Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un autopiloto de un quadcopter.

Autor

Diego Luis Marco

Directores

José Luis Villarroel Salcedo
Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Zaragoza, Noviembre de 2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Diego Luis Marco,

con nº de DNI 73017931-S en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Electrónica y Automática, (Título del Trabajo) Desarrollo de un Autopiloto de un quadcopter.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de Noviembre de 2015

Fdo: Diego Luis Marco

A mis padres y hermana, por su apoyo incondicional.

RESUMEN

DESARROLLO DE UN AUTOPILOTO DE UN QUADCOPTER

Los drones o UAV son unas de las tecnologías más de moda en estos últimos años. El auge de esta tecnología se debe a la reducción de su precio y a las múltiples aplicaciones que se han desarrollado.

En este Trabajo Fin de Grado se va a diseñar e implementar un autopiloto para un dron de gama media, el F450. Este autopiloto, además de estabilizar el vuelo, va a tener 2 funcionalidades diferentes: seguimiento de referencias en posición mediante órdenes de un control de navegación y seguimiento de referencias en velocidad dadas por un mando de radiofrecuencia. Lo que se busca es obtener una primera experiencia con multihélices para permitir posteriores desarrollos más complejos.

Para conseguir estos objetivos se han realizado simulaciones mediante la herramienta matemática Matlab-Simulink para poder validar los controladores diseñados. Estas simulaciones son vitales para hacernos a la idea de lo complejo que es el sistema y conseguir conocerlo perfectamente antes de empezar a utilizar el quadcopter real.

Se analiza el hardware impuesto, como sensores o sistemas de comunicaciones, y se integra para obtener un sistema lo más robusto y completo posible. Además se ha realizado un sistema de tiempo real basado en tareas, estando éste implementado sobre el sistema operativo de SYS/BIOS. El procesador utilizado es el F2812 que es un microcontrolador de gama media (150 MHz, 32 bits, arquitectura harvard, CPU en coma fija...). Procesador y sistema operativo así como el entorno de desarrollo utilizado son de Texas Instruments.

Se ha implementado el control de velocidad y se han conseguido realizar pequeños vuelos en interior, consiguiendo la estabilización del quadcopter en el aire.

Índice general

Capítulo 1

Introducción

1.1. Contexto y estado del arte

Un dron es un vehículo aéreo no tripulado reutilizable. Éste a través de múltiples sensores es capaz de mantener un vuelo controlado y sostenido mediante la propulsión de sus hélices o por un motor de explosión o reacción.

Los primeros drones surgieron en 1916, los cuales eran controlados mediante radiofrecuencia y su uso era el de afinar puntería de la artillería antiaérea.

Figura 1.1: Uno de los primeros drones de la historia, el Queen Bee. [5]

La evolución de los drones fue de la mano de la evolución de los misiles hasta el punto de que la diferencia más significativa entre estos dos fuera que los drones son reutilizables.

En la actualidad tenemos drones de muy diferentes configuraciones como drones con forma de avión, helicóptero o multirotor (como en el caso que nos ocupa, que es un quadrotor).

También cabe destacar la diferencia en cuanto a tamaños, por ejemplo, el PD 100-PRS de Prox Dynamics tiene un peso de 18 g y una hélice de 120 mm y en cambio el MQ-9

Reaper mide 86 m de largo y es usado por los ejércitos de Estados Unidos, Reino Unido o España. Éste tiene un precio de 30 millones de euros.

(a) Dron PD 100-PRS de Prox Dynamics. [4]

(b) MQ-9 Reaper.

Figura 1.2: La variedad de tipos de drones.

Sin embargo, no todos los drones son utilizados con fines militares, hoy en día existen muchísimos drones de uso civil.

Sus aplicaciones son muy diversas, desde propósitos profesionales como pueden ser la búsqueda de personas desaparecidas, cartografía aérea, prevención de incendios, seguridad en eventos, control de cultivos o usos más lúdicos, como podrían ser filmaciones o su uso recreativo.

La parte más compleja del quadcopter, y que suele ser común en muchos de ellos, es el autopiloto, que es la parte central de este TFG. Existen pocos autopilotos en el mercado hoy día y la gran mayoría de ellos son de código cerrado.

Por ejemplo, uno de los autopilotos más usados es el Naza M lite de DJI de código cerrado. Este autopiloto es, dentro del grupo de los más económicos, de los que mejores prestaciones ofrecen. Integra 4 sensores que son necesarios para poder realizar el control de estabilización. Estos son: un giróscopo, un acelerómetro, una brújula magnética y un barómetro. También tiene la posibilidad de incorporarle un GPS para mejorar su localización. Puede llegar a alcanzar velocidades de ascenso o descenso de $\pm$ 6 m/s, un ángulo de inclinación de 45° y una velocidad de inclinación de 200°/s. Si se incluye el GPS tiene una localización con un error en vertical de 0.8 m y en horizontal 2.5 m. Además tiene 3 modos diferentes de vuelo: manual, automático (donde tiene un control de estabilización de ángulos) y el modo con GPS.

Existe un autopiloto de código abierto, muy conocido, que es Ardupilot. Éste incorpora también barómetro, giróscopo, acelerómetro, brújula digital y GPS. Este software abierto te permite el control y navegación mediante posiciones del GPS de múltiples robots como son los mostrados en la figura:

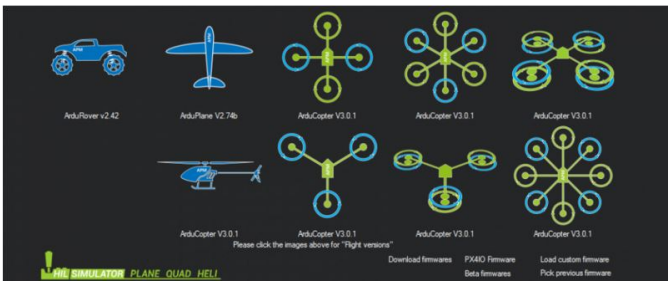


Figura 1.3: Posibles robots para controlador Ardupilot. [1]

¿Por qué diseñar uno propio viendo que en el mercado disponemos de distintas soluciones? La razón es que se busca tener un control total de todo lo que ocurre en el dron y esto pasa por construir desde cero el autopiloto. Además se consigue poder implementar controles más sofisticados ya que se controlan todas las variables que influyen en el quadrotor y se pueden probar nuevas estructuras de control.

Este proyecto se lleva a cabo dentro del grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. Éste es uno de los grupos de investigación del Instituto Universitario de Investigación en Ingeniería de Aragón (I3A) y es considerado Grupo de Investigación por el Gobierno de Aragón. Dicho grupo tiene las siguientes líneas de trabajo:

- Localización y Mapeado Simultáneo.
- Visión por Computador y Percepción.
- Comunicaciones y redes ad-hoc.
- Exoesqueletos y procesamiento de bioseñales.
- Aprendizaje: en robótica, optimización Bayesiana, interfaces cerebro-ordenador...
- Robótica Móvil. Planificación y navegación.

Uno de los objetivos del grupo es pasar de la robótica móvil terrestre a la aérea. Por ello se realiza este TFG como comienzo de esa nueva línea de investigación.

Una de las aplicaciones que se quiere conseguir a largo plazo es la búsqueda de personas atrapadas bajo una alud. Un dron o varios peinarán la zona en busca de una señal emitida por un gadget que llevará el montañero.

Otro de los proyectos es la reconstrucción de un mapa 3D de una cueva a través de sensores laser incluidos en el dron.

Para conseguir todo esto, se comienza consiguiendo controlar el dron.

1.2. Objetivos

El objetivo principal es diseñar e implementar un prototipo de autopiloto que controle el movimiento de un quadcopter FlameWheel450.

Figura 1.4: Dron F450 de DJI.

No se busca crear un autopiloto comercial si no tener una primera experiencia en la problemática de control y estabilización del vuelo de un multihélice.

El autopiloto tendrá dos modos de vuelo, lo que supone dos subobjetivos:

- Control de movimiento del quadcopter mediante referencias de posición, es decir, que pueda alcanzar cualquier localización en el espacio y puedan dichas posiciones ser usadas como *waypoints*[1] para controles superiores sobre este autopiloto.
- Reproducir el uso clásico del quadcopter. Esto significa que, a través de un mando de radiofrecuencia, se pueda manejar el dron imponiéndole referencias de velocidad sin tener ningún control de la posición en la que está más que la realimentación visual del usuario.

