Capítulo 2

Estudio de los Sistemas relativos al Proyecto

2.1 Introducción

En el presente capítulo veremos una descripción de cada uno de los sistemas que intervienen en nuestro proyecto. La intención es la de presentarlos todos, aunque en los siguientes capítulos se profundice más en la complejidad de algunos de ellos. El análisis software del RIL, por ejemplo, se hará en el siguiente capítulo, aunque será inevitable exponer ya aquí algunos conceptos a ese respecto. Habrá sistemas que ya existían antes y otros que crearemos para este proyecto y, a su vez, habrá sistemas reales y sistemas que en realidad simulan a sistemas reales.

· Sistemas reales:

2.2 Sistema Helicóptero

Analicemos uno de nuestros helicópteros como sistema: tenemos actuadores, que implementan las entradas controlables, y sensores, que nos permiten leer algunas de sus salidas.

Figura 2.2-1: Bloque Sistema Helicóptero

2.2.1 Entradas controlables: Actuadores

No consideraremos por simplicidad que existen otras entradas no controlables, como por ejemplo el viento o la densidad del aire.

Colectivo

Colectivo y cíclico se basan en un mismo principio y utilizan prácticamente un mismo mecanismo, con finalidades eso sí diferentes. Ahora veremos brevemente el colectivo, el cíclico se explica en el apartado a continuación; pero antes veamos en qué se fundamentan ambos actuadores. De la Dinámica de Fluidos sabemos que la sustentación de un perfil aerodinámico depende del ángulo de ataque α, es decir, del ángulo entre la dirección de movimiento del fluido y el eje del perfil. De hecho, la evolución es prácticamente lineal para pequeños ángulos ($|\alpha|<10^{\circ}$) y siempre que no se llegue al punto de desprendimiento, donde se comienza a perder sustentación en favor de una mucho mayor resistencia. En el caso que nos ocupa, el perfil es el de las palas del rotor principal, el movimiento relativo con el fluido lo da el giro de las mismas accionadas por el motor y el fluido en cuestión es el aire (ver figura de apoyo en Anexo IV, Fig IV.1). Tanto colectivo como cíclico variarán el ángulo de ataque de las palas, aunque de forma como veremos distinta. El mecanismo que permite esto es un conjunto de varillas que giran con las palas. La altura de dichas varillas respecto a las palas determinan el ángulo de ataque de estas últimas, que en este caso (rotor principal) de ser positivo incrementa la sustentación y de ser negativo genera un empuje hacia abajo. Por último la altura de las varillas depende de la posición y el movimiento de un plato anular sobre el que se apoyan y deslizan (ver Figura 2.2-2).

El colectivo mueve el plato hacia arriba o hacia abajo, con lo que el ángulo de ataque de las palas será mayor o menor, pero el mismo en todo su giro, por lo que el efecto será una variación del empuje vertical y por tanto de la altura.

Cíclico

El cíclico sin embargo gira el plato inclinándolo, desalineando el eje del mismo respecto al de rotación de las palas, con lo que la altura de las varillas varía durante el giro. Por lo tanto, el ángulo de ataque de la pala será también función de la posición instantánea de la misma, con lo que la sustentación cambia a lo largo del giro descrito. El resultado es que el empuje deja de ser totalmente vertical y el helicóptero se inclina y desplaza horizontalmente. Esto nos permite mover al helicóptero en todas las direcciones del plano horizontal.

Gas (*Throttle*)

También podríamos llamarlo acelerador, con lo que su misión nos quedaría mucho más clara. Permite variar la entrada de aire en el carburador, y por tanto la carga de combustible en la cámara de combustión. En consecuencia variará el régimen del motor y por tanto la potencia transmitida para el de giro de las palas. En un punto de equilibrio, mayor potencia se traduce en mayor velocidad de giro, que a su vez se traduce en mayor empuje. Aquí cabría comentar que el control para nuestro UAV se intentará realizar a revoluciones constantes.

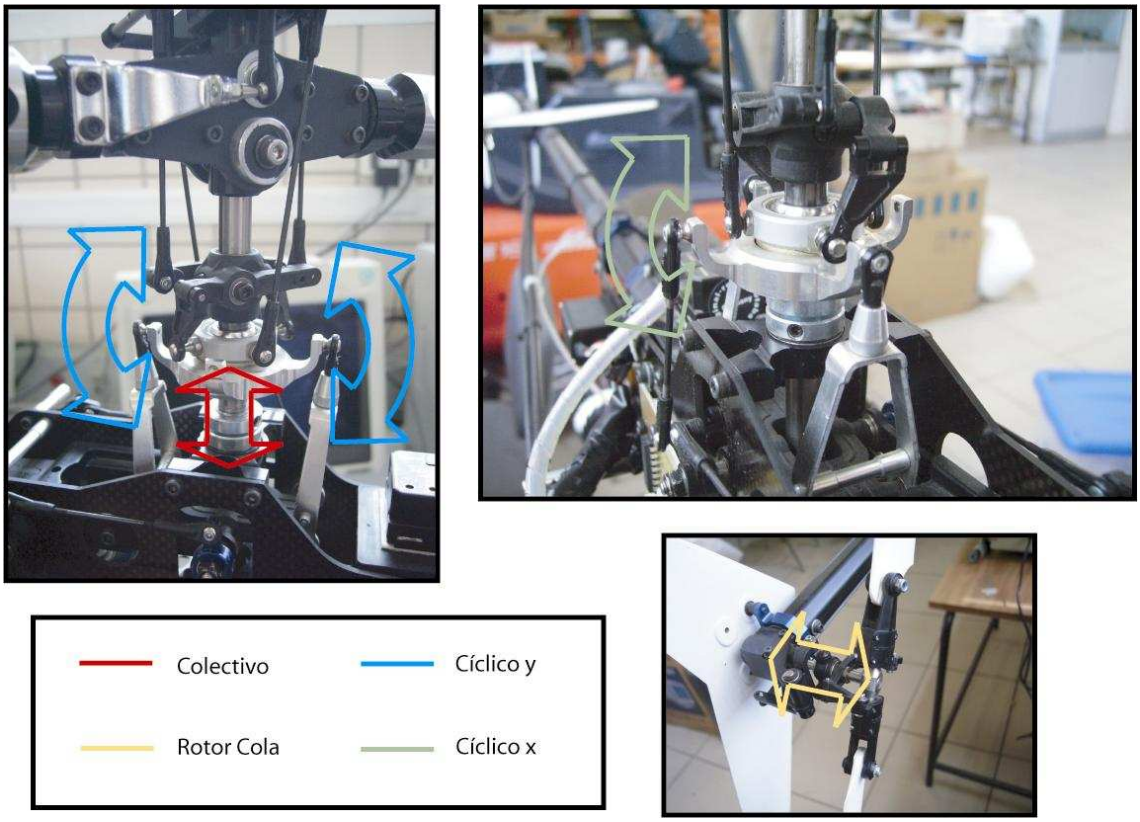

Figura 2.2-2: Detalle mecánica de 4 de los servoactuadores

Rotor de cola

El giro de toda la masa de rotor, eje y palas produce un momento que tiende a hacer que todo el helicóptero entero gire sobre sí mismo junto con esta masa. Para evitar este efecto indeseado se colocan un par de palas más pequeñas en un brazo de palanca que conocemos como cola, cuyo empuje compensa dicho momento. Son accionadas por el mismo motor y su régimen de giro es proporcional al del rotor principal. Por tanto será la variación del ángulo de ataque la que determine el empuje y, además de compensar el momento del principal, permita al rotor de cola girar al helicóptero controladamente respecto al eje de dicho momento. La variación del ángulo se lleva a cabo mediante un mecanismo análogo al ya conocido colectivo, aunque en este caso más simple.

