AUTORIZACIÓN

David Hernández Plaza, Javier García Guerrero y Gerardo Saiz Yanguas, alumnos matriculados en la asignatura de Sistemas Informáticos, autorizan, mediante el presente documento, a la *Universidad Complutense de Madrid (UCM)* a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y el prototipo desarrollado, todo ello realizado durante el curso académico 2009-2010 bajo la dirección de *Matilde Santos Peñas*, profesora del *Departamento de Arquitectura de Computadores y Automática* de la *Facultad de Informática* de dicho organismo.

Plataforma de simulación de maniobras de un cuatrirotor

Autores:

- Javier García Guerrero.
- Gerardo Saiz Yanguas.
- David Hernández Plaza.

Director:

- Matilde Santos Peñas.

Curso: 2009-2010

Proyecto de Sistemas Informáticos

Tabla de contenido

Plataforma de simulación de maniobras de un cuatrirotor.

Resumen

Los vehículos aéreos tripulados por control remoto ó no tripulados (UAV por sus siglas en inglés) son aeronaves autónomas que pueden volar sin necesidad de contar con un piloto humano ya que cuentan con un control de vuelo programado. Su uso está cada vez más extendido en tareas con grandes riesgos ó que presentan cierta dificultad, como en labores de lucha contra incendios, seguridad civil, misiones militares, etc. Dentro de los diferentes helicópteros, en nuestro proyecto vamos a tratar el cuatrirotor que está compuesto de cuatro rotores que dotan a la aeronave de mayor estabilidad, y cuyos movimientos se consiguen variando la velocidad relativa de cada rotor.

En la asignatura de Sistemas Informáticos durante el curso 2009/2010 nos hemos propuesto a realizar una plataforma de simulación de manera que podamos controlar el movimiento programado de un cuatrirotor y simularlo tanto en dos como en tres dimensiones. Para ello se abordarán diferentes problemas, principalmente:

- Establecer la posición en tres dimensiones del cuatrirotor y los diferentes ángulos de orientación proporcionados por el modelo dinámico del vehículo.
- Permitir modificar las propiedades del cuatrirotor por el usuario.
- Crear un sistema de control que regule el comportamiento de cada uno de los rotores con el fin de alcanzar una posición deseada y que sea modificable por el usuario.
- Simular la trayectoria del cuatrirotor, obtenida con un controlador y visualizarlo en dos y tres dimensiones.

El documento está organizado de la siguiente manera: En el capítulo 1 se detallará la introducción y los objetivos de nuestro proyecto. En el capítulo 2 se explica la descripción del sistema. En el capítulo 3 se detalla el diseño de la aplicación y durante el capítulo 4, mostramos las conclusiones. Finalmente, en el capítulo 5 se presentan los trabajos futuros del proyecto.

Summary

The unmanned aerial Vehicle (UAV) are autonomous aircrafts that can fly without human pilot because they have a scheduled flight control. Its use is increasingly widespread in tasks with high risk or with some difficulties, such in fire fighting, civil security, military missions, etc. Among the different helicopters, in our project we will deal with quadrotor which consists of four rotors. It gives the aircraft more stablility and its movements are achieved by varying the speed of each rotor.

In the Computer Systems subjet during the academic year 2009/2010 we try to implement a simulation platform so that we can control the motion of a cuatrirotor and simulate it in both two and three dimensions. This will address several main problems:

- Set the position in three dimensions of quadrotor and different orientation angles provided by the vehicle dynamic model.
- Allow quadrotor modify the properties of the user.
- Simulating the trajectory of quadrotor, obtained with a controller and viewing it in two and three dimensions.
- Create a control system that regulates the behavior of each of the rotors in order to achieve a desired position and is modified by the user.

The document is structured as follows: Chapter 1 will contain the introduction and objectives of our project. Chapter 2 explains the description of the system. Chapter 3 details the implementation and design, during Chapter 4, we show the conclusions. Finally, Chapter 5 presents the future works of the project.

1. Introducción. Motivación y objetivos.

En el proyecto se ha diseñado e implementado una plataforma de simulación para un helicóptero cuatrirotor guiado por un controlador, pudiendo visualizar la trayectoria mediante gráficas en dos dimensiones o a través de un motor gráfico que nos permite representar un escenario virtual.

En primer lugar, el sistema permitirá modificar las propiedades del cuatrirotor tales como su masa o la distancia de los rotores al centro del vehículo. Además, la aplicación no sólo dispondrá de un controlador, sino que permitirá al usuario configurar su propio controlador dando la posibilidad de simular y probar la validez de su comportamiento.

Dado un controlador, el sistema recibirá una posición deseada y éste nos proporcionará la potencia necesaria de cada rotor para alcanzar el estado objetivo. De acuerdo con esta potencia, y por medio de la dinámica que describe el modelo del helicóptero, conoceremos la trayectoria del vehículo, la cual podremos simular.

Objetivos:

- Implementación de la dinámica del cuatrirotor.
- Estudio de la dinámica de vuelo en Matlab.
- Estudio e implementación de métodos de integración en diferencias finitas.
- Implementación de la dinámica y de las limitaciones físicas del cuatrirotor en .NET.
- Implementación de controladores.
- Utilización de las pruebas realizadas en la fase anterior para diseñar un controlador que guíe de manera automática el helicóptero y lo mantenga estable en un punto.

- Desarrollo de una pequeña aplicación que permita modificar los parámetros del controlador y almacenarlo para su posterior simulación.

Implementación de un simulador en tres dimensiones.

- Creación de un modelo tridimensional de un cuatrirotor con una herramienta de modelado.
- Acoplamiento de analizador de mapas de alturas para el desarrollo de diferentes escenarios que sirvan como mapa y siente las bases del simulador.
- Estudio y representación de los movimientos del cuatrirotor con respecto a su modelo dinámico.

Integración del controlador y la dinámica al simulador.

- Diseño e implementación de un controlador para determinar las entradas y salidas del mismo.
- Simulación del cuatrirotor con el controlador y análisis de la respuesta del sistema.

2. Descripción del sistema

En el primer apartado se explicará la planta de cuatrirotor donde se describen los elementos necesarios que hemos utilizado para implementar la dinámica, cinemática y movimientos rotacionales del cuatrirotor. Además se detallarán sus características principales, la estructura con sus movimientos y las ecuaciones físicas que simulan su comportamiento. En el segundo apartado explicaremos los controladores PID y la técnica de ganancia variable que aportará mejoras en el control de nuestro vehículo. Finalmente en el tercer apartado se comentará el integrador y detallaremos la técnica usada para calcular el siguiente estado del vehículo en cada paso.

2.1. Planta del cuatrirotor

El cuatrirotor es un vehículo volador que dispone de 4 hélices que están unidas por 2 ejes que se cruzan, sobre cuyo vértice se sitúa una unidad central. Como se puede ver en la figura 1 el cuatrirotor tiene una disposición simétrica de sus 4 rotores, es decir, uno situado en la zona posterior, otro en la zona anterior, y otros dos a izquierda y derecha, eligiendo correctamente el sistema de referencia. Uno de los objetivos del proyecto consiste en dirigir ese centro de masas de un punto a otro variando sólo la potencia de cada uno de los cuatro rotores.

Figura 1: Prototipo de un cuatrirotor actual

2.1.1. Estructura del cuatrirotor. Movimientos

En el cuatrirotor se aplican fuerzas en la dirección y sentido que se muestran en la figura 2.

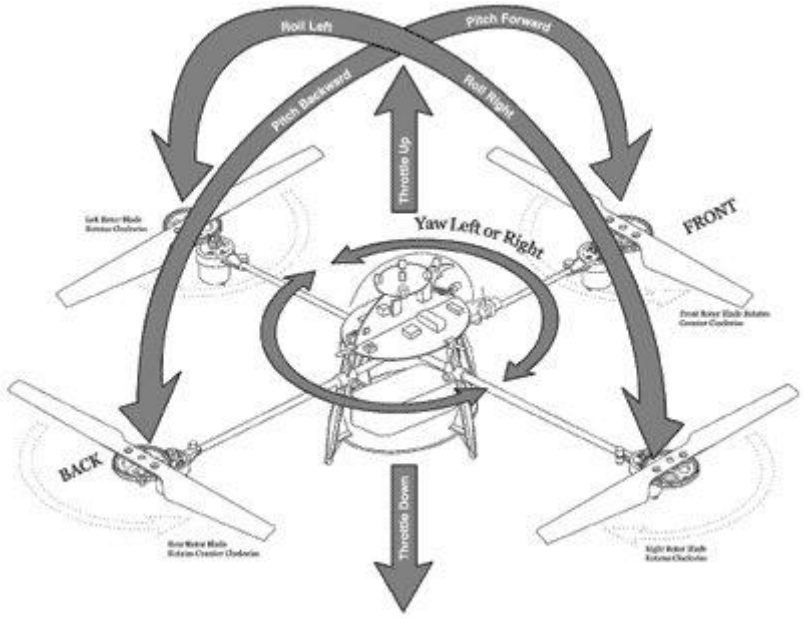

Figura 2: Fuerzas y movimientos en el cuatrirotor

El movimiento de un cuerpo rígido posee seis grados de libertad que se definen de la siguiente forma:

- La posición de un punto de referencia del cuerpo (centro de masas) en un sistema de referencia (x, y, z) requiere 3 grados de libertad.
- Orientación del cuerpo en el sistema de referencia (Φ, θ, Ψ) son los 3 grados de libertad restantes.

Cada grado de libertad toma dos variables de estado (posición y velocidad); por lo tanto necesitamos 12 ecuaciones diferenciales para definir la dinámica del cuatrirotor:

- x = (x, y, z) es el vector de posición del cuatrirotor.
- v = (u, v, w) es el vector de velocidad del cuatrirotor.
- α = (Φ, θ, Ψ) son los ángulos de Euler del cuatrirotor (cabeceo, alabeo y viraje respectivamente)
- ω = (p, q, r) son las tasas de cambio de los ángulos.

2.1.2. Cinemática del cuatrirotor

En un cuatrirotor tenemos un motor delantero y otro trasero que rotan en sentido horario, mientras que los rotores laterales rotan en sentido contrario a las manecillas del reloj. Con ello, los efectos giroscópicos y los momentos aerodinámicos tienden a cancelarse en vuelos estacionarios.

La entrada de control principal ó fuerza principal es la suma de las fuerzas producidas por cada motor tal y como podemos ver en la figura 3.

$$\mu = \sum_{i=1}^{4} fi$$

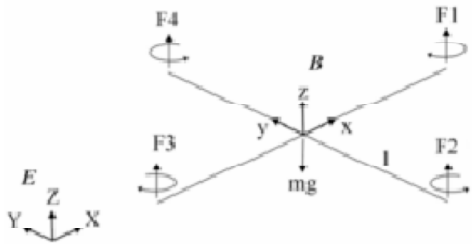

Figura 3: Fuerzas de cada rotor

Dado el vector **x** = (x, y, z) que está definido respecto al sistema de referencia inercial, y el vector **v** = (u, v, w) está respecto al sistema de referencia cuerpo, para

pasar de un sistema de referencia a otro hay que utilizar la matriz de Euler para giros sobre tres ejes. Suponiendo que los giros se producen en sentido horario, la matriz de rotación tendrá la forma conocida:

$$R_{i/v}(\phi,\Theta,\psi)=\begin{pmatrix}\cos\Theta\cos\psi & sen\phi sen\Theta\cos\psi-\cos\Theta sen\psi & \cos\phi sen\Theta\cos\psi-sen\Theta sen\psi\\ \cos\Theta sen\psi & sen\phi sen\Theta sen\psi+\cos\Theta\cos\psi & \cos\phi sen\Theta sen\psi-sen\phi\cos\psi\\ -sen\Theta & sen\phi\cos\Theta & \cos\phi\cos\Theta\end{pmatrix}$$

Se cumplen las siguientes ecuaciones:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{pmatrix}=R_{i/v}(\phi,\Theta,\psi)\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

Ya hemos obtenido las 3 ecuaciones.

Para relacionar **w** = (p, q, r) con d/dt (Φ, θ, Ψ) consideramos las rotaciones del sistema de referencia cuerpo (Srv):

$$\begin{pmatrix}p\\ q\\ r\end{pmatrix}=\begin{pmatrix}\phi\\ 0\\ 0\end{pmatrix}+\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\phi & sen\phi\\ 0 & -sen\phi & \cos\phi\end{pmatrix}\begin{pmatrix}0\\ \dot{\Theta}\\ 0\end{pmatrix}+\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\phi & sen\phi\\ 0 & -sen\phi & \cos\phi\end{pmatrix}\begin{pmatrix}\cos\Theta & 0 & -sen\Theta\\ 0 & 1 & 0\\ sen\Theta & 0 & \cos\Theta\end{pmatrix}\begin{pmatrix}0\\ 0\\ \dot{\psi}\end{pmatrix}$$

Operando con las matrices se obtiene:

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & 0 & -sen\Theta \\ 0 & \cos\phi & sen\phi\cos\Theta \\ 0 & -sen\phi & \cos\phi\cos\Theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\Theta} \\ \dot{\psi} \end{pmatrix}$$

Invirtiendo la matriz anterior obtenemos:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\Theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & sen\phi tg\phi & \cos\phi tg\Theta \\ 0 & \cos\phi & -sen\phi \\ 0 & sen\phi\sec\Theta & \cos\phi\sec\Theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

Teniendo como resultado tres ecuaciones más.

