Universidad de Politécnica de Valencia

Máster de Automática e informática industrial

Tesina de Máster

Desarrollo de una plataforma de tiempo real para la implementación de algoritmos de control multivariables: Ampliación al control de orientación de vehículos aéreos

Andreu Berna Ferri

Valencia - 29 de septiembre de 2010

Departamento de Control y Automática

Universidad de Politécnica de Valencia

Tesina de Master

Desarrollo de una plataforma de tiempo real para la implementación de algoritmos de control multivariables: Ampliación al control de orientación de vehículos aéreos

Autor: Andreu Berna Ferri

Ingeniero Industrial(Automática e Informática Industrial)

Director: Pedro García Gil

Doctor Ingeniero Industrial

Codirector: Pedro Albertos Pérez

Catedrático Ingeniero Industrial

Índice

Índice de figuras

1. Introducción y motivación del proyecto

1.1. Control Multivariable

El siguiente documento se enmarca dentro del proyecto Prometeo, orientado en este caso al Estudio e Implementación de Técnicas de Control Multivariable en procesos con dinámica rápida, concretamente a un Helicoptero quad rotor. El control automático de pequeñas maquinas voladoras permite alcanzar multiples aplicaciones en los campos de seguridad(supervisión aeroespacial, trafico urbano), manejo de riesgos naturales(supervisión de volcanes activos), del medio ambiente(medición de la polución aérea) para la intervención en ambientes hostiles(atmósferas radioactivas, eliminación de minas sin intervención humana), manejo de instalaciones de tierra(Presas, lineas de alta tensión, oleoductos o gaseoductos), agricultura(detección y tratamiento de cultivos infectados), y grabaciones aéreas en la producción de películas, para citar algunos ejemplos [11].

El comportamiento de un sistema dinámico se encuentra condicionado por las acciones que se ejerzan sobre el mismo. Esas acciones pueden ser ejercidas como acciones deseadas, a través de variables manipuladas (válvulas, interruptores, relevos, potenciómetros, calefactores, ventiladores, etc.) o a través de variables no manipuladas directamente, generalmente llamadas perturbaciones (cambios de carga, masas, cambios de concentración, etc.). Los efectos de esas acciones se pueden ver reflejados en una o más variables del sistema (temperaturas, niveles, presiones, velocidad, concentraciones, posición) que bajo ciertas condiciones se desea mantener en un valor determinado (variables controladas).

En general, el objetivo de la teoría de control es el diseñar estrategias que permitan comandar un conjunto de variables (variables manipuladas), de manera que se puedan mantener las variables controladas en unos valores deseados a pesar de las perturbaciones que puedan afectar al sistema.

Un sistema de control multivariable permite alcanzar el objetivo de mantener un conjunto de variables en un valor deseado a diferencia del control de sistemas SISO que solo permite controlar una variable al tiempo.

En control multivariable o MIMO (multiple Input, multiple Output) en general el cálculo del control por medio de un computador implica la sincronización de un número de elementos diferentes para un objetivo común. Las estrategias pueden ser descentralizadas(múltiples unidades de procesado de información con poca o nula intercomunicación) o centralizadas(una unidad de procesamiento que recibe y envía toda la información del bucle). Por supuesto, estrategias mixtas pueden existir.

Un vehículo aéreo no tripulado, comúnmente llamado *UAV* (*Unmanned Aerial Vehicle*, por sus siglas en inglés), es un vehículo aéreo capaz de realizar misiones en modo (semi)autónomo. Equipado, normalmente, con un sistema empotrado de control, este vehículo puede emplear diferentes sensores, como cámaras de vídeo, radares, láser, detectores infra-rojos, detectores ultra-sonidos, etc., para poder navegar en un ambiente desconocido, ver [9] y [1].

Los mini-helicópteros son una clase de *UAVs* que ha ocupado la atención de los investigadores durante las últimas décadas. Su versatilidad y maniobrabilidad, le confieren unas excelentes cualidades, para diversas aplicaciones en las cuales se necesite realizar vuelo estacionario.

El helicóptero con cuatro motores, actualmente es la configuración más utilizada, esto se debe, a que posee la misma naturaleza no lineal del helicóptero clásico y los mismos efectos aerodinámicos, pero con la diferencia de que es más fácil de controlar debido a que los torques están bien definidos [11].

En la actualidad, existen distintas plataformas o prototipos experimentales de vehículos aéreos, siendo la mayor parte de ellas específicas a ciertas aplicaciones, ver [29], [76], [56], [22], [55], [6]. En dichos sistemas, la implementación de las leyes de control está, en muchos casos, restringida a ciertos criterios y condiciones preestablecidos por el sistema comercial (o el constructor). En el caso de que el prototipo cuente con un sistema empotrado, éste es implementado en un algoritmo cíclico sin definir de manera especifica las prioridades del sistema [2], [76],

[77] y [54]. En cualquiera de los casos anteriores, se puede considerar que estos sistemas experimentales están limitados por el software de control y restringidos a un sólo sistema operativo.

Un problema crucial, cuando se trabaja con sistemas de control empotrados, es la carencia de una plataforma, no comercial, para validar y/o evaluar tanto el esquema de control (antes de ser implementado en el microprocesador) como los sensores y el circuito electrónico. Esta pre-validación permitiría, entre otras funcionalidades, ahorrar mucho tiempo en el ajuste de los parámetros del controlador. De igual forma, esta plataforma, podría ser utilizada para analizar, visualizar y detectar posibles fallos del sistema de control del vehículo, ante perturbaciones agresivas, y de esta manera evitar posibles pérdidas materiales. En nuestro conocimiento, no existe ninguna plataforma genérica de este tipo, en la cual se pueda intercambiar y/o implementar diferentes tipos de sistemas operativos, al mismo tiempo que ajustar, *on-line*, diferentes estrategias de control.

En este trabajo, se presenta una nueva plataforma de tiempo real para el desarrollo y validación de algoritmos de control en vehículos aéreos no tripulados. La plataforma se compone de una unidad de control terrestre (UCT) y un sistema aéreo. La UCT se ha desarrollado a partir del *Hypervisor XtratuM* [30], este sistema permite la ejecución concurrente de diferentes sistemas operativos. El sistema aéreo, se compone de un mini-helicóptero, una central inercial, que es utilizada para medir la orientación del vehículo, y una estructura en forma de peonza. El objetivo de la estructura en forma de peonza, consiste en poder realizar de forma sencilla y segura, el ajuste de los parámetros de los algoritmos de control, antes de proceder a su validación en vuelo libre.

Para poder validar el sistema desarrollado, se ha implementado un control en estabilización del vehículo aéreo. Como primer paso para el diseño del algoritmo de control.

1.2. Sistemas de Tiempo Real en sistemas Empotrados

Un sistema de tiempo real es una combinación de uno o varios computadores, dispositivos hardware de entrada / salida, y software de propósito especial, que interaccionan con el medio que les rodea [66]. Debido a la naturaleza cambiante del entorno con el que interactúan, junto con la necesidad de coordinación entre sus componentes, la validez de los resultados no depende solo de que la lógica e implementación de los programas sean correctos, sino también del tiempo en el que dicha operación entregó su resultado. Si las restricciones de tiempo no son respetadas, se dice que el sistema ha fallado.

Por lo tanto, es esencial que las restricciones de tiempo en los sistemas sean cumplidas. El garantizar el comportamiento en el tiempo requerido necesita que el sistema sea predecible, por lo tanto los tiempos de respuesta deben ser acotados [62].

En los sistemas de tiempo real se requieren métodos y herramientas fiables, que garanticen los requisitos temporales y que permitan utilizar plataformas en las que realizar pruebas y medir los tiempos con precisión. Los sistemas convencionales no tienen un comportamiento temporal predecible, es decir, no permiten garantizar los tiempos de respuesta y no acotados.

La informática de tiempo real es una tecnología de apoyo para varias áreas de aplicación importantes. Una lista de estas áreas incluye: control de procesos, plantas de energía nuclear, fabricación ágil, sistemas inteligentes de trafico de carretera, la aviónica, control del tráfico aéreo, telecomunicaciones (autopista de la información), multimedia, simulaciones en tiempo real, la realidad virtual, las aplicaciones médicas (por ejemplo, la telemedicina y seguimiento de cuidados intensivos), y en defensa (por ejemplo, control y comunicaciones). En particular, prácticamente todos los sistemas críticos de seguridad y muchos sistemas informáticos empotrados son sistemas de tiempo real. Además, la tecnología de tiempo real es cada vez más importante y penetrante, por ejemplo, la infraestructura del mundo depende cada vez más de ella.

Orientaciones estratégicas para la investigación en computación en tiempo real implica abordar nuevos tipos de sistemas en tiempo real, incluyendo, sistemas de tiempo real abiertos, sistemas de tiempo real distribuidos

globalmente, y los sistemas multimedia. Para cada uno de estos, se requiere investigación en las áreas de la evolución del sistema, modularidad, ingeniería de software,la garantía de rendimiento, fiabilidad y verificación formal, las cuestiones generales del sistema, lenguajes de programación, y la educación. También deben tenerse en cuenta en las soluciones consideraciones económicas y de seguridad, como así como los problemas especiales que causan restricciones de tiempo.

1.2.1. Tipos de sistemas de Tiempo Real

Podemos clasificar los sistemas de tiempo real como:

- Sistemas de tiempo real estricto (Hard Real Time): Cuando el sistema debe responder siempre a los eventos de una forma determinista, con tiempo acotado. El no cumplimiento puede tener consecuencias más o menos graves y en algunos casos puede ser preferible un trabajo imperfecto pero terminado a tiempo. Un ejemplo sería el control electrónico de estabilidad de Mercedes-Benz (ESP), que debe asegurar la respuesta dentro de un tiempo acotado o de lo contrario la acción de control sería inútil.
- Sistemas de tiempo real no estricto (Soft Real Time): Cuando el sistema debe responder de una forma determinista, pero las restricciones temporales son suaves siendo suficiente que el comportamiento del sistema esté dentro de un rango de tolerancia. Por ejemplo, un sistema de apertura de una puerta.
- Sistemas de tiempo real firme (Firm Real Time): Son sistemas de tiempo real estricto pero pueden tolerar pérdidas, si la probabilidad de ocurrencia de las mismas es baja. Por ejemplo la pérdida de una trama de audio o vídeo.

Para implementar un sistema del cualquier tipo, necesitaremos un sistema operativo que garantice la ejecución del código generado con restricciones de tiempo real.

1.2.2. Características de los sistemas operativos de Tiempo Real

Se caracterizan por presentar requisitos especiales en cinco áreas generales:

- Predecibilidad
- Latencia
- Planificabilidad
- Fiabilidad
- Tolerancia a los fallos

Una característica distintiva de un sistema de tiempo real es la **predecibilidad**. Implica que debe ser posible demostrar o comprobar a priori que los requerimientos de tiempos se cumplen en cualquier circunstancia. Como consecuencia, la predecibilidad implica:

- Una cuidadosa planificación de tareas y recursos.
- Cumplimiento predecible de requisitos temporales: determinismo.
- Anticipación a fallos, y sus requerimientos temporales.
- Consideraciones de sobrecargas: degradación controlada.
- Consideraciones de elementos de impredecibilidad.

- Dotar al sistema con capacidades de monitorización y control de tiempos (hardware,software, sistema operativo, lenguaje, lineas y protocolos de comunicaciones).

La **latencia** se refiere a cuanto tiempo consume un sistema operativo en dar servicio a la interrupción después de reconocerla. Las características de la latencia, entre otras:

1. La cantidad de tiempo necesario para iniciar la gestión de interrupciones, cambios de contexto y a servicios bloqueantes.
2. La cantidad de tiempo necesario para ejecutar la gestión. Generalmente, depende de la plataforma del hardware y el retraso del bus.

Un sistema operativo típico que no sea de tiempo real, el usuario no tiene control sobre la función de **planificación** del sistema operativo. En un sistema de tiempo real resulta esencial permitir al usuario un control preciso sobre la prioridad de las tareas. El usuario debe poder distinguir entre tareas rígidas y flexibles y especificar prioridades relativas dentro de cada clase. Un sistema de tiempo real también permitirá al usuario especificar qué procesos deben estar siempre residentes en la memoria principal.

Un tipo de planificación común, consiste en asignar prioridades a los procesos, en cada momento, la CPU se cede al trabajo más prioritario. Esta prioridad puede ser asignada permanentemente, o bien puede variar, por ejemplo en función de información nueva que se tenga sobre el comportamiento de un proceso.

La **fiabilidad** es normalmente mucho mas importante en sistemas de tiempo real que en los que no lo son. Un fallo transitorio en un sistema que no sea de tiempo real puede resolverse simplemente volviendo a reiniciar el sistema. Un fallo de un procesador en un multiprocesador que no sea de tiempo real produce una reducción del nivel de servicio hasta que se repara o sustituye el procesador averiado. Pero un sistema de tiempo real responde y controla sucesos en tiempo real. Las perdidas o degradaciones del rendimiento pueden tener consecuencias catastróficas, que pueden ir desde perdida financieras hasta daños en equipo e incluso perdida de vidas humanas.

La **tolerancia a los fallos** es una característica que hace referencia a la capacidad de un sistema de conservar la máxima capacidad y los máximos datos posibles en caso de fallos. Un sistema de tiempo real intentara corregir el problema o minimizar sus efectos mientras continua la ejecución (no todos los sistemas operativos de tiempo real cumplen este requisito, pero sí aquellos sistemas críticos).

Un aspecto importante a la tolerancia a los fallos es la estabilidad. Un sistema de tiempo real es estable si, en los casos en los que es imposible cumplir todos los plazos de ejecución de las tareas, el sistema cumple los plazos de las tareas mas criticas y de mayor prioridad, incluso si no se cumplen los de alguna tarea menos critica.

Para cumplir los requisitos anteriores los sistemas operativos actuales de tiempo real incluyen normalmente las siguientes características:

- Cambios rápidos de procesos o hilos.
- Pequeño tamaño (con una mínima funcionalidad asociada).
- Capacidad de responder rápidamente a interrupciones externas.
- Multitarea con herramientas de comunicación entre procesos, como semáforos, señales y sucesos.
- Planificación preferente basadas en prioridades.
- Reducción de intervalos en los que están inhabilitadas las interrupciones.
- Primitivas para demorar tareas durante un tiempo fijo y para detenerlas y reanudarlas.
- Alarmas especiales y temporizadores de alta resolución.
- Gestión de memoria con tiempos de respuesta predecibles

- Tiempos de respuesta acotados en todos los servicios.

El corazón de un sistema de tiempo real es el planificador de tareas a corto plazo. Lo que resulta importante es que todas las tareas de tiempo real estricto acaben (o comiencen) en su plazo y que también acaben (o comiencen) en su plazo tantas tareas flexibles de tiempo real como sea posible.

La mayoría de los sistemas operativos actuales de tiempo real son incapaces de trabajar directamente con plazos. En su lugar, se han diseñado para ser tan sensibles como sea posible a las tareas de tiempo real, de forma que, cuando se aproxima un plazo se pueda planificar rápidamente. Las aplicaciones de tiempo real normalmente necesitan tiempos de respuesta acotado en un rango de varios milisegundos, pero hay sistemas, como por ejemplo el que se ha desarrollado en este proyecto, presentan restricciones en un rango de diez a cien microsegundos.

1.2.3. Ejemplos de Sistemas de Tiempo Real y Logros de Investigación

Un sistema en tiempo real es aquel en el que la corrección del sistema no depende sólo de los resultados lógicos obtenidos, sino también en el momento en que los resultados se producen. Muchos sistemas de tiempo real son los sistemas empotrados, es decir, son componentes de un sistema más grande. Si el funcionamiento incorrecto de un sistema puede conducir a la pérdida de vida u otras catástrofes se le denomina un sistema de seguridad crítico. Como ejemplo, una importante infraestructura con sistema de tiempo real en el cual economía y la seguridad es altamente crítico es el control de tráfico aéreo. Un sistema de control aéreo, continuamente, debe manejar cantidades masivas de datos. A diferencia de otros sistemas de gestión de grandes datos, tales como las reservas de vuelos, los datos de control de tránsito aéreo están cambiando constantemente, y tienen un valor extremadamente alto (Relacionados con la seguridad pública) para cantidades muy cortas de tiempo (requisitos de tiempo de respuesta que varían desde unos pocos milisegundos para los datos del radar a unos segundos para la información de control de vuelo). Al finalizar, el nuevo sistema de control de tráfico aéreo de EE.UU. se estima que cuesta más de cinco mil millones de dólares. Sin embargo, el sistema es tan grande y complejo (la sistema tendrá entre 1 y 2 millones de líneas de código y miles de consolas) que nueva investigación en tiempo real se requiere para mejorar la seguridad aún más, bajar el costo del sistema y su mantenimiento, y mantener la seguridad conforme el sistema crece en tamaño y complejidad. Junto con esta necesidad de seguridad, existe la necesidad de una base más científica para el tratamiento coherente de tiempo, la concurrencia y confiabilidad.

La informática en tiempo real y empotrada también está jugando un papel clave en muchas sectores industriales. Tenga en cuenta la industria del automóvil. Los fabricantes de automóviles pueden permanecer en el mercado sólo si incorporan sistemas de computación en tiempo real avanzados en sus coches. En el futuro, los sistemas distribuidos de control en tiempo real reemplazará y mejoraran muchos de los sistemas convencionales de control del coche, por lo que los coches más eficiente y mejorando la seguridad pública. Antes de que la distribución de sistemas de control tiempo sea real, varios importantes retos a la investigación deben ser abordados: precisa respuesta en tiempo real a los microsegundos en un sistema distribuido, tolerancia a fallos bajo requisitos de tiempo estricto, mantenimiento fácil, y capacidad de prueba bajo las competitivas presiones de precios de la industria automotriz.

La investigación en computación en tiempo real ha sido muy eficaz. Por ejemplo, los avances en la ciencia de calidad de servicios en tiempo (QoS) han llevado a muchas condiciones de planificabilidad y a eficiencia, robusta y precisa algoritmos de validación eficientes, robustos y precisos para aplicaciones en tiempo real tales como el control digital y lectura de vídeo y de audio. El uso exitoso de la tecnología de validación incorporado con estos avances teóricos [45] en el diseño y desarrollo de sistemas para la vida real existe en estos momentos. Un ejemplo es el sistema de software a bordo de los satélites en el NAVSTAR el Sistema de Posicionamiento Global (GPS). Junto con hardware especializado, este sistema mantiene una señal de temporización de alta precisión para los usuarios, monitoriza la integridad de la información de navegación, estima los parámetros orbitales del satélite, y mantiene la sincronización de la constelación GPS. La realización puntual de muchas tareas de este sistema

debe ser garantizada, ya que muchas aplicaciones terrestres depende de la información GPS, y si la información del GPS es errónea, podría tener consecuencias graves para estas aplicaciones.

Otro ejemplo del éxito de la tecnología en tiempo real está en el software de tiempo real de la industria cerámica. Esta industria ha seguido de cerca el progreso de los semiconductores de la industria, en general, y el sector de los microprocesadores, en particular. Estimaciones actuales estima que se gastan más de 2 billones de dólares anualmente en herramientas, aplicaciones software y sistemas operativos integrados. Este mercado también está creciendo a aproximadamente el 25 % por año. El mercado de sistemas operativos de tiempo real, que comenzó alrededor de 1981, es actualmente de más de $ 100 millones por año con una tasa de crecimiento del 30-35 % anual . Sistemas operativos en tiempo real comerciales se utilizan en una amplia variedad de sistemas empotrados incluyendo aviónica, medicina, comunicaciónes y en aplicaciones de instrumentación. Muchos de estos sistemas son de "misión crítica" y han sido certificados por organismos gubernamentales, entre ellas la US FAA y otros equivalentes. Por ejemplo, el McDonnellDouglas MD11, Boeing 757, 767 y 747-400 vuelan con sistemas operativos comerciales en tiempo real. Éxitos futuros también se esperan en otras áreas. Por ejemplo, el comercio en tiempo real en Internet está transformando la forma en la que se hacen negocios. Dado que por cada PC que existe hay 1000 de sistemas empotrados, una próspera industria sistemas empotrados es de esperar. Habrá pequeños procesadores integrados en millones de productos, haciéndolos más inteligente. Básicamente, el potencial de la utilización futura de tecnología de tiempo real es ilimitado.

1.2.4. Desafíos futuros y Motivaciones

Uno de los problemas latentes cuando se trabaja con sistemas de control empotrados(ECS) es la falta de una plataforma para evaluar el algoritmo de control antes de ser "empotrado". Esta prevalidación del algoritmo de control ahorrara mucho tiempo en el ajuste de las ganancias de control. Además, con esto, podemos detectar fallos del sistema potenciales antes de permitir cualquier posible perdida económica. Se ha escrito mucho, desde el inicio de los sistemas empotrados sobre la necesidad de tener plataformas de pruebas para estos, puesto que los errores de programación ocurren y puede que no se detecten hasta el momento de las pruebas en hardware. Como se comenta en [21] una de las maneras mas comunes de probar el programa es realizar simulaciones en ordenadores, trasladando tanto código como sea posible a las simulaciones, la pega, que también se comenta, es la no posibilidad de utilizar el 100 % del código de un sistema empotrado en una simulación por ordenador y, también, un gran defecto de este método en control es que existen diferencias entre el modelo utilizado en una simulación y el proceso real. Se han desarrollado diferentes soluciones para diferentes ámbitos, para sistemas informáticos se puede observar que una de las más utilizadas es la creacion de una plataforma de pruebas(ver [27], [94], [70]).

Avances recientes en tecnología de software, han producido una tendencia ha diseñar, desarrollar e integrar más componentes de software y hardware para sistemas empotrados de tiempo real, como, por ejemplo, en equipamiento automovilístico y en telecomunicaciones móviles. La capacidad para permitir actividades de tiempo real, acceso a los controladores de los dispositivos, tolerancia a fallos y mínima distribución de hardware y software son las principales características que se tienen en cuenta para el desarrollo de todo ECS.

Sin embargo, es común, que las características del desarrollo software sean mal implementadas, debido a que no existe un gran entendimiento del concepto de *tiempo real*. Esto significa que un buen entendimiento podría influenciar positivamente en la efectividad del producto final. Ademas, el uso de un lenguaje y herramientas apropiadas para desarrollar aplicaciones ECS es una de las acciones clave en el diseño e implementación de software.

Actualmente existen métodos de evaluación para sistemas empotrados, pero estos solo son para aplicaciones especificas o lenguajes específicos de programación. Igualmente, el uso de estos sistemas conlleva elevados costes de desarrollo y mantenimiento, ademas de requerir un profundo conocimiento de estos. Cabe destacar también, que el uso de estos métodos puede necesitar que se sigan ciertos criterios, preestablecidos por el sistema, para la

implementación de leyes de control. Esto puede acarrear problemas, ya que el algoritmo de control tendrá que ser ajustado o reconfigurado, produciéndose modificaciones en la estructura y en el comportamiento del sistema.

En mi conocimiento, no existe aun una plataforma genérica, en procesos de control, en la cual es posible implementar diferentes tipos de sistemas operativos y/o procesos. El uso de un sistema operativo de código abierto, como es el *Real-Time Linux*, puede mejorar el proceso de desarrollo de muchas maneras(bajo precio de mantenimiento y desarrollo y robustez a la hora de ejecutar otras aplicaciones). Especialmente, el modelo de desarrollo concebido en la comunidad de código abierto puede ser adaptada fácilmente al diseño e implementación de aplicaciones de tiempo crítico, mejorando la confianza y la escalabilidad de los componentes de software.

En esta tesis de master introducimos una nueva plataforma genérica, trabajando en un ambiente Linux, compuesta por los sistemas *Xtratum* y *Partikle*.

- Linux nos proporcionan una interacción entre el usuario y el proceso completamente abierta
- Partikle asegura la ejecución en tiempo real, aunque se estén ejecutando otros procesos a la vez.
- Xtratum sirve de supervisor y permite mantener una comunicación entre Linux y Partikle.

1.3. VTOL, UAV(Unmanned Aereal Vehicules) y Quadrotor

Desde tiempos muy remotos el hombre ha querido volar; es así, como detrás de la imaginación y la innovación de diferentes científicos, investigadores y hombres en el mundo, se ha podido ver como la humanidad ha ido avanzando en el tema, logrando grandes resultados y dominando cada día más el espacio aéreo. Alrededor del siglo XV, y a sus 67 años de edad, Leonardo Da Vinci diseñó lo que sería para la humanidad, el primer helicóptero del mundo como se puede apreciar en la foto 1.

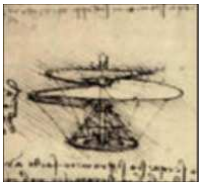


Figura 1: Diseño Da Vinci

Posteriormente, cuando la humanidad festejaba sus 1900 años después de Cristo, los hermanos Wilbur y Orville Wright desarrollaron el primer avión, el cual logró volar unos cuantos metros antes de caer al suelo, sin embargo, a pesar de las caídas, siempre existió la fe de construir un aparato volador que funcione. En la primera década del siglo XX, se diseñó en San Petersburgo (Rusia), una especie de helicóptero que se elevaba por efecto de dos hélices de 30 aspas cada una; este vehículo usaba un motor de 50 HP. Sus creadores lo denominaron Sorokin (foto 2) y pese a que optimizaron el diseño una y otra vez, éste nunca llegó a fabricarse. Paralelamente en Petrogrado y contando con el apoyo del ministro de guerra de Rusia, se diseñó un vehículo llamado Aeromóvil que contaba con cuatro rotores ubicados en los extremos de una estructura metálica. En un primer momento se colocó un motor de 25 HP en la parte inferior, que movería las cinco aspas de cada hélice y los 1300 Kg. de peso que tenia la nave. Este proyecto, al igual que el Sorokin, nunca fue terminado ya que el tiempo de estudio y diseño se prolongó, no contando con más fondos que lo solventaran.

Figura 2: Vehículo Sorokin

Luego de 12 años, en enero de 1921, el Doctor George Bothezart e Ivan Jerome ganaron un contrato del gobierno de Estados Unidos de Norteamérica para desarrollar un vehiculo aeronáutico que pueda despegar y aterrizar verticalmente. Esta nave pesaba 1678 Kg., tenia forma de cruz y el piloto se colocaba al centro de ésta.

Cuatro hélices de 8.1 metros de diámetro estaban colocadas en sus extremos con una pequeña inclinación, tal que si se intersecaban todas las proyecciones de éstas, se unían en un punto justo encima del centro de gravedad. En este caso el motor que utilizó la nave tuvo una potencia de 180 HP. Para poder controlar la estabilidad y la altura necesaria, cada hélice de 6 aspas tenía

un control para variar la velocidad de giro y traslación de la nave. Así, en octubre de 1922 el vehículo hizo su primer vuelo con un peso total de 1700 Kg. como se puede ver en la foto 3 . Luego de un año y con un nuevo motor de 220 HP marca Bentley BR-2 ya se habían hecho más de 100 vuelos logrando demostrar que el vehiculo con esta configuración podía mantenerse en el aire con buena estabilidad a un máximo de 5 metros de altura. A pesar de las ventajas mencionadas era mecánicamente muy complejo y susceptible a problemas de confiabilidad ya que no contaba con mucha maniobrabilidad.

A diferencia de los vehículos aeronáuticos antes mencionados, en 1945, la empresa Bell comenzó una investigación para evaluar la configuración coaxial, la cual tuvo éxito anteriormente en Francia y California. Es así que se logró fabricar un vehículo que contaba con dos hélices montadas una sobre otra en un mismo eje. Para evitar que la cabina comience a girar, las hélices giraban en sentidos opuestos anulando el torque generado sobre el eje. En este caso los problemas en la caja de transmisión hicieron que la investigación se deje de lado.

Figura 3: Prueba de vuelo de vehículo DAV

El 11 de febrero de 1953 un diario Canadiense reportó que un platillo volador estaba siendo desarrollado en Malton, Canadá. Esta historia comenzó en 1952, en el contexto de la guerra fría entre Rusia y EEUU, cuando este último, luego de un trabajo de inteligencia encontró que en Canadá, una alianza Británico-Canadiense formada desde 1950 investigaba sobre naves de despegue y aterrizaje vertical, también conocidas como VTOL por las siglas en el idioma inglés Vertical take-off and landing. Esta alianza estaba en proceso de fabricación de un platillo volador, dentro del cual, el encargado de diseñarlo fue el ingeniero John Frost, que conocía mucho sobre este tipo de tecnología. Al inicio, el vehículo contó con una turbina, que en su primera prueba derritió toda la base de la nave, por lo que fue necesario buscar otro tipo de propulsión, pero el presupuesto ya estaba acabándose. Fue en ese momento que la Fuerza Aérea de EEUU llegó donde se desarrollaba esta nave y propuso auspiciar el proyecto que le había costado al gobierno Canadiense $400 000. Es así que se rediseño todo el vehículo. Se colocaron 3 turbinas más pequeñas alrededor de una nave circular, pero cuando se hicieron las pruebas, la estabilidad era muy poca. Luego se fijó un gran ventilador radial en la parte central el cual aspiraba aire por la parte superior, generando el empuje necesario para poder mantener la nave en vuelo. El piloto y el copiloto iban sentados en cabinas colocadas a los extremos Este vehículo llegó a volar a bajas alturas como se observa en la foto 4 , es decir hasta 2 metros, porque cuando se elevaba más se volvía inestable llegando a voltearse 180 grados durante las pruebas. Los estudios siguieron pero esa configuración generaba poca estabilidad, por tal motivo en 1960 fue oficialmente terminado el proyecto. El prototipo del Avro VZ-9 así llamado, esta ahora en EEUU en el museo de Fort Eustis, Virginia.

Figura 4: AVROCAR

Bajo la idea de buscar algo diferente, Igor Bensen de nacionalidad rusa, formó en E.E.U.U. la Bensen Aircraft Corporation con la idea de seguir investigando el tema de vehículos con alas rotatorias. Es así que en 1954 fue probado el Midjet, que contaba con una hélice de 4.5 metros de diámetro sobre una estructura pequeña donde se sentaba el piloto como se muestra en la foto 5.

