Universidad Nacional Autónoma de México

Facultad de Ingeniería

Estabilización de un helicóptero a escala mediante sistemas neuro-difusos

T E S I S

Para obtener el título de:

Ingeniero Mecatrónico

Presentan:

David Retana Pérez

Aldo Enrique Vargas Moreno

Director de Tesis: Ing. Serafín Castañeda Cedeño

Agradecimientos

Generales:

A Gabriela Valencia por habernos patrocinado a lo largo de esta tesis, confiar en que funcionaria, ademas de su amabilidad, cariño y consejos que fueron claves para el desarrollo de este proyecto.

Al Ing. Serafín Castañeda y Dr. Marcos González, por sus invaluables asesorías y por el tiempo que nos dedicaron. Sin su apoyo la realización de este trabajo no se hubiera completado.

A PROTECO por alojarnos durante el trayecto de nuestras carreras.

Aldo Vargas

A Valve software, por haber creado tan maravillosos juegos (Half-Life 1 y 2, Portal) y como olvidar a Blizzard Entertainment por títulos como WoW, Warcraft y Diablo.

A mi familia principal (Raul, Rosalba, Spayro y Trinidad) por haberme apoyado todo tiempo, por aguantar mi especial humor y por crearme. También a Aldux por haber aportando maravillosas ideas a este proyecto ademas de ser un compañero emprendedor.

David Retana

La culminación del proyecto es soportado por los apoyos de mi familia haciendo énfasis en la ayuda incondicional de la Dra. María Guadalupe Pérez H.

Sin menospreciar a los demás que por no mencionar puntualmente sus nombres también les extiendo este agradecimiento.

Contenido

Introducción

Los helicópteros a escala son cada ves mas populares para plataformas de vehículos aéreos no tripulados (UAV's), este tipo de tecnología ha experimentado un importante desarrollo motivado por sus ventajas en misiones con riesgo para vuelos tripulados o con requerimientos de elevada maniobrabilidad o dimensiones reducidas en las aeronaves. Los costes de explotación de los UAV's también pueden ser menores que los vuelos tripulados en diferentes aplicaciones. Los vehículos aéreos no tripulados de ala rotatoria (RUAV's) son básicamente helicópteros autónomos.

Recientemente, la evolución de las tecnologías de los UAV's, la miniaturización de los sensores y los progresos en sistemas de control apuntan hacia su uso generalizado en aplicaciones como desastres naturales, búsqueda y rescate, vigilancia, inspección de instalaciones y estructuras, cinematografía y realización de mapas del terreno.

En algunas de las mencionadas aplicaciones se necesitan funcionalidades como el despegue-aterrizaje vertical y el vuelo estacionario (*hovering*). Los helicópteros autónomos son las plataformas más utilizadas en robótica aérea debido a las propiedades mencionadas anteriormente y a una mayor carga útil que la ofrecida por otras aeronaves de despegue y aterrizaje vertical.

Pilotar un helicóptero a escala es un gran reto debido a que se requiere conocer y controlar los mandos principales como el colectivo, cíclico lateral y longitudinal, colectivo de cola y velocidad del motor. Mantener el helicóptero en *hovering* es una de las maniobras mas difíciles, porque requiere manipular los controles principales de forma activa ya que el helicóptero no tiende a un estado estable estacionario.

La necesidad que se cubre en este trabajo es que el usuario pueda pilotar de forma más transparente el helicóptero dando una mejor experiencia de vuelo, reducir las probabilidades de estrellar el helicóptero y por ende reducir los costos de reparación debido a choques.

El objetivo de esta tesis, es el desarrollo de un sistema de control para la estabilización del helicóptero a escala, de tal forma que sea fácil el manejo para las personas con poca experiencia.

Para lograr que las personas puedan pilotar el helicóptero de manera mas segura se promediará la señal de control con la señal del control del piloto, de esta manera se le provee estabilidad al helicóptero sin necesidad de ser un piloto experto.

Para poder lograr este objetivo, es importante considerar:

- La necesidad de contar con un piloto de radio-control para el desarrollo del sistema autónomo y por razones de seguridad.
- La importancia del mantenimiento mecánico y prueba de todos los componentes mecánicos, eléctricos y electrónicos antes de cada vuelo, lo que es especialmente critico en plataformas de bajo costo que se derivan de helicópteros de radio-control.
- La relevancia de las limitaciones de peso y consumo, particularmente para helicópteros pequeños que imponen importantes restricciones en el hardware y software que puede ejecutarse abordo. Estas limitaciones hacen especialmente interesante el empleo de avances recientes en miniaturización de sensores y actuadores y el empleo de sistemas de control.
- Relevancia del diseño mecatrónico involucrando conjuntamente aspectos mecánicos, sensores y controladores a bordo. Interés de la aplicación de técnicas de "hardware in the loop" y eventualmente de plataformas para prueba en interiores.
- Interés de la aplicación de técnicas de detección e identificación de fallos en sensores y actuadores.
- Requerimientos estrictos de seguridad en las pruebas y vuelos del helicóptero, sin olvidar la importante potencia que desarrolla el rotor principal y las consecuencias de los fallos.

Para alcanzar el objetivo del sistema de estabilización es importante conocer la estimación del estado del helicóptero. Para realizar esta estimación del estado resulta conveniente aplicar técnicas de fusión sensorial que utilicen las medidas de los sensores y que tengan en cuenta sus propiedades estadísticas, tales como un filtro Kalman ó *DCM* (Direction Cosine Matrix).

Lo que se logró en este trabajo es utilizar una unidad de medición inercial (*IMU, por sus siglas en ingles*) y convertirlo mediante software en un sistema de referencia de actitud y rumbo (*AHRS, por sus siglas en ingles*) bastante confiable, con esto reduciendo el gran costo que conlleva un sistema de estas características. Un *AHRS* certificado por la Administración Federal de Aviación (*FAA, por sus siglas en ingles*) puede costar aproximadamente $15,000 dólares.

Una vez conocida la estimación del helicóptero, se procederá a aplicar un control para poder determinar la señal que se enviará a los servos para que el helicóptero se estabilice. Cabe destacar la importancia de conocer la relación entrada-salida para facilitar el diseño del algoritmo de control, ya

que el helicóptero es un sistema multivariable y subactuado cuya dinámica presenta gran complejidad. No obstante, haciendo ciertas simplificaciones es posible obtener la relación entrada-salida básicas que capturen los principales efectos dinámicos.

La razón por la cuál se escogieron sistemas neuro-difusos es porque hace posible la aplicación de métodos adaptables que permiten obtener un comportamiento robusto frente a incertidumbres del modelado, inclusive evitar la parte del modelo.

Otro de los logros de este trabajo consiste en la creación de un sistema de control difuso totalmente modificable y adaptable, no importando el lenguaje de programación que se use (cambiando solamente la sintaxis) que puede ser usado en diversas aplicaciones, tales como: péndulo invertido, cámaras estabilizadas, dirección de navegación en vehículos aéreos y terrestres, sistemas de control industriales, mejora de eficiencia del uso de combustible en motores, etc.

En el capítulo 1 de este trabajo se discuten los temas relacionados con el helicóptero, como sus características principales, su funcionamiento, así como el modelado matemático del mismo. Cabe resaltar que con los sistemas neuro-difusos podemos evitarnos modelar matemáticamente al helicóptero. El capítulo 2 trata de la instrumentación del helicóptero que es una parte esencial del trabajo, ya que para controlarlo es necesario conocer la orientación del mismo. En el capítulo 3 tratamos la parte del control, que incluyen los temas de lógica difusa y redes neuronales. En el cuarto capítulo mostramos las pruebas realizadas así como los resultados obtenidos.

1.- Principio de funcionamiento del helicóptero

La posición y orientación del helicóptero se controla normalmente mediante 5 variables: la inclinación colectiva de las palas del rotor principal (colectivo) que tiene un efecto directo en la altura del helicóptero (eje z en el sistema X-Y-Z); el cíclico longitudinal que modifica el ángulo de cabeceo del helicóptero (rotación sobre el eje Yb en el sistema Xb - Yb - Zb) y la traslación longitudinal; el cíclico lateral, que afecta el ángulo de balanceo (rotación sobre el eje Xb en el sistema Xb - Yb - Zb) y la traslación lateral; el rotor de cola, el cual controla el ángulo de guiñada del helicóptero (rotación sobre el eje Zb en el sistema Xb - Yb - Zb); y el control de la potencia del motor. La figura 1.1 muestra los sistemas de referencia que suelen utilizarse para expresar sus parámetros principales de funcionamiento.

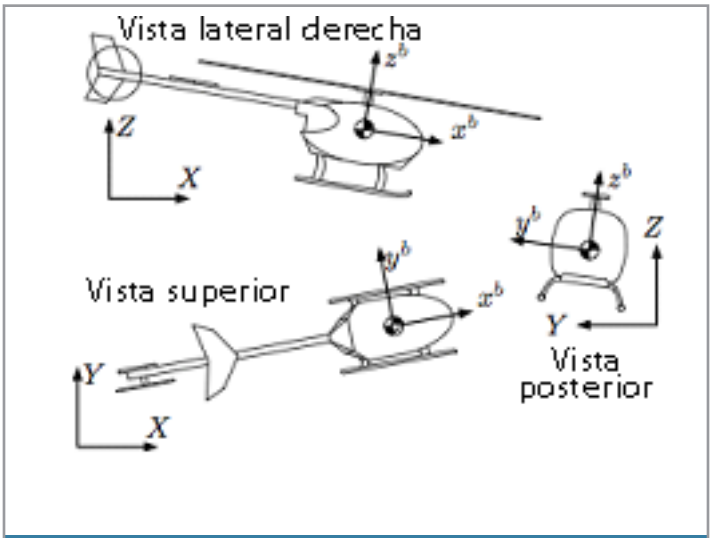


Fig. 1.1 Sistema inercial (X Y Z) y local (Xb Yb Zb) [1]

Los componentes básicos de un helicóptero son: rotor principal, cabina (en donde se encuentra la carga útil ó *payload*), estructura (que aloja varios otros componentes), máquina, transmisión, sistema de antipar, tren de aterrizaje (que pueden ser o llantas ó patín ó flotadores). Se aprecian en la figura 1.2.

Es propulsado horizontalmente mediante la inclinación del rotor principal, mientras que el vertical se logra variando el ángulo de ataque de las palas y la potencia del motor.

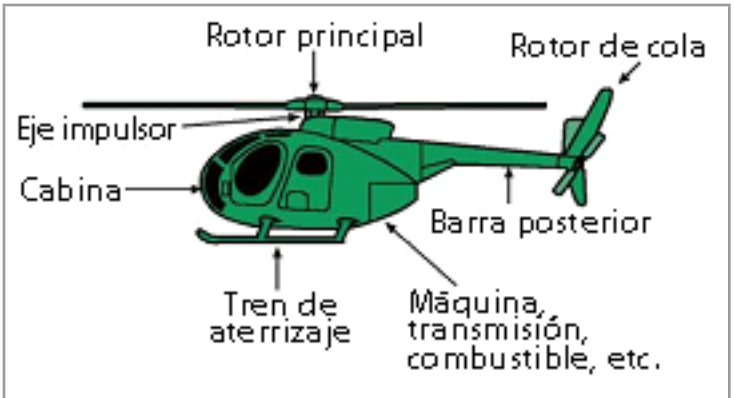


Fig. 1.2 Partes principales de un helicóptero [1]

Rotor principal

Las palas del rotor principal, tienen una forma aerodinámica similar a las alas de un avión, es decir, un perfil alar. Al girar el rotor este tipo de perfil hace que se genere sustentación, la cual eleva al helicóptero.

Hay dos tipos de rotores principales, los sencillos y los duales.

Los helicópteros requieren de un sistema antipar para compensar el momento de fuerza de reacción a causa del giro del rotor del rotor principal.

En la figura 1.3 se puede apreciar el mecanismo que hace que las palas puedan ser inclinadas para generar sustentación, además de la inclinación de todo el rotor para mover a todo el helicóptero a alguna posición deseada.

Se añade el rotor de cola para controlar el efecto de giro y compensar los momentos de fuerza de reacción.

En los helicópteros de doble hélice, los rotores giran en direcciones opuestas y no se necesita un efecto "antipar" del rotor de cola, ya que el par generado por cada rotor dual, se neutraliza uno con el otro.

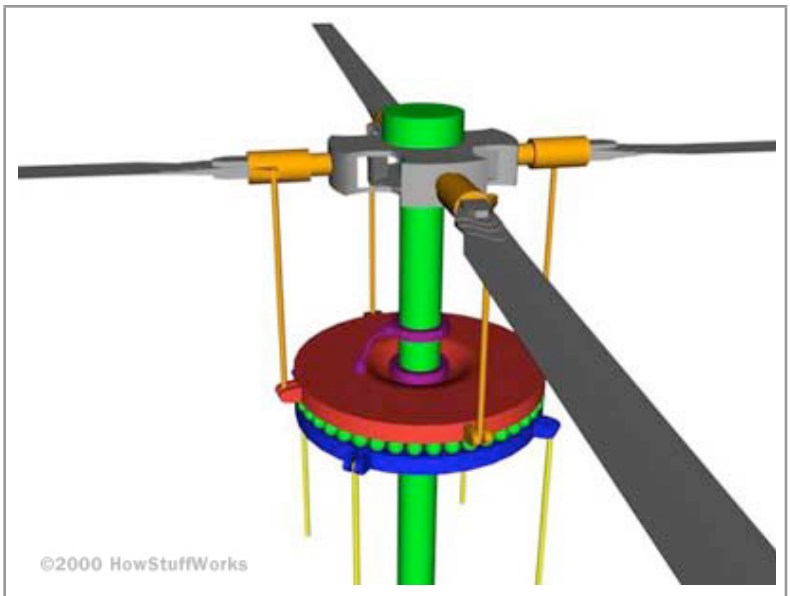


Fig. 1.3 Rotor principal [2]

El rotor principal no solo sirve para mantener el helicóptero en el aire, así como para elevarlo o descender, sino también para impulsarlo hacia delante o hacia atrás, hacia los lados o en cualquier

dirección. Esto se consigue con un mecanismo que hace variar el ángulo de incidencia (inclinación) de las palas del rotor principal dependiendo de su posición.

El helicóptero es una aeronave versátil, ya que se puede mover en todas direcciones (figura 1.4). Puede volar hacia atrás, adelante, izquierda, derecha, arriba, abajo, y rotar en su propio eje, además de mantenerse en sustentación totalmente inmóvil, se conoce como vuelo estático (*hovering*).

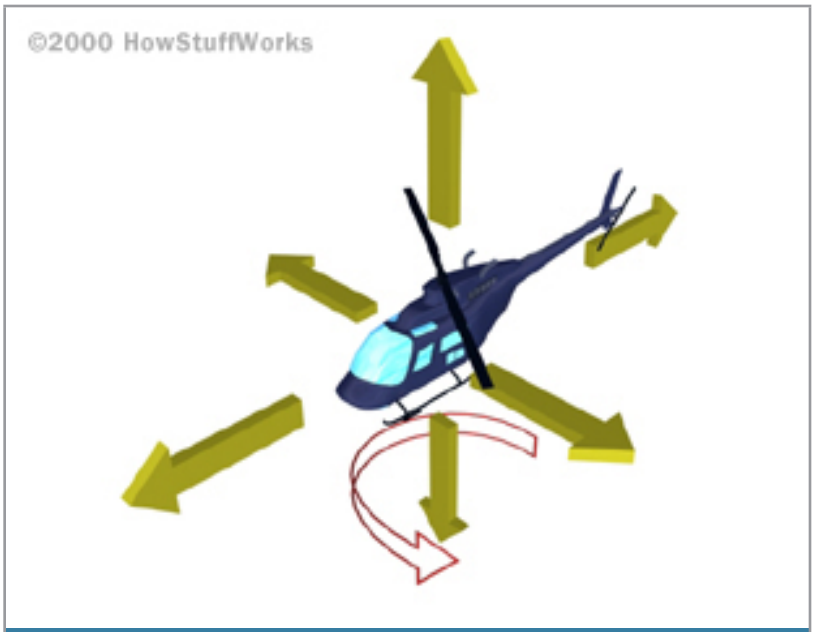


Fig. 1.4 Movimientos posibles en un helicóptero [2]

Rotor trasero (o de cola)

Es el rotor encargado de generar un efecto antipar (figura 1.5) para evitar que le helicóptero gire en sentido opuesto al sentido de giro del rotor principal.

Esto se logra cambiando el ángulo de ataque de las palas del rotor trasero, haciendo que se genere mayor o menor fuerza de sustentación, según sea el caso, y así poder hacer girar el helicóptero para cualquier lado deseado.

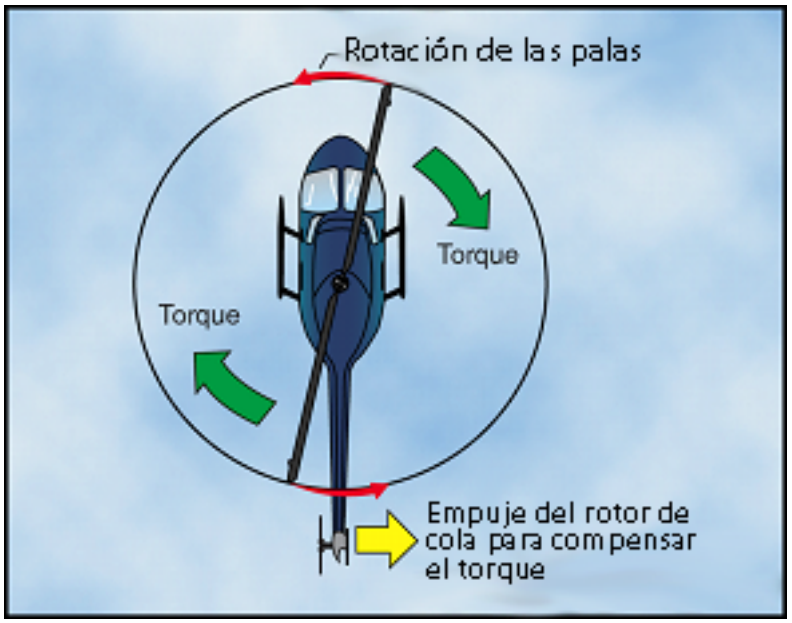


Fig. 1.5 Efecto del sistema antitorque [2]

Se tiene que hacer variar el empuje del sistema antipar para mantener el control direccional no importando si el par del rotor principal cambia, o se hacen cambios de dirección mientras se encuentra en sustentación (*hovering*).

Existen 3 tipos de sistemas antipar, sin contar el sistema de rotores duales.

El primero es el más sencillo, ya que consta de una pala de mucho menor tamaño que la pala del rotor principal, colocada a uno de los costados de la parte posterior del

helicóptero.

El segundo tipo tiene el rotor de cola encapsulado que es un tipo de ventilador colocado "en línea" con la cola del helicóptero. Se le conoce como *Fenestron* (figura 1.6), son menos susceptibles a dañarse por obstrucción, gran reducción de ruido y aumenta la seguridad del personal en tierra.

Fig. 1.6 Sistema Fenestron [3]

El tercero se le conoce como *NOTAR* (*NO TAil Rotor*) realiza la compensación o el efecto de antipar mediante un chorro de aire de dirección y fuerza variables mediante los controles. Se muestra en la figura 1.7.

El sistema de rotores duales, o rotores coaxiales (figura 1.8), son una pareja de de rotores girando en direcciones opuestas, pero montados sobre un único mástil, con el mismo eje de rotación, uno encima de otro. El par generado por un rotor se compensa con el par que genera el otro rotor, mientras ambos tengan la misma velocidad.

El sistema de rotores en *tandem*, mostrado en la figura 1.9, tienen 2 rotores montados uno enfrente de otro en el sentido longitudinal del helicóptero, y el efecto del par no importa, porque se comportan como un sistema de rotores coaxiales.