2.1.3. Dinámica del cuatrirotor

Las aceleraciones en las coordenadas x, y, z son términos importantes a controlar. Hay que saber que la aceleración se puede extraer de la segunda ley de Newton, en la que se relacionan las fuerzas que actúan sobre el cuerpo con la masa total y la aceleración del cuatrirotor respecto del sistema de referencia. Puesto que la masa es un dato conocido, bastará con hallar las fuerzas que actúan para saber las aceleraciones.

Si analizamos las fuerzas a las que estará sometido el cuatrirotor, éstas serán la sustentación producida por cada motor, la ejercida por la gravedad, y los correspondientes rozamientos.

$$F_T = F_m - P - F_r$$

La fuerza del motor que tiene que expresar el movimiento en vertical viene dado por la siguiente expresión:

$$F_T = F_m - P - F_r$$

Desde el punto de vista inercial la fuerza queda de la siguiente forma:

$$F^i = R_{i/v} F^v = \begin{pmatrix} (\cos\phi sen\Theta\cos\psi + sen\phi sen\psi)\sum_{i=1}^{4} F_i \\ (\cos\phi sen\Theta sen\psi - sen\phi\cos\psi)\sum_{i=1}^{4} F_i \\ \cos\phi\cos\Theta\sum_{i=1}^{4} F_i \end{pmatrix}$$

La fuerza del peso P viene dado por la expresión $P = m \cdot g$.

Donde la gravedad se ha considerado constante e igual a 9.8m / s^2

Desde el punto de vista inercial la fuerza queda de la siguiente forma:

La fuerza de rozamiento ρ que sufre el vehículo, en cada eje de coordenadas, tiene la misma expresión:

$$F_r = \tfrac{1}{2} C_i \cdot A_c \cdot \rho \cdot i \cdot |i|$$

Donde: **i** = x, y, z. El coeficiente C_i depende de varios factores (Stevens, 2003), pero para nuestro caso nos bastará considerarlo constante e igual a 1.3 en todos los casos.

Igualmente la densidad del aire ρ, la consideramos constante e igual a 1.293 Kg / m^3.

A_c es el área que presenta en su movimiento la unidad central (supuestamente esférica), por lo que:

$$A_c = \pi \cdot R^2 = 0.0113 \quad m^2$$

Siendo R el radio de la unidad central.

Como vemos, los términos correspondientes a estos rozamientos son muy pequeños, por lo que, aunque se han incorporado en la dinámica del cuatrirotor, podrían ser despreciados sin grandes efectos en el comportamiento del modelo simulado.

Finalmente juntando todas las fuerzas presentes obtenemos:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} \dfrac{(\cos\phi sen\Theta\cos\psi + sen\phi sen\psi)\sum_{i=1}^{4} F_i - \frac{1}{2} C_x A_c \rho u |u|}{M_T} \\ \dfrac{(\cos\phi sen\Theta sen\psi + sen\phi\cos\psi)\sum_{i=1}^{4} F_i - \frac{1}{2} C_y A_c \rho v |v|}{M_T} \\ \dfrac{\cos\phi\cos\Theta\sum_{i=1}^{4} F_i}{M_T} - g - \dfrac{\frac{1}{2} C_z A_c \rho w |w|}{M_T} \end{pmatrix}$$

2.1.4. Movimientos rotacionales

El movimiento general de un cuerpo rígido en el espacio se puede representar por la combinación de los movimientos de rotación y traslación.

Para el movimiento rotacional del cuatrirotor tendremos que:

$$dM / dt_i = T$$

Donde M es el momento angular y $T = (\tau_\varphi, \tau_\theta, \tau_\psi)$ el torque aplicado.

Por una parte, $M = J \cdot \omega$ donde:

$$J = \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{yx} & J_y & -J_{yz} \\ -J_{zx} & -J_{zy} & J_z \end{pmatrix}$$

Donde J es la matriz de inercia, que si consideramos al cuatrirotor como un elemento simétrico respecto de sus ejes, tomara la forma diagonal:

$$J = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}$$

Considerando al cuatrirotor como una esfera central de masa M y radio R de la que se proyectan cuatro varillas sin masa, de longitud l, en cuyos extremos se encuentra una masa puntual m, que representa al conjunto del motor y hélice, tendremos que:

$$J_i = \frac{2}{5}\cdot M\cdot R^2+2\cdot l^2\cdot m \quad \text{para i = x, y}$$

y

$$J_z = \frac{2}{5}\cdot M\cdot R^2+4\cdot l^2\cdot m$$

Por otra parte, los torques aplicados (τ_φ, τ_θ, τ_ψ) tomaran distintas formas, al depender cada uno de una configuración distinta de los motores.

La rotación en torno al eje X (cabeceo) se deberá a una diferencia entre las fuerzas F_2 y F_4.

Concretamente tendremos que:

$$\tau_\varphi = l\cdot(F_2-F_4)$$

La rotación en torno al eje Y (cabeceo) se deberá a una diferencia entre las fuerzas F1 y F3.

Concretamente tendremos que:

$$\tau_\theta = l\cdot(F_1-F_3)$$

La rotación del cuatrirotor entorno al eje Z (viraje) se deberá a una diferencia en el torque aplicado por los rotores delantera y trasero, y los laterales:

$$\tau_\psi = \tau_1+\tau_2-\tau_3-\tau_4$$

Y tomando la ecuación de Coriolis:

$$dM/dt_i = dM/dt_v + \omega \times M = T$$

De esta manera llegamos a las conocidas ecuaciones de Euler que describen como las componentes del vector velocidad angular evolucionan en el tiempo, en respuesta a los torques aplicados:

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = J^{-1} \cdot \left\{ \begin{pmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} - \begin{pmatrix} i & j & k \\ p & q & r \\ J_x p & J_y q & J_z r \end{pmatrix} \right\} = \begin{pmatrix} \frac{\tau_\varphi}{J_x} \\ \frac{\tau_\theta}{J_x} \\ \frac{\tau_\psi}{J_x} \end{pmatrix} + \begin{pmatrix} \frac{J_x - J_z}{J_x} \cdot q \cdot r \\ \frac{J_z - J_x}{J_y} \cdot p \cdot r \\ \frac{J_x - J_y}{J_z} \cdot p \cdot q \end{pmatrix}$$

2.2. Modelo del cuatrirotor

El modelo del que partimos (Sánchez, 2009) que muestra la figura 4, está implementado en Matlab©, software explicado en el anexo de herramientas utilizadas, y es en el que nos hemos basado para implementar nuestro modelo dinámico para poder implementarlo en .Net y desarrollar con él nuestro simulador.

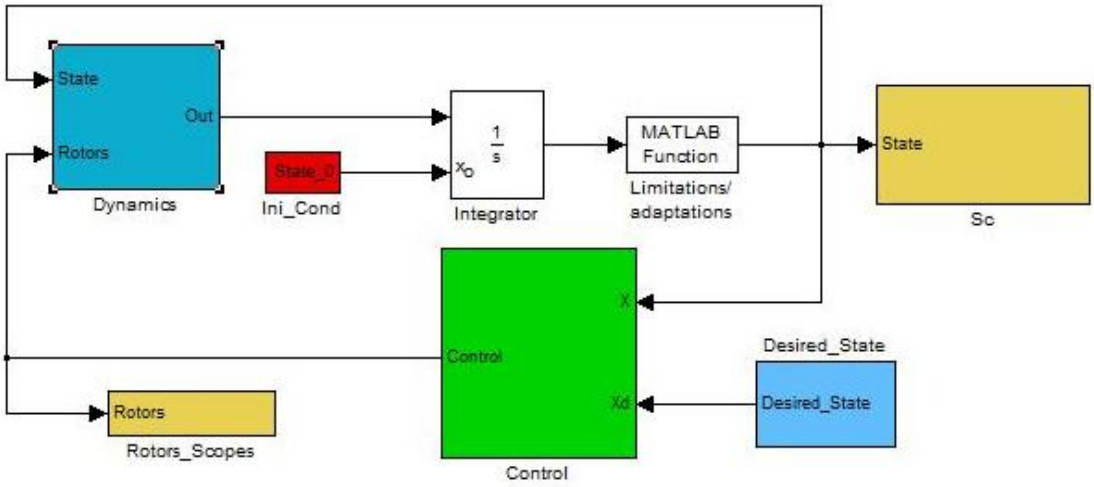

Figura 4: Modelo dinámica de cuatrirotor en Matlab.

Tal y como se muestra en la figura 4 podemos encontrar los siguientes bloques:

- Dynamics
- Ini_Cond
- Integrator
- Limitations/Adaptations
- Control
- Desired_State

2.2.1. Dynamics

Entrada: State, Rotors

Salida: Out

Ficheros: quad_dynamic.m

Este bloque es la planta del sistema, que contiene las ecuaciones físicas que describen su comportamiento. Recibe como entrada el estado actual del cuatrirotor y la potencia necesaria que se debe ejercer a cada rotor para llegar al estado que se desea. Con estas entradas se calcula el estado siguiente que tomará en el próximo periodo de tiempo. Obsérvese que el siguiente estado, podría coincidir o no con el estado deseado. Con el tiempo el sistema se estabiliza y el estado siguiente coincidirá con el estado deseado, cuando el control actúe correctamente.

Estas ecuaciones físicas están guardadas en el archivo de Matlab quad_dynamics.m, el cual se apoya en el fichero auxiliar limitations.m que contiene datos constantes relativos al cuatrirotor como características del cuatrirotor y suposiciones físicas.

2.2.2. Ini_Cond

Salida: Xd

En este bloque se fijan las condiciones iniciales del cuatrirotor, es decir, define el estado inicial. Es de gran utilidad para hacer diferentes pruebas.

2.2.3. Integrator

Entrada: Ini_Cond, Out_dynamics

Salida: St

El integrador es una función del Simulink de Matlab que sirve para mejorar la sintonía y eliminar ruidos. La primera vez recibirá como entrada Ini_Cond con el estado inicial del cuatrirotor, luego cuando el sistema se realimente en sucesivas iteraciones recibirá como entrada Out_dynamics. Devuelve el estado integrado de lo que le llegue a la entrada.

2.2.4. Limitations/Adaptations

Entrada: Out_integrator

Salida: State

Ficheros: physical_limitations.m

A este módulo le llega el estado actual y se encarga de restringir los valores de estados que no podemos permitir.

Entre las limitaciones están:

- No permitir alturas negativas. Lo que significaría que no se permitiría que el cuatrirotor estuviera por debajo del nivel del suelo. En caso de que a este módulo le llegara un valor negativo lo pondría a cero y todos los ángulos y velocidades se pondrían a cero también.
- Limitaciones en los ángulos y velocidades del cuatrirotor son necesarias para garantizar la estabilidad del vehículo. Esto es útil cuando el controlador produce señales de control muy bruscas ante errores muy grandes. Los límites marcarían de esta forma el límite de seguridad. Es el caso del cabeceo y del alabeo que sólo se permitirán giros en el rango de (-1.3, 1.3) radianes, mientras que para el viraje solo permitimos Ⅱ/12 radianes.

2.2.5. Control

Entrada: X, Xd

Salida: Rotors

Ficheros: selección.m, PID.m

Es el módulo del controlador del sistema. Contiene el PID para estabilizar el sistema. Recibe como entradas el estado actual y el estado deseado. Selecciona de ahí

las variables necesarias para el controlador PID a través del fichero selección.m y con esa información el controlador es capaz de calcular las señales de control necesarias para conducir al sistema, que serán enviadas al bloque dynamics. El controlador PID está a su vez compuesto por varios controladores PID que controlan cada uno de ellos una variable del cuatrirotor.

2.2.6. Desired State

Salida: Desired_State

Este bloque es el equivalente al Ini_Cond, su función es fijar los valores de estado deseados. Se usa junto al Ini_cond para probar el sistema.

2.3. Controlador PID

Para nuestro sistema de control hemos implementado un controlador PID, ya que teníamos más conocimientos de este tipo de controladores que del resto, por lo que se nos hacia mas factible hacer un desarrollo eficaz y limpio de este.

Un controlador PID (Proporcional, integral y derivativo) es un mecanismo de control por realimentación que se utiliza en sistemas de control industriales. Un controlador PID corrige el error entre un valor medido y el valor que se quiere obtener calculándolo y obteniendo una acción correctora que puede ajustar al proceso acorde. El algoritmo de cálculo del control PID depende de tres parámetros de sintonía: la ganancia proporcional, integral, y derivativa. La acción proporcional determina la reacción ante el error actual. La integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. La derivativa determina la reacción del tiempo en el que el error se produce. La suma de estas tres acciones es usada para ajustar al proceso a un elemento de control como la posición de una válvula de control o la energía suministrada a un calentador, por ejemplo. Ajustando estas tres constantes en el

algoritmo de control del PID, el controlador puede proveer una acción diseñada para lo que requiera el proceso a realizar. La respuesta del controlador puede ser descrita en términos de respuesta del control ante un error, el grado el cual el controlador llega al "set point", y el grado de oscilación del sistema. Nótese que el uso del PID para control no garantiza control óptimo del sistema o la estabilidad del mismo. Algunas aplicaciones pueden solo requerir de uno o dos modos de los que provee este sistema de control. Un controlador PID puede ser llamado también PI, PD, P o I en la ausencia de las acciones de control respectivas. Los controladores PI son particularmente comunes, ya que la acción derivativa es muy sensible al ruido; la ausencia del proceso integral puede conllevar que no se alcance al valor deseado en el estacionario.

2.3.1. Ganancia variable o adaptativa

Un sistema de control adaptativo es un sistema que en forma continua y automática mide las características dinámicas de la planta, las compara con las características deseadas, y usa las diferencias para variar los parámetros ajustables del controlador, es decir, genera la señal de control de modo que se mantenga un funcionamiento óptimo, independiente de las modificaciones del entorno.