Figura 5: MIDJET

La diferencia de esta nave con otras diseñadas anteriormente, fue que la hélice era movida por dos turbinas de 40 HP colocadas en sus extremos, llegando a 120 Km/h en velocidad de crucero y logrando levantar hasta 180 Kg., teniendo en cuenta que el peso total era de 45 Kg.

Ese mismo año, en el Instituto de Aviación de Moscú (Rusia) se organizó un grupo de diseño para desarrollar vehículos aeronáuticos VTOL. Para este vehículo se seleccionaron cuatro turbinas situadas en los bordes de la nave cruciforme y con propulsores de 6 metros de diámetro dispuestos verticalmente como el fuselaje. Aproximadamente la masa del denominado Bratukhin (foto 6) fue de 55 toneladas con un empuje completo de 90 toneladas. Este proyecto se caracterizó por su originalidad y la novedad de las soluciones técnicas propuestas, una de las cuales fue el circuito eléctrico de la transmisión que substituyó al mecánico tradicional con los reductores ejes y embragues.

Figura 6: BRATHUKIN

Figura 7: PAWNEE

Pero mientras Rusia avanzaba con estos estudios en Norteamérica la Oficina de Investigación Naval, basándose en la idea de una plataforma voladora con un gran ventilador para poder hacer un vehículo unipersonal VTOL, comenzó con los estudios desarrollando el proyecto YHO-1E, el cual serviría para vigilancia y transporte.

Así, luego de dos años, en febrero se hicieron las pruebas, obteniendo buenos resultados. Se rediseñó el proyecto cambiando de nombre a VZ-1 Pawnee el cual contaba con un conducto de 1.5 metros de diámetro y sobre éste, 8 nervaduras equidistantes para lograr una buena estabilidad, el piloto iba en el centro con correas de seguridad maniobrando la altura en función de la potencia que le exigía al motor como se muestra en la foto 7.

En noviembre de 1956 la Oficina de Investigación Naval, mejoró el diseño e hizo el tercer prototipo, colocándole un motor adicional de 40HP y ampliando el ducto a 4 metros de diámetro. Pero las desventajas de esta plataforma voladora se hicieron notar, siendo pesada, lenta y muy delicada para un campo de batalla. Mientras la OIN iba dejando de lado el estudio de esta nave, se comenzaron a diseñar otras aeronaves con el mismo concepto de plataforma voladora. Así se desarrolló y probó, para la armada de Estados Unidos, el De Lackner DH4 Helivector, más conocido como Aerocycle, que por cambios en el diseño mejoró notablemente al VZ-1 Pawnee. El DH4 unipersonal, tenia un motor de 42.9 HP que movía a una hélice situada debajo del piloto. Esta plataforma llegaba a velocidades cercanas a los 110 Km/h teniendo como ventajas la estabilidad, la maniobrabilidad, el poco tiempo de adiestramiento para conducirla, y que para su aterrizaje contaba con 4 colchones de aire en la parte inferior. Sin embargo, el concepto del dispositivo de elevación individual era juzgado de no ser práctico bajo un pensamiento táctico, por lo que al igual que el VZ-1, el Aerocycle como se puede ver en la foto 8 , se dejó de estudiar. En los años siguientes hasta el día de hoy, la plataforma voladora se ha seguido estudiando para mejorarla en diferentes aspectos. Actualmente la estructura es más liviana y la maniobrabilidad es mejor. Existen empresas que las venden con fines recreativos en diferentes portales de Internet con un costo aproximado de 50 mil dólares americanos.

Figura 8: Aerocycle

Figura 9: QUADROTOR

1956 fue un año en el que se llevaron a cabo muchos estudios sobre los VTOL, ya que aparte de las plataformas antes mencionadas, se desarrollaron otros dos vehículos. Uno de ellos fue el Convertawings Model “A” Quadrotor tratando de revivir el concepto de un vehículo hecho en Francia

en 1922. Lo especial de esta nave era que la estabilidad y la maniobrabilidad se controlaban variando el empuje generado por cuatro rotores en los extremos (foto 9). Estos eran movidos con dos motores de 90 HP cada uno, que por medio de fajas en V transmitían la potencia. La nave de aluminio con una longitud de 6.6 metros y 1 tonelada de peso llegaba a velocidades de 128 Km/h. El mecanismo de control era bien simple por lo que el primer prototipo fue fabricado y voló con éxito. Luego se planificaron otros modelos pero solo en eso quedaron.

El otro vehículo desarrollado en ese año fue el VZ-6 diseñado por la empresa Chrysler, luego de ganar un contrato con la armada de Estados Unidos. Esta máquina voladora debía ser ligera y comportarse en el aire como lo hace un jeep en la tierra. Bajo estas condiciones se logró hacer una nave, con dos hélices grandes movidas con un motor de 500 HP (foto 10).

Figura 10: Vehículo VZ-6

A finales de 1958 la armada utilizó dos de estos vehículos y se hicieron los primeros vuelos. Con 1080 Kg. el VZ-6 era muy pesado, tenía poca potencia y estabilidad, tanto así que durante la primera prueba se volteó 180 grados, logrando salir con vida el piloto, pero marcando un daño económico muy grande, por lo que se decidió no seguir con esta investigación.

Pese al frustrado intento con el VZ-6, la idea del jeep aéreo permaneció, fue así como en 1957 la división Curtis Wrigth ganó el contrato para fabricar estos prototipos diseñando el VZ-7, un vehículo ligero con despegue y aterrizaje vertical. Alrededor de un rectángulo metálico en donde se ubicaban el piloto, el combustible y los controles, iban 4 hélices grandes movidas por una turbina de 425 HP. Para elevar su eficiencia, los prototipos tenían ductos para las hélices. La estabilidad se lograba variando el empuje de cada una de éstas.

Se fabricaron dos vehículos para el programa de prueba pero, pese a que eran relativamente estables, no se pudieron resolver los requisitos de altitud y velocidad especificados por el ejército, por lo que ambas naves fueron retiradas de servicio.

Figura 11: Vehículo VZ-8P

Ese mismo año Piasecki Aircraft Corp. ganó un contrato para la investigación y desarrollo de un vehículo denominado “el jeep volador tipo VTOL”. Este vehículo, debía volar muy bajo y tener una velocidad de avance de 110 Km/h. Se diseñó entonces el modelo 59K SKY CAR, el cual era muy parecido al VZ-6, que poseía dos ductos con rotores y hélices de tres aspas, movidas por motores de combustión interna Lycoming de 181 HP de potencia. En octubre de 1958 realizó su primer vuelo exitoso, por lo que lo bautizaron como VZ- 8P (foto 11). Luego de que fuese aceptado por la milicia norteamericana, se le cambió de motores, usando turbinas de 420 HP. Fue denominado 59N, y rebautizado por Piasecki como el Airjeep II.

Este vehículo, incorporó cambios significativos en su diseño, como el cambio del ángulo del ducto, la instalación de dos turbinas a gas de 400 HP que movían a dos rotores de 2.5 metros de diámetro. Tenía una longitud de 7.44 metros, altura de 1.78 metros y ancho de 2.82 metros, llegando a los 900 metros de altitud. También contó con asientos eyectores, y la colocación de dos espacios para pasajeros. Este vehículo hizo su primer vuelo en el verano de 1962. Ninguna de las dos versiones de los Airjeep VZ-8P presentaba problemas con la elevación superficial, y aunque en un principio el diseño fue planteado para bajas alturas que le permitieran camuflarse, podían elevarse a más de mil pies de altura. Ambas versiones fueron diseñadas para ser absolutamente estables y relativamente capaces de volar debajo de árboles y entre los edificios u otros obstáculos.

Además, el Airjeep, era capaz de atacar objetivos solamente dejando ver el arma sobre la línea de ataque. Aún en la actualidad, este sigue siendo un talento único, porque incluso los helicópteros modernos del campo de batalla deben levantarse sobre la línea de ataque para que sus armas se monten debajo del plano del rotor,

revelándose así y proporcionando un área mucho mayor para ser atacado.

El Airjeep II fue también el predecesor del avión moderno Stealth. Sus ductos hacían a las hélices invisibles al radar y al ojo humano. A pesar de sus múltiples cualidades, el Airjeep, como la mayoría de los "jeeps del vuelo" desarrollados durante este período, fue juzgado en última instancia por el ejército para ser declarado mecánicamente impropio a los rigores de las operaciones de campo. El concepto "jeep del vuelo", fue abandonado en favor del desarrollo posterior de los helicópteros convencionales del campo de batalla. Ambas naves fueron dejadas de lado a mediados de los años sesenta.

En 1962, el Dr. Moller, Master en Ingeniería y con PHD en la Universidad McGill fabricó un pequeño modelo de una nave VTOL que tenia en mente y la llamó XM-2. Dos años después, en el garaje de su propia residencia en California, comenzó la construcción de la nave pero esta vez a escala natural. Cuando fundó la Moller Aircraft Corporation, logró completar la construcción de este prototipo usando motores McCulloch de dos tiempos, que producían suficiente potencia para mantener el XM-2 volando a poca altura. En 1966, con el éxito de este primer vehículo VTOL, el Dr. Moller rediseñó el XM-2, colocándole dos motores fuera de borda Mercury, con el auspicio de UC Davis, California. El nuevo modelo, fue considerado un éxito total y en 1968, el Dr. Moller recibió su primera patente por el diseño del VTOL XM-2 (foto 12).

Figura 12: Moller XM-2

Las naves VTOL tenían una desventaja, y era que no podían llevar a muchos pasajeros, es así que con la misma tecnología se llevaron a cabo muchos estudios para desarrollar aviones con hélices en alas giratorias que pudieran rotar 90 grados, para hacer posible un despegue y aterrizaje vertical. Desde 1954 hasta 1966, se construyeron vehículos aéreos con estas características, como el Trascendental 1-G, el Bell XV-3, el Vertol VZ-2, Doak VZ-4 y el Hiller X-18, el Curtis-Wright, Canadair, Hiller-Ryan XC-142, VFW VC-400 y 500, solo para citar algunos ejemplos. Aún en la actualidad se estudian diferentes tipos de vehículos de despegue y aterrizaje vertical como por ejemplo el AV8-B Harrier.

Bell fue una de las compañías que más investigó este tipo de vehículos aeronáuticos. Destaca el X-22, desarrollado con la idea de tener una nave V/STOL (despegue pequeño y aterrizaje vertical). Durante las pruebas, esta nave demostró una estabilidad variable que la hacía más versátil que cualquier otro vehiculo V/STOL antes desarrollado. Las fuerzas militares de Estados Unidos tenían como objetivo, desarrollar investigaciones sobre vehículos V/STOL con variantes que permitieran aprovecharlos de maneras diferentes. Al respecto, los estudios demostraron que una estructura de dos ductos para las hélices, permitía alas más cortas y ligeras. El ducto alrededor de los cuatro rotores, mejoraba la eficiencia y proveía mayor seguridad a la tripulación.

Los estudios de Bell comenzaron en noviembre de 1962, luego de que ganara un contrato de 27 millones de dólares por el diseño y desarrollo de dos naves X-22. Esta empresa tenía experiencia en naves V/STOL como el X-14, XF-109. Este vehículo pesaba 7530 Kg., tenia una longitud de 11.9 m., y capacidad para 540 Kg. Cuatro turbomotores de 1250 HP y un tanque de 465 galones, se encargaban de mover los cuatro rotores. Tenía diez transmisiones que reducían de 19 500 rpm a 2600 rpm. Las hélices de tres aspas construidas por Hamilton Standard, medían 2.1 m. de diámetro, estaban hechas de fibra de vidrio reforzada con acero y colocadas dentro de un ducto para un mejor funcionamiento.

Figura 13: Moller XM-3

La nave VTOL de dos pasajeros más pequeña en el mundo fue la sucesora del XM-2, también creada por el Dr. Moller, el XM-3, se comenzó a fabricar en 1966. Un original diseño, con una sola hélice que giraba alrededor de los pasajeros, creando el empuje necesario para elevarse verticalmente era movida por 8 motores de go-kart (foto 13). En 1968, el Dr. Moller voló con éxito en este vehículo, patentándolo en 1969.

Con un diseño bastante similar al del XM-3; el XM-4 se fabricó, con la forma

de platillo volador. Avalado por el éxito del XM-2 y XM-3, la fabricación de este modelo comenzó en 1970 y debutó en 1974. Su diseño contaba con 8 motores, igual que el XM-3, pero en este caso eran rotativos cuya marca era Fitchel-Sachs y rodeaban al piloto en un arreglo circular patentado.

Figura 14: AROD

Como ya se dijo anteriormente, se ha estudiado mucho la tecnología de vehículos VTOL con diferentes formas, también el desarrollo de aviones que cuentan con hélices rotativas en los extremos de las alas. Así también existen los Vehículos Aeronáuticos no Tripulados conocidos por las siglas UAV que en el idioma ingles significa Unmanned Aerial Vehicles, en los que se han venido desarrollando con éxito, grandes investigaciones. Las aplicaciones para este tipo de naves son muchas, ya sea militar, de seguridad, reconocimiento o para competencia. En 1982, la marina inició en Hawai, Estados Unidos, el proyecto AROD (Airborne Remotely Operated Device); es decir, Dispositivo aerotransportado operado de manera remota. Un vehículo VTOL muy pequeño, que contaba con cuatro ductos con hélices que le permitía volar con gran facilidad. Tenia motores eléctricos y un cable que lo unía a tierra, a través del cual, pasaba la corriente (foto 14). Luego se convirtió en una nave de motores de combustión interna de dos tiempos, con 26 HP y un solo propulsor. Contaba con un sistema de dirección colocado en la parte inferior de la hélice para controlar dirección y la altura. La primera generación de AROD fue desarrollada por el Dr. Moller a manera de subcontrato, y continuó posteriormente, como parte de un proyecto mayor denominado GATERS, que en sus siglas en inglés significa Sistema Telerobótico de Aire y Tierra. AROD fue probado con éxito en diferentes oportunidades, pero dejó de ser estudiado debido a que los fondos fueron asignados a otros proyectos.

En los años que pasaron, el Dr. Moller antes mencionado por las naves XM- 2, 3 y 4, no se quedó con las manos cruzadas y continuó con la búsqueda y mejora de un vehículo aéreo VTOL. En 1985, luego de adquirir de la Outboard Marine Corporation, la tecnología sobre motores rotativos, comenzó con las modificaciones de sus motores tipo Wankel, logrando aumentar la potencia en 20 % y disminuir el peso en 50 %. Dos años después estos motores fueron probados en el XM-4 obteniendo buenos resultados. Luego de algunos cambios en la nave esta fue rebautizada con el nombre de M200X en 1989. Es así que el 10 de mayo de ese año fue presentado ante la prensa mundial. Luego de muchos años en los que se llevaron estudios sobre helicópteros y aviones, una empresa dedicada a la investigación de vehículos VTOL obtuvo grandes logros. Es así que en 1996, la empresa Trek Aerospace comenzó a desarrollar un esqueleto aéreo conformado por un motor rotativo, dos hélices de cinco aspas dentro de ductos y timones en la parte inferior para controlar la dirección.

Figura 15: DRAGON FLY

Así en 1999, Trek Aerospace firmó un contrato con la NASA (Administración Nacional Aeronáutica y Espacial) para compartir información sobre la tecnología de los ductos. Al colocar éstos con la configuración antes mencionada se lograba eliminar el torque que hacia difícil la maniobrabilidad de la nave haciéndola girar sobre el eje de la hélice. Con un equipo de 5 personas este trabajo se terminó en tres años, tiempo en el cual también fueron desarrollando, construyendo y probando vehículos VTOL, como el Dragonfly que se puede ver en al foto 15 entre otros, que tuvieron la característica fundamental de una buena estabilidad, pudiendo resistir en un túnel de viento velocidades de hasta 21.6 Km/h. Esta nave con una longitud de 4 metros, se basaba en un motor rotativo de 170 HP que mediante un sistema de hélices transmisión movía dos hélices en sentidos contrarios para contrarrestar los torques que producían levantando hasta 480 kilos y con una capacidad para almacenar combustible de 78 litros.

Dentro de los vehículos VTOL no tripulados el CL-327Guardián fue una versión mejorada de otros UAV como el Centinela. Los estudios para mejorar este vehículo venían desde 1964. El Centinela participó en los programas de la Marina de Guerra de Estados Unidos en los años 90, con varios vuelos de demostración. En 1996 se propuso un Centinela modificado.

El Guardián tuvo la ventaja de tener una producción de bajo costo, y fue uno de tres sistemas UAV seleccionados para participar en el programa de demostración de la Marina de Guerra VTOL UAV en 1998. Más de 50 horas de vuelos acertados terminaron a causa de un accidente en junio de 1998, cuando un depósito de gasolina se separó de la nave causando la pérdida de ésta. El CL-327Guardián era accionado por un motor de 125HP Williams WTS117-5, que transmitía la potencia a los rotores de 3.96m. de diámetro, pesaba 350 Kg., y la velocidad máxima de la travesía era de 155 Km/h.

En 1998 la compañía Sikorsky (Estados Unidos), diseñó el Cypher, un vehículo aéreo sin tripulación, conformado por dos rotores que giraban en sentido contrario, cuatro aspas en cada hélice, conducidos por un motor rotatorio de 52HP UEL AR 801, dentro de una cubierta de 1.95m. de diámetro hecha de grafito/epoxy. Este diseño podía ser accionado tanto automáticamente, como por el control de tierra. En pruebas realizadas por el ejército de los E.E.U.U, se buscaron diferentes objetivos para que el vehículo lograra encontrarlos, consiguiendo también detectar artillería (foto 16). Esta tecnología permitió que el Ministerio de Energía de los E.E.U.U., localizara las estructuras y los túneles subterráneos en el Estado de Nevada.

Figura 16: SIKORSKY UAV

Así como todos los vehículos mencionados, antes de poder fabricarlos y hacer las pruebas fue necesario inventarlos, basándose en una idea y un buen capital para lograr grandes avances tecnológicos. Air bike es una expresión de vehículos VTOL, una idea que se quiere llevar a cabo basada en una bicicleta aérea, la cual contaría con dos motores rotativos en la parte posterior que impulsan a una hélice que al aumentar la velocidad aumenta también el empuje para variar la velocidad. Sobre esta hélice se colocará un deflector de aire para poder darle dirección al vehículo. También contaría con un paracaídas en caso de emergencia por si fallaran los dos motores o sucede cualquier otro accidente. Esta bicicleta aérea no podrá retirar victimas pero, debido a su tamaño y rapidez esperada, podría llegar donde está el accidente en tiempos muy cortos con el médico para que pueda estabilizar los signos vitales del herido. El peso que se estima para esta nave está entre los 100 y 150 kilos.

Un nuevo carro volador VTOL está en construcción desde el año 2001 por un pequeño grupo de personas bajo la dirección del Dr. RAFI YOELI, Director Gerente de la AD&D Ltd. en Israel. Este nuevo vehículo es llamado City Hawk y podrá llevar a dos personas, aparte de contar con una autonomía de 1 hora de vuelo.

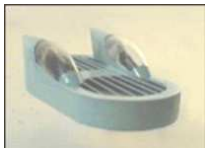


Figura 17: CITY HAWK

Llegará hasta los 2500 metros de altura, contará con dos hélices movidas por dos motores de 4 tiempos de combustión interna. Para poder girar y avanzar cuenta con timones por encima y debajo de las hélices. Las futuras versiones de esta nave serán usadas para operar en lugares donde los helicópteros no pueden entrar debido a su gran hélice; para transporte urbano, ambulancias, patrulleros entre otras aplicaciones. Todo el sistema utilizado en esta nave es muy parecido con el utilizado en investigaciones anteriores por lo que se espera se termine pronto el modelo y se logren hacer las pruebas, ya no con el prototipo como se puede ver en la foto 17 , sino con el vehículo a escala real.

Pasaron los años y el Dr. Moller continuó con la investigación de los vehículos VTOL, y fue así que, a lo largo de su carrera, fue reuniendo experiencia y conocimientos para continuar en el diseño de estas naves. Desarrolló el M150 Skycar y fabricó un prototipo que fue llevado y mostrado en Alemania, diseñado para una persona, con un peso de 386 Kg. Pero los esfuerzos se dirigieron al M400 Skycar, el cual, basándose en el M150 mejoró muchas características. Cuando el Dr. Moller terminó de fabricar, probar y hacer la presentación ante el mundo del M200X, modelo sucesor del M150, se presentó un nuevo reto; hacer un vehículo que pueda despegar y aterrizar como helicóptero, pero que logre volar como avión, siendo esta última la cualidad que faltaba en las otras naves. De esta manera el diseño varió por completo y las investigaciones no cesaron hasta que en marzo del 2003 se

hicieron las pruebas dando buenos resultados.

El M400 Skycar como se puede ver en la foto 18, basándose en el principio del Avión Harrier, que se puede elevar gracias a unos deflectores que giran dirigiendo el flujo de aire hacia abajo o hacia atrás, utiliza un sistema de 4 rotores giratorios con timones para direccionar y controlar el flujo de aire. Dentro de cada uno tiene dos motores, haciendo un total de ocho motores de 150 HP, cada uno de estos puede ser cargado fácilmente por una persona y puede utilizar combustibles como el gas, gasolina o petróleo.

Figura 18: SKYCAR

Este sistema patentado por el Dr. Moller permite que ascienda verticalmente y luego pueda viajar horizontalmente. También está hecho para recorrer distancias pequeñas en tierra como si fuese un auto. El fuselaje de esta nave está hecho con fibra de plástico reforzada lo que le da resistencia y poco peso. Se requiere bastante potencia para elevar la nave y lograr un buen aterrizaje, por eso, para el control se cuenta con dos palancas que están conectadas a una computadora que dirige los rotores, es decir, el piloto solo mueve la palanca hacia donde quiere ir y la computadora se encarga de dirigir la nave hacia esa dirección. Esta nave cuenta con un paracaídas de emergencia para cualquier incidente, la altura máxima a la que podrá llegar será 9.6 Km., cuenta con un peso de 1 tonelada y tiene un consumo de 45 Km. por galón de combustible. Podrá albergar a 4 pasajeros y llegar a una velocidad de subida de 90 Km/h. El precio del M400 Skycar va desde los $500 000 hasta $ 1 000 000 de dólares.

En el año 2000 El MARINER UAV de Sikorsky (empresa norteamericana de helicópteros), fue desarrollado y seleccionado por los infantes de marina de E.E.U.U. para probar conceptos de funcionamiento. El MARINER emplea un ventilador canalizado, consistente en dos rotores coaxiales de cuatro palas que generan la elevación. Un ala convencional se une al fuselaje, para proporcionar la elevación en vuelo delantero, reduciendo la carga en el ventilador de elevación. El ala, en conjunto con un segundo ventilador canalizado más pequeño, ubicado en la cola del avión, es usada para la propulsión delantera. El "Cypher II", también conocido así por ser de la misma familia del "Cypher", es capaz de llevar una carga útil de 22 Kg., con una autonomía de vuelo de dos horas. El aparato tiene un peso de 100 Kg. y una velocidad superior a 230km/h. Las alas pueden quitarse para operaciones militares en terreno urbano. De la misma manera como su precursor el "Cypher I". En la actualidad Sikorsky tiene un contrato de $5.46 millones para entregar 2 prototipos.

Figura 19: Sky Rider

A partir del segundo milenio, se comenzaron a hacer naves mucho más sofisticadas y estéticamente atractivas como por ejemplo el Skycar, es así que Macro Industries compañía estadounidense viene desarrollando el Sky rider X2R, nave VTOL donde su diseño cuenta con rotores giratorios para un despegue vertical y un vuelo horizontal. Se han escrito muchos artículos en torno a esta nave y en el mes de octubre del 2000 se terminó de fabricar un prototipo. El diseño de este vehículo tiene cabida para dos personas y un motor eléctrico que mueve las hélices que se encuentran dentro de los rotores (foto 19). La nave necesitaría 700 HP de potencia para elevarse, con una capacidad de carga para 150 Kg. y 100 galones de combustible. La velocidad de subida será de 72 Km/h. El precio se estima entre US$ 500 000 y US$ 1 000 000. El diseño terminado se ha programado para el 2006 y está dentro de los planes, la posibilidad el poder dirigirla mediante la voz.

La imaginación y el ánimo de siempre querer hacer algo nuevo y diferente ha llevado al hombre a proyectar para un futuro, diferentes vehículos que todavía están en estudio, es así que, inspirados en la tragedia del 11 de setiembre en EEUU, diferentes diseñadores aeronáuticos han creado proyectos de naves de rescate. Un ejemplo es Roy LoPresti y Larry Gordon que forman parte de un grupo de apoyo para el desarrollo de dos vehículos VTOL de salvataje, el TurboHawk y el Guardián que contarán con un sistema de propulsión basado en dos motores con una hélice y un turbopropulsor de 500

Figura 20: Guardian

HP respectivamente. El objetivo del Guardián será el rescate de personas a gran altura determinada, teniendo la cualidad de colocarse cerca de un edificio para poder evacuarlas como se puede ver en la foto 1.20. El TurboHawk, al contrario, está pensado para el traslado de tropas militares.

2. Estructura de la Plataforma desarrollada

2.1. Estructura Hardware

Como introducción se explicara que se entiende por helicóptero: Un helicóptero es una aeronave capaz de despegar y aterrizar de forma vertical. También llamado nave rotatoria, este aparato puede planear y rotar en el aire y puede moverse de lado y hacia delante y hacia atrás mientras esta en vuelo. Este vehículo aéreo puede modificar su dirección muy rápido y parar completamente, para comenzar a planear. El helicóptero comenzó como un principio básico de aviación mediante alas rotatorias. La precisión de los elementos constructivos que se obtuvo después de la Revolución industrial, permitió transformar los helicópteros en las modernas maquinas voladoras que podemos ver en la actualidad. También se considera necesario tener un conocimiento de que significa una *UAV*: Un *Unmanned Aerial Vehicule*(UAV), vehículo aéreo no dirigido, también llamado *drone*, es un termino auto descriptivo que se usa para definir aplicaciones militares y civiles de las últimas generaciones de aeronaves sin piloto.

Para implementar las diferentes Técnicas de control se tiene que trabajar en algún tipo de plataforma, en el caso de este proyecto se poseen 2:

1. Plataforma HOVER: Plataforma cuatrirotor que esta sujeta en el eje central y permite un control en yaw, pitch y roll.
2. Plataforma X-UFO: Plataforma cuatrirotor completamente libre y permite un control en yaw, pitch, roll y thrust.

Para la mayor parte de las técnicas de control se realizan pruebas sobre la Plataforma HOVER y, en cuanto son estables, se implementan en la plataforma X-UFO.

2.1.1. Modelo Dinámico de un helicóptero Quad-Rotor

Antes de presentar las plataformas seria necesario explicar como se ha obtenido el modelo utilizado. Este modelo se ha obtenido básicamente representando el mini-helicóptero como un cuerpo solido que se mueve en 3D y esta sujeto a una fuerza y tres momentos. La dinámica de los cuatro motores eléctricos es relativamente rápida y, por tanto, se desestiman, al igual que la flexibilidad de las aspas. Las coordenadas generalizadas del helicóptero son

$$q = (x, y, z, \psi, \theta, \phi) \in R^6 \tag{1}$$

donde (x, y, z) denotan las posición del centro de masa de los cuatro motores del helicóptero relativo a la estructura, y (ψ, θ, ϕ) son los tres ángulos de Euler (ángulo de guiñada, cabeceo y alabeo; o respectivamente *yaw*, *pitch* y *roll* en ingles) y representan la orientación del helicóptero.

Por tanto, el modelo se puede dividir en las coordenadas de traslación y las de rotación

$$\xi = (x, y, z) \in R^3,\ \eta = (\psi, \theta, \phi) \in S^3 \tag{2}$$

La energía cinética de traslación es

$$T_{trans} \triangleq \frac{m}{2}\dot{\xi}^T\dot{\xi} \tag{3}$$

donde m se corresponde a la masa del helicóptero. La energía cinética de rotación es

$$T_{rot} \triangleq \frac{1}{2}\dot{\eta}^T \mathbb{J}\dot{\eta} \tag{4}$$

La matriz J actúa como la matriz de inercia para toda la energía cinética de rotación del helicóptero expresada directamente en términos de la coordenada generalizada η. El único potencial de energía que necesita ser considerado es el debido al potencial gravitacional que viene dado por

$$U = mgz \tag{5}$$

El Lagrangiano es

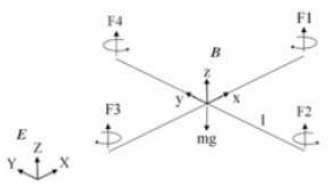


Figura 21: Modelo dinámico de un Quadrotor

$$\begin{aligned} L(q,\dot{q}) &= T_{trans} + T_{rot} - U \\ &= \frac{m}{2}\dot{\xi}^T\dot{\xi} + \frac{1}{2}\dot{\eta}^T\mathbb{J}\dot{\eta} - mgz \end{aligned} \tag{6}$$

El modelo dinámico del helicóptero se obtiene de las ecuaciones de Euler-Lagrange con la fuerza externa generalizada

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = F \tag{7}$$

donde $F = (F_\xi, \tau)$ es el momento generalizado y F_ξ es la fuerza de traslación aplicada al helicóptero debido a las variables de entrada. Se ignoran las fuerzas debido al cuerpo porque generalmente son de un orden de magnitud menor que la variables de entrada de control u y τ, entonces se puede escribir

$$\widehat{F} = \begin{pmatrix} 0 \\ 0 \\ u \end{pmatrix} \tag{8}$$

donde

$$u = f_1 + f_2 + f_3 + f_4 \tag{9}$$

y

$$f_i = k_i w_i^2,\ i \in [1,4] \tag{10}$$

donde $k_i > 0$ es una constante y w_i es la velocidad angular del motor i (M_i, $i \in [1,4]$), entonces

$$F_\xi = R\widehat{F} \tag{11}$$

donde R es una matriz de transformación que representa la orientación del helicóptero. Considerando la aproximación de c_α para cos_α y s_α para sin_α.

$$R = \begin{pmatrix} c_\theta s_\psi & s_\psi s_\theta & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta s_\phi \end{pmatrix} \tag{12}$$

los momentos generalizados de la variable η son

$$\tau = \begin{pmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{pmatrix} \tag{13}$$

donde

$$\tau_\psi = \sum_{i=1}^{4} \tau_{M_i} \tag{14a}$$

$$\tau_\theta = (f_2 - f_4)l \tag{14b}$$

$$\tau_\phi = (f_3 - f_1)l \tag{14c}$$

donde l es la distancia de los motores al centro de gravedad y τ_{M_i} es el par producido por el motor M_i.