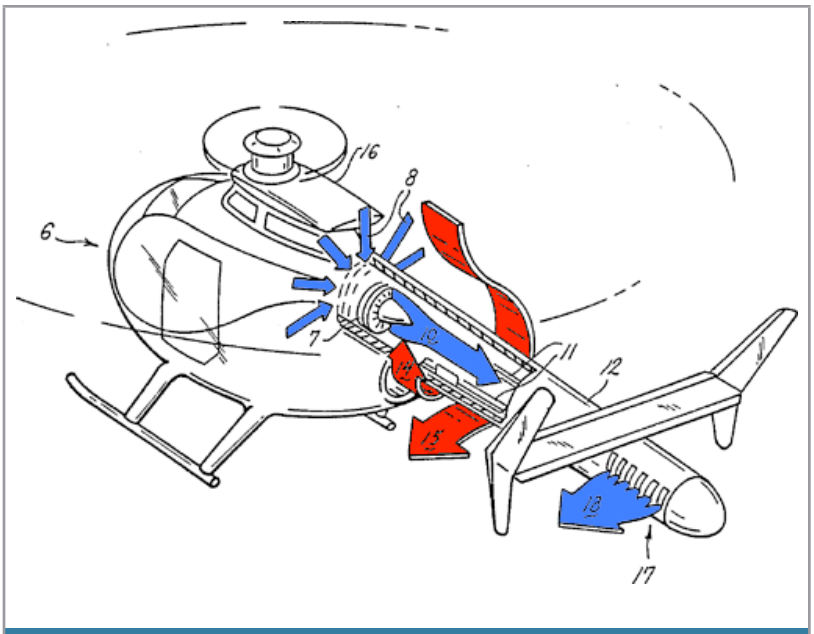


Fig. 1.7 Sistema Notar [3]

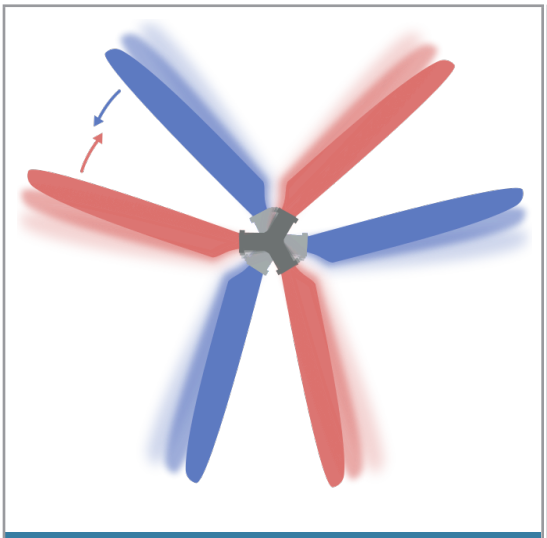


Fig. 1.8 Sistema rotores coaxiales [3]

Fig. 1.9 Sistema rotores tandem (Chinook) [3]

Controles de vuelo

Un helicóptero requiere un sistema de mandos y control que permita libertad de maniobras sobre los 3 ejes, vertical, longitudinal y transversal.

Para conseguir los movimientos indicados es preciso que aparezcan componentes de la resultante general que tiendan a cambiar la actitud del helicóptero. La figura 10 muestra los controles como se ven en una cabina real de helicóptero.

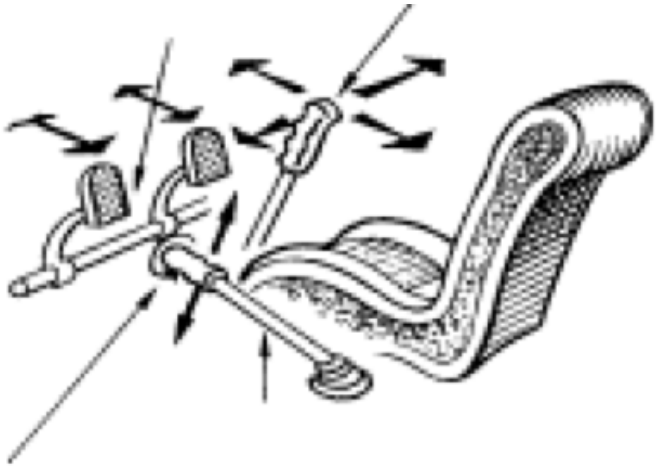


Fig. 1.10 Controles en un helicóptero [3]

Actitud (Attitude)

Los ángulos de navegación son un tipo de ángulos de Euler y se usan para describir la orientación de un objeto en 3 dimensiones. Los ejes se muestran en la figura 1.11.

Son 3 coordenadas angulares que definen un tiedro rotado desde otro que se considera el sistema de referencia.

En una aeronave, existen 3 rotaciones principales, guiñada, cabeceo y alabeo.

El ángulo de guiñada (*yaw*) es la rotación respecto a un eje vertical.

El ángulo de cabeceo (*pitch*) es la rotación respecto al eje ala-ala.

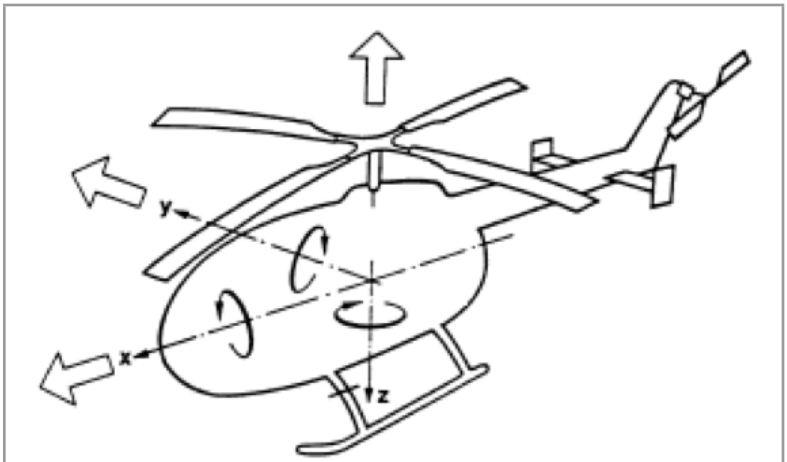


Fig. 1.11 Ejes de movimiento en un helicóptero [3]

El ángulo de alabeo (*roll*) es la rotación intrínseca alrededor del eje ortogonal al eje ala-ala.

Un helicóptero tiene 6 grados de libertad (GDL) y para poder actuar sobre estos GDL, basta con 4 controles independientes:

- Mov. Vertical
- Mov. longitudinal
- Mov. Lateral
- Mov. guiñada

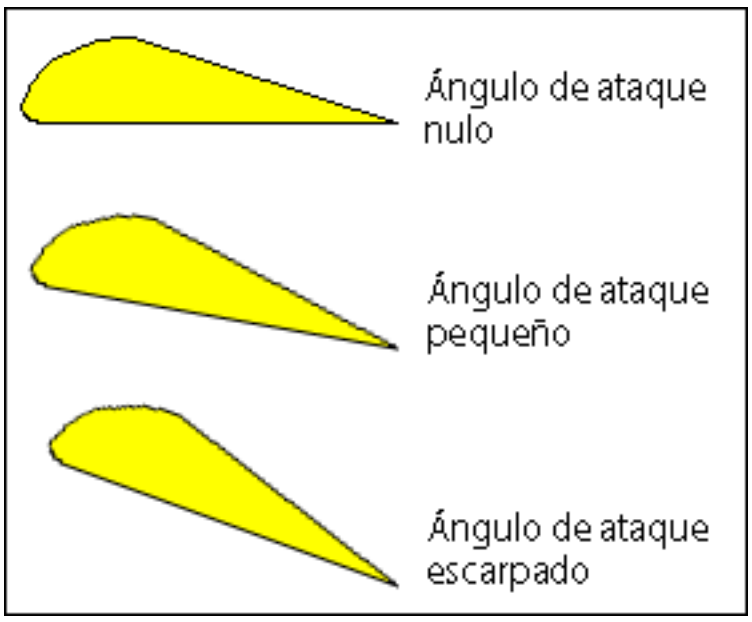


Fig. 1.12 Angulos de ataque en las palas [3]

Los nombres de los cuatro controles son: colectivo, acelerador, cíclico y antipar.

Control de paso colectivo

Es el responsable del desplazamiento vertical del helicóptero, junto con el acelerador.

Este control cambia simultáneamente el ángulo de ataque (figura 1.12) de todas las palas del rotor principal, de ahí el nombre de colectivo, provocando que se aumente o

disminuya la sustentación, por lo tanto, se modifica la componente vertical de la fuerza y se desplaza el helicóptero sobre el eje vertical.

Cuando se cambia el ángulo de ataque de las palas, se provoca un cambio en el arrastre (*drag*), lo que hace que se afecten las revoluciones del motor principal. Es por esto que el control de paso colectivo esta unido al control de potencia del motor. Cuando el ángulo de las palas aumenta, se tiene que compensar aumentando las revoluciones del motor.

Control de paso cíclico

El control de paso cíclico es aquel que proporciona el control longitudinal y lateral del helicóptero.

Cuenta con un mecanismo capaz de inclinar el plano de rotación del disco del rotor en el sentido del vuelo deseado, así aparecerá una componente de la sustentación del rotor en la dirección deseada.

El comportamiento del rotor es como el de un giroscopio, junto con su fenómeno de precesión (movimiento asociado con el cambio de dirección en el espacio que experimenta el eje instantáneo de rotación de un cuerpo).

Un rotor de helicóptero girando a las revoluciones habituales se comporta como un solido con punto fijo, si queremos inclinar el plano del disco en una dirección habrá que introducir un par en el rotor de 90 grados antes en el sentido de giro.

Control de sistema antipar

Como se mencionó anteriormente, el rotor de cola es el sistema antipar más común para el control sobre el eje de guiñada (además de compensar el par de reacción).

Aumentará o disminuirá la tracción del rotor trasero y, por lo tanto, el momento o par que produce respecto al centro de gravedad de la aeronave.

Cuando la potencia del motor aumenta, el rotor antipar debe de ser capaz de compensar el aumento de par asociado a este incremento de potencia. En la mayoría de los helicópteros el rotor trasero esta conectado mecánicamente el rotor principal.

Control de potencia del motor

Comúnmente conocido como acelerador en motores reciprocantes cuya función es regular las revoluciones por minuto en algunos helicópteros, principalmente en los de turbina. El acelerador esta unido mediante un mecanismo al control de paso colectivo para que; cuando las revoluciones disminuyan debido al arrastre generado por las palas aumenten mecánicamente las revoluciones del motor.

Paso colectivo y paso fijo

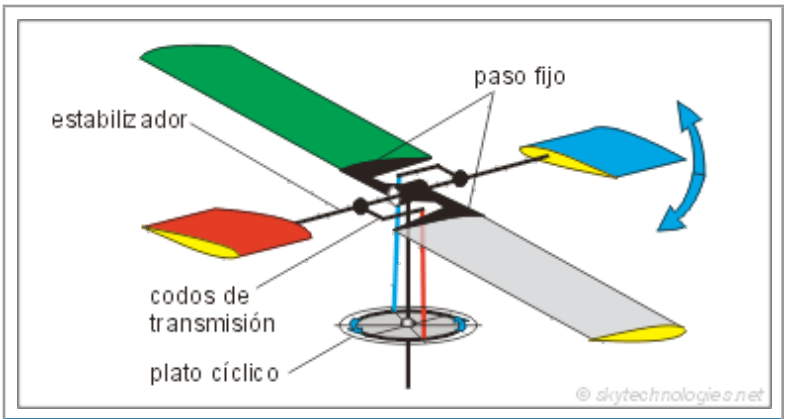


Fig. 1.13 Paso fijo [4]

Los helicópteros de paso fijo (figura 1.13) son aquellos que tienen palas que no se pueden girar alrededor del eje de palas, únicamente el estabilizador es capaz de girar.

Los codos de transmisión se ocupan para adelantar las varillas que recorren el plato cíclico en los 90 grados necesarios para que el rotor se incline hacia el mismo lado que se inclina el plato cíclico.

El los helicópteros de paso fijo el control de altura se consigue variando las revoluciones del motor.

En cambio en los helicópteros de paso colectivo es posible cambiar la incidencia de todas las palas a la vez subiendo o bajando el plato cíclico. Se aprecia el mecanismo de paso colectivo en la figura 1.14.

En este tipo de helicóptero no se controla la altura del helicóptero mediante las revoluciones del motor, sino por el cambio de paso.

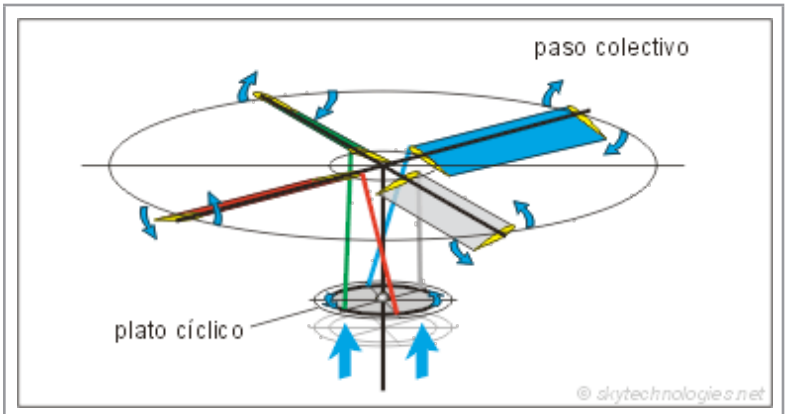


Fig. 1.14 Paso colectivo [4]

Helicópteros de radiocontrol

El radiocontrol es una técnica para controlar un objeto a distancia, de forma inalámbrica con dominio del objeto.

Los helicópteros de radiocontrol, tienen básicamente los tipos de control que un helicóptero normal, profundizaremos en 2 modelos, uno coaxial y otro de vuelo 3D.

Hay dos tipos de helicópteros de radiocontrol, los de motores a explosión y los eléctricos. Los primeros son semejantes a uno real, usando paso cíclico, mientras que los eléctricos, los hay de 2 tipos, los coaxiales, y los 3D.

Los eléctricos son accesibles y económicos, el costo de funcionamiento es sumamente reducido, porque solo requiere una batería.

Los de explosión son costosos para adquirirlos y mantenerlos, dado que están formados por mayor cantidad de elementos mecánicos, además de ser frecuentes sus averías y reparaciones necesitadas.

Helicóptero de radiocontrol 3D

Es un helicóptero con paso fijo, tiene un rotor principal totalmente articulado, así como un rotor trasero, que está conectado con el rotor delantero mediante un eje cardan. En la figura 1.15 se muestra el helicóptero que se utilizará en este proyecto.

Fig. 1.15 Falcon 400 [5]

Tiene un motor eléctrico sin escobillas (*brushless*), y la capacidad de hacer vuelo acrobático así como vuelo 3D, que es exactamente lo que los helicópteros reales pueden hacer.

La tabla 1.1 indica especificaciones del mismo.

Tabla 1.1 Falcon 400 [5]	
Diámetro rotor principal:	670 mm
Diámetro rotor trasero:	152 mm
Altura:	210 mm
Peso:	603 g
Motor:	Brushless B20/10T
Servos:	4x 9g servo
Batería:	LiPo 11.1 v 1300mA
Duración vuelo:	13 min

Se empleó este helicóptero por su bajo costo, disponibilidad en piezas de reemplazo, sistema de radio control fácil de operar, capacidad de modificación, mejoramiento, piezas intercambiables, simulador de vuelo virtual usando el mismo radio control, su tamaño nos permite una fácil transportación.

Modelo matemático de un helicóptero

El modelo de un helicóptero es de naturaleza no lineal, variante con el tiempo y con fuerte acoplamiento en algunos bucles [1]. La técnica de identificación frecuencia-dominio, determina las funciones que describen el mejor ajuste lineal de la respuesta del sistema, basado en la primer aproximación armónica de la seria completa de Fourier. Es por eso que existen modelos lineales para el helicóptero que es un sistema dinámico no lineal. La figura 1.16 es un diagrama de cuerpo libre en donde se muestran las variables involucradas en el modelado del helicóptero.

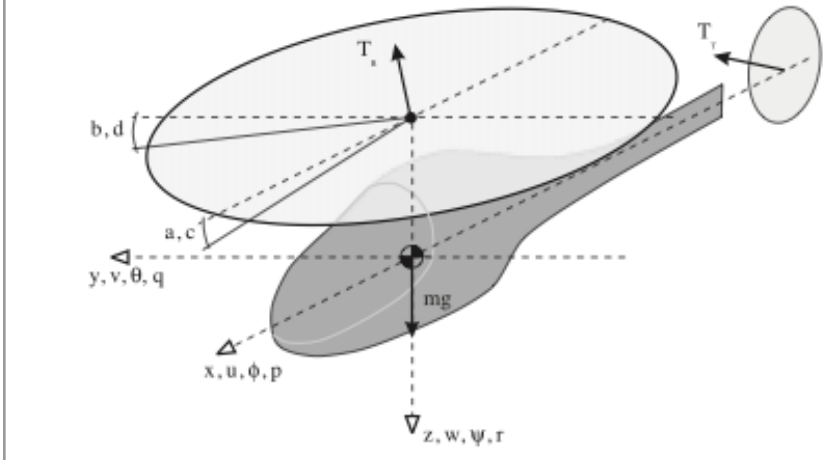


Fig. 1.16 Variables y ejes de coordenadas del modelo matemático

La estructura del modelo especifica el orden y forma de las ecuaciones diferenciales que describen la dinámica del helicóptero. Típicamente, la dinámica del helicóptero es representada como un cuerpo rígido, con una dinámica de fuselaje de seis grados de libertad (6 DoF), que puede estar asociada a dinámicas adicionales, como la del rotor o la dinámica del motor.

La estructura del modelo completo se obtiene mediante la recopilación de todas las ecuaciones en la ecuación diferencial matricial (1.1), que contiene un vector de estados (1.2) y un vector de entradas (1.3).

$$M\dot{x} = Fx + Gu$$

Ecuación 1.1

$$x = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta \end{bmatrix}^T$$

Ecuación 1.2

$$u = \begin{bmatrix} \delta_{lon} & \delta_{lat} & \delta_{ped} & \delta_{col} \end{bmatrix}^T$$

Ecuación 1.3

Donde:

u , v , w : Velocidades en las coordenadas del fuselaje

p , q , r : Velocidades angulares de cabeceo, alabeo y guiñada

θ, ϕ: Angulos Euler

δ (lat, lon, ped, col): Entrada del control lateral, longitudinal, pedal y colectivo

La matriz *F* (1.4) contiene los derivados de la estabilidad, la matriz de entrada *G* (1.4) contiene los derivados de entrada, y la matriz *M* las constantes del rotor. El sistema completo de espacio de estados se muestra en la ecuación 1.5.

$$F = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & -g & X_{a_{1s}} & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & g & 0 & 0 & Y_{b_{1s}} & 0 & 0 & 0 \\ L_u & L_v & 0 & 0 & 0 & 0 & L_{a_{1s}} & L_{b_{1s}} & 0 & 0 & 0 \\ M_u & M_v & 0 & 0 & 0 & 0 & M_{a_{1s}} & M_{b_{1s}} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1/\tau_f & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & B_{a_{1s}} & -1/\tau_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z_{a_{1s}} & Z_{b_{1s}} & Z_w & Z_r & 0 \\ 0 & 0 & N_p & 0 & 0 & 0 & 0 & 0 & N_w & N_r & MN_{ped} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_r & MK_{rfb} \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A_{lat} & A_{lon} & 0 & 0 \\ B_{lat} & B_{lon} & 0 & 0 \\ 0 & 0 & 0 & Z_{col} \\ 0 & 0 & N_{ped} & N_{col} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ecuación 1.4 [6]

Para la identificación de los parámetros se utiliza un método de frecuencia-dominio llamado CIFER [7] (Comprehensive Identification method based on FrEquency Response), que fue desarrollado por la NASA y el ejercito de los Estados Unidos de America, es un software que tiene un costo aproximado de $50,000 dólares [7], dados los alcances de esta tesis, este tipo de software seria prohibitivo.

Con el objetivo de esta tesis, se puede eliminar el paso del modelado del helicóptero, por ende todo este tipo de conocimiento no es crucial en el desempeño del control.

2.- Instrumentación

Introducción

Para poder aplicar control a un helicóptero es necesario conocer variables físicas (orientación espacial) que nos indiquen en que estado se encuentra y tener una forma para poder manipularlo.

Un instrumento es un dispositivo que mide o manipula variables en un proceso, en nuestro caso, las variables instrumentadas son la orientación espacial del helicóptero y los pulsos enviados por el radiocontrol que manipulan a los servos los cuales cambian la orientación del helicóptero.

La instrumentación se aplica al sensado y procesamiento de la información proveniente de las variables físicas, a partir de las cuales realiza un monitoreo y control de procesos, empleando dispositivos y tecnologías electrónicas.

Se compone de varias etapas, la de sensado, la de acondicionamiento, filtrado y de digitalización.