1.3. Organización de la memoria

Este trabajo está dividido en 5 capítulos, siendo el primero de ellos esta introducción, que se detallan a continuación:

- Plataforma hardware y software: Se utiliza un harware y software impuesto por el grupo de investigación de robótica. En este apartado se analiza y explica cómo se ha integrado cada uno de los componentes que se incluyen en el quadcopter. Además se explican la plataforma software utilizada junto con el sistema operativo de tiempo real.

[1] Puntos de referencia calculados para dividir una trayectoria compleja, que el quad debe seguir, en varias trayectorias sencillas.

- Análisis y diseño: En este capítulo se analiza el modelo matemático del quadcopter y se explica las estructuras de control utilizadas.
- Implementación: Este apartado es el grueso del trabajo. Es aquí donde se explica cómo se ha implementado todo el harware en el DSP utilizado y las distintas funcionalidades que el software desarrollado nos ofrece. Además se analiza el sistema de tiempo real verificando su determinismo temporal.
- Pruebas en el quadcopter: Durante el transcurso del trabajo se han realizado muchas pruebas a partir de las simulaciones previas a éstas. En este apartado se describen algunas de las más significativas.
- Conclusiones: Al final del presente trabajo se recogen las conclusiones obtenidas de este TFG.

Capítulo 2

Plataformas hardware y software

En este capítulo se describe y explica cómo se ha integrado el hardware en el sistema, la plataforma software utilizada y el sistema operativo que tenemos en el DSP.

2.1. Hardware

En la siguiente imagen se puede ver un esquema del hardware utilizado y sus conexiones con el sistema:

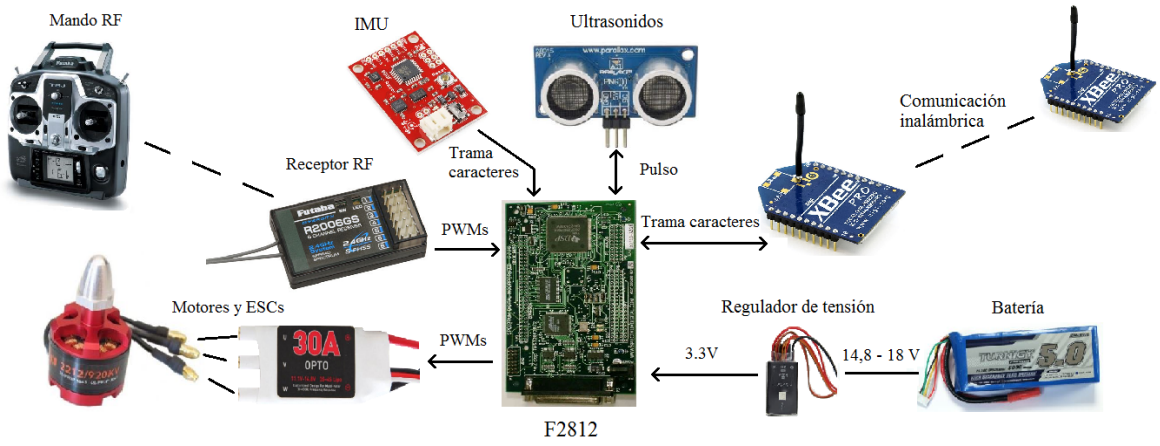


Figura 2.1: Diagrama de bloques del hardware.

2.1.1. Estructura mecánica

La estructura mecánica donde se coloca todo el hardware se ha comprado a DJI [11]. Son cuatro barras reticuladas que forman la estructura principal del quadcopter. Dos son

de color rojo y dos de color blanco. El rojo indica la parte delantera del quadcopter ya que es totalmente simétrico.

Contiene también dos placas centrales donde va situada toda la electrónica principal del dron. En la placa inferior tiene integradas pistas para la alimentación y masa. Además incluye 4 patas y 4 hélices. Las hélices son diferentes 2 a 2 ya que el ángulo de ataque de hélices contiguas debe ser el contrario.

Figura 2.2: Estructura quad. [3].

2.1.2. DSP F2812

Es un microcontrolador específico para el procesado digital de señales en tiempo real. Por ello dispone de una frecuencia de bus de 150MHz, la cual es suficientemente alta. Tiene arquitectura *harvard*[1] con 2 buses de datos de 32 bits y uno de programa de 22 bits.

Algunas de las características más usadas del micro son los *Event Managers* para leer sensores y actuar sobre los motores y la comunicación serie SCI, la cual sirve para comunicarse con los sensores que lo permitan y por donde se manda la telemetría. Las interrupciones vectorizadas con prioridades se utilizan para guardar los datos de los sensores justo cuando han llegado.

2.1.3. Sensores

Los sensores son necesarios a la hora de tener un sistema de control ya que debemos saber el estado de las variables controladas. Para la realización de este proyecto disponemos de dos sensores:

[1]La arquitectura harvard se basa en tener separado el banco de memoria de programa y de datos accediendo a ellos por buses diferentes.

El primero es el sensor Ping de Parallax que mide distancia hacia un objeto situado enfrente de él. Su rango de medida es desde 2 centímetros a 3 metros. Su funcionamiento es muy sencillo, si pones un pulso en su patilla central, él te devuelve uno proporcional a la distancia que mide. Ésta se recoge con el *Input Capture*[2].

El segundo es el IMU (unidad de medida inercial) modelo 9 Degrees Of Freedom - Razor IMU, el cual es el sensor principal de esta aplicación. Realmente es un conjunto de 3 sensores diferentes que contiene: un giróscopo, una brújula magnética y un acelerómetro. Cada uno de ellos proporciona una medida para cada eje (XYZ). Estos sensores proporcionan la velocidad angular, el polo norte magnético y la aceleración respectivamente. Además contiene un micro (AtMega328) que calcula los ángulos y emite una trama de caracteres con los datos por SCI.

2.1.4. ESCs y Motores

Los ESC son controladores electrónicos de velocidad de motores. El modelo utilizado es 30A OPTO ESC de DJI. Se necesita uno por cada motor, donde éstos a través de un PWM que es mandado por el DSP generan la tensión y corriente necesarias en las tres fases que alimentan el motor para conseguir la velocidad. Los motores son el modelo 2212 920KV Brushless Motor de DJI. Su velocidad máxima es de 15416 rpm.

2.1.5. Batería y regulador de tensión

La batería utilizada es el modelo Turnigy 5.0. Dicha batería es de 4 celdas y tiene una carga mínima de 14,8 voltios. Su capacidad es de 5000 mAh, lo que conlleva un vuelo medio entre 10 y 12 minutos. Pesa 552 g y tiene unas dimensiones de 149 x 49 x 33 mm.

También se utiliza un regulador de tensión de 5V para alimentar el DSP y después éste alimenta los sensores a 3,3 V.

2.1.6. Xbee

Es un módulo de comunicación serie inalámbrica. Sirve para poder comunicarse con el dron desde un ordenador. Como característica más destacable tiene la distancia máxima a la que se pueden enviar datos siendo en interiores o núcleos urbanos de 550 m y en campo abierto hasta 40 km.

Su velocidad de transmisión de datos es de 24 mil bits por segundo, suficiente para las comunicaciones realizadas.

[2]Funcionalidad del F2812 que a través del *Event Manager* es capaz de medir el tiempo de un pulso.

2.1.7. Mando y receptor de radiofrecuencia

El mando de radiofrecuencia es el modelo T6J de Futaba. Emite a 2.4 GHz y dispone de 6 canales de transmisión. Éste tiene dos sticks, el derecho con autoretorno en las dos direcciones y el izquierdo con autoretorno sólo en la horizontal, siendo la vertical la utilizada para la potencia total del quadrotor.

Figura 2.3: Controles del mando de radiofrecuencia. [2] .

El receptor de radiofrecuencia es el modelo R2006GS y dispone también de 6 canales recibiendo a 2.4 GHz.

