Facultad
de
Ciencias

DESARROLLO DEL MODELO UML/MARTE DE UN SISTEMA CIBERFÍSICO: APLICACIÓN A UN CUADRICOPTERO

(UML/MARTE MODEL DEVELOPMENT OF A CYBER-PHYSICAL SYSTEM: A QUADRICOPTER CASE)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: María Fernández Ortiz

Director: Eugenio Villar Bonet

Septiembre – 2015

Agradecimientos

Nos pasamos la vida esperando, esperando a que salga el sol y la luna, a que llueva, granice o simplemente nieve, a que un día llegue el día en que seamos lo que luchamos y vivamos como soñamos. Al término de esta etapa de mi vida estoy segura de ser lo que he luchado, pero no he llegado aquí sola, por eso quiero aprovechar estas lineas para expresar mi agradecimiento a quienes con su ayuda y apoyo me ayudaron en el camino.

En primer lugar, especial agradecimiento a mi familia y a mi compañero Raúl por su apoyo y dedicación, sin los que este camino no habría sido posible.

En segundo lugar, mi mas sincero agradecimiento a la familia que escogí, aquellas personas que comenzaron siendo conocidos, colegas y compañeros, por las incontables horas de trabajo y ocio compartidas, por las que nos quedan. Especialmente a la chica de la habitación de al lado, Aida, porque estos años no habrían sido lo mismo. A Sandra y a Myriam porque la distancia y el tiempo nunca son excusas.

Y por otra parte, agradecer a Eugenio Villar la oportunidad y el apoyo recibido que ha hecho posible el desarrollo de este trabajo. También quiero hacer una mención especial al grupo de personas que forman el Departamento TEISA por toda la ayuda aportada.

Finalmente dar las gracias a la Universidad de Cantabria por disponer de un gran equipo de profesionales capaces de trasmitir año tras año los conocimientos y capacidades necesarias para alcanzar y superar esta gran etapa.

Resumen

El ritmo al que la tecnología crece hoy en día genera un amplio mercado de necesidades que han de ser cubiertas. Durante los ocho capítulos que conforman este trabajo se explicará de forma detallada el problema que surge con los sistemas ciber-físicos existentes en la actualidad, a continuación y partiendo de un cuadricóptero, sistema ciber-físico real, se generará un modelo UML/MARTE como solución al problema planteado.
El problema principal se basa en la evolución tecnologica, esta evolución genera la necesidad de reducir el tiempo que un producto tarda en salir al mercado lo que conlleva a que su diseño no esta creado para cubrir todas la necesidades de la demanda del mercado, sino aquellas mas básicas. La falta de algunas características concretas en los productos implica que una vez adquirido por parte del consumidor este se dedique a adaptarlo a sus necesidades. Pero hablar de sistemas ciber-físicos implica hablar también de sistemas con un coste alto por lo que antes de realizar cualquier cambio sobre un aparato real se recomienda realizar pruebas sobre sistemas simulados evitando que el producto se pueda quedar inutilízable de por vida.
Por este motivo la generación del modelo UML permitirá, a futuro, realizar proyectos sobre el cuadricóptero corriendo el menor número de riesgos posibles.

Palabras clave: Ciberfísico, Cuadricóptero, UML/MARTE, PHARAON, Sistema, Entorno de aplicación.

Abstract

The pace at which technology is growing nowadays produces a wide market that needs to be covered. During the eight chapters of this work will be explained in detail the problem that arises with existing cyber-physical real-time systems currently, being one of the most famous the quadricopter. Its UML/MARTE model will be generated to solve some of its issues.
The main problem cames from the technological progress and the market necessities, which determines the design by covering only the most imporant features. The absence of some specific features in a product means that once it is acquired it still needs to be modified or extended to fulfill every specific requirement. Cyber-physical systems are narrowly linked with high costs, therefore most of them need simulators to save resources, test in a secure environment and increase the lifetime.
For this reason the UML model will allow to do some projects with the cuadricopter running the fewest possible risks in the near future.

Keywords: Cyber-physical, Cuadricopter, UML/MARTE, PHARAON, System, Application Environment.

Estructura del Documento

- El capítulo 1 esta compuesto por los motivos principales que han llevado a cabo la realización de este trabajo.
- El capítulo 2 muestra los objetivos a cumplir por el proyecto.
- El capítulo 3 esta compuesto por una pequeña introducción que tiene como objetivo situar al lector en el entorno del trabajo.
- El capítulo 4 define en varias secciones la arquitectura del cuadricóptero sobre el que se realiza el modelado. La descripción de su estructura se distingue por los componentes mecánicos, eléctricos e informáticos.
- El capítulo 5 muestra a lo largo de sus tres secciones la tecnología usada y los pasos llevados a cabo para finalmente generar el modelo UML/MARTE.
- El capítulo 6 se centra en la descripción del modelo físico gracias al cual se puede realizar una simulación en 3D del funcionamiento del cuadricóptero.
- El capítulo 7 esta compuesto por una pequeña descripción de la librería usada para generar un entorno en 3D capaz de interpretar los datos generados por el modelo físico anterior.
- El capítulo 8 muestra como se ha llevado a cabo la captura de datos del cuadricóptero y los resultados obtenidos tras aplicar el modelo generado.
- El capítulo 9, ultimo capítulo de este trabajo esta compuesto por dos secciones, la primera de ellas muestra las conclusiones alcanzadas y la segunda define algunos de los trabajos futuros que se podrían llevar a cabo a partir después de este.
- En la ultima parte de este trabajo se añade la bibliografía consultada y los anexos referentes al modelo UML/MARTE.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación

El ritmo al que la tecnología crece hace cada vez más difícil especificar correctamente un nuevo sistema antes de ser lanzado al mercado, esta dificultad viene dada por la reducción del *Time to market*[1].

La creación de nuevos productos se puede llevar a cabo mediante dos procesos; el primer proceso implica la creación de un producto completo con todas la funcionalidades que se requieran, esta opción conlleva invertir tiempo y dinero en investigación y desarrollo, el segundo proceso se basa en crear el producto con las funcionalidades mas básicas del sistema que satisfagan la demanda del mercado y lanzar el producto lo antes posible. Este ultimo proceso es el mas utilizado en el mercado tecnológico debido su velocidad de crecimiento, es decir, si se le dedica mucho tiempo a la creación de un producto en concreto cuando este salga al mercado su demanda ya habrá sido cubierta por algún prototipo con funcionalidades básicas creado por la competencia y el producto no resultará rentable. En ningún caso, la reducción del *time to market* debe ser un motivo para no simular y verificar sistemas críticos antes de lanzarse a su producción.

Otra de las principales dificultades que genera el gran crecimiento de la tecnología es el modelado de sistemas ciberfísicos (CPS). Estos sistemas se definen por su iteración continua y dinámica con el entorno, por lo que además de tratarse de sistemas que se ejecutan en un determinado plazo de tiempo (sistema de tiempo real) deben asegurar un comportamiento fiable, seguro, eficiente y previsible al entorno cambiante.

La existencia de estas dificultades crea un espacio vació en el mercado que debe ser cubierto. La motivación de este trabajo se basa en acercarse a ese espacio generando un modelo de un producto real, ciberfísico, a través del lenguaje unificado de modelado (UML), lenguaje que hasta el momento solo se ha utilizado para modelar sistemas únicamente electrónicos.

[1]Tiempo de mercado, es el tiempo que tarda un producto en salir al mercado desde su creación.

Capítulo 2

Objetivos

El objetivo de este trabajo se centra en la especificación simulable de un cuadricóptero real a través de un modelo UML/MARTE, *Modeling and Analysis of Real-Time and Embedded systems*.

Se trata por tanto de un proyecto de re-ingeniería SW que permitirá la verificación, simulación, análisis de prestaciones y generación automática de código sobre plataformas heterogéneas ya que, para su implementación, se usarán herramientas software (eSSYN) generadas por el grupo de Ingeniería de Microelectrónica del departamento de Tecnología Electrónica e Ingeniería de Sistemas de Automática (TEISA) de la Universidad de Cantabria(UC).
Concretamente se usan dos herramientas, la primera de ella se trata de PHARAON un perfil que amplia las características de UML/MARTE para la especificación de sistemas de tiempo real y embebidos, la segunda herramienta se trata de un generador de código ejecutable y librerías de comunicación del sistema modelado.

Con este modelo se pretende probar la capacidad de estas herramientas contra sistemas complejos de tiempo real e incrementar la certidumbre de haber realizado una implementación correcta.

Además la generación del modelo simulado servirá como material de apoyo a futuras asignaturas que traten de realizar proyectos con el cuadricóptero, de forma que sus pruebas sean realizadas primero en el entorno simulado (entorno controlado) reduciendo así la probabilidad de generar daños físicos sobre el cuadricóptero real.

Capítulo 3

Introducción

Los *UAVs* [1], del inglés *Unmanned Aerial Vehicle*, se definen como el conjunto de aeronaves que vuela sin tripulación. En sus comienzos estas aeronaves se crearon con fines militares. Durante la Primera Guerra Mundial se usaron como una forma mas segura de volar, ya que no requerían de piloto, además debido a su estructura y tamaño estas aeronaves consumían menos combustible y tenían una aerodinámica mayor que les permitía realizar maniobras aéreas con mas facilidad.

Debido a los avances tecnológicos producidos a lo largo de los años han surgido dos variantes de los UAVs, los que son controlados desde una base de manera remota y los que vuelan de forma autónoma usando planes de vuelo preprogramados, es decir son capaces de despegar, volar y aterrizar automáticamente.

En la actualidad se ha extendido el uso de este tipo de aeronaves, de militar a civil, los UVAs pueden comunicarse con una base y transmitir información en tiempo real relativa al estado, posición, velocidad de aire, etc. incluso si se dispone de la tecnología apropiada (cámaras, sensores...) se pueden enviar datos relativos sobre imagen (térmica, óptica, etc).

Los *UAVs* se pueden clasificar de diferentes maneras cabe destacar las clasificaciones según el tamaño, la autonomía o el fin al que son destinados. Con respecto su utilidad, fin al que son destinados, nos podemos encontrar desde aeronaves militares como se ha visto anteriormente, destinadas al control y ataque sobre el enemigo, hasta juguetes, destinados al entretenimiento y diversión de los mas pequeños o no tan pequeños.

Este trabajo se centra en analizar y simular la variante de los *UAVs* que recibe el nombre de cuadricóptero, o *Cuadricopter* por su termino en ingles, esta variante se caracteriza por propulsarse por 4 motores que reducen los problemas de inestabilidad en los helicópteros convencionales.

Para el modelado del sistema se usará UML[1], *Unified Modeling Language*, concretamente la variante UML-MARTE[2], como su nombre indica se trata de la variante que permite el modelado tanto software como hardware del sistema.

[1]Lenguaje Unificado de Modelado que se usa para visualizar, especificar, construir y documentar un sistema.
[2]Modeling and Analysis of Real Time and Embedded systems.

Capítulo 4

Arquitectura del Cuadricóptero

Un cuadricóptero es una variante de los helicópteros, la diferencia que existe entre ellos radica en el numero y posición de los motores[1] que usan para realizar el vuelo.

Figura 4.1: Cuadricóptero

Los helicópteros típicos poseen un motor principal situado en el centro del mismo que se encarga de la elevación del aparato y otro motor en la cola encargado de la realización de los giros y su estabilidad, por su parte el cuadricóptero posee cuatro brazos, cada uno de ellos tiene en su parte final un motor y una hélice. La estabilidad de esta variante se realiza configurando la dirección e intensidad de giro de las cuatro hélices, dos de ellas siempre giran en sentido de las agujas del reloj y las otras dos en sentido contrario.

La maniobrabilidad de un cuadricóptero [2] con respecto a los helicópteros convencionales es altamente distinguible gracias a la estabilidad que le producen los motores girando de forma independiente tal y como veremos mas adelante, en cambio su funcionamiento a nivel técnico se complica debido a su composición a partir de componentes hardware (mecánicos y eléctricos) y software (informáticos).

[1] En la tecnología a la que nos referimos se suele utilizar el termino *rotor*, es decir, la parte física que gira de un motor donde se instalan las hélices, para la realización de este trabajo ambos términos se refieren al motor en su conjunto.

El desarrollo de este proyecto se ha llevado a cabo mediante el análisis de un **MK Quadro-Copter XL** (MK-XL), ver figura 4.1 fabricado por MikroKopter, empresa alemana que se dedica al desarrollo y distribución de productos, hardware y software, para multicópteros.

A continuación se detallan los componentes mecánicos, eléctricos y electrónicos e informáticos que conforman el cuadricóptero, tomando así una visión mas real de los elementos a modelar.

4.1. Componentes mecánicos

El conjunto de componentes mecánicos constituye una de las partes fundamentales para la construcción de cualquier aparato electrónico ya que se trata de los elementos básicos que conforman la parte física denominada chasis.

4.1.1. Estructura

La estructura de cualquier aparato electrónico es la parte donde se ensamblan y apoyan el resto de componentes.
Su principal objetivo es la reducción de las vibraciones que producen los motores, por lo tanto el material empleado para éste ha de ser fuerte, rígido y ligero.

La estructura básica del MK-XL esta compuesta por elementos de fibra de carbono, ver figura 4.2, cumpliendo así con la propiedad del tipo de material. Los elementos que constituyen dicha estructura son los siguientes:

- **Estructura rígida**, *frame*, esta compuesta por dos placas circulares rígidas y los 4 brazos principales donde se conectan el resto de componentes tanto físicos como electrónicos.
- **Pata**, *flexlander*, compuesta por dos arcos unidos en su base mediante dos varillas, encargadas de mantener la estructura rígida del aparato.
- **Carcasa**, *transparent cover*, se trata de la cobertura que envuelve cada uno de los elementos electrónicos, su función es la de proteger dichos elementos de las inclemencias del entorno (meteorológicas, choques físicos,...).

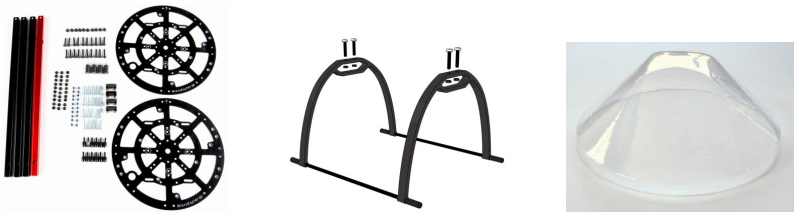
4.1.2. Hélices

El conjunto de hélices de un cuadricóptero tiene como tarea el control del mismo, en función de la velocidad y la resistencia a la que circulen. El MK-XL se impulsa gracias al giro de 4 hélices tipo *12x4,5 EPP-NY*, ver figura 4.2d.

4.1.3. Motores

El cuadricóptero tiene cuatro motores *MK-3638*, ver figura 4.2e. En cada uno de ellos se conecta, a través del rotor[2], una hélice.
La función del motor es transformar la energía eléctrica en mecánica, esta ultima energía es la encargada de hacer que las hélices se muevan a una determinada velocidad.

[2]Componente que gira, rota, en un motor.



(a) Estructura Rígida (b) Pata (c) Carcasa

(d) Hélices (e) Motores

Figura 4.2: Componentes Físicos

4.2. Componentes eléctricos y electrónicos

Los componentes eléctricos[3] se dividen en dos tipos, los discretos aquellos que están encapsulados uno a uno (transistores, resistencias, etc), y los integrados que forman conjuntos más complejos a partir de la unión de componentes discretos, por lo tanto, se denomina componente electrónico o eléctrico al conjunto de elementos electrónicos que forman cada una de las placas físicas que controlan el funcionamiento del Cuadricóptero.

4.2.1. Batería

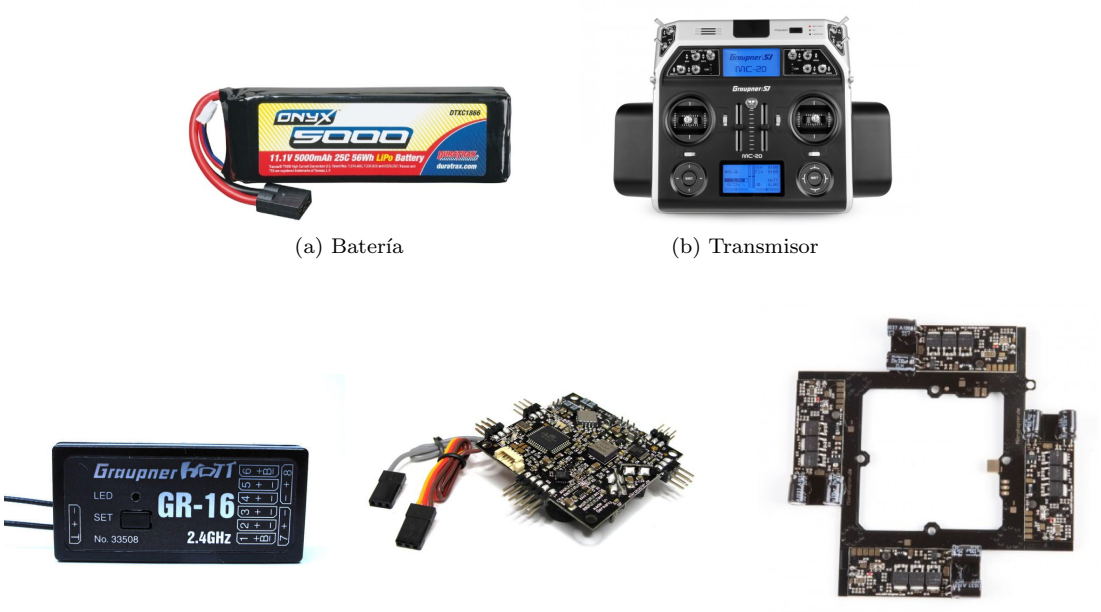
La batería, para el cuadricóptero MK-XL una *5000 mAh LiPo* (ver figura 4.3a, es la encargada de alimentar el resto de componentes eléctricos.

4.2.2. Control remoto

El control remoto (RC) es el elemento encargado de controlar el funcionamiento de cuadricóptero, además también tiene la función de informar al usuario, a través de una pantalla LCD, del estado en el que se encuentra el aparato. El estado incluye entre otras la altura a la que circula, la batería que esta consumiendo, la potencia de cada motor y la posición en la que se sitúa (si incluye GPS). Todo control remoto esta formado por dos elementos:

- **Transmisor**, encargado de transmitir las ordenes al cuadricóptero para un correcto funcionamiento. El transmisor, emisora, del MK-XL es un MC-20, ver figura 4.3b.
  Sus características mas destacables son la disposición de 12 canales por lo que realiza la comunicación, el radio de alcance de 3km, su pantalla LCD por la que muestra datos en tiempo real y su capacidad para permitir conexiones bidireccionales, es decir, además de enviar las ordenes al receptor se encarga de recoger de éste la información del estado y mostrársela al usuario. Este transmisor además permite ajustar los parámetros de configuración del cuadricóptero sin necesidad de conexión a un ordenador.

[3]Dispositivo que forma parte de un circuito electrónico.



(a) Batería (b) Transmisor

(c) Receptor (d) Flight Ctrl V2.5 (e) Brushless-Controller V2.0

Figura 4.3: Componentes Electrónicos

Una de las características limitantes de las emisoras es el numero de canales que soportan, en el caso del MK-XL no resulta un problema ya que como se ha visto soporta hasta 12 canales, cabe destacar que el numero mínimo de canales a utilizar para cualquier sistema de vuelo son dos, uno dedicado al giro y otro dedicado a la aceleración, aunque por lo general se utiliza un canal por cada servo[4].

Con respecto a la emisora MC-20 se analizarán los 4 canales que usa para transmitir la información correspondiente a los movimientos del cuadricóptero.