Pan & Tilt

Este actuador no nos proporcionará control sobre el vuelo sino sobre la visión de nuestro robot. Un par de servomotores nos permite orientar la cámara de visión montada, cambiando los ángulos que en inglés se denominan *pan* y *tilt*. En la actualidad nuestro compañero Víctor Manuel Vega está desarrollando, entre otras cosas, una estructura para dicho mecanismo en fibra de carbono, que montará una cámara de video digital de pequeñas dimensiones.

2.2.2 Salidas medidas: Sensores

Altímetro barométrico

La variación de la presión atmosférica con la altitud le permite a este aparato darnos una estimación de la altura. Aunque su precisión es menor que la del GPS que veremos un poco más adelante, sí es más fiable ya que no depende del nivel de señal de los satélites.

Giróscopo

Usando el Principio del Giróscopo, este aparato calcula el ángulo girado en torno a un eje. En realidad cualquier helicóptero de radiocontrol debe traer incorporado al menos uno, que permite corregir automáticamente el ataque de las palas del rotor de cola en caso de un giro en torno al eje del rotor principal.

GPS

El GPS (*Global Position System*) es un elemento que cada vez nos es menos extraño y que se usa ya en todo tipo de vehículos. Los GPS usan señales de satélites en órbita para triangular la posición propia. Para el UAV usaremos un GPS diferencial que nos permite conocer la posición con una precisión de centímetros. Esta posición viene expresada en formato de latitud, longitud y altura, que como veremos es también el elegido para conformar los *waypoints*.

IMU

La *Inertial Measurement Unit* utiliza giróscopos, acelerómetros y lecturas del campo magnético para poder facilitarnos información sobre orientación, aceleraciones y velocidades en los tres ejes del espacio cartesiano. Existen aparatos comerciales, pero también fuimos testigos de la realización de un modelo propio experimental por parte de dos de nuestros compañeros de laboratorio. Dicho modelo está operativo y podría incorporarse en cualquier momento a la electrónica ya existente.

Medidores de niveles

Con el fin de medir los niveles de combustible y baterías se deben incorporar los sensores adecuados. A pesar de que no usaremos sus medidas en el control de vuelo, es fácil entender que éstas son críticas para el correcto funcionamiento.

Sónar

El sónar se vale del fenómeno de reflexión del sonido para poder conocer la distancia respecto a un objeto. En el caso del helicóptero el objeto cuya distancia más nos preocupa es el suelo, por lo que el sónar se colocará en la parte baja y enfocado hacia él. Su medida será especialmente útil en las delicadas maniobras de despegue y aterrizaje.

2.2.3 El Problema de Control

El control del sistema helicóptero con las variables que acabamos de ver es un problema altamente acoplado e inestable. La inestabilidad es intrínseca al sistema y además no es raro que se presenten perturbaciones de todo tipo: vientos variables, cambios de densidad del aire, fluctuaciones del régimen de giro del motor... Por otro lado, el grado de acoplamiento es fácil de ver con un ejemplo: imaginemos que desde

una posición de *hover*, es decir, flotando en el aire pero sin movernos, queremos que el helicóptero de desplace hacia adelante. Para ello variamos el cíclico volcando el plato hacia adelante, con lo que el ángulo de ataque y por tanto el empuje es mayor en la sección posterior de giro. Esto hace que el helicóptero se vuelque ligeramente hacia adelante y comience a trasladarse en esa dirección. Pero la fuerza que lo empuja adelante se le ha "robado" a la sustentación, por lo que si queremos que no comience a caer tenemos que además aumentar el cíclico. Todo esta alteración de la geometría del movimiento de las palas del rotor principal hace que cambie el momento generado por las mismas, con lo que tenemos que actuar también sobre el ataque de las palas del rotor de cola. Además el motor se vendría abajo por todo el esfuerzo extra de empujar hacia adelante, con lo que para mantener el régimen de giro habría que dar más gas. En el caso de estar visionando un objeto fijo en el suelo habría también que variar *pan* y *tilt* para no perderlo de vista... y ya no nos quedan actuadores por utilizar, todo para un simple movimiento hacia adelante.

Todo esto no es más que una pequeña introducción al sistema real del helicóptero. En el capítulo 4 estudiaremos un modelo, simple pero bastante realista, que de este sistema hemos utilizado en el simulador. Para comprenderlo mejor, en dicho capítulo se explica con más detalle parte de la física de vuelo. Por lo tanto, para algo más de información sobre los principios del vuelo real, dirigirse a él.

2.3 Sistemas de Coordenadas

Aunque no se traten de sistemas tangibles o software, analizaremos ahora los distintos sistemas de referencia que utilizaremos en el proyecto para describir el estado cinemático del helicóptero. Lo que nos interesa en este momento es obtener los conceptos, y no tanto conocer las unidades en las que se expresan en concreto cada una de las magnitudes. Eso lo veremos cuando sea necesario.

2.3.1 Sistemas Cartesianos Locales

Robots terrestres:

De aplicaciones anteriores de la arquitectura general, en concreto para el vehículo terrestre ROMEO, nos viene definido un sistema de referencia bidimensional.

El eje y se encuentra orientado hacia el Norte (N) y el x hacia el Este (E), por lo tanto la orientación respecto al Norte se define a partir de y, positiva en sentido antihorario y acotada entre $[-\pi, \pi]$. El estado cinemático vendrá entonces determinado por un par (x, y) que define la posición, un valor para la orientación y un par (v_x, v_y) para la velocidad instantánea.

Esto que es perfectamente válido para robots terrestres, deja de serlo para el caso de nuestro helicóptero, ya que nuestro movimiento deja de ser plano. Además, los ejes de esta forma definidos son totalmente distintos a los que definiremos a continuación para el HERO.

Este hecho no tiene en realidad ninguna importancia, ya que en ambos casos estamos hablando de sistemas de referencia locales. Finalmente cualquier posición de cualquiera de los robots, deberá ser transformada según el sistema de referencia que utiliza el Control Center, y que veremos en último lugar. De la misma manera, los *waypoints* nos vendrán expresados en coordenadas del Control Center, con lo que habrá que adaptarlos al sistema local.

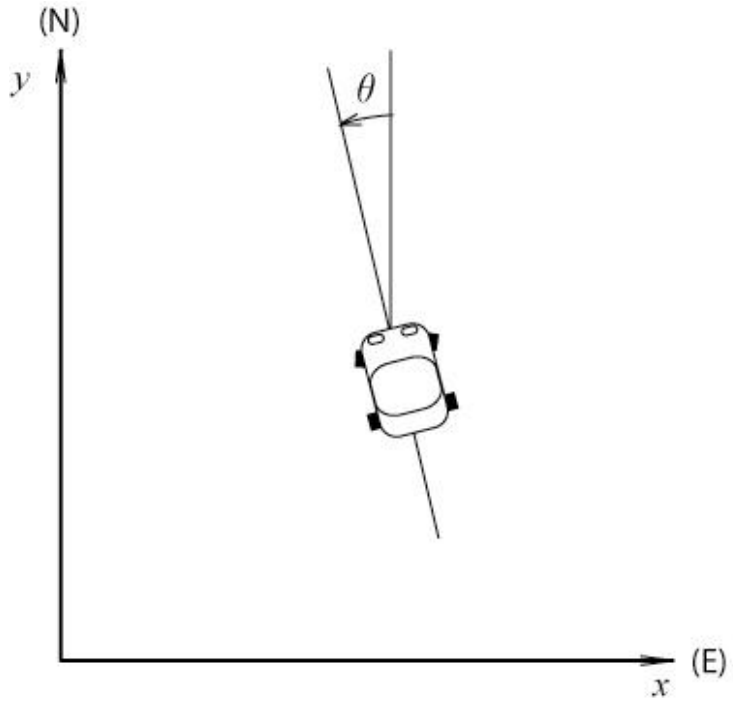

Figura 2.3-1: Sistema de referencia para robots terrestres

Modelo Simulación HERO:

En el HERO el caso se complica. El sistema que pasamos a explicar brevemente es el que utilizaremos en el simulador, por lo que lo volveremos a ver en el capítulo correspondiente a la simulación, en el que se harán algunas simplificaciones.