Se considerara al "helicóptero instrumentado" cuando podamos leer la orientación espacial del mismo, así como cuando se puedan controlar los servomotores de los actuadores.

Sensado

Se seleccionan que variables físicas se van a medir y se definen los parámetros de cada una, como el rango, sensibilidad, resolución, la exactitud y precisión de las medidas. La linealidad y grado de error se evaluaron de forma empírica. En esta etapa se analiza que es lo que se entiende por orientación espacial y las diferentes maneras de medirlo.

Orientación Espacial

La orientación de un objeto en el espacio es cada una de las posibles elecciones para colocarlo sin cambiar un punto fijo de referencia. Puesto que el objeto con un punto fijo puede ser todavía rotado alrededor de ese punto fijo, la posición del punto de referencia no especifica por completo la posición, por tanto para especificar completamente la posición necesitamos especificar también la orientación, esta puede visualizarse añadiendo una base vectorial ortogonal al punto de referencia del objeto. También se lo conoce como actitud (Attitude).

Angulos de Euler

Son tres coordenadas angulares que permiten relacionar la orientación de un sistema de ejes respecto a otro. Estos proporcionan tres coordenadas generalizadas adecuadas para describir el movimiento de sólidos rígidos. En la figura 2.1 vemos que el sistema de referencia en rojo esta referido al sistema de referencia azul mediante estos ángulos.

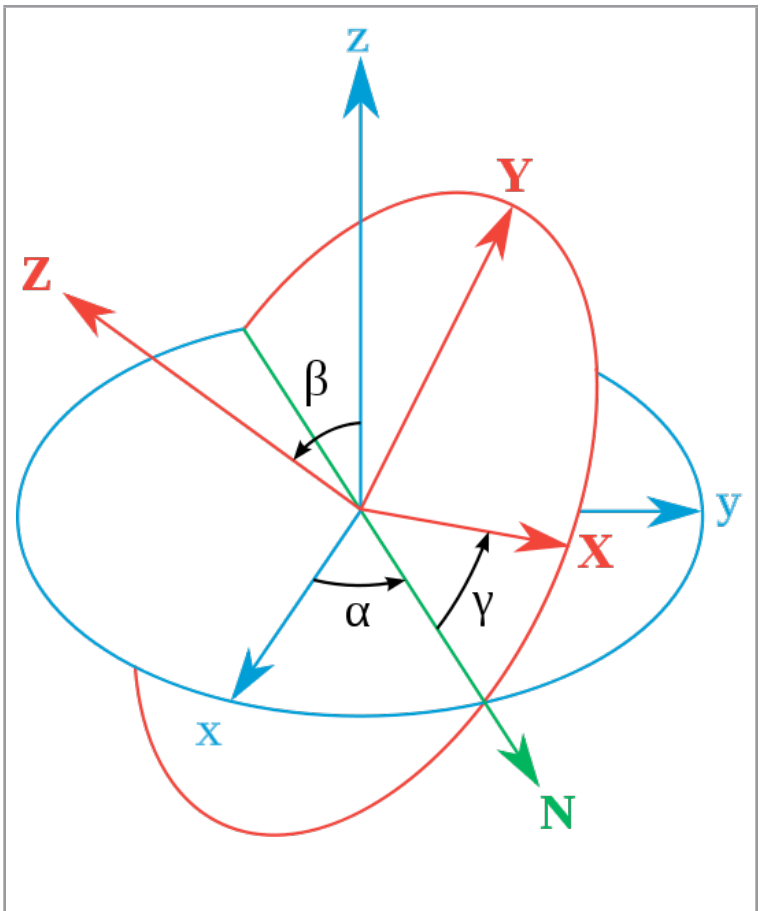


Fig. 2.1 Angulos Euler [1]

El orden de aplicación de los giros sobre los ejes coordenados afecta la configuración final del sistema de referencia. Como ejemplo tenemos la figura 2.2 donde existen tres ángulos de giro llamados *psi*, *theta* y *phi* aplicados a los ejes coordenados ilustrados en azul. En la primera imagen el orden de aplicación es girar sobre el eje *Z psi* radianes desde el eje *X*, luego aplicar el giro *theta* sobre el vector N a partir de *y'* y por último, hacer el giro *phi* sobre el eje Y del sistema de referencia en rojo.

En el segunda imagen el orden de los giros es realizado primero en el eje *z* del sistema de referencia azul a partir del eje *y* luego se gira *theta* radianes sobre el eje *X* del sistema de referencia en rojo a partir de *y'*, por último se realiza el giro *phi* sobre el eje *Y* del sistema de referencia en rojo a partir del vector N.

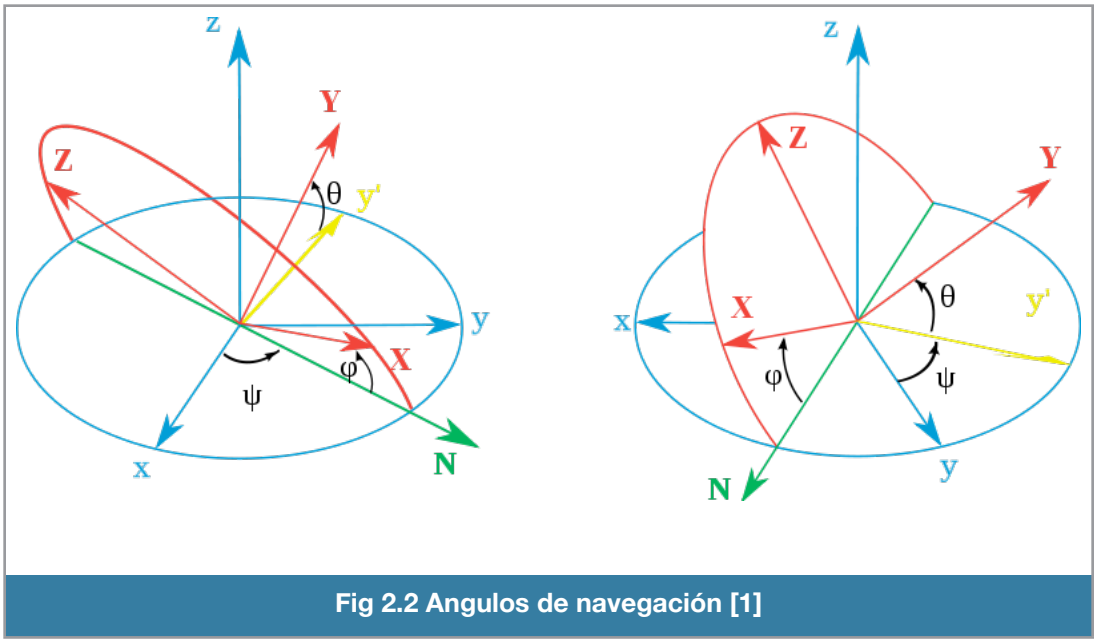


Fig 2.2 Angulos de navegación [1]

Angulos de navegación

Son un tipo de ángulos de Euler usados para describir la orientación de un objeto en tres dimensiones. También se les conoce como ángulos de Tait-Bryan, son tres coordenadas angulares que definen un triedro rotado desde otro que se considera el sistema de referencia. Se muestran en la figura 2.3.

Dado un sistema de tres ejes fijos en la aeronave, se llaman guiñada (yaw), cabeceo (pitch) y alabeo (roll).

Pitch: Es una inclinación del morro del avión, o rotación respecto al eje ala-ala. (ψ).

Roll: Rotación intrínseca alrededor del eje morro-cola del avión. (θ).

Guiñada: Rotación respecto de un eje vertical. (φ).

Maneras de medir los ángulos de navegación

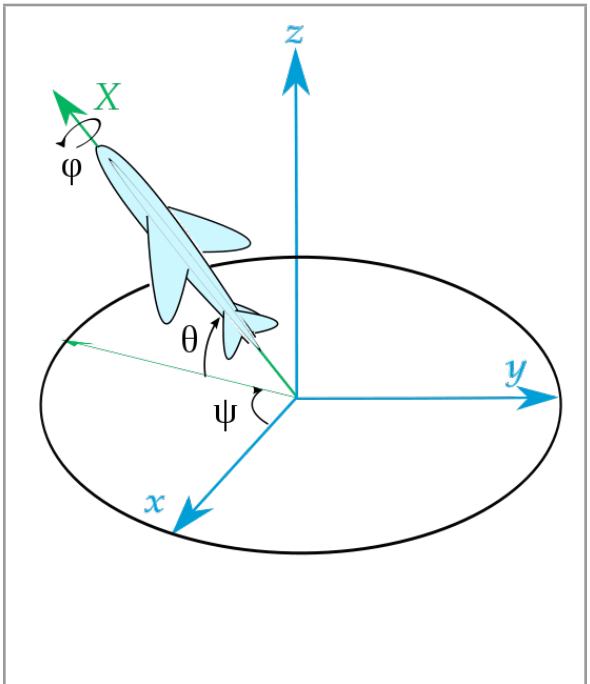


Fig. 2.3 Angulos Euler en una aeronave [1]

Existen varias maneras de medir los ángulos de navegación en una aeronave. (figura 2.3), que serán los que nos permitan controlar la estabilidad del helicóptero, una de ellas es usar termopilas, que son dispositivos electrónicos que convierten energía térmica en energía eléctrica, mediante el efecto termoeléctrico, se requieren dos termopilas por eje de rotación para poder detectar el ángulo de navegación en dicho eje. El problema de las termopilas es que deben de ser capaces de ver el horizonte, para hacer las comparaciones, eso limita de gran manera los lugares en donde se pueda pilotar el helicóptero.

Otra manera de medir la posición relativa del helicóptero es usar una cámara de video ccd para comparar cada fotograma tomado por la cámara y así poder saber para donde se movió el helicóptero. Esta es una manera muy usada en muchos sistemas de estabilización para helicópteros, la desventaja de este sistema es que requiere una gran capacidad de cómputo para el análisis de las imágenes y está limitado ambientalmente si la cámara se encuentra expuesta a un ambiente con neblina o polvo ya que el sistema no podrá hacer las comparaciones entre imágenes, resultando en una estabilización no correcta del helicóptero.

La manera mas antigua utilizada para medir ángulos de navegación es mediante el uso de un giroscopio. Pero estos sistemas sufren de un efecto llamado deriva, este efecto hace que sobre el tiempo, el valor del giroscopio cuando ha estado en una posición estable, se cambie o se mueva del valor inicial tomado.

Los acelerómetros son otra manera de poder medir los ángulos de navegación de una aeronave, el gran problema de los acelerómetros es que son muy susceptibles a la vibración, provocando errores en la medición y por ende en los ángulos de navegación.

Pero si se combinan estos dos últimos sistemas, los giroscopios y acelerómetros, y se utilizan técnicas de filtrado y fusión, se pueden solventar los problemas que tienen ambos. La medida dada

por el acelerómetro ayuda a eliminar drásticamente el problema de la deriva del giroscopio, de esta manera obteniendo unos ángulos de navegación que sean bastante confiables. La desventaja de esta unión de sensores es que requieren un sistema de filtrado tipo Kalman o DCM, de los que discutiremos más adelante.

En la figura 2.4 se describen los ángulos que indican la orientación espacial de la nave.

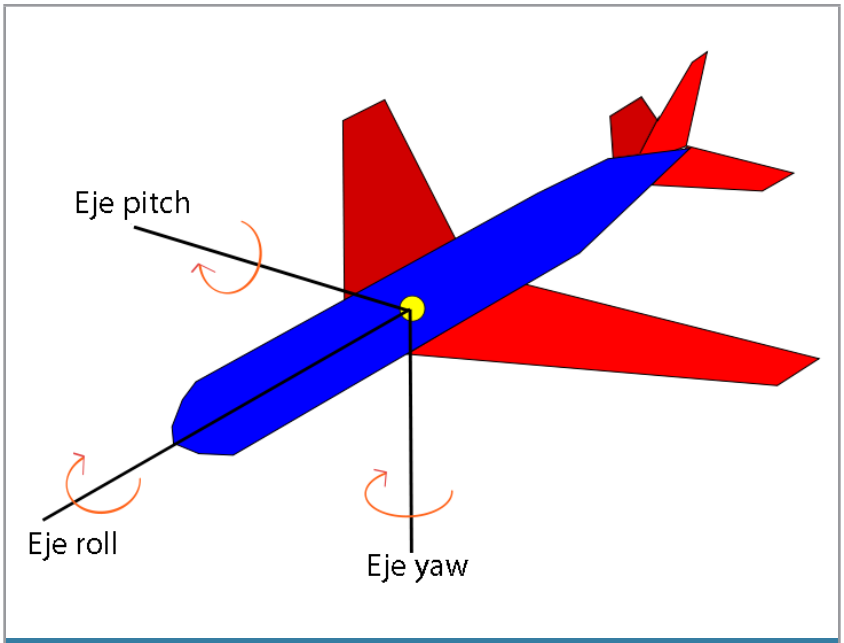


Fig. 2.4 Eje de guiñada, cabeceo y alabeo de una aeronave [1]

Giroscopio

Es un dispositivo capaz de medir la orientación, basado en los principios de conservación del momento angular. Cuando se somete el giroscopio a un momento de fuerza que tiende a cambiar de dirección como lo haría un cuerpo que no girase, cambia la orientación en una dirección perpendicular a la dirección intuitiva.

Son usados en aplicaciones muy críticas como en aviones. Existen opciones que son de bajo costo y de tamaño pequeño, estos relativamente nuevo tipo de giroscopios son fabricados usando MEMS (Sistemas MicroElectroMecanicos).

Uno de estos nuevos giroscopios tienen un rango hasta +-300 grados por segundo, cuentan con una mayor sensibilidad para captar los desplazamientos angulares rápidos, además de que incorporan un acondicionamiento de señal y aumentan la resistencia al choque y la vibración. Por consiguiente puede operar en los entornos más demandados por aplicaciones industriales y de consumo como la navegación de vehículos.

Los giroscopios MEMS utilizan el efecto Coriolis (Existencia de una aceleración relativa del cuerpo en dicho sistema de rotación) creados por el movimiento.

La figura 2.5 muestra que cuando una masa resonante se mueve sobre el eje exterior de rotación, esta es acelerada a la derecha y la masa se acerca al centro de rotación, eso precisamente es lo que se mide para calcular la velocidad angular.

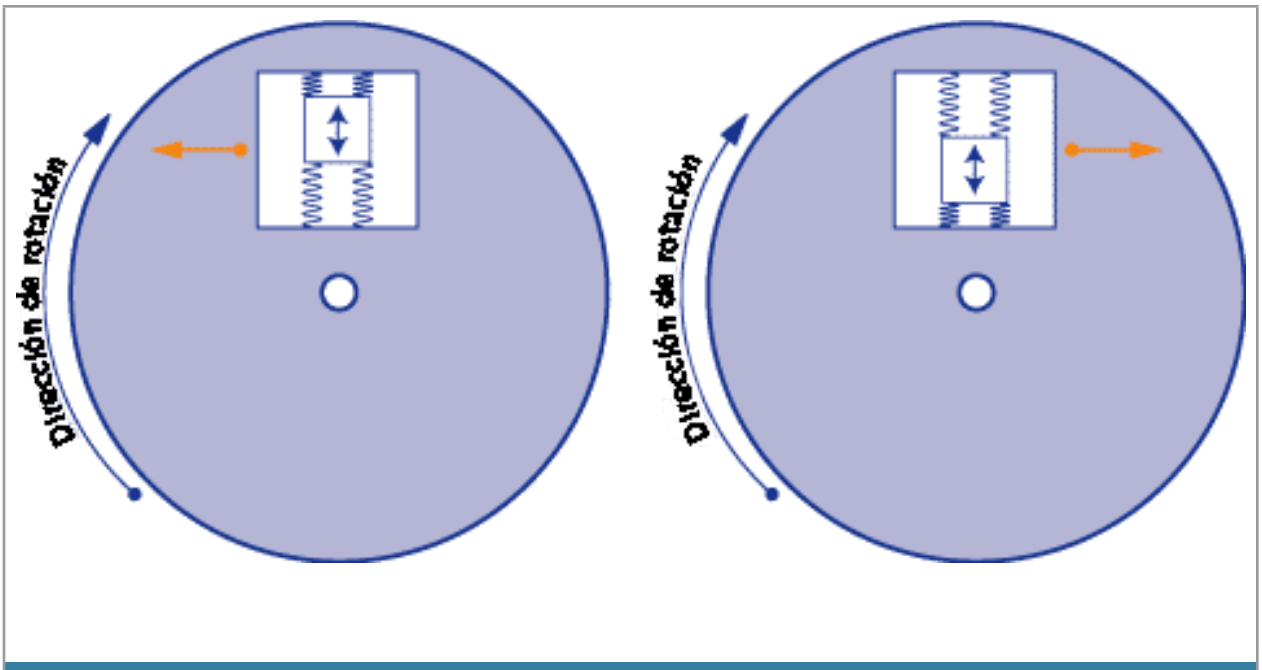


Fig. 2.5 Movimiento de la masa en un giroscopio MEMS [2]

La figura 2.6 muestra un diagrama de la estructura mecánica de un giroscopio con tecnología MEMS.

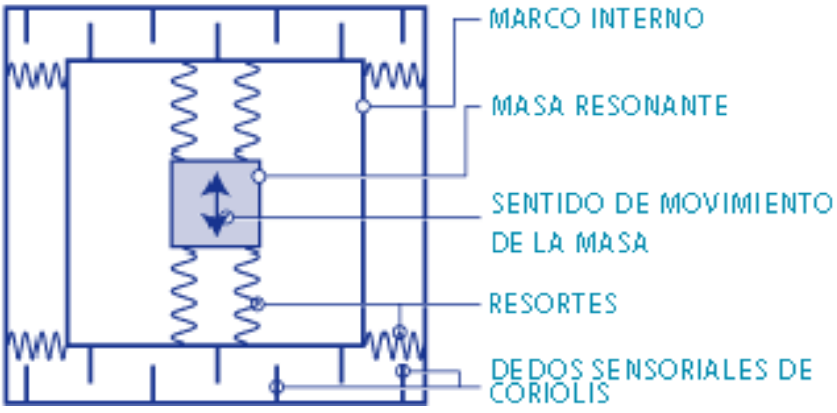


Fig. 2.6 Esquemático de la estructura mecánica de un giroscopio MEMS [2]

Es de notarse que el giroscopio se puede colocar en cualquier parte del objeto que rota y a cualquier ángulo.

Lo que el giroscopio nos entrega es la medida de la velocidad angular, si se integra ese valor, se obtiene el ángulo de navegación.

En la figura 2.7 se puede apreciar la salida (línea azul) de un giroscopio, haciendo una prueba de movimiento al sensor, medida por un ADC (Convertidor Analógico a Digital) de 10-bits de resolución proveniente del microcontrolador Arduino, de este modo nuestro rango de la señal medida está entre valores de 0 a 1024 bits. Cada bit corresponde a 3.22mV.

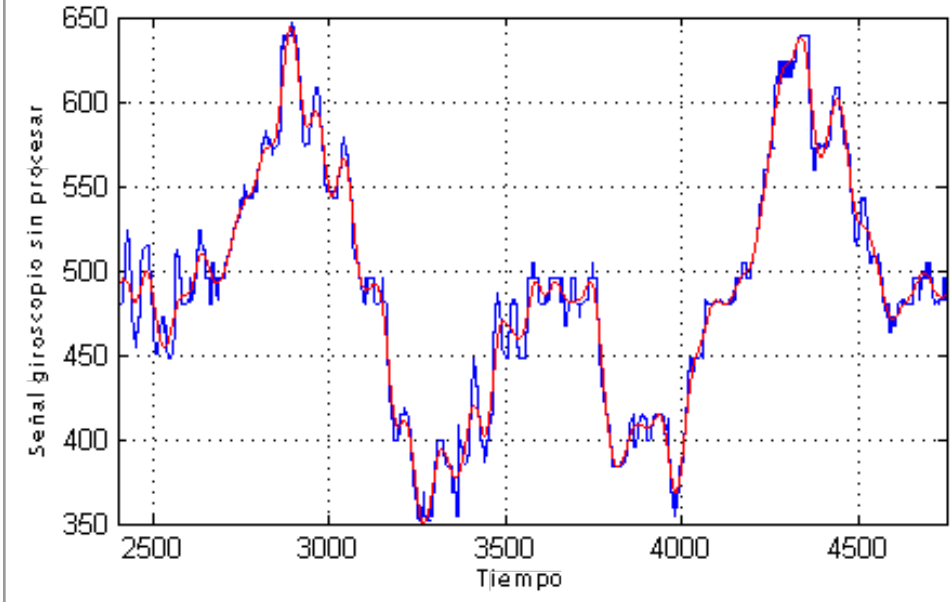


Fig. 2.7 Señal de salida del giroscopio

La línea roja de la figura es solamente una versión que pasó a través de un filtro digital paso bajas, con una frecuencia de corte en 0.5 hz, para atenuar la señal. Ahora se procede a normalizar las mediciones para asegurarnos de que los valores negativos correspondan a una velocidad angular negativa, para esto se resta aproximadamente 490 a los valores de la figura 2.8.