- **Receptor**, encargado de recibir las ordenes que el usuario envía través del transmisor. El receptor del MK-XL es un MX-16, ver figura 4.3c, además de recibir ordenes es el encargado de enviar al transmisor información en tiempo de ejecución sobre el aparato.

La distinción de estos elementos por funciones, a pesar de que ambos transmitan y reciban información, se lleva a cabo por la capacidad e importancia de realizar dichas funciones, es decir, mientras que en receptor solo transmite información, el transmisor envía parámetros de configuración (información sensible), lo mismo ocurre para el caso contrarío cuando el receptor lo que recibe son los parámetros de carácter sensible y el transmisor solo parámetros de información sobre el estado. Cuando la información transmitida es de carácter sensible esta tiene mayor prioridad para ser enviada y recibida.

[4]Dispositivo encargado de mover una parte especifica de un sistema RC, de forma proporcional al movimiento que se realiza en la emisora.

4.2.3. Placas de control

Las placas de control son la pieza fundamental para el funcionamiento del aparato. Estas placas se dividen en dos grupos, básicas y extras, las básicas son aquellas que todo cuadricóptero debe tener, y las extras como su propio nombre indica son las que el usuario decide poner con funcionalidades extras para su aparato, por ejemplo, la placa encargada de la navegación (*NaviCtrl*) que integra la orientación GPS del cuadricóptero.

Las placas se conectan de forma jerárquica, es decir, su conexión se configura respecto al orden en el que transmiten la información. El MK-XL tiene integradas únicamente las placas básicas, las cuales se detallan a continuación:

- **Controlador de vuelo**(FC), como su propio nombre indica es la placa encargada de controlar el vuelo, por lo tanto la pieza mas importante en el cuadricóptero. El MK-XL dispone de la *Flight Ctrl V2.5*, ver figura 4.3d, las funciones principales que realiza son; procesar la información que el usuario le envía a través del radio control y enviar las ordenes correspondientes al controlador de los motores, además recibe información de los motores que procesa en tiempo real para enviar nuevas ordenes a estos y al radio control.

  Los principales elementos mas importantes que conforman esta placa son los buses de conexión PPM e I2c que se describen en la sección 4.3 y el controlador *(*PID) [3].

  El controlador *proporcional(P), integral (I) y derivativa (D)* [4] se integra en la placa principal, este controlador se encarga ayudado por el giroscopio[5], que se integrado en la misma, de gestionar la estabilidad del cuadricóptero mediante la realimentación el sistema con los parámetros recibidos de los motores, de forma que el sistema recalcula los valores de control de los motores, es decir, su objetivo es lograr un bucle de control que corrija eficazmente y en el mínimo tiempo posible los efectos de las perturbaciones, errores producidos.

  - P, constante de proporcionalidad, se trata del valor de la ganancia o el porcentaje de esta necesario para corregir el error.
  - I, constante de integración, es la velocidad con la que se repite la acción proporcional. Si se aplica en el momento adecuado el error se reducirá a cero.
  - D, constante de derivación, se refiere a la acción de control proporcional al cambio de rango del error, es decir, se trata del efecto estabilizante.

  Por lo tanto el controlador de vuelo es el que se encarga de, partiendo de un estado inicial, generar las acciones necesarias para llegar a un estado deseado.

- **Controlador de motores**(BL), uno por cada motor, tiene la función de enviar al motor la corriente necesaria para su correcto funcionamiento, concretamente el *Brushless-Controller V2.0*, ver figura 4.3e, esta diseñado para optimizar el uso del motor MK-3638 estableciendo los puntos de ajuste de forma instantánea al motor y controlando los picos de tensión que se dan lugar cuando los cambios son muy rápidos.

[5]Dispositivo mecánico que se utiliza para medir, mantener y cambiar la orientación en el espacio de algún aparato o vehículo.

4.3. Componentes informáticos: Buses y protocolos de comunicación

Todo sistema ciberfísico esta formado por personas que conforman los recursos humanos, hardware al que pertenecen los recursos físicos, y software que esta compuesto de recursos lógicos.

Hasta ahora se han definido los componentes físicos y electrónicos que trabajan juntos para producir datos de salida procesados, estos elementos conforman el denominado hardware del sistema. Estos datos procesados se obtienen gracias al funcionamiento de los recursos lógicos, software.

Sin embargo aunque en la definición de componente informático se incluya el hardware del aparato, por distinción con el resto de componentes, componente informático, se refiere únicamente al software encargado del procesamiento de datos para la obtención de resultados finales.
Concretamente para el cuadricóptero se analizan las señales, buses y protocolos que se utilizan para la comunicación de los componentes electrónicos.

Las comunicaciones mas importantes que se dan en el cuadricóptero son entre el control remoto, el controlador de vuelo y los controladores de los motores, la primera comunicación se realiza mediante una señal modulada y la segunda mediante el uso de un bus al que se conectan esclavos y maestros.

4.3.1. Control remoto(RC) - Controlador de vuelo(FC)

La comunicación entre el control remoto y la placa principal del cuadricóptero, *Flight Ctrl*, se realiza mediante el envío de una señal en 2,4Ghz codificada mediante la modulación por posición de pulso (PPM),*Pulse Position Modulation*, método desarrollado por la NASA hace mas de 40 años y el mas usado en radio control.

La señal PPM [5] es una señal de baja frecuencia que se encarga de codificar, en forma de tren de impulsos, la posición de los diferentes controles de la emisora (palancas, interruptores, etc.). El tren de impulsos esta compuesto por tantos impulsos como canales, además se añade un ultimo impulso que marca el inicio de cada tren (sincronía). La señal se transforma, a posteriori, en radiofrecuencia de esta forma se envía al receptor, el cual será el encargado de su decodificación para una correcta interpretación.

La señal PPM tiene muchas variantes en función de como se transmiten y analizan los pulsos de bits. La variante mas usada, idea inicial de la señal PPM, se caracteriza por poseer una anchura fija para cada impulso (1-2ms) de manera que la señal, como su nombre indica, varía por la posición en la que se encuentre el pulso[6] de la señal dentro del impulso, el decodificador será el encargado de obtener esta variación.

La señal PPM que genera el cuadricóptero se trata de una variante en la que el tiempo del pulso de la señal varía, es decir, cada canal transmite un impulso diferente delimitado por un periodo, margen de seguridad[7], de tiempo fijo. Esta señal se analiza con mas detalle en la sección 8.1.1

[6]Periodo de tiempo en el que se recibe corriente eléctrica, se interpreta con un bit a 1.
[7]Periodo de tiempo por el que no circula corriente eléctrica, se interpreta con un bit a 0.

4.3.2. Controlador de vuelo(FC)

Tal y como se indicaba en la sección anterior el controlador de vuelo representa la parte mas importante, además de ser la pieza hardware principal del aparato, contiene el software necesario para que se pueda realizar el vuelo de forma correcta, es decir, tanto la lógica de vuelo como de su control se encuentra cargada en el software del procesador de esta placa.

4.3.3. Controlador de vuelo(FC) - Controladores de motor(BL)

La comunicación del controlador de vuelo y los controladores de motores se realiza mediante el bus de comunicación en serie Ínter-Circuitos Integrados (I^2C) [6], *Inter-integrated Circuit*. Se trata de un bus diseñado por Philips, usado principalmente para comunicar microcontroladores y sus periféricos en sistemas embebidos. El I^2C es un bus multi-maestro, es decir, permite la existencia de mas de un maestro, esta característica incluye la rotación del maestro por los esclavos. En el cuadricóptero que se esta analizando existe únicamente un maestro, la FC y cuatro esclavos, uno por cada BL.

El bus I^2C se caracteriza por la utilización de tres lineas, dos para transmitir la información: una para los datos (SDA) y otra para la señal de reloj (SCL) y una tercera para como referencia a tierra (GND). Si los circuitos que comunica el bus se sitúan en la misma placa la linea de GND no es necesaria, en el caso del MK-XL y como se trata de conectar circuitos integrados independientes se hace uso de ella.
Las lineas SDA y SCL se conectan a dos resistencias Pull-Up encargadas de mantener el valor lógico alto (1) a no ser que un dispositivo lo ponga a valor bajo (0). Esta resistencia es necesaria ya que factores como el ruido eléctrico o variaciones en la fuente de alimentación pueden producir cambios en los estados (alto y bajo) y en determinados momentos el sistema no sería capaz de distinguir en que estado se encuentra, de esta forma se producirían errores al enviar los datos provocando situaciones criticas incontrolables de ahí la necesidad de integrar resistencias que eviten dichos errores.

Este bus posee siempre un valor fuerte y otro débil, para la placa FC V2.5 el valor fuerte corresponde al 0 y el valor débil al 1. Esto es así porque el valor fuerte es aquel que se consigue forzando la linea, por el contrario el valor débil es el valor por defecto en la línea, en este caso la resistencia *pull-up* hace que sea 1, este será por lo tanto el valor de reposo del bus. Debido a esta configuración el valor en la línea de SDA solo varía cuando el valor de la línea SCL esta a 0, además por cada flanco de SCL solo se captura un bit de SDA, es decir, el flanco de subida de la línea SCL marca un valor valido en SDA y en flanco de bajada indica que el dato en SDA cambia.

La comunicación es de tipo *half-duplex*, es decir, se trata de una comunicación bidireccional que se realiza por la línea SDA, como solo existe una linea esta comunicación no se realiza de manera simultanea y necesita por lo tanto un control de acceso al bus.
El maestro será el encargado de controlar este acceso mediante el uso de la línea SCL, iniciando, cuando considere oportuno, el proceso de transmisión de información. El esclavo por su parte deberá esperar a que el maestro le solicite los datos para poder ocupar el bus y enviárselos.

El esquema básico de comunicación del I^2C se basa en:

- Condición de comienzo (*start*) ⇒ Maestro.
- Envío de 8 bits de dirección del esclavo con el que se realiza la comunicación ⇒ Maestro.
- Envío del bit de lectura(R) o escritura(W), 0 indica leer y 1 Escribir ⇒ Maestro.

- Envío del bit de aceptación (ACK) ⇒ Esclavo.
- Envío del byte de dirección de memoria al que se desea acceder ⇒ Maestro.
- Envío del bit de aceptación (ACK) ⇒ Esclavo.
- Envío del byte de datos escritura ⇒ Maestro o lectura ⇒ Esclavo.
- Envío del bit de aceptación escritura ⇒ Esclavo o lectura ⇒ Maestro.
- Condición de finalización (*stop*) ⇒ Maestro.

Capítulo 5

Modelo UML

El lenguaje unificado de modelado (UML) es un lenguaje estándar de modelado visual, definido por OMG[1], que se usa generalmente para modelar software, en general, su uso abarca la visualización, especificación, construcción y documentación de los diferentes elementos que componen un sistema.
Un modelo es la representación simplificada bien definida[2] de la realidad, se modela con el objetivo de entender de forma mas clara el sistema a desarrollar o en el caso de este trabajo el funcionamiento de un sistema ya desarrollado.

El uso de UML esta extendido debido a dos factores fundamentales, su capacidad de abstracción, es decir, la simplificación del sistema real incluyendo solo los detalles mas relevantes para alcanzar su objetivo y su capacidad de legibilidad, es decir, se trata de un lenguaje fácil y rápido de interpretar. Sin embargo y a pesar de que UML es un lenguaje de propósito general existen casos en los que es necesario usar algún lenguaje más especifico a la hora de modelar y representar los sistemas.

OMG define dos soluciones ante la necesidad de aplicar lenguajes mas específicos, la primera solución es la definición de un nuevo lenguaje alternativo a UML y la segunda solución es la extensión del propio UML mediante la creación de perfiles (*UML Profiles*) que, respetando la semántica original, definen nuevos conceptos y restringen alguno de los existentes.

En este trabajo, como ya se había adelantado en la introducción, se genera el modelo a partir del perfil especifico UML/MARTE [7]. La generación de este modelo se realiza con la ayuda del entorno `PAPYRUS` [8].

5.1. UML/Marte

El perfil UML/MARTE esta diseñado para modelar y analizar sistemas de tiempo real y embebidos, entre sus principales características destacan:

- Soporte de propiedades no funcionales.
- Añade modelos de tiempo y recursos al UML convencional.
- Definición de conceptos tanto para plataformas software como hardware.

[1] Object Management Group, consorcio dedicado al cuidado y establecimiento de diversos estándares.
[2] Un lenguaje bien definido se especifica mediante una semántica y sintaxis precisa capaz de ser interpretada por un ordenador.

- Soporte al análisis cuantitativo (rendimiento, etc)
- Remplaza el perfil UML/SPT, *Scheduling, Performance and Time*, haciendo de este uno mas completo.

Cabe destacar que entre los colaboradores académicos para la creación de este perfil se encuentra la Universidad de Cantabria.

Tal y como se indica en la sección 2 uno de los objetivos es desarrollar el modulo en base al perfil PHARAON, un perfil extendido al UML/MARTE, desarrollado por investigadores de la UC, que ofrece características extra para la especificación de modelos de sistemas embebidos y de tiempo real.

5.2. Metodología de especificación

Concretamente el perfil PHARAON [9] forma parte de un proyecto, el cual recibe el mismo nombre, que se creo a a partir de las necesidades surgidas a la hora de especificar sistemas de forma diferente a como hasta ese momento se habían especificado.
El sistema de especificación actúa como entrada al proyecto PHARAON, el resultado de este proyecto abarca un modelo creado que permite al diseñador la descripción de todas las características del sistema para obtener un diseño óptimo de la plataforma física. Este modelo se crea gracias a la combinación tanto de la experiencia obtenida durante el desarrollo de otros proyectos como las tareas de investigación (llevadas a cabo por sus desarrolladores) para permitir una especificación mas real de los sistemas, cabe destacar la inclusión del proyecto COMPLEX con los requisitos específicos establecidos para las tareas de síntesis, paralelización y tiempo de ejecución.

La complejidad de los sistemas embebidos y la cantidad de plataformas disponibles hace necesario el uso de metodologías de desarrollo que permitan a los equipos de diseño trabajar de una manera eficiente, por ello es muy importante desarrollar de manera independiente cada una de las partes del sistema (diseño de aplicaciones, diseño *hardware(HW)* y *software (SW)* de la plataforma, etc.).

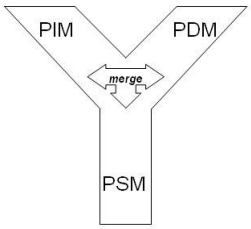


Figura 5.1: Metodología: Estructura "Y"

Por lo tanto, y debido a las necesidades surgidas, PHARAON implementa herramientas para llevar a cabo la especificación de un modelo basado en la metodología **UML/MARTE**. Esta metodología se basa en la creación de tres modelos siguiendo una estructura de tipo "Y", ver figura 5.1 que permite a los desarrolladores centrarse en un dominio concreto (HW, SW, aplicaciones, etc.) y, aun así, garantizar la consistencia del sistema debido al uso del mismo lenguaje de especificación.

La estructura Y se caracteriza por poseer tres modelos independientes capaces de conectarse para crear un sistema común. Las dos ramas de la "Y."ªbarcan dos modelos separados, por un lado

el *software* de aplicación (PIM) y por el otro la plataforma (PDM). Estos dos modelos se conectan a través de la tercera rama (PSM) que define el mapeo del *software* dentro del *hardware*.

Modelos de la estructura "Y":

- **PIM** (*Plataform Independent Model*), el modelo independiente de la plataforma describe los aspectos funcionales y no funcionales del comportamiento de la aplicación (e.g. modelo funcional). Al tratarse de un modelo independiente puede ser reutilizado en la implementación de la aplicación en otras plataformas.
- **PDM** (*Plataform Description Model*), el modelo de descripción de la plataforma describe los diferentes recursos SW y HW que componen la plataforma del sistema, se trata de los elementos de aplicación independientes.
- **PSM** (*Plataform Specific Model*, el modelo especifico de la plataforma describe la arquitectura del sistema y la distribución de las fuentes de la plataforma. Este modelo sitúa las funcionalidades del sistema en los recursos de la plataforma en los que hayan sido implementados.

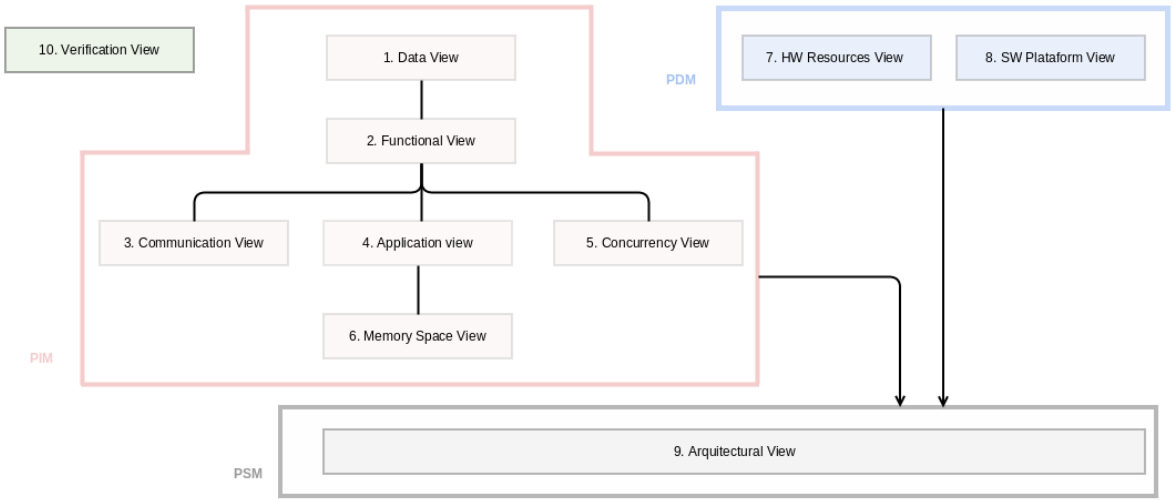


Figura 5.2: Metodología: Jerarquía de Vistas

Cada uno de los tres modelos explicados anteriormente esta construido a partir de una colección de "elementos de modelo" (e.g. tipos de datos, componentes *software*, interfaces, etc.). Existen un gran numero de elementos del modelo (primitivas UML/MARTE-PHARAON) con diferentes propósitos. Para mantener la legibilidad del lenguaje UML estos elementos se definen organizados en vistas, cada vista define por lo tanto un aspecto especifico del modelo, ver figura 5.2, a continuación se describen las principales funcionalidades de cada una de ellas:

- Vista de datos (1), define los tipos de datos usados en el intercambio de información. Estos datos pueden ser de dos tipos, primitivos (entero, decimal, etc.) o no, es decir, se pueden definir tipos de datos específicos para la aplicación (estructuras, array de tipos primitivos, etc.).

- Vista funcional (2), esta vista esta compuesta por las interfaces que aportan y requieren los diferentes componentes de la aplicación para mantener la comunicación. Al tratarse de la vista funcional también contiene los archivos que definen la funcionalidad de cada componente, por ejemplo, si tenemos un componente "suma"que realiza la suma de dos elementos, en esta vista se definirá el archivo que contiene la función suma.

- Vista de comunicación (3), esta vista incluye lo canales de comunicación, es decir los medios de comunicación utilizados por interfaces para conectar los diferentes componentes. En caso de ser necesaria la definición de mecanismos para la sincronización de hilos y de procesos, estos se definen también en esta vista. Esta vista es opcional si la comunicación no es considerada.

- Vista de aplicación (4), en esta vista se definen los componentes de aplicación y su estructura, dicha estructura se corresponde con la vista general del sistema que incluye las conexiones de los componentes usando las interfaces y los mecanismos de comunicación definidos en las anteriores vistas. También puede incluir la relación que existe entre los componentes y los archivos de funcionalidad definidos en la segunda vista.