2.1.8. Multiplexor

El multiplexor modelo 4-Channel RC Servo Multiplexer está diseñado para aplicaciones como la que se quiere utilizar. Es capaz de multiplexar dos entradas de 4 canales a través de un 5 º canal y obtener la salida que se seleccione con este último.

A la hora de testear el sistema y realizar pruebas de vuelo se tiene la posibilidad de, rápidamente, cambiando el switch auxiliar, cambiar al autopiloto industrial que se tiene instalado en el quadcopter. Éste es el autopiloto Naza M Lite detallado en la sección 1.1.

Por lo tanto es una medida de seguridad para la realización de pruebas.

2.2. Software

El software que se explica en este capítulo no es el desarrollado en este TFG si no la plataforma usada de programación y el sistema operativo de tiempo real utilizado.

2.2.1. Code Composer Studio

Es un entorno de desarrollo integrado de aplicaciones de Texas Instruments y es de los más completos en el ámbito de los procesadores digitales. Tiene editor, *debugger*, compilador y *linker*.

Se pueden programar microprocesadores tales como MSP430, Stellaris, C6000, C55x o C28x. Se ha utilizado la versión 5.4.

2.2.2. SYS/BIOS

SYS/BIOS es un núcleo de tiempo real de Texas Instruments. Está integrado en Code Composer Studio y es de código abierto. Permite una planificación del procesador basada en prioridades fijas y herencia de prioridad y tiene herramientas de análisis de tiempo real como: tiempos de cómputo, gráficos de ejecucción, carga del procesador, instrumentalización del programa... Elementos que forman SYS/BIOS son: las interrupciones hardware, interrupciones software, tareas, servidores o semáforos. [6]

Capítulo 3

Análisis y diseño

El sistema real con el que se parte es el FlameWheel 450 [11]. Un dron de cuatro hélices con alimentación eléctrica y de 70 cm de longitud que se puede ver en la figura 1.4. Para realizar el control es necesario obtener su modelo matemático.

3.1. Modelo del quadcopter

El modelo del quadcopter se basa en las ecuaciones obtenidas en [7]:

$$\text{Roll} \Rightarrow \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{(I_{yy} - I_{zz})}{I_{xx}} + \frac{L}{I_{xx}}U_2 + \frac{I_{yy}}{I_{xx}}\dot{\theta}p \tag{3.1}$$

$$\text{Pitch} \Rightarrow \ddot{\theta} = \dot{\phi}\dot{\psi}\frac{(I_{zz} - I_{xx})}{I_{yy}} + \frac{L}{I_{yy}}U_3 - \frac{I_{xx}}{I_{yy}}\dot{\phi}p \tag{3.2}$$

$$\text{Yaw} \Rightarrow \ddot{\psi} = \frac{1}{I_{zz}}U_4 + \frac{J_r}{I_{zz}}\dot{p} \tag{3.3}$$

$$\text{Z} \Rightarrow \ddot{Z} = -g + (\cos\theta\cos\phi)\frac{1}{m}U_1 \tag{3.4}$$

$$\text{X} \Rightarrow \ddot{X} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\frac{1}{m}U_1 \tag{3.5}$$

$$\text{Y} \Rightarrow \ddot{Y} = (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\frac{1}{m}U_1 \tag{3.6}$$

- X, $\dot{X}$, $\ddot{X}$: Posición en la dirección de avance del quadcopter. La "cabeza"del quadcopter es la parte en la que sus brazos son rojos. La derivada es la velocidad lineal en el eje X y la segunda derivada es la aceleración lineal en dicho eje.

- Y, $\dot{Y}$, $\ddot{Y}$: Posición en la dirección transversal del quadcopter. La izquierda de su "cabeza"es el sentido positivo. La derivada de Y es su velocidad y la segunda derivada la aceleración en este eje.
- Z, $\dot{Z}$, $\ddot{Z}$: Posición respecto a la altura del quad. Su derivada es la velocidad y la segunda derivada la aceleración en el eje Z.
- θ, $\dot{\theta}$, $\ddot{\theta}$: Ángulo Roll que corresponde a la rotación del eje X. Su derivada es la velocidad ángular y la segunda derivada la aceleración angular.
- ϕ, $\dot{\phi}$, $\ddot{\phi}$: Ángulo Pitch que corresponde a la rotación del eje Y. Su derivada es la velocidad ángular y la segunda derivada la aceleración angular.
- ψ, $\dot{\psi}$, $\ddot{\psi}$: Ángulo Yaw que corresponde a la rotación del eje Z. Su derivada es la velocidad ángular y la segunda la aceleración angular.

En la siguiente figura se pueden ver representados dichos ejes y ángulos.

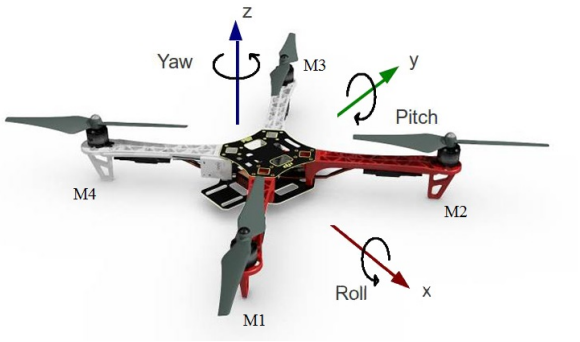


Figura 3.1: Representación de ejes y giros.

También aparecen términos físicos en las ecuaciones:

- I_{xx}, I_{yy}, I_{zz}: Son los términos correspondientes a la inercia y aparecen representados con I y el subíndice que corresponde al eje respecto al que es tomada. Éstas han sido calculadas a través de las ecuaciones A.6, A.7 y A.10.
- L: Longitud de un brazo del quadcopter.
- J_r: Momento de inercia de los rotores. Se calcula siguiendo la ecuación A.11.
- p: Perturbación que se genera por la diferencia de velocidades ángulares de los motores para una misma referencia debido a su no idealidad. También aparece su derivada.

- m: Masa total del quadcopter.
- g: Gravedad terrestre.

Los datos númericos de estos términos junto con sus cálculos aparecen en el anexo A.

Por último faltan las acciones del sistema. La suma o resta de velocidades de los motores hace que el quadcopter se mueva en una dirección o en otra.

Estas acciones son U_1, U_2, U_3 y U_4 que permiten controlar los movimientos en cada eje y vienen relacionadas con las velocidades de los motores de la siguiente manera:

U_1 corresponde a la potencia del quadrotor en global, sumando los 4 motores. Siendo T_i la fuerza vertical que ejerce cada motor:

$$U_1 = T_1 + T_2 + T_3 + T_4 \tag{3.7}$$

U_2 es la utilizada para el movimiento en el eje Y:

$$U_2 = -T_1 + T_2 + T_3 - T_4 \tag{3.8}$$

U_3 se utiliza para el movimiento en el eje X:

$$U_3 = -T_1 - T_2 + T_3 + T_4 \tag{3.9}$$

U_4 se utiliza para el giro sobre el eje Z siendo Q el momento de arrastre de cada hélice:

$$U_4 = -Q_1 + Q_2 - Q_3 + Q_4 \tag{3.10}$$

Donde:

$$T_i = C_t \rho (A\omega_i)^2 \tag{3.11}$$

$$Q_i = C_q \rho A^2 \omega_i^3 \tag{3.12}$$

Siendo C_t y C_q constantes aerodinámicas dependientes de la estructura de la hélice, ρ la densidad del aire, A el área de barrido de la hélice y ω_x la velocidad ángular de cada motor. El cálculo de C_t y C_q se puede ver en las ecuaciones A.12 y A.13 de la sección A.

3.2. Diseño del control de posición

El control de posición se utiliza para la realización de trayectorias definidas por coordenadas (X Y Z) a alcanzar junto con el ángulo Yaw para dirigir la "cabeza" en la dirección

que se quiera. Por lo tanto serán éstas las variables a controlar.

Se realiza un control por espacio estados ya que al estar todas las variables relacionadas en las ecuaciones se realiza un control múltiple del conjunto de las variables.