Esta técnica cambiará los parámetros de control dependiendo del rango de valores en el que se encuentra el error, es decir, actúa como un control secundario influenciando a los parámetros del control primario.

Esta técnica aplicada al control de un cuatrirotor ayudará a que el vehículo disponga de velocidad y potencia a la hora de alcanzar un estado objetivo lejano o cuyo error sea grande, y además aportará estabilidad y precisión cuando las distancias sean pequeñas.

2.3.2 Integrador: Método Runge-Kutta para Ecuaciones Diferenciales

Uno de los métodos más utilizados para resolver numéricamente problemas de ecuaciones diferenciales ordinarias con condiciones iniciales es el método de Runge-Kutta de cuarto orden, el cual proporciona un pequeño margen de error con respecto a la solución real del problema y es fácilmente programable en un software para realizar las iteraciones necesarias.

Es sumamente útil para casos en los que la solución no puede hallarse por los métodos convencionales (como separación de variables). Hay variaciones en el método de Runge-Kutta de cuarto orden pero el más utilizado es el método en el cual se elige un tamaño de paso h y un número máximo de iteraciones n tal que:

$$y_0 = y(t_0)$$

$$k_1 = h \cdot f(t_i, y_i)$$

$$k_2 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$$

$$k_3 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = h \cdot f(t_i + h, y_i + k_3)$$

Y se realiza la iteración:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Para $i = 0,\ldots, n\text{-}1$. La solución se da a lo largo del intervalo $(t_o,\ t_o + hn)$.

El algoritmo para el método de Runge-Kutta de cuarto orden en pseudo código es el siguiente:

INICIO

INPUT: Número de iteraciones n (o tamaño de paso h), punto inicial del intervalo a (punto final del intervalo b), condición inicial $y(t_0) = y_0$.

$n = (b - a) / h;$

$t = t_0;$

$y = y_0;$

$OUTPUT \quad (t, y)$

PARA i = 1,..., n;

$k_1 = h \cdot f(t, y);$

$k_2 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_1}{2}\right);$

$k_3 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_2}{2}\right);$

$k_4 = h \cdot f(t + h, y + k_3);$

$y = y + (k_1 + 2k_2 + 2k_3 + k_4);$

$t = t + i \cdot h;$

$OUTPUT \quad (t, y)$

FIN PARA

FIN

El cual hemos utilizado para resolver las ecuaciones diferenciales del modelo, todo ello implementado con tecnología .NET.

3. Diseño del simulador

Tras una explicación de los aspectos básicos más importantes, en este capítulo presentaremos detalladamente nuestra aplicación, describiendo el funcionamiento de los distintos módulos que componen nuestro proyecto así como la relación que existe entre ellos y que mostramos en la siguiente figura.

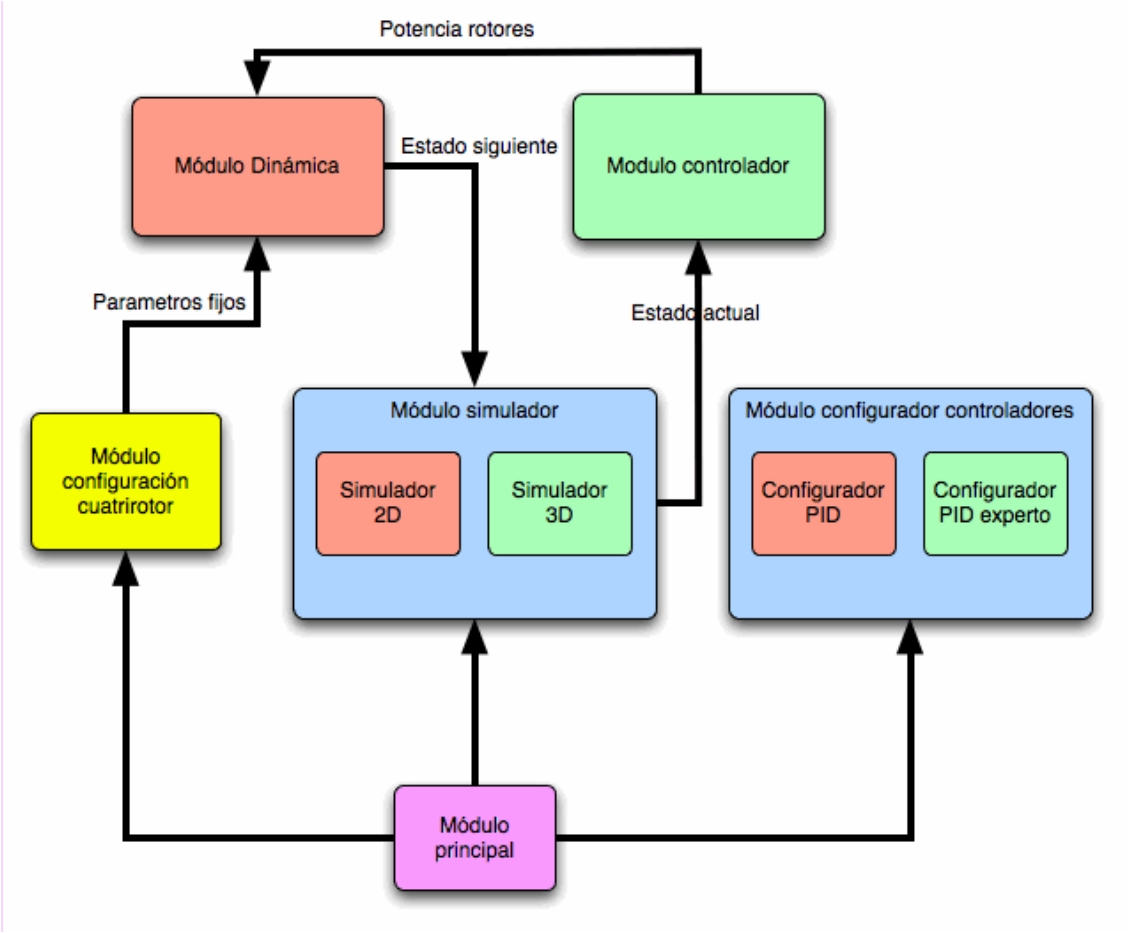

Figura 5: Esquema del sistema implementado

La aplicación está dividida en cuatro grandes módulos para dar funcionalidad a los tres objetivos importantes que componen nuestra aplicación: realizar modificaciones en las características del cuatrirotor, configurar y ajustar los controladores y, por último, simularlo tanto en dos como en tres dimensiones.

La aplicación da comienzo con un pequeño módulo principal o selector, que por elección del usuario, nos da opción a comenzar la simulación en dos ó tres dimensiones, la modificación de algunos parámetros del vehículo o la configuración de los controladores. El cierre de este módulo supondrá la finalización de la aplicación, cerrando así todas las ventanas que estén abiertas en ese momento.

El módulo de configuración del cuatrirotor nos autorizará a cambiar diferentes parámetros del vehículo tales como la masa de la unidad central y del conjunto motor-engranaje-hélice, la distancia de cada uno de los rotores al centro del cuatrirotor, la longitud de las varillas de las hélices, el radio de la unidad central y el límite del voltaje de la pila que alimentará a los rotores. Por defecto, los parámetros son los establecidos por el modelo de David Sánchez Benítez para una maqueta del cuatrirotor de la que se dispone en el Departamento de ACYA, en el grupo de Ingeniería de Sistemas y Automática.

3.1 Módulo dinámica

Es el módulo encargado de la parte física del cuatrirotor, describiendo la evolución en el tiempo en relación a las causas que provocan los cambios de estado físico y de movimiento. Este módulo describe los factores capaces de producir alteraciones del sistema físico, planteado por las doce ecuaciones descritas en la sección 2.1.

Podemos decir que del módulo dinámica obtendremos la posición siguiente del cuatrirotor, dados una posición inicial y la potencia de cada uno de los cuatro rotores.

Partiendo del modelo implementado por Sánchez, 2009 en Matlab, hemos realizado una traducción al lenguaje de implementación del simulador, obteniendo el siguiente esquema de implementación.

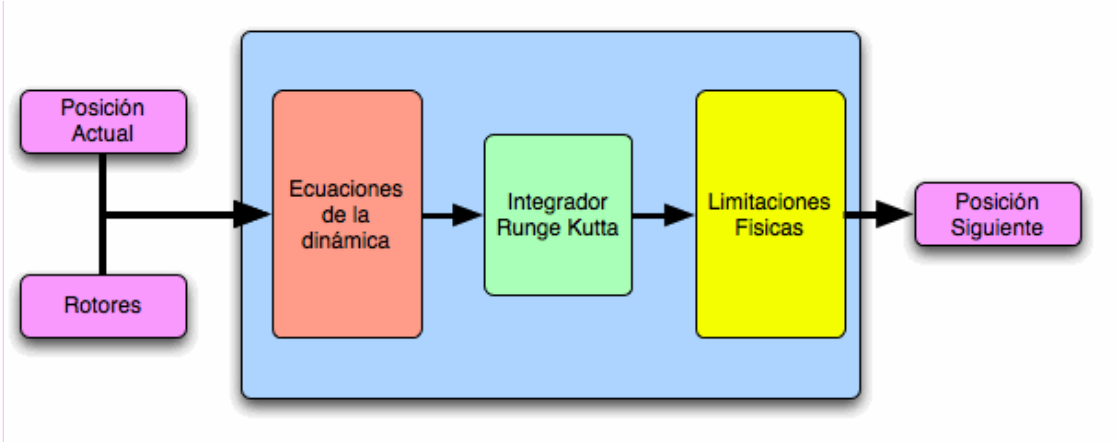

Figura 6: Esquema de la dinámica del sistema implementada.

Tal y como muestra la figura 6, está compuesto por otros tres módulos importantes. El primero contiene las doce ecuaciones diferenciales que describen el movimiento del vehículo, que se ha explicado con detenimiento en el capítulo 2. Este es el apartado que partiendo del código de Matlab, hemos traducido a nuestro lenguaje de implementación.

El segundo módulo es el que contiene el integrador. En el modelo de Matlab, es un componente basado en el método numérico de diferencias finitas llamado Runge – Kutta de grado cuatro. Debido a ello hemos tenido que realizar su estudio y posteriormente implementarlo. La función de este método numérico, es la integración de las ecuaciones de la dinámica, cuyo resultado son los siguientes valores del estado del cuatrirotor.

La tercera parte del modelo dinámico alberga las limitaciones físicas. Aquí recogemos aquellos estados que un cuatrirotor no puede alcanzar. Su función es restringir valores de estado que no podemos permitir.

Entre estas limitaciones están:

- No se permiten alturas negativas para que el vehículo no esté por debajo del nivel del suelo. Si le llegasen desde el integrador una altura negativa, este módulo pondrá a cero tanto los ángulos como las velocidades.

– Hay limitaciones en los ángulos y en la velocidad del cuatrirotor necesarias para garantizar la estabilidad del vehículo. Esto es necesario ya que el controlador, frente a errores muy grandes, puede producir señales de control muy fuertes. Éste módulo marcaría entonces el límite de seguridad.

En la aplicación este módulo interactúa tanto con el módulo simulador como con el módulo encargado de configurar diferentes controladores (ver figura 5). El simulador, en la tarea de mostrar la visualización de una trayectoria del helicóptero, solicitará repetidamente la nueva posición que adquirirá el vehículo de acuerdo a su posición actual y la fuerza de los rotores suministrado por el controlador elegido.

Los parámetros físicos de la dinámica podrán ser variados desde el módulo configurador del cuatrirotor cuya interfaz mostramos en la figura 7. La modificación de dichos parámetros cambiará el movimiento descrito por el cuatrirotor, aunque la aplicación no asegura el buen funcionamiento del vehículo para cualquier configuración, por lo que dichos cambios se harán bajo el conocimiento y la responsabilidad del usuario**.**

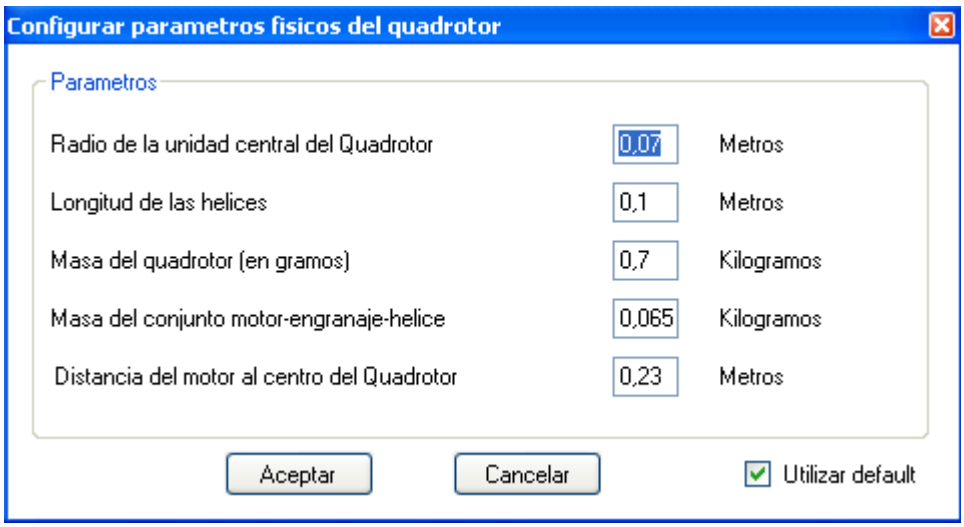

Figura 7: Interfaz del módulo de configuración de la dinámica.