- Vista de concurrencia (5), esta vista incluye todos los hilos (*threads*) y su correspondiente mapeo entre los componentes. Se trata de una vista opcional ya que no es necesaria su definición en el caso en el que sistema solo necesite un único hilo para ejecutarse.

- Vista del espacio de memoria (6), como su nombre indica define las diferentes particiones de memoria donde se alojan los procesos del sistema.

- Vista de los recursos *hardware* (7), describe los procesadores disponibles en la plataforma, los buses y el resto de recursos HW.

- Vista de la plataforma *software* (8), describe los recursos SW, principalmente el sistema operativo sobre el que se desplegará el sistema.

- Vista arquitectural (9), se trata de la vista encargada de mostrar la relación entre el resto de vistas explicadas, por lo tanto define la arquitectura de la plataforma y el mapeo del sistema y los procesos en los recursos disponibles. Además puede incluir la relación entre los hilos y los procesadores, si la hubiese. Debido a la importancia de conocer la relación entre los recursos y el sistema la definición de esta vista es obligatoria.

- Vista de verificación (10), en esta vista se definen los componentes de entorno que interactúan con el sistema.

La creación de vistas junto con los componentes permiten modelar el sistema de forma aún mas dinámica, es decir, cada modulo puede ser sustituido en cualquier momento por otro que aunque no se comporte de la misma manera es capaz de interpretar las entradas y/o producir salidas equivalentes al modulo anterior. Esto hace que una vez modulado el sistema se puedan probar diferentes configuraciones tanto *hardware* como *software* destinadas a la mejora del rendimiento del sistema.

Este trabajo se basa en la utilización del perfil adaptado de UML/MARTE para realizar el modelo PIM del cuadricóptero anterior, a partir del modelo UML se generaran los archivos XML[3]

[3]eXtensible Markup Language, es un estándar usado para almacenar datos de forma legible. Se usa para intercambiar información estructurada entre diferentes plataformas.

necesarios para la creación del binario y las librerías de la aplicación. Estos archivos se obtienen a través de la herramienta eSSYN (*embedded software synthesis*) desarrollada por el mismo departamento de la UC.

eSSYN es una herramienta de síntesis de software para sistemas embebidos. Con la combinación de esta herramienta y el modelo base tanto de la aplicación como de la plataforma se pueden generar un conjunto completo de binarios enfocados a aplicaciones embebidas. La ventaja de esta herramienta es la reducción de tiempo a la hora de desarrollar aplicaciones.

Los archivos XML correspondientes a cada uno de los componentes software de la aplicación y sus correspondientes archivos de funcionalidad, desarrollados en C, junto con la descripción de la plataforma HW disponible (numero y tipo de procesadores, sistema operativo) y la relación entre ambos son las entradas que requiere el software eSSYN para generar el código y las llamadas al sistema necesarias para realizar las comunicaciones entre los componentes, esta herramienta además genera los *makefiles*[4] necesarios para la compilación. A partir de estos *makefiles* y tras su compilación (desde el entorno eSSYN) se obtienen todos los archivos ejecutables necesarios para cargar al *hardware* especifico (procesadores, sistema operativo, etc.).

5.3. Descripción del modelo del cuadricóptero

En la siguiente sección se describe el modelo llevado a cabo del cuadricóptero descrito en el capítulo 4, objetivo del presente trabajo, aplicando la metodología explicada en el apartado anterior. El modelo esta compuesto por dos estructuras, el sistema y el entorno, ver figura 5.3. La simulación del sistema se realiza en un computador de forma nativa bajo la distribución Ubuntu 14.04, dentro de este estará alojado tanto el sistema como una parte del entorno de la aplicación, la parte restante corresponde al radio control, el cual se comunica con la aplicación a través de los puertos usb.

Tal y como se puede ver ambas estructuras están formadas por componentes electrónicos e informáticos del cuadricóptero. En el entorno se definen aquellos elementos que interactúan con el usuario y el sistema abarca los elementos que como su propio nombre indica dependen del entorno al que nos queramos referir. El sistema, que forma parte del entorno, esta formado por los componentes de aplicación del subsistema, es decir, aquellos que se comunican dentro del cuadricóptero y con los que el usuario no tiene ningún contacto, se trata por lo tanto de los elementos principales que definen la funcionalidad del sistema embebido.

Una vez estudiados y definidos los componentes de alto nivel que forman parte de la aplicación, se procede a la creación del repositorio de componentes con sus respectivas características.

5.3.1. Sistema de la aplicación (`Quadcopter`)

El sistema de aplicación es el pilar fundamental de este trabajo ya que engloba los elementos internos del cuadricóptero real, FC y BL. En los siguientes apartados se explica de forma detallada los componentes que forman parte del sistema modelado.

[4]Ficheros de texto necesarios para llevar a cabo la gestión de la compilación de los programas, su principal objetivo es informar de las dependencias entre los diferentes componentes de un proyecto.

Figura 5.3: Representación Entorno-Sistema

5.3.1.1. Decodificador PPM (`PPM`)

El decodificador PPM es el encargado de interpretar (decodificar) la señal recibida del receptor del radio control y enviar los datos que obtenga al siguiente componente, el controlador de vuelo. En el sistema real ambos componentes se encuentran en la misma estructura física, por lo que la comunicación es directa.

5.3.1.2. Controlador de vuelo (`Flight Controller, FC`)

En el sistema real los datos recibidos del radio control a través de la señal PPM son analizados y procesados dentro del software de aplicación propio del cuadricóptero, dado que no se trata de un *software open source*[5] y además no se dispone de ningún tipo de información acerca de el mismo, este componente se encarga de definir una aproximación a dicha funcionalidad.

Por lo tanto, hablamos del elemento de la FC encargado de recibir los datos, procesarlos y enviarlos a los controladores de los motores, a través del bus I^2C. El objetivo de estos datos es trasmitir a cada controlador la velocidad a la que deben operar los rotores de cada motor.
Tal y como se explico previamente el bus I^2C, en el cuadricóptero, únicamente realiza la comunicación entre el procesador de datos, los controladores y viceversa.

[5]Denominado *Software.* de código abierto, es decir, su código de funcionalidad esta disponible para el usuario

5.3.1.3. Controlador de los motores (`Rotors Controller, BL`)

El la arquitectura del Cuadricóptero se ha visto como este esta compuesto por cuatro controladores cada uno de ellos encargado, principalmente, de transmitir la corriente eléctrica necesaria a cada motor. Cada controlador recibe del bus I^2C la velocidad a la que debe circular su rotor que transforma y envía en la cantidad de corriente eléctrica que es necesaria para producir esa velocidad.

En el modelo y dado que el objetivo es la simulación por pantalla del sistema real, es decir no existen motores a los que proveer de corriente, se agrupan los cuatro controladores en uno único cuya finalidad consiste en calcular la fuerza resultante que produce cada motor en función de la velocidad real a la que circularían en el cuadricóptero, después se encarga de enviar esta información al modelo físico. Esta fuerza se calcula a partir de las especificaciones de los motores tal y como se verá en la sección 8.1.3.

5.3.1.4. Giroscopio (`Gyroscope`)

La funcionalidad de este componente es mantener la estabilidad del cuadricóptero. Para poder llevar a cabo esta función el giroscopio se encarga de recibir datos (posición, ángulo de giro, velocidad del viento, altura, etc.), procesarlos y obtener los valores de corrección aplicables a cada una de las corrientes que controlan los motores.

Este trabajo se lleva a cabo en la placa principal del aparato tratándose así de un proceso de realimentación continuo, es decir, en función del estado inicial calcula las constantes de corrección a aplicar en el siguiente estado para alcanzar el estado ideal esperado.

Durante la primera fase del modelado del aparato no se incluyó el desarrollo de este modulo, es decir, fue considerado como una caja negra a tratar en proyectos futuros, pero durante una de las pruebas de simulación se observo como el aparato al realizar cualquier tipo de maniobras no alcanzaba la estabilidad en ningún caso, por lo que se procedió a su inclusión modificando el modelo inicial.

Al no tratarse como un objetivo del proyecto inicial la funcionalidad que realiza no representa la realidad, se trata únicamente de obtener los valores de corrección necesarios, minimizando al máximo el error producido durante las maniobras de giro.
Su funcionalidad se basa en la realización de tres pasos:

- **Recoger datos**, se encarga de recibir los datos generados por el modelo físico referentes a los ángulos x, y y z.
- **Generar valores de corrección**, genera los coeficientes, perdida o ganancia, necesarios para aplicar a la intensidad de los motores en el siguiente estado. El objetivo de estos coeficientes es por un lado lograr que el ángulo del aparato con respecto a los ejes x e y sea 0, y por otro lado reducir la velocidad de giro sobre el eje z (sobre si mismo), siempre teniendo en cuenta que la suma de las fuerzas sea superior a su peso, impidiendo así que durante una maniobra de giro el cuadricóptero se caiga.
- **Trasmitir datos**, se encarga de transmitir los coeficientes calculados al controlador de los motores para que este calcule la nueva intensidad aplicable a cada uno de ellos.

5.3.2. Entorno de la aplicación (`Environment Structure`)

Hasta ahora se han visto los componentes que forman parte del sistema de la aplicación. El entorno de aplicación esta compuesto por los componentes de entrada, procesamiento y salida del

sistemas, es decir, el entorno de la aplicación se refiere al conjunto de todos los elementos que conforman el modelo físico y lógico del cuadricóptero.

5.3.2.1. Radio Control (`Radio Control`)

El radio control es el componente principal que interactúa con el sistema, a través de los movimientos de las palancas de control (*stick*) de la radio el usuario mueve el cuadricóptero. Este componente se refiere al modelado del transmisor explicado en el apartado 4.2.2, pero a diferencia del modelo real, la simulación se realiza mediante el uso de los *joystick* de un *gamepad*[6].

Se estudio la posibilidad de realizar el modelo usando la radio original del cuadricóptero y se llego a la conclusión de que su realización incluía el diseño *hardware y software* especifico para interpretar la señal que el transmisor enviaría al ordenador.

Hoy en día, unos cuantos meses después de la investigación, existe una librería diseñada para *hardware* Arduino capaz de interpretar las señales de los receptores de este tipo de radios.
Debido al desconocimiento en aquel momento del trabajo que podía generar la implementación de este sistema y teniendo en cuenta que el objetivo es el modelado del sistema y no del entorno se opto por investigar otras soluciones mas sencillas que permitieran alcanzar el objetivo, llegando entonces al uso del *gamepad*.

La principal funcionalidad de este componente es la inserción de datos en el sistema, esta se realiza a través de dos funcionalidades:

- **Controlador *gamepad***, se trata del *software* encargado de recibir los movimientos que se producen en los *Joysticks* (palancas del mando) a través de los puertos usb del computador. Además interpreta dichos movimientos para que posteriormente el codificador pueda realizar su función.

  Cabe destacar que el computador dispone tanto de puertos usb 2.0 como usb 3.0 en los que la comunicación con el mando se realiza de forma diferente.

  Durante el desarrollo de la funcionalidad de este modulo se tienen en cuenta ambas variantes. Es tarea del usuario indicarle a la aplicación en el momento de su ejecución a que puerto ha conectado el mando. Además se han comprobado los posibles efectos en la conexión de este en diferentes computadores donde se ha visto que los datos de entrada que se obtienen directamente del dispositivo (/dev/) no siempre se mapean en el mismo archivo, de igual forma que en el caso del puerto, el usuario debe indicar cual se corresponde con su computador en concreto (e.g. js0).

- **Codificador PPM**, se trata del elemento encargado de mantener la comunicación con la placa principal del cuadricóptero (FC), esta comunicación se realiza mediante la modulación y codificación de la señal PPM, esta señal se obtiene a partir de los datos generados por el controlador *gamepad*.

En el diseño de este componente se han separado ambas funcionalidades, a pesar de formar parte de un componente tanto en el sistema real como en el modelado, con el objetivo de generar un modelo mas dinámico. La separación de componentes ya realiza esta función, tal y como se ha visto, cualquier componente se puede sustituir por otro capaz de interpretar las entradas y las salidas, pero a su vez la independencia de tareas dentro de cada componente permite la modificación unitaria de los respectivos códigos de funcionalidad.

[6]Mando de control usado generalmente en videojuegos.

5.3.2.2. Subsistema (`sUT Quadcopter`)

El subsistema es una instancia que encapsula los componentes del sistema explicados anteriormente, por lo tanto su datos de entrada se corresponden con la señal PPM, dichos datos los recibe a través de la conexión con el codificador PPM implementado en el componente `Radio Control` y produce como resultante las fuerzas que generan los motores, dichos valores son enviados al siguiente componente, el modelo físico.

El encapsula-miento de estos componentes se produce conectando los elementos a un componente superior, el subsistema (`sUT`), que no posee ninguna funcionalidad propia. Debido a este encapsula-miento tanto el PPM como el controlador de motores se conectan al componente exterior y este será por su parte el que se conecte tanto al radio control como al modelo físico, de esta forma se mantiene la abstracción del sistema y el entorno, tal y como se define en el *profile*.

5.3.2.3. Modelo físico (`Operations Model`)

El modelo físico por su parte se trata de la implementación de un modelo matemático lineal basado en los formalismos de Newton-Euler [10] que definen los movimientos y posiciones que realiza el cuadricóptero en función de la fuerza producida por cada motor, el detalle de este modelo se puede ver en la sección 6.3. Este componente es por lo tanto el encargado de enviar la posición y orientación (coordenadas) del sistema a la aplicación de visualización.

5.3.2.4. Sistema de visualización (`Visual tk`)

El ultimo componente del entorno es el sistema de visualización en 3D, aplicación encargada de mostrar los movimientos que realiza el cuadricóptero en un entorno simulado, ver detalle en la sección 7.

5.3.3. Vistas

Al comienzo de esta sección se explicó como realizar un modelo completo siguiendo la metodología *Y*. Para la realización del modelo del cuadricóptero se simplifica su uso y solo es necesario la realización de las vistas que forman parte del modelo PIM además de la vista de verificación.

La definición del modelo de forma simplificada se debe a la falta de necesidad de implementar el hardware, tampoco existe la necesidad de especificar el software para el que se desarrolla el trabajo ya que la compilación se realizará de forma nativa, y además la plataforma esta diseñada para la utilización de un único hilo (*thread*) de ejecución, por lo que el modelo PDM y PSM carecen de utilidad.

Hasta ahora se han descrito los componentes que forman parte de la **vista de aplicación** de los que únicamente se conoce su funcionalidad, pero hay que tener en cuenta que cada componente puede y debe ser configurado de forma independiente aplicándole las primitivas UML que le correspondan, estas primitivas describen las características de cada elemento.

A continuación se detallan cada una de las vistas generadas, sus componentes y la relación entre ellas.

5.3.3.1. Vista de datos (`Data View`)

Forman parte de esta vista, ver anexo A, los tipos de datos utilizados en las operaciones de las interfaces.

En primer lugar se definen los datos básicos, `PrimitiveType`, *double* e *int* correspondientes al uso de números en coma flotante y enteros respectivamente. A partir de estos se crean dos tipos de datos nuevos, *ArrayDouble* y *ArrayInt*, definidos mediante la primitiva `DataType` y correspondientes a estructuras *array* de ambos tipos. Finalmente con ayuda de la primitiva `collectionType` se definen los tipos de datos específicos a utilizar y su tamaño.

5.3.3.2. Vista de funcionalidad (`Functional View`)

Forman parte de esta vista, ver anexo B, los elementos que especifican las funcionalidades de los componentes, es decir, los ficheros de funcionalidad, también se incluyen en esta vista las interfaces a utilizar.

En primer lugar se definen los ficheros que contienen los códigos fuentes de la aplicación siguiendo la primitiva `File` de UML, a continuación se definen las interfaces.

Las interfaces abarcan las operaciones que se realizan entre los diferentes componentes de la aplicación, por lo tanto las operaciones definidas han de estar implementadas en los archivos de funcionalidad que se acaban de definir.

Tal y como se ve en el anexo B.2 todas las operaciones tienen parámetros, estos parámetros se definen utilizando los tipos de datos creados en la vista anterior, además la definición de los parámetros, nombre, lleva la estructura `orden:nombreArgumento`, especificando de esta forma el orden en el que se han de pasar los argumentos a la función.
Observando de nuevo el anexo se puede ver como las interfaces implementan el estereotipo `Cliente Server Specification`, esto es debido a la necesidad de mantener la sincronización de los elementos según la semántica cliente-servidor.

5.3.3.3. Vista de comunicación (`Communication View`)

Forman parte de esta vista, ver anexo C, los mecanismos utilizados para realizar la comunicación entre los componentes. Cabe destacar que se han definido siete canales, uno por cada comunicación. En la definición común de los canales únicamente se definiría un canal por cada tipo de comunicación, en este caso solo sería necesaria la definición de un único canal concurrente que se caracteriza por realizar la comunicación de forma unidireccional(*simplex*) de un elemento cada vez, además se trata de un canal que no es bloqueante.

Definir siete canales en lugar de uno permite a trabajos futuros integrar otras funcionalidades disponibles en el cuadricóptero, como por ejemplo la comunicación bidireccional entre el controlador de vuelo y los motores, modificando únicamente las características de los canales que se vean involucrados.

5.3.3.4. Vista de aplicación (`Application View`)

Forman parte de esta vista, ver anexo D, los componentes software que se utilizan para construir el sistema. El sistema esta compuesto por instancias de estos componentes conectadas entre si a través de los canales definidos en la vista anterior que realizan su comunicación mediante el uso de operaciones definidas en las interfaces (provistas/requeridas) e implementadas en los ficheros de funcionalidad.

En primer lugar se definen los componentes a modelar (ver sección 5.3.1). Dentro del modelo se diferencian tres tipos de componentes por los estereotipos que implementan; `System`, `RtUnit`, `TestComponent`. En el primero se engloban las instancias necesarias de los componentes `RtUnit` pertenecientes al sistema de aplicación, creando de esta forma el componente que servirá para crear la instancia del sistema en el entorno de aplicación. Por lo anterior se deduce que del segundo, `RtUnit`, forman parte todos los componentes que se han instanciado incluyendo además los componentes del entorno (ver sección 5.3.2) que también implementan `TestComponent` indicando de esta forma que se trata de elementos del entorno y no del sistema.

En esta vista además se definen las relaciones entre los ficheros y los componentes, ver anexo D.3, además se definen los *paths*, directorios, donde se almacena cada uno de los componentes.

5.3.3.5. Vista del espacio de memoria (`Memory Space View`)

Forman parte de esta vista, ver anexo E, la definición de los espacios de memoria disponibles en la aplicación, para este caso concreto únicamente existe un único espacio de memoria donde se alojan cada uno de los componentes de la aplicación.

5.3.3.6. Vista de verificación (`Verification View`)

Forman parte de esta vista, ver anexo F, la definición de los escenarios de la aplicación, es decir, esta vista incluye las instancias de los elementos definidos en la vista de aplicación que componen el sistema de entorno tales como el radio control, el sistema, el modelo físico y el sistema de visualización.
La relación entre estos elementos se define a través de la implementación de puertos asignándoles a estos los canales e interfaces que les correspondan tal y como se hizo cuando se creo que el componente del sistema (`Quadcopter`).

5.3.4. Limitaciones

Cabe destacar que durante el desarrollo del modelo UML/MARTE se han encontrado dos limitaciones, la primera se refiere a la carencia por parte de la metodología para representar "tiempos de ejecución" y la segunda se corresponde con el soporte a lenguajes de programación.

