinción de incendios, evaluación de riesgo químico o vigilancia de lugares de acceso restringido.

Se realiza un estudio del movimiento del vehículo para obtener su modelo dinámico. A partir de las ecuaciones que describen su movimiento, se realiza una simulación numérica del vehículo. Se incorpora el controlador diseñado y se evalúa su funcionamiento. Finalmente, se implementa el sistema en un microcontrolador.

Summary:

This project consists of designing a control algorithm for an unmanned autogyro. Its principal application is to carry out routine or dangerous tasks for the pilot, such as firefighting, chemical risk assessment or monitoring restricted access areas.

A study of vehicle motion is done in order to obtain its dynamic model. From the equations that describe its motion, a numerical simulation of the vehicle is done. The designed controller is added and its performance is evaluated. Finally, the system is implemented in a microcontroller.