3.2. Módulo de Control

El objetivo de este módulo es reemplazar la figura del piloto por un control inteligente de manera que determine la potencia que se debe aplicar a cada rotor del vehículo para que este llegue a su posición objetivo. Para que este módulo sea completamente funcional debe recibir la posición actual de cuatrirotor y su posición objetivo.

Este módulo podría implementarse de muchas maneras, siempre y cuando calcule de manera más o menos precisa la potencia que se debe aplicar a cada uno de los cuatro rotores del vehículo. Nosotros hemos utilizado dos estrategias de control para realizar la simulación y que explicamos a continuación:

3.2.1. Controlador PID:

Se compone de controladores PID's diferenciados en dos niveles. La causa de esta estructuración se debe a que por una parte tenemos un nivel que controla la posición que alcanzará el vehículo y por otra, tenemos un nivel que asegura a que la estabilidad del cuatrirotor se mantenga.

Los que dirigen la posición del vehículo son los controladores X, Y y de la altura Z, mientras que los que afianzan el equilibrio son los controladores cabeceo, alabeo y viraje, que mostramos en la figura 8.

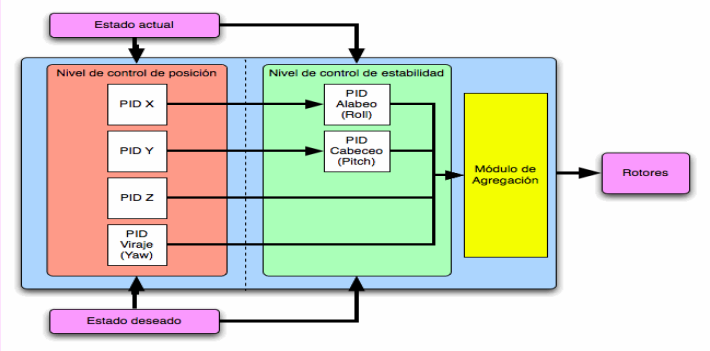

Figura 8: Implementación del controlador PID

Para la sintonización de todos los controladores PID hemos utilizado la "sintonización cualitativa", que se trata en ir modificando manualmente los parámetros Kp, Ki y Kd hasta encontrar una respuesta estable y que mantenga la salida esperada. El proceso que hemos seguido en cualquier sintonización ha sido siempre el mismo. Como primer paso, ponemos los parámetros Ki y Kd a 0, y aumentamos Kp hasta que el sistema oscila. A continuación disminuimos Kp hasta la mitad aproximadamente y aumentamos Ki hasta que el proceso se ajusta en un tiempo requerido. Por último incrementamos Kd para que el sistema sea rápido.

Estas sintonizaciones han sido realizadas gracias al módulo de configuración de controladores PID que mostramos en la figura 9. Esta herramienta nos permite simular diferentes configuraciones de PID, pudiendo variar los parámetros observando la evolución de los resultados en gráficas. Una vez sintonizados los parámetros el configurador nos permite guardarlos para ir generando una biblioteca de PID's.

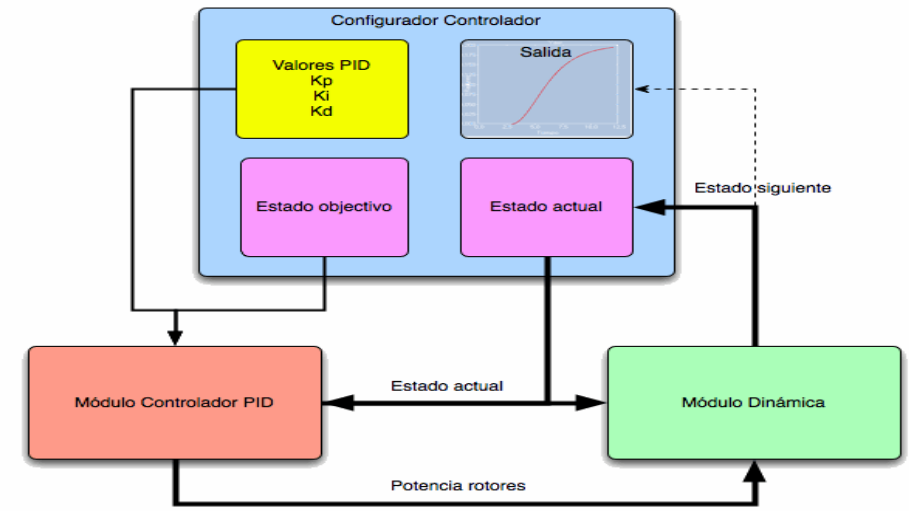

Figura 9: Implementación del configurador de controlador PID

Tal y como podemos ver en la figura, el configurador, tras solicitar al usuario el estado objetivo, los valores Kp, Ki y Kd y el tiempo de simulación, comunica al PID estos datos creando un bucle entre este, la dinámica y nuestro configurador. El proceso comienza en el PID al recibir el estado objetivo, el estado actual (inicialmente es 0) y los valores Kp, Ki y Kd (fijos durante toda la iteración), que al realizar los cálculos necesarios, proporciona a la dinámica la potencia de los rotores, siendo además guardados en un registro del configurador para ser mostrados gráficamente. El módulo dinámica recibe además el estado actual del configurador, cuya salida es el estado siguiente, que se guardará también en dicho registro y servirá para actualizar el estado actual. Durante el tiempo de simulación establecido, se repetirá esta iteración y cuando finalice el proceso será mostrado al usuario en forma de gráfica.

Vamos a ayudarnos de un ejemplo para ver el funcionamiento de nuestro módulo de configuración de controlador PID, explicando los resultados que hemos obtenido en la sintonización.

Los datos de entrada que vamos a utilizar para sintonizar un PID van a ser: un tiempo de simulación de 10 segundos, en el que queremos alcanzar una altura de 4 metros, 0.2 radianes de alabeo, 0.1 radianes de cabeceo y 0.2 de viraje. Utilizando la sintonización cualitativa encontramos los siguientes valores de Kp, Ki y Kd para los cuatro controladores diferentes.

Altura:

- Kp=5
- Kd=2,25
- Ki=0,50180

Roll:

- Kp=0,009
- Kd=0,02
- Ki=0,00001

Pitch:

- Kp=0,01
- Kd=0,02
- Ki=0,00001

Yaw:

- Kp=0,1
- Kd=0,3
- Ki=0,00001

En las gráficas superiores de la figura 10, se muestran los resultados obtenidos en función del tiempo para la altura, alabeo, cabeceo y viraje. Además en las cuatro gráficas inferiores, podemos ver la evolución de los rotores.

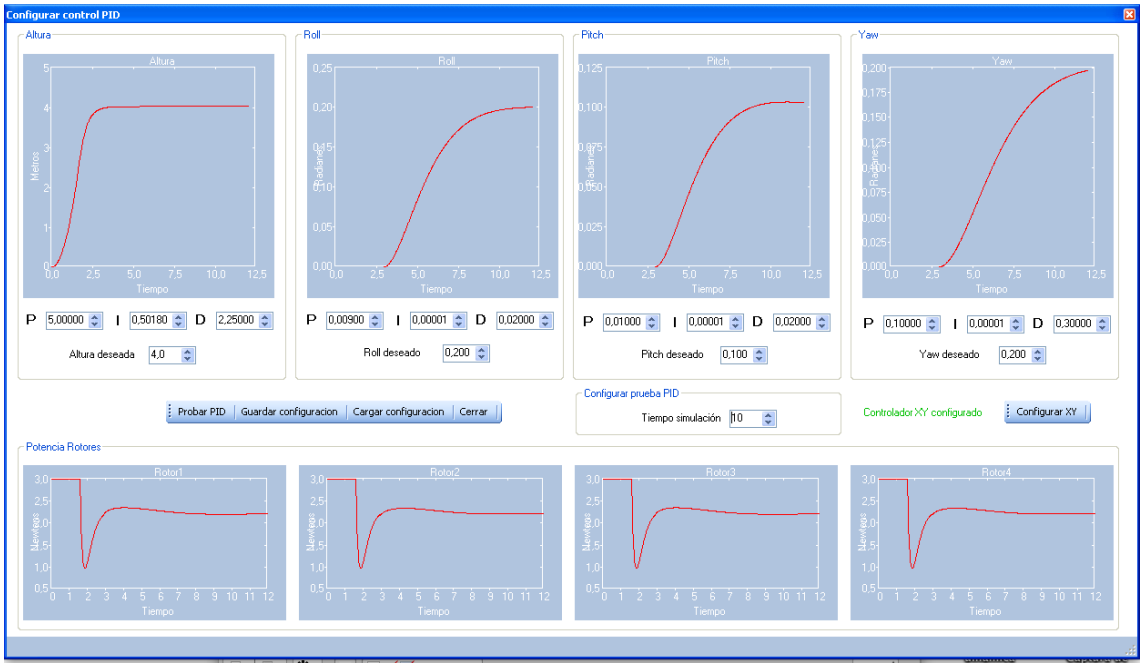

Figura 10: Interfaz configurador del controlador PID

A continuación vamos a explicar los 6 controladores que forman nuestro controlador PID junto al módulo de agregación y que se muestran en la figura 10.

3.2.1.1. Control altura

Devuelve la potencia que se debe aplicar a cada uno de los cuatro rotores para corregir el error en el eje Z, o lo que es lo mismo en la altura. En un cuatrirotor se aplicará la misma potencia a cada rotor para aumentar o disminuir la altura, por lo que el módulo de agregación interpretará la salida de este controlador como la potencia de cada cuatrirotor y servirá como potencia base sobre la que serán sumados o restados los demás valores. Es por ello que lo más prioritario a corregir sea la altura.

En la siguiente gráfica vemos como para una altura de 4 metros y para unos valores de Kp = 5.0, Ki = 0.5018 y Kd = 2.25, el controlador estabiliza la salida en un tiempo aproximado de 5 segundos.

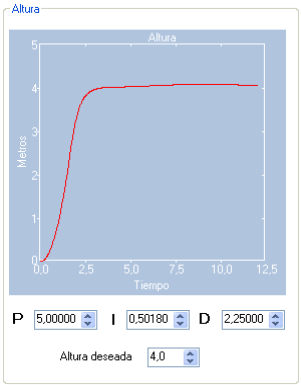

Figura 11: Gráfica del control de la altura.

3.2.1.2. Control del alabeo

Este controlador determina el giro apropiado en el eje X para que el vehículo se encuentre estable y es por ello que la salida deba inferir sobre los motores laterales dos y cuatro, manteniendo los rotores delantero y trasero con la potencia establecida.

El control del alabeo es un controlador dependiente del control X ya que recibirá la salida de este, que le determinará la inclinación para obtener la X deseada. El control del alabeo lo tomará entonces como estado objetivo mientras que el estado actual será la inclinación en dicho momento del cuatrirotor.

Este controlador devolverá un único valor que será sumado a los rotores dos ó cuatro dependiendo de que se quiera girar en sentido horario (derecha) o antihorario (izquierda). Además, y para conservar la fuerza constante y que el vehículo no se desestabilice, lo que sumemos a un rotor, se lo tenemos que restar al otro. De esta manera nos encontraremos con las siguientes situaciones:

- Giro a la derecha
  - Rotor 2 = Rotor 2 + Salida Controlador alabeo.
  - Rotor 4 = Rotor 4 - Salida Controlador alabeo.

- Giro a la izquierda
  - Rotor 2 = Rotor 2 - Salida Controlador alabeo.
  - Rotor 4 = Rotor 4 + Salida Controlador alabeo.

De la misma manera, mostramos con un ejemplo como el controlador de alabeo, estabiliza la salida para un alabeo de 0.2 radianes, con unos valores de Kp = 0.009, Ki = 0.00001 y Kd = 0.02. En este ejemplo se puede ver como se tarda 12 segundos en llegar a la inclinación deseada.

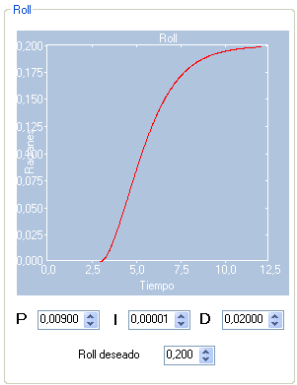

Figura 12: Gráfica de control del alabeo.

3.2.1.3. Control del cabeceo

Este controlador actúa de la misma manera que el controlador del alabeo, pero corrige el giro en el eje Y en lugar del X, actuando entonces en los motores uno y tres. Tal y como veíamos en el apartado anterior, el control depende a su vez del valor

resultante del controlador Y, que le determinará la inclinación necesaria para alcanzar una posición Y fijada. Esta inclinación será el cabeceo deseado, que junto al cabeceo actual del vehículo, calcularemos los diferentes errores para obtener el resultado final.

Del mismo modo que en el alabeo, el valor devuelto será incrementado en un rotor y decrementado en el otro no. De esta manera, disponemos de dos situaciones dependiendo de hacia donde se quiera girar el vehículo:

- Giro hacia delante
  - Rotor 1 = Rotor 1+ Salida Controlador cabeceo.
  - Rotor 3 = Rotor 3 - Salida Controlador cabeceo.

- Giro a la izquierda
  - Rotor 1 = Rotor 1 - Salida Controlador cabeceo.
  - Rotor 3 = Rotor 3 + Salida Controlador cabeceo.