Inicialmente se realizó el desarrollo de todas las funcionalidades en `C++` pero durante el proceso de compilación del código se descubrió la incompatibilidad de este lenguaje con la herramienta de generación de código por lo que se realizo el migrado a `C`, lenguaje soportado por las herramientas.

Capítulo 6

Modelo Físico

Se define como modelo físico a la descripción matemática de un sistema real, en este caso el cuadricóptero. El modelo físico abarca el proceso desde el que se recibe como entrada la fuerza de cada motor y se produce como salida la posición y el ángulo de giro del aparato.

La implementación de este modelo se lleva a cabo para verificar que se esta realizando un modelo del cuadricóptero correcto, donde aplicándole un numero de entradas controladas el sistema reaccione tal y como se espera.
Sin este modelo únicamente se podría comprobar si las intensidades de cada motor tienden a funcionar como se espera pero no se podría verificar si el sistema es estable o no ya que no se conocerían los movimientos que realiza.

Este capítulo se dividirá en varios apartados en los que se irán introduciendo los conceptos necesarios para, finalmente, entender y definir el modelo.

6.1. Conceptos básicos

En este apartado se definen los conceptos básicos necesarios para entender los parámetros que intervienen en el desarrollo del modelo.

En la figura 6.1 se muestra la relación que existe entre las variables de entrada sombreadas en verde y las de salida sombreadas en rojo.

6.1.1. Variables de entrada

Se trata de las variables dependientes del entorno físico y electrónico del sistema real.

- Las **variables físicas** son variables fijas, es decir, su valor viene dado por la propia física del aparato y del entorno.
    - Longitud (**d**): distancia en metros de los motores al centro del cuadricóptero.
    - Peso (**m**): peso en kilogramos del cuadricóptero.
    - Fuerza de la gravedad (**g**): correspondiente a la fuerza que la gravedad que se ejerce sobre el eje principal del cuadricóptero. Valor constante medio de 9.8 m/s2.
- Las **variables electrónicas** son aquellas en las que su valor se ve afectado por la lógica del sistema del aparato, por lo tanto son variantes en el tiempo.

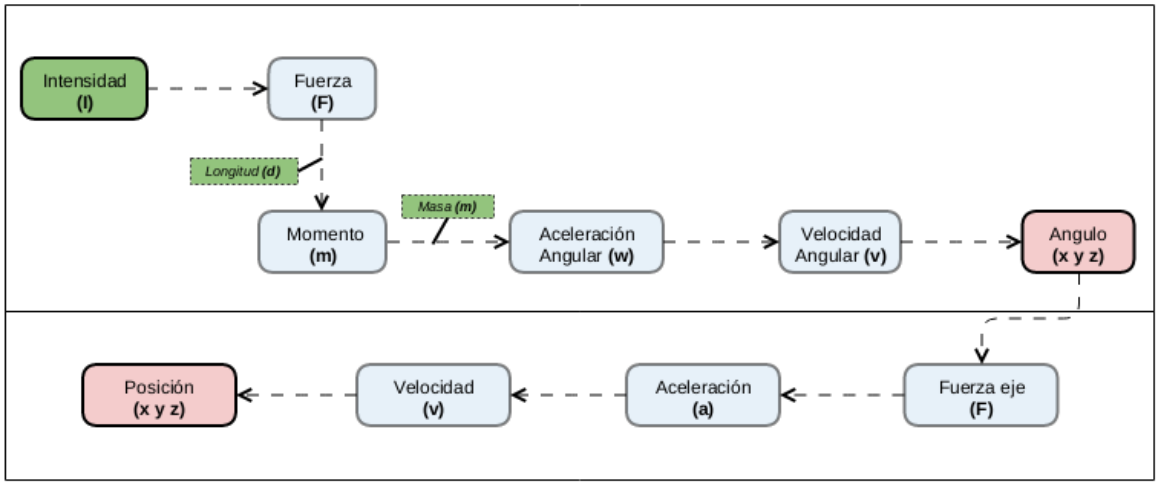


Figura 6.1: Esquema Variables

Cada una de las cuatro intensidades de los rotores (**I**) forman parte de las variables electrónicas. La intensidad se corresponde a la cantidad de corriente que recibe cada uno de los 4 motores haciendo posible que el cuadricóptero se mueva.
Si bien es cierto que el modelo físico no recibe como parámetro de entrada las intensidades, sino que recibe la fuerza que cada motor realiza en función de su intensidad. La intensidad ha sido incluida en el esquema para darle una mayor claridad.

6.1.2. Variables de salida

- Ángulos (Φ, Θ, ψ): el conjunto de los tres ángulos describe la rotación del aparato en el espacio.
- Posición (**X, Y, Z**): define la posición en la que se encuentra el cuadricóptero en el espacio.

6.1.3. Variables de proceso

- Fuerza (**F**): valor de influencia que como consecuencia produce cambios en el estado del aparato.
- Momento (**M,** τ): valor que representa la cantidad de movimiento que se realiza sobre el cuadricóptero.
- Aceleración Angular (α): valor que representa el cambio que se realiza en la velocidad angular.
- Velocidad Angular (ω): valor que representa la velocidad de giro a la que se desplaza cada uno de los ángulos.
- Fuerza ejes (**Fe**): valor de influencia por eje (x, y, z) que como consecuencia produce cambios en la posición del cuadricóptero.
- Aceleración (**a**): valor que representa la aceleración que se produce en el aparato.
- Velocidad (v): valor que representa la velocidad a la que circula el cuadricóptero.

6.2. Características del Cuadricóptero

El cuadricóptero tiene 4 motores con sus correspondientes hélices los cuales están numerados del 1 al 4 en sentido horario comenzando por el motor principal, motor 1, como si del morro de un avión se tratase, ver figura 6.2.

La principal característica de los motores es que estos giran dos a dos en sentido horario y anti-horario, es decir, el motor 1 y 3 que se encuentran uno en frente del otro giran en sentido horario y los motores 2 y 4 en sentido contrario. Esto no implica que los motores pares e impares compartan la misma velocidad, es decir, cada motor funciona de forma independiente permitiendo así un mayor control de movimiento y estabilidad sobre el cuadricóptero, en cambio, esta misma independencia es la que hace que la propia lógica del sistema, el control de vuelo, resulte compleja de desarrollar.

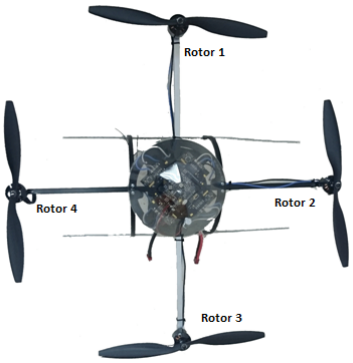


Figura 6.2: Cuadricóptero

Cada uno de los motores se encuentra situado a 25 cm del centro del aparato, además por cada uno de ellos atraviesa su correspondiente intensidad eléctrica (I), que varía en el tiempo, encargada de producir una fuerza vertical (F) en dirección hacia arriba, a partir de esta fuerza y como veremos en la sección 6.3.1 se obtiene el momento angular (M) de cada motor, es decir, la cantidad de movimiento[1] que la fuerza está produciendo sobre el centro del aparato, ver figura 6.3.

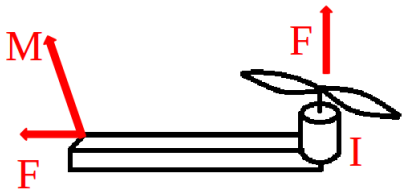


Figura 6.3: Efecto Físico Motor-Hélice

Como ya se ha visto un cuadricóptero se mueve sobre seis grados de libertad, tres de traslación y tres de rotación. En esta sección se explicara su origen.

[1] Describe la cantidad de movimiento.

Los tres grados de **traslación** definen la posición del cuadricóptero, es decir, las coordenadas **X Y Z** en el espacio.

Los tres grados de **rotación** se definen como ***roll*** (Φ), ***pitch*** (Θ) y ***yaw*** (Ψ) ó por su traducción como guiñada, cabeceo y balanceo respectivamente, de ahora en adelante se usarán los términos en ingles comúnmente conocidos. Estos tres grados son conocidos como ángulos de navegación de Euler encargados de definir la orientación de un objeto en tres dimensiones, es decir, se encargan de definir los posibles movimientos angulares del cuadricóptero en función de la velocidad de cada motor, ver figura 6.4.

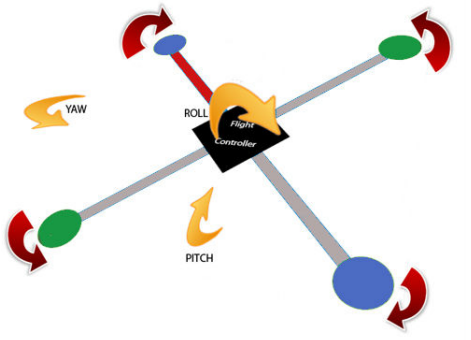


Figura 6.4: Movimientos del Cuadricóptero

En los siguientes tres apartados se definen cada uno de los movimientos angulares [11] que se producen en el cuadricóptero.

Además hay que tener en cuenta que cuando la intensidad de los 4 motores genere una fuerza conjunta superior al peso del cuadricóptero éste se elevará, ver Figura 6.5.

Figura 6.5: Movimiento de Elevación

6.2.1. Movimiento Roll

El movimiento de *roll* o guiñada, que afecta al eje vertical, define si el cuadricóptero se desplazará hacia la derecha o hacia la izquierda, ver figura 6.6. Este movimiento viene dado por la diferencia de intensidad aplicada en los motores 2 y 4.

El cuadricóptero se desplazará en la dirección del motor que menor intensidad este aplicando sobre el centro del mismo, es decir, se desplazará a la derecha en el caso de que la intensidad del rotor 4 sea superior a la del rotor 2 y se desplazará hacia la izquierda en caso contrario. Este

desplazamiento se produce en función de la fuerza ejercida sobre los motores tal y como se explica al comienzo de esta sección.

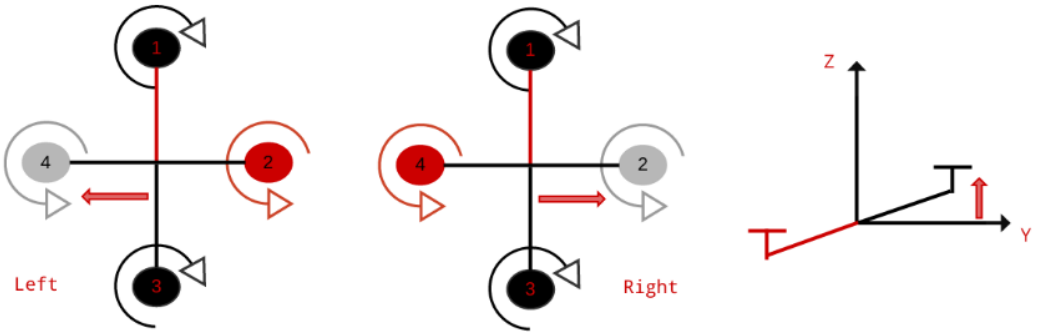


Figura 6.6: Movimiento Angular Roll

6.2.2. Movimiento Pitch

El movimiento *pitch* o cabeceo, que afecta al eje longitudinal, define si el cuadricóptero se desplazará hacia delante o hacia atrás, ver figura 6.7. Este movimiento viene dado por la diferencia de intensidad aplicada en los motores 1 y 3.

Como en el caso del movimiento *roll* el cuadricóptero se desplazará en la dirección del motor que menor intensidad este aplicando sobre el centro del mismo, es decir, se desplazará hacia delante en el caso de que la intensidad del motor 3 sea superior a la del motor 1 y se desplazará hacia atrás en caso contrario.

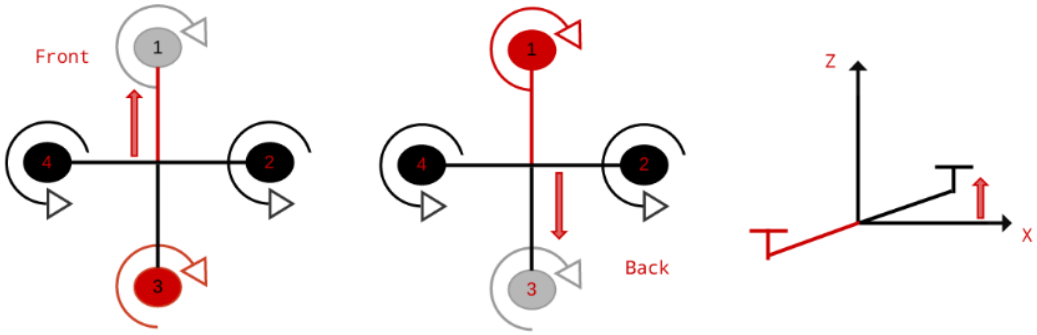


Figura 6.7: Movimiento Angular Pitch

6.2.3. Movimiento Yaw

El movimiento *yaw* o balanceo, que afecta al eje lateral, define el giro del cuadricóptero sobre su propio eje, ver figura 6.8. Este movimiento se produce cuando los motores trabajan 2 a 2, es decir, el par del motor 1 y 3 funcionan a la misma intensidad y en ese mismo instante el otro par, motores 2 y 4, funcionan a otra intensidad diferente al primer par o incluso este par podría haberse

quedado inoperativo, es decir, no se encuentran funcionando ninguno de los dos motores.

El cuadricóptero se desplazará en el sentido de las agujas del reloj cuando los motores 2 y 4 trabajen a más intensidad que los motores 1 y 3, en el caso contrario girará en sentido anti-horario.

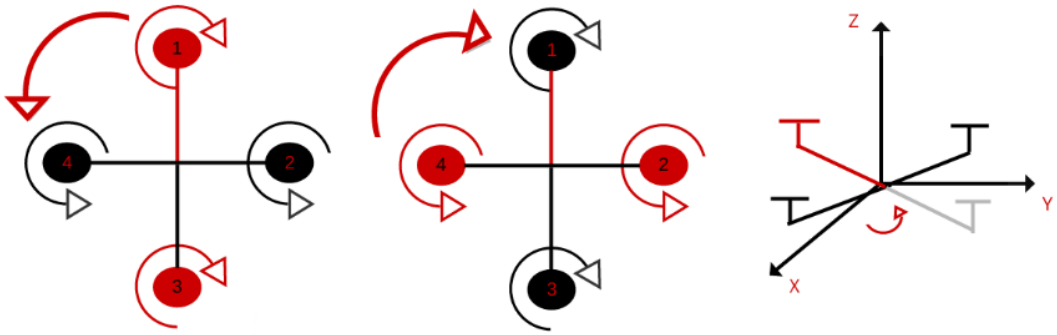


Figura 6.8: Movimiento Angular Yaw

6.3. Modelo lineal

Las ecuaciones que se definen para el control de un cuadricóptero, normalmente, se tratan de ecuaciones diferenciales, esto es debido al proceso de formulación del propio modelo físico.
En el primer paso del proceso de formulación se deben identificar las variables que causan cambios en el sistema para posteriormente establecer el conjunto de hipótesis que definen los cambios en el propio sistema. En la propia definición de las hipótesis intervienen derivadas, es decir, ecuaciones diferenciales.

En cambió, para la realización de este proyecto se definirá un modelo lineal, es decir, se formularán las hipótesis de forma que, para una variable X se explique su comportamiento mediante su relación lineal con los valores de aquellas variables que puedan influirla.
Este modelo permitirá de forma sencilla observar cómo evoluciona el sistema en un intervalo de tiempo delimitado.

6.3.1. Momento angular (τ)

Para obtener los 6 grados de libertad explicados en la sección 6.2, lo primero que se definen son las ecuaciones que relacionan las variables de entrada y de salida.

En la primera relación se obtienen los **momentos angulares** (τ) [12] [13], ver ecuaciones 6.1, 6.2, 6.3 que calculan la cantidad de movimiento de cada ángulo definido anteriormente.

Los momentos se definen a partir de la fuerza ejercida por cada motor sobre el eje al que se refieren y la distancia de esa fuerza sobre el centro del cuadricóptero, d (valor fijo perteneciente a las entradas).

$$Momento\_Roll(\tau_\phi) = \tau_2 - \tau_4 = (F_2 - F_4) * d \tag{6.1}$$

$$Momento\_Pitch(\tau_\theta) = \tau_1 - \tau_3 = (F_1 - F_3) * d \tag{6.2}$$

$$Momento\_Yaw(\tau_\psi) = (\tau_1 + \tau_3) - (\tau_2 + \tau_4) = ((F_1 + F_3) - (F2 + F4)) * d \tag{6.3}$$

Como se introducía en la sección 6.2 se disponen como entradas las 4 intensidades que producen el giro de cada uno de los motores además en el párrafo anterior se especificaba la dependencia de los momentos respecto de la fuerza, por ello el siguiente paso es especificar cómo se calcula dicha fuerza.

La fuerza **(F)** de cada motor, ver ecuación 6.4, se obtiene a partir de la multiplicación de un valor constante (K) con la diferencia de la intensidad eléctrica que se produce cuando éste se encuentra en reposo (I_0) y la intensidad en el momento actual (I).

El valor de I es una variable de entrada dependiente del momento temporal en el que se encuentre. Además tal y como se ha mostrado anteriormente a la intensidad en un momento t hay que aplicarla el coeficiente de corrección que se calculo en t_0 en el giroscopio.

Los valores I_0 y K son constantes, es decir, no cambian en el tiempo. El valor I_0 es igual a la corriente eléctrica que circula por el motor en el instante 0 y la constante K se obtiene a partir de las características reales del motor físico que relacionan el empuje $(thurst)$ que este produce con respecto a la intensidad (I) que recibe. Esta ultima relación se explica detalladamente mas adelante, ver sección 8.1.2.

$$Fuerza_{1 \leq i \leq 4} = F_i = K(I_i - I_0) \tag{6.4}$$

Finalmente se tiene en cuenta que todos los motores en su conjunto afectan a la fuerza de empuje total (μ) producida sobre el cuadricóptero, ver ecuación 6.5, esta fuerza genera un vector que se aplica sobre el centro del cuadricóptero.

En situación de reposo el vector fuerza se encuentra apuntando en dirección hacia arriba, por lo que afecta al eje Z. Cuando la intensidad de eléctrica de un motor aumenta dicho vector fuerza se inclina, lo cual introduce una componente horizontal en la fuerza, permitiendo así la realización de giros.

$$EmpujeTotal = \mu = F_1 + F_2 + F_3 + F_4 \tag{6.5}$$

A continuación se explica de forma más detallada de donde se obtienen las ecuaciones de fuerza, momentos y empuje.

Se dispone de cuatro motores con su correspondiente corriente de giro, **I**, estos motores se encuentran situados en la estructura del cuadricóptero de forma simétrica, es decir, cada uno está situado a una distancia, **d** valor constante, del centro del mismo.
Con estos datos, tal y como se ha visto anteriormente, se deducen la Fuerza y los momentos de cada uno de ellos.

Una vez que se dispone de los momentos generados por cada uno de los motores se debe de tener en cuenta el efecto que produce cada uno de ellos sobre los ejes $X\ Y\ Z$ con el fin de obtener los momentos angulares *roll* (ϕ), *pitch* (θ) *y yaw* (ψ).
Esta relación se puede observar en las Figuras 6.6, 6.7, 6.8 respectivamente, donde se ve:

- Los motores 2 y 4 afectan al momento en el eje X, *roll*, ver ecuación 6.1.
- Los motores 1 y 3 afectan al momento en el eje Y, *pitch*, ver ecuación 6.2.

- El conjunto de los cuatro motores afecta al momento en el eje Z, *yaw*, de dos formas diferentes.
  - La primera por la diferencia entre pares, es decir, por la rotación sobre si mismo, ver ecuación 6.3.
  - La segunda se corresponde al empuje total que se realiza sobre el vector fuerza en dirección hacia arriba, ver ecuación 6.5.