Se elige un control por prealimentación de la consigna con integrador. La realimentación del estado permite estabilizar el sistema y ajustar el régimen transitorio del control (tiempo de respuesta, sobreoscilación, acciones máximas), y la prealimentación de consigna y el integrador eliminan el error de posición de la variable correspondiente en régimen permanente. [8]

El esquema de control queda de la siguiente manera:

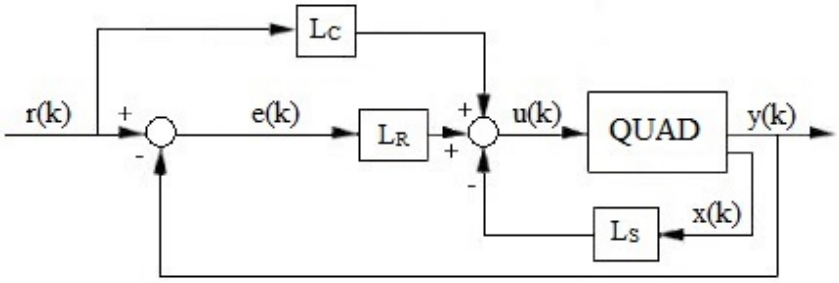


Figura 3.2: Esquema de control de prealimentación con integrador.

La señal r(k) es la referencia, que la componen: X, Y, Z y ψ, e(k) el error y u(k) la acción. Y(k) es la salida realimentada que la forman: X, Y, Z y ψ, y x(k) el estado siendo éste todas las variables de estado: X, $\dot{X}$, Y, $\dot{Y}$, Z, $\dot{Z}$, θ, $\dot{\theta}$, ϕ, $\dot{\phi}$, ψ y $\dot{\psi}$. K es el instante en el que se encuentra el bucle de control.

Por lo tanto la acción queda de la siguiente manera:

$$u = -L_S \cdot x + L_C \cdot r + L_R \cdot e \tag{3.13}$$

Donde L_C, L_R y L_S se han calculado utilizando la técnica de asignación de polos [8] a partir de las especificaciones dinámicas del control como tr=1 s. El periodo de muestreo elegido es T=0.1 ms.

$$L_c = \begin{pmatrix} -0{,}0259 & 71{,}1233 & 0 & 0 \\ 0 & 0 & 0 & -4{,}4593 \\ 0 & 0 & 4{,}5023 & 0 \\ 0{,}9220 & -0{,}0025 & 0 & 0 \end{pmatrix} \tag{3.14}$$

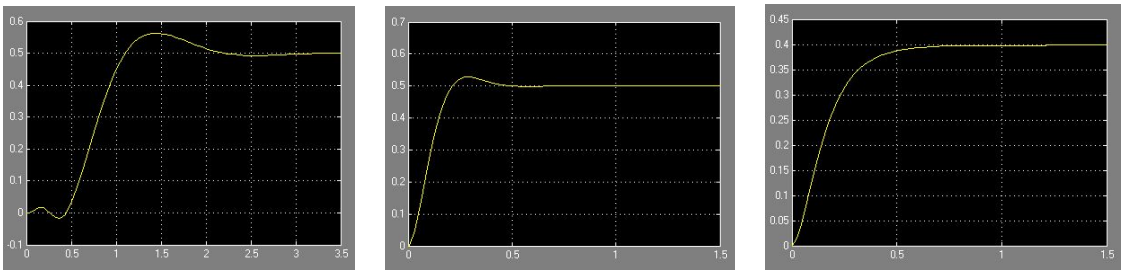
$$L_r = \begin{pmatrix} 16{,}3053 \\ 0 \\ 0 \\ -0{,}0009 \end{pmatrix} \tag{3.15}$$

$$L_{s1} = \begin{pmatrix} 0 & 0 & 0 & 0 & -0{,}0259 & -0{,}0016 \\ 5{,}5731 & 0{,}4939 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5{,}6268 & 0{,}4987 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}922 & 0{,}1572 \end{pmatrix} \tag{3.16}$$

$$L_{s2} = \begin{pmatrix} 71{,}1233 & 8{,}0094 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4{,}4593 & -1{,}9856 \\ 0 & 0 & 4{,}5023 & 2{,}0057 & 0 & 0 \\ -0{,}00250 & -0{,}0001 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.17}$$

$$L_s = \begin{pmatrix} L_{s1} & L_{s2} \end{pmatrix} \tag{3.18}$$

Como se puede ver en la figura 3.3 se alcanzan las referencias marcadas para el dron:



(a) Escalón en X e Y de 0.5 m. (b) Escalón en Z de 0.5 m. (c) Escalón en Yaw de 0.4 m.

Figura 3.3: Control de posición.

3.3. Diseño del control de velocidad

El control de velocidad permite generar movimientos sin especificar destino y utilizar el autopiloto desde un mando de radiocontrol del quadcopter. Las variables que se controlan en este apartado son las velocidades lineales en los ejes X e Y y la velocidad angular en el eje Z.

Se han estudiado varias posibilidades para realizar este control:

3.3.1. Control PD sobre velocidades

El siguiente esquema de control utiliza un modelo modificado del usado en la tesis de Peter Corke [9]. En ella realiza un control de posición con la diferencia respecto a los demás controles de que relaciona posición y velocidad lineal con ángulos y velocidades angulares. Se modifica para hacer un control de velocidad pero sigue teniendo esa característica principal.

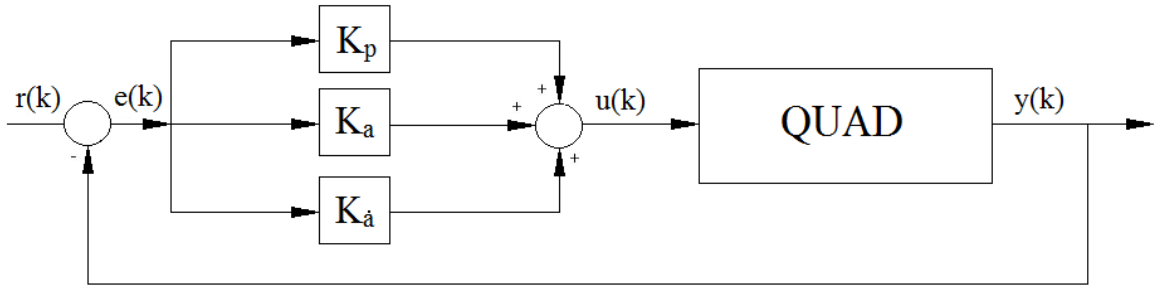


Figura 3.4: Esquema de control sobre velocidades.

Siendo las referencias: $\dot{X}$, $\dot{Y}$ y $\dot{\psi}$ y las salidas o estado realimentado: $\dot{X}$, $\dot{Y}$, θ, $\dot{\theta}$, ϕ, $\dot{\phi}$ y $\dot{\psi}$. K_a es la constante referente al ángulo y $K_{\dot{a}}$ a la velocidad ángular.

Las acciones son las siguientes:

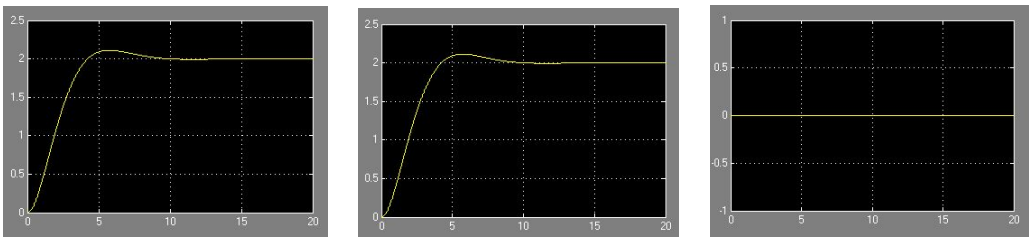
$$U1 = \text{Potencia directamente impuesta desde el mando.} \tag{3.19}$$

$$U2 = -((\dot{Y}_{ref} - \dot{Y}) \cdot k_y + \theta \cdot k_\theta + \dot{\theta} \cdot k_{\dot{\theta}}) \tag{3.20}$$

$$U3 = ((\dot{X}_{ref} - \dot{X}) \cdot k_x - \phi \cdot k_\phi - \dot{\phi} \cdot k_{\dot{\phi}}) \tag{3.21}$$

$$U4 = ((\dot{\psi}_{ref} - \dot{\psi}) \cdot k_{\dot{\psi}}) \tag{3.22}$$

Se consiguen muy buenos resultados en simulación con este esquema pudiéndose controlar perfectamente las velocidades lineales en X e Y y la velocidad angular en el eje Z.