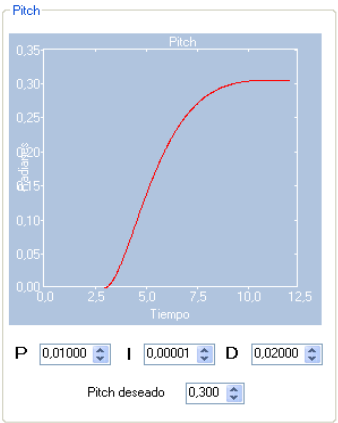

Figura 13: gráfica de control del cabeceo

La siguiente figura muestra como el controlador estabiliza la salida en un tiempo de 12 segundos, para un cabeceo de 0.3 radianes, con unos valores de Kp = 0.01, Ki = 0.00001 y Kd = 0.02.

3.2.1.4. Control X e Y

Este controlador es el encargado de corregir tanto la posición X como la posición Y, cuya salida será la inclinación necesaria sobre dicho eje, es decir el alabeo o el cabeceo, para que el vehículo alcance la posición deseada. Este controlador recibirá el error en ambas posiciones, es decir la diferencia entre la posición deseada y la posición actual del cuatrirotor. El valor que sea calculado por este controlador será interpretado como el alabeo o el cabeceo que debe adoptar el vehículo para que el error disminuya en ambos ejes. Por ello se calculará el error existente entre la salida de este controlador y el alabeo o cabeceo del vehículo, y dicha diferencia es el que se le introducirá al controlador encargado de la estabilidad.

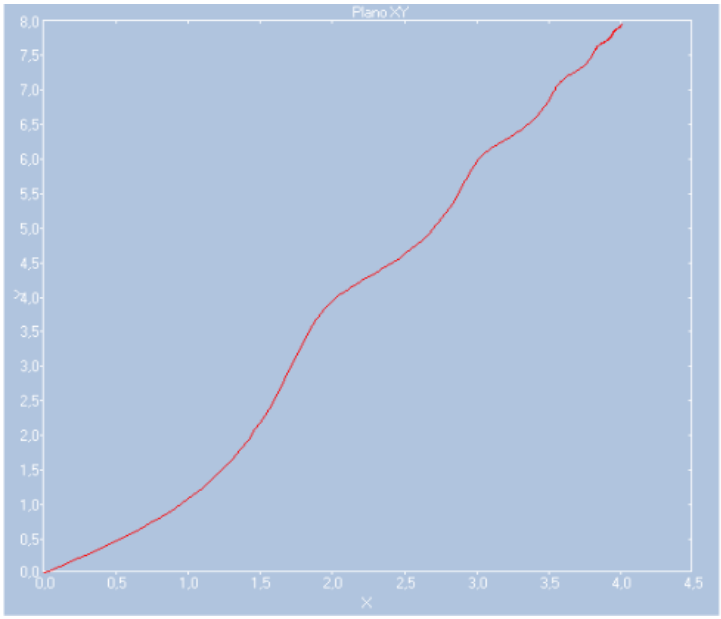

Figura 14: Gráfica del control X e Y

Observando el siguiente ejemplo, podemos ver como el controlador, para una posición de x = 4 metros, y = 8 metros m y una altura de 10 metros, muestra la trayectoria que sigue el cuatrirotor hasta alcanzar la posición objetivo desde una posición inicial x = 0, y = 0 y z = 0.

3.2.1.5. Control integrado (Módulo agregación)

El módulo de agregación determina la potencia que se le debe aplicar a cada uno de los cuatro rotores de acuerdo a los tres valores que recibe de cada controlador: controlador altura, alabeo y cabeceo. El valor de salida de cada uno lo podemos interpretar como la potencia que se debería aplicar para reducir el error en dicho parámetro, y ya que nos podemos encontrar situaciones donde tengamos error en más de un movimiento, se establecerán prioridades para reducirlos sin que aumentemos errores colaterales.

La prioridad principal del módulo de agregación es alcanzar la altura deseada, sin desestabilizarse en ninguno de sus tres ángulos, es decir, alabeo, cabeceo y viraje. A esto es a lo que llamamos una rutina de despegue que facilita el control del cuatrirotor. Una vez que el error en la altura es cercano a 0, agrega las demás señales de control, inclinando y desplazándose en los ejes X e Y, disminuyendo dichos errores. Si durante una simulación, el vehículo se encuentra en la altura deseada y se está controlando la posición X e Y, si se le cambia a una nueva posición X, Y y Z; el controlador volvería a establecer prioridad en controlar la altura y sólo cuando este se situase en el Z deseado, comenzaría a controlar la nueva X e Y.

En la figura 15, mostramos una simulación con un posición objetivo inicial a la que realizamos un cambio de posición a mitad de simulación. Vemos como deja de controlar la dirección mientras disminuye el error en la altura, desviando la trayectoria en X e Y. Una vez controlada la altura, vuelve a rectificar la trayectoria, hasta llegar a la posición indicada.

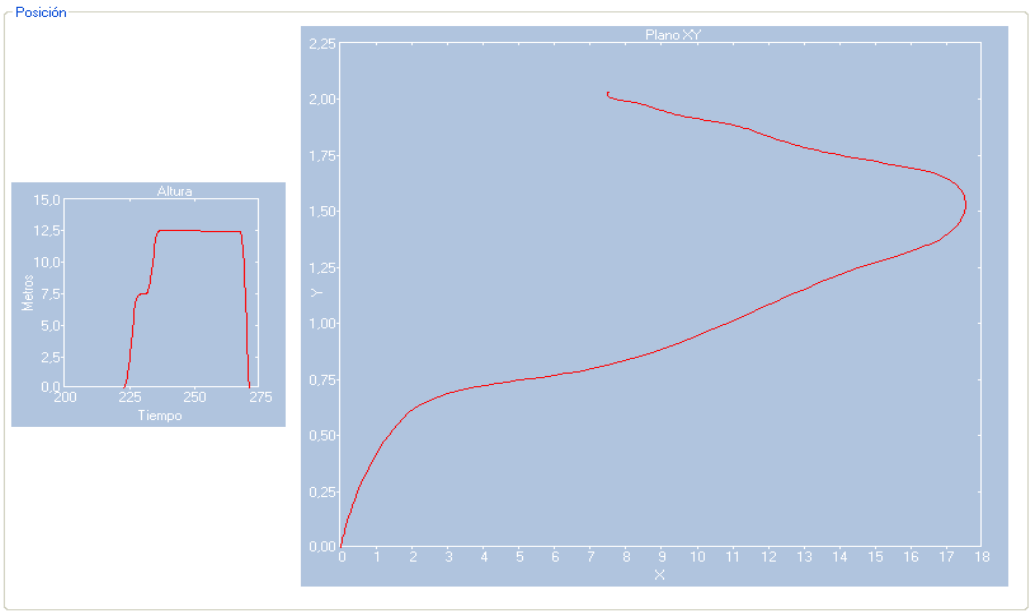

Figura 15: Simulación de un cambio de posición

En la figura 16 observamos como varían los rotores durante una simulación. Los rotores consiguen su máxima potencia durante la rutina de despegue, que como observamos se saturan en 3 voltios. El límite de los rotores viene marcado por el voltaje máximo de la fuente de alimentación del cuatrirotor dividido por los cuatro rotores. Una vez terminada la rutina de despegue observamos como los rotores varían sus voltajes dependiendo de la inclinación deseada. Como en el ejemplo queremos que el cuatrirotor se incline 0,1 radianes de alabeo (Roll) vemos como el voltaje que les llega a los rotores 1y 3 es mayor que el voltaje del 2 y 4, consiguiendo que se incline el cuatrirotor al ángulo deseado.

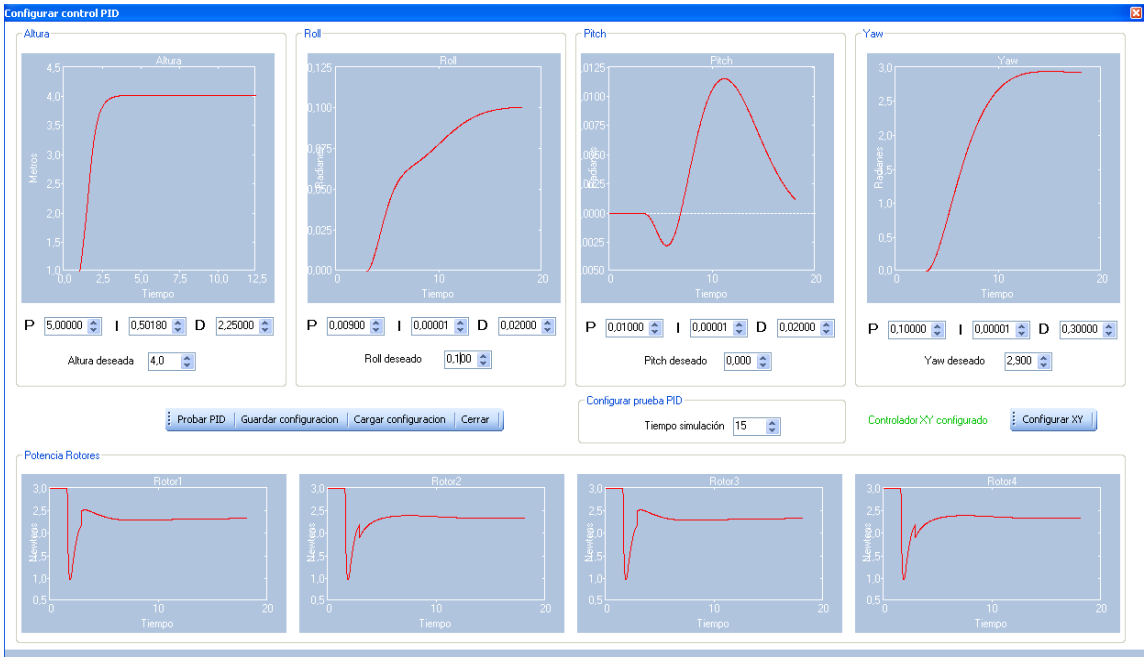

Figura 16: Simulación de un controlador PID.

Debido a la cantidad de parámetros a controlar y a la inestabilidad que presenta el sistema cuando queremos controlar diferentes rangos, no hemos encontrado una configuración fija de un PID que tenga buenos resultados para diferentes rangos de alturas. Por este problema, hemos diseñado un controlador experto con el que obtendremos mejores resultados.

3.2.2. Controlador Experto:

Este controlador selecciona mediante reglas de decisión que controlador debe utilizarse de una biblioteca de PID's. Esta biblioteca estará determinada por PID's configurados para su funcionamiento óptimo en una altura. De esta manera dependiendo de la altura deseada, el controlador experto seleccionará el PID óptimo para esa simulación.

En el siguiente esquema (figura 17) explicamos el funcionamiento de nuestro sistema experto, donde podemos ver como las reglas seleccionan los datos de un PID generando un controlador para su posterior simulación.

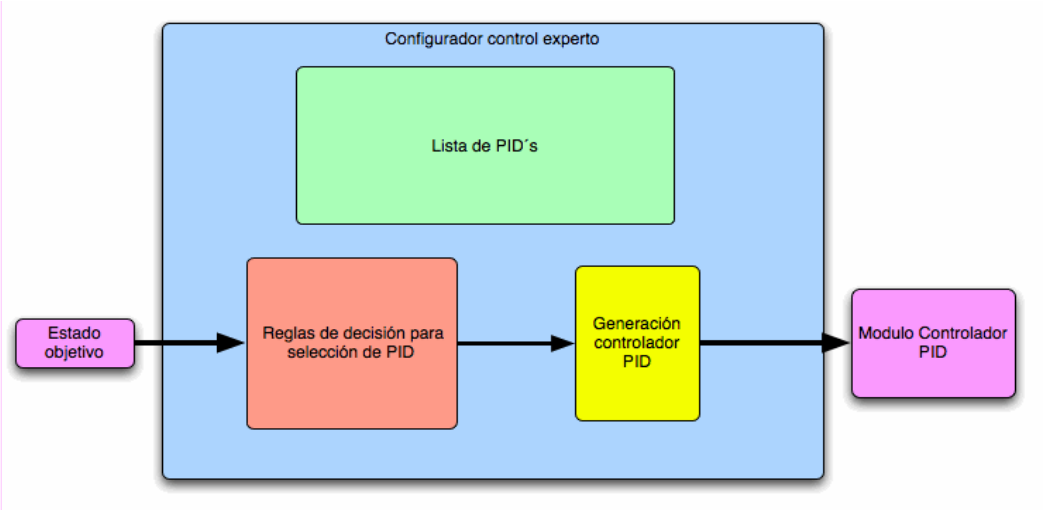

Figura 17: Implementación control experto.

Para poder generar controladores expertos hemos realizado un modulo de configuración de controladores expertos. Antes de crear nuestro controlador experto deberemos hacer un paso previo, que consistirá en sintonizar con el configurador PID distintos controladores que tengan un funcionamiento óptimo para el rango de alturas que queremos ocupar. Una vez tenemos los PID´s en nuestra biblioteca, mediante al interfaz del configurador experto, seleccionaremos los PID's con su rango de activación. Guardaremos nuestro controlador en la biblioteca para poder probarlo en los simuladores.

En la figura 18 observamos la interfaz del configurador con un ejemplo de controlador el cual tiene los siguientes rangos de la altura.

1 De 0 a 6 se activaría el PID 4, ya que es el que mejor resultado tiene en ese rango.

2 De 7 a 21 se activaría el PID10.

3 De 22 a 60 se activaría el PID 30.

4 De 61 a +100 se activaría el PID 100.

Se tomarían estas reglas de decisión ya que el controlador experto observa cual es el PID de rango más cercano a la altura deseada y lo selecciona.