6.3.2. Aceleración Angular (α)

Hasta este momento se han obtenido los momentos que se ejercen sobre el cuadricóptero. Para obtener la aceleración del aparato hay que relacionar dichos momentos con el concepto de aceleración.

La fuerza ejercida sobre un cuerpo es igual al producto de su masa por la aceleración, $F = m * a$, por lo tanto la aceleración de ese cuerpo sera igual a la división de la fuerza entre la masa. Una vez relacionado el concepto de aceleración y fuerza y sabiendo que los momentos angulares son la resistencia que ofrece el aparato a la variación de la velocidad angular, es decir, la fuerza que se aplica sobre este, se obtiene la ecuación 6.6.

$$Aceleracion\_Angular(\alpha) = \frac{Momento\_Angular(\tau)}{Tensor\_Inercia(m)} \tag{6.6}$$

6.3.3. Velocidad Angular (ω)

En la sección anterior se definió la relación de la aceleración angular con los momentos producidos en el cuadricóptero. Una vez obtenida esta relación y sabiendo que la aceleración es el cambio que experimenta la velocidad angular por unidad de tiempo se obtiene la ecuación 6.7.
Esta ecuación representa el cambio, es decir, la variación que sufre en Δt la velocidad. Entonces para obtener el valor de la velocidad en el instante de tiempo Δt^2 se le incrementa a la velocidad obtenida en el instante anterior de tiempo (*t-1*) la variación obtenida en *t*, ver ecuación 6.8.

$$Variacion\_Velocidad\_Angular(\Delta\omega) = Aceleracion\_Angular(\alpha) * \Delta t \tag{6.7}$$

$$Velocidad\_Angular(\omega) = Velocidad\_Angular'(\omega') + Variacion\_Velocidad\_Angular(\Delta\omega) \tag{6.8}$$

6.3.4. Rotación ($\phi\theta\psi$)

Hasta el momento se han explicado las definiciones y relaciones entre las intensidades, fuerzas, momentos, aceleraciones y velocidades.
A continuación se define la relación y los pasos necesarios para obtener los tres grados de rotación a partir de estas variables.

- La rotación (**R**) en el eje *i*, donde la variable *i* toma un valor entre *x, y, z* es igual a la rotación en el instante anterior (R') de ese eje más la variación de la rotación (ΔR), es decir, la rotación en un punto equivale al valor que tenía más el movimiento producido desde un instante anterior hasta el momento actual, ver ecuación 6.9.

$$Rotacion(R) = Rotacion'(R') + Variacion\_Rotacion(\Delta R) \tag{6.9}$$

[2]Valor temporal que representa el mínimo cambio entre estados.

- La variación de la rotación, ver ecuación6.10, de un eje es igual al producto de la velocidad de rotación ($\Delta\omega$) del mismo eje, también conocida como velocidad angular[3] por Δt.

$$Variacion\_Rotacion(\Delta R) = Velocidad\_Angular(\omega) * \Delta t \tag{6.10}$$

- La velocidad angular de la que depende la variación de la rotación rotación es por tanto dependiente de la aceleración angular, ver ecuación 6.6, que sufre el cuadricóptero en el instante Δt.

Sustituyendo entonces los correspondientes valores por las variables, momentos, obtenidos anteriormente se deducen las ecuaciones 6.11, 6.12, 6.13 que definen los 3 grados de rotación.

$$\Phi \Rightarrow R_x = R'_x + \Delta R_x \Rightarrow \omega_x * \Delta t \Rightarrow \omega'_x + (\alpha_x * \Delta t) \Rightarrow \frac{\tau_\phi}{m} \Rightarrow \frac{d(F_2 - F_4)}{m} \tag{6.11}$$

$$\Theta \Rightarrow R_y = R'_y + \Delta R_y \Rightarrow \omega_y * \Delta t \Rightarrow \omega'_y + (\alpha_y * \Delta t) \Rightarrow \frac{\tau_\theta}{m} \Rightarrow \frac{d(F_1 - F_3)}{m} \tag{6.12}$$

$$\psi \Rightarrow R_z = R'_z + \Delta R_z \Rightarrow \omega_z * \Delta t \Rightarrow \omega'_z + (\alpha_z * \Delta t) \Rightarrow \frac{\tau_\psi}{m} \Rightarrow \frac{d((F_1 + F_3) - (F_2 + F_4))}{m} \tag{6.13}$$

Para completar el modelo físico queda obtener los tres grados de traslación. Estos grados se obtienen a partir de las fuerzas que se producen sobre cada uno de los ejes.

6.3.5. Fuerza de los ejes (F_x, F_y, F_z)

La fuerza que se produce sobre cada eje se obtiene a partir de las rotaciones obtenidas anteriormente y el vector fuerza que se ejerce en el centro del helicóptero en dirección hacia arriba, empuje total (μ).

Los tres grados de rotación son necesarios debido a que el cuadricóptero se encuentra girado sobre los tres ejes, tal y como reflejan los ángulos de Euler *roll, pitch y yaw* definidos anteriormente.

Entonces, para calcular la fuerza producida sobre los tres ejes se debe girar el vector fuerza, μ, de acuerdo a la rotación del aparato, es decir, se debe obtener la matriz que considere la rotación en los ángulos *roll y pitch.*
No se tiene en cuenta la rotación en el ángulo *yaw* debido a que se trata de la rotación que gira sobre su propio eje, dicha rotación no produce ningún efecto sobre el vector fuerza situado en el centro del aparato. La veracidad de esta ultima afirmación se demuestra en las siguientes lineas, durante la obtención de la matriz.

La matriz se obtiene de estudiar los ángulos de navegación que describen la orientación del cuadricóptero en 3D, concretamente para el caso de ingeniería aeronáutica se definen los ángulos de Tait-Bryan [14]. Estudiando la definición de dichos ángulos(*roll, pitch y yaw*) con respecto al sistema de referencia (x y z) nos encontramos ante una rotación intrínseca Z-Y-X dentro del plano estático xy — YZ.

A partir de las matrices de rotación [15] de vectores alrededor de los ejes x, y o z siguientes:

$$R_Z(\psi) = \begin{vmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad R_Y(\theta) = \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{vmatrix} \qquad R_X(\phi) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{vmatrix}$$

[3] Ángulo girado por unidad de tiempo.

Y usando la convención Z-Y-X se obtiene, a partir de la siguiente secuencia, la matriz de rotación (R_{xyz}):

- La primera rotación mueve el cuerpo alrededor del eje Z un ángulo con valor yaw, por lo que la primera ecuación con la que operamos es R_z.

$$R_Z(\psi) = \begin{vmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

- La segunda rotación mueve el cuerpo alrededor del eje Y un ángulo con valor pitch. Teniendo en cuenta que la primera rotación produjo un giro sobre R_z al que en la segunda rotación le aplica un nuevo giro de R_y se obtiene R_{yz}.

$$R_{yz} = \begin{vmatrix} \cos\psi\cos\theta & -\sin\psi & \cos\psi\sin\theta \\ \sin\psi\cos\theta & \cos\psi & \sin\psi\sin\theta \\ -\sin\theta & 0 & \cos\theta \end{vmatrix}.$$

- La tercera rotación mueve el cuerpo alrededor del eje X un ángulo con valor roll. Debido a las dos primeras rotaciones el cuerpo se encuentra girado sobre los ejes yz por lo que para obtener la rotación R_{xyz} se le aplica el nuevo giro R_x a R_{yz}.

$$R_{xyz} = \begin{vmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{vmatrix}.$$

A partir de la secuencia anterior, y sabiendo que el ángulo Z (ψ) no afecta al vector fuerza debido a que el giro de un cuerpo sobre si mismo no modifica su valor, este ángulo pasa a valer 0, obteniendo así la matriz de rotación *xyz*.

$$R_{xyz} = \begin{vmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{vmatrix}.$$

Aplicando el giro (R_{xyz}) al vector fuerza (μ) se obtiene las fuerzas, ver ecuaciones 6.14, 6.15, 6.16, que se producen sobre cada uno de los ejes.

$$Fuerza_X = -\mu * \sin\theta \tag{6.14}$$

$$Fuerza_Y = \mu * \cos\theta * \sin\phi \tag{6.15}$$

$$Fuerza_Z = \mu * \cos\theta * \cos\phi \tag{6.16}$$

Por otro lado hay que tener en cuenta que la fuerza de la gravedad[4] se aplica sobre cualquier cuerpo en el espacio, por lo que la fuerza en el eje Z quedará de la siguiente forma:

$$Fuerza_Z = (\mu * \cos\theta * \cos\phi) - (m * g) \tag{6.17}$$

, donde m es el peso del cuadricóptero y g es la constante de gravedad ($9.8\text{m}/s^2$).

[4]Vector fuerza en dirección hacia abajo.

6.3.6. Aceleración (a) y Velocidad (υ)

Una vez obtenida la fuerza y conociendo la relación entre fuerza y aceleración tal y como se vio en la sección 6.3.2 se obtiene para cada uno de los ejes que,

- La aceleración que sufre el cuadricóptero se define mediante la siguiente ecuación:

$$Aceleracion(a) = \frac{Fuerza}{m} \tag{6.18}$$

- La variación de la velocidad en el eje es igual a la aceleración que se produce durante el tiempo Δ t.

$$Variacion\_Velocidad(\Delta\upsilon) = \Delta a * \Delta t \tag{6.19}$$

  Por lo que la velocidad en Δt_i es igual a la velocidad a la que circula el aparato en Δt_{i-1} en el momento anterior mas su variación en ese instante del tiempo.

$$Velocidad(\upsilon) = Velocidad'(\upsilon') + \Delta\upsilon \tag{6.20}$$

6.3.7. Traslación

Para completar el modelo básico solo queda obtener los tres grados de traslación. Estos grados se calculan a partir de la siguiente ecuación que relaciona el espacio que recorre un cuerpo con la velocidad a la que circula.

$$Espacio(\upsilon) = Velocidad(\upsilon) * \Delta t \tag{6.21}$$

De nuevo, tal y como se ha visto ahora, el espació es una variable que depende de su valor en el instante anterior. Aplicando este concepto a cada uno de los ejes y teniendo en cuenta la matriz de traslación de un objeto se obtiene,

$$\begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & t_X \\ 0 & 1 & 0 & t_Y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix}$$

$$x = x' + t_X \qquad y = y' + t_Y \qquad z = z' + t_Z \tag{6.22}$$

,donde $x'y'z'$ marcan la posición en el momento anterior y t_X, t_Y y t_Z son los equivalentes a los movimientos en el espacio XYZ obtenidos anteriormente en la ecuación 6.21 (aplicada a cada eje) a partir de la velocidad a la que circula el cuadricóptero.

Capítulo 7

Sistema de Visualización

La ultima pieza del modelo esta compuesta por el sistema de visualización, tal y como se ha venido adelantando en capítulos anteriores esta pieza se utiliza como medio para mostrar de forma gráfica la respuesta a cambios, producidas por las entradas del sistema, del cuadricóptero.

Para el desarrollo de este componente se ha utilizado el kit de herramientas de visualización (*Visual Tool Kit*, VTK). Se trata de un sistema de *software* libre multi-plataforma[1] para la realización de gráficos en 3D, procesamiento de imagen y visualización. VTK esta implementada como un conjunto de herramientas de `C++` de esta forma los usuarios que realicen el desarrollo con ella deben combinar varios objetos para crear una aplicación.

Tal y como se había comentado en la sección de limitaciones del modelo UML/MARTE únicamente se soporta el uso de la compilación automática para código `C`. VTK se trata de una librería que permite su programación en `C++` y además es capaz de interpretar `Phyton`, `Java` y `Tcl` permitiendo su desarrollo en estos lenguajes. Debida a la limitación anterior se ha llevado a cabo el desarrollo del sistema de visualización mediante la creación de una librería, es decir, las funciones para la creación y representación del cuadricóptero se encuentran compiladas y listas para su uso.

El sistema de visualización desarrollado, `visualTk`, dispone de varias funciones entre las que se encuentran el lanzado y actualización del entorno VTK, funciones a través de las cuales se realiza la comunicación con la aplicación principal.

- Función `throwVtk`: se trata de la función principal que se encarga de configurar y construir el sistema. Esta función es llamada una única vez al inicio de la ejecución de la aplicación.

```
void throwVtk()
{
  //Create Renderer and RenderWindow
  renderWindow = vtkSmartPointer<vtkRenderWindow>::New();
  renderer = vtkSmartPointer<vtkRenderer>::New();

  renderWindow->AddRenderer(renderer);
  renderWindow->SetSize(700,700);

  Building build;
  build.printBuilding(renderer);

  actorBackground = insertBackground("support/cityislands/CityIslands/
      CityIslands.obj");
```

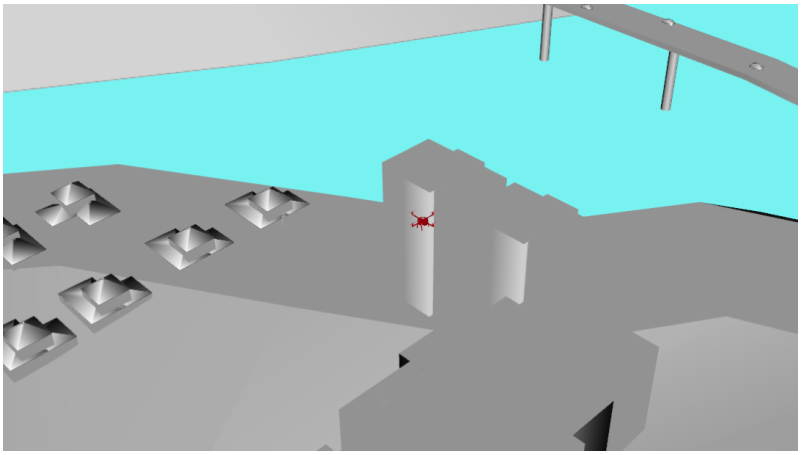
[1]Soporta su ejecución tanto en Linux, Windows, Mac y Unix.

```
    renderer->SetBackground(0.41,0.95,0.95);

    string filename = "support/quadcopter/model.obj"; //insert quadcopter
        object
    actorQuadcopter = insertObj(filename);

    double rotation [3] = {0,0,0};
    double translation[3] = {0,0,0}; //Initial position

    //Create camera
    camera = vtkSmartPointer<vtkCamera>::New();

    //Throw view
    updateViewCamera(rotation, translation);
  }
```

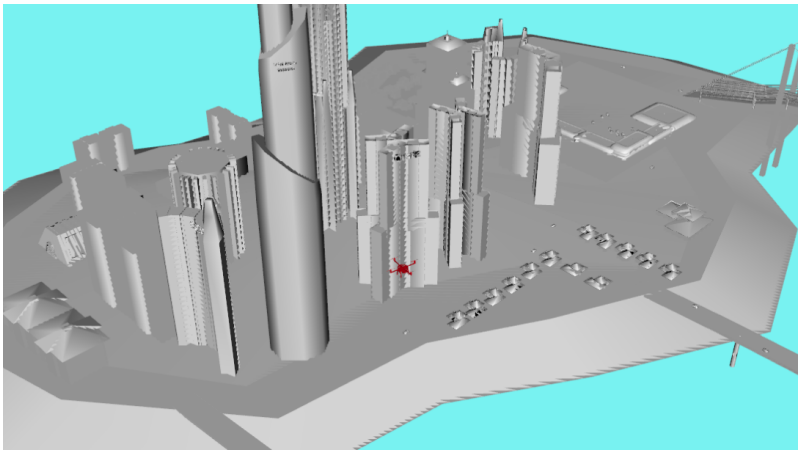
- Función `updateViewCamera`: se trata de la función encargada de realizar el refresco de la imagen.
  Esta función recibe una llamada periodica cada segundo realizada desde el componente `OperationsModel` encargado de obtener la posición y orientación del cuadricóptero.

```
  void updateViewCamera(double rotation[], double translation[]){

    //Normalize axis
    normalizeAxis(translation);
    normalizeAxis(rotation);

    //SCALE 1:10
    translation[0] = translation[0]/10;
    translation[1] = translation[1]/10;
    translation[2] = translation[2]/10;

    printf("Rotation_Vtk x: %.2f, y: %.2f, z: %.2f \n", rotation[0],
        rotation[1], rotation[2]);
    printf("Traslacion_Vtk x: %.2f, y: %.2f, z: %.2f \n", translation[0],
        translation[1], translation[2]);

    //Update position of quadcopter
    setPositionActor(actorQuadcopter, rotation, translation);

    //New position of camera
    double x=25.0+translation[0];
    double y=25.0+translation[1];
    double z=25.0+translation[2];

    camera->SetPosition(x, y, z);
    camera->SetFocalPoint(translation[0], translation[1], translation[2]);

    renderer->SetActiveCamera(camera);

    renderWindow->Render();

    renderWindow->Start();
  }
```

La figura7.1 muestra algunos ejemplos de uso de la librería, en ella se puede ver al cuadricóptero en vuelo.



(a)



(b)

Figura 7.1: Capturas Aplicación `Visualtk`

Capítulo 8

Medidas y Resultados

En este capitulo y después de haber defino el modelo desarrollado en los anteriores se explican los pasos dados para adaptar dicho modelo a la realidad física del cuadricóptero.

8.1. Medidas

Lo primero que hay que tener en cuenta son los valores físicos, ver tabla 8.1 dependientes de las características del aparato que vienen de fabrica.

Características	Valor
Motores	4 unid.
Longitud brazo (d)	25 cm.
Masa (m)	1.5 Kg.
Alcance Radio	4 Km.
Gravedad (g)	9.8 m/s^2

Cuadro 8.1: Características Físicas

Además se tienen en cuenta las siguientes consideraciones:

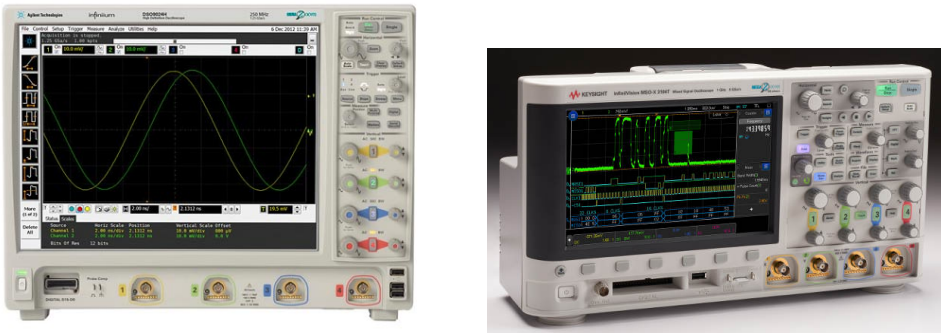
- La estructura del cuadricóptero es rígida de tal manera que no se tiene en cuenta ninguna posible deformación.
- El centro de gravedad (COG) y el origen del eje del cuerpo coinciden.
- Las hélices son rígidas.
- No se tienen en cuenta las perturbaciones que puedan ocasionar las inclemencias del entorno.

Una vez que se tienen claros los valores fijos se pasan a recoger datos, valores variables, mediante la realización de medidas sobre el cuadricóptero real.

La recogida de datos consiste en analizar que sucede desde que se realiza un movimiento a alguno de los *sticks* del radio control hasta que los controladores de los motores reciben la orden que les indica la intensidad que deben proporcionar a los motores.