Realmente pasamos a tener dos sistemas de referencia:

- *World Coordinate System* (WCS): Es el sistema que tomaremos como fijo a tierra, con *x* orientado hacia el norte (N), *y* hacia el oeste (W) y *z* hacia el cielo. Su origen puede ser cualquier punto conocido, como por ejemplo la posición de la estación de integración del GPS.

- *Helicopter Coordinate System* (HCS): Es un sistema solidario al helicóptero, con el eje *x* orientado hacia la parte delantera del vehículo, *z* hacia la superior e *y* hacia la izquierda (para responder a un sistema dextrógiro). De esta forma, los ángulos *roll*, *pitch* y *yaw* corresponden a giros en torno a *x*, *y*, *z* respectivamente.

En todos los casos, los ángulos se definirán positivos según la "regla del sacacorchos", es decir, en sentido antihorario si el eje en cuestión apunta hacia el observador.

La posición del helicóptero será la de su sistema de referencia relativa al WCS. Por lo tanto el cambio de coordenadas de las magnitudes que definen el estado pasa por una transformación que supone traslaciones y giros en todos los ejes. Gracias al las simplificaciones que veremos y justificaremos en el capítulo 4, estas transformaciones serán mucho más sencillas de lo que en un principio nos podríamos imaginar. Por ejemplo, una magnitud tan importante como la orientación respecto al norte, vendrá determinada simplemente por el ángulo girado respecto al eje *z*.

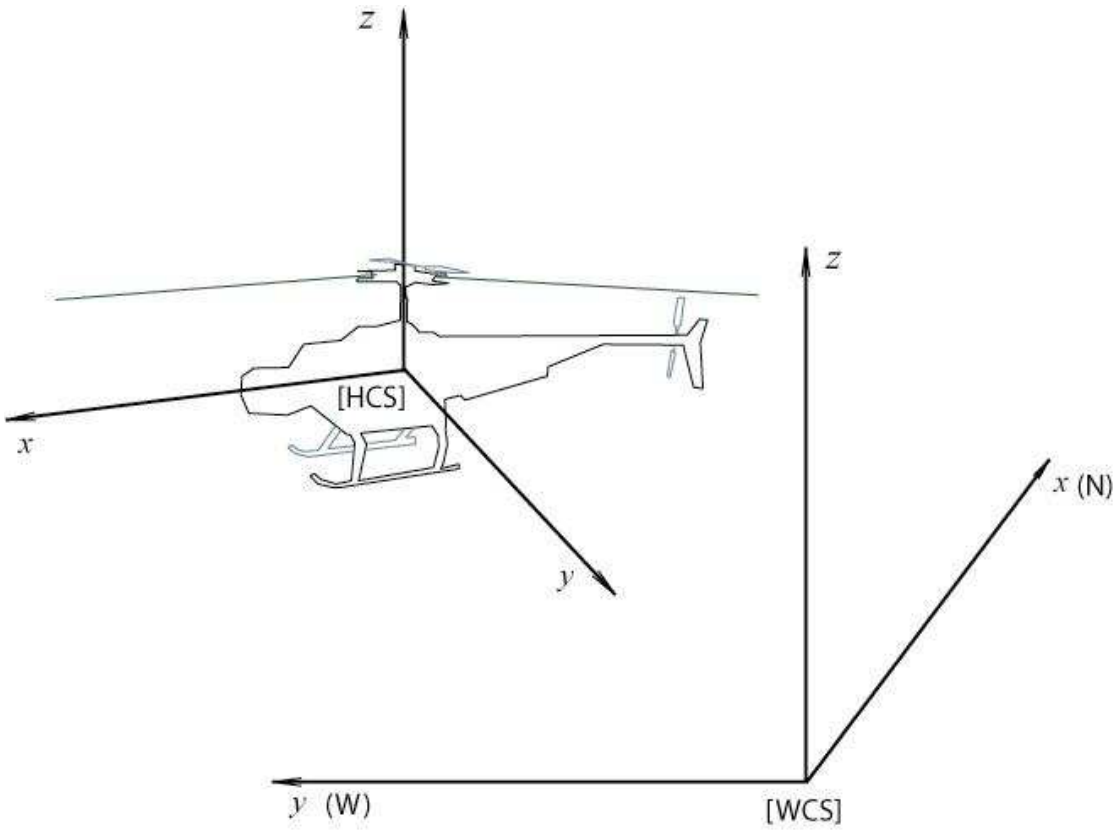

Figura 2.3-2: Sistemas de referencia para el HERO

2.3.2 Sistema Geográfico

El sistema de coordenadas geográficas o terrestres define la posición de cualquier punto *P* sobre la superficie de la Tierra en función de su latitud y su longitud, como veremos a continuación. Para entender la forma en que lo hace, es necesario conocer previamente el concepto de meridiano. Si además agregamos a estas dos magnitudes la altitud, tendremos capacidad de describir la posición de cualquier punto del espacio terrestre y, curiosamente, no terrestre.

Meridiano

Es el círculo máximo que pasa los polos terrestres (*Pn* Polo Norte, *Ps* Polo Sur). De los infinitos meridianos, el meridiano de *P* será aquel que además contiene a dicho punto. Este meridiano puede dividirse en dos mitades determinadas por el eje que une los polos. El meridiano superior es aquella mitad que sigue conteniendo a *P*, mientras que a la otra mitad se le conoce como meridiano inferior.

Latitud

Es el arco de meridiano descrito desde el Ecuador al punto *P*. Se suele representar por medio de la letra *f*. El valor absoluto de la latitud siempre es menor a 90º, y su signo se puede considerar positivo si es latitud Norte (*P* se encuentra en el Hemisferio Norte) y negativo si es latitud Sur (cuando *P* está en el Hemisferio Sur). El lugar geométrico de los puntos de igual latitud es lo que conocemos como paralelo.

Longitud

Es el arco de Ecuador desde el meridiano superior de Greenwich (punto *G* en la Figura 2.3-3) hasta el meridiano superior de *P*. Su valor absoluto es siempre menor a 180º, y se llama longitud Oeste (W) cuando *P* queda al oeste de Greenwich y longitud Este (E) de lo contrario. En nuestros cálculos, a las longitudes Este se les dará signo positivo y a las longitudes Oeste signo negativo. Podemos decir que los meridianos son los lugares geométricos de los puntos que tienen la misma longitud. Esta magnitud se representa por la letra *L*.

Altitud

Representa la tercera dimensión, igualmente necesaria para determinar la situación en el espacio de nuestro punto *P*. El único consenso al que hay que llegar es la referencia que se tomará como altitud cero. Lo más común es tomar como nula la altura a nivel del mar. Ya que en cualquier caso coincidirá con la magnitud *z* de los ejes cartesianos que vimos anteriormente, podemos designar la altitud con esta misma letra sin ningún problema.