Ya que el Lagrangiano no contiene términos cruzados en la combinación de la energía cinética de $\dot{\xi}$ y $\dot{\eta}$ las ecuaciones de Lagrange-Euler se pueden dividir por una parte en la dinámica para las ξ coordenadas y en la dinámica de η. Se obtiene

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_\xi \tag{15a}$$

$$\mathbb{J}\ddot{\eta} + \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T \mathbb{J}\dot{\eta}) = \tau \tag{15b}$$

Si definimos el vector de Coriolis como

$$\bar{V}(\eta, \dot{\eta}) = \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T \mathbb{J}\dot{\eta}) \tag{16}$$

se puede escribir

$$\mathbb{J}\ddot{\eta} + \bar{V}(\eta, \dot{\eta}) = \tau \tag{17}$$

pero también podemos reescribir $\bar{V}(\eta, \dot{\eta})$ como

$$\bar{V}(\eta, \dot{\eta}) = \left(\dot{\mathbb{J}} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T \mathbb{J})\right)\dot{\eta} = C(\eta, \dot{\eta})\dot{\eta} \tag{18}$$

donde $C(\eta, \dot{\eta})$ se refiere a los términos de Coriolis y contiene los términos giroscópicos y centrífugos asociados con η que depende de $\mathbb{J}$. Los detalles del calculo de $C(\eta, \dot{\eta})$ se pueden encontrar por ejemplo en [23].

Finalmente se obtiene

$$m\ddot{\xi} = \begin{pmatrix} -sin\theta \\ cos\theta \sin\phi \\ cos\theta \cos\phi \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \tag{19a}$$

$$\mathbb{J}\ddot{\eta} = C(\eta, \dot{\eta})\dot{\eta} + \tau \tag{19b}$$

Con el fin de simplificar vamos a proponer un cambio de las variables de entrada.

$$\tau = C(\eta, \dot{\eta})\dot{\eta} + \mathbb{J}\tilde{\tau} \tag{20}$$

donde

$$\tilde{\tau} = \begin{pmatrix} \tilde{\tau}_\psi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\phi \end{pmatrix} \tag{21}$$

son las nuevas variables de entrada. Entonces

$$\ddot{\eta} = \tilde{\tau} \tag{22}$$

Reescribiendo las ecuaciones:

$$m\ddot{x} = -u \sin\theta \tag{23a}$$

$$m\ddot{y} = u \cos\theta \sin\phi \tag{23b}$$

$$m\ddot{z} = u \cos\theta \cos\phi - mg \tag{23c}$$

$$\ddot{\psi} = \tilde{\tau}_\psi \tag{23d}$$

$$\ddot{\theta} = \tilde{\tau}_\theta \tag{23e}$$

$$\ddot{\phi} = \tilde{\tau}_\phi \tag{23f}$$

donde x e y son las coordenadas en el plano horizontal, y z es la posición vertical. ψ es el ángulo de guiñada alrededor del eje z, θ es el ángulo de cabeceo alrededor del eje y, ϕ es el ángulo de alabeo alrededor del eje x. Las variables de entrada u, $\tilde{\tau}_\psi$, $\tilde{\tau}_\theta$ y $\tilde{\tau}_\phi$ son la variable de entrada del empuje total (en dirección alejada a la porta superior de la aeronave) y los nuevos momentos angulares (momento de guiñada, momento de cabeceo y momento de alabeo.

Como se describe en el desarrollo de este modelo, este se divide en coordenadas traslacionales y rotacionales. En el caso de la plataforma hover tenemos la plataforma fija en un punto del espacio por lo tanto no se tienen en cuenta las traslacionales y solo se tienen en cuenta las coordenadas de rotación. Esto quedara más clarificado en la descripción del modelo matemático del Hover.

2.1.2. Hover de Quanser (helicóptero de 4-rotores con base fija)

La plataforma HOVER se obtuvo mediante compra a la empresa QUANSER(que actualmente es la empresa líder mundial en el diseño y construcción de sistemas avanzados para controle en tiempo real para el uso en

industria, educación e investigación), con el que obtuvimos, la aplicación QUARC que es un sistema en tiempo real para matlab. Con esta aplicación se ha trabajado, en conjunción con matlab para realizar todas las pruebas que se han desarrollado con el sistema operativo windows.

El 3 DOF Hover es un desafío de control que tiene como objetivo estudiar el comportamiento del vuelo de aeronaves sin tener que volar. El sistema consiste en un cuerpo con cuatro hélices. Este cuerpo está montado sobre un pilar que tiene 3 grados de libertad, lo que permite que el cuerpo gire nos ejes Roll, Pitch y Yaw. El control sobre el hover se consigue realizando un control diferencial del empuje generado por cada uno de los motores con las hélices.

- El movimiento de cabeceo se controla incrementando y decrementando la potencia de los motores delantero y trasero, por ejemplo.
- El movimiento de alabeo se controla incrementando y decrementando la potencia de los motores izquierdo y derecho, por ejemplo.
- El movimiento de guiñada se controla incrementando y decrementando la potencia del grupo de motores izquierdo y derecho y frontal y trasero.

Con el sistema de tiempo real Quarc se poseen las siguientes características:

1. "Sistema tiempo real" Matlab-Simulink Quad RC.
2. 3 grados de libertad (ψ, θ, ϕ).
3. Medición de los ángulos utilizando encoders.
4. Programación por bloques (Simulink).
5. 4 señales de control(V_r, V_l, V_f, V_b) con valores entre $\pm 10V$.

Figura 22: Esquema de funcionamiento plataforma HOVER con Quarc

Con el sistema de tiempo real brindado por Xtratum+Partikle el funcionamiento es realmente parecido al de Rt-Linux con la diferencia que al ser un hard real-time se puede asegurar su funcionamiento bajo cualquier nivel de carga de trabajo del ordenador.

Con el sistema de tiempo real Rt-Linux el sistema tiene las siguientes características:

1. Utilización de la central inercial (ángulos y velocidades angulares).
2. Periodos de muestreo muy pequeños (1ms), lo que permite un ajuste de ganancias mas fino y rápido.
3. La ley de control tiene a tener una respuesta mas fina.
4. Ley de control debe programarse con código C++

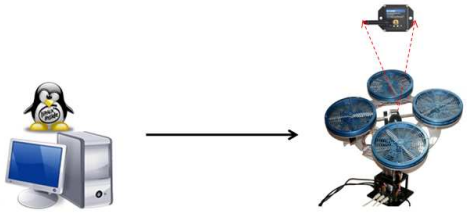


Figura 23: Esquema de funcionamiento plataforma HOVER con Rt-linux

Modelo Matemático Hover

Variable	Variable en Matlab	Descripción
$K_{t,n}$	Ktn	Constante de rotación normal de la hélice
$K_{t,c}$	Ktc	Constante de contra rotación de la hélice
K_f	Kf	Constante de la fuerza de empuje la hélice
l	L	Distancia entre el pivote hasta cada motor
J_y	Jy	Momento de inercia sobre el eje Yaw
J_p	Jp	Momento de inercia sobre el eje Pitch
J_r	Jr	Momento de inercia sobre el eje Roll
g	G	Constante de la gravedad
R_m	Rm	Resistencia de la armadura del motor
$K_{t,m}$	Kt_ m	Constante corriente-torque del motor
m_{hover}	m_ hover	Masa total que puede moverse en el sistema Hover
m_{prop}	m_ prop	Masa total de cada hélice, incluso el motor, protección, etc

Para obtener el modelo, primero es necesario considerar las fuerzas que actúan en el sistema, Las cuales se describen en la figura 24

Las ecuaciones para los grados de libertad Pitch y Roll son prácticamente las mismas. El movimiento en estos ejes es realizado por la diferencia entre las fuerzas generadas por los motores en las extremidades de cada eje, en el caso del eje Roll se puede considerar que el movimiento positivo ocurre cuando el motor Right hace una fuerza arriba y el motor Left hace una fuerza abajo, entonces, de forma sencilla se puede decir que el eje roll comportase de la siguiente manera:

$$I_r(\frac{dt}{dt^2}r) = lK_f(V_r - V_l) \tag{24}$$

Y, como el eje Pitch tiene un comportamiento casi igual, se substituye el motor Right por el Motor Front y el motor Left por el motor Back, entonces:

$$I_p(\frac{dt}{dt^2}p) = lK_f(V_f - V_b) \tag{25}$$

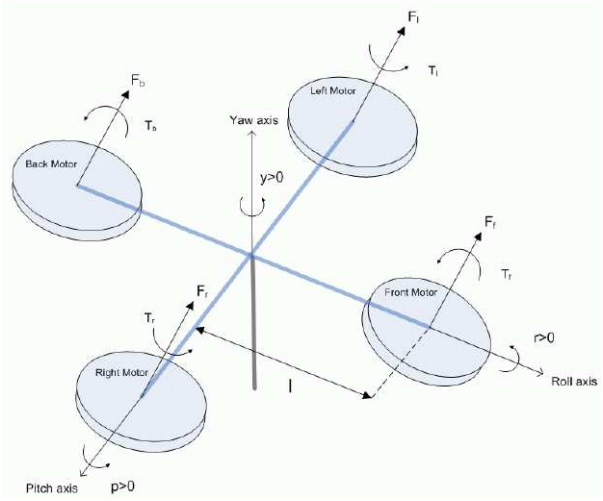


Figura 24: Modelo Matemático del Hover

Para conseguir el modelo del eje Yaw se tiene que considerar que todos los motores hacen una fuerza de torque que genera el movimiento en el eje Yaw:

$$I_y(\frac{dt}{dt^2}y) = (\tau_f + \tau_b + \tau_r + \tau_l) \tag{26}$$

Las hélices Left y Right generan un torque normal, y las hélices de los motores Front y Back generan un torque contra-normal, por tanto, para realizar un movimiento positivo del eje Yaw se alimentan los motores Left y Right con voltaje positivo y los motores Front y Back con voltaje negativo.

$$I_y(\frac{dt}{dt^2}y) = K_{t,c}(V_f + V_b) + K_{t,c}(V_l + V_r) \tag{27}$$

Con estos datos ya se puede escribir el modelo del hover en la forma de ecuaciones de estado.

$$\frac{d}{dt}x = Ax + Bu \tag{28}$$

$$\frac{d}{dt}y = Cy + Du \tag{29}$$

Con el estado:

$$x^T = [y, p, r, \frac{d}{dt}y, \frac{d}{dt}p, \frac{d}{dt}r] \tag{30}$$

La entrada de control:

$$u^T = [V_f, V_b, V_r, V_l] \tag{31}$$

Y las salidas medidas:

$$y^T = [y, p, r] \tag{32}$$

Utilizando las ecuaciones de movimiento y estos estados se puede obtener las matrices de espacio-estado abajo.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{33}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{K_{t,c}}{J_y} & \frac{K_{t,c}}{J_y} & \frac{K_{t,n}}{J_y} & \frac{K_{t,n}}{J_y} \\ \frac{lK_f}{J_p} & -\frac{lK_f}{J_p} & 0 & 0 \\ 0 & 0 & \frac{lK_f}{J_r} & -\frac{lK_f}{J_r} \end{bmatrix} \tag{34}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{35}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{36}$$

2.1.3. X-UFO(helicóptero de 4-rotores de vuelo libre)

El X-UFO es un helicóptero impulsado por cuatro rotores que trata de obtener, mediante el balanceo de las fuerzas producidas por los rotores, planeo y vuelo preciso(ver 25). El helicóptero es por supuesto una aeronave muy versátil. Difiere marcadamente en apariencia de una aeronave de ala fija pero debe operar de acuerdo a las mismas leyes aerodinámicas. No obstante, mientras que el helicóptero puede, a veces, parecer rudimentario es de hecho un vehículo sofisticado que impone grandes demandas sobre sus diseñadores. De la manera que se miren, los helicópteros son una clase única de maquinas voladoras.

Los mini-helicópteros son una clase de UAVs que ha ocupado la atención de los investigadores durante las

Figura 25: Helicóptero Cuatrirotor.

últimas décadas. Su versatilidad y maniobrabilidad, le confieren unas excelentes cualidades, para diversas aplicaciones en las cuales se necesite realizar vuelo estacionario. El helicóptero con cuatro motores, actualmente es la configuración más utilizada, esto se debe, a que posee la misma naturaleza no lineal del helicóptero clásico y los mismos efectos aerodinámicos, pero con la diferencia de que es más fácil de controlar debido a que los torques están bien definidos.

Figura 26: Esquema de transmisión de señales en la plataforma de vuelo libre

La plataforma posee el siguiente método de transmisión de datos.

1. Sistema en tiempo real (Rt-Linux)
2. Utilización de tarjetas de salida PCI
3. Inicializa tarjetas de salida

4. Lee las señales provenientes del joystick y condiciona la señal en un rango de O a 5 V.
5. Inicializa central inercial (IMU) y obtiene los ángulos de Euler y la velocidades angulares del helicóptero.
6. Calcula la ley de control propuesta
7. Envía la ley de control a las tarjetas de salida.

2.2. Sistemas operativos de tiempo real

Se necesita que el control se realice de forma constante ya que el helicóptero es una plataforma que fácilmente se puede volver inestable. Por ello el sistema de control externo debe funcionar con un sistema de tiempo real.

Un sistema en tiempo real (STR) es aquel sistema digital que interactúa activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad, controlabilidad y alcanzabilidad.

Los sistemas en tiempo real están presentes en nuestra vida diaria, prácticamente en todo lo que nos rodea; en los aviones, trenes y automóviles; en el televisor, la lavadora o el horno de microondas, en los teléfonos celulares y en las centrales telefónicas digitales. Son un elemento imprescindible para garantizar la generación, transmisión y distribución de la energía eléctrica y para asegurar la calidad y la seguridad de incontables procesos industriales.

La principal característica que distingue a los STR de otros tipos de sistemas es el tiempo de interacción. Sin embargo, antes de continuar es necesario aclarar el significado de las palabras "tiempo" y "real". La palabra "tiempo" significa que el correcto funcionamiento de un sistema depende no sólo del resultado lógico que devuelve la computadora, también depende del tiempo en que se produce ese resultado. La palabra "real" quiere decir que la reacción de un sistema a eventos externos debe ocurrir durante su evolución. Como una consecuencia, el tiempo del sistema (tiempo interno) debe ser medido usando la misma escala con que se mide el tiempo del ambiente controlado (tiempo externo). Un STR tiene tres condiciones básicas:

1. Interactúa con el mundo real (proceso físico),
2. emite respuestas correctas y
3. cumple restricciones temporales.

En contraste con la definición de STR, un sistema rápido produce su salida sin considerar las restricciones de tiempo del ambiente con que interactúa, para esa clase de sistemas no es importante el tiempo en el cual los datos llegan al sistema digital sino solamente el tiempo en que la salida es producida, en otras palabras únicamente interesa la rapidez de dar la respuesta dentro del intervalo de tiempo cuya medida, entre más pequeña es mejor, sin importar el costo de generar esa respuesta.

2.2.1. QUARC

El quarc no es un sistema operativo de tiempo real, si no una aplicación de matlab preparada para hacer que matlab funcione en tiempo real(básicamente nos permite conocer los tiempos en lo que trabajara matlab para adecuarlos a nuestras necesidades).

Esta aplicación ha sido desarrollada por la empresa Quanser y se adquirió junto con el HOVER, puesto que, para poder trabajar con este es necesario un sistema de tiempo real y este funciona de forma correcta.

Las ventajas de esta aplicación son su sencillez tanto de instalación como de uso, para alguien acostumbrado al manejo de matlab resulta bastante intuitivo. Sus desventaja es que, de las tres opciones de trabajo en tiempo

real que se van a mostrar en este documento, es la menos fiable. Puesto que al ser solo una aplicación el enviar señales hasta el exterior debe pasar por varios procesos del sistema operativo, los cuales no son de tiempo real.

Instalación del sistema operativo

En este subapartado se explicará la manera de instalar los programas necesarios para hacer control del Hover en Tiempo Real con el software de *Quarc*. Programas necesarios:

- Matlab R2007a o R2007b
  - Simulink
  - Real Time workshop
- MS Visual Studio o C++2005(8.0) Professional o Express Edition
- QuaRC

Para simplificar su instalación se ha notado que los programas se reconocen entre si, al ser instalados en el siguiente orden:

1. **MS Visual Studio o C++2005** En caso de instalar MS visual Studio sera necesario contar con todas las librerías para C++.
2. **Matlab 2007** Al instalar Matlab en segundo puesto permite que este reconozca como compilador al MS visual Studio. También sera necesario instalar SImulink y la toolbox "Real Time Workshop". Se sugiere realizar la siguiente comprobación en matlab,
   a) ejecutar el comando "mex -setup".
   b) Esto provocara la aparición de las siguientes lineas: Please choose your compiler for building external interface (MEX) files:

   Would you like mex to locate installed compilers [y]/n? y

   Select a compiler: [1] Lcc-win32 C 2.4.1 in C:\ARCHIV 1\MATLAB\R2007a\sys\lcc [2] Microsoft Visual C++ 2005 in C:\Archivos de programa\Microsoft Visual Studio 8 [0] None

   Compiler:

   c) Pulsar el numero "2" en el teclado.
   d) Esto provocara la aparición de las siguientes lineas: Please verify your choices:

   Compiler: Microsoft Visual C++ 2005 Location: C:\Archivos de programa\Microsoft Visual Studio 8

   Are these correct?([y]/n):

   e) Pulsar la tecla "y".
3. QuaRC Una vez instalado los programas anteriores, ya se podrá instalar "QuaRC", que de forma automática ya reconfigura todo el sistema (incluido el Matlab).

Para realizar una última comprobación se puede ejecutar el programa "demo quarc". En este programa se detalla todos los pasos a seguir para la validación de la instalación de QuaRC.

2.2.2. Rt-linux

RTLinux es un sistema operativo de tiempo real que ejecuta Linux como un thread (hilo de ejecución) de menos prioridad que las tareas de tiempo real. Con este diseño, las tareas de tiempo real y los manejadores de interrupciones nunca se ven retrasados por operaciones que no son de tiempo real.

La primera versión de RTLinux estaba diseñada para ejecutarse en la plataforma x86 y proporcionaba una pequeña API y un pequeño entorno de programación. La versión 2, que fue totalmente reescrita, fue diseñada para el soporte de multiprocesamiento simétrico (SMP) y para ser ejecutada en una amplia variedad de arquitecturas.

RTLinux proporciona la capacidad de ejecutar tareas de tiempo real y manejadores de interrupciones en la misma máquina que el Linux estándar. Estas tareas y los manejadores ejecutan cuando se necesitan en detrimento de lo que estuviera ejecutando Linux. El peor caso de tiempo es entre que se detecta la interrupción hardware y el procesador ejecuta la primera instrucción del manejador de la interrupción. Este tiempo es del orden de los 10 microsegundos en la plataforma x86.

Cabe destacar que este sistema operativo es un sistema de tiempo real *soft*. Esto implica que ,según sus parámetros, la finalización de una tarea fuera de plazo no se considera inservible, solo se considera una reducción de la calidad del servicio(por ejemplo, en un televisor se entregaría la imagen con menos frames).

Instalación del sistema operativo

La instalación de Rt-Linux es bastante sencilla y existen multitud de manuales en la red que se pueden seguir.A modo de resumen los pasos genéricos a seguir serian:

1. Obtención de la distribución RTLinux y el código fuente de Linux
2. Copia del código fuente y la distribución
3. Comprobar si la versión de la distribución de RTLinux soporta la versión del kernel que se desea configurar.
4. Parcheado del Kernel
5. Configuración del Kernel
6. Configuración del Gestor de arranque
7. Reiniciar el ordenador
8. Configurar RTLinux
9. Compilar RTLinux
10. FIN

2.2.3. Xtratum y Partikle

Xtratum hipervisor
El xtratum es un hipervisor, esto significa que es un programa que permite el funcionamiento de varios sistemas operativos de forma concurrente en el mismo ordenador. Se ha desarrollado en la Universidad Politécnica de Valencia con contribuciones de la Universidad de Lanzhou(China). Esta especialmente diseñado para sistemas empotrados y para sustituir el HAL(Hardware Abstraction Layer) del Rtlinux.

Básicamente el xtratum esta compuesto por dos "device drivers" : Interrupciones y reloj. Normalmente un sistema operativo esta estructurado en una serie de bloques: Interfaz de usuario, "drivers",manejo de memoria, estructura virtual de archivos, pila de red. En este escenario, Xtratum se puede considerar como una parte del nivel mas bajo del sistema operativo.

La consecuencia práctica de controlar interrupciones y temporizaciones de hardware es que Xtratum posee el control completo del sistema. Esto significa que Xtratum opera como un sistema supervisor el cual es ejecutado a la prioridad más alta. Xtratum maneja un "sistema operativo cliente" de una manera similar a la que cualquier sistema operativo maneja un proceso: El sistema operativo puede crear, suspender, matar, etc. cualquier proceso.

Xtratum provee una API simple y uniforme para utilizar las interrupcions y los relojes sin necesitar tener en cuenta los sistemas hardware libres reales a través de técnicas de virtualización. Con lo cual, Xtratum puede ejecutar varios dominios de forma concurrente.

Aunque pueda parecer que Xtratum es un nivel de código complejo y pesado que retardara la ejecucion del OS cliente. De echo la complejidad de Xtratum esta más relacionada a la programación de él Hardware. Ha sido diseñado para ser rápido y tener una respuesta atada a un orden de tiempo para poder ser utilizado por sistemas de tiempo real *hard*.

Para poder hacer funcionar con varios dominios de forma concurrente, cada dominio tiene que estar adaptado a la infraestructura del Xtratum. En particular, la secuencia de encendido se tiene que eliminar y las llamadas a las interrupciones y los "drivers" de tiempo se tienen que redirigir a llamadas a la API de Xtratum.

El Partikle es un kernel de tiempo real de código abierto para sistemas empotrados. Ha sido distribuido bajo los términos de la licencia pública de GNU. El kernel posee programación de hilos, sincronización, reloj y comunicaciones elementales. Maneja recursos de hardware, tales como interrupciones, excepciones, memorias, relojes, etc..

Partikle fue diseñado para ser compatible con POSIX. El API nativo es "C" hilos de POSIX. Pero también soporta C++, ADA y Java(tareas, sincronización objetos protegidos, manejo de excepciones, etc.)

Utilizando estos dos programas se puede obtener en un ordenador un sistema funcionando con Rt-linux y con Xtratum para su uso según convenga. El uso de Partikle+Xtratum proporciona un sistema Operativo de tiempo real *hard* lo cual nos da la mayor seguridad sobre los tiempos de control. Sin embargo su uso es restringido ya que no posee compatibilidad con tarjetas PCI(solo ISA).

Instalación del sistema operativo

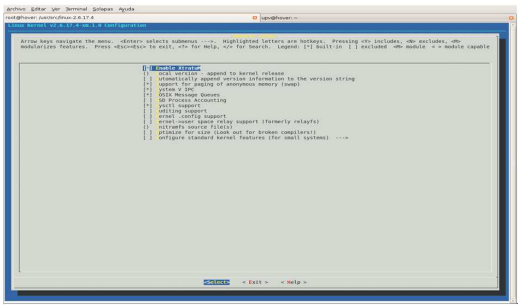
Puesto que la instalación del Xtratum+Partikle no fue sencilla se va a realizar una explicación detallada de la instalación de este sistema.
La instalación se divide en dos etapas:

1. Parchear el núcleo de linux 2.6.17.4(configurarlo, compilarlo e instalarlo)
2. Instalar Xtratum+Partikle sobre el núcleo 2.6.17.4.

Los pasos a realizar seran:

1. Se inicia una sesión de Linux (cualquier variante de incio en el grub) en la máquina Hover
2. Desde la línea de comandos (consola) se demanda acceso como root.
   `# su`
3. Se sitúa la consola en el directorio `/usr/src`
   `# cd /usr/src`

4. Descompactar fuentes linux-2.6.17.4.tar.bz2 `# tar -xvf linux-2.6.17.4.tar.bz2`
5. Copiar las fuentes de Xtratum + Partikle (xmertl)
   `cp /Loclización-Xtratum/xmertl . -r`
6. Se situa la consola en el directorio de las fuentes `# cd /usr/src/linux-2.6.17.4/`
7. Parchear instalación
   a) Con el parche oficial de linux
   `patch -p1 </usr/src/xmertl/xtratum/patches/i386-use-c-code-for-current_thread_info.patch`
   b) Con el parche XtratuM
   `patch -p1 </usr/src/xmertl/xtratum/patches/xtratum_linux_2.6.17.4.patch`
8. Configurar la instalación `Make menuconfig`
   Se deben tener escogidas las siguientes opciones:



   a) En General setup: Enable XtratuM
   b) En Processor type and features :
      1) Symmetric multi-processing support [OFF]
      2) Preemption Model (No Forced Preemption (Server))
      3) Local APIC support on uniprocessors [OFF]
      4) High Memory Support (off)
      5) Memory model (Flat Memory)
      6) Use register arguments [OFF]
   c) En Power management options: deshabilitarlo todo
   d) Modifica lo que se crea conveniente
9. Recompilamos el núcleo
   `# make-kpkg version=custom kernel_image kernel_headers`
10. Nos pasamos al directorio /usr/src `# cd /usr/src/`
11. Instalamos el núcleo
    `# dpkg -i linux-image-2.6.17.4-xm.1.0_custom.1.0_i386.deb`
12. Una vez terminado el proceso, reiniciar

3. Técnicas de Control Utilizadas

3.1. Estado del Arte del control de helicópteros Autónomos

A continuación se expondrán las principales técnicas de control propuestas para helicópteros autónomos en los últimos años. Es posible encontrar en la literatura algunas enfoques simples como el control PID clásico o la utilización de estrategias lineales, cuya implantación ha dado buenos resultados. la consideración de las incertidumbres y la minimización del efecto de las perturbaciones externas ha llevado a la formulación de estrategias de control más sofisticadas.

3.1.1. Estrategias Lineales

Con la utilización de estrategias estrictamente lineales y deterministas, no es suficiente para controlar la dinámica no lineal tan compleja como la de un helicóptero. De esta forma algunas referencias proponen la conmutación entre distintos conmutadores lineales según el punto de funcionamiento que se encuentre el sistema, otras se basan en el regulador lineal con índice cuadrático LQR y su extensión para considerar los ruidos gaussianos LQG.

Así en [88] proponen un control lineal conmutado que se compara con el rendimiento de un LQR. En [69] se presenta otro controlador en el espacio de estados, en este caso un LQG. En línea con la conmutación de controladores lineales, en [44] y [43] se asocia una ley de control a cada régimen de vuelo (lateral,vertical,...). Los algoritmos empleados son control por realimenración y control robusto. Otras aplicaciones en control óptimo cuadrático se presentan en [84] para controlar en posición y orientación un helicóptero. En [28] se emplean también los controladores LQR y LQG con términos integrales. Finalmente cabe citar los trabajos de [83] que plantean la comparación de un controlador lineal robusto multivariable con estrategias más avanzadas de control no lineal.

3.1.2. Linealización por Realimentación

Es una de las técnicas más utilizadas como base de numerosos trabajos. Esta técnica consiste en aplicar una transformación no lineal para obtener un sistema resultante lineal sobre el que se aplican técnicas clásicas de control como PIDs.

De esta forma, [52] convierten la dinámica del helicóptero en un conjunto de 4 subsistemas lineales controlables de una única entrada sobre las que se aplican estrategias básicas lineales. También [37] proponen una linealización por realimentación para realizar un control en altitud y guiñada. Asimismo, se pueden encontrar trabajos en esta línea aplicados al helicóptero Yamaha RMAX, como en [71]. En el conocido trabajo de [10] se propone la utilización conjunto de la linealización por realimentación y redes neuronales. El objetivo de las redes es compensar el error de inversión producido en la linealización por realimentación, asociado mayormente a posibles discrepancias entre el modelo y el sistema real.

3.1.3. Control Robusto

Existen bastantes referencias de control robusto en la bibliografia. Los esquemas típicos de H_2 y H_∞ se basan en la optimización de índices con las normas del mismo nombre.

En [20] se describe un esquema mixto, controlador PID para ángulos y dinámica traslacional regulada mediante control H_∞. Con tal fin se diseñan cuatro funciones de transferencia que tengan la forma definida por las

especificaciones frecuenciales. En la misma línea, pero haciendo uso de la versión no lineal del control H_∞, [96] garantizan el rechazo de perturbaciones necesario para la estabilidad. Otro uso de la variedad no lineal de H_∞ es el desacoplo entre las dinámicas traslacionales y rotacionales, tal y como se muestra en [97].

De gran interés por su carácter experimental son los trabajos de [12], [13] y [14] en la Carnegie Mellon University en los que desarrolló un controlador H_∞ para el control del Yamaha R-50. Los resultados muestran el rendimiento del controlador en maniobras de elevada velocidad con dicho helicóptero. Existen otras alternativas de control robusto que también han sido exploradas por los investigadores. En [15] se consideran también estrategias robustas frente a incertidumbres en los parámetros. En [59], [61], [31] y [7] se presentan leyes no lineales de control robusto basadas en modelo interno y en leyes de saturación. La realización de maniobras agresivas o el aterrizaje autónomo en la cubierta de un buque son algunas de las posibles aplicaciones.

3.1.4. Algoritmos de Aprendizaje

En [5] se emplean modelo de máxima similitud de Markov y algoritmos clásicos de reforzamiento basados en modelo para calcular los criterios óptimos para aprendizaje. Asimismo, [75] plantean la adición de una precompensación a un módulo PID que controla la guiñada. Dicha precompensación se hace mediante modelos borrosos del tipo Takegi-Sugeno, que aprenden a sintonizarse automáticamente en línea. Por último, en [68], se propone una estructura de control neuro-borrosa generada y sintonizada mediante un algoritmo de aprendizaje para emular el comportamiento de un operador humano mientras pilota un helicóptero.