Recordando lo visto en capítulos anteriores se sabe que el primer paso es el intercambio que realiza el radio control con el controlador de vuelo a través de una señal *PPM*, esta información es procesada por dicho controlador (caja negra, no se dispone de información) que se encarga de obtener y transmitir la intensidad que le corresponde a cada motor. Esta ultima transmisión se realiza en dirección a los controladores de los motores a través del bus de comunicación *I2C*.
Finalmente se analizará de donde se obtiene la relación Intensidad-Fuerza que permite la definición y el desarrollo del modelo físico.

Para realizar las medidas se han utilizado dos variantes de osciloscopios, ver figura 8.1



(a) Serie 9000 (b) Serie 3000

Figura 8.1: Osciloscopios *Agilent Technologies*

8.1.1. PPM

En la sección 4.3.1 se analizaban las características de la señal PPM.
La recogida de datos de la señal *PPM* se ha realizado de forma simultanea mediante dos pasos:

- Uso del osciloscopio Serie 9000 para medir los pulsos de la señal *PPM* que recibe el receptor conectado a la placa principal, siendo necesaria la soldadura de dos cables a la dicha placa, tierra y *PPM*.

  Las características de la señal analizada son las siguientes:

  - Periodo de la señal (T) tiene una duración de 30ms.
  - Señal de 12 canales, 13 flancos de subida y 13 de bajada.
  - Valor de la señal a 0: periodos constantes de 0.5ms, identificativos del cambio de canal.
  - Valor de la señal a 1: periodos variantes entre 0.58ms y 1.42ms entre los canales 1 y 12.
  - La sincronización de la señal se realiza mediante el envió final de una cadena larga de valor 1. Esta cadena tiene una duración t variante, dependiente de la suma de los 12 primeros pulsos, encargada de mantener la periodicidad de la señal.
  - Los canales *gas*, *roll*, *pitch* y `yaw` se encuentran alojados en los 4 primeros pulsos de la señal.
  - En estado de reposo los 4 pulsos tienen una duración de 1.00ms.

- Uso del software de aplicación del cuadricóptero para la obtención del valor decimal de 8 *bits* que el transmisor (emisora) envía por cada uno de los cuatro canales (gas, roll, pitch y yaw),

conexión mediante la placa MK-USB.
El *hardware* de la placa MK-USB permite la conexión del cuadricóptero al PC, a través de esta conexión se realizan las configuraciones y actualizaciones del aparato (número de motores, posición de estos, configuración de canales, etc.) además muestra en tiempo real la variación que sufren los canales.

Posición	Roll-Pitch-Yaw		Gas	
	RC (Decimal)	PPM (ms)	RC (Decimal)	PPM (ms)
-5	0	0.58	-	-
-4	25	0.58	-	-
-3	51	0.64	-	-
-2	76	0.76	-	-
-1	102	0.88	-	-
0	128	1.0	128	1.0
+1	151	1.12	151	1.12
+2	177	1.24	177	1.24
+3	203	1.36	203	1.36
+4	229	1.42	229	1.42
+5	255	1.42	255	1.42

Cuadro 8.2: Radio Control - PPM

La tabla que se encuentra sobre estas lineas muestra la relación entre los datos obtenidos. Dicha tabla es la base de la que parte la generación del algoritmo de codificación y decodificación de la señal.

8.1.2. I²C

En la sección 4.3.3 se analizaban las características del funcionamiento del bus I^2C.
La recogida de datos se ha realizado mediante el uso del osciloscopio serie 3000 capaz de decodificar la señal que se transmite por el bus, extrayendo los datos en formato decimal.
Lo primero a tener en cuenta son las características propias de la comunicación:

- Direccionamiento de los motores: direcciones pares para la comunicación entre el controlador de vuelo y los motores y direccionamientos impares para la comunicación en sentido opuesto:
  - *Master to Slave*: direcciones 0x52, 0x54, 0x56 y 0x58 correspondientes a los motores 1, 3, 2 y 4 respectivamente.
  - *Slave to Master*: direcciones 0x53, 0x55, 0x57 y 0x59 correspondientes a los motores 1, 3, 2 y 4 respectivamente.
- Velocidad de trasmisión de 400.000kHz
- Principalmente se envían tramas de datos de 2 Bytes (16bits), aunque depende de la información que se transmita. El formato principal de la trama posee la siguiente forma:
  `D_10, D_9, D_8, D_7, D_6, D_5, D_4, D_3, C_4, C_3, C_2, C_1, C_0, D_2, D_1, D_0`
  ,los bits `D` se corresponden con datos de información, los bits `C` corresponden a datos de configuración.

- 11 bits para indicar la corriente que debe circular por el motor (unidad: 10mA), por lo tanto el rango de esta varía de 0 a 2047 (de 0 a 20.47 Amperios).
  `D_10, D_9, D_8, D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0`
- 5 bits de configuración, normalmente toma valor 0 (`00000`). Si la velocidad es cero el valor `10000` se usa para indicar el paso a modo configuración.
  `C_4, C_3, C_2, C_1, C_0`

- Protocolo de comunicación:

  - *Master to Slave*: los bits correspondientes a los datos de información de la corriente están distribuidos entre el primer byte, que contiene los 8 primeros bits y el segundo byte que en su parte menos significativa (LSB's[1]) contiene los 3 bits restantes. Los bits correspondientes a configuración, por tanto, se encuentran en los primeros 5 bits (MSB's[2]) del segundo byte.
    Si el segundo byte tiene valor 0, el maestro únicamente envía una trama de un 1 byte correspondiéndose los 8 bits al valor de la corriente en cuyo caso el rango varia de 0 a 255 en lugar de 0 a 2047. Es la placa de control de los motores la encargada de realizar la equivalencia entre un rango y el otro.
  - *Slave to Master*: en lugar de enviar 2 bytes, los esclavos mandan 3 bytes de información. El primer byte se corresponde con el consumo de potencia que esta realizando el motor que se utiliza para realizar los cálculos de la duración de la batería en tiempo real. El segundo byte se corresponde con `Max PWM`. El tercer y ultimo byte indica la temperatura del motor en grados centígrados.

Las características anteriores abarcan la comunicación bidireccional entre las placas, para la realización de este proyecto inocentes se tienen en cuenta las tramas de información relativas a la corriente de los motores (ver figura 8.2).

(a) Señal I^2C (b) Trama señal I^2C

Figura 8.2: Señal y Trama I^2C

Las pruebas se realizan con el cuadricóptero colocado en una superficie plana con la menor inclusión de perturbaciones posibles, aun así se puede observar la existencia de vibraciones producida por los motores. Se realizan varias pruebas de las que se obtiene la figura 8.3 como tabla-resumen de los datos obtenidos.

[1] *Least Significant Bit*, posiciones de bits en un número binario que tienen menor valor.
[2] *Most Significant Bit*, posiciones de bits en un número binario que tienen mayor valor.

			GAS-0				GAS-1			
ROLL	PITCH	YAW	0x52	0x54	0x56	0x58	0x52	0x54	0x56	0x58
-1	-1	-1	40	224	224	82	342	1840	1840	675
-1	-1	0	64	224	224	64	506	1840	1840	506
-1	-1	1	82	224	224	40	675	1840	1840	342
-1	0	-1	124	124	224	82	1564	1564	1840	675
-1	0	0	143	143	224	64	1680	1680	1840	506
-1	0	1	124	124	224	82	1564	1564	1840	675
-1	1	-1	224	40	224	82	1840	342	1840	675
-1	1	0	224	64	224	64	1840	506	1840	506
-1	1	1	224	82	224	40	1840	675	1840	342
0	-1	-1	40	224	224	224	342	1840	1806	1806
0	-1	0	64	224	143	143	506	1840	1680	1680
0	-1	1	82	224	143	143	675	1840	1564	1564
0	0	-1	112	112	176	176	1520	1520	1840	1840
0	0	0	143	143	143	143	1680	1680	1680	1680
0	0	1	176	176	112	112	1840	1840	1520	1520
0	1	-1	224	40	224	224	1840	342	1806	1806
0	1	0	224	64	143	143	1840	506	1680	1680
0	1	1	224	82	143	143	1840	675	1564	1564
1	-1	-1	40	224	82	224	342	1840	675	1840
1	-1	0	64	224	64	224	506	1840	506	1840
1	-1	1	82	224	40	224	675	1840	342	1840
1	0	-1	124	124	82	224	1564	1564	675	1840
1	0	0	143	143	64	224	1680	1680	506	1840
1	0	1	220	220	40	224	1806	1806	342	1840
1	1	-1	224	40	82	224	1840	342	675	1840
1	1	0	224	64	64	224	1840	506	506	1840
1	1	1	224	82	40	224	1840	675	342	1840

Figura 8.3: Resumen de datos analizados con i2c

La figura 8.3 muestra una tabla con las características siguientes; sus tres primeras columnas están compuestas por las posibles combinaciones generadas por el valor máximo, medio y mínimo de los movimientos *roll*, *pitch* y *yaw*, el resto de las columnas muestran los valores obtenidos por cada motor para estas combinaciones tanto si hay presencia de *gas* como si no.

Analizando todos los datos obtenidos finalmente se realizo un algoritmo de interpolación para obtener la corriente a partir del valor decodificado de la señal PPM, es decir, el algoritmo se basa en los datos recogidos para calcular valores medios, teniendo en cuenta de que el peso del valor varía en función del giro que se quiera realizar, en la siguiente matriz se muestra la relación que existe entre las variables (*Gas, Roll, Pitch, Yaw*) y los motores (`M1, M2, M3, M4`).

$$\begin{vmatrix} - & G & R & P & Y \\ M1 & 1 & 0 & 1 & 1 \\ M2 & 1 & -1 & 0 & -1 \\ M3 & 1 & 0 & -1 & 1 \\ M4 & 1 & 1 & 0 & -1 \end{vmatrix}$$

8.1.3. Fuerza

En el ultimo paso de la toma de datos se encuentra obtener la relación entre la corriente eléctrica que recibe cada uno de los motores y la fuerza que estos producen, dicha relación se encuentra de forma gráfica tal y como se muestra en la figura 8.4 en la propia descripción del motor.
A partir de la gráfica anterior se realiza un ajuste lineal, es decir, se obtiene una recta de regresión que se ajuste a la nube de puntos y se calcula sobre esta la función `y = mx + b` que relaciona la corriente (variable x) con la fuerza (variable y).
La función resultante obtenida queda de la siguiente forma: `Fuerza = 0.75*Corriente + 216.67`.

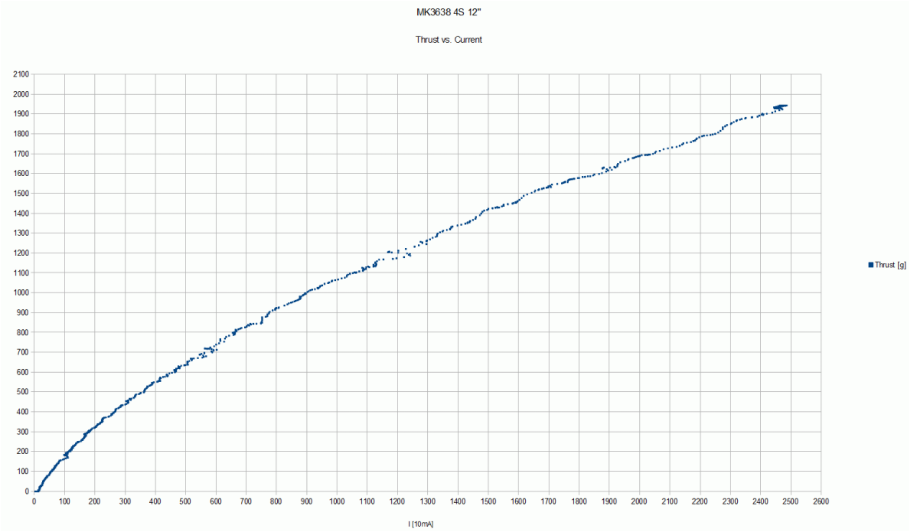
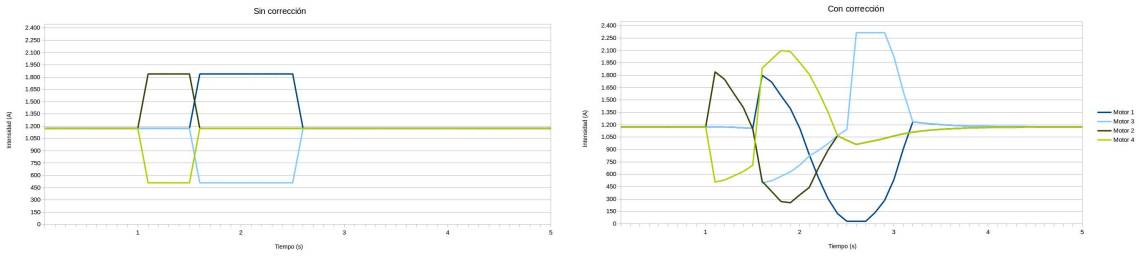


Figura 8.4: Relación Corriente-Fuerza del Motor MK-3638

8.2. Resultados

El objetivo de esta sección es mostrar el conjunto de resultados que se produce cuando los motores se alimentan de una determinada intensidad comprobando así, a partir de gráficas, el funcionamiento del sistema modelado desde que recibe la intensidad hasta que produce los valores tanto de la rotación como de la traslación que sufre el aparato.



(a) Sin aplicar corrección

(b) Aplicada la corrección

Figura 8.5: Intensidad de los Motores

La figura 8.5 muestra dos gráficas correspondientes a la evolución de los valores de entrada del sistema. La primera gráfica muestra la intensidad que se recibe como entrada, es decir, muestra el valor indicado por el radio control para cada uno de los motores. La segunda gráfica muestra el valor corregido de la intensidad, es decir, se corresponde con el valor que el modelo recibe como entrada una vez aplicados los coeficientes de corrección del giroscopio a la intensidad de cada uno de los motores que muestra la primera gráfica.

Tal y como muestra la primera figura el primer cambio se realiza en el segundo uno sobre los `motores 2 y 4` por lo tanto se simula una perturbación igual a la que se produce para realizar el giro *roll*, el segundo cambio que se provoca afecta a los `motores 1 y 3` simulando la perturbación que se produce en los giros *pitch*.
En la segunda gráfica se observa como una vez comenzada cualquier maniobra el sistema comienza a aplicar las correcciones necesarias para estabilizar el aparato, este punto se alcanza un segundo y medio después de generar las perturbaciones.

En el párrafo anterior se menciona la existencia de valores de corrección necesarios para estabilizar el aparato, la figura 8.6 muestra el motivo de esa necesidad. Si se observa dicha figura se puede ver como el cuadricóptero no encuentra su valor de consigna[3] hasta pasado el tercer segundo.
Además hay que tener en cuenta que el valor de consigna se obtiene cuando la velocidad angular es cero, la figura 8.7 muestra la evolución de dicha velocidad por cada uno de los ejes.

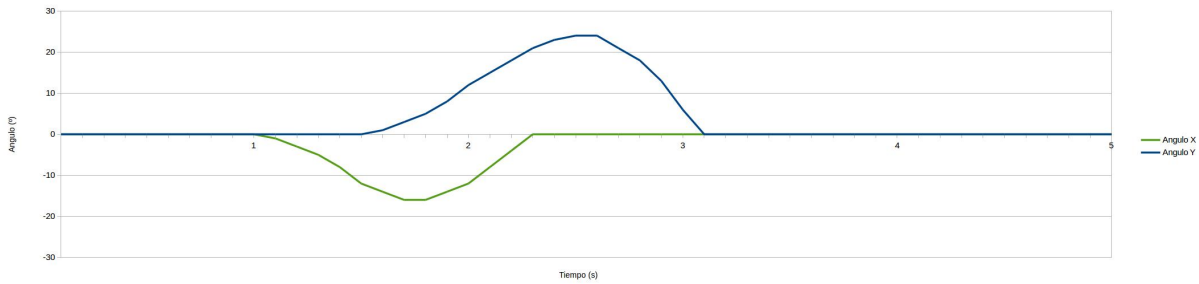


Figura 8.6: Posición Angular

En la gráfica que muestra la velocidad angular se puede ver como los cambios en las intensidades debidos a los giros *roll* y *pitch* necesarios para estabilizar el vuelo provocan la aparición del tercer movimiento, *yaw*. Debida a la aparición de esta nueva variable aunque el aparato quede estabilizado en los ejes X e Y en el tercer segundo, es decir, un segundo después de la introducción de los giros, este no queda realmente estabilizado hasta que reduce la velocidad del eje Z añadiéndole otro segundo de estabilización.

La ultima gráfica presentada, ver figura 8.8, muestra la evolución que sufre el aparato con respecto a su posición de inicio. Con esta gráfica únicamente se controla que el aparato siga la trayectoria deseada, es decir, los datos introducidos indican que debe seguir una trayectoria ascendente y girar levemente hacia al izquierda a la vez que comienza ir marcha atrás tal y como indican las intensidades aplicadas a cada motor.
Observando la gráfica se ve como cumple su trayectoria, incluso cuando al tercer segundo, después de haber estabilizado los ángulos X e Y, comienza a desplazarse en dirección contraría (hacia la

[3]Valor que se corresponde físicamente con el valor que mantiene al cuadricóptero estable respecto a los ejes de referencia, es decir, valor cero.

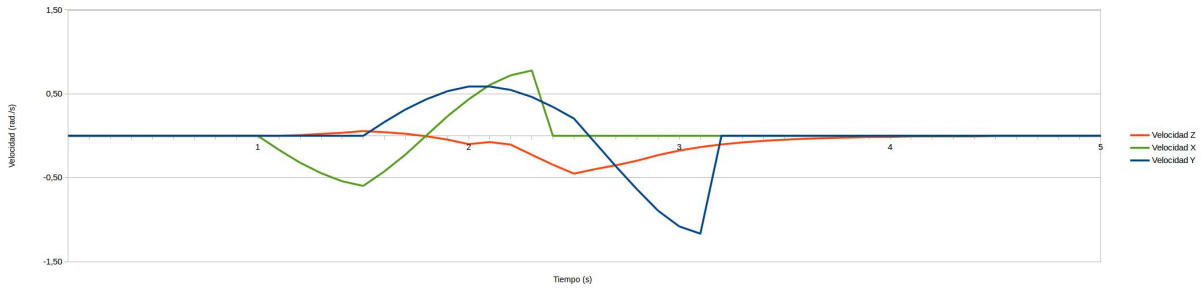


Figura 8.7: Velocidad Angular

derecha y adelante) esto se debe al cambio en la aceleración de ambos ejes que se introdujo para lograr la estabilidad. Hay que tener en cuenta que el valor cero en la velocidad angular únicamente hace cero el valor de la aceleración que sufre el cuadricóptero en ese eje pero no implica que la velocidad de traslación sea cero por lo que el cuadricóptero continua desplazándose en la dirección y con la velocidad que tenía previamente.

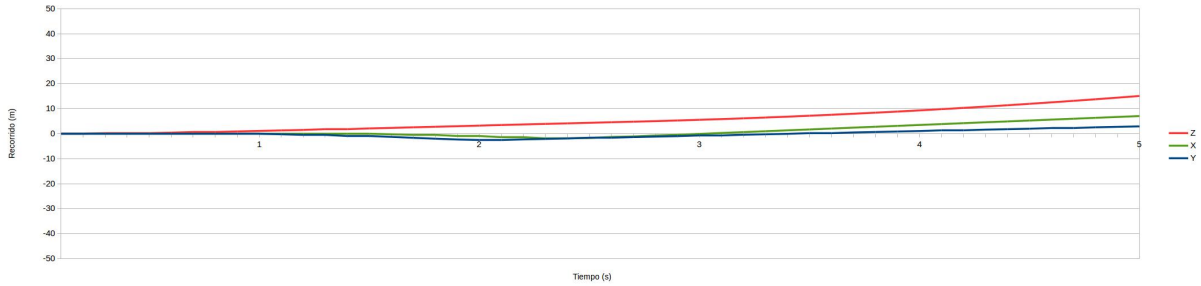


Figura 8.8: Traslación

Capítulo 9

Conclusión y Trabajos Futuros

Para finalizar en las dos siguientes secciones de este proyecto se muestran las conclusiones alcanzadas y los posibles trabajos futuros a desarrollar a partir de este proyecto.