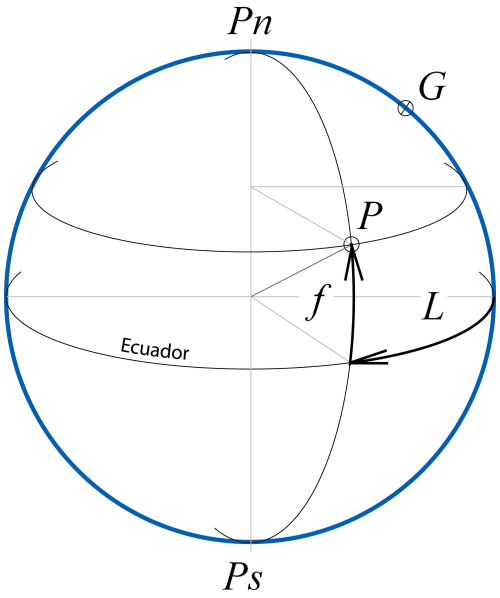

Figura 2.3-3: Latitud y longitud de un genérico punto *P* sobre la superficie terrestre

Conociendo las coordenadas geográficas (*f*, *L*) podemos situar el punto donde nos encontramos en la superficie terrestre. Con la altitud *z*, somos además capaces de situarlo en el espacio. Por ejemplo, un punto de la ciudad de Sevilla se encuentra aproximadamente a 37.23º N, 5.59º W y unos pocos de metros por encima del nivel del mar. Por supuesto, para la descripción de la posición de nuestro robot necesitaremos más precisión, por lo que latitud y longitud vendrán expresadas con resolución de hasta diezmillonésimas de grado. Esta información será facilitada por el GPS, que utiliza este mismo sistema de coordenadas.

2.3.3 Sistema Control Center

Poco antes de la finalización del presente proyecto, se llegó a un nuevo consenso para el sistema de referencia general de la arquitectura. Respecto a él vendrá definido el estado de los robots y en este sistema se expresarán los puntos de paso (*waypoints*). En resumen, será el que utiliza el Control Center; podemos verlo en la Figura 2.3-4.

Se trata de un sistema de ejes cartesianos que realiza una serie de suposiciones sobre el sistema geográfico que acabamos de ver. Supondrá que el arco descrito en un cambio de latitud (o de longitud) es una recta, y por lo tanto que puntos de igual altitud conforman un plano: es decir, supondrá que la Tierra es localmente plana. Según esto, para pasar de ángulos de latitud y longitud a distancias en el plano, bastará realizar unas sencillas operaciones que veremos en seguida. Conforme a todo esto, se definen los ejes: *y* hacia el Norte, *x* al Este y *z* hacia el cielo. Si lo analizamos, es como si le hubiéramos añadido la tercera dimensión al sistema local de los robots terrestres.

El origen del sistema ($x = 0$, $y = 0$, $z = 0$) se encuentra en algún lugar del Golfo de Guinea, pero eso no nos afecta en absoluto, ya que nosotros trabajaremos con incrementos.

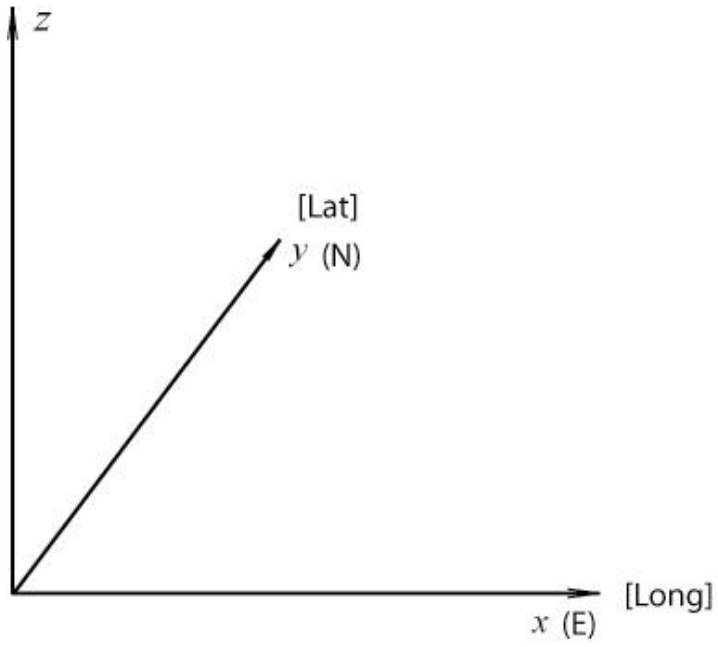

Figura 2.3-4: Sistemas de coordenadas del Control Center

Suponiendo que la Tierra es una esfera perfecta de radio R_{Veq} (radio de la esfera de volumen equivalente al de la Tierra), y que cos(f) es constante en nuestros trayectos (trayectos cortos), si queremos tener una unidad lineal en la que expresar la posición en el plano horizontal, las operaciones a realizar sobre los ángulos f y L (en radianes) son:

$$\begin{cases} \Delta x = R_{Veq} \cdot \cos(f) \cdot \Delta L \\ \Delta y = R_{Veq} \cdot \Delta f \end{cases}$$

que, bajo todas las suposiciones anteriores, resultan en simples escalados. El cos(f), que se introduce para tener en cuenta la disminución del radio del paralelo (disco paralelo al Ecuador que contiene al P) con latitudes que se alejan de 0º, se calculará sólo una vez (por ejemplo en la posición inicial). El otro dato será siempre una constante para todos los puntos de la tierra: $R_{Veq} = 6371 \cdot 10^3$ m. Sobre las conversiones de unidades seguiremos hablando en el capítulo 4.

Todas estas suposiciones, dada la curvatura de la tierra y para los trayectos que cubrirán nuestros robots, son admisibles.

La orientación respecto al Norte, vuelve a tener el significado que ya vimos para el robot terrestre, y su expresión matemática es:

$$\theta = -\arctan\frac{\Delta x}{\Delta y}$$

Particularmente interesante para este proyecto nos resulta la relación de este sistema respecto al WCS que definimos como sistema local del simulador del HERO. Si hacemos coincidir sus orígenes, se tienen las siguientes relaciones:

$$\left\langle \begin{array}{l} x^{\{CC\}} = -y^{\{WCS\}} \\ y^{\{CC\}} = x^{\{WCS\}} \\ z^{\{CC\}} = z^{\{WCS\}} \end{array} \right\rangle$$

Por lo que:

$$\theta = \arctan\frac{\Delta y^{\{WCS\}}}{\Delta x^{\{WCS\}}}$$

En definitiva y por el momento, al introducir las coordenadas de un punto en el Control Center, estamos indicando:

x:	Longitud en grados, positiva hacia el Este.
y:	Latitud en grados, positiva hacia el Norte.
z:	Altitud en metros, desde el nivel del mar.

2.4 Sistema DSP

Toda la responsabilidad de control en nuestro UAV cae finalmente sobre un DSP, que supone el controlador de bajo nivel. Es él el que recibe directamente la información de los distintos sensores y es él el que envía las señales de control a los servos de los actuadores, haciendo que el vehículo se mueva o enfoque en una dirección concreta la cámara.

Figura 2.4-1: Bloque del DSP

Tanta responsabilidad es quizá excesiva y, sobre todo en la fase de prueba, se cree conveniente la posibilidad de conmutar el control hacia un piloto humano mediante un conmutador (*switch*) situado en el mando de radiocontrol. Esto posibilita salvar situaciones en las que el modo automático pierda el control por alguna circunstancia, evitando así posibles colisiones.

El DSP se encuentra ya programado por proyectos anteriores. El tener acceso a ese código fue fundamental para la comprensión de su funcionamiento. No olvidemos que después tuvimos que simular su comportamiento en el DSP Simulator.

Especialmente importante a respecto del funcionamiento es el concepto de Puerto Virtual. Los puertos virtuales del DSP son variables internas a éste a las que se referencia por un número. Estas variables pueden ser leídas y escritas desde el PC mediante el uso de funciones pertenecientes al protocolo de comunicaciones entre DSP y PC, que veremos cuando estudiemos el software. Los puertos virtuales se usan para ordenar al DSP que haga ciertas acciones o para enviarle algún dato. A su vez, el DSP puede comunicar al PC cosas a través de estos puertos virtuales si éste último hace lecturas periódicas de estos puertos (*polling*). La numeración de los puertos virtuales relacionados con las comunicaciones con el PC, junto con una pequeña descripción de los mismos, se encuentra en el documento "*Puertos virtuales del DSP asociados al protocolo de comunicación con el PC*".