3.1.5. Algoritmos Genéticos

Los algoritmos genéticos también han sido empleados para el ajuste de controladores de helicópteros autónomos. Como ejemplo puede mencionarse el trabajo de [73], donde leyes de evolución genética sirven para generar reglas eficientes para la lógica borrosa de control. En [92], de nuevo se incorporan estos algoritmos para sintonizar reglas borrosas y conjuntos lingüísticos que mejoren el rendimiento de control.

3.1.6. Leyes adaptativas

Se emplean métodos que tienen como objetivo adaptar el modelo y el controlador al efecto de las variaciones del entorno y en particular del viento.

Los trabajos de [46] y [47] presentan un control adaptativo para hacer frente a las incertidumbres ambientales y de modelado. Se consiguen buenos resultados incluso para maniobras generales de velocidad arbitraria y con un gran ancho de banda. En [26] se plantea la complementariedad entre las redes neuronales y los mecanismos adaptativos para su ajuste. Se demuestra que el esquema propuesto supera al del enfoque clásico de los controladores PI. En [3] y [8] se aplica el control adaptativo como modelo de referencia para filtrar las trayectorias propuestas y adecuarlas así a la dinámica del helicóptero.

3.1.7. Redes Neuronales

Como se comentaba en apartados anteriores, el empleo de redes neuronales hace posible la aplicación de métodos adaptativos que permiten obtener un comportamiento robusto frente a incertidumbres de modelado o incluso de las condiciones del entorno, como puede ser el efecto del viento.

La aplicación de las redes neuronales a helicópteros autónomos se presentó en [42], donde se planteaba el uso combinado de la linealización por realimentación y redes neuronales. Estas últimas se utilizaban tanto para la

transformación correspondiente a la linealización como para compensar los errores de modelado que pudieran aparecer.

En [50], se estudia el esfuerzo de control requerido por la técnica, que resulta ser leve y sólo perceptible en el primer periodo de entrenamiento de la red. Hasta el momento, los resultados se habían quedado en simulación. Sin embargo, en [16] se presentan ya trabajos experimentales. Existen también aplicaciones en simuladores de alta fidelidad de helicópteros militares con en [51]. Continuando en esta línea, [74] y [48] presentan algunas mejores como la utilización de PCH(Pseudo Control Hedging) mencionando anteriormente para tener en cuenta la adaptación en los actuadores.

En [33] se presentan refinamientos adicionales como la técnica de ubicación de polos para mejorar la separación de la dinámica en dos escalas de tiempo. En [72] se presenta la apliación de las redes neuronales de control del Yamaha RMAX. Además en [18] y [19] se emplea la denominada programación dinámica neuronal de redes directas (DNDP), generándose un sistema de control capaz de realizar maniobras complejas. En este primer caso se empleó un simulador avanzado del militar helicóptero Apache.

3.1.8. Métodos de Liapunov

Existen también trabajos en los que se diseñan controladores garantizando la estabilidad mediante los métodos de Liapunov. Así [57] utilizan funciones de Liapunov para conseguir acotaciones en las prestaciones del seguimiento de trayectorias en la aceleración lineal y en sus derivadas. En [40] se utiliza el método directo de Liapunov para demostrar la estabilidad del controlador de un helicóptero de 2 grados de libertad situado en una plataforma robótica, demostrándose la estabilidad semiglobal en el seguimiento de la consigna actitud. En [65] se presentan técnicas de adelanto para generar una función de control de Liapunov que estabiliza exponencial (localmente) y asintóticamente (globalmente) de forma robusta al sistema completo. Asimismo, [46] demuestran la estabilidad global del sistema de control multivariable adaptativo que proponen para desarrollar maniobras generales de velocidad arbitrarias y con prestaciones de gran ancho de banda. Finalmente, cabe mencionar los trabajos de [24] que presentan un análisis de estabilidad de dinámica rápida correspondiente a un esquema de control con dos escalas de tiempo, dinámicas traslacional (lenta) y rotacional (rápida), de uso típico en vehículos aéreos autónomos, empleando también una función de Liapunov.

3.2. LQR Espacio de estados

La teoría de control óptimo se basa en operar un sistema dinámico con el coste mínimo. El caso en el que el sistema dinámico es definido mediante un grupo de ecuaciones diferenciales lineales y el coste es descrito por una función cuadrática es denominado problema LQ. Para trabajar con realimentación del estado con esta técnica de control se tiene que asumir que el total de los estados de x pueden se medidos(ver figura 27) y, por lo tanto, pueden ser utilizados para el control. En esta configuración, el modelo del proceso del espacio de estados es de

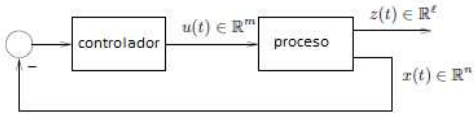


Figura 27: LQR con realimentación del estado

la forma:

$$\dot{x} = Ax + Bu, \quad y = Cx, \quad z = Gx + Hu \tag{37}$$

El controlador optimo para un LQR con realimentación de espacio de estados es una simple matriz de ganancia igual a:

$$u = -Kx \tag{38}$$

donde K es la matiz mxn dada por

$$K = (H'QH + \rho R)^{-1}(B'P + H'QG) \tag{39}$$

y P es la única solución definida positiva de la ecuación siguiente:

$$A'P + PA + G'QG - (PB + G'QH + \rho R)^{-1})(B'P + H'QG) = 0 \tag{40}$$

A esta ecuación se la conoce como la *ecuación algebraica Riccate* (ARE).

La ley de control de realimentacón del estado 38, se convierte en un sistema de bucle cerrado con la forma:

$$\dot{x} = (A - BK))x \tag{41}$$

Una propiedad crucial del diseño del controlador LQR es que este bucle cerrado es asimptoticamente estable mientras que se cumplan las dos condiciones siguientes:

- El sistema 37 es controlable
- El sistema 37 es observable cuando se ignora y se toma z como la única salida.

Incluso más importante que es el echo de que los controladores LQR son inherentemente robusto respecto a las incertidumbres del proceso. Para entender por que, considerar la matriz de transferencia de bulce abierto desde la entrada del proceso u a la salida del controlador $\overline{u}$, ver figura 28 El modelo del espacio de estados de u hasta $\overline{u}$ nos viene dado por:

$$\dot{x} = Ax + Bu, \quad \overline{u} = -Kx, \tag{42}$$

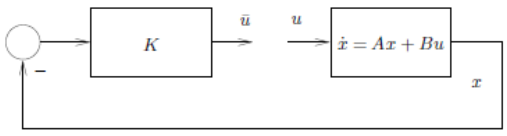


Figura 28: LQR con realimentación del estado con ganancia en bucle abierto

Implementación en el Proyecto

Como ya se ha indicado en el Apéndice 3 el modo de trabajo con cualquier tipo de control se puede dividir en varias fases:

1. Calculo matemático del controlador
2. Simulación en código matlab.
3. Simulación en simulink continua.
4. Simulación en simulink discreta.
5. Implementación en Quarc.
6. Implementación en Linux.

El calculo matemático de un LQR se basa en encontrar una Q y una R para ponderar los valores de la K. En este caso, con un modelo:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{43}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{K_{t,c}}{J_y} & \frac{K_{t,c}}{J_y} & \frac{K_{t,n}}{J_y} & \frac{K_{t,n}}{J_y} \\ \frac{lK_f}{J_p} & -\frac{lK_f}{J_p} & 0 & 0 \\ 0 & 0 & \frac{lK_f}{J_r} & -\frac{lK_f}{J_r} \end{bmatrix} \tag{44}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{45}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{46}$$

La Q y la R que se han utilizado son:

$$Q = \begin{bmatrix} 150 & 0 & 0 & 0 & 0 & 0 \\ 0 & 250 & 0 & 0 & 0 & 0 \\ 0 & 0 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{47}$$

$$R = \begin{bmatrix} 0{,}005 & 0 & 0 & 0 \\ 0 & 0{,}005 & 0 & 0 \\ 0 & 0 & 0{,}005 & 0 \\ 0 & 0 & 0 & 0{,}005 \end{bmatrix} \tag{48}$$

Al realizar los cálculos se obtiene que la K obtenida sera:

$$K = \begin{bmatrix} -86{,}6025 & 158{,}1139 & 0 & -36{,}4479 & 37{,}0589 & 0 \\ -86{,}6025 & -158{,}1139 & 0 & -36{,}4479 & -37{,}0589 & 0 \\ 86{,}6025 & 0 & 158{,}1139 & 36{,}4479 & 0 & 37{,}0589 \\ 86{,}6025 & 0 & -158{,}1139 & 36{,}4479 & 0 & -37{,}0589 \end{bmatrix} \tag{49}$$

De esta K solo se desea utilizar la parte que controla el roll para lo cual nos quedamos con el valor que controla los motores izquierdo y derecho con las salidas de velocidad y posición de roll. Entonces la "Kroll" sera:

$$Kroll = \begin{bmatrix} 158{,}1139 & 37{,}0589 \\ -158{,}1139 & -37{,}0589 \end{bmatrix} \tag{50}$$

Obtenido el controlador que se desea utilizar se realizaran las tres simulaciones a la vez(matlab, simulink continuo y simulink discreto). El resultado se podrá observar en la figura 29 se puede observar que las tres señales coinciden completamente. En la implementación real esto no ocurrirá así puesto que el seguimiento de la

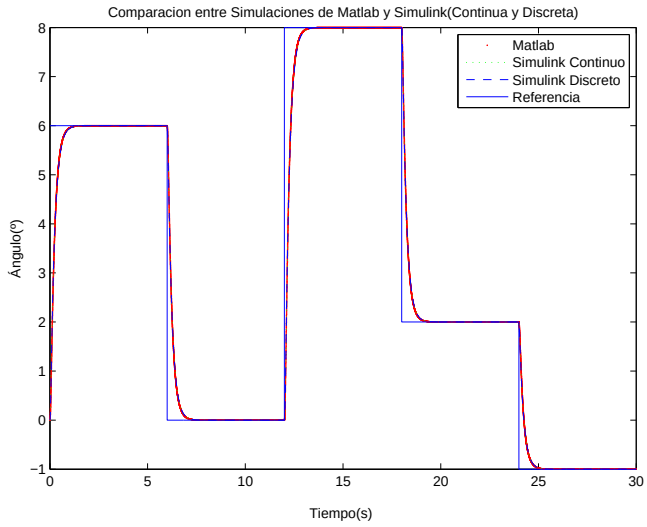


Figura 29: Gráfica de controlador LQR en simulación

referencia que se realiza con el LQR es, en ultima instancia un PD, lo cual generara cierto error de posición. Todo esto se puede confirmar en la figura 30. Se puede observar que existe sobreoscilación y error de posición. Esto es debido a que el sistema es no lineal y que el modelo utilizado(doble integrador) no es del todo correcto(aunque es bastante aproximado).

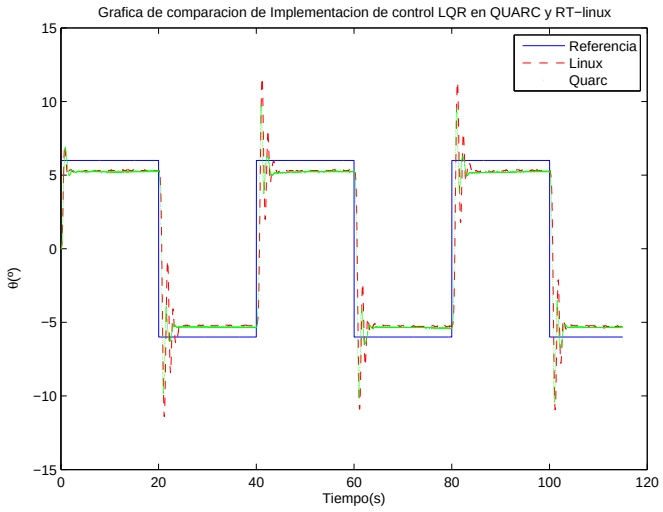


Figura 30: Gráfica de controlador LQR implementado

3.3. Lógica borrosa

La lógica borrosa (Fuzzy Logic) ha surgido como una herramienta lucrativa para el control de subsistemas y procesos industriales complejos, así como también para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos. Aunque la lógica borrosa se inventó en Estados Unidos el crecimiento rápido de esta tecnología ha comenzado desde Japón y ahora nuevamente ha alcanzado USA y también Europa. La lógica borrosa es todavía un boom en Japón, el número de cartas patentando aplicaciones aumenta exponencialmente. Principalmente se trata de aplicaciones más bien simples de lógica borrosa.

Lo borroso ha llegado a ser una palabra clave para vender. Los artículos electrónicos sin componetes borrosos se están quedando gradualmente desfasados. Como una mordaza, que muestra la popularidad de la lógica borrosa, cada vez es más frecuente un sello con "fuzzy logic" impreso sobre el producto. En Japón la investigación sobre lógica borrosa es apoyada ampliamente con un presupuesto enorme. En Europa y USA se están realizando esfuerzos para alcanzar al tremendo éxito japonés. Por ejemplo, la NASA emplea lógica borrosa para el complejo proceso de maniobras de acoplamiento.

La lógica borrosa es básicamente una lógica multievaluada que permite valores intermedios para poder definir evaluaciones convencionales como sí/no, verdadero/falso, negro/blanco, etc. Las nociones como "más bien caliente" o "poco frío" pueden formularse matemáticamente y ser procesados por computadoras. De esta forma se ha realizado un intento de aplicar una forma más humana de pensar en la programación de computadoras. La lógica borrosa se inició en 1965 por Lotfi A. Zadeh, profesor de ciencia de computadoras en la Universidad de California en Berkeley.

Si bien se ha hecho ya referencia a algunos trabajos que incorporan algún tipo de lógica borrosa, en esta sección se presentan las aportaciones más específicas.

La aplicación del control borroso a los helicópteros autónomos tiene su origen en los trabajos propuestos por Sugeno desde principios de los años 90 [85] y [86]. Mediante una organización jerárquica en módulos básicos (control de cola, etc.) que son activados por modos de alto nivel (asociados a regímenes de vuelo, como puede ser el vuelo frontal), las estrategias borrosas planteadas posibilitan la ejecución de maniobras complejas, como giros combinados con elevaciones. También son conocidas las aportaciones de [36], [34] y [35]. En este caso se considera

un controlador borroso que consigue maniobrabilidad agresiva de forma robusta y estable. La estructura de control consta de un planificador de ganancias y controlador lingüístico de tipo Mamdani. Finalmente en [80] y [81] se combinan las leyes clásicas de PID con técnicas de control borroso para mimetizar la forma en que un piloto experto maneja un helicóptero.

Implementación en el Proyecto

Se han realizado una serie de simulaciones de un sistema de lógica borrosa sobre el modelo del helicoptero. Para poder hacerlo, se preparon funciones de pertenencia para las dos entradas al controlador(error de posición y error de velocidad), con un rango de acción de $\pm 10^{o}$ para la posición y de $\pm 30^{o}/s$ para la velocidad. Las funciones de pertenencia se pueden observar en las figuras 31 y 32. La idea detrás de estas funciones de transferencia era evitar un movimiento de la posición(por eso el solape de las funciones en el centro de la figura 31) mientras se permite un poco más de holgura con la velocidad, para no saturar las acciones de control.

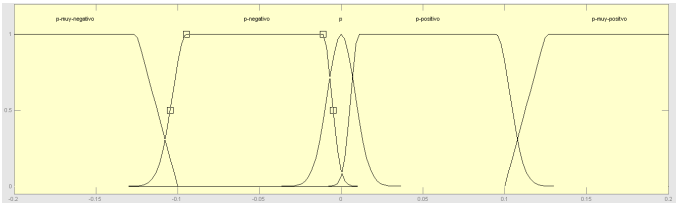


Figura 31: Funciones de pertenencia para el error de posición

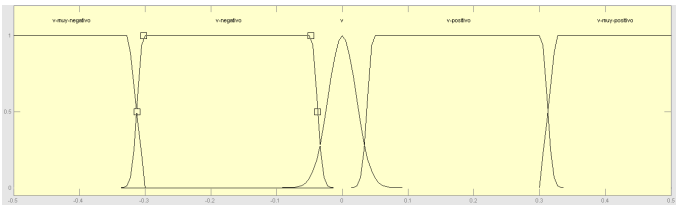


Figura 32: Funciones de pertenencia para el error de velocidad

También, se generaron funciones de pertenencia para la acción de control resultante, esta se puede observar en la figura 33

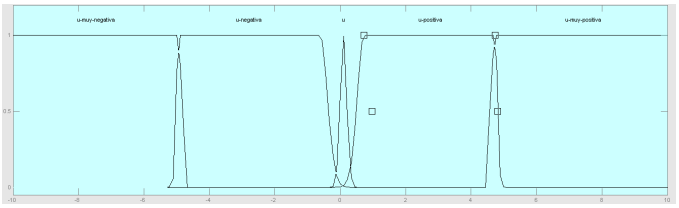


Figura 33: Funciones de pertenencia para la acción de control

Se puede observar que la terminología para las funciones de transferencia es idéntica en todos los casos, el propósito de esto es poder realizar una tabla de reglas fácil de comprender y utilizar. Por ello primero se abreviara la terminología de la siguiente manera:

- error de posición
  - p P
  - p-muy-negativo PMN
  - p-negativo PN
  - p-positivo PP
  - p-muy-positivo PMP
- error de velocidad
  - v V
  - v-muy-negativo VMN
  - v-negativo VN
  - v-positivo VP
  - v-muy-positivo VMP
- Acción de control
  - u U
  - u-muy-negativo UMN
  - u-negativo UN
  - u-positivo UP
  - u-muy-positivo UMP

Ahora se expondrá la serie de reglas que conforman la relación de reglas que conforman el control borroso.

	VMP	VP	V	VN	VMN
PMP	UMP	UMP	UMP	UMP	UP
PP	UMP	UMP	UP	UP	U
P	UMP	UP	U	UN	UMN
PN	U	UN	UN	UMN	UMN
PMN	UN	UMN	UMN	UMN	UMN

Al realizar esta simulación se comparo directamente con el sistema LQR realizado con anterioridad. Los resultados se pueden observar a continuación.

Como se puede observar el sistema LQR realiza una mejor actuación que el sistema fuzzy, sin embargo, el ajuste de las funciones de pertenencia se puede realizar de una manera más efectiva. Queda como futuro trabajo realizar un sistema más optimo y comprobar si la respuesta en lógica borrosa puede mejorar el LQR.

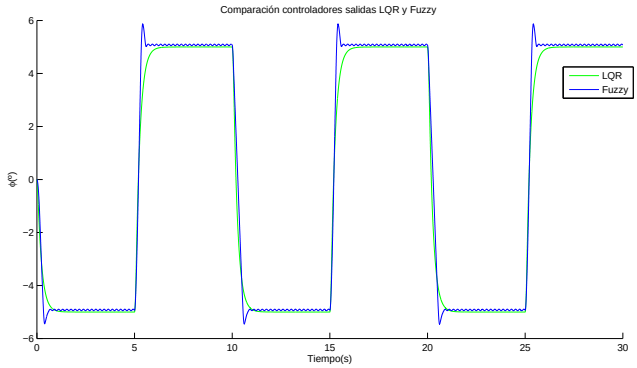


Figura 34: Comparación de controles LQR y fuzzy

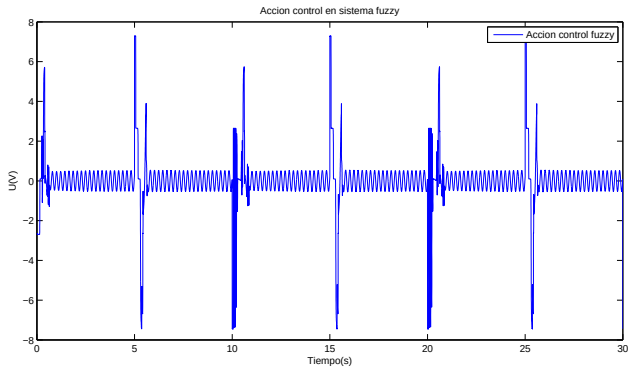


Figura 35: Acción de control del sistema fuzzy

3.4. Control Predictivo

El control predictivo también ha sido propuesto como alternativa para controlar UAVs, si bien el número de trabajos es bastante limitado y en muchas casos se restringe a modelos simplificados de la dinámica del modelo.

En particular en [17] se presenta un controlador predictivo no lineal aplicado a un modelo de 2 grados de libertad, en donde las predicciones sobre las futura señales de control se utilizan para calcular las predicciones sobre el estado del sistema. [95] presentan un controlador predictivo no lineal multivariable, basado en la linealización del sistema en cada instante de muestreo que aplican aun modelo simplificado de 3 grados de libertad. Puede mencionarse también el trabajo de [93] con un modelo completo de la dinámica que combina una red neuronal con el control basado en la ecuación de Ricatti dependiente del estado. En este caso se utiliza el control predictivo con horizonte deslizante para optimizar la red neuronal.

El control predictivo tiene como objetivo resolver de forma efectiva, problemas de control y automatización de procesos industriales que se caractericen por presentar un comportamiento dinámico complicado, multivariable, y/o inestable. La estrategia de control en que se basa este tipo de control, utiliza el modelo matemático del proceso a controlar para predecir el comportamiento futuro de dicho sistema, y en base a este comportamiento futuro puede predecir la señal de control futura.

El control predictivo integra disciplinas como el control óptimo, control estocástico, control de procesos con

retardo de tiempo, control multivariable, control con restricciones.

El tipo de control utilizado en esta tesina, es el Control Predictivo Basado en Modelo (CPBM), conocido también como Model Based Predictive Control (MBPC) o simplemente Model Predictive Control (MPC). Esta estrategia también se conoce como control por horizonte deslizante, por ser ésta la forma en la que se aplican las señales de actuación. Existen muchos algoritmos de control predictivo que han sido aplicados con éxito: GPC, IDCOM, DMC, APC, PFC, EPSAC, RCA, MUSMAR, NPC, UPC, SCAP, HPC, etc.

El control predictivo basado en modelo se puede definir como una estrategia de control que se basa en la utilización de forma explícita de un modelo matemático interno del proceso a controlar (modelo de predicción), el cual se utiliza para predecir la evolución de las variables a controlar a lo largo de un horizonte temporal de predicción especificado por el operador, de este modo se puede calcular las variables manipuladas futuras (señal de control futura) para lograr que en dicho horizonte, las variables controladas converjan en sus respectivos valores de referencia.

El MPC se enmarca dentro de los controladores óptimos, es decir, aquellos en los que las actuaciones responden a la optimización de un criterio. El criterio a optimizar, o función de coste, está relacionado con el comportamiento futuro del sistema, que se predice gracias a un modelo dinámico del mismo, denominado modelo de predicción.

El intervalo de tiempo futuro que se considera en la optimización se denomina horizonte de predicción. Dado que el comportamiento futuro del sistema depende de las actuaciones que se aplican a lo largo del horizonte de predicción, son éstas las variables de decisión respecto a las que se optimiza el sistema. La aplicación de estas actuaciones sobre el sistema conduce a un control en bucle abierto.

La posible discrepancia entre el comportamiento predictivo y el comportamiento real del sistema crean la necesidad de imponer cierta robustez al sistema incorporando realimentación del mismo. Esta realimentación se consigue gracias a la técnica del horizonte deslizante que consiste en aplicar las actuaciones obtenidas durante un periodo de tiempo, tras el cual se muestrea el estado del sistema y se resuelve un nuevo problema de optimización. De esta manera, el horizonte de predicción se va deslizando a lo largo del tiempo.

Una de las propiedades más atractivas del MPC es su formulación abierta, que permite la incorporación de distintos tipos de modelos de predicción, sean lineales o no lineales, monovariables o multivariables, y la consideración de restricciones sobre las señales del sistema. Esto hace que sea una estrategia muy utilizada en diversas áreas del control. El CPBM es una de las pocas técnicas que permiten controlar sistemas con restricciones incorporando éstas en el propio diseño del controlador.

Estas características han hecho del control predictivo una de las escasas estrategias de control avanzado con un impacto importante en problemas de ámbito industrial Por tal motivo es importante resaltar que el control predictivo se ha desarrollado en el mundo de la industria, y ha sido la comunidad investigadora la que se ha esforzado en dar un soporte teórico a los resultados prácticos obtenidos.

Merece la pena destacar que el control predictivo es una técnica muy potente que permite formular controladores para sistemas complejos y con restricciones. Esta potencia tiene un precio asociado: el coste computacional y la sintonización del controlador. Recientes avances en el campo del MPC proveen un conocimiento más profundo de estos controladores, obteniéndose resultados que permiten relajar estos requerimientos. Así por ejemplo, se han establecido condiciones generales para garantizar la estabilidad ([63]), condiciones bajo las cuales se puede relajar el carácter optimal del controlador garantizando su estabilidad ([82]).

Implementación en el Proyecto

Se han realizado simulaciones en matlab con control predictivo, para poder realizarlo se ha necesitado utilizar la toolbox de “Model Predictive Control Toolbox”, el funcionamiento se basa en darle al controlador un modelo del helicóptero para que el pueda prever cual tiene que ser la acción de control antes de ejecutarla. En el desarrollo

de un control predictivo, a parte del modelo, es necesario ver que horizonte de predicción y de control, estos datos son el limite en el cual el control va a dejar de predecir muestreos más alejados y el limite de señales predichas en el cual el control va a empezar a realizar modificaciones en las variables de entrada a la planta. En nuestro caso, los horizontes se situaron en 20 (horizonte de predicción) y 10(horizonte de control).

Los resultados obtenidos con este control se pueden observar en 36 y en 37

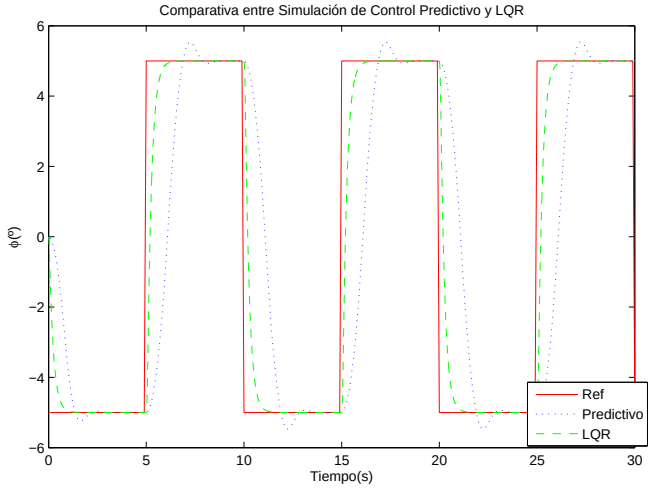


Figura 36: Comparación de controles LQR y Predictivo

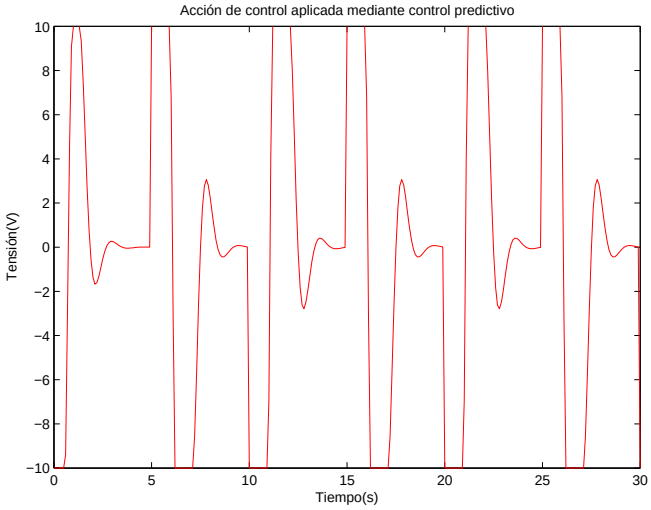


Figura 37: Acción de control con control Predictivo

Se puede observar que la respuesta es menos optima que la del control LQR, esto da pie a considerar este control como mejor, sin embargo, como trabajo futuro, se podría obtener un control predictivo más fino.

3.5. Modo Deslizante

En teoría de control, los modos deslizantes son una forma de control de estructura variable. Es un método de control no lineal que altera la dinámica de un sistema no lineal aplicando un control alternante, ver [98]. La ley de control no es una función constante en el tiempo. En cambio, va alternando de una estructura continua a otra basándose en la posición en ese instante del espacio de estados. Las estructuras de control múltiple se diseñan para que las trayectorias siempre se muevan hacia la condición de alternancia, y, así, la trayectoria última no existirá completamente en una sola estructura de control. Lo que ocurrirá es que la trayectoria última se deslizara por la frontera entre las diferentes estructuras de control.

La mayor ventaja del control por modo deslizante es su robustez. Ya que el control puede ser tan simple como un salto entre dos estados, no necesita ser preciso y no se vera afectado a variaciones de parámetros que entren en el canal de control. Además, como la ley de control no es una función continua, el modo deslizante se puede alcanzar en tiempo finito, ver [49].

El control en modo deslizante debe ser aplicado con mucho más cuidado que otras formas de control no-lineal que poseen una acción de control mas moderada. En particular, por que los actuadores tienen retardos y otras imperfecciones, el modo de control deslizante extremo puede llevar a chattering, perdida de energía, daño en la planta y excitación de dinámicas no modeladas, ver [41].