(a) Velocidad en X con referencia 3. (b) Velocidad en Y con referencia 3. (c) Yaw con referencia 0.

Figura 3.5: Simulaciones del control PD de velocidad.

3.3.2. Control PD sobre ángulos

Como segunda opción se realiza un control sin usar las velocidades, controlando directamente los ángulos. No es un control de velocidad exactamente pero es equivalente ya que si se le impone un ángulo al quadcopter, éste irá a una velocidad determinada constante. El utilizar ángulos ante velocidades se debe a la dificultad que conlleva calcular la velocidad lineal a partir de las aceleración proporcionada por el IMU. Es un control proporcional-derivativo que sigue el esquema de la figura 3.6 y las siguientes ecuaciones:

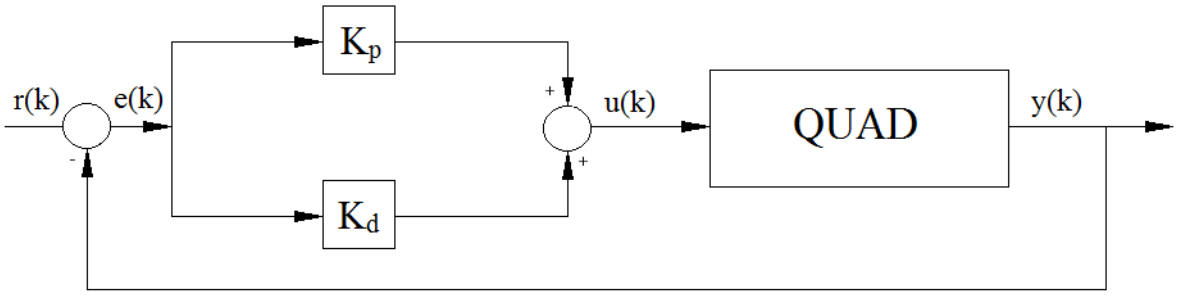


Figura 3.6: Esquema de control PD.

Siendo las referencias: θ, ϕ y $\dot{\psi}$ y las salidas realimentadas: θ, $\dot{\theta}$ ϕ, $\dot{\phi}$ y $\dot{\psi}$.

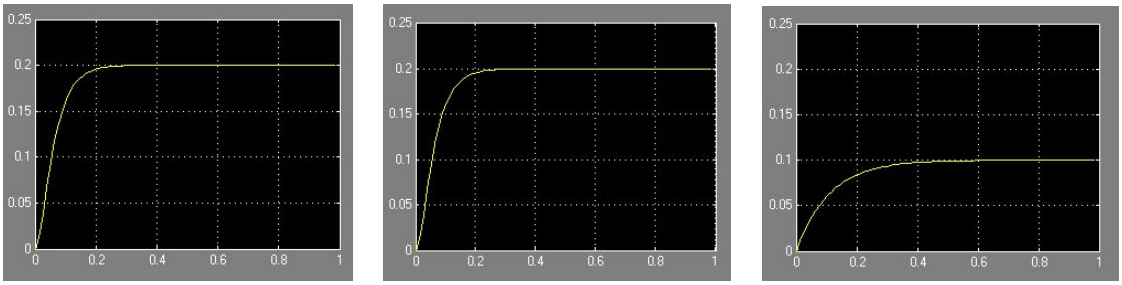
$$U1 = \text{Potencia directamente impuesta desde el mando.} \tag{3.23}$$

$$U2 = -((\theta_{Ref} - \theta) \cdot k_{p\theta} - \dot{\theta} \cdot k_{d\theta}) \tag{3.24}$$

$$U3 = (\phi_{Ref} - \phi) \cdot k_{p\phi} - \dot{\phi} \cdot k_{d\phi} \tag{3.25}$$

$$U4 = (\dot{\psi}_{Ref} - \dot{\psi}) \cdot k_{d\psi} \tag{3.26}$$

Se consigue controlar los ángulos Roll, Pitch y velocidad angular de Yaw. Ante una referencia de 0.2 *rad* para los ángulos y de 0.1 rad/s para $\dot{\psi}$, las salidas son las siguientes:



(a) Ángulo Roll. (b) Ángulo Pitch. (c) Velocidad angular Yaw.

Figura 3.7: Simulaciones del control PD sobre ángulos.

Capítulo 4

Implementación

Se ha decidido implementar un sistema de tiempo real frente a uno secuencial clásico debido al determinismo temporal necesario para este tipo de controles.

Un sistema de tiempo real es el cual la exactitud de éste no depende sólo del resultado lógico computacional si no también del momento en el que los resultados son generados [10].

Se identifican las actividades a realizar junto con sus características:

- Gestión del IMU. Manda una trama cada 13 ms y se lee a través de la comunicación serie SCI.
- Control del sistema. Se realiza con un periodo de muestreo de 13 ms.
- Gestión del ultrasonidos. Se necesita un dato cada 25 ms.
- Envío de comandos al dron. Se interrumpirá y se recogerá el dato.
- Obtener las referencias impuestas por el mando de radiocontrol. Llega una nueva referencia cada 13 ms.

4.1. Diseño del software

La estructura del software está basada en tareas. Es la herramienta que proporciona SYS/BIOS para los sistemas de tiempo real. Cada una de estas tareas tiene una primera parte de inicialización de las funcionalidades que se van a usar, un bucle infinito donde se encuentra el código en si de la tarea y una primitiva bloqueante, la cual en este caso son los semáforos. Se asignan tareas a cada actividad con requisitos temporales distintos o funcionalidades muy diferentes.

Se distinguen varios tipos de tareas dependiendo de cuando se activan:

- **Tareas periódicas:** Éstas se ejecutan cada un cierto tiempo definido constante.
- **Tareas esporádicas**: Son activadas mediante una interrupción del sistema. Puede ser por un estímulo externo, y por tanto son interrupciones hardware, o interno, lo que corresponde a una interrupción software.

Las tareas necesitan permiso para ser ejecutadas y éste se lo proporcionan los semáforos. Cuando una tarea está lista debe esperar a la señal de un semáforo que tiene asociado que le dé permiso para ejecutarse. La manera más utilizada para activar las tareas es mediante un reloj cuyo funcionamiento es el siguiente:

Se dispone del reloj interno del sistema (1), siendo éste la referencia para cualquier otro reloj implementado. A partir de éste se crea un reloj con la frecuencia que se desee para cada una de las tareas que se quieran implementar. Cada vez que hay un tick, se produce una interrupción software 3) activando el semáforo 4) y si la tarea también está lista 5), ésta pasará a ejecución 6).

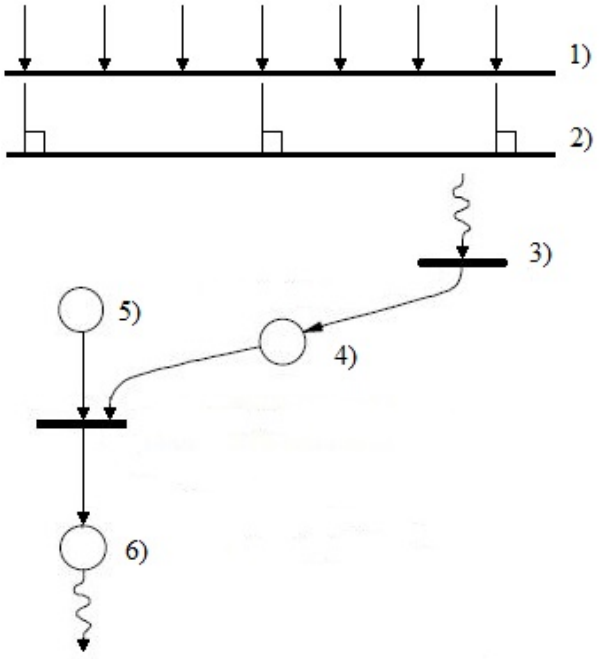


Figura 4.1: Diagrama de accionamiento de tareas. [6]

Otra de las maneras utilizadas para activar el semáforo es a través de interrupciones hardware.