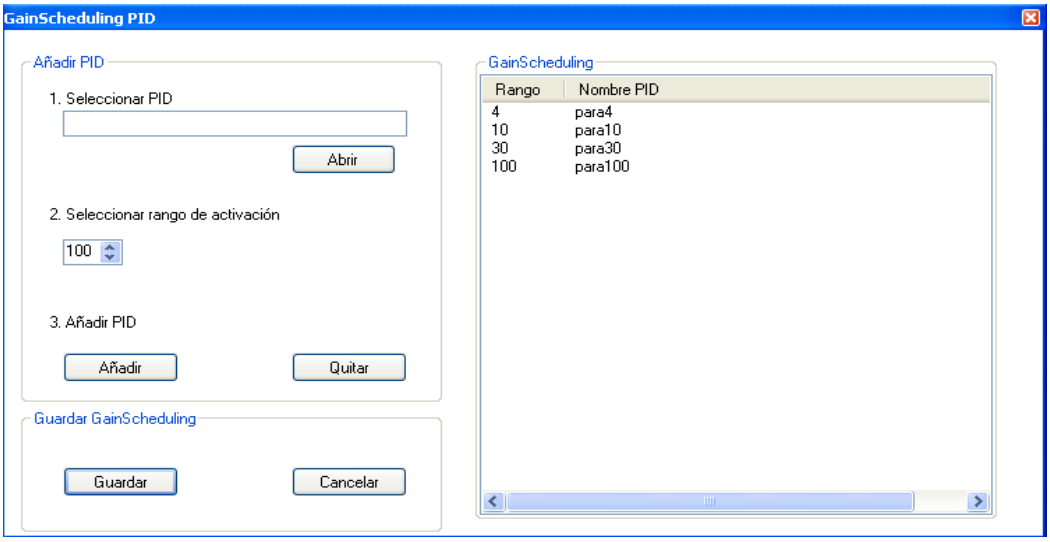

Figura 18: Interfaz del configurador de controladores expertos.

Nuestro controlador experto no es de ganancia programada, ya que el cambio de los parámetros del controlador dependiendo del rango de altura que nos encontramos desestabilizaba el sistema. El cambio de parámetros es muy brusco y no los soporta el sistema, para realizarlo deberíamos suavizar ese cambio y que no desestabilice el sistema.

Luego ya tenemos un controlador inteligente para el cuatrirotor.

3.3 Módulo simulador

El motivo que nos impulsó al desarrollo de este módulo, se debe en gran parte a la necesidad de complementar la funcionalidad principal del proyecto es decir, poder probar distintos controladores con distintas configuraciones.

Tenemos dos simuladores distintos, uno en 2 dimensiones y otro en 3. Vemos la estructura de la implementación de los simuladores en la figura 19.

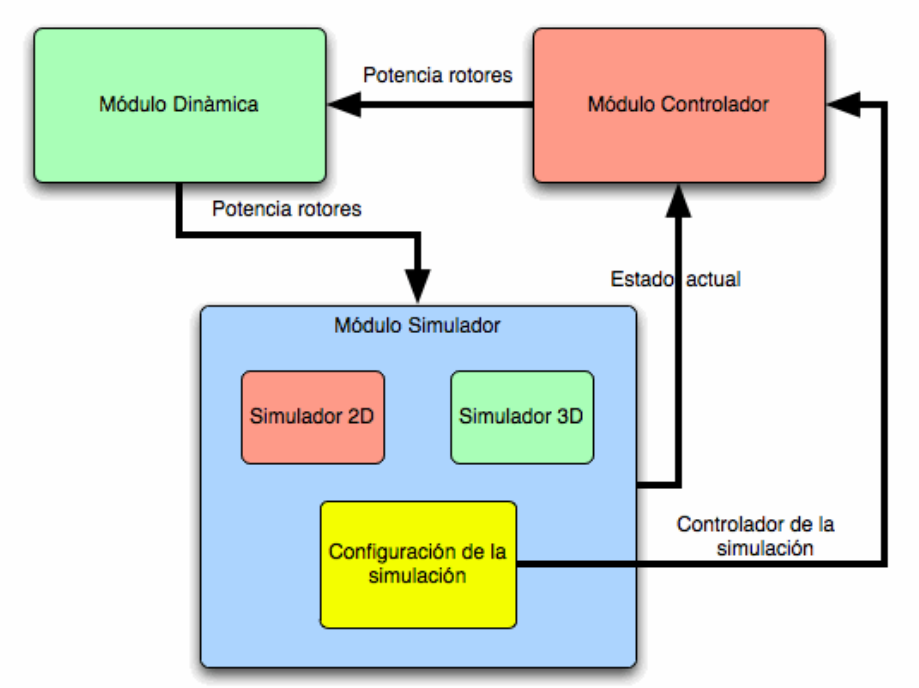

Figura 19: Estructura de la implementación de los simuladores

Vemos en el esquema de la figura 19, como se configura en el simulador el controlador a utilizar, ya guardado en la biblioteca de controladores, y el estado objetivo. Luego el simulador configura el modulo controlador que se comunicara con la dinámica para ir calculando los estado siguientes de la simulación que se muestran en el simulador.

3.3.1. Simulador 2D

Con el simulador 2D observaremos los resultados más exactos en forma de gráficas. Esto nos valdrá para poder estudiar mejor los resultados del sistema.

Para ver como funciona el simulador 2D utilizaremos un ejemplo. Primero seleccionamos el controlador que queremos que dirija el cuatrirotor y la posición final

en la que queremos que termine la simulación. Como simulador elegimos uno que previamente hemos afinado para una altura de 10 metros y en los datos de la simulación introducimos como posición de llegada x = 2, y = 4 y una altura de 11.

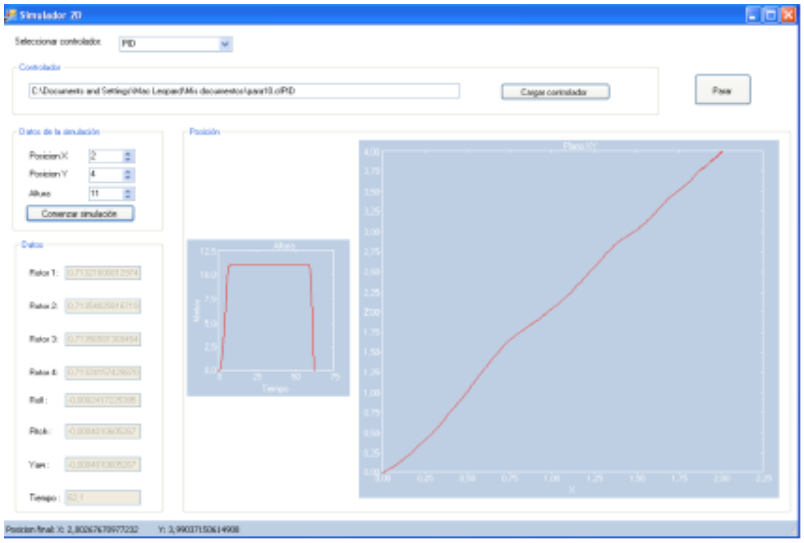

Figura 20: Ejemplo de simulación 2D.

Tal y como vemos en la figura 20, durante la simulación proporcionamos unos datos que ayudan a saber como se comporta el controlador elegido. Estos son las inclinaciones del vehículo (alabeo, cabeceo y viraje) , las potencias de los rotores y el tiempo transcurrido de la simulación.

La funcionalidad del simulador 2D sirve de gran utilidad para ver con mayor precisión el comportamiento del sistema controlado. A diferencia del simulador 3D, que explicamos a continuación, observamos con exactitud los cambios en la trayectoria y por tanto se pueden realizar comparaciones detalladas entre controladores.

3.3.2. Simulador 3D

Con nuestro entorno virtual podemos probar todo lo anterior sin poner en peligro la integridad estructural del cuatrirotor, algo que debemos evitar debido a que es una pieza muy sensible a los golpes y tiene una alta probabilidad de sufrirlos ante una mala configuración de los atributos del controlador.

Además en las pruebas realizadas en el exterior con el cuatrirotor influyen una serie de características que pueden influir en los resultados del estudio de la configuración del vehículo y en su funcionamiento, tales como vientos o turbulencias.

Por tanto podremos visualizar el comportamiento de la configuración de nuestro cuatrirotor, para después poder probarlo en el modelo real, sin temor a que este pueda sufrir daño alguno.

3.3.2.1. Funcionamiento del simulador:

El simulador nos presenta un entorno visual en el que podemos observar el modelo de nuestro cuatrimotor en movimiento, volando sobre un mapa previamente elegido, en el cual además se nos agrega una la función para aterrizar en una plataforma situada en la posición de destino que hayamos elegido anteriormente. En la figura 21 podemos observar todos estos elementos durante la ejecución del programa.

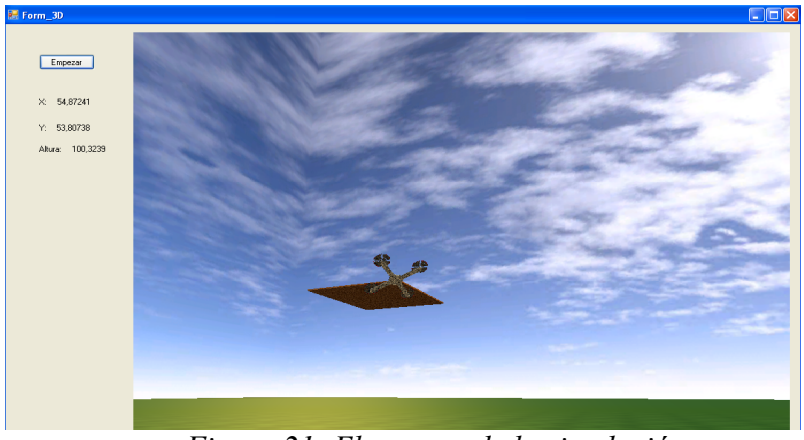

Figura 21: Elementos de la simulación.

Para el funcionamiento de la simulación es necesaria la intervención de los módulos descritos anteriormente:

- Módulo de Control
- Módulo de Dinámica

La combinación de ambos módulos nos permitirá alcanzar el objetivo final descrito anteriormente, probar configuraciones sin riesgo para el cuatrirotor, ni depender de las condiciones medio ambientales.

El dibujo de la ejecución de nuestro simulador queda retratado en la figura 21:

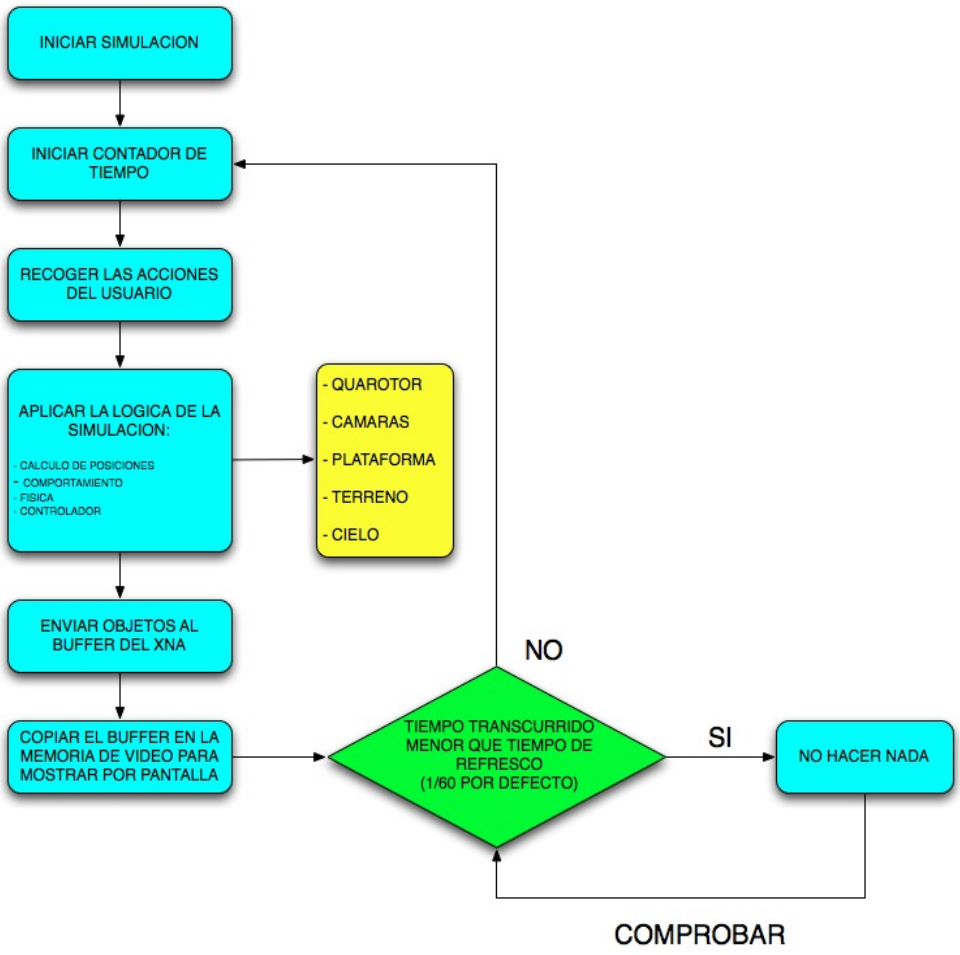

Figura 21: Ciclo de XNA

Como podemos observar en la figura 21 la simulación se trata de un bucle infinito en el que se van actualizando todos los modelos, la posición de las cámaras, se comprueban los botones pulsados, y se repintan todos los objetos en pantalla.