9.1. Conclusión

En vista de las pruebas y resultados obtenidos se puede concluir que se ha alcanzado el objetivo principal del proyecto, es decir, se ha probado la capacidad de las herramientas eSSYN a partir de la generación de un modelo UML/MARTE, observando que aunque carece de soporte relativo a los tiempos de ejecución o al uso de cualquier lenguaje de programación que no sea `C` es capaz de generar el código necesario para realizar la simulación de un sistema real.

Por lo tanto se ha generado un modelo de un sistema ciberfísico en UML, del que con ayuda de las herramientas eSSYN se genera automáticamente la implementación del modelo reduciendo en este paso la probabilidad de error.
La existencia de este modelo ciberfísico nos permite la simulación y reducción del error en el uso del aparato para futuros trabajos, por ejemplo, la inclusión en el cuadricóptero de un software de vuelo automático que sustituya el uso del radio control.

Sin embargo teniendo en cuenta las medidas y resultados relativos a las primeras pruebas realizadas se observo la dificultad de modelar un sistema que represente completamente la realidad del cuadricóptero. Esta dificultad viene dada por la imposibilidad de realizar la recogida de datos cuando el cuadricóptero se encuentra en vuelo. Por este motivo se puede asegurar que el modelo simulado representa la realidad del cuadricóptero teniendo en cuenta que los datos han sido obtenidos cuando este se encontraba sobre una superficie plana.

9.2. Trabajos futuros

Entre los trabajos futuros que se pueden realizar a partir de este caben destacar los siguientes:

- Incrementación del número de variables a modelar aumentando así el nivel de detalle del modelo, por ejemplo, la inclusión de las variables relativas a las inclemencias meteorológicas.
- En base a los resultados obtenidos se puede realizar la mejora de la implementación del algoritmo de estabilización del giroscopio, ya que para este trabajo únicamente se genero un algoritmo básico que mantuviese al aparato estable, pero tal y como muestran los resultados

aunque cumple con su objetivo no posee la robustez suficiente, por tanto la implementación ideal consistiría en la inclusión del algoritmo de control PID descrito en la sección 4.2.3.

- Respecto a los datos recogidos del cuadricóptero se podría generar un "entorno de simulación controlado", es decir, el objetivo sería la equipación de un laboratorio con sensores que permita desde fuera la recogida de datos del cuadricóptero durante el vuelo (altura, inclinación, etc.), además se le podría dotar de generadores de viento y sonido para observar su respuesta ante dichas perturbaciones. Este trabajo incluiría también la colocación de sensores en el cuadricóptero que permitan la lectura de la velocidad a la que circula cada motor en tiempo real. La nueva recogida de datos implicaría la posterior modificación del modelo generado para ajustarlo más realidad.
- Inclusión del modelo en un entorno de verificación formal con el objetivo de demostrar la correcta implementación del modelo.

Como se puede observar los trabajos que se acaban de describir se orientan a la mejora de la especificación del nivel de detalle del modelo de manera que durante la simulación se reduzca la probabilidad de error respecto del sistema real. Aumentar el nivel de detalle de este modelo permitiría el traspaso de pruebas del sistema simulado al sistema real incrementando la certeza de haber reducido el error entre ambos al máximo posible.

Bibliografía

[1] A. Barrientos, J. del Cerro, P. Gutiérrez, R. San Martín, A. Martínez, C. Rossi, *Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones*, Universidad Politecnica de Madrid, 2007.

[2] Manuel Cosío, *"El poder de los cuadricopteros"*, 2013, http://www.cassetteblog.com/2013/06/el-poder-de-los-cuadricopteros.

[3] Virginiea Mazzone, *Controladores PID*, Universidad Nacional de Quilmes, 2002.

[4] ESLANEK, *"Control PID en multicopteros"*, 2014, http://fpvunlimited.com/control-pid-en-multicopteros

[5] Microrobotica, *"Codificación en RC"*, http://www.microbotica.es/h1/doku.php?id=eh1:codificacionrc

[6] I2C.info, *"Everything About I2C"*, http://i2c.info/i2c-bus-specification.

[7] omgmarte.org, *"The UML Profile for MARTE"*, http://www.omgmarte.org.

[8] eclipse.org, *"PAPYRUS"*,https://eclipse.org/papyrus.

[9] P. Peñil, *UML/MARTE Methodology for Heterogeneous System design*, Universidad de Cantabria, 2014.

[10] wikipedia.org, *"Newton-Euler equations"*, https://en.wikipedia.org/wiki/Newton-Euler_equations

[11] Quadcopter|Drone Flyers, *"Quadcopters Yaw Roll and Pitch Defined"*, http://www.quadcopterflyers.com/2015/02/quadcopters-yaw-roll-and-pitch-defined.html

[12] Rodrigo Alberto Mayorga Rodríguez, *Sistema de Navegación para Vehículos Aéreos Cuadricópteros*, Universidad Politécnica de Cataluña, 2009.

[13] C. Balas, *Modeling and linear control of cuadcopter*, Cranfield University, 2006-2007.

[14] wikipedia.org, *"Euler angles"*, https://en.wikipedia.org/wiki/Euler_angles

[15] wikipedia.org, *"Rotation matrix"*, https://en.wikipedia.org/wiki/Rotation_matrix

Anexos

Anexo A

Vista de Datos (Data View)

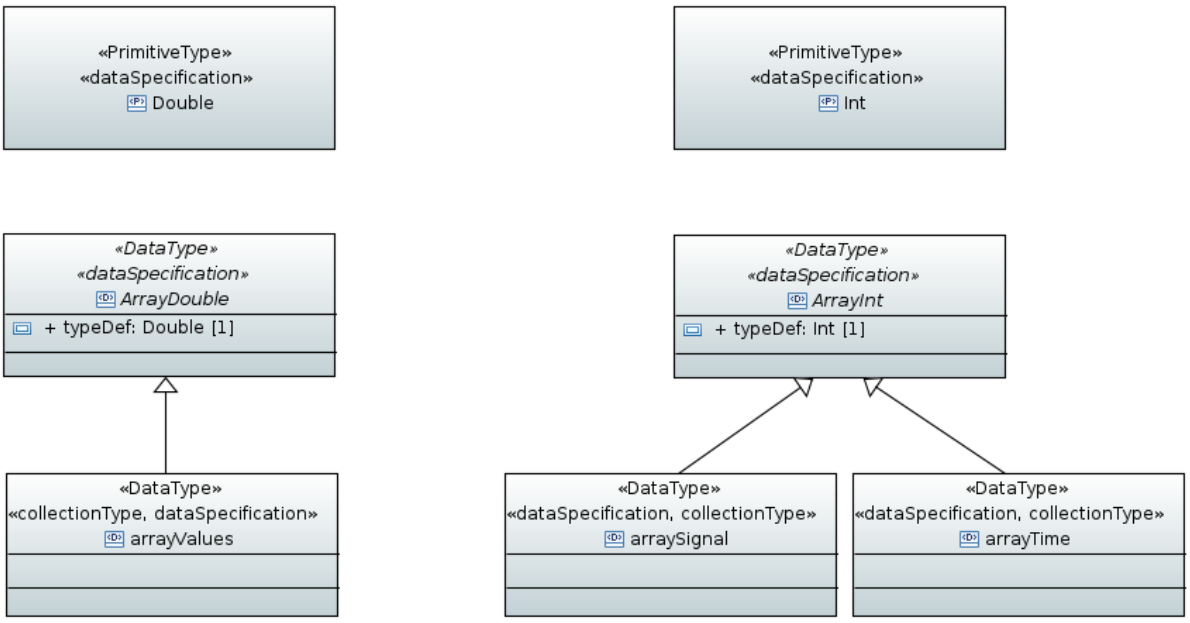


Figura A.1: Tipos de Datos

Anexo B

Vista de Funcionalidad (Functional View)

B.1. Ficheros

Figura B.1: Ficheros de Funcionalidad

B.2. Interfaces

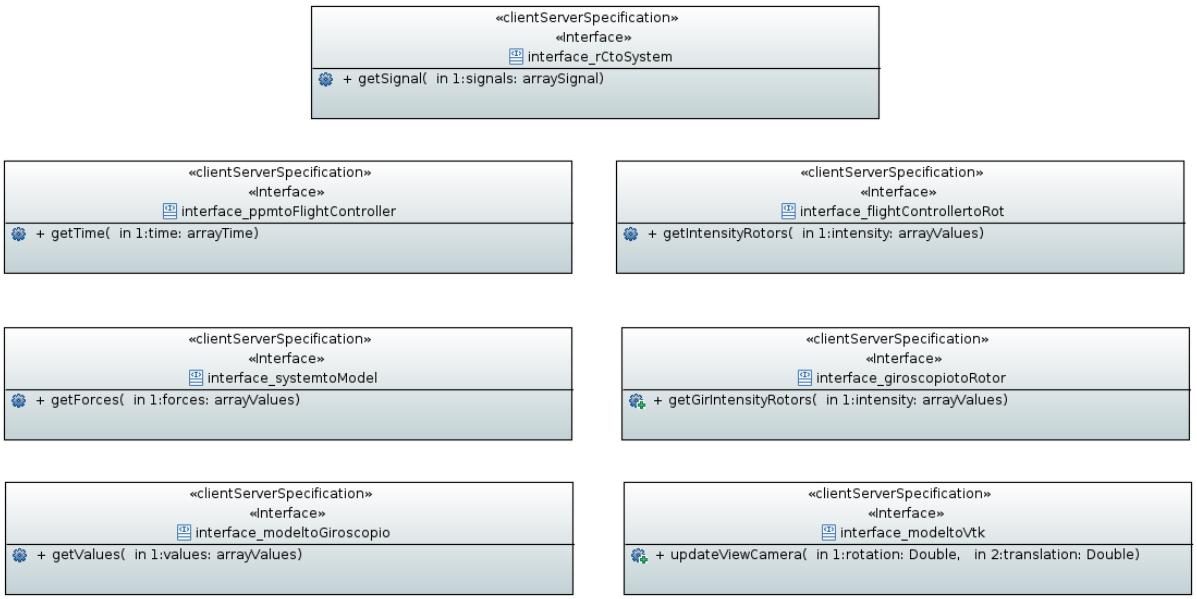


Figura B.2: Interfaces

Anexo C

Vista de Comunicación (Communication View)

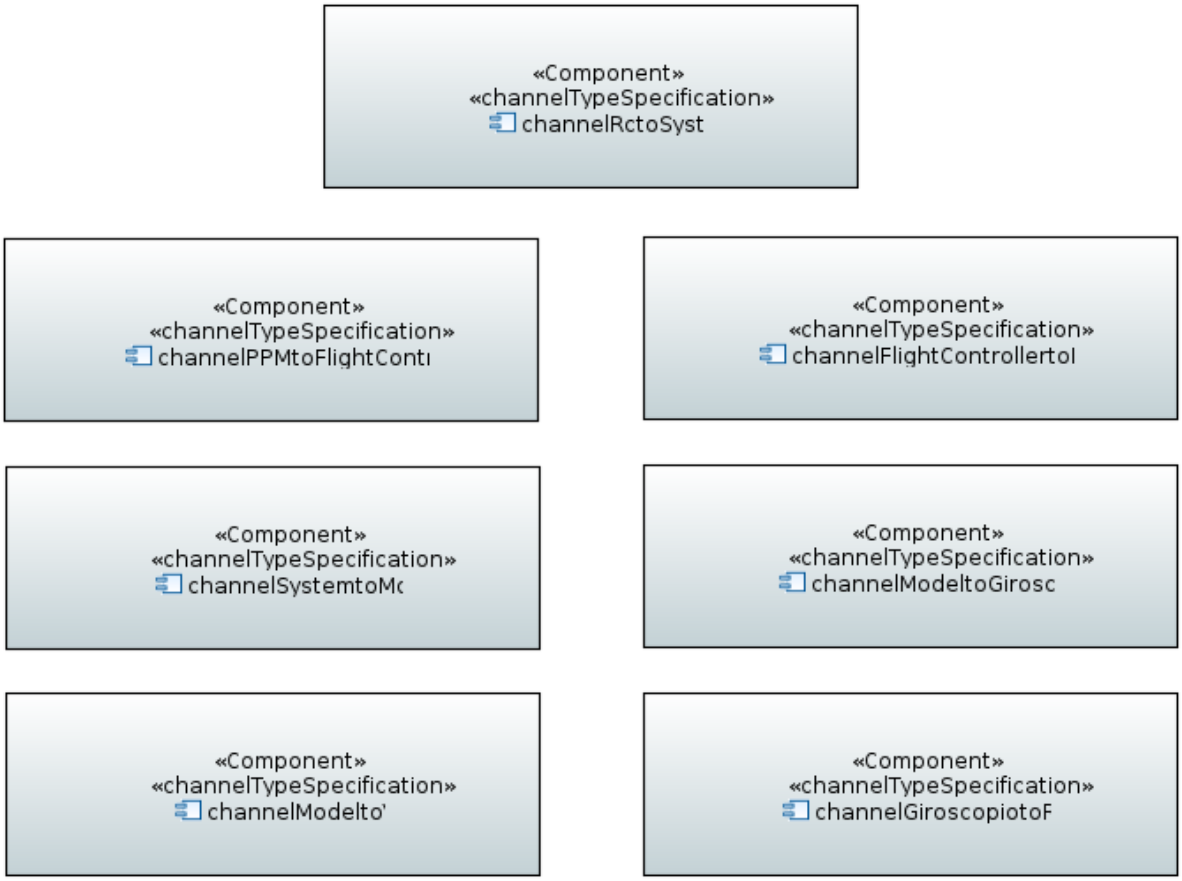


Figura C.1: Canales de Comunicación

Anexo D

Vista de Aplicación (Application View)