En el tema de esta tesis de máster cabe mencionar el trabajo de Pieper (1995) en el que se discuten los métodos de diseño de hiperplanos para obtener controladores por modos deslizantes. Asimismo, en Shin *et al.*(2002) se propone este tipo de controladores para el cálculo de trayectorias.
Un ejemplo muy básico del uso de la ley de estructura variable esta dado por:

$$u(t) = \begin{cases} 1, \text{si } s(y, \dot{y}) > 0 \\ -1, \text{si } s(y, \dot{y}) < 0 \end{cases} \tag{51}$$

donde la función de conmutación está dada por:

$$s(y, \dot{y}) = my + \dot{y} \tag{52}$$

donde es un escalar positivo. La razón para el uso del término "función de conmutación" es claro, si la función dada en la ecuación 52 es usada para decidir que estructura de control se pondrá en uso para cualquier punto en el plano de fase. La expresión de la ecuación 51 es escrita usualmente como:

$$u(t) = -sgn(s(t)) \tag{53}$$

Ejemplo Practico

Supongase que el motor de posicionamiento se emplea en un sistema de producción seriada en el cual es deseable alcanzar la posición deseada en el tiempo mínimo posible. Es sabido que las estrategias de control de tiempo mínimo requieren entregar la máxima potencia de control (en un sentido o en otro) así como una adecuada conmutación entre ambas. Tómese como ejemplo el siguiente sistema (Figura 38).

La conmutación entre máxima (u_{max}) y mínima (u_{min}) potencia de control se realiza según el signo del error de posición. En la Figura 1.12 se representa la respuesta temporal del sistema, con la excitación variando entre su máximo y su mínimo valor. Se puede observar que el sistema no alcanza la referencia, sino que oscila en torno a ella.

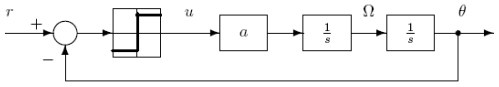


Figura 38: Doble Integrador

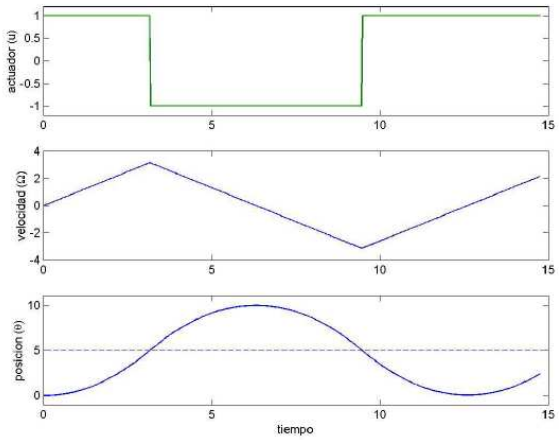


Figura 39: Respuesta del motor controlado con relé

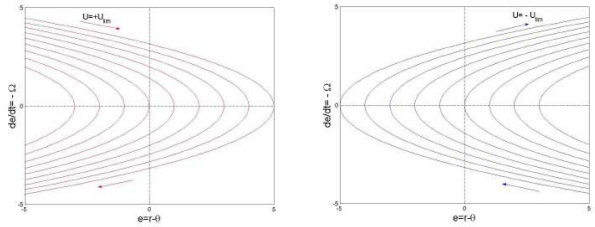


Figura 40: Planos de fase para u=+1(izquierda) u=-1(derecha)

Analizando el sistema en el plano de fase, y tomando como máxima y mínima acción de control $u_{min} = -u_{lim} = -1$ y $u_{max} = u_{lim} = 1$ respectivamente, se obtiene:

$$u = 1 \longrightarrow \begin{cases} \Omega = t + \Omega_0 \\ \theta = \frac{t^2}{2} + \Omega_0 t + \theta_0 \end{cases} \tag{54}$$

$$u = -1 \longrightarrow \begin{cases} \Omega = -t + \Omega_0 \\ \theta = -\frac{t^2}{2} + \Omega_0 t + \theta_0 \end{cases} \tag{55}$$

Tomando como variables de estado del sistema el error y su derivada:

$$e = r - \theta \tag{56}$$

$$e = -\Omega \tag{57}$$

$$u = 1 \longrightarrow \begin{cases} \dot{e} = -t+ \\ dote(0) \\ e = -\frac{t^2}{2} + \dot{e}(0)t + e(0) \end{cases} \tag{58}$$

Las ecuaciones del sistema en función de los estados son:

$$u = -1 \longrightarrow \begin{cases} \dot{e} = t+ \\ dote(0) \\ e = \frac{t^2}{2} + \dot{e}(0)t + e(0) \end{cases} \tag{59}$$

Eliminando el tiempo de las ecuaciones del sistema y despejando el error se llega a la expresión:

$$u = 1 \longrightarrow e = -\frac{\dot{e}^2}{2} + \frac{e(0) + \dot{e}(0)^2}{2} = -\frac{\dot{e}^2}{2} + cte \tag{60}$$

$$u = -1 \longrightarrow e = \frac{\dot{e}^2}{2} + cte \tag{61}$$

Las expresiones anteriores corresponden a parábolas en el plano de fase, que dependen de las condiciones iniciales. Las figuras de 40 representan ambas curvas para diferentes valores de e(0) y $\dot{e}(0)$.

Para mostrar el efecto de la utilización del esquema de control de la figura 38, se superponen ambos planos de fase (ver). Si se conmuta el relé cuando el error de posición se hace nulo(e=0), se obtiene una respuesta oscilatoria(como ya se indica en la en la figura 39), y no se alcanza el punto de equilibrio. (e, $\frac{de}{dt} = (0, 0)$).
El problema de las oscilaciones se solventa si se conmuta antes de alcanzar el error nulo. En la figura se muestra el resultado de conmutar a u_{min} cuando se alcanza $\theta = \frac{\pi}{2}$. Esto requiere desplazar la recta de conmutación en función del valor de referencia. Una mejor alternativa es conmutar cuando se alcanza la parábola que pasa por el origen, como se muestra en la figura para distintas condiciones iniciales.

Implementación en el Proyecto

Para comprobar las posibilidades de los modos deslizantes se pensó en implementar un control de este tipo sobre el Hover. Se implemento una simulación con el modelo teórico del Hover para comprobar su viabilidad. Los resultados fueron satisfactorios. El sistema en simulación respondía correctamente(ver figura 41). Sin embargo, al implementar el control al Hover real, el control no era del todo correcto, ya que, aunque sigue la referencia entraba y salia del modo deslizante sin llegar a estabilizarse(ver figura 42). La razón para ello se concreto que era una mala sensorización. En la implementación de Quanser del Hover sus sensores son los encoders instalados en cada ángulo, estos dan información sobre la posición del Hover y, si los derivas, tienes la información de la velocidad. Para hacer esta derivada Quanser dispone de un modulo ya implementado. Sin embargo, aunque para realizar muchos tipos de control la derivada que otorga este modulo es mas que aceptable, para un control extremo como es el de modos deslizantes, este valor no es lo suficientemente correcto para poder ser utilizado con eficacia por el controlador. Para solucionarlo se pensó en la implementación en el sistema Rt-Linux, puesto que con él, se tenia implementado un nuevo sensor -La central Inercial- Este aparato, como se explica en otros apartados, es, básicamente tres acelerometros, los cuales nos entregan directamente la señal de velocidad(sin necesidad de ningún otro aparato). Sin embargo con la IMU(central inercial) surgió otro problema y es que los valores de velocidad y posición que otorga la IMU tienen un retardo.

Por esta razón la implementación final del control en modos deslizantes se a postergado para futuros trabajos ya que sera necesario la implementación de un predictor.

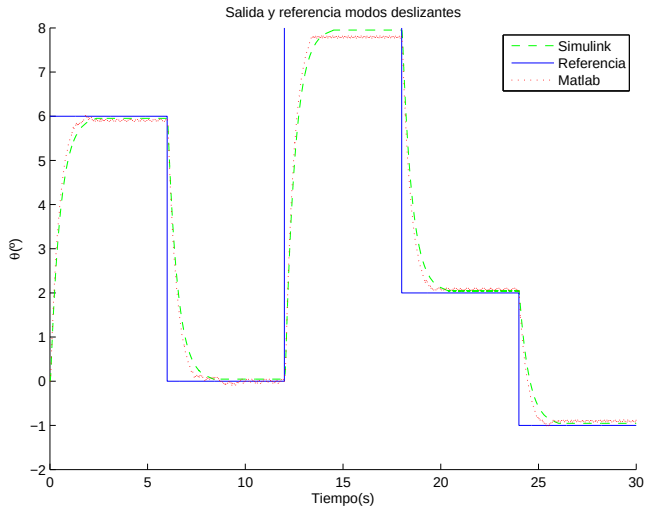


Figura 41: Simulación del sistema con un control en modo deslizante

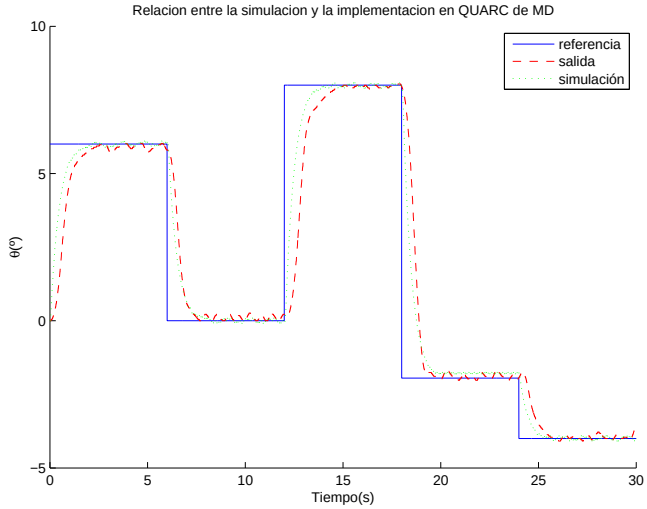


Figura 42: Implementación del sistema con un control en modo deslizante

3.6. Saturaciones Anidadas

Las saturaciones anidadas es un tipo de control no lineal que se ha utilizado de forma generalizada para la estabilización de PVTOL diversas. Este tipo de control se propuso por [89], con un teorema no lineal de pequeña ganancia. Este control ha sido muy fructífero. Primero permitió resolver problemas teóricos(ver [90],[87]) y aplicados([91]) y, segundo, es el origen de una nueva metodología llamada "forwarding"(ver, por ejemplo [64],[32]).

Desde un punto de vista estructural, para sistemas descritos por una cadena generalizada de integradores lin-

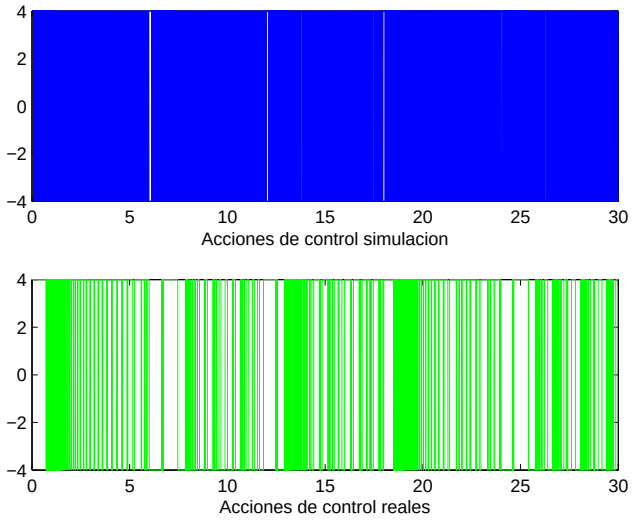


Figura 43: Acciones de control de simulacion e implementacion del modo deslizante

eales(ver [89],[87] y [53]) o no lineales(ver [60] y [25]), la ley de control consiste en una generalización del esquema de las saturaciones anidades de [89] o una combinación lineal de saturaciones [87]. Es también interesante ver [4], en el cual el esquema de las saturaciones anidadas de Teel se robustece en contra de dinámicas no modeladas.

Recientemente, en los autores [81] y [79] han propuesto un algoritmo de control basado en funciones de saturación separadas. En este tipo de controlador la saturación depende de diversas variables de estado. Los autores han comparado simulaciones de este algoritmo con el de Teel. Los resultados muestran que las saturaciones separadas es, generalmente, más eficiente que el de Teel. Además, cabe destacar, que las ganancias y limites del controlador de Teel son complejos de establecer.

El interés en estudiar sistemas con saturaciones proviene, principalmente de el echo que, debido a restricciones físicas, de seguridad o tecnológicas los actuadores no pueden proveer señales de amplitud o de velocidad de reacción ilimitadas. Esto significa que los sistemas de control están, en general, sujetos a saturaciones de los actuadores en amplitud y en dinámica. Tener estas limitaciones en control, y no tenerlas en cuenta, puede ser la fuente de comportamientos indeseables o, incluso, catastróficos, como la inestabilidad.

En particular, es interesante estudiar el tipo de estructuras que aparecen cuando se trabaja con sensores o actuadores no lineales. Por ejemplo, son bastante comunes en los sistemas de control aeroespacial, ya que los actuadores están limitados, tanto en amplitud como en frecuencia(dinámica). La presencia de estas limitaciones puede llevar a un sistema en bucle cerrado que presente saturaciones anidadas. Este sera el caso en sistemas controlados por dinámica mediante controladores mediante “feedback” a la salida, con sautraciones de sensores y actuadores. Por otra parte, el análisis y diseño de metodologías para sistemas que presentan saturaciones anidadas pueden ser útiles para investigar la estabilidad de clases más generales de sistemas. Por ejemplo, el uso de saturaciones anidadas es muy interesante cuando uno utiliza técnicas de forwarding para sistemas en cascada con parte lineal.

Implementación en el Proyecto

En primer lugar, hay que decir que el *X-Ufo* posee su propio sistema de control. Para poder realizar un control en tiempo real al margen del que el *X-Ufo* ya posee, se ha reducido el control propio del *X-Ufo* tanto como ha sido posible pero sin hacer nulos los valores de las ganancias del control propio del *X-Ufo*. Esto es debido a que se trata de un sistema con control en cascada y al hacer nulo el bucle interno, no es posible mediante el bucle externo realizar acciones de control sobre la aeronave. Por tanto, haciendo la hipótesis que los valores de las ganancias del control interno del *X-Ufo* son suficientemente pequeños pero no nulos, se desarrollará a continuación la estrategia de control realizada para conseguir la estabilización de la aeronave.

La estrategia de control propuesta es relativamente simple, pero esta propiedad es interesante cuando es utilizada en aplicaciones en tiempo real. Así mismo, las cuatro variables de control pueden funcionar tanto en modo manual como automático. Como se puede observar anteriormente en la ecuaciones (23) del apartado anterior se asociará una entrada de control a cada uno de los grados de libertad correspondientes a la estabilización de la aeronave en vuelo libre. Por ello, se despreciarán los grados de libertad relacionados con los desplazamientos en los ejes x e y. A continuación, se muestra la relación entre las entradas de control asociadas a cada grado de libertad:

- La variable de control u es utilizada para situar a la aeronave en la altura deseada.
- La variable de control $\tilde{\tau}_\psi$ es utilizada para controlar el ángulo de guiñada (*yaw angle*).
- De la misma forma, la variable de control $\tilde{\tau}_\theta$ es utilizada para controlar el ángulo de cabeceo (*pitch angle*)
- Análogamente, la variable de control $\tilde{\tau}_\phi$ es utilizada para controlar el ángulo de alabeo (*roll angle*)

Algoritmo de control del *Yaw*, *Pitch* y *Roll*.

Dadas las ecuaciones 23, tenemos para cada eje un sistema con doble integrador.

La estabilización de cadenas de integradores ha sido bastante estudiada estos últimos años. Así, la estabilización de una cadena de integrador mediante una entrada limitada con retraso [65], [67] o sin [89] fue estudiada. Son estos últimos trabajos ([89]) los que permitieron resolver el problema (abierto hasta aquí) de cadenas de integradores de dimensión superior estrictamente a dos. Gracias a estos resultados, problemas teóricos (ver [90], [87]) pero también mas aplicados (ver [91]) fueron solucionados en seguida. Ademas, el trabajo de [89] están al origen del método de concepción de leyes de control llamado *forwarding* (ver por ejemplo [64], [32]).

De manera general, para cadenas de integradores lineales ([53]) o non lineales ([60], [25]), la ley propuesta consistía en una generalización del esquema con saturaciones encajadas de [89] o en una combinación lineal de saturaciones introducidas por [87]. Particularmente, una ley basada en estos últimos trabajos fue propuesta por [58], quien introdujo funciones con limites variables. Eso es interesante porque permite a un estado aprovechar de una reserva de saturación, disponible gracias a los estados de nivel superior que todavía no están saturados

Un control limitado para une clase de sistemas no lineales de tipo *feedforward* fue propuesta por [38] y [39], ocupando funciones de saturación separadas. Mas reciéntemente, [81] y [79] hicieron un algoritmo basado en el mismo principio para estabilizar un avión de tipo *PVTOL* (*Planar Vertical Take Off and Landing*). Esos resultados inspiraron a [78] quien hizo una generalización para sistemas de orden n. Ademas la ley de [78] tiene como ventaja utilizar funciones donde cada estado es separado. Así, el ajuste de ganancias es mucho mas fácil en experimentos. La leyes que describimos en esta parte representa un caso simple donde $n = 2$.

Entonces, el ángulo del *yaw* puede ser controlado mediante la aplicación de la siguiente ley de control

$$\psi_{\text{control}} = -\sigma_{\psi_2} - \sigma_{\psi_1} \tag{62}$$

siendo

$$\sigma_{\psi_2} = f_{\text{saturación}}(k_{d_\psi}\dot{\psi}, \zeta_{\psi_2}) \tag{63a}$$
$$\sigma_{\psi_1} = f_{\text{saturación}}\Big(k_{p_\psi}(\psi - \psi_d), \zeta_{\psi_1}\Big) \tag{63b}$$

donde k_{d_ψ}, k_{p_ψ}, ζ_{ψ_2}, ζ_{ψ_1} son constantes positivas y ψ_d es la referencia de angulo. Teniendo en cuenta que $f_{\text{saturación}}$ es una función de saturación definida de la manera siguiente:

$$f_{\text{saturación}}(s, \zeta) = \zeta, \text{ cuando } : s > \zeta \tag{64a}$$
$$f_{\text{saturación}}(s, \zeta) = s, \text{ cuando } : -\zeta \leq s \leq \zeta \tag{64b}$$
$$f_{\text{saturación}}(s, \zeta) = -\zeta, \text{ cuando } : s < -\zeta \tag{64c}$$

Ademas, para asegurar la estabilidad global, se necesita (ver [78])

$$\zeta_{\psi_2} > \zeta_{\psi_1} \tag{65a}$$
$$k_{d_\psi}^2 > k_{p_\psi} \tag{65b}$$

De forma análoga para los ángulos de *pitch* y *roll* se obtiene

$$\theta_{\text{control}} = -\sigma_{\theta_2} - \sigma_{\theta_1} \tag{66}$$
$$\phi_{\text{control}} = -\sigma_{\phi_2} - \sigma_{\phi_1} \tag{67}$$

siendo

$$\sigma_{\theta_2} = f_{\text{saturación}}(k_{d_\theta}\dot{\theta}, \zeta_{\theta_2}) \tag{68a}$$
$$\sigma_{\theta_1} = f_{\text{saturación}}\Big(k_{p_\theta}(\theta - \theta_d), \zeta_{\theta_1}\Big) \tag{68b}$$

y

$$\phi_{\text{control}} = -\sigma_{\phi_2} - \sigma_{\phi_1} \tag{69}$$

siendo

$$\sigma_{\phi_2} = f_{\text{saturación}}(k_{d_\phi}\dot{\phi}, \zeta_{\phi_2}) \tag{70a}$$
$$\sigma_{\phi_1} = f_{\text{saturación}}\Big(k_{p_\phi}(\phi - \phi_d), \zeta_{\phi_1}\Big) \tag{70b}$$

cumpliendose de nuevo que

$$\zeta_{\theta_2} > \zeta_{\theta_1} \tag{71a}$$
$$k_{d_\theta}^2 > k_{p_\theta} \tag{71b}$$
$$\tag{71c}$$

y

$$\zeta_{\phi_2} > \zeta_{\phi_1} \tag{72a}$$
$$k_{d_\phi}^2 > k_{p_\phi} \tag{72b}$$

de forma que se asegura la estabilidad global del sistema.

Resultados experimentales

El principal objetivo de este trabajo, consiste en la validación experimental de la plataforma desarrollada. Para este fin, se ha desarrollado una serie de experimentos. Las primeras pruebas consistieron en estabilizar la orientación del helicóptero a una altura cercana al suelo (distancia fija por la estructura de peonza), una vez ajustadas las ganancias, se procedía a la etapa final, que consistía en realizar el vuelo libre controlando de forma autónoma la orientación y solo de forma manual la altura del helicóptero.

Estabilización del mini helicóptero en la configuración de peonza.

Dadas las características y finalidad de la estructura de peonza. El objetivo de estos experimentos, consistía en el ajuste y validación de los parámetros de control, para su posterior ejecución en vuelo libre.

El objetivo de estos experimentos consistían en de realizar el control en orientación del mini-helicóptero. Dada la rápida dinámica del helicóptero y por cuestiones de seguridad, esta etapa fue realizada en dos partes; primero se realizó el control de *pitch* y *roll*, una vez estabilizados se pasó a estabilizar el ángulo de *yaw*. Los valores de ángulos deseados son: $\theta_d = \phi_d = 0°$, la dinámica de *yaw* es estabilizada utilizando la siguiente configuración:

$$\psi_d = \begin{cases} 0° & t < 40s \\ 40° & 40s \leq t < 45s \\ -20° & 45s \leq t < 53s \\ 0° & 53s \leq t < 83s \\ 40° & 83s \leq t < 88s \\ -20° & t > 88s \end{cases} \tag{73}$$

Las ganancias en la ley de control fueron ajustadas experimentalmente hasta obtener un desempeño adecuado del helicóptero. Dichos valores son mostrados en la tabla 1. El helicóptero posée en su circuito electrónico giroscopios que utiliza para medir la velocidad angular ($\dot{\eta}$), y con el cual realiza un control interno en velocidad ($-\bar{k}_d\dot{\eta}$) que se opone a cambios rápidos en la velocidad angular. Desactivar estos componentes significaría deteriorar el desempeño dinámico del vehículo y generalmente es poco aconsejable realizar esto. Esta acción derivativa se "agrega" al controlador, generando de esta manera una ganancia total ($k_d + \bar{k}_d$) en el controlador en la parte de $\dot{\eta}$. Como el valor de $\bar{k}_d$ es desconocido, no podemos asegurar teóricamente las condiciones (65b), (71b) y (72b). Sin embargo, los valores obtenidos experimentalmente aseguran un buen desempeño del controlador en lazo cerrado.

Cuadro 1: Valores de las ganancias.

	k_p	k_d
ϕ	0.002	0.012
θ	0.002	0.012
ψ	0.002	0.026

Las Figuras 44 y 45 presentan la respuesta de los ángulos de *pitch* y *roll* en lazo cerrado cuando se aplican las leyes de control (66) y (67) respectivamente. De las gráficas se puede observar cómo a pesar de las ligeras oscilaciones el sistema se mantiene estable con un error aproximado de 3°.

La Figura 46 introduce la respuesta de la dinámica de *yaw* al aplicar la ley de control (62). Notar que los valores deseados descritos en (73) fueron enviados mediante la unidad de control terrestre y la emisora radio en los tiempos especificados al helicóptero.

Se puede observar en la Figura 46, el buen desempeño de la plataforma y de la ley de control, puesto que ante cambios introducidos manualmente, estos son enviados rápidamente para no alterar o afectar con la estabilidad

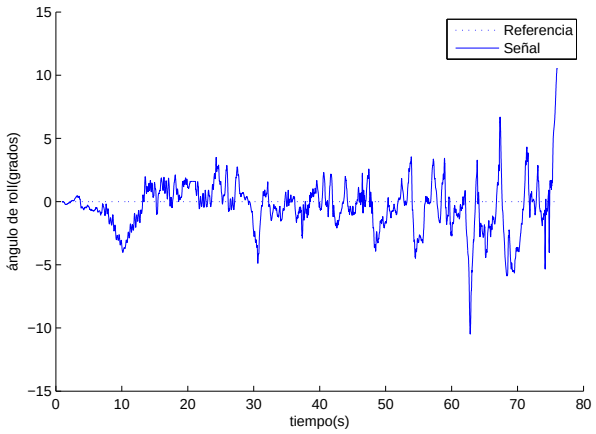


Figura 44: Respuesta del ángulo de *roll* cuando se aplica la ley de control (67). Las líneas punteadas describen el valor de ϕ_d, mientras que la línea solida representa el valor medido de ϕ.

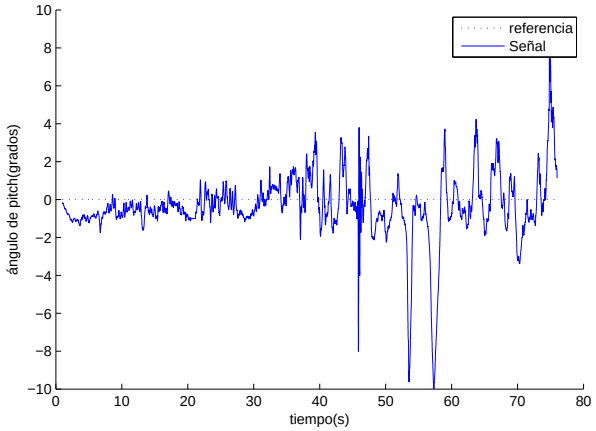


Figura 45: Respuesta del ángulo de *pitch* cuando se aplica la ley de control (66). Las líneas punteadas describen el valor de θ_d, mientras que la línea solida representa el valor medido de θ.

y dinámica del helicóptero. Notar también que la unidad receptora sirve como un *buffer* de almacenamiento de todos los eventos realizados durante el experimento.

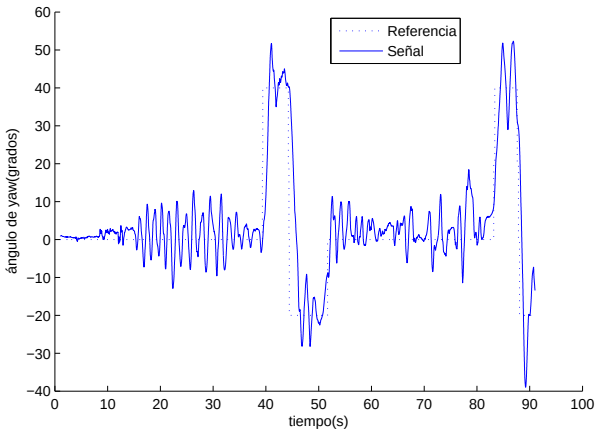


Figura 46: Respuesta del ángulo de *yaw* cuando se aplica la ley de control (62). Las líneas punteadas describen el valor de ψ_d, mientras que la línea solida representa el valor medido de ψ.

Estabilización en vuelo libre

Una vez ajustadas las ganancias del controlador utilizando la estructura de peonza, se pasa a la ejecución del sistema en vuelo libre realizando un control manual a una altura deseada. Llamamos este experimento vuelo semi-autónomo porque la orientación del helicóptero estará controlada por la unidad de control terrestre de manera autónoma, mientras que los desplazamientos traslacionales (principalmente la altura) estarán controlados por el usuario.

El objetivo es desplazar el helicóptero a una altura deseada Para mantener el helicóptero a una cierta altura sin la utilización de ningún dispositivo de control es necesario equilibrar el peso del helicóptero con el empuje de los motores. Lo anterior es claramente explicado utilizando la ecuación (23c), es decir;

$$m\ddot{z} = u\cos\theta\cos\phi - mg$$

como $\theta, \phi \approx 0$, se tiene

$$m\ddot{z} = u - mg$$

Existe por tanto un valor de empuje, u, que mantiene al helicóptero en este punto de funcionamiento. Para evitar accidentes, una vez conocido el valor de empuje que contrarresta el peso, se satura el valor del empuje a este valor como medida de seguridad. Los valores de las ganancias del controlador son los mismos que fueron obtenidos para el caso de la estructura de peonza. En la figura 47 se presenta una foto del vuelo semi autónomo del helicóptero.

En las Figuras 48 y 49 se muestran las respuestas del sistema θ y ϕ cuando se aplica la ley de control respectiva. De nuevo, se observa como el helicóptero se mantiene estable con un error de 2°. Notar que estos resultados son relativamente mejores que esos obtenidos en la configuración peonza. La razón de ello, puede estar relacionada con la inexistencia de la fricción de deslizamiento de la estructura con respecto al suelo.

Los experimentos para la dinámica de *yaw* son realizados de igual modo a como se realizaron para la configuración peonza. Los valores deseados fueron ligeramente cambiados e incrementados para verificar experimental-

Figura 47: Vuelo semi autónomo del mini helicóptero en 3D.

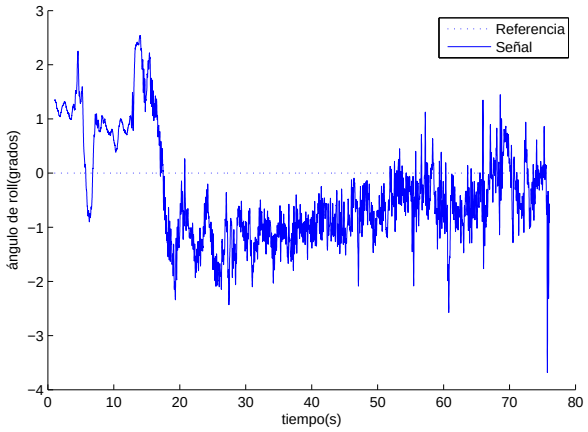


Figura 48: Respuesta del ángulo de *roll* cuando se aplica la ley de control (67). Las líneas punteadas describen el valor de ϕ_d, mientras que la línea solida representa el valor medido de ϕ.

mente la robustez de la plataforma y del controlador en lazo cerrado. Los valores seleccionados son;

$$\psi_d = \begin{cases} 0^\circ & t < 65s \\ 40^\circ & 65s \leq t < 70s \\ -20^\circ & 70s \leq t < 74s \\ 0^\circ & 74s \leq t < 85s \\ 80^\circ & 85s \leq t < 95s \\ 0^\circ & t > 95s \end{cases}$$

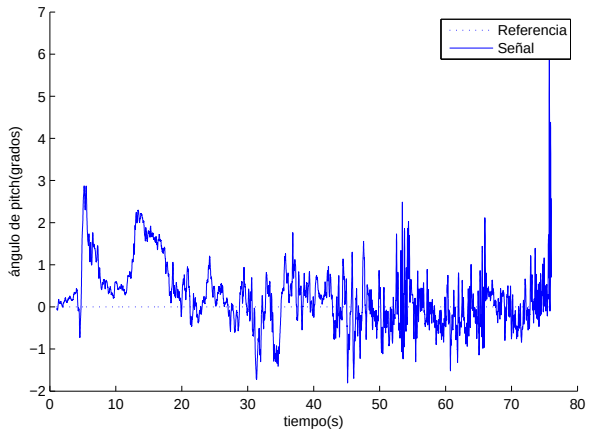


Figura 49: Respuesta del ángulo de *pitch* cuando se aplica la ley de control (66). Las líneas punteadas describen el valor de θ_d, mientras que la línea solida representa el valor medido de θ.