La ejecución del entorno virtual consiste en dibujar el vehículo, calcular la diferencia entre nuestra posición actual respecto a la posición final deseada, hallar la potencia que debemos aplicar a nuestros rotores para movernos a una nueva posición, y repintar nuestro cuatrirotor en la nueva posición recién calculada, repitiendo este proceso hasta que finalicemos la ejecución del simulador.

Hemos dividido en 3 partes distintas el simulador que nos permitirán manejar el mundo virtual de una manera sencilla:

- Manejador de mundo
- Manejador de modelos
- Manejador de cámaras

En la figura 22 podemos observar el modelo que acabamos de describir.

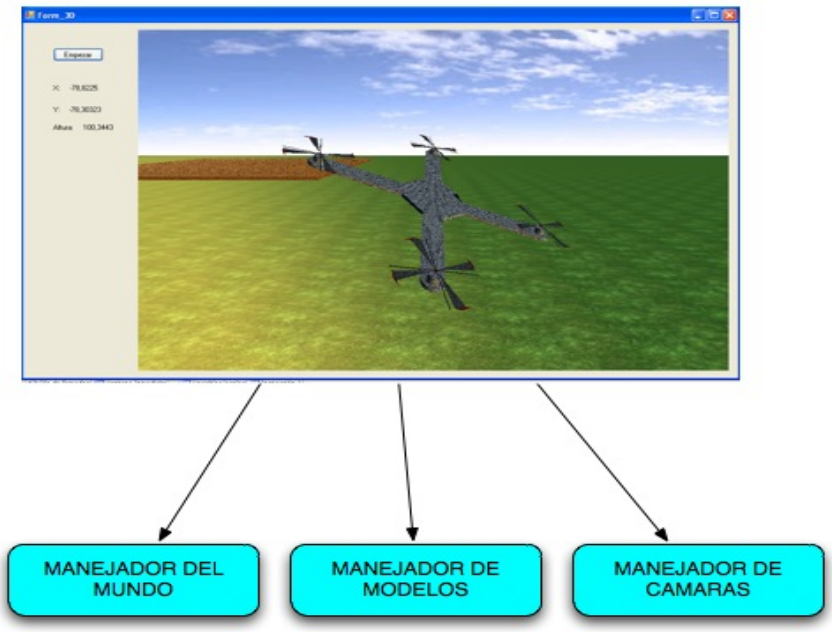

Figura 22: Componentes del simulador.

Todos estos manejadores los añadiremos al planificador de XNA que será el que pintará y refrescará cada 60 frames por segundo nuestro mundo virtual, que a su vez actualizará sus elementos internos con las funciones que hemos desarrollado.

Una vez descrito el modelo general que sigue el simulador proseguiremos con cada parte individual.

3.3.2.2. Manejador de Modelos:

Es el manejador más importante ya que gestiona el modelo del cuatrirotor y la plataforma.

La ejecución de este manejador es bastante sencillo, gracias a la tecnología de XNA que permite un desarrollo rápido de aplicaciones visuales. En base al funcionamiento de refresco de XNA adaptamos nuestro controlador y nuestra dinámica para efectuar los cálculos necesarios, la fuerza que debemos dar a nuestros rotores, y la nueva posición obtenida en base a esta fuerza.

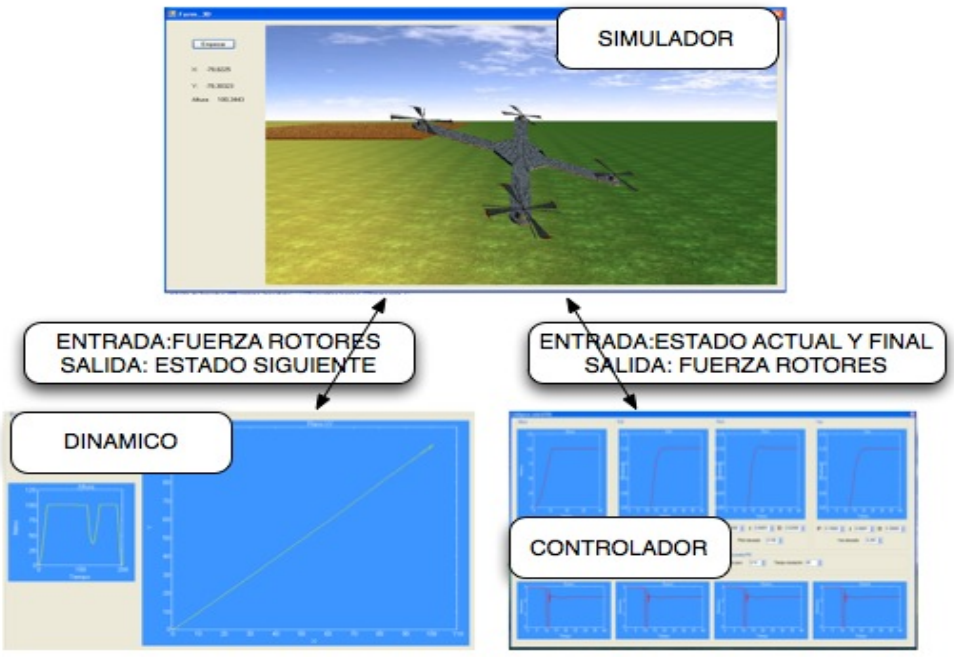

Figura 23: Ejecución del simulador.

En la figura 23 describimos visualmente el funcionamiento del modelo de la nave. Una vez iniciado el simulador ejecutaremos el módulo dinámico pasándole la fuerza de los rotores que habremos calculado con el módulo controlador, y recibiendo como respuesta el estado siguiente. También ejecutaremos el módulo de controlador al cual pasaremos el estado actual calculado con el módulo dinámico, y éste nos devolverá la fuerza que debemos aplicar a los rotores para llegar a nuestra posición objetivo.

Dentro del manejador de modelos la parte más relevante es el cuatrirotor, el cual hereda de un modelo básico, al que añadimos las propiedades del controlador y la dinámica que seguimos. Por tanto podríamos simular varios modelos con distintos tipos de controladores y dinámicas.

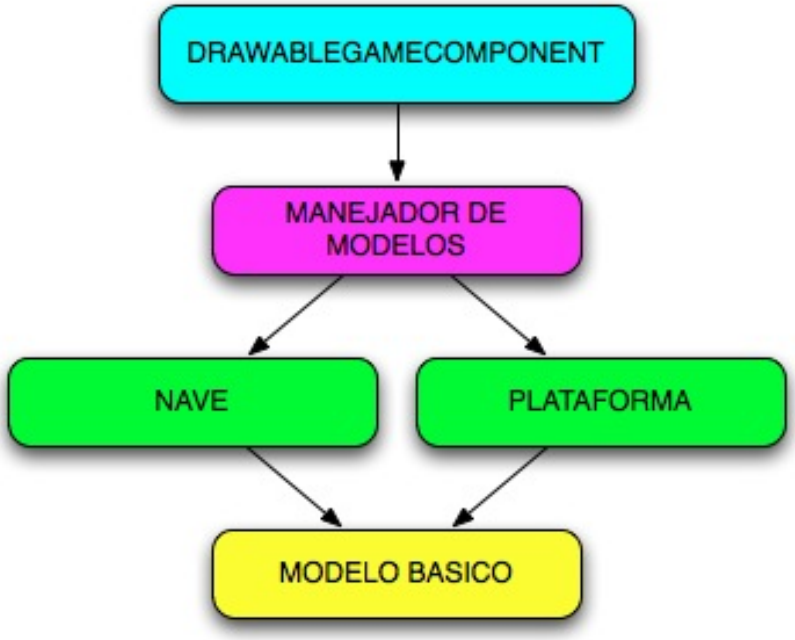

Figura 24: Estructura de clases del modelo en el simulador

En la figura 24 presentamos una parte de la estructura de clases del simulador, la que hace referencia a cuatrirotor y la plataforma, en el gráfico podemos observar que las clases naves y plataforma son hijas y por tanto heredan las cualidades del modelo básico añadiendo además algunas propiedades exclusivas de cada una. También podemos comprobar que el manejador de modelos hereda de DrawableGameComponent para

poder dibujar las clases Nave y Plataforma en la pantalla.

Para calcular esta nueva posición hemos tenido que adaptar la manera hacer los cálculos en Matlab© ya que XNA tiene una ejecución especial y los resultados deben ser iguales en ambos entornos.

3.3.2.3. Manejador del mundo

El manejador del mundo es el segundo que creamos y éste se ocupa de cargar inicialmente el mapa de alturas que elegimos, la textura de fondo, y de manejar la última cámara seleccionada o activa, para poder pintar el mundo desde la perspectiva correcta.

En este manejador se guarda el modelo del terreno y el fondo, para pintarlos en la pantalla cada vez que el XNA refresca los objetos que le hayamos indicado. En este manejador no se actualiza ningún modelo ya que el mundo y el cielo no cambian con ninguna tecla ni ninguna realiza otra acción.

Para mejorar el aspecto visual de la simulación se puede elegir entre distintos mapas, con distintas alturas que nos permiten observar cómo se eleva el vehículo con una mayor perspectiva que respecto a un mapa plano. Para ello agregamos distintos mapas de bits, en los cuales analizamos el color de cada píxel y dependiendo de este color se le asigna una altura en el mapa para después poder dibujar un mapa con relieve. Este análisis nos permite generar un modelo que después agregamos al manejador del mundo, el cual se dedica a gestionar el modelo del terreno y el cielo.

En la figura 25 podemos observar varios mapas de alturas distintos que podemos usar en el simulador.

Figura 25: Mapas de alturas.

Para generar el cielo también creamos un cargador de cielo que nos permita poner como textura de fondo cualquier imagen, aunque esta opción no es configurable ya que no añade ninguna funcionalidad atractiva para el desarrollo del programa.

3.3.2.4. Manejador de cámaras

En este manejador guardamos las 4 cámaras del programa. También permite cambiar entre las distintas cámaras en función de la tecla que pulsemos y permite actualizar la posición de las cámaras que pueden moverse, como la cámara libre y la cámara de seguimiento.

Todas las cámaras agregadas heredan de una clase base y añaden cada una sus particularidades, posición original, teclas de movimiento y tecla de asignación (figura 26).

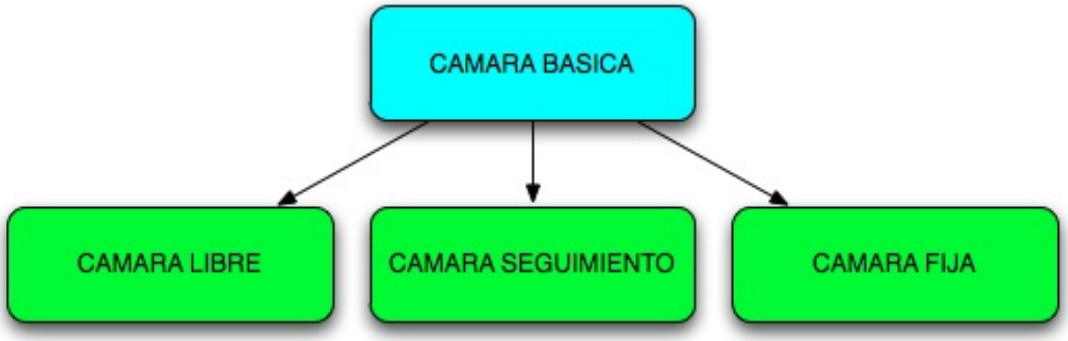

Figura 26: Estructura de clases de las cámaras en el simulador

Para poder visualizar mejor que trayectoria sigue nuestro vehículo se han añadido distintas cámaras al entorno que nos permiten observar la nave desde cualquier posición del mapa. Estas cámaras son cámara con movimiento libre, cámara de cuatrirotor, y dos cámaras fijas. Cada cámara tiene asignada una tecla para poder alternar entre las distintas vistas. Las teclas que permiten seleccionar las distintas cámaras son:

- Vista desde la cámara con movimiento libre (figura 27):

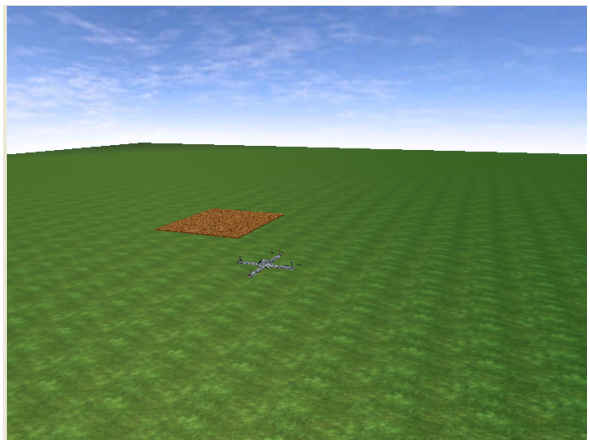

Figura 27: Vista desde la cámara libre

En la figura 27 podemos observar el cuatrirotor y la plataforma desde una posición y ángulo determinada, pero podría cambiarse en cualquier momento usando los controles para la cámara libre y colocarse en cualquier otra posición y cualquier otro ángulo.

- Vista desde la cámara de seguimiento al cuatrirotor (figura 28):

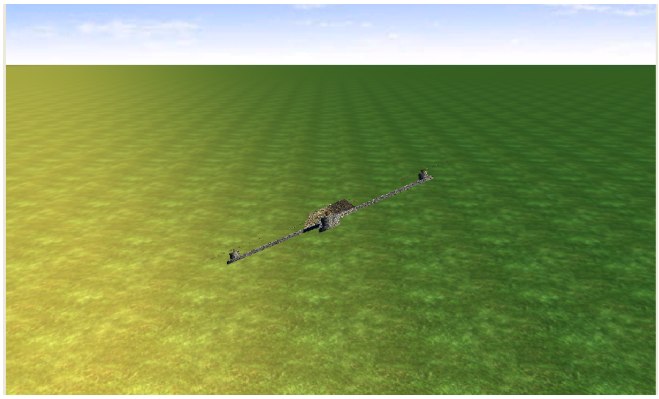

Figura 28: Vista desde la cámara de seguimiento.