Las comunicaciones entre DSP y PC se realizarán a través de puerto serie. Este PC que de esta forma se comunica con el DSP será conocido como PC de control, distinguiéndose así del PC de visión o del de tierra. El resto de comunicaciones se llevan a cabo según una arquitectura hardware que podemos ver más claramente en el esquema de la Figura 2.4-2.

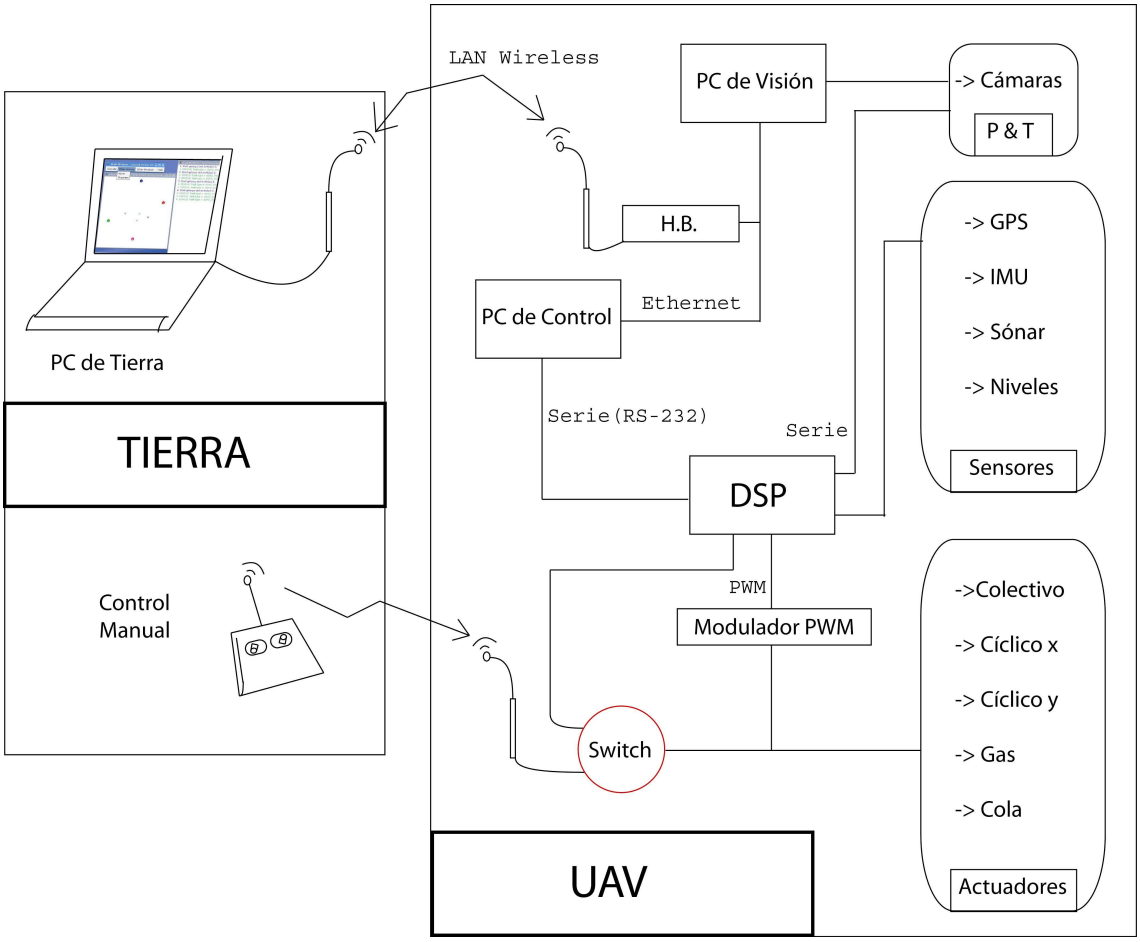

Figura 2.4-2: Esquema hardware

El Control Center (CC) en el PC de tierra, único interfaz con el usuario, envía un mensaje al UAV. Este mensaje puede traducirse en una orden de control o de visión. El mensaje se transmitirá por LAN llegando al PC de visión o al de control, según el tipo de orden. Las tareas de visión quedan fuera de los objetivos del presente proyecto, y de hecho representan un campo del proyecto HERO que en la actualidad tiene a varias personas especializadas trabajando en su desarrollo.

Cuando una tarea de control llega al PC de control, se descompone en subtareas elementales que se envían al DSP por puerto serie, como ya sabemos. Las órdenes al PC de control pueden llegar no sólo del PC de tierra, sino también del PC de visión: por ejemplo, éste último puede enviar una serie de *waypoints* para el seguimiento de un objetivo móvil. El DSP realizará las tareas asignadas y devolverá acuses de recibo cuando corresponda (los conoceremos como respuestas *ack*), así como posibles errores (`DSPError`) y mensajes con la telemetría del vehículo.

A nivel software, es el RIL el que se comunica con el DSP según el esquema software de la Figura 2.4-3:

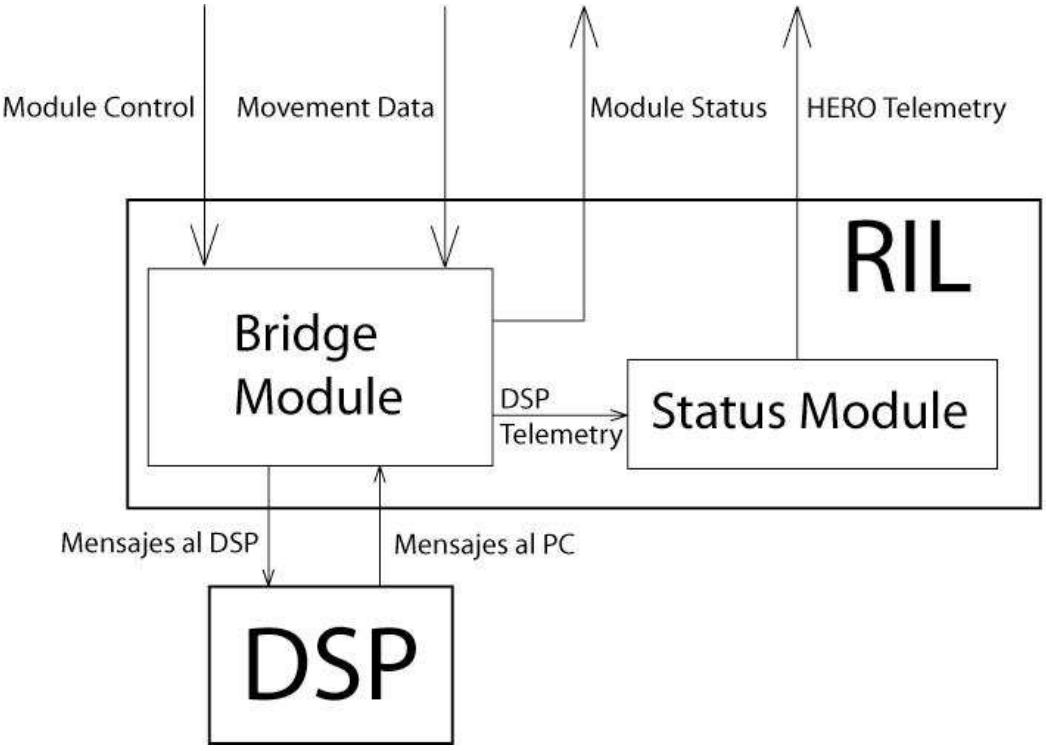

Figura 2.4-3: Esquema software

En este esquema se adelanta mucha información que en realidad no veremos hasta llegar al capítulo 3. Vemos que el RIL lo conforman en realidad dos módulos: el Bridge Module y el Status Module. Vemos qué información circula y en qué dirección lo hace: sobre el RIL encontraríamos el resto de la arquitectura distribuida en capas. Vemos también que, dentro del RIL, es el Bridge Module el que realmente se comunica con el DSP. La información entre estos dos elementos, en forma de mensajes, debe circular en ambos sentidos, por lo que tendremos en síntesis dos tipos de mensajes.