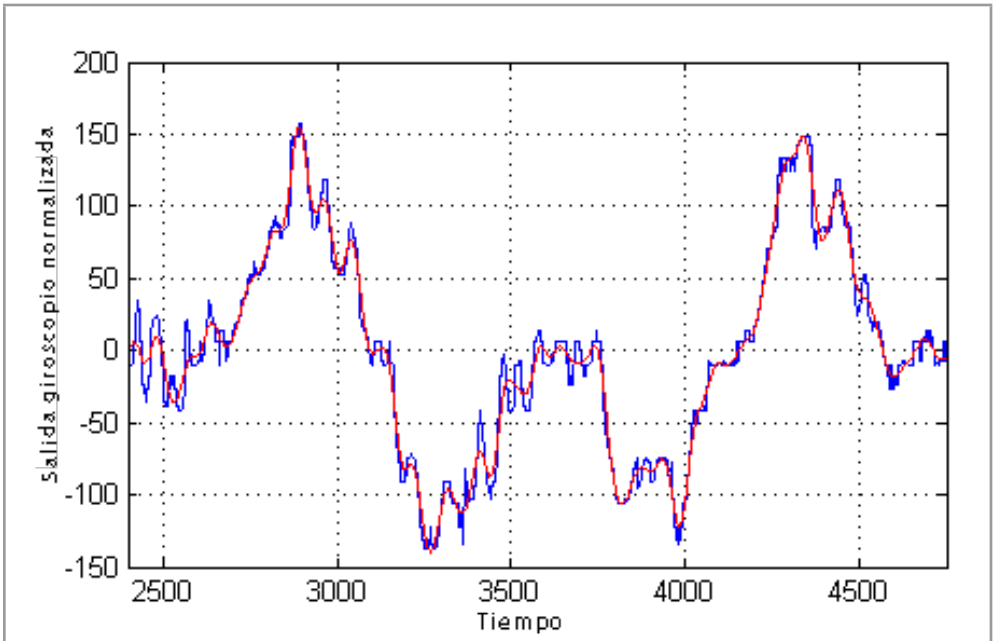


Fig. 2.8 Salida normalizada del giroscopio

Ahora, para obtener una medición angular se realiza una integración discreta (sumar todos los valores), se realiza esa operación y se multiplica por un escalar para obtener resultados en grados, se aprecia el resultado en la figura 2.9.

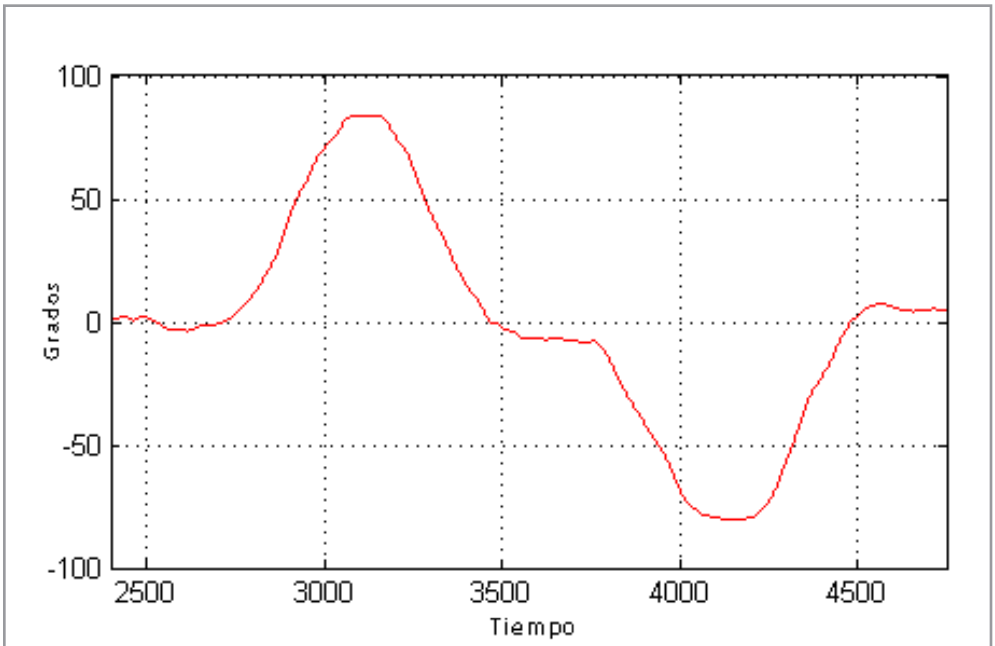


Fig. 2.9 Angulo obtenido por el giroscopio

Por el efecto de deriva el valor que el giroscopio tenía en una posición estable, se deriva o se mueve, con esto perdiendo toda la referencia y por ende, una lectura de ángulo equivocada.

En la figura 2.10 se muestra el efecto de la deriva.

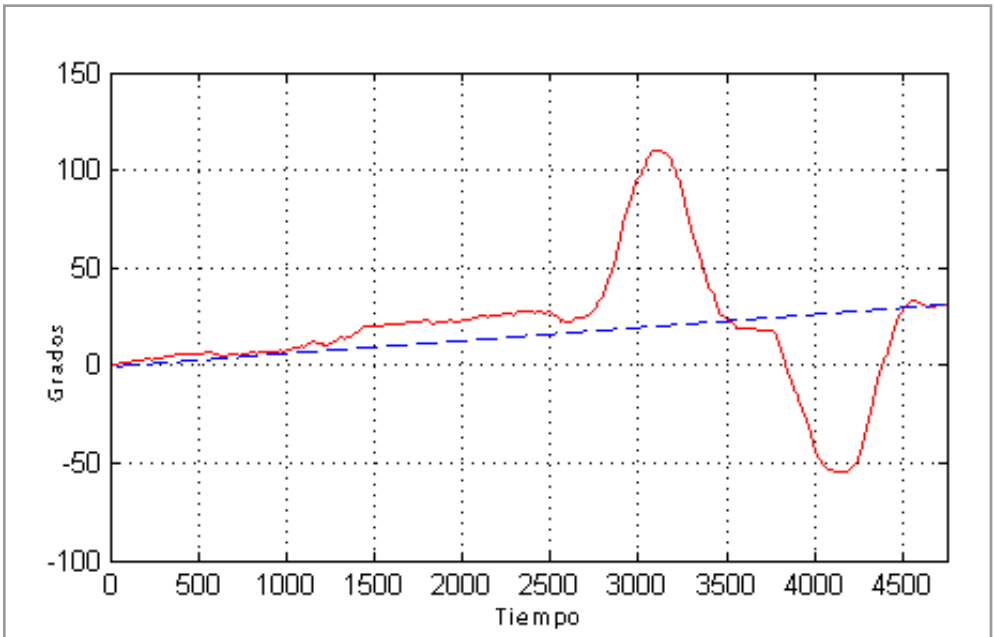


Fig. 2.10 Efecto drift afectando la medida del angulo

Durante 4500 muestras el sensor se movió o perdió 30 grados de precisión.

El efecto de deriva no sucede en los acelerómetros, ellos son la clave para poder obtener un sistema de medición de los ángulos de navegación precisos.

Acelerómetro

Es un dispositivo que mide aceleración, la aceleración que es relativa a la caída libre, es la misma que experimentan las personas y objetos. Como consecuencia de esto, un acelerómetro en reposo sobre la superficie de la tierra, leerá aproximadamente 1g en dirección radial al centro de la tierra. Para encontrar la aceleración debida al movimiento respecto a la tierra se debe de substraer el offset de la gravedad.

Un acelerómetro arroja un valor de medición nulo durante caída libre. Para usos prácticos con el propósito de encontrar la aceleración de objetos respecto de la tierra, se necesita saber el valor de la gravedad local. Esto se hace calibrando el dispositivo en reposo.

Conceptualmente un acelerómetro se comporta como una masa amortiguada sobre un resorte. Cuando experimenta aceleración, la masa es desplazada a un punto el cual el resorte es capaz de acelerar la masa a la misma velocidad. El desplazamiento que se mide es proporcional a la aceleración en ese eje de medición.

Los acelerómetros modernos son construidos de MEMS y de hecho es el sistema MEMS más sencillo que se puede construir. Estos consisten de una masa sísmica y un cantilever. La amortiguación resulta del gas residual que se encuentra sellado en el dispositivo. Bajo la influencia de aceleraciones externas la masa sísmica se mueve de su posición neutral. La deflexión de la masa es medida de manera análoga o digital.

Muchos acelerómetros MEMS funcionan solo en su plano de trabajo, esto implica que hay que orientar el sensor para poder medir el plano deseado. Los MEMS mas modernos integran las mediciones de tres planos en un solo CI (Circuito Integrado). De esta manera solo se tiene que referenciar la posición del CI para saber que plano es cada cual.

Los acelerómetros MEMS están disponibles en una gran variedad de rangos de medición, ya sea con la capacidad de captar magnitudes del orden de miles de g's o de tener una sensibilidad que pueda sensar el mínimo cambio en la magnitud. El diseñador debe hacer un compromiso entre sensibilidad y la máxima aceleración que puede ser medida.

Un acelerómetro por si solo no es apropiado para determinar cambios en orientación cuando la vertical de gravedad es significante, como sucede en una aeronave o cohetes. En la presencia de un gradiente gravitacional, la calibración y el proceso de reducción de datos es numéricamente inestable.

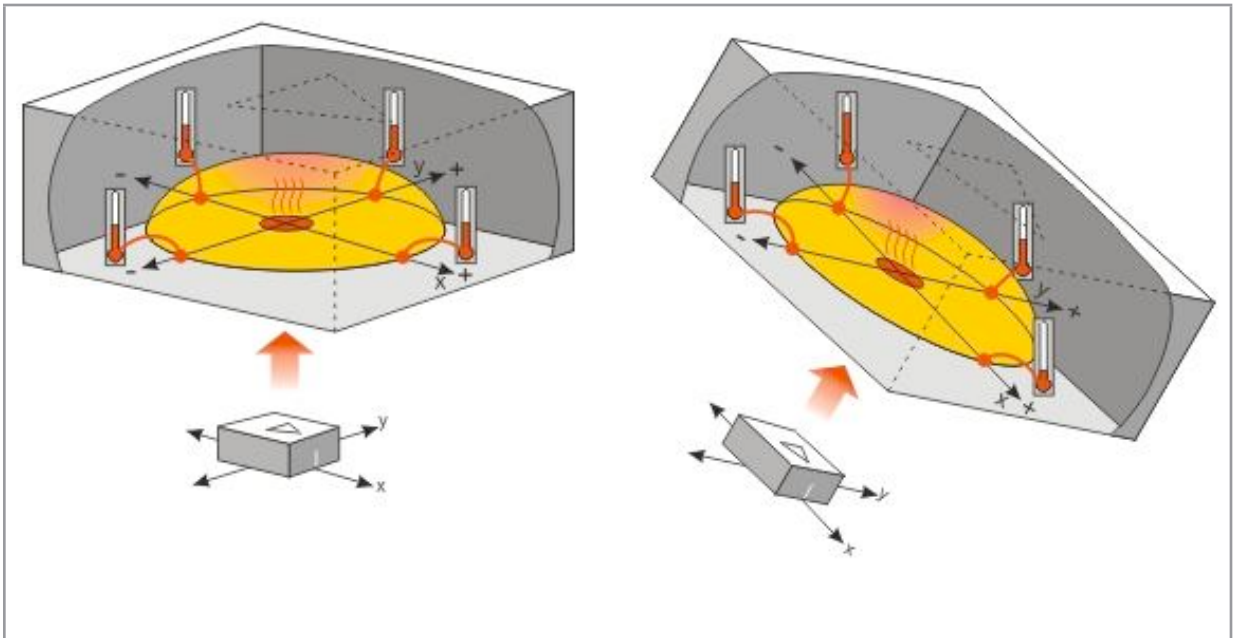


Fig. 2.11 Burbuja de gas dentro del acelerómetro MEMS [2]

Otro tipo de MEMS para medir la aceleración está formado de paredes paralelas que registra el movimiento entre cada una de ellas, dependiendo de la vibración o aceleración aplicada al dispositivo se obtiene la lectura. Estos sensores son muy delicados, susceptibles a degradación por polvo, agua y a cualquier contacto físico. Tales degradaciones se atenúan cuando el dispositivo está encapsulado.

Un acelerómetro puede ser usado para determinar la actitud de una aeronave, en especial los ángulos de navegación pitch y roll.

En la figura 2.12 se aprecia la gravedad en la flecha roja, y la flecha azul es la aceleración que está sensando el dispositivo.

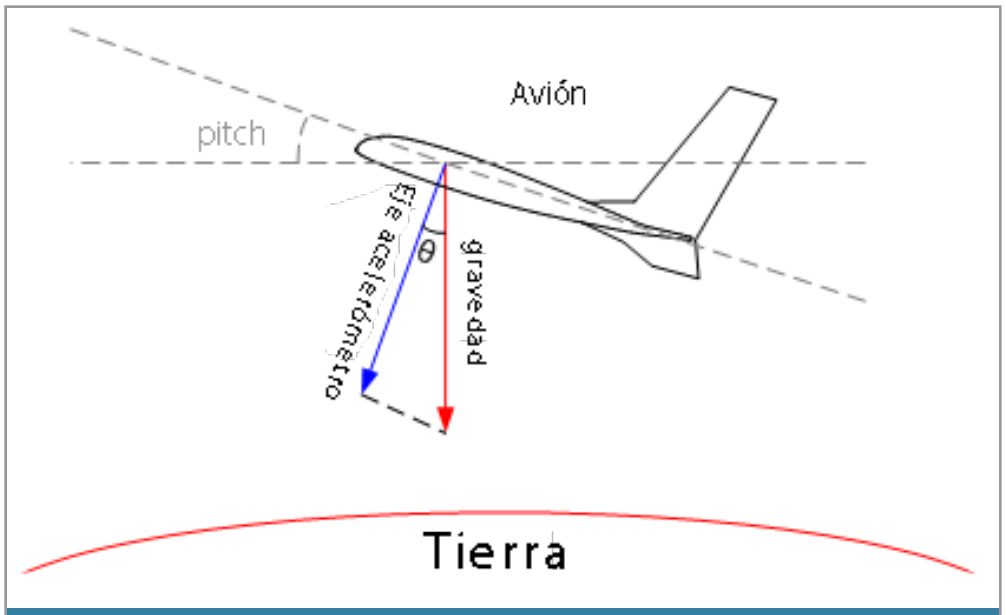


Fig. 2.12 Demostración de lectura de un acelerómetro

El ángulo theta entre el vector gravedad y el medido por el sensor, está relacionado con el ángulo de navegación pitch, se puede calcular de la siguiente manera:

Ángulo Pitch = Θ + 90ª

Se conoce la gravedad, entonces se puede calcular el ángulo theta:

*Acelerómetro = cos (Θ) * gravedad*

Entonces:

Θ = acos (acelerómetro / gravedad)

Para calcular el ángulo roll, se hace básicamente lo mismo, solamente se necesita un acelerómetro extra con un eje perpendicular al acelerómetro del pitch.

Al tener la función coseno dos soluciones posibles para un mismo valor en el intervalo [-pi pi], es necesario determinar cuál de ellas es la correcta de acuerdo con el marco de referencia.

El problema del coseno inverso, es que no nos puede dar los 360 grados del ángulo de navegación pitch, porque un plano que esta apuntando al cielo, y otro apuntando al suelo, nos daría una medición de cero.

Una solución puede ser el uso de otro acelerómetro de referencia para distinguir estos casos. La otra solución es usar la tangente inversa de dos argumentos la cual nos dará el ángulo resultante dentro del cuadrante correcto:

Pitch = atan2 (acelerómetro / gravedad , z / gravedad)

Por ende, para poder calcular de manera correcta los ángulos de navegación pitch y roll, se necesitan 3 acelerómetros, 2 para cada plano de navegación y el acelerómetro Z para tenerlo de referencia para cada plano.

Existe un problema con los acelerómetros relacionado a la medición de ángulos. Cuando el sensor es desplazado experimenta una aceleración debida al movimiento inercial. Esta aceleración crea componentes sobre los ejes de referencia del acelerómetro haciendo que el cálculo se vuelva inválido. Por ejemplo, en una trayectoria circular, la aceleración centrípeta actúa en el sensor haciendo que estos pierdan la referencia y por ende una lectura incorrecta de orientación además de los problemas de vibración. Estos problemas se resuelven en conjunto con los giroscopios, usando los datos provenientes de los giroscopios y de los acelerómetros.

Unidad de medición inercial (IMU)

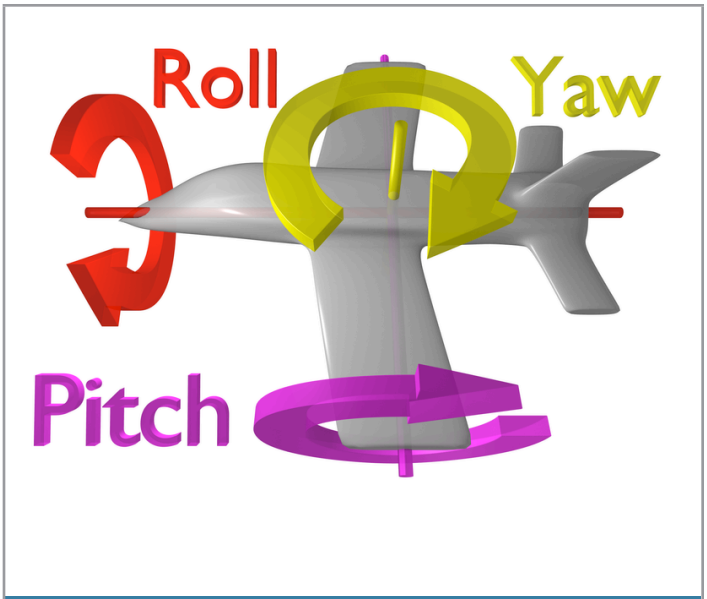


Fig. 2.13 Orientación espacial [1]

Los giroscopios y acelerómetros en conjunto son la tecnología adecuada para determinar la orientación de una aeronave (figura 2.13). Soportan mejor las vibraciones y las condiciones ambientales adversas y en conjunto son más precisos que cualquier otro dispositivo.

Las unidades de medición inercial son dispositivos electrónicos que miden la velocidad angular y la aceleración que experimenta la aeronave, usando una combinación de acelerómetros y giroscopios.

Básicamente las IMU's son sistemas que constan de diversos componentes eléctricos y electrónicos montados en un circuito impreso (PCB). Los acelerómetros y giroscopios son acondicionados con los demás componentes eléctricos; para que funcionen correctamente y entreguen al sistema central las medidas analógicas de cada uno de ellos.

Fig. 2.14 IMU 6DOF v4 [3]

Los hay de diferentes configuraciones, desde 3 Grados de libertad (DOF) hasta 9 DOF. El más común es el de 6 DOF que consta tres acelerómetros y tres giroscopios (cada uno para un eje) lo que hace que se puedan obtener las mediciones exactas de los ángulos de navegación *pitch* y *roll*; además de poder tener una medición no referenciada del ángulo *yaw*.

Para tener una medición correcta del *yaw* se requiere una referencia en el plano ortogonal al eje Z; lo cual solo lo puede ofrecer un magnetómetro. Este sensor mide la intensidad del campo magnético en tres ejes ortogonales, dos de estas mediciones se usan como referencia para corregir la velocidad angular del giroscopio *yaw*.

Las IMU's son solamente un subsistema del sistema de navegación. Otros sistemas ya sean de hardware o software son necesarios para corregir las inexactitudes que inevitablemente se presentan.

Las primeras IMU's eran grandes y estorbosas ya que los acelerómetros y giroscopios tenían que estar posicionados respecto al plano o eje que medían. Se presentan ejemplos en las figuras 2.14 y 2.15.