La figura 28 se observa el cuatrimotor. Si siguiéramos su movimiento observaríamos como la cámara se mueve al mismo ritmo que el vehículo. El ángulo y la posición de esta cámara son modificables durante la ejecución aunque siempre se moverá de acuerdo a la posición del cuatrirotor.

- Vista desde la cámara fija paralela al suelo (figura 29):

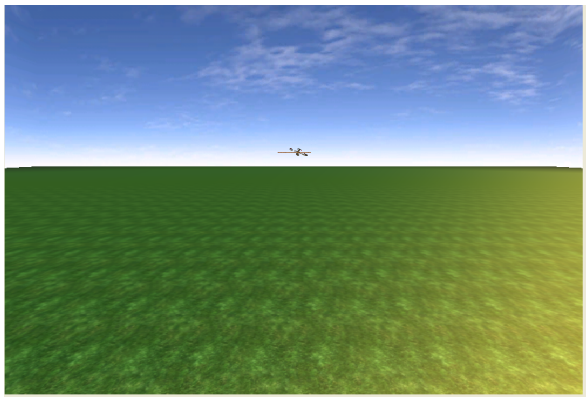

Figura 29: Vista desde la cámara fija paralela al suelo.

La figura 29 es una cámara fija paralela al suelo que permite ver la distancia que hay entre el suelo y el cuatrirotor.

- Cámara fija picada (figura 30):

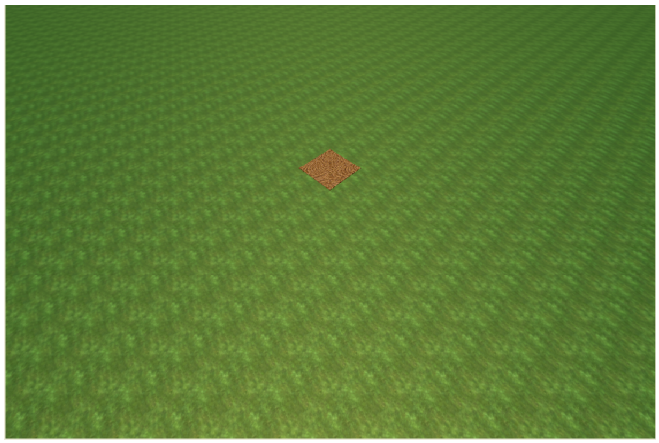

Figura 30: Vista desde la cámara fija picada.

La vista picada que se observa en la figura 30 nos permite tener una vista general de la situación de la simulación.

Debido a que hemos utilizado XNA para desarrollar nuestro entorno hemos tenido que adaptarnos al modelo de programación para XNA. Esto nos ahorra horas de programación pero sin embargo conlleva muchas horas de estudio y compresión de la tecnología ya que era nueva para todos nosotros.

Una de las partes que más tiempo de investigación nos ocupó fue cómo crear una aplicación híbrida que usara XNA y una aplicación con ventanas, debido a que esta mezcla no es muy habitual, ya que suelen ser de un tipo o del otro. Al final se encontró una solución creando un formulario de Windows y dejando que el XNA lleve la gestión de las ventanas.

Para el simulador se desarrolló un modelo tridimensional de cuatrirotor que intentara representar fielmente la figura de nuestro aparato. Su evolución fue rápida gracias al programa Google SketchUp desarrollado por Google.

4. Conclusiones

Hemos construido una aplicación implementada en .NET que permite simular y configurar controladores para el modelo de un cuatrirotor.

Por una lado la aplicación implementa la dinámica del cuatrirotor a partir de un modelo facilitado en Matlab. Esta engloba las ecuaciones dinámicas del sistema, el integrador basado en diferencias finitas y unas limitaciones físicas que debe cumplir el modelo. Para poder simular diferentes cuatrirotores, hemos diseñado un configurador con el que podemos variar los parámetros del modelo.

Además hemos implementado un controlador PID con el que podemos obtener la fuerza de los rotores, permitiendo dirigir la trayectoria del cuatrirotor. Debido a la inestabilidad del sistema hemos desarrollado un sistema experto que por medio de reglas de decisión, aplica PID's diferentes mejorando el comportamiento de este primero.

Hemos elaborado un configurador de controlador con el que podemos sintonizar controladores PID's y sistemas expertos fácilmente con la posibilidad de almacenar las sintonizaciones para posteriormente probarlas en el simulador.

Por último hemos construido tanto un simulador 2D y otro 3D con los que podemos observar la trayectoria que describe el modelo con ayuda de los controladores. El 2D nos aporta precisión en la trayectoria mientras que el 3D nos da una perspectiva visual.

5. Trabajos futuros.

Mejorar o modificar el modelo del cuatrirotor, añadiéndole condiciones físicas como perturbaciones medioambientales.

Sustituir el modelo del cuatrirotor por el modelo de otro vehículo tales como aviones o helicópteros, cambiando el aspecto visual de estos y pudiendo simularlos reutilizando los módulos ya implementados.

Dentro del módulo de control, podemos añadir diferentes controladores nuevos de manera sencilla ya que hemos facilitado esta tarea permitiendo que el módulo controlador sea implementado por cualquier tipo de controlador.

Se puede desarrollar un controlador con ganancia programable que dependiendo de la altura varíe el tipo de controlador realizándose este cambio de manera suave, ya que el sistema se vuelve inestable ante cambios bruscos.

Mejorar la interfaz del simulador 2D y 3D. Añadir nuevos modelos gráficos al vehículo. Construir nuevos escenarios para amenizar el proceso de simulación. Implementar un módulo dentro del simulador que detecte las colisiones entre el vehículo y el escenario.

6. Anexo

6.1. Herramientas utilizadas

6.1.1. Microsoft XNA: Proviene de **X**NA's **N**ot **A**cronymed, XNA no es un acrónimo, son un grupo de herramientas que facilita el desarrollo de juegos de ordenador y videoconsola en un entorno administrado que proporciona Microsoft.

XNA esta basado en .NET y permite el desarrollo para XBOX 360 y .NET Framework 2.0 para Windows. La plataforma incluye un conjunto de bibliotecas de clases específicas para el desarrollo de juegos, evitando así a los programadores la escritura de código repetitivo. El entorno de ejecución es una versión de Common Language Runtime optimizada para los juegos. Es ejecutable de Windows XP, Windows Vista y XBOX 360, pero dado que esta programado para tiempo de ejecución admite su lanzamiento en cualquier plataforma compatible con XNA Framework con mínimas modificaciones.

Para el desarrollo de este proyecto nosotros lo hemos hecho a través de XNA Game Studio 3.0 integrador con Visual Studio 2008. Esta versión fue lanzada el 30 de Octubre de 2008. En XNA 3.0 se puede desarrollar bajo lenguaje C# 3.0 el cual fue usado durante el proyecto y LINQ.

La decisión de utilizar esta tecnología frente a otras fue debido a la gran facilidad que ofrecía para integrarse con el resto del proyecto que iba a ser desarrollado en C#, la gran comunidad de desarrolladores que existe para esta plataforma, lo que permitió consultar un gran número de ejemplos y desarrollos realizados por los integrantes de la comunidad. Gracias a todos estos documentos nos han ayudado a elegir lo que creemos la mejor distribución de nuestro código para el entorno virtual.

6.1.2. Google SketchUp: es un programa de diseño y modelaje en 3D desarrollado por Google es útil para todo tipo de entornos: arquitectónicos, ingeniería civil, videojuegos o películas.

Esta herramienta permite visualizar rápidamente volúmenes y formas arquitectónicas del espacio, por lo que nos pareció una herramienta interesante para el desarrollo rápido del modelo de nuestro cuatrirotor.

Esta diseñado para usarlo de manera intuitiva y flexible facilitando el uso y el aprendizaje frente a otros programas de modelado 3D usados en entornos profesionales.

Gracias a la inclusión de tutoriales se pudo desarrollar rápidamente un modelo de cuatrirotor que nos permitió centrarnos en otros aspectos más relevantes del proyecto.

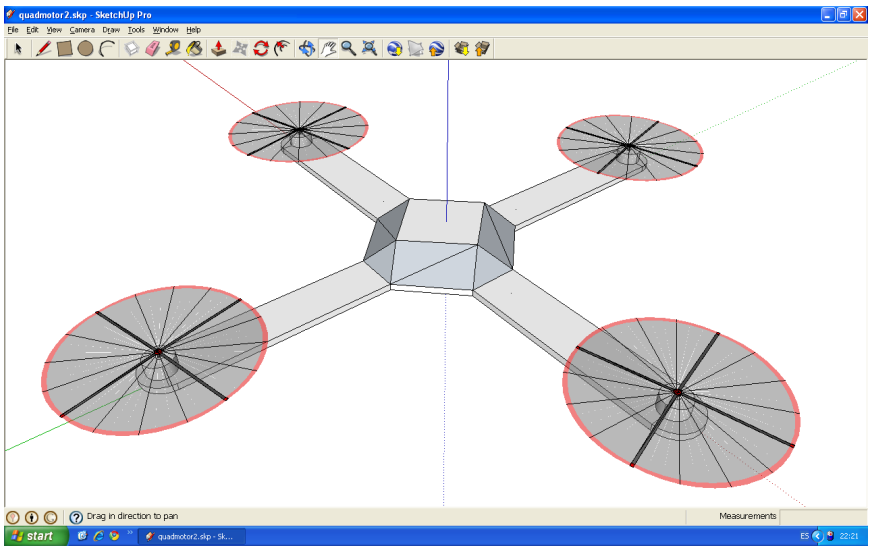

Figura 31: Modelo del cuatrirotor en Google SketchUp.

Algunas característica de Google Sketchup es que podemos realizar los modelos de edificios, coches, personas o cualquier cosa que pueda imaginarse y subidos a la red para compartirlo con la comunidad de modeladores. La ultima versión fue lanzada el día 17 de Noviembre de 2008, que corresponde a su revisión numero 7 e incluye como

novedades respecto a la ultima versión la integración con el online de Google 3D Warehouse y los componentes dinámicos que responden apropiadamente al reescalado.

SketchUp puede correr en Windows XP y Windows Vista como también en el Mac OS X y es binario universal. Aún no hay una versión disponible para Linux.

6.1.3. MatLab se puede considerar como un entorno de cálculo técnico, el cual brinda grandes prestaciones para cálculo numérico y visualización de análisis numéricos, cálculo matricial, procesamiento de señales y gráficos. Una gran ventaja que ofrece MatLab es que consiste de un entorno fácil de usar. Los problemas y las soluciones son expresados de la misma manera en que se escriben matemáticamente, sin el uso de la programación tradicional. MatLab es considerado un sistema interactivo, su elemento básico de datos es la matriz, la cual no requiere dimensionamiento. Este método resulta muy útil, ya que gran cantidad de los problemas numéricos se pueden resolver en una fracción de tiempo, mientras que en programas como C, BASIC o FORTRAN llevaría mucho más tiempo.

Una de las características más importantes de MatLab es su gran capacidad de crecimiento. El usuario se puede llegar a convertir en un autor contribuyente a este crecimiento creando sus propios programas y aplicaciones. MatLab cuenta con una escritura del programa en lenguaje matemático. Se pueden implementar las matrices como elemento básico del lenguaje, de esta manera se pueden ahorrar muchos códigos.

Otro de sus grandes componentes es el editor de propiedades que puede ser utilizado en cualquier momento que se esté lidiando con gráficas de MatLab. El editor de propiedades por separado se puede concebir como una herramienta de mejora de trazado, y asistente de codificación (revisión de nombres y valores de propiedades). Cuando se fusiona con el panel de control, editor de llamadas de función, editor de menú y herramienta de alineación. Resulta una combinación que brinda inigualable control de los gráficos en MatLab.

6.1.4. Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta utilizada en la línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un momento dado.

6.1.5. AnkhSVN 2.0 puede acceder al repositorio a través de redes y siempre desde la interfaz del Visual Studio, lo que nos ha permitido trabajar desde distintos ordenadores. Gracias a esto hemos conseguido mayor progreso, brindándonos la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde nuestras respectivas ubicaciones. Otro aspecto que tiene, es que el trabajo se encuentra bajo el control de versiones, por lo que no hemos tenido razón para temer por que el trabajo haya podido verse afectado frente a cambios incorrectos en los datos, ya que nos permite también deshacer los cambios o volver a versiones anteriores.

7. Bibliografía

- Sistemas de control adaptativo . By: Feng, Gang; Lozano, Rogelio 1999 Elsevier Por: Gang, Feng,; Lozano, Rogelio 1999 Elsevier.
- Control PID Avanzado. Karl J. Åström, Tore Hägglund. Prentice Hall 2009.
- Practical PID Control. Antonio Visioli. Springer – Verlag 2006.
- Learning XNA 3.0 .Aaron Reed. O'REILLY 2009.
- Professional XNA Game Programming: For Xbox 360 and Windows. Benjamin Nitschke. Wilee Publishing 2007.
- XNA Game Studio Express: Developing Games for Windows and the Xbox 360. Joseph Hall. Thomson Course Technology 2008.
- PID Controllers: Theory, Design, and Tuning. Karl J. Åström, Tore Hägglund
- Cinemática y Dinámica del Cuadrotor , David Sanchez . UCM, 2008.