2.4.1 Mensajes PC→DSP

Waypoints (WP)

Son puntos a los que se quiere que el robot vaya. En principio se habían pensado dos sistemas de coordenadas distintos, por lo que este tipo de dato podía presentarse en dos formatos distintos. Pero en algún momento se decidió que no se utilizarían las cartesianas (x, y, z) y por tanto sólo consideraremos el caso de las coordenadas geográficas (f, L, z), que se utilizarán de la forma en la que se definieron en el sistema de referencia del Control Center. Nuestra posición será por lo tanto absoluta y en consecuencia podremos además utilizar directamente las medidas del GPS. Los *waypoints* se van almacenando en un buffer que posee el DSP.

Comandos

Más adelante veremos con más detalle el tipo de comandos que podemos enviar al DSP. Ahora los resumimos y vemos que los podemos agrupar en:

- Comandos de lectura o escritura de puerto virtual. Ya hemos visto que el DSP cuenta con una serie de puertos virtuales; en estos se pueden almacenar variables de 16 ó 32 bits, dependiendo del puerto. Estos comandos permiten tener acceso a dichos puertos.

- Limpieza del código de error. Este comando limpia la cola de errores que el DSP enviaría al PC. Así evitamos, por ejemplo, que lleguen al PC errores generados en el arranque del sistema y que no son causa de un mal funcionamiento sino del proceso de arranque en sí.

- Transmisión de estado. A partir de la recepción de éste comando, el DSP comenzará a transmitir el estado del helicóptero. Este estado, como veremos, podrá componerse de más o menos variables según la máscara de estado. Además vendrá expresado en formatos distintos según se transmite, de modo que se generalizará conforme asciende por la jerarquía software.

- Invalidar camino. El DSP borrará los *waypoints* de su buffer y el helicóptero quedará haciendo *hover* (flotando) en la posición en la que estaba cuando se recibió tal comando. Esta circunstancia se dará, por ejemplo, cuando desde el CC se mande abortar una tarea consistente en ir a un *waypoint* o una serie de ellos. No será sin embargo posible, por motivos de seguridad de vuelo, abortar un despegue o un aterrizaje.

- Des/Activación de seguimiento. En función de que el seguimiento esté o no activado, el DSP tomará o no los *waypoints* que encuentre en su buffer de *waypoints*. Este buffer tiene asociado otro tipo de mensajes que veremos en último lugar, el control de flujo de *waypoints*.

2.4.2 Mensajes DSP→PC

Respuestas ack

Cuando al DSP llega un comando, éste responde con un *ack* (de la palabra inglesa *ACKnowledgment*, reconocimiento) que puede interpretarse como un asentimiento y que puede además venir acompañado por algún dato. Este último será el caso si el comando recibido es, por ejemplo, una lectura de puerto virtual.

Estado

El DSP lee los sensores que tiene distribuidos el helicóptero y cuyas medidas podríamos considerar que describen un estado del helicóptero (posición, orientación, velocidad...). Entonces se conforma el `ESTADO_HELICOPTERO` (estructura de datos que veremos en el simulador de DSP) que se empaqueta y manda vía puerto serie al RIL. Dentro del RIL, el Bridge Module desempaqueta el mensaje que pasa a un `ESTADO` (estructura de datos propia del Bridge donde se decodifican todos los mensaje provenientes del DSP) para después ser traducido a `DSP_HERO_TELEMETRY` (estructura de datos que podríamos pensar como una subestructura de la anterior). Como veremos, antes de salir del RIL, el estado sufre otra transformación de formato a

`HERO_TELEMETRY` (estructura de datos característica del HERO) en el Status Module. Ya hemos comentado que asociado al estado tenemos una importante variable de nuestro proyecto, la máscara de estado, que decide qué variables de estado son las que interesan. Lo veremos más adelante.

Control de flujo

Para que el buffer de *waypoints* no se desborde en ningún momento, el DSP implementa un control de flujo en forma de flag. Dicho flag indica si la cola está o no llena, permitiendo o no la transmisión de más *waypoints* por parte del Bridge Module.

Errores

En el DSP pueden generarse distintos errores, que deben pasar a la RIL, que debe decidir qué hacer con ellos: ignorarlos o hacerlos subir a la siguiente capa en forma de errores menos específicos.

2.5 Sistema RIL

En la introducción del presente proyecto decíamos que el primero de los objetivos era el de realizar la capa RIL del HERO. Por ahora dicha capa la compondrán únicamente dos módulos: el Bridge Module y el Status Module.

Figura 2.5-1: Bloque RIL

En el siguiente capítulo estudiaremos el RIL en profundidad, así que aquí basta decir que en su comunicación con el resto de capas de la arquitectura (ya que la relación con el DSP la acabamos de ver), la información que circula puede ser de cuatro tipos diferentes, que veremos a continuación según sean entradas o salidas del módulo.

2.5.1 Entradas

Control

Que indica si un módulo debe comenzar o parar su labor. En realidad esta entrada está dirigida sólo al Bridge Module, ya que el Status Module estará ininterrumpidamente trabajando.

Data

Por el momento se refiere en realidad a datos asociados al movimiento, por lo que lo que se nos pasa son las órdenes de *GoToWP*, *TakeOff* o *Land* que se envían desde el Centro de Control con los parámetros necesarios. Esta entrada también va dirigida exclusivamente al Bridge Module dentro del RIL, que será el encargado de transmitir la orden al DSP.

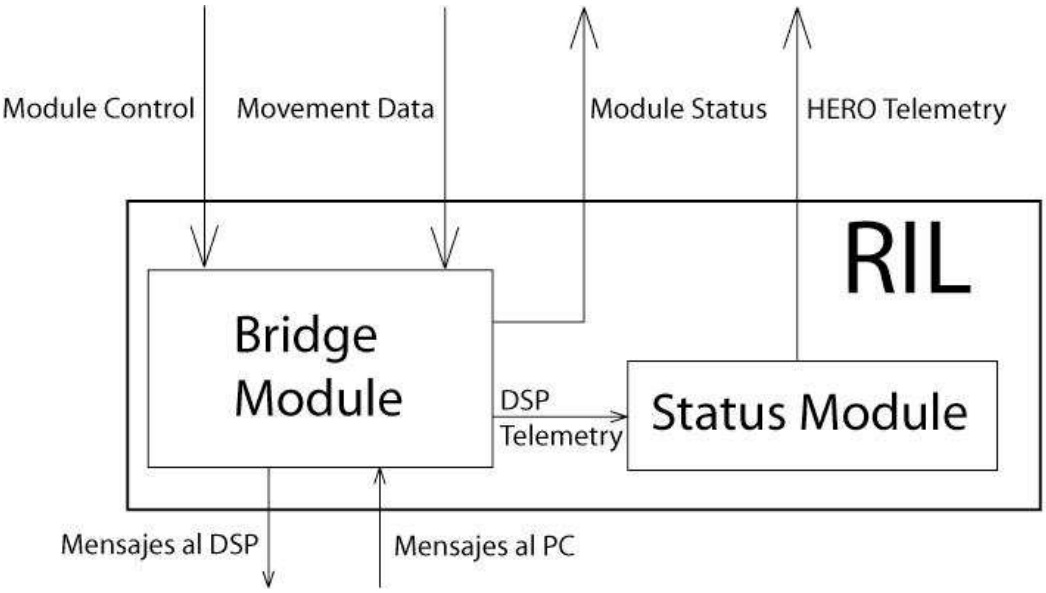

Figura 2.5-2: Dos módulos dentro del RIL

2.5.2 Salidas

Module Status

Se refiere al estado del módulo. Un módulo puede estar realizando una tarea (`MODULE_RUNNING`), haberla terminado (`MODULE_ENDED`), haberla abortado (`MODULE_ABORTED`), o no haberla ni siquiera comenzado (`MODULE_WAITING_DATA_TO_START`). Sólo el Bridge Module manda su estado como módulo, ya que como hemos dicho el Status Module estaría siempre corriendo.