La gráfica correspondiente a los ensayos realizados se muestra en la Figura 50. Puede observarse de la figura anterior el buen desempeño del sistema aún ante cambios grandes en los valores deseados.

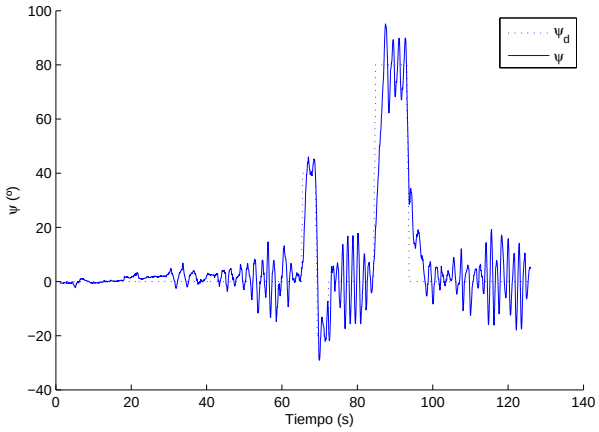


Figura 50: Respuesta del ángulo de *yaw* cuando se aplica la ley de control (62). Las líneas punteadas describen el valor de ψ_d, mientras que la línea solida representa el valor medido de ψ.

4. Conclusiones

Durante este periodo, se ha presentado una plataforma de pruebas para el control de vehículos aéreos no tripulados. Para la validación de la plataforma, se ha implementado un control para estabilizar la orientación de un mini-helicóptero de cuatro rotores. El cálculo de la ley de control, se realiza en la unidad de control terrestre, y se envía solamente, utilizando la emisora radio, las señales de control al helicóptero. El uso de un sistema de tiempo real en la unidad de control terrestre, *RT-Linux*, permite tanto garantizar los tiempos de muestreo del algoritmo de control, como hacer una manipulación *on-line* de los parámetros de control de los algoritmos desarrollados. Los resultados obtenidos, muestran la viabilidad de la plataforma proyectada.

La utilización de otros sistemas operativos, tales como *PartiKle*, en la plataforma está actualmente en etapa de validación. De forma paralela, se está trabajando en la incorporación de un sistema sensorial (sensores de altura y perimetrales) que permita referenciar al vehículo aéreo con respecto al medio en el que se encuentra. Estos sensores ya se disponen y se esta trabajando en su ajuste para poder ser utilizados como sensores de confianza, el conjunto son tres sensores de infrarojos y un sensor de ultrasonidos para la altura y un sensor láser para el perímetro. Cuando estén operativos se tendrá una plataforma completa para la implementación y validación de estrategias de navegación más complejas.

Referencias

[1] *Model Aircraft Aerodynamics*. Argus Book, Londres, 1987.

[2] L. Acosta, J. Toledo, G.N. Marichal, S. Hernández, J.N. Rodriguez, and S.Torres. Diseño e implementación de una maqueta de un helicóptero de 4 rotores para un laboratorio de control. In *XXV Jornadas de Automática*, Ciudad Real, España, 2004.

[3] A. Fradkov Andrievsky, B. and D. Peaucelle. Adaptive control experiments for a "helicopter´´ benchmark. *International Conference on Physics and Control, PhysCon*, 2005:760–765, 2005.

[4] Kokotovic PV Arcak M, Teel AR. Robust nonlinear control of feedforward systems with unmodeled dynamics. *Automatica*, 36:265–272, 2001.

[5] J. A. Bagnell and J. G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. *IEEE International Conference on Robotics and Automation*, 2:1615–1620, 2001.

[6] J. Balderud and D.I. Eilson. Application of predictive control to a toy helicopter. In *IEEE International*, Glasgow, Escocia,UK, 2002.

[7] A. Isidori L. Marconi Bejar, M. and R. Naldi. Robust vertical/lateral/longitudinal control of an helicopter with constant yaw attitude. *44th IEEE Conference on Decision and Control, and the European Control Conference*, 5:6192–6197, 2005.

[8] M. Bejar and J. C. Raimundez. Application of adaptive control to 2dof helicopter model. *RIAI, Revista Iberoamericana de Automatica e Informatica Industrial*, 4:35–40, 2007.

[9] J. Borenstein. The hoverbot: An electrically powered flying robot,"`http://www-personal.umich.edu/~johannb/hoverbot.htm`".

[10] B. S. Kim J. Leitner Calise, A. J. and J. V. R. Prasad. Helicopter adaptive flight control using neural networks. *Proceedings of the 33rd IEEE Conference on Decision and Control. Part 1 (of 4)*, 4:3336–3341, 1994.

[11] P. Castillo, R. Lozano, and A. Dzul. *Modelling and Control of Mini-Flying Machines*. Springer-Verlag in Advances in Industrial Control, juillet 2005.

[12] G. Papageorgiou W. C. Messner Civita, M. La and T. Kanache. Integrated modeling and robust control for fullenvelope flight of robotic helicopters. *IEEE International Conference on Robotics and Automation*, 1:552–557, 2003.

[13] G. Papageorgiou W. C. Messner Civita, M. La and T. Kanade. Design and flight testing of a gain scheduled h infinite loop shaping controller for widenvelope flight of a robotic helicopter. *American Control Conference*, 5:4195–4200, 2003.

[14] G. Papageorgiou W. C. Messner Civita, M. La and T. Kanade. Design and flight testing of an h infinite controller for a robotic helicopter. *Journal of Guidance, Control, and Dynamics*, 29(2):485–494, 2003.

[15] G. Papageorgiou W. C. Messner Civita, M. La and T. Kanade. Nonlinear modelling and control of helicopters. *Automatica*, 39(9):1583–1596, 2003.

[16] A. J. Calise Corban, J. E. and J. V. R. Prasad. Implementation of adaptive nonlinear control for flight test on an unmanned helicopter. *Proceedings of the IEEE Conference on Decision and Control*, 4:3641–3646, 1998.

[17] A.W. Ordys andM. J. Grimble Dutka, A. S. Nonlinear predictive control of 2 dof helicopter model. *42nd IEEE Conference on Decision and Control*, 4:3954–3959, 2003.

[18] R. Enns and J. Si. Helicopter flight control design using a learning control approach. *39th IEEE Confernce on Decision and Control*, 2:1754–1759, 2000.

[19] R. Enns and J. Si. Helicopter tracking control using direct neural dynamic programming. *International Joint Conference on Neural Networks (IJCNN'01)*., 2:1019–1024, 2001.

[20] J. Shin K. Hazawa Fujiwara, D. and K. Nonami. H infinite hovering and guidance control for autonomous smallscale unmanned helicopter. *Nippon Kikai Gakkai Ronbunshu, C Hen/Transactions of the Japan Society of Mechanical Engineers, Part C*, 70(6):1708–1714, 2004.

[21] Jack Ganssle. *The Art of Designing Embedded Systems*. NEWNES, 2008.

[22] M. García-Sanz, J. Elso, and I. Egaña. Control del Ángulo de cabeceo de un helicóptero como benchmark de diseño de controladores. *RIAI*, 3(2):111–116, 2006.

[23] H. Goldstein. *Classical Mechanics*. Addison Wesley Series in Physics. Adison Wesley, 1980.

[24] R. Mahtani M. Bejar Gonzalez, A. and A. Ollero. Control and stability analysis of an autonomous helicopter. *Robotics: Trends, Principles, and Applications, International Symposium on Robotics and Applications, ISORA, Sixth Biannual World Automation Congress*, pages 399–404, 2004.

[25] F. Grognard, R. Sepulchre, and G. Bastin. Global stabilization of feedforward systems with exponentially unstable jacobian linearization. *Systems and Control Letters*, 37(2):107–115, 1999.

[26] C. Melhuish Guo, L. and Q. Zhu. Towards neural adaptive hovering control of helicopters. *Proceedings of the 2002 IEEE International Conference on Control Applications*, 1:54–58, 2002.

[27] Jürgen Hartung, Florian Prester, and Robert Schachner. Cost efficient test system for embedded systems (cetes). *embedded world 2010 conference*, 2010.

[28] J. Shin D. Fujiwara K. Igarashi D. Fernando Hazawa, K. and K. Nonami. Autonomous flight control of hobby class small unmanned helicopter (modeling based on experimental identification and autonomous flight control experiments). *Nippon Kikai Gakkai Ronbunshu, C Hen/Transactions of the Japan Society of Mechanical Engineers, Part C*, 70(3):720–727, 2004.

[29] Sitio internet *Quanser*. "`http://www.quanser.com/`".

[30] Sitio internet *Xtratum*. "`http://www.xtratum.org/`".

[31] L. Marconi Isidori, A. and A. Serrani. Robust nonlinear motion control of a helicopter. *Automatica*, 48(3):413–426, 2003.

[32] M. Janković, R. Sepulchre, and P.V. Kokotović. Constructive Lyapunov stabilization of nonlinear cascade systems. *IEEE Transactions on Automatic Control*, 41(12):1723–1735, 1996.

[33] E. N. Johnson and S. K. Kannan. Adaptive trajectory control for autonomous helicopters. *Journal of Guidance, Control, and Dynamics*, 28:524–538, 2005.

[34] B. Kadmiry and D. Driankov. A fuzzy flight controller combining linguistic and model-based fuzzy control. *Fuzzy Sets and Systems*, 146:313–347, 2004.

[35] B. Kadmiry and D. Driankov. A fuzzy gain-scheduler for the attitude control of an unmanned helicopter. *IEEE Transactions on Fuzzy Systems*, 12:502–515, 2004.

[36] P. Bergsten Kadmiry, B. and D. Driankov. Autonomous helicopter control using fuzzy gain scheduling. *IEEE International Conference on Robotics and Automation*, 3:2980–2985, 2001.

[37] H. Katayama Kagawa, M. and A. Ichikawa. Attitude control of a helicopter model by feedback linearization. *SICE Annual Conference*, pages 1870–1875, 2005.

[38] G. Kaliora and A. Astolfi. Nonlinear control of feedforward systems with bounded signals. *IEEE Transactions on Automatic Control*, 49(11):1975–1990, 2004.

[39] G. Kaliora and A. Astolfi. On the stabilization of feedforward systems with bounded control. *Systems and Control Letters*, 54:263–270, 2005.

[40] C. Ham Kaloust, J. and Z. Qu. Nonlinear autopilot control design for a 2dof helicopter model. *IEE Proceedings: Control Theory and Applications*, 144:612–616, 1997.

[41] H.K Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[42] B. S. Kim and A. J. Calise. Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics*, 20:26–33, 1997.

[43] Y. Chang Kim, B. and M. H. Lee. System identification and 6 dof hovering controller design of unmanned model helicopter. *JSME International Journal, Series C: Mechanical Systems,Machine Elements and Manufacturing*, 49(4):1048–1057, 2006.

[44] Y. Chang J. Keh H. Ha Kim, B. and M. Lee. Design of 6 dof attitude controller of hovering model helicopter. *IECON 2004, 30th Annual Conference of IEEE Industrial Electronics Society*, 1(1):104–110, 2004.

[45] Ralya T. Pollak B. Obenza R. Klein, M. H. and M. G. Harbour. *A Practitioner's Handbook for RealTime Analysis*. Kluwer Academic Publishers, Boston, Massachusetts, 1993.

[46] A. M. Annaswamy Krupadanam, A. S. and R. S. Mangoubi. Multivariable adaptive control design with applications to autonomous helicopters. *Journal of Guidance, Control, and Dynamics*, 25:843–851, 2002.

[47] A. M. Annaswamy Krupadanam, A. S. and R. S. Mangoubi. A multivariable adaptive controller for autonomous helicopters. *American Control Conference*, 3:2052–2057, 2002.

[48] A. J. Calise M. Idan Kutay, A. T. and N. Hovakimyan. Experimental results on adaptive output feedback control using a laboratory model helicopter. *IEEE Transactions on Control Systems Technology*, 13:196–202, 2005.

[49] Vadim l. Utkin. Sliding mode control design principles and applications to electric drives. *IEEE Transactions on Industrial Electronics*, 40:23–36, 1993.

[50] A. Calise Leitner, J. and J. V. R. Prasad. Analysis of adaptive neural networks for helicopter flight control. *Journal of Guidance, Control, and Dynamics*, 20:972–979, 1997.

[51] A. Calise Leitner, J. and J. V. R. Prasad. Full authority helicopter adaptive neuro-controller. *Proceedings of the 1998 IEEE Aerospace Conference. Part 1 (of 5)*, 2:117–126, 1998.

[52] R. Bradley Liceaga-Castro, E. and R. Castro-Linares. Helicopter control design using feedback linearization techniques. *Proceedings of the 28th IEEE Conference on Decision and Control. Part 1 (of 3)*, pages 533–534, 1989.

[53] Z. Lin and A. Saberi. Semi-global exponential stabilization of linear systems subject to input saturation via linear feedbacks. *Systems and Control Letters*, 21:225–239, 1993.

[54] R. Lozano, editor. *Unmanned Aerial Vehicles, Embedded Control*. ISTE-WILEY, mayo 2010.

[55] M. López-Martínez, M.G. Ortega, and F.R. Rubio. Control robusto de un sistema de dos rotores en cuadratura.

[56] M. López-Martínez and F.R. Rubio. Control longitudinal de un helicóptero de laboratorio vía backstepping aproximado constructivo. In *XXV Jornadas de Automática*, Ciudad Real, España, 2004.

[57] R. Mahony and T. Hamel. Robust trajectory tracking for a scale model autonomous helicopter. *International Journal of Robust and Nonlinear Control*, 14:1035–1059, 2004.

[58] N. Marchand and A. Hably. Global stabilization of multiple integrators with bounded controls. *Automatica*, 41(12):2147–215, 2005.

[59] A. Isidori Marconi, L. and A. Serrani. Autonomous vertical landing on an oscillating platform: An internal model based approach. *Automatica*, 38(1):21–32, 2002.

[60] L. Marconi and A. Isidori. Robust global stabilization of a class of uncertain feedforward nonlinear systems. *Systems and Control Letters*, 41(4):281–290, 2000.

[61] L. Marconi and R. Naldi. Robust full degree of freedom tracking control of a helicopter. *Automatica*, 43(11):1909–1920, 2007.

[62] Aldea Rivas Mario. Planificación de tareas en sistemas sistemas operativos de tiempo real estricto para aplicaciones empotradas. Universidad de Cantabria, 2002.

[63] D.Q. Mayne and W. Langson. Robustifying model predictive control of constrained linear systems. *Electronics Letters*, 23:1422–1423, 2001.

[64] F. Mazenc and L. Praly. Adding integrations, satured controls, and stabilization for feedforward systems. *IEEE Transactions on Automatic Control*, 41, 1996.

[65] R. E. Mahony Mazenc, F. and R. Lozano. Forwarding control of scale model autonomous helicopter: A lyapunov control design. *42nd IEEE Conference on Decision and Control*, 4:3960–3965, 2003.

[66] Gonzalez Michael. Posix de tiempo real. Universidad de Cantabria, 2001.

[67] W. Michiels and D. Roose. Global stabilization of multiple integrators with time-delay and input constraints. In 3^{rd} *IFAC Workshop Time Delay Systems*, Santa Fe, États-Unis, décembre 2001.

[68] James F. Montgomery and George A. Bekey. Learning helicopter control through "teaching by showing". *Proceedings of the IEEE Conference on Decision and Control*, 4:3647–3652, 1998.

[69] Michiel van Nieuwstadt Morris, John C. and Pascale Bendotti. Identification and control of a model helicopter in hover.). *Proceedings of the 1994 American Control Conference. Part 1 (of 3)*, 2(5):1238–1242, 1994a.

[70] M.Prasanna and K.R. Chandran. Automatic test case generation for uml objects diagrams using genetic algorithm. *ICSRS Publication*, 1(1):19–32, 2009.

[71] H. Nakanishi and K. Inoue. Development of autonomous flight control systems for unmanned helicopter by use of neural networks. *International Joint Conference on Neural Networks (IJCNN 02)*, 3:2626–2631, 2002.

[72] H. Nakanishi and K. Inoue. Development of autonomous flight control systems for unmanned helicopter by use of neural networks. *International Joint Conference on Neural Networks (IJCNN 03)*, 3:2400–2405, 2003.

[73] C. L. Karr Phillips, C. and G. Walker. Helicopter flight control with fuzzy logic and genetic algorithms. *Engineering Applications of Artificial Intelligence*, 9(2):175–184, 1996.

[74] A. J. Calise Y. Pei Prasad, J. V. R. and J. E. Corban. Adaptive nonlinear controller synthesis and flight test evaluation: On an unmanned helicopter. *Proceedings of the 1999 IEEE International Conference on Control Applications (CCA) and IEEE International Symposium on Computer Aided Control System Design (CACSD)*, 1:137–142, 1999.

[75] S. Puntunan and M. Parnichkun. Online self tuning precompensation for a pid heading control of a flying robot. *International Journal of Advanced Robotic Systems*, 3(4):323–330, 2006.

[76] G.V. Raffo, M.G. Ortega, and F.R. Rubio. Plataforma de pruebas para un vehículo aéreo no tripulado utilizando labview.

[77] G. Sanahuja. *Commande et localisation embarquée dún drone en utilisant la vision.* PhD thesis, Université de Technologie de Compiègne.

[78] G. Sanahuja, P. Castillo, and A. Sanchez. Stabilization of n integrators in cascade with bounded input with experimental application to a VTOL laboratory system. *International Journal of Robust and Nonlinear Control*, 2009.

[79] A. Sanchez, P. Garcia, P. Castillo, and R. Lozano. Simple real-time stabilization of a VTOL aircraft with bounded signals. *AIAA Journal of Guidance, Control and Dynamics*, 31(4), 2008.

[80] H. M. Becerra Sanchez, E. N. and C. M. Velez. Combining fuzzy and pid control for an unmanned helicopter. *NAFIPS 2005 - 2005 Annual Meeting of the North American Fuzzy Information Processing Society*, 2005:235–240, 2005.

[81] H. M. Becerra Sanchez, E. N. and C. M. Velez. Combining fuzzy, pid and regulation control for an autonomous mini-helicopter. *Information Sciences*, 177:1999–2022, 2007.

[82] P.O.M. Scokaert and D.Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43:1136–1142, 1998.

[83] T. J. Koo F. Hoffmann Shim, H. and S. Sastry. Comprehensive study of control design for an autonomous helicopter. *Proceedings of the IEEE Conference on Decision and Control*, pages 3653–3658, 1998.

[84] D. Fujiwama K. Hazawa Shin, J. and K. Nonami. Model based optimal attitude and positioning control of smallscale unmanned helicopter. *Nippon Kikai Gakkai Ronbunshu, C Hen/Transactions of the Japan Society of Mechanical Engineers, Part C*, 70(9):2631–2637, 2004.

[85] Howard Winston Isao Hirano Sugeno, M. and Satoru Kotsu. Intelligent control of an unmanned helicopter based on fuzzy logic. *Proceedings of the 1995 51st Annual Forum. Part 1 (of 3)*, 1:791–803, 1995.

[86] I. Hirano S. Nakamura Sugeno, M. and S. Kotsu. Development of an intelligent unmanned helicopter. *Proceedings of the 1995 IEEE International Conference on Fuzzy Systems. Part 1 (of 5)*, 5:33–34, 1995.

[87] H.J. Sussmann, E.D. Sontag, and Y. Yang. A general result on the stabilization of linear systems using bounded controls. *IEEE Transactions on Automatic Control*, 39:2411–2425, 1994.

[88] A. Budiyono E. Joelianto Sutarto, H.Y. and G.T. Hiong. Switched linear control of a model helicopter. In *9th International Conference on COntrol, Automation, Rbotics and vision*, 2006.

[89] A.R. Teel. Global stabilization and restricted tracking for multiple integrators with bounded controls. *Systems and Control Letters*, 18:165–171, 1992.

[90] A.R. Teel. Semi-global stabilization of minimum phase nonlinear systems in special normal forms. *Systems and Control Letters*, 19:187–192, 1992.

[91] A.R. Teel. Semi-global stabilization of the ball and beam using output feedback. In *American Control Conference*, San Francisco, États-Unis, 1993.

[92] Robert L. Wade and Gregory W. Walker. Flight test results of the fuzzy logic adaptive controller helicopter (flach). *Engineering Applications of Artificial Intelligence*, 2738:200–208, 1996.

[93] E. A. Wan and A. A. Bogdanov. Model predictive neural control with applications to a 6 dof helicopter model. *American Control Conference*, 1:488–493, 2001.

[94] Hongcheng Wang, Zhigang Ding, and Yuwei Zhong. Static analysis test platform construction for embedded systems. *IEEE*, 2008.

[95] S. Boonto Witt, J. and H. Werner. Approximate model predictive control of a 3 dof helicopter. *46th IEEE Conference on Decision and Control 2007*, pages 4501–4506, 2008.

[96] C. D Yang and W. H Liu. Nonlinear h infinite decoupling hover control of helicopter with parameter uncertainties. *American Control Conference*, 4:3454–3459, 2003.

[97] W. H Liu Yang, C. D and C. C Kung. Nonlinear h infinite decoupling control for hovering helicopter. *American Control Conference*, 6:4353–4358, 2002.

[98] A.S.I Zinober. *Deterministic control of uncertain systems.* London: Peter Peregrinus Press, 1990.

5. Anexos

5.1. Detalle De Las plataformas QuadRotor

5.1.1. HOVER

Figura 51: Plataforma Experimental Hover.

La plataforma esta compuesta por:

1. 4 amplificadores de potencia
2. Tarjeta de adquisición de datos
3. Plataforma cuatrirotor

Una descripción detallada de los componentes de la plataforma cuatrirotor seria la siguiente:

1. Protección de las hélices	11. Base
2. Motor	12. Conector del motor Front
3. Presilla (contra-peso)	13. Conector del motor Right
4. Encoder Roll	14. Conector del motor Back
5. Encoder Pitch	15. Conector del encoder Yaw
6. Yugo	16. Conector del encoder Roll
7. Circuito del Motor/Encoder	17. Conector del encoder Pitch
8. Cuerpo	18. Encoder Yaw
9. Cuerpo	19. Conector del motor Left
10. Hélice	

Motors: Cuatro motores DC 12V, los motores Front y Back actúan en el eje Pitch, y los motores Right y Left actúan en el eje Row, para actuar en el eje Yaw usamos la combinación de los cuatro motores.
Encoders: Los encoders tienen resolución de 8192 divisiones por vuelta, o sea, 0.0439 grados por división.

Se le ha añadido una *IMU*. Este dispositivo electrónico permite medir y realizar una estimación de la velocidad, la rotación y las fuerzas gravitacionales de un objeto sobre el que este instalado, a través de una serie de girómetros, acelerómetros y magnetómetros.

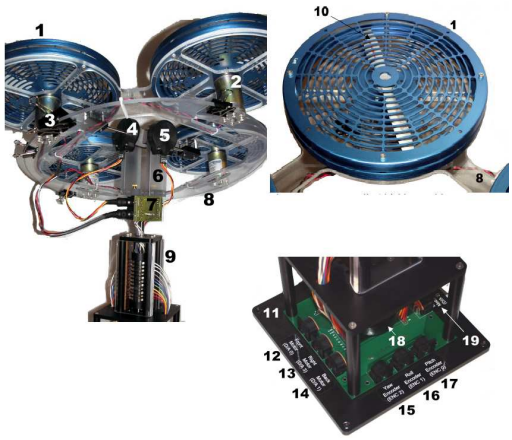


Figura 52: Plataforma Experimental Hover señalizada.

5.1.2. X-UFO

La plataforma *X-Ufo* esta compuesta por:

La unidad central de control se encarga de comunicar todas las partes del X-ufo, mientras que en un ordenador se realiza el control tanto de estabilidad como de orientación. La información del control externo se al envía al *X-Ufo* mediante una emisora que recibe la información a través de la unidad central para, realizar su propio control y enviar en forma de señal de voltaje la acción de control necesaria para alcanzar las referencias deseadas.

En la parte de sensorización cabe destacar que el helicóptero, para este proyecto, se le ha añadido una *IMU*. Este dispositivo electrónico permite medir y realizar una estimación de la velocidad, la rotación y las fuerzas gravitacionales de un objeto sobre el que este instalado, a través de una serie de girómetros, acelerómetros y magnetómetros.

Los *drivers* de control de cada motor son necesarios para acoplar la señal que llega a los cuatro motores desde la unidad central de control. Estos *drivers* están configurados específicamente para estos motores, por lo que se recomienda utilizar el mismo tipo de motores. Aquí se muestra un esquema de la plataforma X-UFO, 53.

La estructura del helicóptero está compuesta por una combinación de fibra de carbono, madera balsa y magnesio. Estos materiales se han elegido debido a su ligereza y resistencia. Además, el helicóptero dispone de una protección externa, para evitar posibles golpes sobre las hélices de los motores.

Unidad de Medida Inercial MicroStrain 3DM-GX2

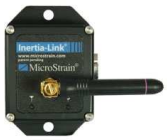
La Unidad de Medida Inercial 3DM-GX2 es un giroscopio de alto rendimiento mejorado con un sensor de orientación que utiliza la tecnología de sensores miniatura MEMS. Está compuesto por un acelerómetro triaxial, un autogiro triaxial, un magnetómetro triaxial, sensores de temperatura, y un procesador a bordo que ejecuta un algoritmo de fusión de sensores sofisticados.

Este sensor ofrece una amplia gama de datos de salida de las variables inerciales medidas calibradas (aceleración, velocidad angular y el campo magnético) y las estimaciones de la orientación computarizada (ángulo de *roll*, *pitch*, *yaw* y matriz de rotación).

- cuatro motores *brushless*

- cuatro *drivers* de control, uno para cada motor

- una unidad central de control interno



- una *IMU* (*Inertial Measurement Unit*)

- la Estructura del helicóptero.

La interfaz *hardware* de comunicaciones de la *IMU* se encuentra en un módulo separable, y por tanto se pueden particularizar fácilmente. En la actualidad, los módulos de interfaz disponibles incluyen un transmisor-receptor inalámbrico, USB 2.0, RS232 y RS422. Una versión OEM está disponible sin la interfaz de comunicaciones, lo que permite que el sensor sea integrado directamente en un sistema *circuitboard*, proporcionando una solución de detección muy compacta.

La arquitectura del sistema se ha diseñado cuidadosamente para eliminar sustancialmente las fuentes comunes de error, como histéresis inducidas por los cambios de temperatura y sensibilidad a las variaciones del voltaje del suministro. El uso de seis convertidores A/D Delta-Sigma independientes (una para cada sensor) garantiza que todos los sensores estén incluidos en la muestra de forma simultánea, y que los resultados de integración logrados sean mejores posibles hasta la fecha. Además se puede mejorar el índice de rendimiento de muestreo interno, mediante el uso de un tipo reducido de datos de salida. La unidad inercial 3DM-GX2 incorpora de forma opcional un magnetómetro triaxial integral; el cual se puede localizar de forma remota para reducir las interferencias asociadas al hierro.

El Software Development Kit (SDK) de la *IMU* proporciona aplicaciones para la creación de protocolos robustos y muestras de código fuente para el funcionamiento del *Inercia-Link* y los sensores de orientación 3DM-GX2. El

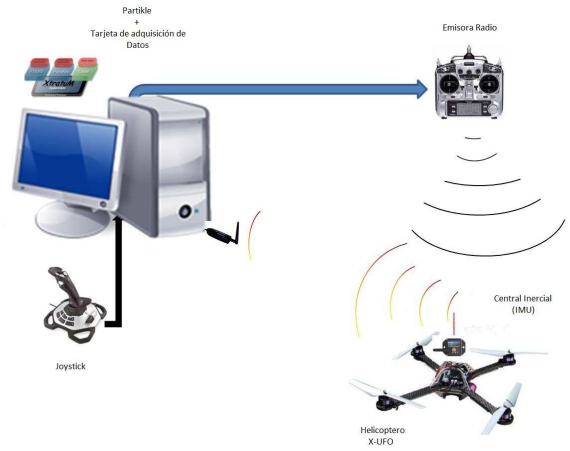


Figura 53: Plataforma Experimental Cuatrirotor

SDK incluye un completo manual de protocolo de comunicaciones de datos, un usuario manual detallado, aplicaciones para ANSI C y API. Loscomandos del protocolo que se proporcionan son completamente compatibles con el código fuente de *Microsoft Visual Studio C++ Express* 2003/2008, *LabVIEW* 7.1.1, *Microsoft Visual Studio* 2005 y *Visual Basic* 6.0.

A continuación se muestra de forma resumida las principales características de la Unidad de Medida Inercial *MicroStrain 3DM-GX2*.

- Pequeña, ligera, de baja potencia. Diseño ideal para las aplicaciones sensibles, incluyendo el tamaño de los dispositivos portátiles.
- Compensación total de temperatura sobre todo el radio de acción.
- Calibración de la alineación del sensor, de la sensibilidad del G-giroscopio, y del factor de escala de giro no-lineal.
- Muestreo simultáneo para mejorar el rendimiento de integración en el tiempo.
- Disponible con interfaces USB inalámbricas. Comunicación de tasa de datos de usuario (1 a 256Hz) y el sensor de ancho de banda (1 a 100 Hz) ajustables.
- Las Las salidas incluyen los ángulos de Euler, la matriz de rotación, y las derivadas de los ángulos y la velocidad, la aceleración y los vectores de velocidad angulares.