Las más modernas IMU's son planas, porque los nuevos sensores MEMS no necesitan estar en el plano que miden. Algunos ejemplos se muestran en las figuras 2.16 y 2.17.

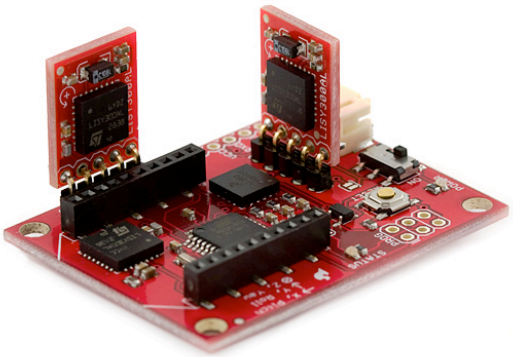


Fig. 2.15 Atomic IMU 6DOF [3]

Recientemente desarrollaron una IMU que, además de incorporar acelerómetros y giroscopios, también acomodan magnetómetros; esto lo convierte en 9DOF, ya que puede mantener referencia magnética en los 3 ejes, agregando de esta forma 3 grados más de libertad, además de construirse con componentes que lo hacen plano.

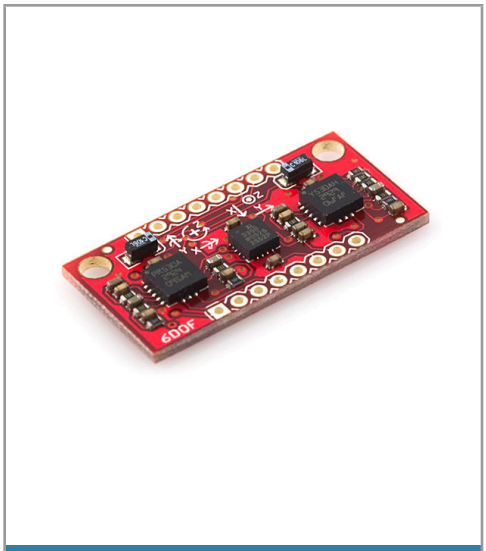


Fig. 2.16 IMU 6DOF Razor [3]

Estos dispositivos por si solos son relativamente económicos, el más completo se muestra en la figura 2.17 (9DOF), cuesta alrededor de 150 dólares; lo que hace expensivo a un sistema de referencia es el software de filtrado y acondicionamiento de la señal. Se le conoce de 9DOF porque incluye tres acelerómetros, tres giroscopios y tres magnetómetros; los últimos para poder tener una referencia magnética con respecto a los polos magnéticos.

Sistema de referencia de actitud y rumbo

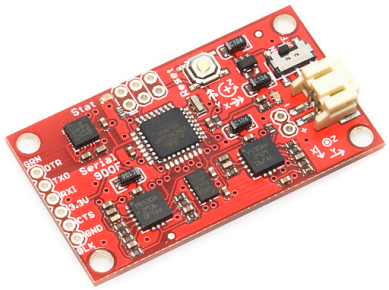


Fig. 2.17 IMU 9DOF Razor [3]

Attitude Heading Reference System (AHRS) es un sistema que provee orientación y dirección de una aeronave. Están diseñados para reemplazar los tradicionales instrumentos de vuelo giroscopios. Incluyen una IMU y una computadora que es capaz de procesar toda la información proveniente de la IMU y generar los ángulos de navegación *pitch*, *roll* y *yaw*.

Los AHRS han demostrado ser bastante óptimos y confiables para aeronaves de uso comercial o negocios.

Están típicamente integrados con los Sistemas de Información Electrónica de Vuelo (*EFIS*), a los que reportan la orientación y dirección de la aeronave.

Selección del sensor

En el desarrollo de la instrumentación del helicóptero falcon 400 se emplea la *IMU 6DOF Ultra Thin* debido a su tamaño reducido, bajo costo, a que sus dispositivos MEMS tienen una tolerancia media respecto al promedio para soportar vibraciones y al bajo consumo de energía.

Unidad de medición inercial

Esta unidad IMU (Inertial Measurement Unit) no cuenta con regulación de voltaje, por lo que se tiene que alimentar con un voltaje estable de 3.3 volts, contiene capacitores de filtrado, un filtro paso bajas, y otro paso altas. Se muestra el sensor en proporción en la figura 2.18.

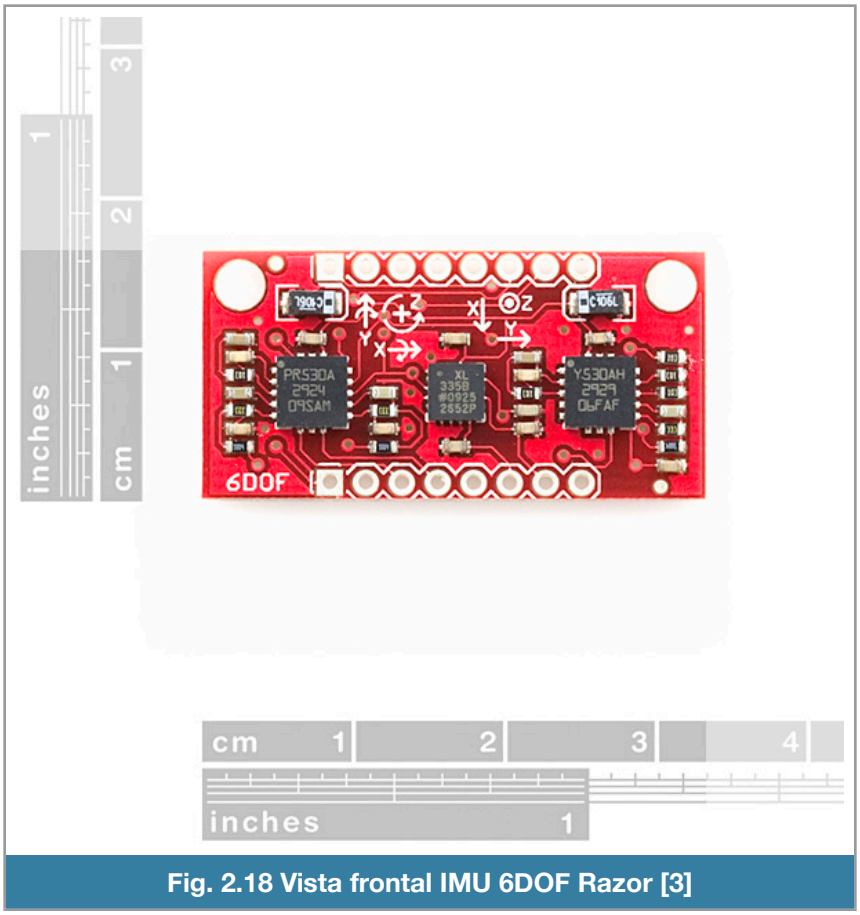


Fig. 2.18 Vista frontal IMU 6DOF Razor [3]

Acelerómetro ADXL335

El ADXL335 es un acelerómetro MEMS de 3 ejes, de bajo consumo de energía, tamaño reducido (4mm x 4mm x 1.45mm).

Este producto mide aceleración con un rango mínimo de escala total de +- 3g. Puede medir aceleración estática de la gravedad en aplicaciones de inclinación, o aceleración dinámica proveniente de movimiento, vibración o golpes. Nos entrega información de aceleración en cada uno de los ejes de medición (x , y , z).

Contiene un sensor de polisilicio micromaquinado (estructura de wafer-on-wafer) además de circuitería de acondicionamiento de señal para implementarlo en una arquitectura de medición de

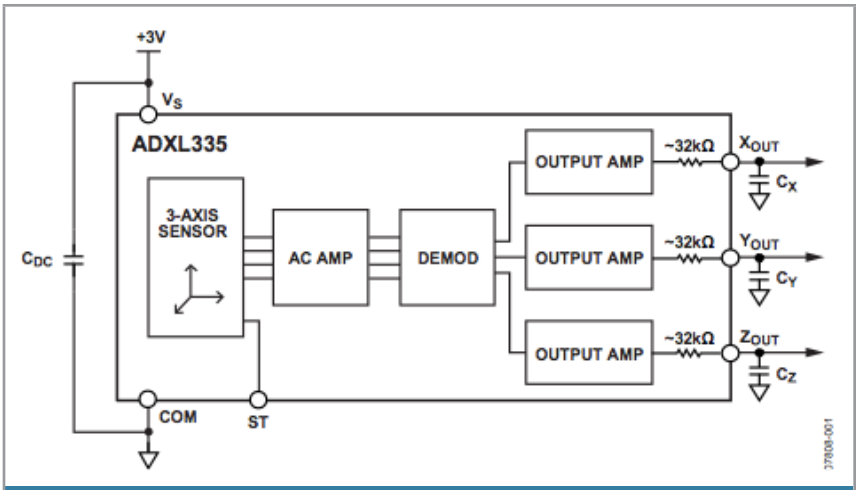


Fig. 2.19 Diagrama de funcionamiento [2]

aceleración de lazo-abierto. Las señales de salida son voltajes analógicos que son proporcionales a la aceleración. Tal descripción se aprecia en la figura 2.19.

En la figura 2.20 se indican los ejes de referencia del acelerómetro con respecto a la orientación espacial del encapsulado. En la figura 2.21 se muestran ejemplos de diferentes posiciones del encapsulado y los valores obtenidos como resultado de la medición de la aceleración gravitatoria.

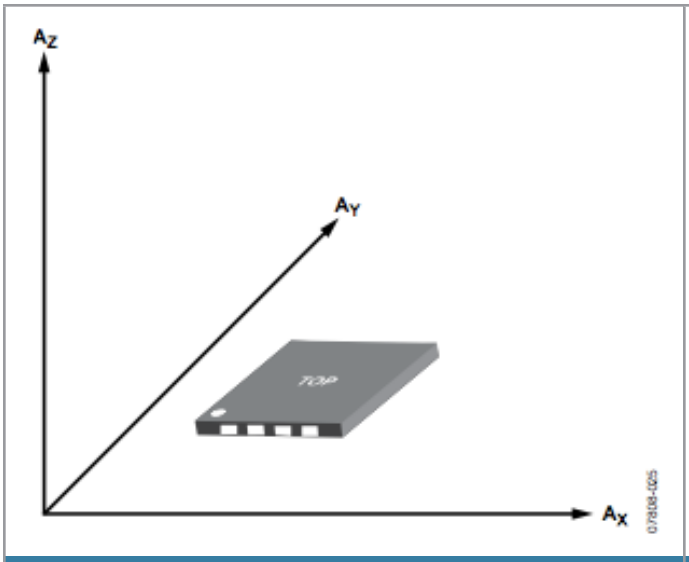


Fig. 2.20 Ejes de aceleración [2]

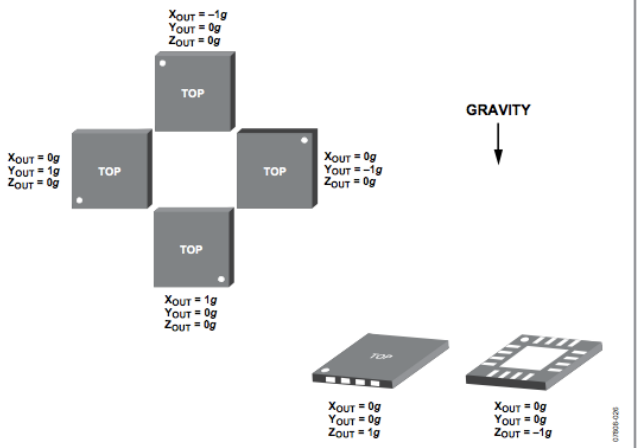


Fig. 2.21 Respuesta VS Orientación [2]

Giroscopio LPR530AL - LY530AHL

Estos dispositivos MEMS son giroscopios cuya función es detectar la razón de cambio en los ejes *Roll*, *Pitch* y *Yaw*. Son la combinación de un actuador y un acelerómetro en una estructura micromaquinada.

Incluye un elemento sensor compuesto de una masa movible, que se mantiene en movimiento de oscilación continuo y es capaz de reaccionar cuando una razón de cambio angular es aplicada con el principio de *Coriolis*.

Los giroscopios producen voltajes analógicos proporcionales a la razón del cambio angular.

Si la rotación es en sentido inverso a las manecillas del reloj, se producirá un voltaje en sentido positivo, incrementándose respecto al voltaje en posición estable.

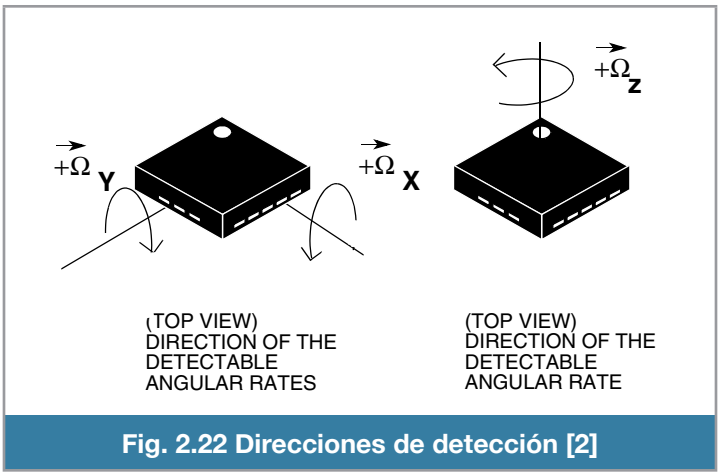


Fig. 2.22 Direcciones de detección [2]

Si la rotación es en sentido a las manecillas del reloj, se producirá un voltaje en sentido negativo, decrementándose respecto al voltaje en posición estable.

En la figura 2.24 se aprecia el sentido positivo de la rotación con respecto a los ejes de referencia del encapsulado.

Provee una excelente estabilidad en diferentes rangos de temperaturas, siendo más eficiente en el rango de -40 ºC a 85 ºC.

En los diagramas de funcionamiento (figura 2.22 y 2.23) se aprecian los dispositivos involucrados en la medición de la razón angular para cada eje X, Y y a Z. Ambos giroscopios integran filtros paso-bajas, consumen poca energía y tienen alta supervivencia a impactos y vibración.

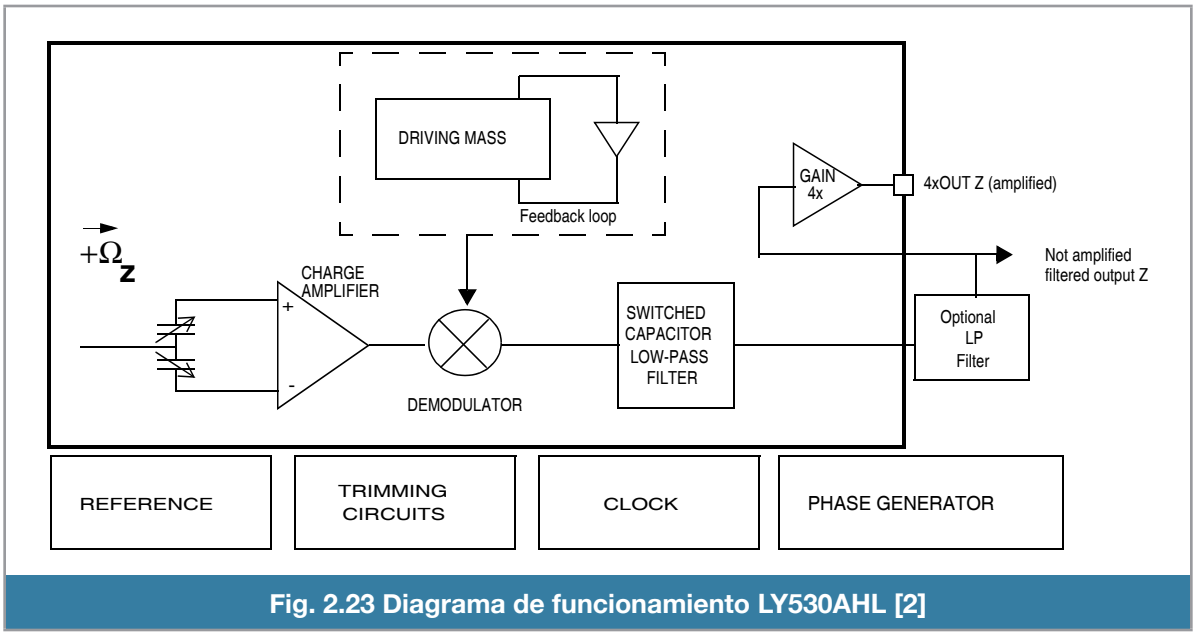


Fig. 2.23 Diagrama de funcionamiento LY530AHL [2]

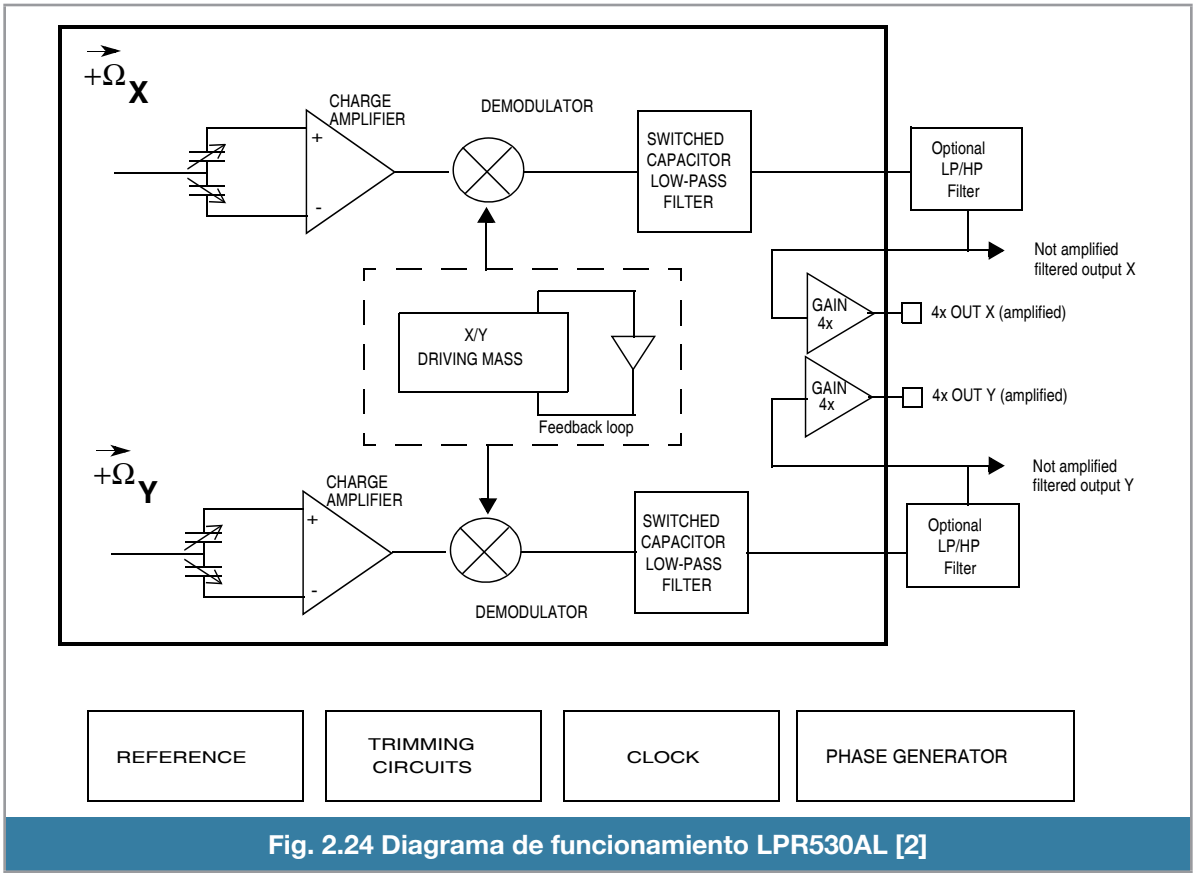


Fig. 2.24 Diagrama de funcionamiento LPR530AL [2]

Acondicionamiento

La señal de un sensor suele ser validada para su procesado. Por lo general requiere de una amplificación para adaptar sus niveles a los del resto de la circuitería.

También se tiene que aislar y filtrar la señal del ruido, ya sea provocado por vibración o por efectos electromagnéticos que son bastante comunes en un helicóptero de radiocontrol eléctrico.