Estado

Del RIL sale el estado del helicóptero en una estructura de tipo `HERO_TELEMETRY`, que es realmente la salida del Status Module. Éste ha tomado la información de la correspondiente salida del Bridge Module, y la ha generalizado un poco más para transmitirla a capas superiores.

Empezamos ya a intuir que, aunque el RIL lo conformen por el momento dos módulos, el peso y complejidad de cada uno de ellos es muy distinto. De hecho podríamos decir sin equivocarnos demasiado que el 80% del RIL lo constituye el Bridge Module. La arquitectura actual permite que se puedan añadir en el futuro nuevos módulos con facilidad para desarrollar o mejorar funciones del sistema. Aún así nos atrevemos a predecir que la importancia del Bridge Module seguirá siendo mucho

mayor que la del resto de los módulos dentro del RIL. Esto es así porque el Bridge Module es el último escalón antes de llegar al bajo nivel, correspondiente al DSP.

· Sistemas que simulan Sistemas reales:

2.6 Simulador del Helicóptero

La simulación es una herramienta que nos permitirá probar el correcto funcionamiento de todo el proyecto, sin los riesgos que supone hacer las pruebas en la realidad. Ninguna simulación es perfecta, y que en simulación todo el sistema funcione correctamente no quiere decir que en la realidad el helicóptero no se vaya a estrellar; pero lo que sí sería raro es que surgiendo problemas en la fase de simulación en la realidad la cosa funcionase perfectamente.

2.6.1 Simulador primitivo

En una fase primitiva, junto al desarrollo del primer DSP Simulator se realizó un sencillo simulador del UAV (implementado en la clase `UAVSimulator`) que lejos de predecir el comportamiento real del helicóptero, tomaba un *waypoint* y se dirigía hacia él a la velocidad indicada. No necesitaba de controlador alguno. Tan sencillo como eso. El tema de la velocidad ya lo veremos cuando se trate en profundidad la simulación, pero adelantamos ya que será problemático.

Figura 2.6-1: Primer simulador

Entradas:

- *Waypoint*: el `UAVSimulator` recibe un dato de tipo `PC_DATO_CAMINO_GEO`, que corresponde al sistema de coordenadas global interpretado por el CC (recordemos que se había desechado cualquier otro sistema para los *waypoints*). Por lo tanto, haremos coincidir el sistema de referencia del `UAVSimulator` con éste, que definía longitud, latitud y altura según los ejes cartesianos x, y, z. De las unidades en que se expresan estas magnitudes nos despreocupamos en absoluto: El helicóptero obedecerá sin problemas, ya que lo único que realmente buscamos con este simulador es comprobar que la información circula correctamente, y que el helicóptero recibiría y acataría las órdenes. Podemos resumir que lo único que hacemos es, a partir del

waypoint y la posición actual, descomponer la velocidad objetivo de forma que nuestra trayectoria sea una línea recta en el espacio. Cómo se hace esto lo veremos en el capítulo correspondiente a la simulación.

Salidas:

- Estado del helicóptero, del tipo `ESTADO_HELICOPTERO`. En cada iteración del bucle de control la posición se irá actualizando según la velocidad impuesta. Cuando se nos confirma que hemos llegado a nuestro objetivo (o a sus proximidades, ya que se tiene cierta tolerancia), las velocidades se ponen a cero y el vehículo se detiene.

- Errores de DSP (`DSPError`). Se deja la posibilidad, aunque no se utiliza por el momento, de mandar mensajes de error en el caso de que se genere algún problema, como si de un error de DSP se tratara. En las simulaciones actuales, de la generación de estos errores se encarga un hilo diseñado para ello, y no el simulador del UAV en sí.

Resulta difícil pensar en un simulador más sencillo, pero su misión ha sido muy importante en el desarrollo software. Esto se debe a que este modelo permite ver que las comunicaciones funcionan, que llegan los datos correctamente de punta a punta de la jerarquía, que los logs son coherentes, que en ningún momento el robot queda en un estado inconsistente... en definitiva a corregir los fallos en la estructura del código.

2.6.2 Simulador adaptado del MARVIN

Más tarde, para probar un control en sí mismo sobre una estructura software ya probada con éxito (gracias al modelo que acabamos de ver), se hacía necesario un nuevo modelo de helicóptero más fiel a la realidad. Entonces se decidió adaptar un simulador preexistente para el helicóptero MARVIN, de la Universidad de Berlín. Aún siendo un modelo bastante simplificado, resulta mucho más realista que el anterior. Las diferencias del MARVIN con respecto a nuestro HERO harán que la adaptación no se quede en el campo del software y sea necesario además cambiar algunos parámetros del modelo. Todo esto lo veremos en mayor profundidad más adelante, en el capítulo dedicado a la simulación.

Figura 2.6-2: Modelo más realista

Entradas:

- Señal de las actuaciones sobre colectivo, cíclico, rotor de cola y gas. Con esto nos basta para ver cuánto más realista es.

Salidas:

- La salida es una estructura de tipo `SimStateS` que contiene la posición, velocidades, orientación y r.p.m. del motor, y que por tanto definirá el estado del helicóptero.

2.7 Simulador del Controlador

La utilización del modelo descrito en el punto anterior (implementado en la clase `HelicopterModel`) hizo a su vez necesaria la implementación de algún tipo de controlador, que calculara los niveles de actuación en función del punto al que queremos mover el helicóptero.

En el HERO real se utiliza en la actualidad una serie de controladores PID, pero para la simulación se prefirió aprovechar toda la información disponible del modelo del helicóptero para diseñar un controlador óptimo LQR que también veremos en el capítulo 4.

Figura 2.7-1: Controlador

Entradas:

- *Waypoint*, tal y como vimos en el simulador primitivo del punto 2.6.1.

Salidas:

- Señal de las actuaciones, como vimos en el punto 2.6.2.

Es decir, el controlador en serie con el simulador adaptado del MARVIN tendría prácticamente el mismo tipo de entradas y de salidas que el simulador primitivo. De hecho, tal conjunto se engloba en la clase `UAVRealSimulator`, que posee la misma interfaz que la `UAVSimulator`. Esto permite que en el entero proceso de simulación, se pueda elegir el modelo de helicóptero a utilizar (aquel primitivo o el adaptado del MARVIN) cambiando simplemente una clase por la otra.