Sensores de distancia

SHARP GP2D12

Sensor que mide distancias con PSD (Detector Sensible a la Posición), LED infrarrojo y circuito de procesado de señal , valor analógico entre 0- 3V dependiendo de la distancia.

Estos dispositivos emplean el método de triangulación utilizando un pequeño Sensor Detector de Posición (PSD) lineal para determinar la distancia o la presencia de los objetos dentro de su campo de visión. Básicamente su modo de funcionamiento consiste en la emisión de un pulso de luz infrarroja, que se transmite a través de

Figura 54: Infrarrojo GP2D12

su campo de visón que se refleja contra un objeto o que por el contrario no lo hace. Si no encuentra ningún obstáculo, el haz de luz no refleja y en la lectura que se hace indica que no hay ningún obstáculo. En el caso de encontrar un obstáculo el haz de luz infrarroja se reflecta y crea un triangulo formado por el emisor, el punto de reflexión (obstáculo) y el detector. La información de la distancia se extrae midiendo el ángulo recibido. Si el ángulo es grande, entonces el objeto está cerca, por que el triangulo es ancho. Si el ángulo es pequeño, entonces el objeto está lejos, por que el triangulo formado es estrecho. Por lo tanto, si el ángulo es pequeño, quiere decir que el objeto está lejos, porque el triangulo es largo y delgado.

El Sharp GP2D12 es un sensor medidor de distancias por infrarrojos que indica mediante una salida analógica la distancia medida. La tensión de salida varia de forma no lineal cuando se detecta un objeto en una distancia entre 10 y 80 cm. La salida esta disponible de forma continua y su valor es actualizado cada 32 ms. Normalmente se conecta esta salida a la entrada de un convertidor analógico digital el cual convierte la distancia en un numero que puede ser usado por el microprocesador. La salida también puede ser usada directamente en un circuito analógico. Hay que tener en cuenta que la salida no es lineal. El sensor utiliza solo una línea de salida para comunicarse con el procesador principal. El sensor se entrega con un conector de 3 pines. Tensión de funcionamiento 5V, Temperatura funcionamiento:-10 a 60ºC, Consumo Medio: 35 mA. Margen de medida 10cm a 80 cm.

SHARP GP2D120

Figura 55: Infrarrojo GP2D120

El Sharp GP2D120 es un sensor medidor de distancias por infrarrojos que indica mediante una salida analógica la distancia medida. Este sensor, es una versión modificada del GP2D12, por lo que eléctricamente es igual y lo único que varia es el rango de trabajo, gracias al empleo de un lente especial. La tensión de salida varia de forma no lineal cuando se detecta un objeto en una distancia entre 4 y 30 cm. La salida esta disponible de forma continua y su valor es actualizado cada 32 ms aproximadamente. Normalmente se conecta esta salida a la entrada de un convertidor analógico digital el cual convierte la distancia en un número que puede ser usado por el microprocesador. La salida también puede ser usada directamente en un circuito analógico. Hay que tener en cuenta que la salida no es lineal. El sensor utiliza solo una línea de salida para comunicarse

con el procesador principal. El sensor se entrega con un conector de 3 pines. Tensión de funcionamiento 5V, Temperatura funcionamiento:-10 a 60ºC, Consumo Medio: 35 mA. Margen de medida 4cm a 30 cm.

Sensor ultrasonidos EZ1

Figura 56: Ultrasonidos EZ1

Este pequeño sensor por ultrasonido ofrece capacidades de detección de presencia y medición de distancia en rango corto y largo y un consumo muy bajo. MaxSonar- EZ1 detecta objetos situados entre 6-45 metros de distancia, proporcionando los datos obtenidos del cálculo de la distancia con una resolución de 1 pulgada (2,54 cm). Entre los formatos de salida se incluyen la salida de ancho de pulso, salida de tensión analógica y salida digital serie.

- **Características destacadas**
- Ganancia variable continua para el control del haz y supresión de lóbulos laterales (SLS).
- La detección de objetos incluye los objetos de distancia cero.
- Alimentación única de 5 voltios con un consumo de corriente de 2mA.
- Las lecturas se pueden realizar cada 50mS, (tasa de 20 Hz).
- Su funcionamiento le permite medir y generar la información de salida de forma continua.
- Se pueden generar las lecturas de mediciones mediante comandos.
- Todas las interfaces están activas simultáneamente.
- Serie, de 0 a 5 voltios.
- 9600 Baudios, 81N.
- Analógico (10m V/pulgada).
- Ancho de pulso (147uS/pulgadas).
- Aprende los patrones de trabajo al ordenar los inicios de las mediciones.
- Diseñado para entornos protegidos en el interior.
- El sensor opera a 42KHz.
- Salida a 10V PP de onda cuadrada del sensor.

- **Ventajas**
- Sensor de distancia con precio muy competitivo.

- Su tamaño es el más pequeño dentro de su categoría.
- Las zonas muertas del sensor desaparecen virtualmente.
- No hay puntos ciegos centrales.
- Haces de gran calidad.
- Orificios de montaje incluidos en la placa de circuito impreso.
- Sensor de distancia de baja potencia, ideal para los sistemas formados por varios sensores alimentados por baterías.
- Activación externa o interna.
- El sensor genera las lecturas directamente, liberando al procesador de la carga de trabajo.
- Ciclos de medición rápidos.
- El usuario puede elegir cualquiera de las 3 salidas del sensor.

Laser Hokuyo

Figura 57: Laser Hokuyo

El sensor láser Hokuyo URG-04LX es utilizado en múltiples aplicaciones debido a su robustez, tamaño y su reducido peso. Además, presenta una simple conexión USB lo que lo hace ideal para todo tipo de aplicaciones de control y robótica. Las principales ventajas de este sensor láser se muestran a continuación

- Área de reconocimiento de ± 120 grados.
- Es capaz de detectar objetos hasta una distancia de 5.6 metros.
- Presenta una frecuencia de mapeo de 10 Hz.
- Es muy compacta y de reducidas dimensiones en comparación con otros sensores láser (50 x 50 x 70 mm)
- Reducido peso (160 gramos). Ideal en aplicaciones con UAVs.
- Funciona con una tensión de alimentación de 5V DC.
- Se conecta al PC con el que se trabaja mediante un puerto USB.

5.2. Esquema de conexiones Hardware

5.2.1. Conexiones Hover

La plataforma HOVER, vista desde el punto de vista de conexiones, posee tres configuraciones diferentes. Aunque primero se van a mostrar las distintas partes que componen las conexiones del Hover.

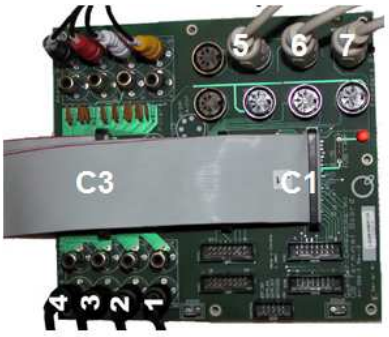


Figura 58: Tarjeta Quarc

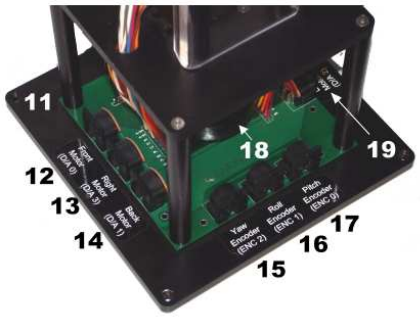


Figura 59: Conexiones base hover

Figura 60: Amplificador de potencia(4 uds)

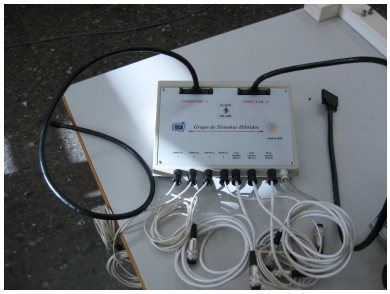


Figura 61: Caja Blanca(Conexión PCI)

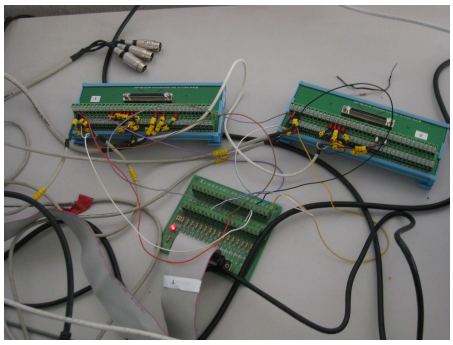


Figura 62: Caja desnuda(Conexión ISA)

1. Quarc

La configuración Quarc es aquella que se obtuvo con la compra de la plataforma. Posee cuatro partes:

- El hover, que es un helicóptero cuatrirotor montado sobre un pilar vertical.
- Los amplificadores de potencia(cuatro cajas negras, una por cada motor)
- La tarjeta quarc(que distribuye la información tanto de entrada como de salida entre las distintas partes)
- El ordenador o controlador externo(en el que tiene instalado la aplicación de matlab para poder hacer funcionar el HOVER)

Al hover tiene que llegarle la potencia para los motores y tiene que enviar la señal de los encoders. La potencia se obtiene recibiéndola de los amplificadores(salida "to load"(amplificador)-conexión en base de HOVER). Los encoders van conectados a la tarjeta QUARC(conexión en base de hover-tarjeta quarc(Ver figura 58) conectores grises).

2 Rt-Linux

La información de los encoders pasa de la tarjeta por los buses de datos al ordenador. En este los datos se procesan y se envían de nuevo a la tarjeta quarc con los valores de las acciones de control (potencia para los motores) necesarias esta información se distribuye a los cuatros amplificadores(tarjeta quarc(ver figura 1) conectores negros-entrada "From D/A"(amplificadores)). La configuración Rt-linux funciona mediante un programa de C++ que se ha desarrollado especialmente para esta plataforma. En esta configuración posee 4 partes:

- El hover, que es un helicóptero cuatrirotor montado sobre un pilar vertical. En esta configuración se le ha añadido una central inercial para poder medir las velocidades angulares. La cual se conecta mediante USB al control externo.
- Los amplificadores de potencia(cuatro cajas negras, una por cada motor)
- La caja "blanca"(que distribuye la información tanto de entrada como de salida entre las distintas partes)
- El ordenador o controlador externo(en el que tiene instalado la aplicación de matlab para poder hacer funcionar el HOVER)

Ahora el Hover posee dos "salidas" y una "entrada". Las "salidas" son los Encoders, que se conectan a la caja blanca y la IMU que se conecta mediante USB a cualquier entrada del Ordenador. La información de las "salidas" se procesa en el ordenador y enviá las acciones de control a la caja "blanca"(ver figura 61) y esta las distribuye por los cuatro amplificadores.

3 Xtratum+Partikle

La última configuración es muy parecida a la conexión de RT-LINUX la única diferencia es la forma de introducir los datos al ordenador. Se utiliza una tarjeta con entrada ISA y por tanto ni la tarjeta QUARC o la caja "blanca" se puede utilizar para realizar las conexiones. Para eso se modifica esa parte para para poner la caja "desnuda".

5.2.2. Conexiones X-ufo

Para el uso experimental del X-ufo se desarrollo un protocolo estricto de conexiones que se debía seguir cada vez que se iniciaba una prueba de la UAV. Para plasmar que conexiones se deberían realizar, para esta maquina, en este apartado se muestra el protocolo tanto de encendido como de apagado.

Protocolo Encendido X-UFO

1. El ordenador se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.
   c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).
   d) La caja de amplificación (caja negra) conectada a la emisora.
   e) La emisora conectada a la alimentación.
   f) EL receptor de la IMU conectado a una entrada USB del ordenador.
   g) El joystick conectado a una entrada USB del ordenador.
   h) El X-UFO conectado con la batería o fuente de alimentación.
3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): **/etc/init.d/mcapi start**
4. Se comprueba la activación de la misma con el comando : **lsmod — grep mcapi**
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: **cd /opt/XUFO-PCI-guillaume**
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.
8. Se inicia el programa con el comando: **./bin/hover -n**
9. Si no da ninguna señal de error, pulsar la tecla "**s**" para iniciar la conexión ordenador - X-UFO.
10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla "**y**" para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.

Protocolo Apagado X-UFO

1. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
2. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
3. Terminar el programa de ordenador. Pulsar tecla "**Esc**".
4. Desconectar la batería del X-UFO.
5. El X-UFO esta apagado. *El resto de pasos solo son necesarios en caso de finalización de jornada de trabajo con el X-UFO.*
6. Separar baterías del X-ufo y situarlas en su caja.
7. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.

8. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
9. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
10. Recoger todo y guardarlo donde corresponda

5.3. Aplicaciones de Control/software de control

Los diferentes programas existentes son:

HOVER-ALFONSO

Aplicación
Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el boton "r")e iniciar el control(con el boton "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea(main) 145: /opt/HOVER-Alfonso/LinuxParche.txt 5 columnas: roll medido de IMU, referencia, acción de control, tiempo, roll medido de encoders.(en estos momentos esta linea esta comentada) Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start.
7. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi.
8. Se situa en la carpeta del programa que se vaya a ejecutar con el comando: cd /opt/HOVER-Alfonso.
9. Se ejecuta el comando: ./bin/hover -n.
10. Se resetean los valores de medida con el botón "r"
11. Se inicia la secuencia de control con el botón "s"
12. Para detener el control se hará con la "barra espaciador"
13. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 235:

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 438:

```
if (TiempoCambio>20){
```

Este if indica el tiempo que se mantendrá buscando la referencia (20 segundos).
En la linea 440:

```
if(k2==6)
```

Este if indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(374-394):

```
double e[3]={0.0, 0.0, 0.0};
double reff[3]={0.0, 0.0, 0.0};
double ref_r=0.0;
double uk[2]={0.0, 0.0};
double uvector[4]={0.0,0.0,0.0,0.0};
float Tmuestreo =  0.005;//Revisar 0.010;
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
int k2 = 0;
double K1R = 50.0;
double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el MAXIMO es 4 y tamaño
                                          //del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
```

El código de control es:
//////////CODIGO DE CONTROL

```
eexterno[0]=ref-(imu_roll*(180/HOVER_PI));
y_A=DenZPI[1]*uvectorexterno[tamano-1];
y_B=NunZPI[0]*eexterno[0]+NunZPI[1]*eexterno[1];
uvectorexterno[tamano]=-y_A+y_B;
refinterna = uvectorexterno[tamano-ret];

        //Actualizacion de las variables
int i;
for (i=0 ; i < tamano ; i++)
{uvectorexterno[i] = uvectorexterno[i+1];}
eexterno[1] = eexterno[0];

                //Control interno para el SP
ref_r = refinterna*(HOVER_PI/180);
e[0] = ref_r - encoder_roll;//////////////ENCODERS!!!!!
uvector[0] = -(Den[1]/Den[0])*uvector[1] + Nun[0]/Den[0]*e[0]
+ Nun[1]/Den[0]*e[1];
uvector[1] = uvector[0];
e[1] = e[0];
uk[0] = uvector[0];
//////////ESCRITURA
```

HOVER-ALFONSO-PCI

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el boton"r")e iniciar el control(con el boton "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea(main) 142: /opt/HOVER-Alfonso/LinuxParche.txt 5 columnas: roll medido de IMU, referencia, acción de control, tiempo, roll medido de encoders. Una diferencia entre este programa y el anterior es que este no tiene comentadas estas lineas. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start.
7. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi.
8. Se situa en la carpeta del programa que se vaya a ejecutar con el comando: cd /opt/HOVER-Alfonso-PCI.
9. Se ejecuta el comando: ./bin/hover -n.
10. Se resetean los valores de medida con el botón "r"
11. Se inicia la secuencia de control con el botón "s"
12. Para detener el control se hará con la "barra espaciador"
13. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 235:

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 438:

```
if (TiempoCambio>20){
```

Este if indica el tiempo que se mantendrá buscando la referencia (20 segundos).

En la linea 440:

```
if(k2==6) {
```

Este if indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(368-388):

```
double e[3]={0.0, 0.0, 0.0};
double reff[3]={0.0, 0.0, 0.0};
double ref_r=0.0;
double uk[2]={0.0, 0.0};
double uvector[4]={0.0,0.0,0.0,0.0};
float Tmuestreo =  0.005;//Revisar 0.010;
```

```
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
int k2 = 0;
double K1R = 50.0;
double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el MAXIMO es 4
//y tamaño del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
double imu_roll;
```

El código de control es:

```
                    //////////CODIGO DE CONTROL

eexterno[0]=ref-(imu_roll*(180/HOVER_PI));
y_A=DenZPI[1]*uvectorexterno[tamano-1];
y_B=NunZPI[0]*eexterno[0]+NunZPI[1]*eexterno[1];
uvectorexterno[tamano]=-y_A+y_B;
refinterna = uvectorexterno[tamano-ret];

//Actualización de las variables

int i;
for (i=0 ; i < tamano ; i++){uvectorexterno[i] =
uvectorexterno[i+1];}
eexterno[1] = eexterno[0];
                    //Control interno para el SP

ref_r = refinterna*(HOVER_PI/180);
e[0] = ref_r - encoder_roll;//////////////ENCODERS!!!!!
uvector[0] = -(Den[1]/Den[0])*uvector[1] + Nun[0]/Den[0]*e[0]
+ Nun[1]/Den[0]*e[1];
uvector[1] = uvector[0];
e[1] = e[0];
uk[0] = uvector[0];
//////////ESCRITURA
```

HOVER-Andreu-PCI

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el boton"r")e iniciar el control(con el boton "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea(main) 142: /opt/HOVER-Alfonso/LinuxParche.txt 5 columnas: roll medido de IMU, referencia, acción de control, tiempo, roll medido de encoders. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start.
7. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi.
8. Se situa en la carpeta del programa que se vaya a ejecutar con el comando: cd /opt/HOVER-Andreu-PCI.
9. Se ejecuta el comando: ./bin/hover -n.
10. Se resetean los valores de medida con el botón "r"
11. Se inicia la secuencia de control con el botón "s"
12. Para detener el control se hará con la "barra espaciador"
13. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 229(main):

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 433(hover):

```
if (TiempoCambio>20){
```

Este "if" indica el tiempo que se mantendrá buscando la referencia (20 segundos).

En la linea 435:

```
if(k2==6) {
```

Este "if" indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(369-389):

```
double e[3]={0.0, 0.0, 0.0};
double reff[3]={0.0, 0.0, 0.0};
double ref_r=0.0;
double uk[2]={0.0, 0.0};
double uvector[4]={0.0,0.0,0.0,0.0};
float Tmuestreo =  0.005;//Revisar 0.010;
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
int k2 = 0;
double K1R = 50.0;
double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el  MAXIMO es 4
```

```
//y tamaño del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
```

El código de control es:

```
                    //////////CODIGO DE CONTROL
eexterno[0]=ref-(imu_roll*(180/HOVER_PI));
y_A=DenZPI[1]*uvectorexterno[tamano-1];
y_B=NunZPI[0]*eexterno[0]+NunZPI[1]*eexterno[1];
uvectorexterno[tamano]=-y_A+y_B;
refinterna = uvectorexterno[tamano-ret];
//Actualizacion de las variables

int i;
for (i=0 ; i < tamano ; i++)
{uvectorexterno[i] = uvectorexterno[i+1];}
eexterno[1] = eexterno[0];
                    //Control interno para el SP
ref_r = refinterna*(HOVER_PI/180);
e[0] = ref_r - encoder_roll;//////////////ENCODERS!!!!!
uvector[0] = -(Den[1]/Den[0])*uvector[1] + Nun[0]/Den[0]*e[0]
 + Nun[1]/Den[0]*e[1];
uvector[1] = uvector[0];
e[1] = e[0];
uk[0] = uvector[0];
```

HOVER(Francisco)

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el botón"r")e iniciar el control(con el botón "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea(main) 148 :" /opt/HOVER/Hover.txt" 5 columnas: roll medido de IMU, referencia, acción de control, tiempo, roll medido de encoders. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start.
7. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi.
8. Se situa en la carpeta del programa que se vaya a ejecutar con el comando: cd /opt/HOVER(Francisco)
9. Se ejecuta el comando: ./bin/hover -n.

10. Se resetean los valores de medida con el boton “r”
11. Se inicia la secuencia de control con el boton “s”
12. Para detener el control se hará con la “barra espaciador”
13. Para detener el programa se hará con el boton “ESC”

Detalle de parámetros configurables En la linea 226/(main):

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 492(hover):

```
if (TiempoCambio>20){
```

Este if indica el tiempo que se mantendrá buscando la referencia (20 segundos).

En la linea 494(hover):

```
if(k2==6) {
```

Este if indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces)

Las constantes de la ley de control están en las lineas(390-416):

```
static float Tmuestreo = 0.005;

double N1PI = 1000;
double N2PI = -985.7;
double D1PI = 1;
double D2PI = -0.7144;
NunZPI[1] = 20;
NunZPI[2] = -19.2271;
DenZPI[1] = 1;
DenZPI[2] = -0.2271;
NunZ[1] = 0;
NunZ[2] = 0.0001807;
NunZ[3] = 0.0001763;
DenZ[1] = 1;
DenZ[2] = -1.929;
DenZ[3] = 0.9286;
FNun2[1] = 0.5125;
FNun2[2] = -1.0221;
FNun2[3] = 0.5096;
FDen2[1] = 1;
FDen2[2] = -1.9975;
FDen2[3] = 0.9975;
```

El código de control es:

```
                    //////////CODIGO DE CONTROL
eexterno[1]=ref*(pi/180)- imu_roll;
y_A=DenZPI[2]*uvectorexterno[tamano-1];
```

```
y_B=NunZPI[1]*eexterno[1]+NunZPI[2]*eexterno[0];
y_aux=-y_A+y_B;
uvectorexterno[tamano]=y_aux;
refinterna = uvectorexterno[tamano-ret];
u = 35.0*(refinterna - imu_roll_rate);// Hay que hacer todo en radianes.
// Realimentacion interna K = 35
int i;
for (i=0 ; i < tamano ; i++)
{uvectorexterno[i] = uvectorexterno[i+1];}
for (i=0 ; i < 2 ; i++){y_e[i] = y_e[i+1];}
for (i=0 ; i < 2 ; i++){y_p[i] = y_p[i+1];}
for (i=0 ; i < 2 ; i++){e_p[i] = e_p[i+1];}
for (i=0 ; i < 2 ; i++){e_aux[i] = e_aux[i+1];}
eexterno[0] = eexterno[1];
```

HOVER-Alfonso-ISA-Linux(PD)

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el boton"r")e iniciar el control(con el boton "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea(hover) 412:" Linux_Exp01.txt" 4 columnas: roll medido de IMU, referencia, acción de control, roll medido de encoders. Este programa funciona con la tarjeta ISA Puesta en marcha y paro

1. El ordenador(con tarjeta ISA) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con la serie de comandos siguiente(en dos terminales):
   a) consola 1. cd /opt/HOVER-Alfonso-ISA-Linux
   b) consola 2. cd /opt/HOVER-Alfonso-ISA-Linux/driver-t-6126/
   c) consola 2 insmod driver_6126.ko
   d) comprobación con: lsmod
   e) Si driver_6126 used by 0 entonces consola 2. dmesg
   f) Dependiendo del numero obtenido en punto anterior se introduce el siguiente comando en la consola 2. mknod /dev/d6126 c 253(este es el numero a modificar) 0
   g) Se ejecuta el comando en la consola 1. : ./bin/hover -n.
7. Se resetean los valores de medida con el botón "r"
8. Se inicia la secuencia de control con el botón "s"
9. Para detener el control se hará con la "barra espaciador"

10. Para detener el programa se hará con el botón “ESC”

Detalle de parámetros configurables En la linea 226(main):

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 447(hover):

```
if (TiempoCambio>10){
```

Este “if” indica el tiempo que se mantendrá buscando la referencia (20 segundos). En la linea 449(hover):

```
if(k2==4) {
```

Este “if” indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(402-406):

```
double N1 = 1000;
double N2 = -996.7;
double D1 = 1;
double D2 = -0.9349;
int j,i=0,ii=0;
```

El código de control es:

```
ref_r = ref*(pi/180); //Cambia la variable degrees > radians
reff = (-N2/N1)*reffant + ((D1 + D2)/(N1))*(ref_r*K1R);
e = reff -imu_roll;
u = -(D2/D1)*uant+N1/D1*e + N2/D1*eant;
uant = u;
reffant = reff;
eant = e;
```

HOVER-FIDEL-ISA-Linux

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el boton“r”)e iniciar el control(con el boton “s”). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en el archivo especificado en la linea (main) 174:“ Linux_Exp01.txt” 4 columnas: roll medido de IMU, referencia, acción de control, roll medido de encoders. Este programa funciona con la tarjeta ISA Puesta en marcha y paro

1. El ordenador(con tarjeta ISA) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con la serie de comandos siguiente(en dos terminales):
   a) consola 1. cd /opt/HOVER-FIDEL-ISA-Linux

b) consola 2. cd /opt/HOVER-FIDEL-ISA-Linux/driver-t-6126/

c) consola 2 insmod driver_6126.ko

d) Comprobación con: lsmod

e) Si driver_6126 used by 0 entonces consola 2. dmesg

f) Dependiendo del numero obtenido en punto anterior se introduce el siguiente comando en la consola 2. mknod /dev/d6126 c 253(este es el numero a modificar) 0

g) Se ejecuta el comando en la consola 1. : ./bin/hover -n.

7. Se resetean los valores de medida con el botón "r"
8. Se inicia la secuencia de control con el botón "s"
9. Para detener el control se hará con la "barra espaciador"
10. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 226(main):

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de ref indica el ángulo que tratará de alcanzar el hover. En la linea 390(hover):

```
if (TiempoCambio>20){
```

Este "if" indica el tiempo que se mantendrá buscando la referencia (20 segundos). En la linea 392(hover):

```
if(k2==7) {
```

Este if indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(326-346):

```
double e[3]={0.0, 0.0, 0.0};
double reff[3]={0.0, 0.0, 0.0};
double ref_r=0.0;
double uk[2]={0.0, 0.0};
double uvector[4]={0.0,0.0,0.0,0.0};
float Tmuestreo =  0.005;//Revisar 0.010;
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
int k2 = 0;
double K1R = 50.0;
double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el MAXIMO es 4
//y tamaño del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
```

El código de control es:

```
//Comienzo de control

        //////////CODIGO DE CONTROL

eexterno[0]=ref-(imu_roll*(180/HOVER_PI));
y_A=DenZPI[1]*uvectorexterno[tamano-1];
y_B=NunZPI[0]*eexterno[0]+NunZPI[1]*eexterno[1];
uvectorexterno[tamano]=-y_A+y_B;
refinterna = uvectorexterno[tamano-ret];



//Actualizacion de las variables

int i;
for (i=0 ; i < tamano ; i++)
{uvectorexterno[i] = uvectorexterno[i+1];}
eexterno[1] = eexterno[0];
        //Control interno para el SP
ref_r = refinterna*(HOVER_PI/180);
e[0] = ref_r - encoder_roll;//////////////ENCODERS!!!!!
uvector[0] = -(Den[1]/Den[0])*uvector[1] + Nun[0]/Den[0]*e[0]
 + Nun[1]/Den[0]*e[1];
uvector[1] = uvector[0];
e[1] = e[0];
uk[0] = uvector[0];

        //////////ESCRITURA
```

HOVER-FIDEL-ISA-Partickle

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el botón "r")e iniciar el control(con el botón "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en un archivo especificado en la linea(main) 132: " Partikle_Exp05.txt" 3 columnas: referencia, roll medido de IMU y acción de control. Este programa funciona con la tarjeta ISA Puesta en marcha y paro

1. El ordenador(con tarjeta ISA) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).
6. Se inicia la tarjeta de adquisición de datos con la serie de comandos siguiente(en dos terminales):
   a) consola 1. cd /opt/HOVER-FIDEL-ISA-Partickle/Linux

b) consola 2. cd /opt/HOVER-FIDEL-ISA-Partickle/partikleApp/

c) consola 1. make

d) consola 1. /usr/src/xmertl/xtratum/user_tools/scripts/xmcmd.sh -l

e) consola 2. make

f) consola 2. /usr/src/xmertl/xtratum/user_tools/xmloader/loader.xm partikleApp.prtk

g) verificación de módulos con: cat /proc/xtratum (debería aparecer una referencia a linux y otra a xtratum)

h) consola 2 insmod driver_6126.ko

i) comprobación con: lsmod

j) si driver_6126 used by 0 entonces consola 2. dmesg

k) dependiendo del numero obtenido en punto anterior se introduce el siguiente comando en la consola 2. mknod /dev/d6126 c 253(este es el numero a modificar) 0

l) Se ejecuta el comando en la consola 1. : ./bin/hover -n.