No solo hay que adaptar niveles, también puede que la salida del sensor no sea lineal o incluso ésta dependa de las condiciones de funcionamiento por lo que hay que linealizar el sensor y compensar sus variaciones. La compensación puede ser por hardware o por software, en este último caso ya no es parte del acondicionador.

En el caso de la IMU seleccionado, desde los acelerómetros y giroscopios, hasta la PCB final, contienen etapas de acondicionamiento de señal, de esta forma se redujo mucha circuitería, y el trabajo de acondicionamiento fue en mayor grado en la parte mecánica y de software.

La colocación del conjunto microcontrolador - sensor se encuentra en el tren de aterrizaje del helicóptero, esto acerca al sensor al centro de gravedad del helicóptero, de esta manera se evita tener que hacer offsets en las mediciones del sensor.

Las primeras experiencias con este sensor, se tuvieron directamente en una tableta de desarrollo (protoboard), y conectados a un microcontrolador (*ATMEL ATMEGA1280*) como se aprecia en la figura 2.25

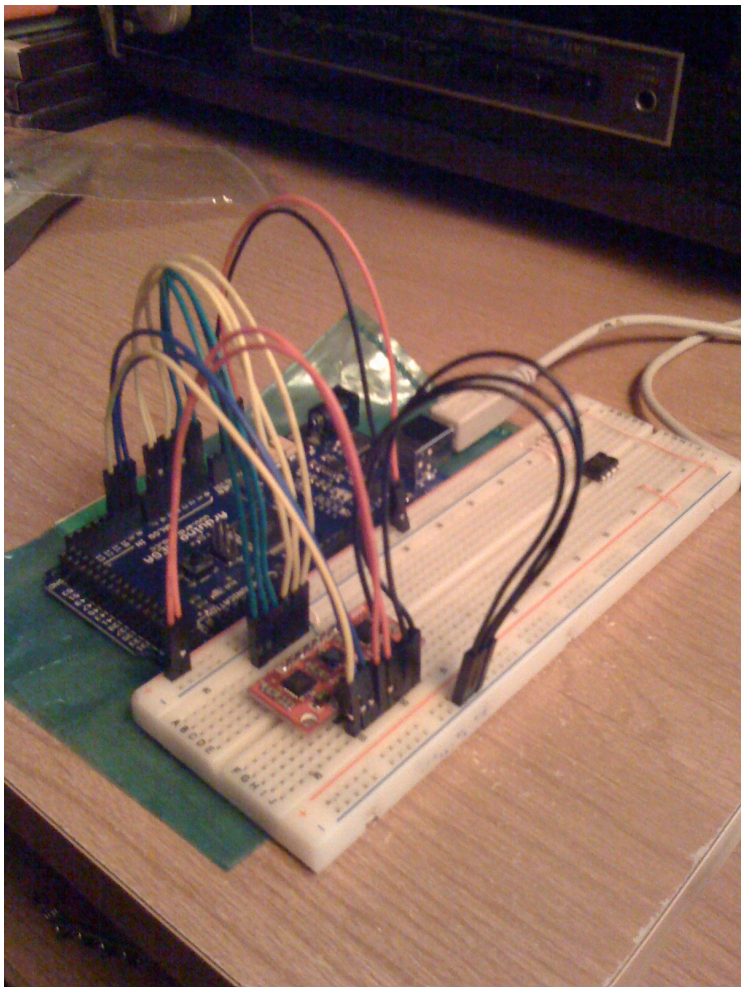


Fig. 2.25 IMU 6DOF Razor montado en una protoboard

Los primeros experimentos con este sensor montado en el helicóptero fue una sujeción del conjunto mediante una placa de cartón que estaba fija al tren de aterrizaje, después por una placa de lámina de calibre 20 (que fue descartada por el peso extra que agregaba al conjunto) y por último se utilizó una placa de estireno y una configuración de ligas que suspenden al conjunto y lo aíslan del ruido mecánico que produce el helicóptero.

Esta última configuración se ilustra en las figuras 2.26 y 2.27

La forma de la placa fue diseñada de esta manera para permitir colocar el tren de entrenamiento.

Microcontrolador

Con los subsistemas de sensado y acondicionamiento se tiene que hacer el filtrado y la fusión de las mediciones de los sensores mediante el sistema de control.

Se decidió que el microcontrolador más adecuado es el *ATMEL ATMEGA328*, que viene en una plataforma de desarrollo muy popular llamada Arduino.

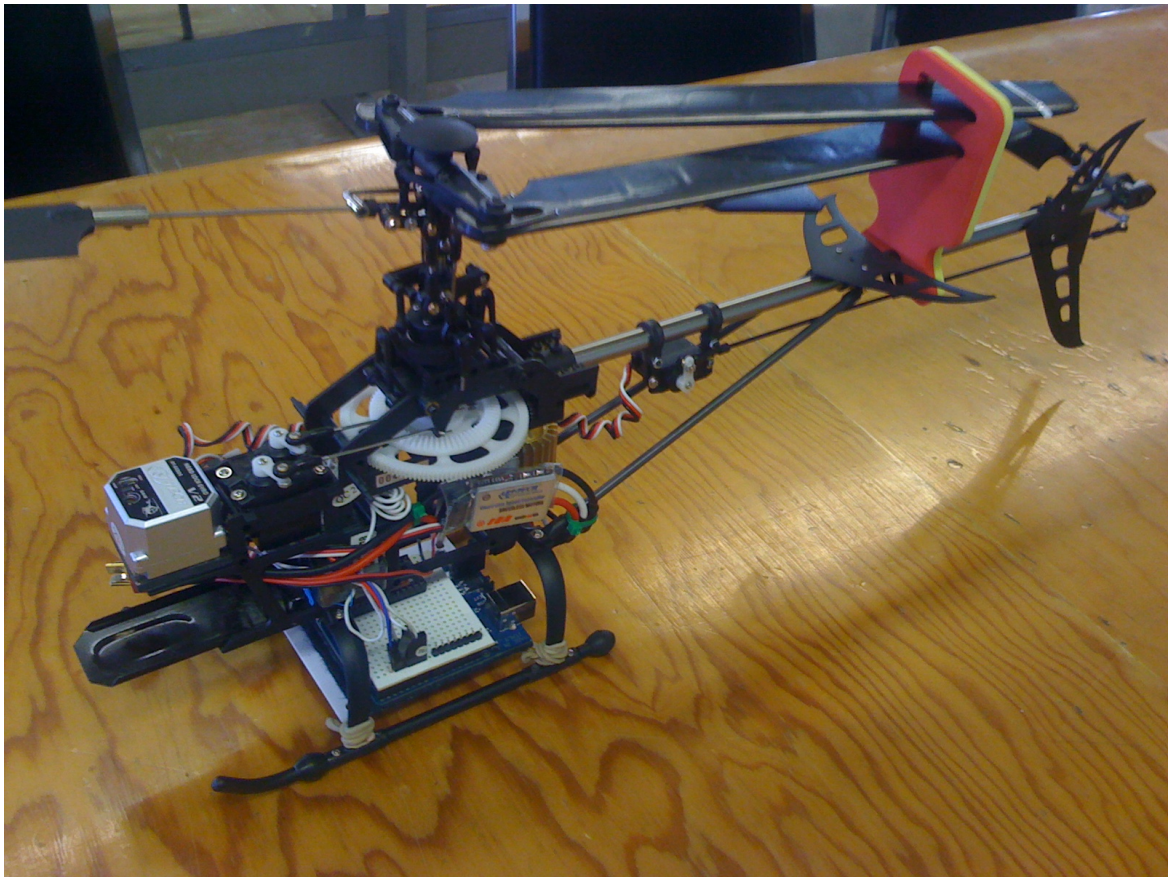


Fig. 2.26 Helicóptero con el conjunto (Arduino-Razor) montado

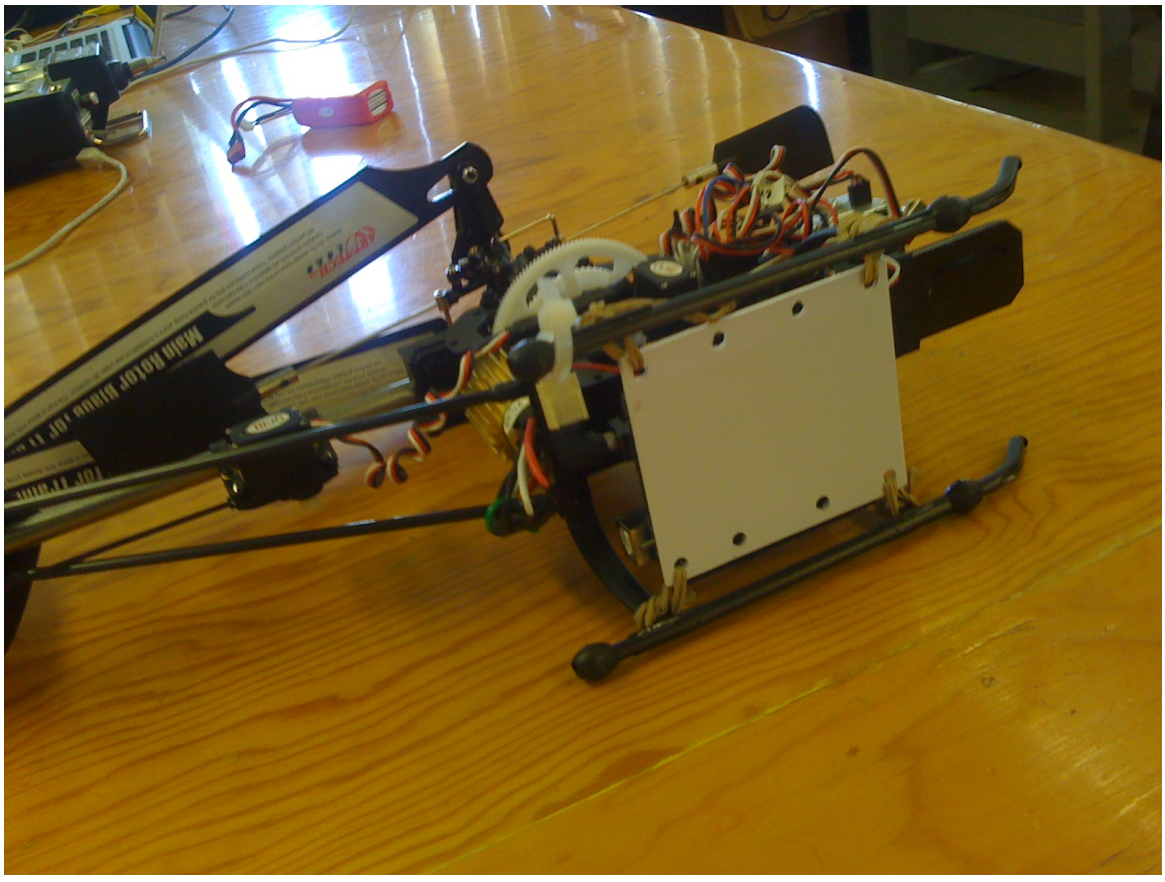


Fig. 2.27 Vista inferior del helicóptero con la instrumentación montada

Arduino

Es una plataforma de desarrollo - creación de prototipos de fuente abierta (open source) que está basada en software y hardware flexible.

Existen varias versiones de Arduino, todas son compatibles las unas con las otras; las únicas diferencias radican en el número de entradas o en el tamaño de memoria que contengan pero la estructura es similar en todas las versiones. En este proyecto se usó el Arduino Duemilanove.

ATmega328

Microcontrolador de alto rendimiento, bajo consumo de energía y de 8 bits. Arquitectura *RISC* avanzada, ejecutando instrucciones poderosas en un ciclo de reloj este microcontrolador logra acercarse a 1MIPS (millones de instrucciones por segundo) por MHZ [5]. Con todo esto, el Arduino a 16MHZ logra una capacidad de procesamiento de 16MIPS que es mas que suficiente para poder lograr el cometido de la estabilidad.

Funciones del Arduino

- Proveer un voltaje regulado y estable al sensor Razor, mediante el uso de su regulador de voltaje.
- Leer las 6 señales de salida que entrega la IMU Razor. con el convertidor analógico a digital (ADC) que tiene integrado
- Calcular los ángulos roll y pitch mediante el uso de un filtro digital paso-bajas y la implementación para 8bits de un filtro Kalman.
- Leer el canal de modulación por posición de pulsos (PPM) proveniente del receptor del radiocontrol para poder obtener los comandos del piloto humano.
- Calcular la acción de control necesaria (pulsos) mediante la implementación de un controlador difuso tipo Takagi - Sugeno, con salidas singleton.
- Comandar a los servos mediante la acción de control calculada anteriormente.
- Enviar todos los datos obtenidos por el puerto serial

El Arduino, tiene un regulador de voltaje, que además de poder ser alimentado con un mínimo de 6 volts y un máximo de 20 volts tiene 2 salidas de voltaje para alimentar sensores, una de 5 V y otra de 3.3 V. La que se usa para la IMU *Razor* es la segunda opción.

Adquisición de datos

Para un procesado de la señal eficaz hay que convertir la señal en digital. La instrumentación también estudia la conversión analógica-digital, así como la conversión digital-analógica. Ahora nos centraremos en la parte de la conversión analógica-digital, como ya se comentó en párrafos anteriores, la *IMU 6DOF Razor* nos entrega voltajes analógicos, uno por cada señal de salida. Se cuentan con 6 señales de salida, acelerómetro en cada eje (x, y, z) y giroscopio en cada eje (*roll*, *pitch*, *yaw*).

El Arduino tiene un convertido analógico digital (*ADC*) que cuenta con 6 canales de entrada analógica, el diagrama de conexión entre el sensor y el Arduino se aprecia en la figura 2.28.

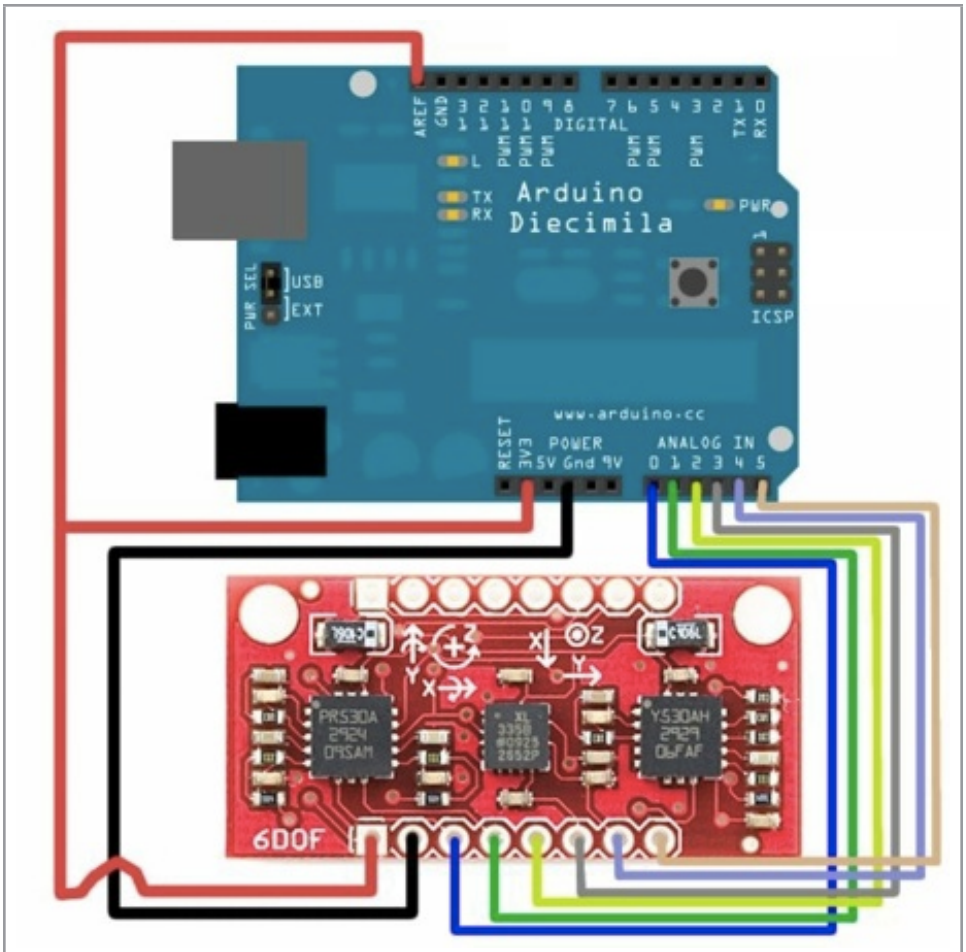


Fig. 2.28 Conexión entre IMU 6DOF Razor y Arduino

La conexión entre la alimentación y el pin *AREF* del Arduino es para poder escalar de manera correcta al *ADC*, de esta manera el valor máximo del *ADC* 1024 corresponderá a 3.3 V y no a 5 V que es la configuración por defecto.

Se ha usado el idioma inglés en la programación del Arduino, para poder llevar la filosofía de *open source*, y que sea mas fácil de entender para una audiencia internacional cuando este se libere.

El *ADC* del Arduino es de 10 bits de resolución, está conectado a un multiplexor de 6 canales que permite 6 entradas simples de voltaje. El *ADC* convierte un voltaje de entrada a través de aproximaciones sucesivas. El valor mínimo es representado por la conexión a tierra (*GND)* mientras que el valor máximo esta dado por *AREF*.

En la figura 2.29 se muestra el código para esperar y hacer que la *IMU* llegue a su temperatura adecuada de trabajo.

```
----------//Esperar a que el sensor alcance su temperatura óptima de trabajo//----------
for(int c=0; c<75; c++){
  read_adc_raw();
  delay(100);
}
for(int y=0; y<=5; y++){
  AN_OFFSET[y]=AN[y];
  Serial.println((int)AN_OFFSET[y]);
}
AN_OFFSET[5]=AN[5]-GRAVITY;
```

Fig. 2.29 Código para alcanzar la temperatura de trabajo de la IMU

La función *read_adc_raw()* (figura 2.30) fue diseñada para leer todos los canales analógicos del Arduino, además de que tiene implementado un filtro digital paso-bajas. Esta función guarda todos los resultados en un arreglo de enteros llamado *AN[]*.

```
----------//Leer los canales analógicos//----------
void read_adc_raw(void){
    float filterVal = 0.6;
    AN[0]= smooth(((Analog_Read(1)*.9)+(AN[0]*.1)), filterVal + 0.2 , AN[0]); // Gx
    AN[1]= smooth(((Analog_Read(0)*.9)+(AN[1]*.1)), filterVal + 0.2 , AN[1]); // Gy
    AN[2]=  1;//smooth(((Analog_Read(2)*.9)+(AN[2]*.1)), filterVal + 0.2 , AN[2]); // Gz
    AN[3]= smooth(((Analog_Read(5)*.9)+(AN[3]*.1)), filterVal , AN[3]); // Ax
    AN[4]= smooth(((Analog_Read(4)*.9)+(AN[4]*.1)), filterVal , AN[4]); // Ay
    AN[5]= smooth(((Analog_Read(3)*.9)+(AN[5]*.1)), filterVal , AN[5]); // Az
}
```

Fig. 2.30 Lectura de los canales analógicos

Después de varias pruebas, con un código básico de lectura, se apreció que la *IMU* necesita al menos 20 segundos para que los sensores se encuentren en el rango de temperatura adecuado.

Alisado exponencial

Los valores obtenidos del sensor tienen un componente de ruido que perjudica la función del control. La técnica de alisado exponencial simple opera sobre una serie de datos que no tiene ni tendencia ni periodicidad. Su función es pronosticar el siguiente valor de la observación a partir de la realización de un promedio ponderado entre el valor observado actual y el pronóstico anteriormente hecho [4].

$$y_{t+1} = \alpha x_t + (1-\alpha) y_t$$

Ecuación 2.1

$y_t \equiv$ el pronóstico en el tiempo t,

$x_t \equiv$ la observación en el tiempo t

$\alpha \equiv$ el coeficiente de alisado con $0 < \alpha < 1$

Analizando la ecuación vemos que cuando el coeficiente de alisado tiene valores cercanos a cero el sistema reacciona muy lento a los cambios que suceden dando por resultado una señal extremadamente alisada con un mínimo de ruido.

Al contrario, un coeficiente cercano a uno produce que el sistema tenga el mínimo retraso posible en los pronósticos efectuados y por ende la señal tenga mayor componente de ruido.