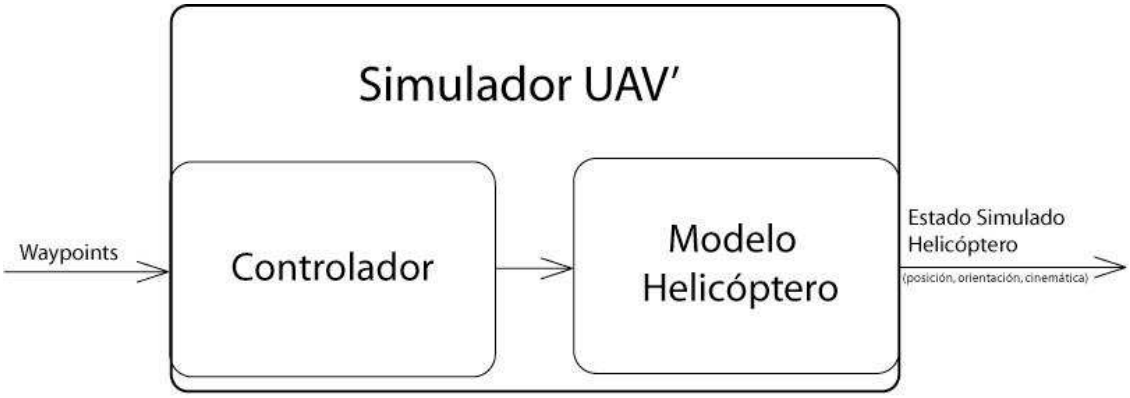

Figura 2.7-2: El `UAVRealSimulator`

2.8 Simulador del DSP

Para realizar las simulaciones no podíamos utilizar el verdadero DSP, ya que éste está programado para trabajar sobre un verdadero helicóptero. Por lo tanto teníamos también que realizar un simulador de DSP. En concreto se necesitaba tener capacidad de comunicación con el PC a través del puerto serie y con cualquiera de los dos simuladores del UAV que acabamos de ver de la forma que sea posible.

El PC no debe notar la diferencia, por lo que las entradas y salidas a él referidas no varían respecto a las que vimos en el punto 2.4 correspondiente al DSP real. Lo que sí cambia es que, en lugar de recibir la información de los sensores, recibe la del simulador elegido del helicóptero, y en lugar de calcular la señal para los actuadores manda directamente el *waypoint* recibido al simulador del UAV.

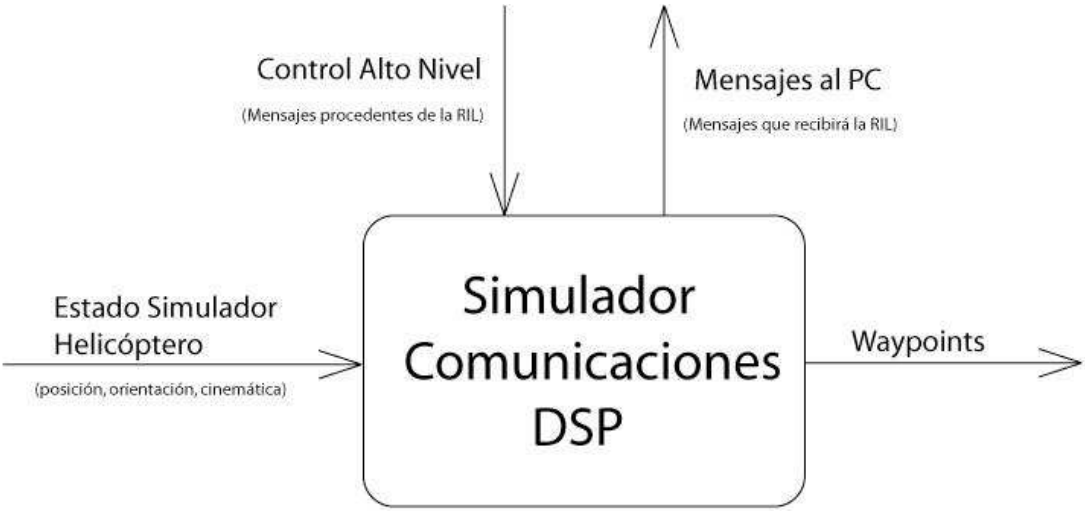

Figura 2.8-1: Simulador de las comunicaciones del DSP

Para todo ello contábamos con el código del DSP real que se adaptó en parte a las posibilidades de un PC y de C++ (el DSP está programado en lenguaje C). En el siguiente capítulo, que es el primero que trata en profundidad sobre el software, se explica en qué puede consistir esta adaptación y mejora.

Finalmente, con la participación de todos estos sistemas que simulan a otros sistemas, se consiguió realizar un proceso (`DSPSIM`) que lograba interpretar los mensajes que le llegaban por puerto serie desde el RIL, que respondía con mensajes de tipo *ack*, control de flujo o `DSPError` y que en todo momento mandaba el estado estimado por el simulador de helicóptero... todo como si del verdadero conjunto DSP-Helicóptero se tratara.

· Relaciones entre Sistemas:

Reales o simulados, los sistemas de nuestro proyecto tienen relación entre ellos. En la Figura 2-1 se expresan dichas relaciones de forma gráfica. Hemos introducido también el puerto serie, pero no como sistema, sino como interfaz. Esto quiere decir que las comunicaciones entre DSP (incluso simulado) y RIL se realizan en todo momento a través del puerto serie. Las otras entradas y salidas del RIL proceden o se dirigen a capas superiores del software.

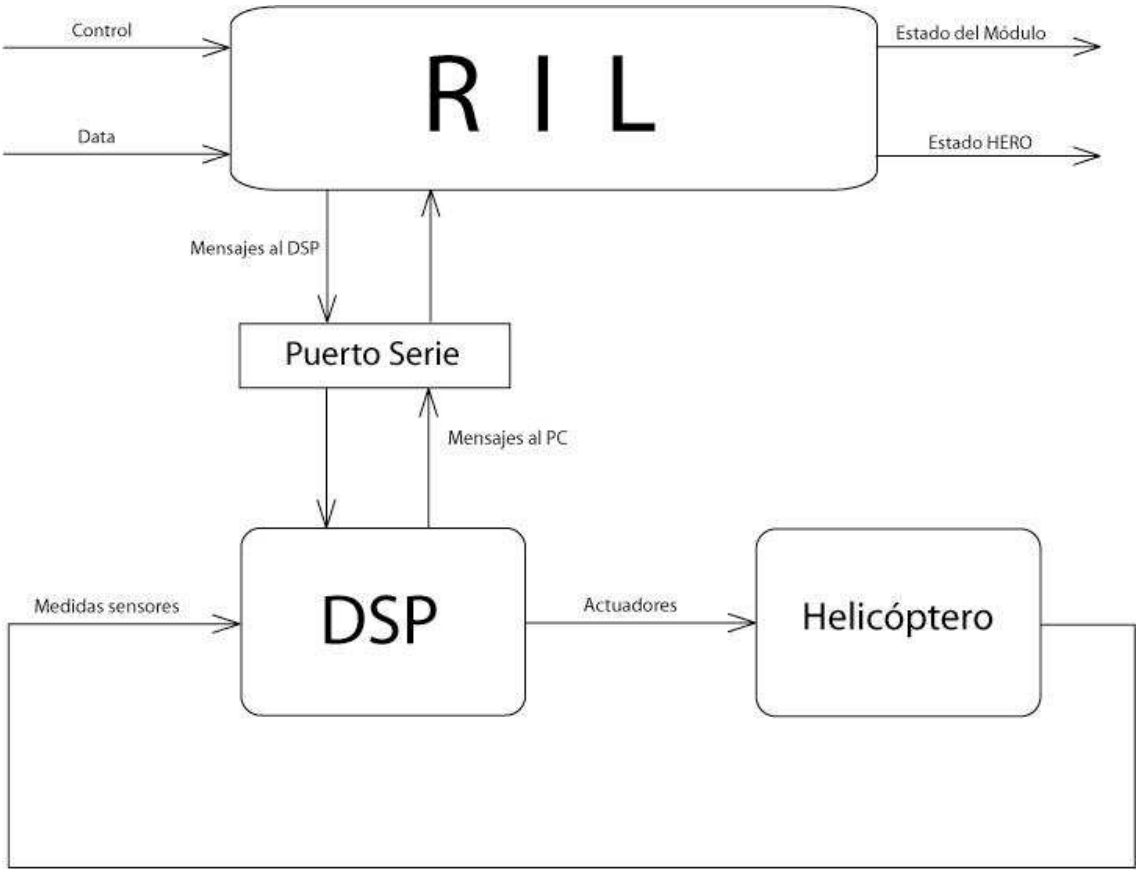

Figura 2-1: Relaciones entre los sistemas