7. Se resetean los valores de medida con el botón "r" 8. Se inicia la secuencia de control con el botón "s" 9. Para detener el control se hará con la "barra espaciador" 10. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 129(main):

```
double u=0.0, ref=6.0;  //Inicialización de la referencia sobre el Roll
```

El valor de "ref" indica el ángulo que tratará de alcanzar el hover. En la linea 390(hover):

```
if (TiempoCambio>20){
```

Este "if" indica el tiempo que se mantendrá buscando la referencia (20 segundos). En la linea 392(hover):

```
if(k2==7) {
```

Este "if" indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(326-346):

```
double e[3]={0.0, 0.0, 0.0};
double reff[3]={0.0, 0.0, 0.0};
double ref_r=0.0;
double uk[2]={0.0, 0.0};
double uvector[4]={0.0,0.0,0.0,0.0};
float Tmuestreo =  0.005;//Revisar 0.010;
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
int k2 = 0;
double K1R = 50.0;
 double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el  MAXIMO es 4
```

```
//y tamaño del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
```

El código de control es:

```
//Comienzo de control

        //////////CODIGO DE CONTROL
eexterno[0]=ref-(imu_roll*(180/HOVER_PI));
y_A=DenZPI[1]*uvectorexterno[tamano-1];
y_B=NunZPI[0]*eexterno[0]+NunZPI[1]*eexterno[1];
uvectorexterno[tamano]=-y_A+y_B;
refinterna = uvectorexterno[tamano-ret];

//Actualizacion de las variables
int i;
for (i=0 ; i < tamano ; i++)
{uvectorexterno[i] = uvectorexterno[i+1];}
eexterno[1] = eexterno[0];

        //Control interno para el SP
ref_r = refinterna*(HOVER_PI/180);
e[0] = ref_r - encoder_roll;//////////////ENCODERS!!!!!
uvector[0] = -(Den[1]/Den[0])*uvector[1] + Nun[0]/Den[0]*e[0]
 + Nun[1]/Den[0]*e[1];
uvector[1] = uvector[0];
e[1] = e[0];
uk[0] = uvector[0];

        //////////ESCRITURA
```

HOVER-SM-PCI

Aplicación Con este programa se poseen dos opciones: resetear la IMU y los encoders (con el botón "r")e iniciar el control(con el botón "s"). Al iniciar el control se inicia una secuencia que enviara una referencia al roll que el HOVER tratara de seguir. Y grabara los valores en un archivo especificado en la linea(hover) 399: "/opt/HOVER-SM-PCI/Superficie.txt" 5 columnas: roll medido de IMU, encoder roll, referencia, acción de control y Sup_des1. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia en Linux.
2. La caja de conexiones(caja blanca) en modo HOVER QUANSER.
3. La caja de conexiones(caja blanca) conectada a los amplificadores y al HOVER(acción de control y señal de encoders, respectivamente).
4. Los amplificadores conectados.
5. La IMU conectada con el USB al controlador externo(Ordenador).

6. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start.
7. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi.
8. Se situa en la carpeta del programa que se vaya a ejecutar con el comando: cd /opt/HOVER-SM-PCI.
9. Se ejecuta el comando: ./bin/hover -n.
10. Se resetean los valores de medida con el botón "r"
11. Se inicia la secuencia de control con el botón "s"
12. Para detener el control se hará con la "barra espaciador"
13. Para detener el programa se hará con el botón "ESC"

Detalle de parámetros configurables En la linea 230(main):

```
quad_hover->ref=6;    //Inicialización de la referencia sobre el Roll
```

El valor de "ref" indica el ángulo que tratará de alcanzar el hover. En la linea 437(hover):

```
if (TiempoCambio>20){
```

Este "if" indica el tiempo que se mantendrá buscando la referencia (20 segundos). En la linea 439(hover):

```
if(k2==6) {
```

Este "if" indica la cantidad de veces que la referencia pasara de un valor a el mismo valor pero de signo contrario. (6 veces) Las constantes de la ley de control están en las lineas(374-392):

```
float Tmuestreo =  0.01;//Revisar 0.010;
double Nun[2] = {1000.0,-985.7};
double Den[2] = {1.0,-0.7144};
double P,I,D,Iant,TiempoCambio=0.0;
int k = 0;
double ksm = 0.5;
float Sup_des1=0;
float Sup_des2=0;
int k2 = 0;
double K1R = 50.0;
double DenZPI[3]={1.0,-1.0, 0.0};
double NunZPI[3]={0.05, -0.045, 0.0};
int tamano=199,ret=150;//Valor del retardo, el MAXIMO es 4
//y tamaño del vector U es 199!!
double uvectorexterno[200];
double eexterno[2]={0,0};
double y_A=0.0,y_B=0.0,refinterna=0.0;
```

El código de control es:

```
//Comienzo de control

//////////CODIGO DE CONTROL
e[0]=ref-(imu_roll*(180/HOVER_PI));
eexterno[0] = ref - encoder_roll;
```

```
y_A = ksm*eexterno[0];
y_B = (e[0]-e[1])/(0.01);
Sup_des1 = y_A+y_B;
if (Sup_des1>0.001) Sup_des2 = 1;
if (Sup_des1<0.001 && Sup_des1>-0.001) Sup_des2 = 0;
if (Sup_des1<-0.001) Sup_des2 = -1;
uk[0] = 2.5*Sup_des2;
//Actualizacion de las variables
e[1] = e[0];


//////////ESCRITURA
```

XUFO-PCI-PEDRO

Aplicación Con este programa se poseen 8 opciones: resetear la IMU (con el botón"r"), iniciar la comunicación con el helico(con el botón "s"), iniciar el control externo(con el botón "y"), apagar la comunicación con el helicóptero(con la "barra espaciadora"), apagar el control externo (con el botón"j"), modificar en 20 grados positivos la yaw deseada(con el botón "m"), modificar en 20 grados negativos la yaw deseada(con el botón "z"), aumentar el empuje en 0.02(con el botón "q"), disminuir el empuje en 0.02(con el botón "p"). Al iniciar el control se iniciará un control sobre el helico que hará seguir las referencias que se le manden(en este programa en ejecución solo se puede modificar la yaw). Y grabará los valores en dos archivos especificados en: linea(hover) 415:"//opt/XUFO-PCI-Pedro/XUFO_EXP14.txt" 5 columnas: yaw deseada, yaw medida, acción de control de yaw, empuje. Linea (main) 155:"/opt/XUFO-PCI-Pedro/XUFO-EXP0.txt" 5 columnas: tiempo, referencia, yaw medida, acción control, empuje. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.
   c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).
   d) La caja de amplificación (caja negra) conectada a la emisora.
   e) La emisora conectada a la alimentación.
   f) El receptor de la IMU conectado a una entrada USB del ordenador.
   g) El joystick conectado a una entrada USB del ordenador.
   h) El X-UFO conectado con la batería o fuente de alimentación.
3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start
4. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: cd /opt/XUFO-PCI-PEDRO
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.

8. Se inicia el programa con el comando: ./bin/hover -n
9. Si no da ninguna señal de error, pulsar la tecla “s” para iniciar la conexión ordenador - X-UFO.
10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla “y” para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.
13. Protocolo Apagado X-UFO
14. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
15. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
16. Terminar el programa de ordenador. Pulsar tecla “Esc”.
17. Desconectar la batería del X-UFO.
18. El X-UFO esta apagado.
19. Separar baterías del X-ufo y situarlas en su caja.
20. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.
21. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
22. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
23. Recoger todo y guardarlo donde corresponda

Detalle de parámetros configurables En este caso el programa se modifica en ejecución. Las constantes de la ley de control están en las lineas(32-45) y (383-404):

```
double  hover::k1=0.00001;   // ganancia D del control de velocidad yaw
double  hover::k2= 0.003; // ganancia P del control de yaw
double  hover::kpitch1=0.0001;   // ganancia D del control de velocidad pitch
double  hover::kpitch2= 0.003; // ganancia P del control de pitch
double  hover::kroll1=0.0001;   // ganancia D del control de velocidad roll
double  hover::kroll2= 0.003; // ganancia P del control de roll
double hover::yaw_d = 0.0;
double hover::u_s=0.0;
double yaw_j=2.5,pitch_j=0.0,roll_j=2.5;
        // variables del control de yaw
double satura1=0.01, satura2=0.2;// la 1ª k1 2ª k2(inicialmente estaba en 0.3)
double sigma1, sigma2;
        // variables del control de pitch
double satpitch1=0.01, satpitch2=0.2;   //Antes estaban en 0.3
double sigmapitch1, sigmapitch2;
        // variables del control de roll
double satroll1=0.01, satroll2=0.2;
double sigmaroll1, sigmaroll2;
```

El código de control es:

```
        //Comienzo de control

        ///////////CODIGO DE CONTROL
ControlFunctionJoystickUfo();
/*Lee central inercial (Euler Angles y Euler rate angles)
y cambiar el valor a grados */
yaw =( -1*(180/HOVER_PI)*imu_tdmgx2->GetYaw()) ;
pitch = (180/HOVER_PI)*imu_tdmgx2->GetPitch();
roll = (180/HOVER_PI)*imu_tdmgx2->GetRoll();
yaw_rate = (180/HOVER_PI)*imu_tdmgx2->GetYawRate();
pitch_rate = (180/HOVER_PI)*imu_tdmgx2->GetPitchRate();
roll_rate = (180/HOVER_PI)*imu_tdmgx2->GetRollRate();

        /*Se ejecuta accion de control*/
if ( bRunControlyaw == true)
{sigma2= saturacion(k1 * yaw_rate,satura1);
sigma1 = saturacion(k2 * (yaw-yaw_d),satura2);
yaw_c = -sigma2 - sigma1;
sigmapitch2= saturacion(kpitch1 * pitch_rate, satpitch1);
sigmapitch1 = saturacion(kpitch2 * (pitch-pitch_d), satpitch2);
pitch_c = (sigmapitch2 + sigmapitch1);
sigmaroll2= saturacion(kroll1 * roll_rate, satroll1);
sigmaroll1 = saturacion(kroll2 * (roll-roll_d), satroll2);
roll_c = (sigmaroll2 + sigmaroll1);
}
u_sal= u_j-u_s;
pitch_s = pitch_j+pitch_c;
roll_s  = roll_j+roll_c;
y_sal= yaw_s= yaw_j+yaw_c;
```

XUFO-PCI-PEDRO(Ley de Control)

Aplicación Con este programa se poseen 8 opciones: resetear la IMU (con el botón "r"), iniciar la comunicación con el helico(con el botón "s"), iniciar el control externo(con el botón "y"), apagar la comunicación con el helicóptero(con la "barra espaciadora"), apagar el control externo (con el botón "j"), modificar en 20 grados positivos la yaw deseada(con el botón "m"), modificar en 20 grados negativos la yaw deseada(con el botón "z"), aumentar el empuje en 0.02(con el botón "q"), disminuir el empuje en 0.02(con el botón "p"). Al iniciar el control se iniciará un control sobre el helico que hará seguir las referencias que se le manden(en este programa en ejecución solo se puede modificar la yaw). La diferencia entre este programa y el anterior es la ley de control. Y grabará los valores en dos archivos especificados en : linea(hover) 414 "//opt/XUFO-PCI-Pedro/XUFO_EXP13.txt" 5 columnas: yaw deseada, yaw medida, acción de control de yaw, empuje. Linea (main) 155 "/opt/Alfonso-XUFO-PCI-Pedro/XUFO-EXP0.txt" 5 columnas: tiempo, referencia, yaw medida, acción control, empuje. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.

c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).

d) La caja de amplificación (caja negra) conectada a la emisora.

e) La emisora conectada a la alimentación.

f) El receptor de la IMU conectado a una entrada USB del ordenador.

g) El joystick conectado a una entrada USB del ordenador.

h) El X-UFO conectado con la batería o fuente de alimentación.

3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start
4. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: cd /opt/XUFO-PCI-PEDRO(Ley de Control)
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.
8. Se inicia el programa con el comando: ./bin/hover -n
9. Si no da ninguna señal de error, pulsar la tecla “s” para iniciar la conexión ordenador - X-UFO.
10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla “y” para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.
13. Protocolo Apagado X-UFO
14. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
15. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
16. Terminar el programa de ordenador. Pulsar tecla “Esc”.
17. Desconectar la batería del X-UFO.
18. El X-UFO esta apagado.
19. Separar baterías del X-ufo y situarlas en su caja.
20. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.
21. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
22. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
23. Recoger todo y guardarlo donde corresponda

Detalle de parámetros configurables En este caso el programa se modifica en ejecución. Las constantes de la ley de control están en las lineas(32-45) y (393-403):

```
double  hover::k1=  0.001;
//0.00001;// ganancia D del control de velocidad yaw
double  hover::k2= 0;
//0.003;// ganancia P del control de yaw
double  hover::kpitch1=0.01;
//0.0001;// ganancia D del control de velocidad pitch
double  hover::kpitch2= 0;
//0.003;//ganancia P del control de pitch
double  hover::kroll1=0.01;
//0.0001;   // ganancia D del control de velocidad roll
double  hover::kroll2=0;
// 0.003;// ganancia P del control de roll
double hover::yaw_d = 0.0;
double hover::u_s=0.0;
double yaw_j=2.5,pitch_j=0.0,roll_j=2.5;
        // variables del control de yaw
double satura1=0.01, satura2=0.2;//la 1ª k1 2ª k2 (inicialmente estaba en 0.3)
double sigma1, sigma2;
        // variables del control de pitch
double satpitch1=0.01, satpitch2=0.2;//Antes estaban en 0.3
double sigmapitch1, sigmapitch2;
        // variables del control de roll
double satroll1=0.01, satroll2=0.2;
double sigmaroll1, sigmaroll2;
```

El código de control es:

```
        //Comienzo de control

        //////////CODIGO DE CONTROL

ControlFunctionJoystickUfo();
/*Lee central inercial (Euler Angles y Euler rate angles)
 y cambiar el valor a grados */
yaw =( -1*(180/HOVER_PI)*imu_tdmgx2->GetYaw()) ;
pitch = (180/HOVER_PI)*imu_tdmgx2->GetPitch();
roll = (180/HOVER_PI)*imu_tdmgx2->GetRoll();
yaw_rate = (180/HOVER_PI)*imu_tdmgx2->GetYawRate();
pitch_rate = (180/HOVER_PI)*imu_tdmgx2->GetPitchRate();
roll_rate = (180/HOVER_PI)*imu_tdmgx2->GetRollRate();

/*Se ejecuta accion de control*/
if ( bRunControlyaw == true)
{sigma2= saturacion(k1 * yaw_rate,satura1);
sigma1 = saturacion(k2 * (yaw-yaw_d),satura2);
yaw_c = -sigma2 - sigma1;
sigmapitch2= saturacion(kpitch1 * pitch_rate, satpitch1);
sigmapitch1 = saturacion(kpitch2 * (pitch-pitch_d), satpitch2);
pitch_c = (sigmapitch2 + sigmapitch1);
sigmaroll2= saturacion(kroll1 * roll_rate, satroll1);
```

```
sigmaroll1 = saturacion(kroll2 * (roll-roll_d), satroll2);
roll_c = (sigmaroll2 + sigmaroll1);
}

u_sal= u_j-u_s;
pitch_s = pitch_j+pitch_c;
roll_s  = roll_j+roll_c;
y_sal= yaw_s= yaw_j+yaw_c;
```

XUFO-PCI-guillaume

Aplicación Con este programa se poseen 8 opciones: resetear la IMU (con el botón "r"), iniciar la comunicación con el helico(con el botón "s"), iniciar el control externo(con el botón "y"), apagar la comunicación con el helicóptero(con la "barra espaciadora"), apagar el control externo (con el botón "j"), modificar en 20 grados positivos la yaw deseada(con el botón "m"), modificar en 20 grados negativos la yaw deseada(con el botón "z"), aumentar el empuje en 0.02(con el botón "q"), disminuir el empuje en 0.02(con el botón "p"). Al iniciar el control se iniciará un control sobre el helico que hará seguir las referencias que se le manden(en este programa en ejecución solo se puede modificar la yaw). La diferencia entre este programa y el anterior es la ley de control. Y grabará los valores en dos archivos especificados en: linea (hover) 417:"/opt/XUFO-PCI-guillaume/XUFO_EXP24032010_3.txt" 5 columnas: yaw deseada, yaw medida, acción de control de yaw, empuje. Linea (main) 156:"//opt/XUFO-PCI-guillaume/XUFO-EXP24032010_2.txt" 5 columnas: tiempo, referencia, yaw medida, acción control, empuje. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.
   c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).
   d) La caja de amplificación (caja negra) conectada a la emisora.
   e) La emisora conectada a la alimentación.
   f) El receptor de la IMU conectado a una entrada USB del ordenador.
   g) El joystick conectado a una entrada USB del ordenador.
   h) El X-UFO conectado con la batería o fuente de alimentación.
3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start
4. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: cd /opt/XUFO-PCI-guillaume
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.
8. Se inicia el programa con el comando: ./bin/hover -n
9. Si no da ninguna señal de error, pulsar la tecla "s" para iniciar la conexión ordenador - X-UFO.

10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla "y" para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.
13. Protocolo Apagado X-UFO
14. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
15. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
16. Terminar el programa de ordenador. Pulsar tecla "Esc".
17. Desconectar la batería del X-UFO.
18. El X-UFO esta apagado.
19. Separar baterías del X-ufo y situarlas en su caja.
20. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.
21. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
22. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
23. Recoger todo y guardarlo donde corresponda

Detalle de parámetros configurables En este caso el programa se modifica en ejecución. Las constantes de la ley de control están en las lineas(32-39) y (393-403):

```
double  hover::k1=0.002;    // ganancia D del control de velocidad yaw
double  hover::k2= 0.003; // ganancia P del control de yaw
double  hover::kpitch1=0.002;    // ganancia D del control de velocidad pitch
double  hover::kpitch2= 0.012; // ganancia P del control de pitch
double  hover::kroll1=0.002;    // ganancia D del control de velocidad roll
double  hover::kroll2= 0.012; // ganancia P del control de roll
        // variables del control de yaw
double satura1=0.4, satura2=0.4;//  la 1ª k1 2ª k2 (inicialmente estaba en 0.3)
double sigma1, sigma2;
        // variables del control de pitch
double satpitch1=0.4, satpitch2=0.4;//Antes estaban en 0.3
double sigmapitch1, sigmapitch2;
        // variables del control de roll
double satroll1=0.4, satroll2=0.4;
double sigmaroll1, sigmaroll2;
```

El código de control es:

```
//Comienzo de control

//////////CODIGO DE CONTROL
ControlFunctionJoystickUfo();
        /*Lee central inercial (Euler Angles y Euler rate angles)
          y cambiar el valor a grados */
```

```
yaw =( -1*(180/HOVER_PI)*imu_tdmgx2->GetYaw()) ;
pitch = (180/HOVER_PI)*imu_tdmgx2->GetPitch();
roll = (180/HOVER_PI)*imu_tdmgx2->GetRoll();
yaw_rate = (180/HOVER_PI)*imu_tdmgx2->GetYawRate();
pitch_rate = (180/HOVER_PI)*imu_tdmgx2->GetPitchRate();
roll_rate = (180/HOVER_PI)*imu_tdmgx2->GetRollRate();

        /*Se ejecuta accion de control*/
if ( bRunControlyaw == true)
{sigma2= saturacion(k1 * yaw_rate,satura1);
sigma1 = saturacion(k2 * (yaw-yaw_d),satura2);
yaw_c = -sigma2 - sigma1;
sigmapitch2= saturacion(kpitch1 * pitch_rate, satpitch1);
sigmapitch1 = saturacion(kpitch2 * (pitch-pitch_d), satpitch2);
pitch_c = (sigmapitch2 + sigmapitch1);
sigmaroll2= saturacion(kroll1 * roll_rate, satroll1);
sigmaroll1 = saturacion(kroll2 * (roll-roll_d), satroll2);
roll_c = (sigmaroll2 + sigmaroll1);
}

u_sal= u_j-u_s;
pitch_s = pitch_j+pitch_c;
roll_s  = roll_j+roll_c;
y_sal= yaw_s= yaw_j+yaw_c;
```

XUFO-PCI-Andreu

Aplicación Con este programa se poseen 8 opciones: resetear la IMU (con el botón "r"), iniciar la comunicación con el helico(con el botón "s"), iniciar el control externo(con el botón "y"), apagar la comunicación con el helicóptero(con la "barra espaciadora"), apagar el control externo (con el botón "j"), modificar en 20 grados positivos la yaw deseada(con el botón "m"), modificar en 20 grados negativos la yaw deseada(con el botón "z"), aumentar el empuje en 0.02(con el botón "q"), disminuir el empuje en 0.02(con el botón "p"). Al iniciar el control se iniciará un control sobre el helico que hará seguir las referencias que se le manden(en este programa en ejecución solo se puede modificar la yaw). La diferencia entre este programa y el anterior es la ley de control. Y grabará los valores en dos archivos especificados en: linea (hover)"//opt/XUFO-PCI-Andreu/XUFO_EXP19042010_1.txt" 5 columnas: yaw deseada, yaw medida, acción de control de yaw, empuje. linea (main)"/opt/XUFO-PCI-Andreu/XUFO-EXP19042010_2.txt" 5 columnas: tiempo, referencia, yaw medida, acción control, empuje. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.
   c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).
   d) La caja de amplificación (caja negra) conectada a la emisora.
   e) La emisora conectada a la alimentación.

f) El receptor de la IMU conectado a una entrada USB del ordenador.

g) El joystick conectado a una entrada USB del ordenador.

h) El X-UFO conectado con la batería o fuente de alimentación.

3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start
4. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: cd /opt/XUFO-PCI-Andreu
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.
8. Se inicia el programa con el comando: ./bin/hover -n
9. Si no da ninguna señal de error, pulsar la tecla "s" para iniciar la conexión ordenador - X-UFO.
10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla "y" para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.
13. Protocolo Apagado X-UFO
14. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
15. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
16. Terminar el programa de ordenador. Pulsar tecla "Esc".
17. Desconectar la batería del X-UFO.
18. El X-UFO esta apagado.
19. Separar baterías del X-ufo y situarlas en su caja.
20. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.
21. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
22. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
23. Recoger todo y guardarlo donde corresponda

Detalle de parámetros configurables En este caso el programa se modifica en ejecución. Las constantes de la ley de control están en las lineas(32-39) y (393-403):

```
double  hover::k1=0.002;    // ganancia D del control de velocidad yaw
double  hover::k2= 0.003; // ganancia P del control de yaw
double  hover::kpitch1=0.002;    // ganancia D del control de velocidad pitch
double  hover::kpitch2= 0.012; // ganancia P del control de pitch
double  hover::kroll1=0.002;    // ganancia D del control de velocidad roll
double  hover::kroll2= 0.012; // ganancia P del control de roll
        // variables del control de yaw
```

```
double satura1=0.4, satura2=0.4;//  la 1ª k1 2ª k2 (inicialmente estaba en 0.3)
double sigma1, sigma2;
        // variables del control de pitch
double satpitch1=0.4, satpitch2=0.4;    //Antes estaban en 0.3
double sigmapitch1, sigmapitch2;
        // variables del control de roll
double satroll1=0.4, satroll2=0.4;
double sigmaroll1, sigmaroll2;
```

El código de control es:

```
        //Comienzo de control

        //////////CODIGO DE CONTROL
ControlFunctionJoystickUfo();
        /*Lee central inercial (Euler Angles y Euler rate angles)
          y cambiar el valor a grados */
yaw =( -1*(180/HOVER_PI)*imu_tdmgx2->GetYaw()) ;
pitch = (180/HOVER_PI)*imu_tdmgx2->GetPitch();
roll = (180/HOVER_PI)*imu_tdmgx2->GetRoll();
yaw_rate = (180/HOVER_PI)*imu_tdmgx2->GetYawRate();
pitch_rate = (180/HOVER_PI)*imu_tdmgx2->GetPitchRate();
roll_rate = (180/HOVER_PI)*imu_tdmgx2->GetRollRate();

        /*Se ejecuta accion de control*/
if ( bRunControlyaw == true)
{sigma2= saturacion(k1 * yaw_rate,satura1);
sigma1 = saturacion(k2 * (yaw-yaw_d),satura2);
yaw_c = -sigma2 - sigma1;
sigmapitch2= saturacion(kpitch1 * pitch_rate, satpitch1);
sigmapitch1 = saturacion(kpitch2 * (pitch-pitch_d), satpitch2);
pitch_c = (sigmapitch2 + sigmapitch1);
sigmaroll2= saturacion(kroll1 * roll_rate, satroll1);
sigmaroll1 = saturacion(kroll2 * (roll-roll_d), satroll2);
roll_c = (sigmaroll2 + sigmaroll1);
}
u_sal= u_j-u_s;
pitch_s = pitch_j+pitch_c;
roll_s  = roll_j+roll_c;
y_sal= yaw_s= yaw_j+yaw_c;
```

XUFO-PCI-Peonza

Aplicación Con este programa se poseen 8 opciones: resetear la IMU (con el botón "r"), iniciar la comunicación con el helico(con el botón "s"), iniciar el control externo(con el botón "y"), apagar la comunicación con el helicóptero(con la "barra espaciadora"), apagar el control externo (con el botón "j"), modificar en 20 grados positivos la yaw deseada(con el botón "m"), modificar en 20 grados negativos la yaw deseada(con el botón "z"), modificar en 5 grados positivos el roll deseado (con el botón "q"), modificar en 5 grados negativos el roll deseado(con el botón "p"). Al iniciar el control se iniciará un control sobre el helico que hará seguir las

referencias que se le manden(en este programa en ejecución solo se puede modificar la yaw). La diferencia entre este programa y el anterior es la ley de control. Y grabará los valores en dos archivos: linea (hover) 417:"/opt/XUFO-PCI-Peonza/XUFO_EXP06052010_1.txt" 5 columnas: yaw deseada, yaw medida, acción de control de yaw, empuje. Linea (main) 157:"/opt/XUFO-PCI-Peonza/XUFO-EXP06052010_2.txt" 5 columnas: tiempo, referencia, yaw medida, acción control, empuje. Puesta en marcha y paro

1. El ordenador(con tarjeta PCI) se inicia con Linux.
2. Comprobar que todas las partes hardware están conectadas.
   a) La tarjeta de adquisición de datos conectada a la caja de conexiones(caja blanca).
   b) La caja de conexiones(caja blanca) en modo XUFO ROBOTNICK.
   c) La caja de conexiones(caja blanca) conectada a la caja de amplificación(caja negra).
   d) La caja de amplificación (caja negra) conectada a la emisora.
   e) La emisora conectada a la alimentación.
   f) El receptor de la IMU conectado a una entrada USB del ordenador.
   g) El joystick conectado a una entrada USB del ordenador.
   h) El X-UFO conectado con la batería o fuente de alimentación.
3. Se inicia la tarjeta de adquisición de datos con el comando(en una terminal): /etc/init.d/mcapi start
4. Se comprueba la activación de la misma con el comando : lsmod — grep mcapi
5. Se busca la carpeta del programa que se vaya a ejecutar, un ejemplo de comando: cd /opt/XUFO-PCI-Peonza
6. Se inicia el X-UFO: Botón en la placa superior de la electrónica del aparato.
7. Se espera a la finalización del iniciado del aparato.
8. Se inicia el programa con el comando: ./bin/hover -n
9. Si no da ninguna señal de error, pulsar la tecla "s" para iniciar la conexión ordenador - X-UFO.
10. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). En caso de no funcionar comprobar trimmers de la emisora.
11. Si no da ninguna señal de error, pulsar la tecla "y" para iniciar el control externo del programa.
12. El X-UFO esta encendido y listo para trabajar.
13. Protocolo Apagado X-UFO
14. Con el joystick hacer el movimiento: thrust al mínimo y yaw a uno de los dos extremos(es indiferente cual). Esto debería apagar los motores del X-UFO.
15. Apagar el X-UFO físicamente. Pulsar el botón en la placa superior de la electrónica del aparato.
16. Terminar el programa de ordenador. Pulsar tecla "Esc".
17. Desconectar la batería del X-UFO.
18. El X-UFO esta apagado.
19. Separar baterías del X-ufo y situarlas en su caja.
20. Separar cable y conexión para configuración del X-UFO y situarlas en su caja.

21. Desconectar emisora de alimentación y de caja de amplificación(caja negra). Poner cable de alimentación en caja.
22. Desconectar Joystick y receptor de IMU del ordenador y situarlos en sus correspondientes cajas.
23. Recoger todo y guardarlo donde corresponda

Detalle de parámetros configurables En este caso el programa se modifica en ejecución. Las constantes de la ley de control están en las lineas(32-39) y (397-406):

```
double  hover::k1=0.002;   // ganancia D del control de velocidad yaw
double  hover::k2= 0.030; // ganancia P del control de yaw
double  hover::kpitch1=0.002;   // ganancia D del control de velocidad pitch
double  hover::kpitch2= 0.012; // ganancia P del control de pitch
double  hover::kroll1=0.002;   // ganancia D del control de velocidad roll
double  hover::kroll2= 0.012; // ganancia P del control de roll
        // variables del control de yaw
double satura1=0.4, satura2=0.4;
double sigma1, sigma2;
        // variables del control de pitch
double satpitch1=0.4, satpitch2=0.4;
double sigmapitch1, sigmapitch2;
        // variables del control de roll
double satroll1=0.4, satroll2=0.4;
double sigmaroll1, sigmaroll2;
```

El código de control es:

```
        //Comienzo de control

        //////////CODIGO DE CONTROL
ControlFunctionJoystickUfo();
        *Lee central inercial (Euler Angles y Euler rate angles)
         y cambiar el valor a grados */
yaw =( -1*(180/HOVER_PI)*imu_tdmgx2->GetYaw()) ;
pitch = (180/HOVER_PI)*imu_tdmgx2->GetPitch();
roll = (180/HOVER_PI)*imu_tdmgx2->GetRoll();
yaw_rate = (180/HOVER_PI)*imu_tdmgx2->GetYawRate();
pitch_rate = (180/HOVER_PI)*imu_tdmgx2->GetPitchRate();
roll_rate = (180/HOVER_PI)*imu_tdmgx2->GetRollRate();

        /*Se ejecuta accion de control*/
if ( bRunControlyaw == true)
{sigma2= saturacion(k1 * yaw_rate,satura1);
sigma1 = saturacion(k2 * (yaw-yaw_d),satura2);
yaw_c = -sigma2 - sigma1;
sigmapitch2= saturacion(kpitch1 * pitch_rate, satpitch1);
sigmapitch1 = saturacion(kpitch2 * (pitch-pitch_d), satpitch2);
pitch_c = (sigmapitch2 + sigmapitch1);
sigmaroll2= saturacion(kroll1 * roll_rate, satroll1);
sigmaroll1 = saturacion(kroll2 * (roll-roll_d), satroll2);
roll_c = (sigmaroll2 + sigmaroll1);
```

```
}

u_sal= u_j-u_s;
pitch_s = pitch_j+pitch_c;
roll_s  = roll_j+roll_c;
y_sal= yaw_s= yaw_j+yaw_c;
```

5.4. Protocolo de desarrollo e implementación de los algoritmos de control

- Diseño teórico del algoritmo de control.
- Implementación en Simulink (continuo) del algoritmo de control.
- Implementación en Simulink (discreto) del algoritmo de control.
- Implementación en Matlab (discreto) del algoritmo de control.
- Implementación en código C y ejecución en Linux