Estas tareas, además de tener unos tiempos que deben cumplirse y un semáforo que les dé permiso para empezar a ejecutarse, tienen asignadas unas prioridades. Estas prioridades determinan cual de ellas debe ejecutarse en primer lugar, sin importar que otra tarea esté

ejecutándose. Por ello puede darse el caso de que esté ejecutandose una tarea, se le dé permiso a otra de mayor prioridad y por lo tanto ésta le quite el procesador, ejecutándose y después devolviéndoselo una vez que ha acabado. Por ello se debe elegir bien el orden de prioridad de las tareas.

En la siguiente figura se puede ver el funcionamiento interno de las tareas: cuando un sémaforo le da permiso a una *nueva* tarea ésta pasa a *preparado*. Si el procesador está listo se *ejecuta*, lo que no quiere decir que vaya a acabar toda su ejecución ya que en cualquier momento llega otra de mayor prioridad y esta pasará a *espera*. Cuando el procesador vuelva a estar libre, se ejecutará de nuevo hasta que acabe y entonces pasará a *terminado*.

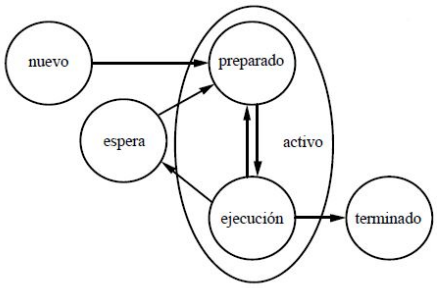


Figura 4.2: Diagrama de funcionamiento de las tareas. [6]

4.1.1. Tareas

Se han creado 5 tareas:

- **Enviar comunicaciones al ordenador:** Es una tarea periódica la cual envía datos de telemetría al ordenador. Se crea un reloj que lanza el semáforo cada segundo. No son datos relevantes en el control y por eso es la tarea de menor prioridad.

  En esta tarea simplemente se leen los datos de los servidores y se envían mediante el Xbee.

- **Recepción de datos del IMU:** Tarea principal del programa donde se reciben los datos del IMU cada 13 ms. Éste interrumpe el sistema cada vez que manda un caracter y cuando se han recogido todos, se activa el semáforo. A continuación se ejecuta el control y se calculan las acciones necesarias para controlar el dron. Se guardan los datos del sensor en los servidores para ser utilizados en las demás tareas.

- **Envío de pulso al ultrasonidos:** Se produce una interrupción software cada 25 *ms* mediante otro reloj que activa el semáforo. En esta tarea se envía el pulso que necesita el ultrasonidos para devolvernos la medida.

- **Recepción de pulso del ultrasonidos:** Cuando el ultrasonidos nos devuelve el pulso en el cual está implícita la medida salta una interrupción que activa el semáforo.

  En esta tarea se ejecuta una función que recoje la medida y que lo guarda en el servidor de los sensores. La altura no es necesaria para el control de velocidad implementado pero se utiliza para dar robustez al sistema detectando en que momento se está en tierra y cuando en el aire.

- **Conversión de los pulsos del receptor RF a referencias** Mediante interrupción recoge los pulsos de todos los canales, los convierte en referencias válidas para el control y los guarda en los servidores correspondientes.

4.1.2. Servidores

Para la comunicación entre tareas con los servidores, se utiliza un *mutex* con herencia de prioridad para proteger el acceso a las variables compartidas (*GateMutexPri*).

Se han creado 3 servidores:

- **Sistema:** En este servidor se guarda el estado del sistema. Se han definido distintas estructuras de variables como son: el propio del quadcopter, de los motores, del switch... En el caso del estado propio del quadcopter tenemos 4 posibilidades: *tierra*, *aire*, *despegando* y *aterrizando*. El estado de los motores puede ser *ON* u *OFF* y el estado del switch, *manual* o *automático*.

  Esta información es necesaria para saber que funciones realizar en cada tarea. Uno de los ejemplos más claros puede ser que si no estamos en tierra, no se pueda cambiar el modo de funcionamiento (de control de posición a control de velocidad) ya que podría acarrear problemas en el vuelo que se está realizando.

- **Sensores:** En este segundo servidor se guardan todos los datos que nos proporcionan los sensores. Este servidor está dividido en dos, uno para el IMU y otro para el ultrasonidos.

- **Radiofrecuencia:** Por último está el servidor de radiofrecuencia donde la tarea que captura los datos del mando escribe en él y la tarea del IMU donde se encuentra el control los lee para convertirlos en referencias.

4.2. Análisis de tiempo real

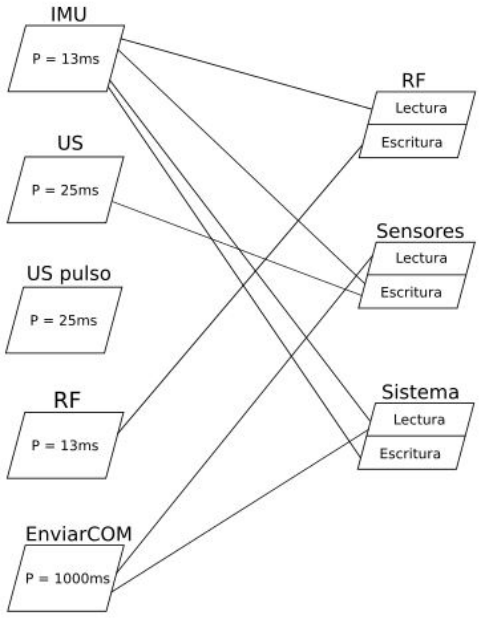


Figura 4.3: Diagrama de tareas y servidores.

Para garantizar que los resultados sean generados en el momento adecuado es necesario calcular el tiempo que tarda cada tarea en ejecutarse y cuanto tiempo tarda en leer y escribir en los servidores. Además deberemos saber si se cumplen sus plazos de respuesta sin excepción. Por ello se realiza una planificación de las tareas:

Tarea	Prioridad	C	T=D	B_{HP}
IMU	5	914 μs	13 ms	7,68 μs
RF	4	59,8 μs	13 ms	7,25 μs
US Pulso	3	2,346 μs	25 ms	7,25 μs
US	2	12 μs	25 ms	3,6 μs
COM	1	3,99 μs	1000 ms	-

Cuadro 4.1: Tabla de planificación de las tareas.

Donde C es el tiempo de computo, T el periodo, D el plazo de respuesta y B_{HP} es el bloqueo por herencia de prioridad.

$$W(T_i) = \sum_{j=1}^{i-1} \lceil \frac{D_i}{P_j} \rceil \cdot C_j + B_i < D_i \tag{4.1}$$

$$W(IMU) = 0{,}921\text{ms} < 13\text{ms}. \tag{4.2}$$

$$W(RF) = 0,981\text{ms} < 13\text{ms}. \tag{4.3}$$

$$W(USPulso) = 1,957\text{ms} < 25\text{ms}. \tag{4.4}$$

$$W(US) = 1,965\text{ms} < 25\text{ms}. \tag{4.5}$$

$$W(COM) = 75,56\text{ms} < 1000\text{ms}. \tag{4.6}$$

Como se puede observar, los plazos de respuesta de este sistema de tiempo real se cumplen muy holgadamente, gracias a este DSP tan potente. Además se calcula la utilización del procesador, donde se puede comprobar que no se utiliza ni un 10 %:

$$U = \sum_{j=1}^{n} \frac{C_j}{P_j} = 7,5\,\% \tag{4.7}$$

4.3. Drivers software implementados

A continuación se explican los drivers que se han diseñado e implementado en el F2812 para integrar el distinto hardware utilizado:

4.3.1. Receptor de radiofrecuencia

El mando de radiofrecuencia emite un tren de pulsos donde cada uno de estos corresponde a un canal del mando. El receptor divide el tren de pulsos en 5 canales diferentes.

Para ello se utiliza el *Event Manager* del F2812 que nos permite a través de la funcionalidad llamada *Input Capture* recoger este valor del pulso. Se utiliza uno para cada canal.