El nombre de esta técnica [4] se elucida con el siguiente hecho

$$y_{t+1} = \alpha x_t + (1-\alpha)[\alpha x_{t-1} + (1-\alpha) y_{t-1}]$$
$$y_{t+1} = \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 y_{t-1}$$

Ecuación 2.2

Si este proceso se hace recursivamente remplazando los términos de y_{t-1} el resultado es

$$y_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \alpha(1-\alpha)^3 x_{t-3} + \alpha(1-\alpha)^4 x_{t-4} + \ldots + \alpha(1-\alpha)^{t-1} x_1 + (1-\alpha)^t y_1$$

Ecuación 2.3

Entonces y_{t+1} representa el promedio ponderado móvil de todas las observaciones pasadas con los pesos decreciendo exponencialmente.

El ruido que alisa esta técnica lo identificamos como "*jitter*" que se define como una desviación o desplazamiento de algunos de los aspectos de los pulsos en una señal digital de alta-frecuencia. Como su nombre lo indica, el "*jitter*" en un sensor puede ser considerado como pulsos temblorosos, vacilantes o tambaleantes [1].

La implementación de la ecuación se realizó en la función llamada *smooth* que recibe tres parámetros: la lectura actual (*data);* el valor de alisado (*filterVal*) y la lectura anterior alisada (*smoothedVal)* y como retorno entrega la lectura actual alisada. El código se aprecia en la figura 2.31.

```
----------//Alisado exponencial//----------
float smooth(int data, float filterVal, float smoothedVal){
  if (filterVal > 1){
    filterVal = .99;
  }else if (filterVal <= 0){
    filterVal = 0;
  }
  smoothedVal = (data * (1 - filterVal)) + (smoothedVal  *  filterVal);
  return smoothedVal;
}
```

Fig. 2.31 Alisado exponencial

Esta implementación tiene la función de un filtro paso bajas digital obteniendo como ventaja una sencilla codificación en el microprocesador.

El valor del coeficiente de alisado ha sido ajustado de forma empírica observando que se tenga la mejor lectura para cada uno de los ángulos sensados.

Calculo de ángulos de orientación espacial

Para el cálculo de los ángulos se usa un filtro Kalman que va a fusionar las señales de los acelerómetros con las de los giroscopios para poder filtrar el ruido que se produce en un helicóptero.

Un filtro Kalman es un observador óptimo que estima los estados de un sistema observable a partir de las mediciones en las variables de salida [6]. Es efectivo cuando las mediciones presentan ruido semejante al ruido blanco *gaussiano* porque usa la varianza del ruido de la salida y del sistema. Para poder aplicar este filtro es necesario tener los valores de varianza del sistema y de la salida. La ecuación (2.4) de estado se ve modificada como :

$$\dot{x} = Ax + Bu + W$$
$$y = Cx + V$$

Ecuación 2.4

Donde *W* es la matriz de varianza del ruido para el sistema y *V* es el vector de varianza de ruido para la salida. La ventaja de este filtro es que la ganancia es autoajustable con el valor óptimo requerido por lo que no necesita valores característicos, lo que lo hace inmune al efecto del ruido.

Este filtro aplica sobre las variables de salida y de estados minimizando cualquier perturbación que interfiera con los datos reales. La razón de estas perturbaciones pueden ser ruidos inherentes al sistema como las no linealidades, dinámica no modelada ó a los sensores como la vibración y temperatura. Esta minimización del ruido lo hace pronosticando el nuevo estado a partir de su estimación previa añadiendo un término de corrección proporcional al error de predicción de tal forma que este último es minimizado estadísticamente.

La varianza del sistema y de la salida se asume que son independientes por lo que su covarianza es cero, tienen media cero y sus valores de varianza permanecen constantes a lo largo del algoritmo de estimación.

El proceso del filtro se divide en dos partes:

- Predicción.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

Ecuación 2.5

Son las ecuaciones de pronóstico, consiste en generar un pronóstico a priori del estado hacia adelante en el tiempo:

Este pronóstico está basado en el valor anterior del vector de estados y de la entrada. El nuevo vector de estados es una estimación la cual la identificamos con el signo -.

Por otro lado se actualiza la matriz de covarianza $P_k^- = AP_{k-1}A^T + Q$ Esta matriz indica la correlación a priori del error entre el estado actual y el estado actual estimado. Su cálculo se realiza mediante la matriz del sistema, la matriz de varianza del ruido del proceso y el valor anterior de la matriz de covarianza a *posteriori*.

- Corrección.

Se calcula la ganancia de kalman: $K_k = P_k^- C^T (CP_k^- C^T + R)^{-1}$. Esta ganancia se ve identificada por los valores de varianza de la medición representados en R y por el error correlacionado del vector de estado a priori. El valor de la ganancia es mínimo mientras la covarianza del error del vector del estado tienda a cero.

Calculamos el vector de estados a posteriori mediante la *innovación*:

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$$

El término *innovación* se refiere a que se usa la diferencia del valor medido actual contra el valor del estado estimado a *priori*, la diferencia se escala con la ganancia de kalman y se suma al valor estimado a *priori*.

Por último actualizamos la matriz de correlación del error a posteriori: $P_k = (I - K_k C)P_k^-$.

Los dos pasos anteriores se ejecutan cíclicamente como se muestra en la figura 2.32.

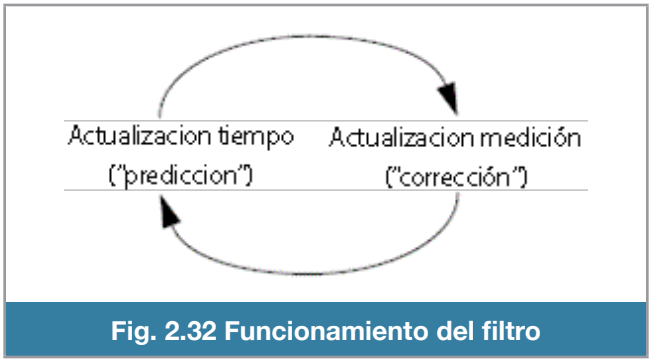


Fig. 2.32 Funcionamiento del filtro

La implementación de este algoritmo se aplica para conocer los ángulos que tiene el helicóptero en sus ejes horizontales, cada uno de los ángulos pitch y roll se obtienen mediante la integración del valor del giroscopio y se corrige con el valor del arcotangente que forman las dos mediciones de los

acelerómetros perpendiculares al giroscopio. La corrección que hacemos con el acelerómetro es necesaria por el efecto de deriva que tiene el giroscopio.

Ahora bien, debemos recalcar que el ángulo no es posible obtenerlo de forma correcta a partir de solo acelerómetros. Suena bien solo usar un par de acelerómetros pero esto solo aplica cuando el cuerpo no tiene alguna aceleración, es decir, si el cuerpo experimenta una aceleración los acelerómetros la registran y por lo tanto la medición que nos proporciona seria un ángulo incorrecto porque una de las componentes utilizadas para medir el ángulo se vería influenciada.

Continuando con la implementación, formamos nuestro espacio de estados discreto el cual está formulado como [7]:

$$x_{k+1} = Ax_k + Bu_k + Qw_k$$
$$\begin{bmatrix} \alpha \\ bias \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -dt \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ bias \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \end{bmatrix} u_k + \begin{bmatrix} q_\alpha & 0 \\ 0 & q_{bias} \end{bmatrix} w_k$$

Ecuación 2.3

El vector de estados son los valores de α y *bias* que corresponden al ángulo y al valor de giroscopio en estado estacionario. La entrada u es el valor de giroscopio en el tiempo k y el diferencial *dt* nos indica el tiempo entre muestras.

$$y_{k+1} = Cy_k + Rv_k, \quad D = 0$$
$$\alpha_{k+1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ bias \end{bmatrix}_k + r_\alpha v_k$$

Ecuación 2.4

Por otro lado nuestra medición proviene del ángulo calculado a través del acelerómetro. Los valores de varianza del sistema está expuesto en la matriz Q y de la medición en el escalar R.

En el contexto del microcontrolador se realizó una simplificación de cálculos. Se definen tres funciones principales:

Inicialización de valores

Coloca los valores iníciales del vector de estado a cero y de la matriz de correlación donde la diagonal principal reflejan la autocorrelación del error para cada componente del vector de estados que es igual a uno. Esto se ejecuta en la función *kalmanInitState*.

Actualización del vector de estados

Esta función representa la estimación del vector de estado. Calcula el vector de estados y calcula la matriz de correlación a priori. El nuevo valor del ángulo es realizado mediante la suma del valor an-

terior del ángulo más el valor del giroscopio en la medición actual (q_m) sin el término del valor de *bias* multiplicado por *dt*. La función que realiza esta tarea es *state_update*.

Corrección del vector de estados

La corrección del vector de estados se realiza en la función *kalman_update* obteniendo el error del ángulo medido por el acelerómetro con el ángulo estimado por el giroscopio. En el cálculo de la ganancia de Kalman $K_k = P_k^- C^T (CP_k^- C^T + R)^{-1}$ hay un término que se usa dos veces que es $P_k^- C^T$ y es calculado como:

```
PCt_0 = C_0 * p_kd->P[0][0];

PCt_1 = C_0 * p_kd->P[1][0];
```

Ecuación 2.5

Se omite el segundo valor de la matriz C (C_1) porque tiene un valor cero.

También se corrige la matriz de covarianza con el cálculo anticipado del término CP_k^- para que solo sea calculado una vez y no dos veces.

La ejecución de estas dos últimas funciones se realiza a intervalos con valor *dt* que es obtenido mediante el temporizador cero el cual cuenta el número de milisegundos que posteriormente se transforman a segundos para utilizarlos en tales funciones.

Lectura del canal PPM

El canal Modulación por Posición de Pulsos (*Pulse Position Modulation*, PPM) es una manera de codificación/decodificación para modulación de señal, es un tipo de modulación en la cual una palabra R bits es codificada por la transmisión de un único pulso que puede encontrarse en alguna de las 2^M posiciones posibles.

Los sistemas PPM consisten en un marco de datos que contiene un pulso sincronizador seguido por un número de pulsos pequeños que son iguales al número de canales que tenga el radiocontrol. Esta estructura se muestra en la figura 2.33 donde vemos que cada canal es codificado a través de la duración del pulso.

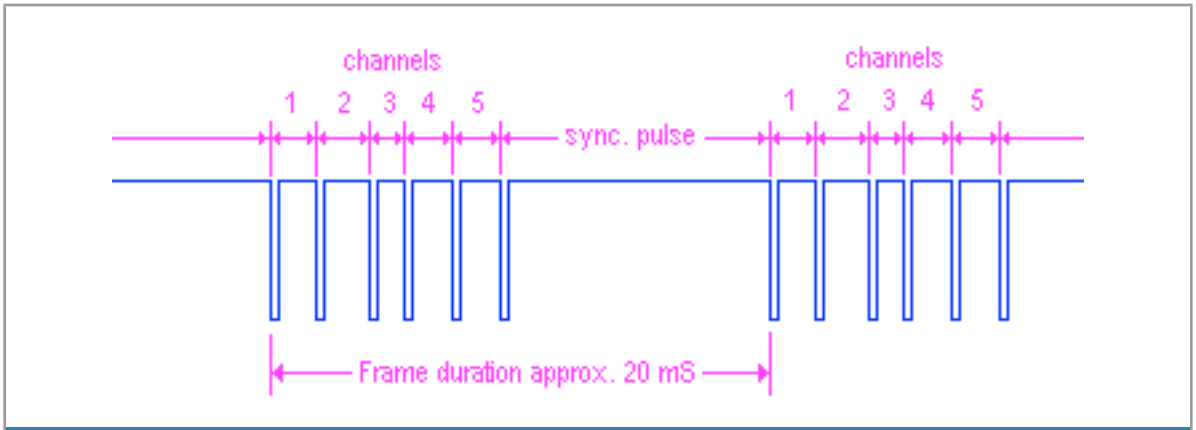


Fig. 2.33 Estructura de un pulso PPM

El circuito codificador del radiocontrol lee cada potenciómetro e interruptor del radiocontrol para después convertirlo en un ancho de pulso y después construye el marco PPM. Este es enviado al receptor y cada ancho de pulso corresponde respectivamente a la posición en un servomotor (figura 2.34).

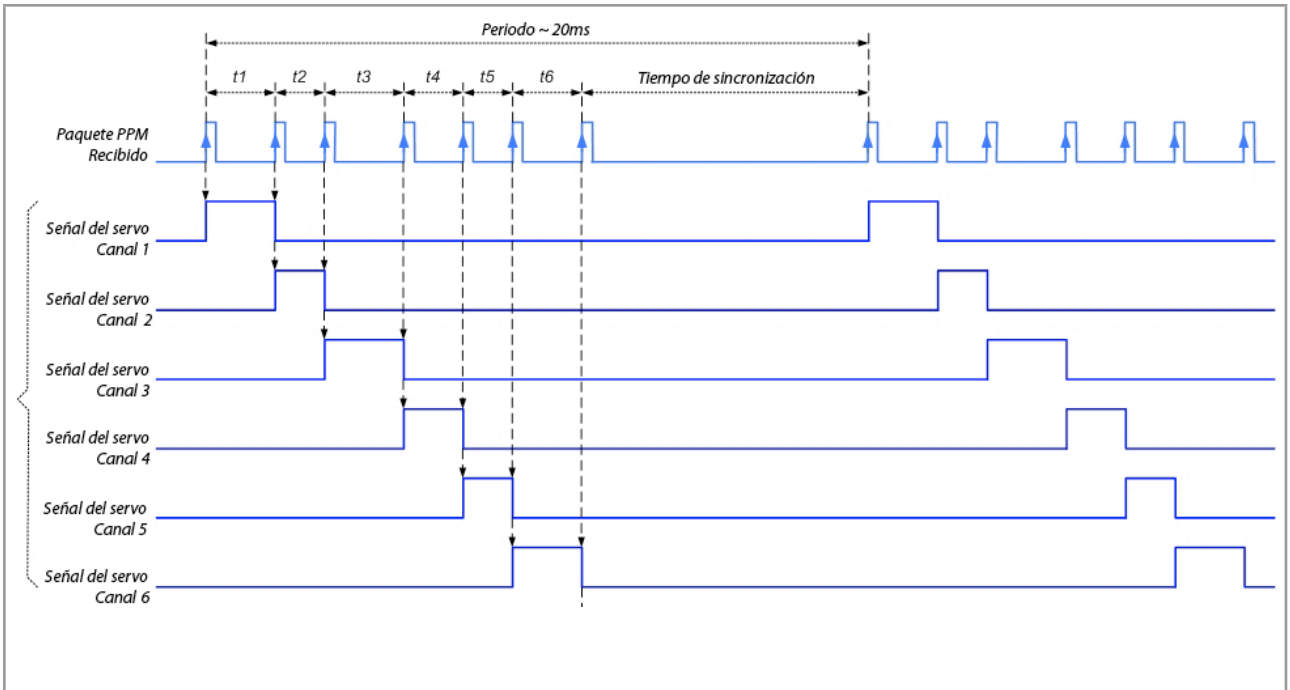


Fig. 2.34 Estructura de un pulso PPM de 6 canales

Se tienen dos partes fundamentales: inicio del marco y contenido del marco. El inicio del marco se representa por un pulso en bajo durante un periodo de 10 ms, después viene la codificación de las señales, cada señal tiene un periodo de 2 ms, el valor de la señal es codificado como un ancho de pulso en bajo siendo la duración el valor de la señal, es decir el ciclo de trabajo de cada señal indica el valor numérico que tienen. Entonces se tienen seis periodos de 2 ms, cada periodo con un ciclo de trabajo.

La duración del ciclo de trabajo codificado de cada señal es proporcional con la duración del ciclo de trabajo de una Modulación por ancho de Pulso (*Pulse Width Modulation*, PWM) para controlar un servomotor lo que hace transparente el manejo de estos valores. Esta proporción fue determinada empíricamente como la mitad de su valor.

Esta es la manera mas eficiente y adecuada para poder obtener los canales de cualquier receptor comercial de radiocontrol, debido a la simplicidad.

Para poder leer esta señal se soldó un cable al pin número uno del registro de corrimiento del receptor *[DS SHIFT REGISTER]* mostrado en la figura 2.35 y se colocó en la entrada número cinco del Arduino. Nuestra tarea es poder identificar la duración de los pulsos en bajo de las señales para conocer el valor que se envían a los servomotores.

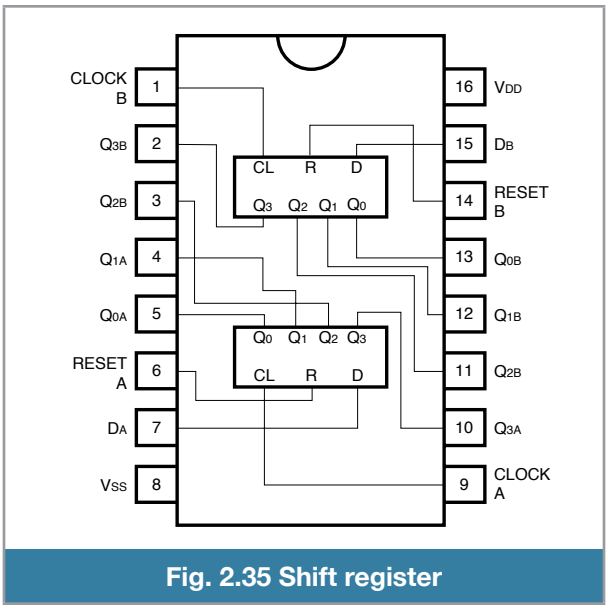


Fig. 2.35 Shift register

Para hacer las cosas mas eficientes, se recurrió a usar uno de los 3 *timers* del Arduino, se usa el *Timer1* para poder leer el canal PPM y obtener todos los canales del radiocontrol.

Se utiliza el *TIMER1* en modo de captura para procesar la información cada vez que exista un flanco de subida en la señal que indica el final del ciclo de trabajo. Por cada flanco comprobamos si fue el inicio de marco o si fue el ciclo de trabajo de alguna de las señales. Por cada señal obtenida se escala a la mitad.

De esta forma podemos conocer los valores de todos los canales a través de uno solo, además que no demanda gran tiempo de procesador ya que está sujeto al vector de interrupciones del *Timer1*.

La funcion *setup_timer1* (figura 2.36) lo que hace es configurar al *Timer1*, para que solamente funcione según se requiera y desactivar las demás funciones que hacen uso del *Timer1*.

```
----------//Configuración del Timer1//----------
void setup_timer1(){
//Desactivar todas las interrupciones | disable all interupts
TIMSK1 &= ~( _BV(TOIE1) | _BV(ICIE1) | _BV(OCIE1A) | _BV(OCIE1B));
//Fijar modo del timer | set timer mode
TCCR1A &= ~( _BV(WGM11) | _BV(WGM10) );
TCCR1B &= ~( _BV(WGM12) | _BV(WGM13) | _BV(ICNC1));
//Capturar flanco de subida | capture raising edge
TCCR1B |= _BV(ICES1); //capture raising edge
//Prescaler 1/8
TCCR1B |= _BV(CS11);
TCCR1B &= ~( _BV(CS12) | _BV(CS10) );
//Desactivar salidas | disable outputs
TCCR1A &= ~( _BV(COM1A0) | _BV(COM1A1) | _BV(COM1B0) | _BV(COM1B1));
//Activar interrupcion de captura | enable capture interrupt
TIMSK1 |= (1<<ICIE1);
}

ISR(TIMER1_CAPT_vect){
//static unsigned int lasticr; //icr at last caputre
static unsigned char cserinp; //Entrada actual de servo | current input servo
unsigned int licr;

if(ICR1>10000){
//Pulso muy largo, comenzar con nuevo frame | pulse too long, means start of new frame
  cserinp=0;
}else{
//Pulso correcto, guardar medición, ir al siguiente canal | pulse good, take reading, go to next channel
  serinp[cserinp]=((ICR1/2));
  cserinp++;
}
TCNT1=0;
}
----------//------------------//----------
```

Fig. 2.36 Lectura del canal PPM mediante el Timer1

Envío de datos a la estación base

Una de las funciones temporales del Arduino, es la envío de datos a la estación base vía puerto serial. Es una función temporal ya que sólo funciona para depurar errores y mantenernos informados acerca del estado del sensor y del Arduino en la etapa de desarrollo.

Nuestras necesidades de obtención de información son las siguientes:

- Datos capturados directamente del sensor, 6 datos de voltajes analógicos (0 - 1024 por dato).
- El estado de los cálculos del filtro Kalman, los ángulos calculados, la razón de cambio y el sesgo para cada ángulo.
- Los valores del radiocontrol de los canales 1, 2 y 5 (cíclico lateral, cíclico longitudinal e interruptor de activación).
- Los valores calculados por el sistema difuso para el PWM controlado de los servomotores.