D.1. Sistema

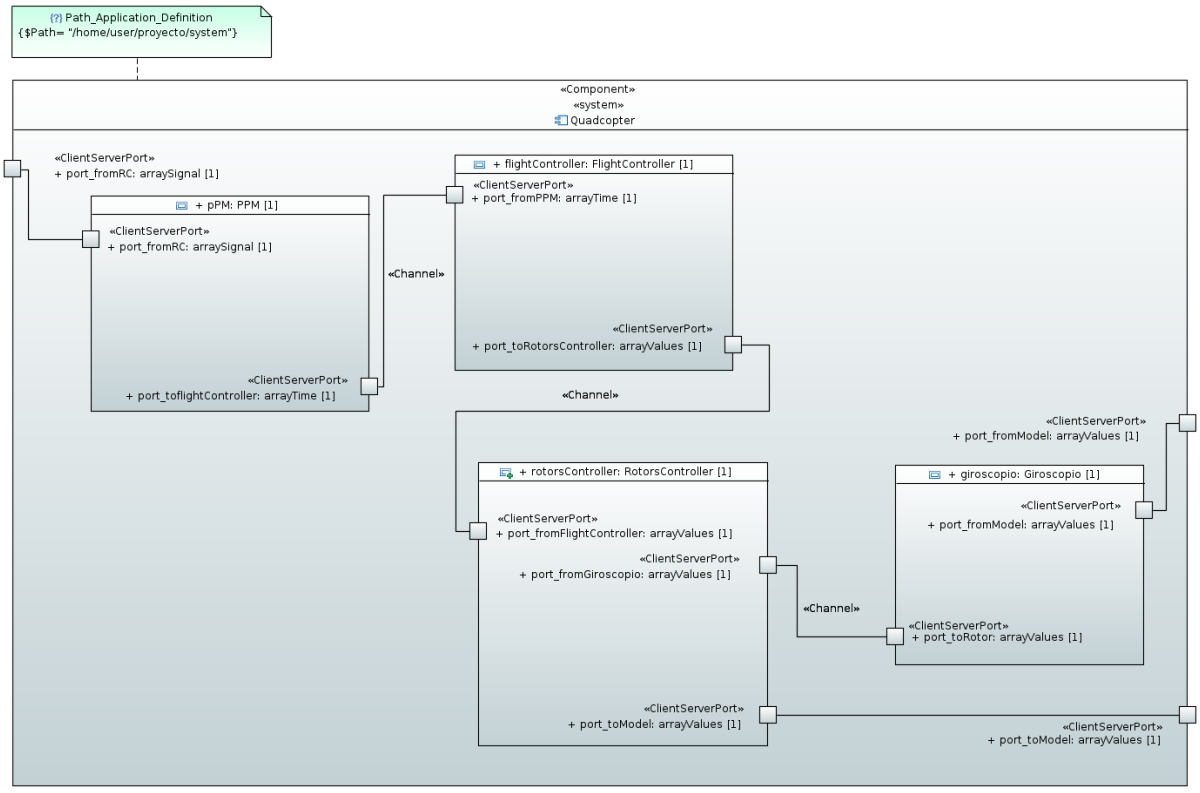


Figura D.1: Estructura del Sistema

D.2. Componentes del Sistema

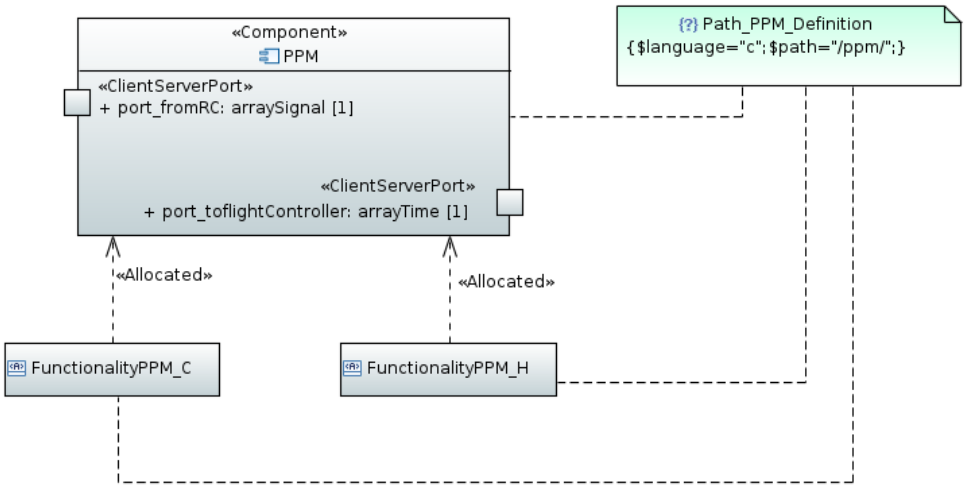


Figura D.2: Decodificador PPM

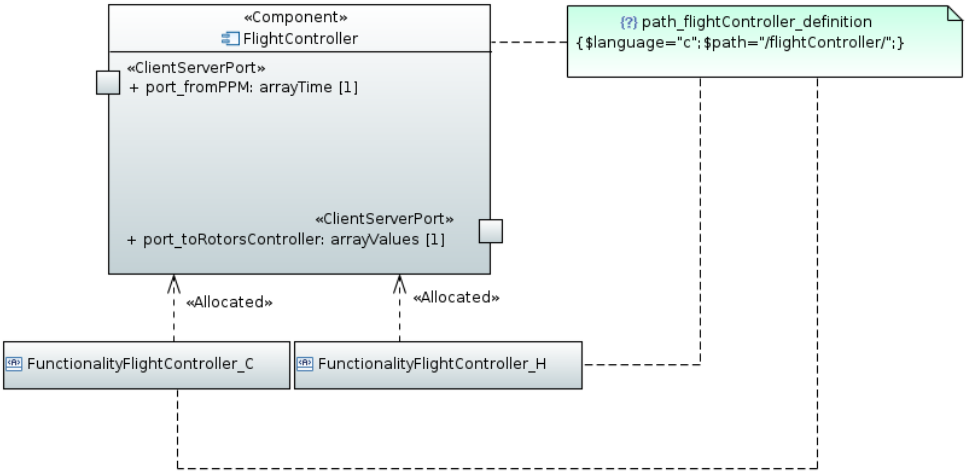


Figura D.3: Control de Vuelo

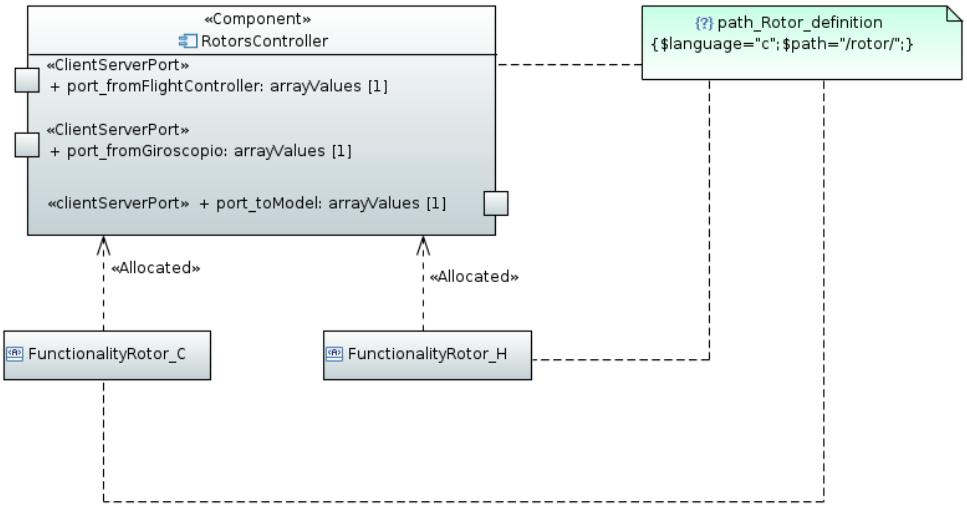


Figura D.4: `Control de los Motores`

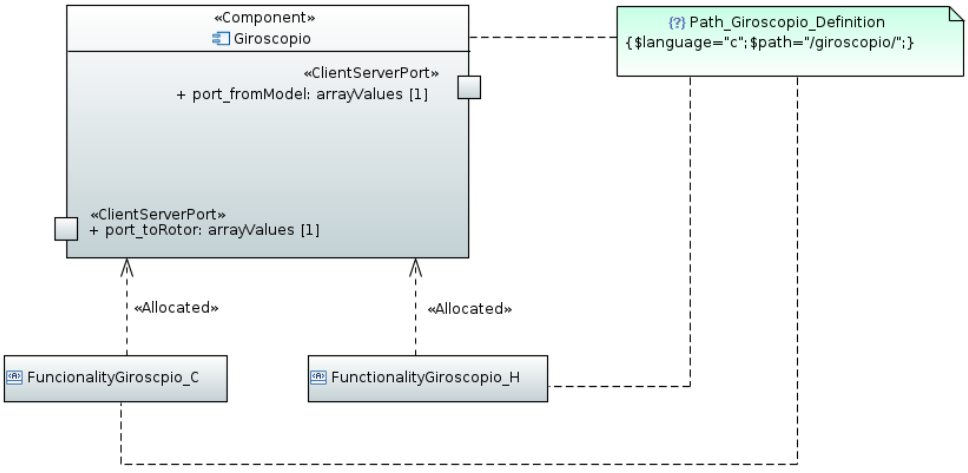


Figura D.5: `Giroscopio`

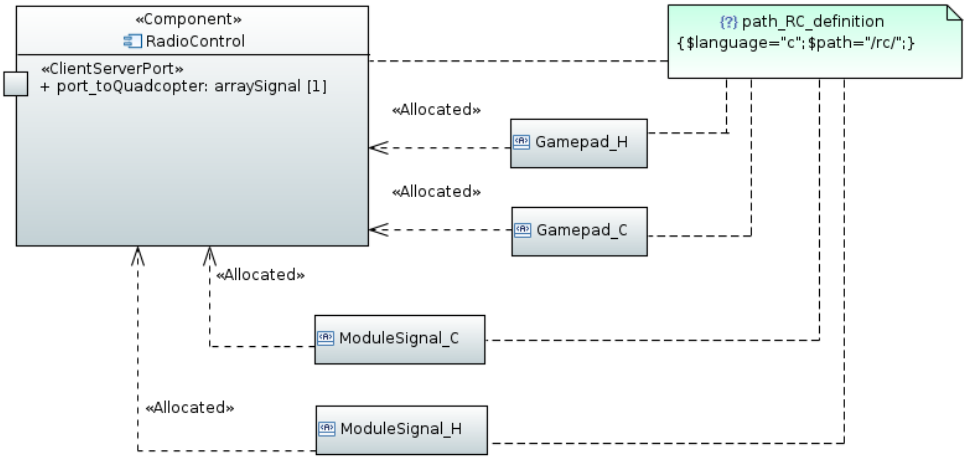


Figura D.6: `Radio Control`

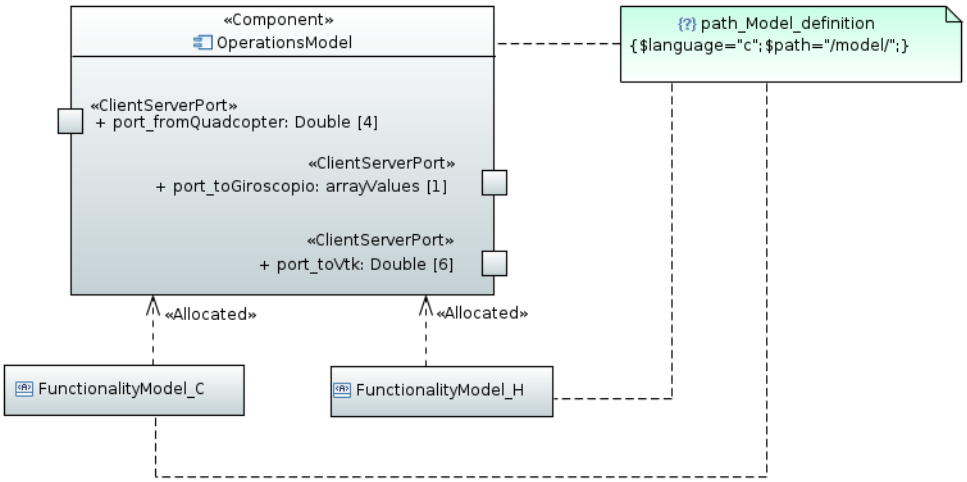


Figura D.7: `Modelo Físico`

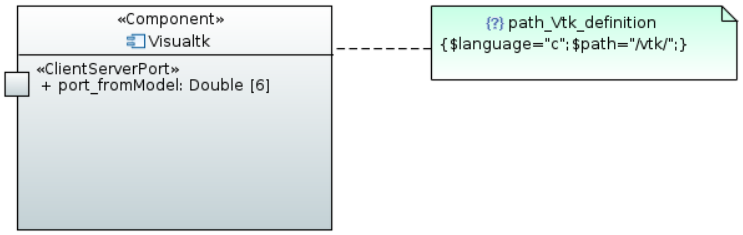


Figura D.8: `VisualTk`

D.3. Asociación Ficheros-Componentes

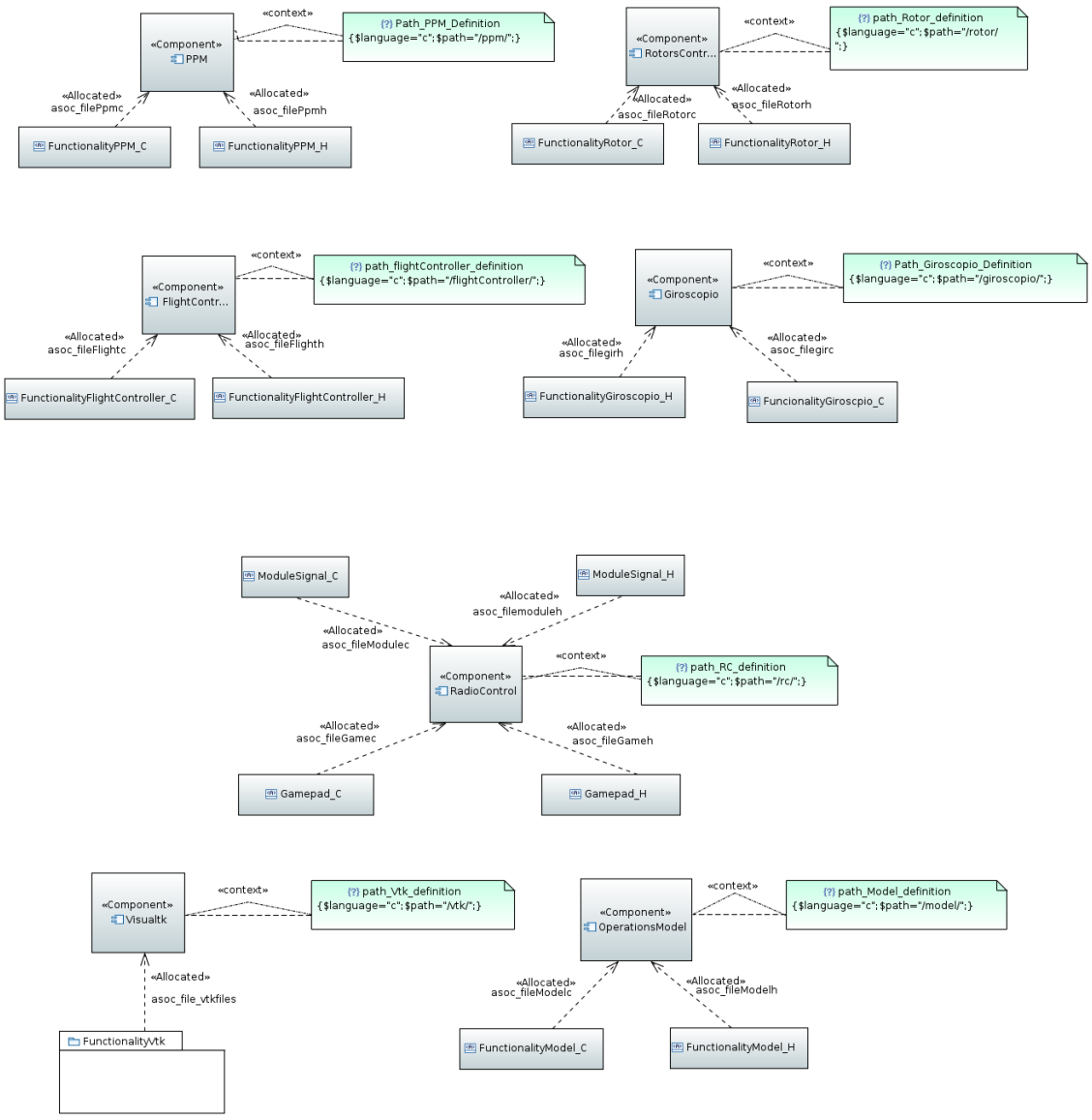


Figura D.9: Asociación Ficheros-Componentes

Anexo E

Vista del Espacio de Memoria (Memory Space View)

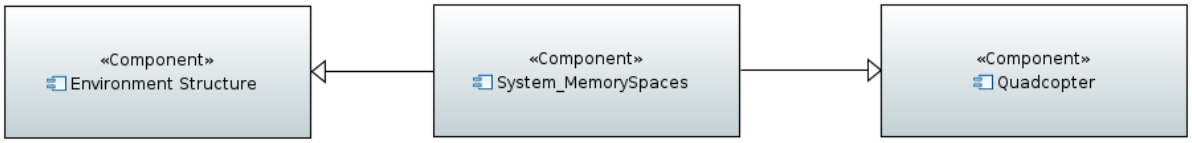


Figura E.1: Espacio de Memoria: Generalización

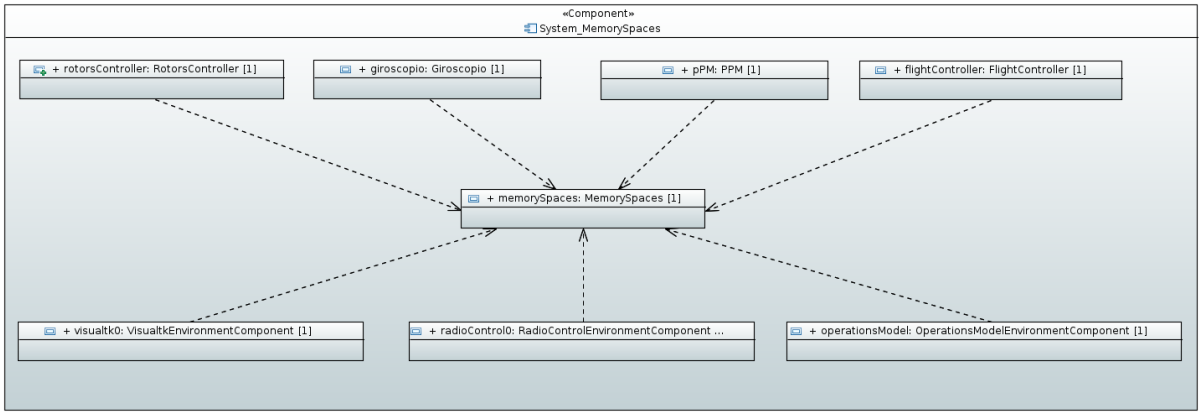


Figura E.2: Espacio de Memoria: Componentes

Anexo F

Vista de Verificación (Verification View)

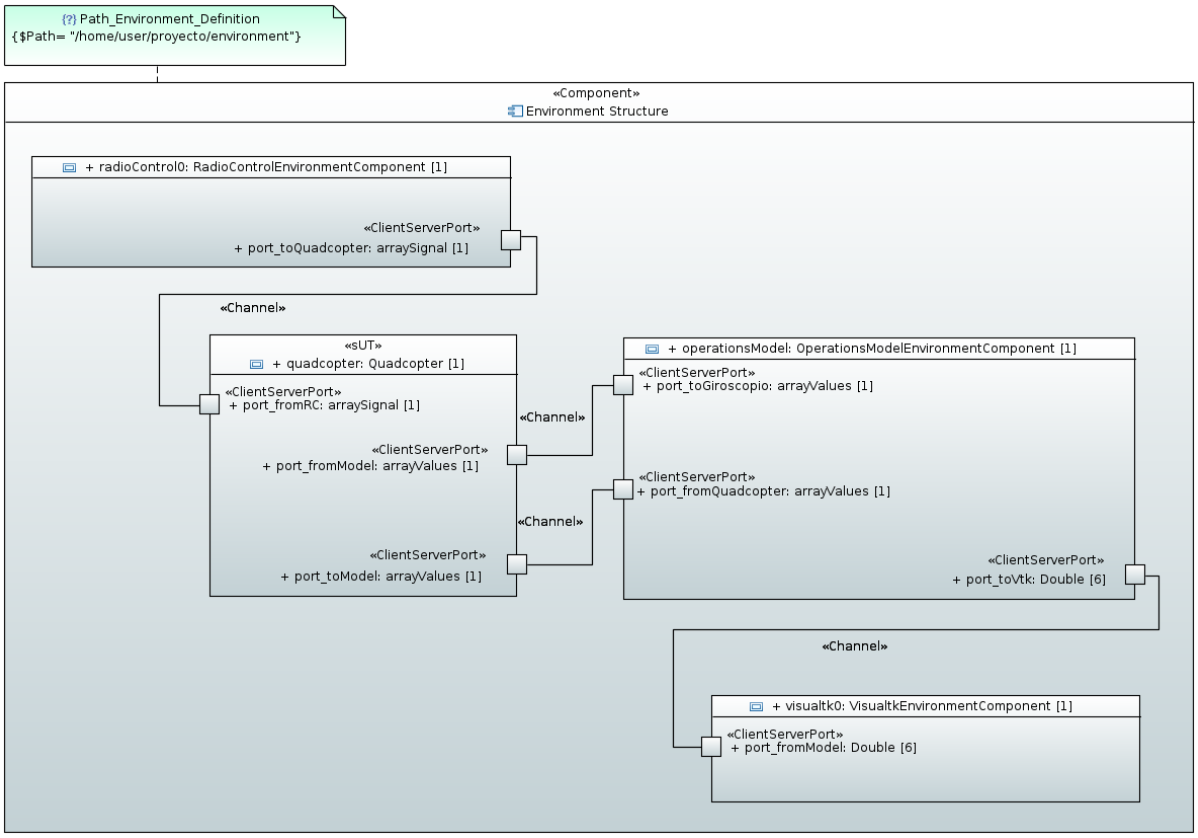
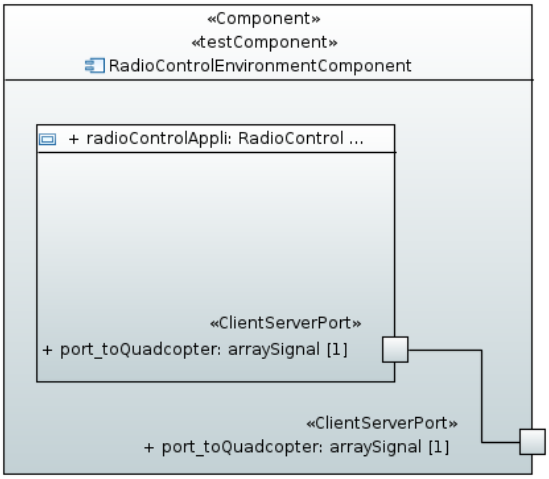
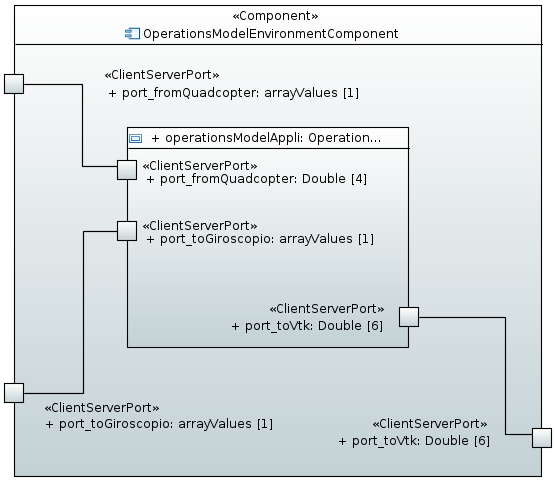


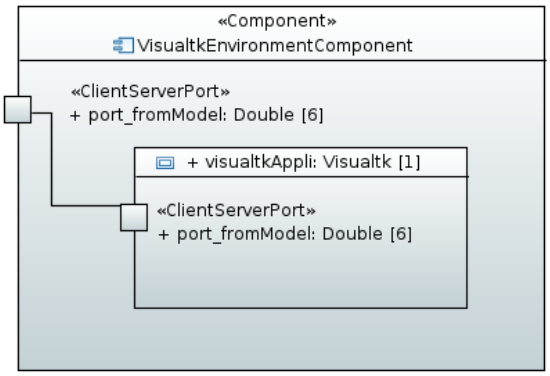
Figura F.1: Estructura del Entorno



(a) Radio Control



(b) Modelo de Operaciones



(c) Visual TK

Figura F.2: `Instancias del Entorno`