El proceso es el siguiente: el *Input Capture* se programa para que dé una interrupción cada vez que haya un flanco en la señal de entrada, que en este caso es un pulso. La primera interrupción se tiene con un flanco positivo y se guarda el valor del *timer* en ese momento. Se espera a que llegue la siguiente interrupción, que es cuando ha habido un flanco negativo, y se vuelve a guardar ese valor del *timer*. Se hace la resta y los cálculos

correspondientes y obtenemos el tiempo de duración del pulso. Los pulsos tienen una duración entre 1 y 2 ms. Esto se realiza con los 5 canales.

4.3.2. Ultrasonidos

El ultrasonidos necesita un pulso de 2 μs como mínimo por su patilla central para devolver la medida por esa misma patilla. Por lo tanto lo que se debe hacer es establecer ese pin como salida, darle el pulso e inmediatamente cambiar ese pin a entrada y utilizar de nuevo un *Input Capture*. Así de la manera explicada en el párrafo anterior se puede medir el tiempo del pulso. La distancia sigue la siguiente fórmula:

$$\text{Distancia en cm} = (\text{tiempo en ticks}) \cdot 853 \cdot \frac{34{,}32}{2} \tag{4.8}$$

Donde el 853 corresponde del paso de ticks a ms, el 34.32 es la velocidad del sonido en el aire y el 2 hace referencia a que hay recorrido de ida y vuelta del sonido y sólo se quiere medir uno.

4.3.3. IMU

El IMU originalmente te ofrece unas tramas predefinidas que son los ángulos calculados por el mismo o los datos de los sensores. Se necesita una mezcla de ambas. El software que el IMU tiene integrado te permite enviarle un comando e indicarle que trama quieres recibir. Una de las opciones es recibir una de las tramas, mandar el comando y esperar a la siguiente trama para completar los datos y repetir este proceso una y otra vez. Esto hace el sistema el doble de lento y por lo tanto se desestima ya que el control requiere un periodo de muestreo muy pequeño.

La otra opción es reprogramar el IMU para obtener la trama que deseamos en el menor tiempo posible. Se reprograma a través de la plataforma de Arduino consiguiendo la trama deseada y además se aumenta la velocidad de transmisión consiguiendo los datos cada 13 ms.

```
YPR=-32.51,0.98,-12.37,49.00,144.00,-16.00
```

Figura 4.4: Trama del IMU.

Esta trama es enviada mediante SCI. Se dispone de unas funciones para la comunicación SCI de [6].

4.3.4. Xbee

La comunicación inalámbrica del quadrotor con el ordenador se realiza mediante el Xbee. Con ésta podemos mandar datos de telemetría del dron al ordenador para analizarlos como también se le pueden dar órdenes al quadcopter desde el ordenador.

Donde más se han utilizado estas funciones son en los entornos de pruebas, después se podrá utilizar para mandar cierta información sobre el estado del quadcopter como nivel de batería o alguna otra información que pueda ser relevante.

El Xbee integrado en el quad va conectado a la segunda SCI que dispone este DSP. Su modo de funcionamiento es igual al utilizado por el IMU usando también las funciones que estaban ya implementadas.

4.3.5. ESCs + Motores

Para el movimiento de los motores se necesita enviar un PWM de unos valores y frecuencia determinados. El rango de velocidad de los motores es: con 1 ms de PWM se tiene velocidad nula y con 2 ms se tiene su velocidad máxima que es de 1618 rad/s. La frecuencia del PWM puede variar entre 50 y 450 Hz. La diferencia de una frecuencia u otra depende de la velocidad de refresco que se quiera de este PWM ya que el control le va a cambiar su valor cada 13 ms. Si hay un PWM cada 15 ms, puede darse el caso de que no se haya actualizado este valor hasta 28 ms después. Por ello se ha puesto una frecuencia de 400 Hz para que su refresco sea prácticamente inmediato.

Para realizar el PWM también se utilizan los *Event Manager*, pero en este caso se utiliza su función de *Output Compare*. A través de un *timer* y dos registros se puede generar una onda PWM de la siguiente manera:

El *timer* comienza a contar y se pone a 1 el valor del PWM. Cuando llega al valor que ha sido puesto en el registro de comparación la onda se pone a 0 obteniendo el estado positivo del PWM. Por último el estado negativo acabará cuando se llegue al registro de periodo donde comienza de nuevo este proceso.

Por lo tanto con el registro de comparación del *timer* se establece el valor del PWM de 1 a 2 ms y con el registro de periodo la frecuencia que éste tiene.

Capítulo 5

Pruebas en el quadcopter

Probar el quadcopter real no es tan trivial como parece. La potencia que éste tiene y el peligro de sus hélices conlleva realizar las pruebas de una manera muy controlada. Además mientras no se tiene el control ajustado, éste puede hacer movimientos realmente bruscos que pueden producir alguna avería o llegar a hacer daño a alguien.

Por lo tanto sólo una vez que en las simulaciones funciona todo perfectamente se procede a probar los controles en el quadcopter.

Como se ha explicado, sólo se ha implementado el control de velocidad. Ante la dificultad de estimar la velocidad con el sensor IMU, se utiliza el control sobre los ángulos Roll y Pitch y la velocidad ángular de Yaw. Éste control aparece en la sección 3.3.2.

Dado que el modelo es aproximado, las constantes utilizadas en simulación no sirven, pero si que dan una idea de como es una en proporción a la otra. Si se prueba directamente se puede ver que no responde como debería siendo imposible su control. Por ello se deberá comenzar una serie de pasos previos hasta conseguir volar totalmente suelto.

Se limitan todos sus movimientos excepto un giro. Ésto es, impidiéndole su desplazamiento en los ejes X e Y, estando atado al techo y al suelo y con un pequeño recorrido en el eje Z.

Además se le impide el giro en el ángulo Yaw y en uno de los otros dos ángulos, para poder ajustar primero solamente un ángulo. Se comienza intentando ajustar el ángulo Pitch.

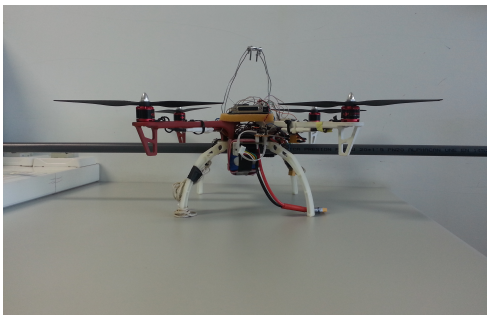


Figura 5.1: Sistema de limitación de ángulos.

La manera de ajustar las constantes K_p y K_d es empírica. Hay varios métodos de ajuste de controladores PD utilizandose el mismo que en simulación, el método de oscilación. Se incrementa la constante proporcional hasta ver que comienza a oscilar. Es en ese momento cuando se incrementa la K_d hasta que éste deja de oscilar.

El primer paso es que con una referencia de 0 *rads* éste se estabilice sobre el eje permitido. Una vez que es bástante estable y responde ante perturbaciones (pequeños golpes que intentan desestabilizarlo) se procede a intentar ajustar el ángulo Roll.

Se repite el proceso cambiando el ángulo que se deja libre. Cuando esté también controlado, se ajusta la velocidad angular en eje el Yaw.

El ángulo Yaw se controla en velocidad. Éste es el más sencillo de controlar.

Una vez que se tienen todos los ángulos controlados, se debe cambiar el escenario de pruebas.

La primera opción era seguir con la estructura de la primera prueba pero permitiéndole los tres giros. Al estar suspendido sobre una cuerda, el sistema se vuelve inestable aunque se prueben los dos controles independientes ajustados juntos.

Por ello se coloca el quadcopter en el suelo, atado a 3 puntos en el suelo permitiéndole despegar hasta 1 metro de altura. Como medida de seguridad se tendrá una caña que sólo lo sujetará si se ve que pierde el control, si no la caña no ejercerá ninguna fuerza sobre el quadcopter.

Con este nuevo sistema se consigue despegar y estar estable en el aire siendo un grandísimo avance para esta primera experiencia con drones.

El siguiente paso sería volarlo totalmente suelto. Se deja como continuación para futuros TFGs seguir con la mejora de estos controles, consiguiendo vuelos totalmente libres, como también el otro subobjetivo principal, que es el control de posición del quadcopter para navegación autónoma.

Capítulo 6

Conclusiones

El objetivo principal de este TFG ha sido el diseño e implementación de un autopiloto junto con la estabilización de vuelo.

Se plantearon inicialmente dos subobjetivos muy ambiciosos: el control de posición del quadcopter para navegación autónoma y el control en velocidad para recrear el uso clásico del quadcopter telemanipulado.

Se ha diseñado el control de posición y validado en simulaciones pero no se ha llegado a implementar debido a las limitaciones de nuestro hardware.

Se necesita realimentar variables de estado que no se disponen. Los sensores que se han integrado nos proporcionan ángulos, velocidades angulares y aceleraciones. Para este tipo de control se necesitan las velocidades lineales y las posiciones. Se han integrado las aceleraciones para obtener las velocidades pero debido al ruido del sensor y también a la dinámica tan inestable del quadcopter ha sido imposible estimar con suficiente precisión las velocidades y por lo tanto las posiciones.

Se plantea como futuro trabajo la integración de otros sensores capaces de obtener velocidades lineales y posiciones como también, utilizando el control diseñado, implementarlo sobre el sistema de tiempo real creado.

Respecto al control en velocidad y estabilización del vuelo, se han conseguido realizar vuelos cortos. Éste es un primer paso muy importante para conseguir un autopiloto completo y ha sido como primera experiencia muy gratificante debido a que se ha conseguido poner en vuelo el dron a pesar de los innumerables problemas que han surgido por la dinámica tan compleja del quadcopter y al desarrollar un sistema tan completo como es éste ya que contiene hardware, software, tiempo real e ingeniería de control.

Apéndice A

Magnitudes físicas del quadcopter.

Las magnitudes físicas del quadcopter se han medido o calculado a partir de otras. En este anexo se recogen las más significativas junto con sus cálculos.

En la sección 3.1 se hace referencia a los términos I_{xx}, I_{yy} y I_{zz} que son la diagonal principal del tensor de inercia. Siendo éste:

$$J = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \tag{A.1}$$

En la figura siguiente aparece representado el modelo simplificado de inercias seguido para los cálculos:

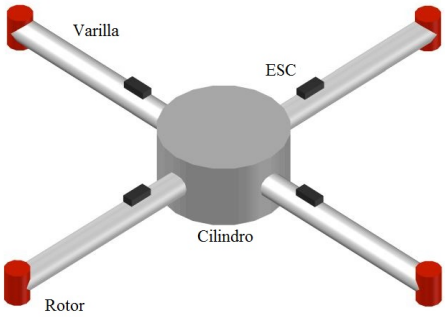


Figura A.1: Modelo de inercias del dron.

$$I_{ii} = \int_M d_i^2 dm \tag{A.2}$$

Y por tanto su cálculo es el siguiente:

$$I_{xx1} = (0{,}5 \cdot m_{varilla} \cdot r_{varilla}^2) + (\frac{2}{12} \cdot m_{rotor} \cdot l_{rotor}^2) + (\frac{1}{12} \cdot m_{varilla} \cdot l_{varilla}^2) + (0{,}5 \cdot m_{cilindro} \cdot r_{cilindro}^2) \tag{A.3}$$

$$I_{xx2} = 2 \cdot (\frac{1}{12} \cdot m_{rotor} \cdot l_{rotor}^2 + m_{rotor} \cdot (d_{rotor})^2) + 2 \cdot (\frac{1}{12} \cdot m_{esc} \cdot (a_{esc}^2 + b_{esc}^2) \tag{A.4}$$

$$I_{xx3} = m_{esc} \cdot (l_{varilla}/4)^2) + 2 \cdot (\frac{1}{12} \cdot m_{esc} \cdot (l_{esc}^2 + b_{esc}^2) + m_{esc} \cdot (\frac{l_{varilla}}{4})^2) \tag{A.5}$$

Siendo:

$$I_{xx} = I_{xx1} + I_{xx2} + I_{xx3} \tag{A.6}$$

Como en el modelo para el cálculo de inercias es un quadcopter simétrico:

$$I_{yy} = I_{xx} \tag{A.7}$$

Ahora I_{zz}:

$$I_{zz1} = (0{,}5 \cdot m_{cilindro} \cdot r_{cilindro}^2) + (\frac{2}{12} \cdot m_{varilla} \cdot l_{varilla}^2) + 4 \cdot (0{,}5 \cdot m_{rotor} \cdot r_{rotor}^2 \tag{A.8}$$

$$I_{zz2} = (d_{rotor}^2) \cdot m_{rotor}) + 4\frac{1}{12} \cdot m_{esc} \cdot (l_{esc}^2 + a_{esc}^2) + m_{esc} \cdot (\frac{l_{varilla}}{4})^2 \tag{A.9}$$

Donde:

$$I_{zz} = I_{zz1} + I_{zz2} \tag{A.10}$$

Otra de las variables que aparecen directamente en el modelo de la sección 3.1 es J_r:

$$J_r = \text{número de aspas} \cdot m_{\text{hélice}} \cdot \frac{r_{\text{hélice}}^2}{4} \tag{A.11}$$

En la ecuación 3.11 aparecen los términos C_t y C_q que dependen de la aerodinámica de la hélice. El primero de ellos se calculó experimentalmente de la siguiente manera:

Cuando el quadcopter está en equilibrio en el aire se cumple esta ecuación:

$$\mathrm{mg} = 4 \cdot C_t \cdot \rho \cdot \mathrm{A} \cdot r^2_{\text{hélice}} \cdot (\omega)^2 \tag{A.12}$$

La velocidad ángular se puede estimar sabiendo el PWM que es mandado al quadcopter, por lo tanto la única variable que queda es C_t.

La otra constante aerodinámica que tenemos es C_Q que se calcula en función de C_t:

$$C_Q = C_t \sqrt{\frac{C_t}{2}} \tag{A.13}$$

Después de dichos cálculos se recogen todas las magnitudes del quadcopter en la siguiente tabla:

Magnitud	Valor	Unidad
$m_{varilla}$	0.0705	kg
$m_{cilindro}$	0.885	kg
m_{rotor}	0.056	kg
m_{esc}	0.03	kg
m_{total}	1.5	kg
$r_{varilla}$	0.015	m
$l_{varilla}$	0.25	m
r_{rotor}	0.015	m
l_{rotor}	0.025	m
d_{rotor}	0.14	m
$r_{cilindro}$	0.075	m
$l_{cilindro}$	0.1	m
l_{esc}	0.045	m
a_{esc}	0.025	m
b_{esc}	0.007	m
I_{xx}	0.0056	kgm^2
I_{yy}	0.0056	kgm^2
I_{zz}	0.0081	kgm^2
número de aspas	2	unidades
$m_{\text{hélice}}$	0.015	kg
$r_{\text{hélice}}$	0.013	m
J_r	$1,267 \cdot 10^{-4}$	kgm^2
g	9.81	m/s^2
$A_{\text{barridohélice}}$	0.053	m^2
ρ	1.184	kg/m^3
C_t	$8,326 \cdot 10^{-6}$	adimensional
C_Q	$6,775 \cdot 10^{-8}$	adimensional

Cuadro A.1: Colección de magnitudes físicas.

Bibliografía

[1] Ardupilot. *www.ardupilot.com.*

[2] Controles del mando de radiofrecuencia.

[3] Estructura quad. *www.electronicarc.com.*

[4] Sergeant rupert frere rlc, crown copyright/mod 2013.

[5] Uav o drone, la historia de un arma diseñada para la guerra. *http://www.erepublik.com/es/article/uav-o-drone-la-historia-de-un-arma-diseada-para-la-guerra-2516757/1/20*, 2015.

[6] Alan Burns y Andy Wellings. *Sistemas de tiempo real y lenguajes de programación 3a edición.* 2003.

[7] Beatriz Frisón. Modelado y control de un helicóptero de cuatro motores. *Universidad de Zaragoza, CPS*, 2008/2009.

[8] Luis Moreno, Santiago Garrido, y Carlos Balaguer. *Modelado y control de sistemas dinámicos.* 2003.

[9] Paul Pounds, Robert Mahony, y Peter Corke. Modeling and control of a quad-rotor robot. *Australian National University, Canberra, Australia.*

[10] J. Stankovic. Misconceptions of real-time computing. 1988.

[11] Referencia web del producto adquirido en DJI. Https://www.dji.com/product/flame-wheel-arf/feature.