Se tiene una cadena aproximadamente de 100 a 150 caracteres por mensaje. Se diseñó un sistema de mensajes que tuviera un inicio de mensaje representado por "*!!!*" y un final de mensaje representado por "*****".

En la figura 2.37 se muestra el código del Arduino para enviar los mensajes a la estación base, el cual se creó de tal manera que, para el caso de *PRINT_ANALOGS* sea igual a uno, se mande imprimir y si vale cero, no se mande a imprimir; es lo mismo para los demás casos. De esta manera se puede controlar fácilmente que mande y que no mande a la estación base.

Para la estación base se escogió el software *Labview* (***L****aboratory* ***V****irtual* ***I****nstrument* ***E****ngineering* ***W****orkbench*) por su facilidad y versatilidad para hacer Interfaces máquina - humano (*Human Machine Interfaces, HMI's*).

En la interfaz se exhiben de manera gráfica y textual los datos que han sido recibidos por el puerto serial, que muestra en figura 2.38. En la parte inferior izquierda se aprecian los valores analógicos de la IMU o sea, los valores de los acelerómetros y de los giroscopios para cada eje.

En ea lado derecho se tienen dos pantallas estilo osciloscopio, en la superior aparecen las 3 señales del acelerómetro y en la inferior las de los giroscopios; estos gráficos tienen historia, de tal manera que podemos apreciar la vibración u otros problemas que se vayan presentando de manera histórica.

En la parte central de la interfaz, se muestran dos indicadores deslizantes para las posiciones del cíclico lateral y longitudinal del radiocontrol del piloto. El cubo verde superior derecho sirve para la simulación 3D y se utiliza para apreciar la orientación espacial del helicóptero.

```
----------//Envío de datos por puerto serie//----------
void printdata(void){
      Serial.print("!!!");
      #if PRINT_ANALOGS == 1
      Serial.print("AN0:");
      Serial.print(read_adc(0));
      Serial.print(",AN1:");
      Serial.print(read_adc(1));
      Serial.print(",AN2:");
      Serial.print(read_adc(2));
      Serial.print(",AN3:");
      Serial.print(read_adc(3));
      Serial.print(",AN4:");
      Serial.print(read_adc(4));
      Serial.print(",AN5:");
      Serial.print(read_adc(5));
      Serial.print (",");
      #endif
      #if PRINT_KALMAN == 1
      Serial.print("ANX:");
      Serial.print(ToDeg(xkaldata.angle));
      Serial.print(",ANY:");
      Serial.print(ToDeg(ykaldata.angle));
      Serial.print(",RTX:");
      Serial.print(ToDeg(xkaldata.rate));
      Serial.print(",RTY:");
      Serial.print(ToDeg(ykaldata.rate));
      Serial.print(",BSX:");
      Serial.print(ToDeg(xkaldata.q_bias));
      Serial.print(",BSY:");
      Serial.print(ToDeg(ykaldata.q_bias));
      Serial.print (",");
      #endif
      #if PRINT_RADIO == 1
      Serial.print("RC1:");
      Serial.print(serinp[0]);
      Serial.print(",RC2:");
      Serial.print(serinp[1]);
      Serial.print(",RC3:");
      Serial.print(serinp[4]);
      Serial.print (",");
      #endif
      #if PRINT_CONTROL == 1
      Serial.print("LAT:");
      Serial.print(lat);
      Serial.print(",LON:");
      Serial.print(lon);
      Serial.print (",");
      #endif
      Serial.println("***");
}
----------//------------------//----------
```

Fig. 2.37 Envío de datos por puerto serial

Finalmente, en la esquina inferior izquierda de la figura 2.38 aparecen los indicadores de los ángulos *roll*, *pitch* y *yaw*. En la figuras 2.39 y 2.40 se ilustra una parte del diagrama de bloques del VI principal de la interfaz gráfica.

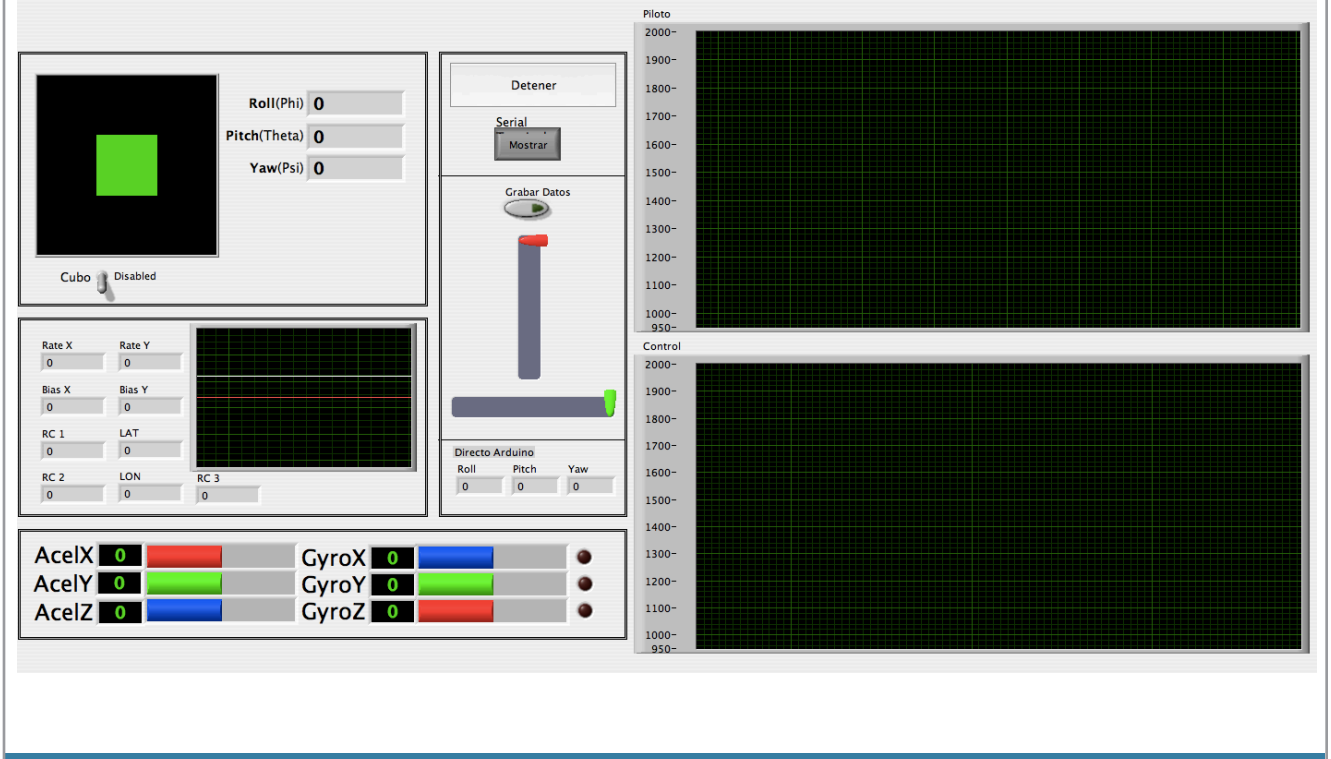


Fig. 2.38 Interfaz gráfica de la estación base

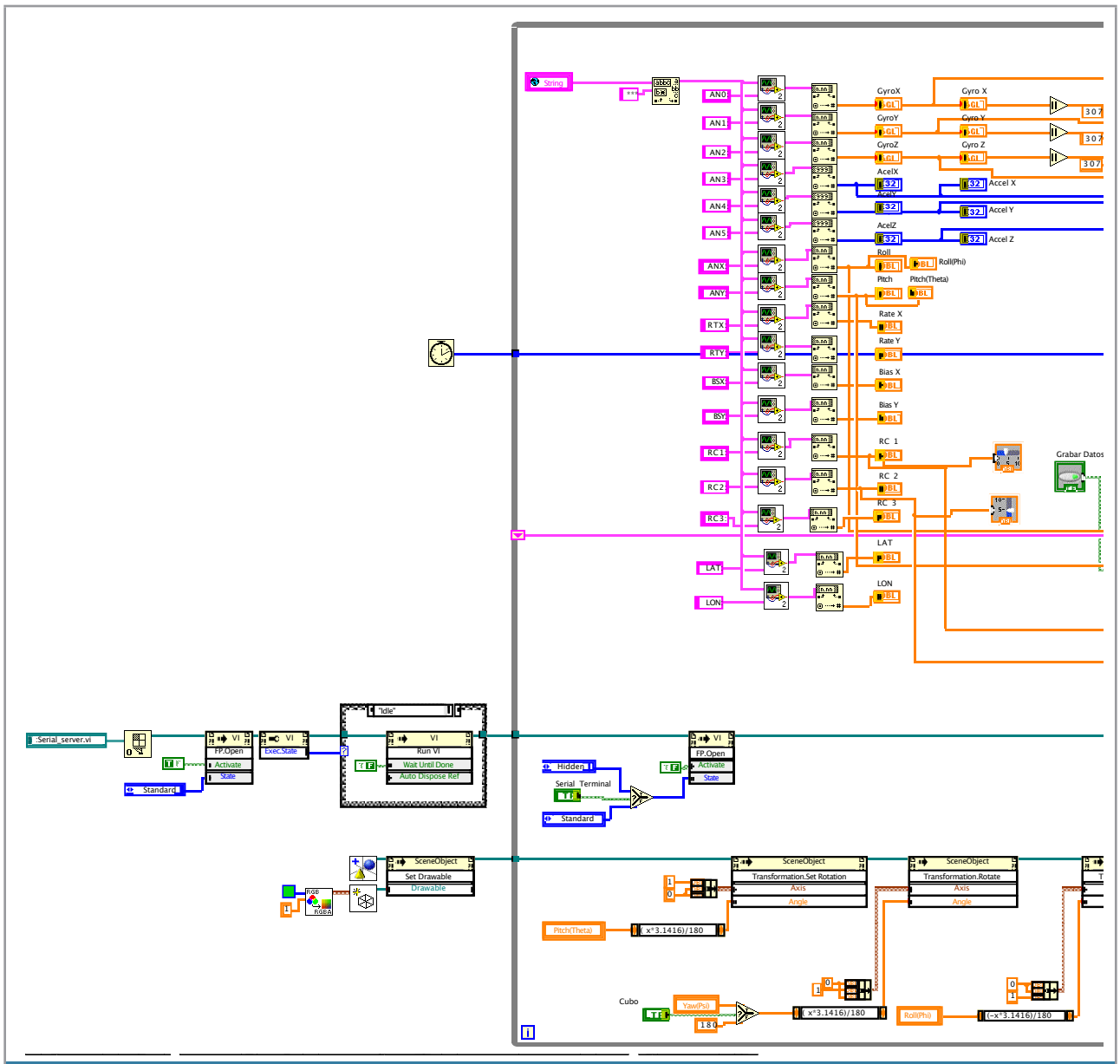


Fig. 2.39 Diagrama de bloques de la interfaz (parte 1)

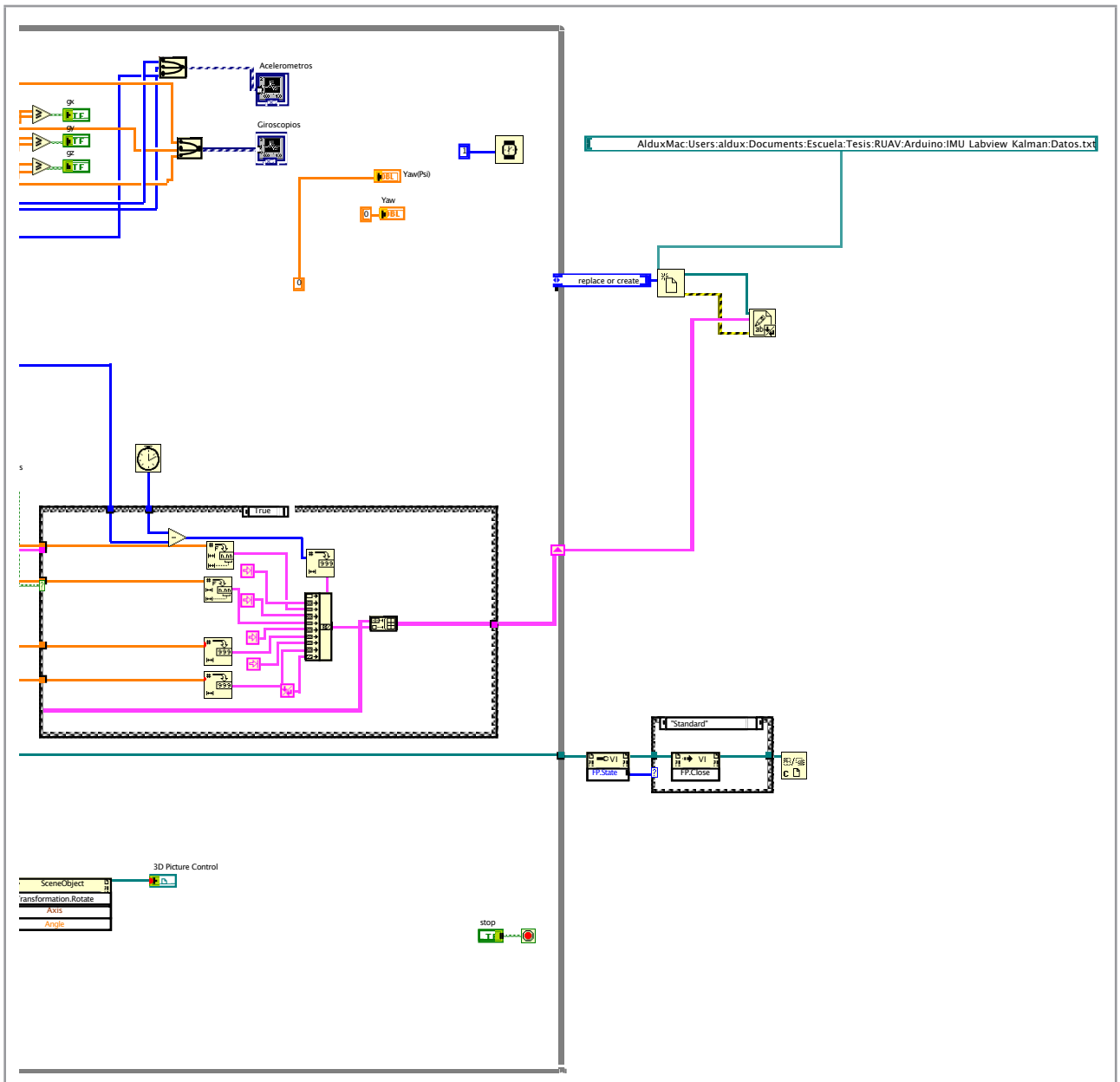


Fig. 2.40 Diagrama de bloques de la interfaz (parte 2)

3.- Control

Sistemas Neuro-Difusos

Los sistemas neuro-difusos forman parte de una nueva tecnología de computación llamada computación flexible (*soft-computing*), la cual tiene su origen en la teoría de la inteligencia artificial. La computación flexible engloba un conjunto de técnicas que tienen en común la robustez en el manejo de la información imprecisa e incierta que existe en los problemas relacionados con el mundo real [1].

Las redes neuronales (*RNA's*) son la implementación en hardware y/o software de modelos matemáticos idealizados de las neuronas biológicas. Las *RNA's* son interconectadas unas a otras y son distribuidas en capas de tal forma que emula en forma simple la estructura neuronal del cerebro. Cada modelo de neurona es capaz de realizar algún tipo de procesamiento a partir de estímulos de entrada y ofrecer una respuesta, por lo que las RNA's en conjunto funcionan como redes de computación paralelas y distribuidas similares a los sistemas cerebrales biológicos. Las RNA's, tal como los sistemas cerebrales biológicos, deben de ser entrenadas.

La lógica difusa o borrosa es una técnica de computación flexible que le permite a una computadora clasificar información del mundo real en una escala infinita acotada por los valores falso y verdadero; tiene por objeto proporcionar un soporte matemático formal al razonamiento basado en el lenguaje natural, el cual se caracteriza por tratarse de un razonamiento de tipo aproximado que hace uso de proposiciones que expresan información de carácter impreciso.

Algunas de las características de la lógica difusa son:

- Su escalamiento es sencillo
- Es tolerante a datos imprecisos
- Puede ser construida sobre la información de la experiencia de los pilotos que manejan el helicóptero

- Está basada en el lenguaje utilizado por humanos (Reglas Lingüísticas)
- Algunos modelos permiten que los conceptos matemáticos empleados sean sencillos (Máximos, Mínimos, Ponderaciones)

La lógica difusa debe de ser distinguida de la incertidumbre en el sentido en que la lógica difusa describe la ambigüedad de un evento, mientras que la incertidumbre la ocurrencia de un evento.

La lógica difusa y las redes neuronales tienen propiedades computacionales particulares que las hacen adecuadas para ciertos problemas particulares. Las redes neuronales ofrecen ventajas como el aprendizaje, adaptación, tolerancia a fallas, paralelismo y generalización mientras que los sistemas difusos razonan con información imprecisa a través de un mecanismo de inferencia bajo incertidumbre lingüística, son buenos explicando sus decisiones pero no pueden adquirir automáticamente las reglas que usan para tomarlas.

Los sistemas neuro-difusos combinan la capacidad de aprendizaje de las RNA's con el poder de interpretación de los sistemas de inferencia difusos cuyas características son las siguientes:

- Aplicabilidad de los algoritmos de aprendizaje desarrollados para redes neuronales
- Posibilidad de promover la integración de conocimiento (implícito que puede ser adquirido a través del aprendizaje y explícito que puede ser explicado y entendido)
- La posibilidad de extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos

Redes Neuro-difusas

El sistema neuro-difuso tiene como componentes un sistema difuso, el cual tiene la función de evaluar las acciones a tomar dependiendo de las entradas recibidas a través de varios mecanismos y de una red neuronal artificial MLP (multi layer perceptron), que tiene el propósito de ajustar los parámetros de las curvas de las funciones de membresía para la fuzificación y los valores de los *singletons* (en nuestro caso) para la construcción de valor numérico de la variable de salida [2].

La justificación para usar estos dos sistemas obedece a poder compensar las debilidades que tiene cada uno.

Las fortalezas del sistema difuso son que puedes tratar el problema mediante un lenguaje natural, definir conjuntos que permitan la estructuración y clasificación de la información de forma que cualquier humano que conozca el proceso lo pueda comprender.

Su debilidad radica en que se deben ajustar los parámetros que definen los conjuntos de información.

Por otro lado las fortalezas de la red neuronal artificial reside en que la información es auto organizada lo que implica que automáticamente la red tiene su propia representación de la información; tiene un proceso de aprendizaje iterativo haciendo que cada vez reproduzca con mejor fidelidad el comportamiento que debe de tener ante una serie de estímulos y; por último, la generalización de la respuesta que, ante estímulos no conocidos puede dar una respuesta asertiva [1].

Una muy grande desventaja es que requiere que los datos de entrenamiento sean representativos, abundantes y sin ruido; además, hay que definir la arquitectura de la red indicando el número de capas, número de neuronas por capa y las funciones de activación de cada capa.

Características de los sistemas neuro - difusos

Los sistemas neuro - difusos trasladan el conocimiento *a priori* en una topología de red donde existen capas que representan las entradas, parámetros de las funciones de membresía, las reglas y la salida. Los sistemas difusos Takagi-Sugeno de orden cero (*singleton*) son los mejores para trabajar con redes neuronales artificiales. El entrenamiento de retropropagación se usa para ajustar los parámetros de las funciones de membresía, la suma de los mínimos cuadrados calcula los coeficientes polinomiales.

Sinergizando las ventajas de cada una se obtienen una rápida convergencia debido al entrenamiento usado, automáticamente se realiza el ajuste fino del sistema difuso y se garantiza que la curva de control sea suave debido a las interpolaciones que hace el mecanismo de inferencia con las reglas.

Su contraparte es que el número de reglas crece exponencialmente a razón de p^n veces, donde p es el número de funciones de membresía por variable de entrada y n es el número de variables de entrada. Existen oscilaciones alrededor de los puntos de la superficie de control debido a la alta cantidad de particiones. El número de salidas está restringido a una variable.

A grandes rasgos, los sistemas difusos proveen interpolación y suavidad en la curva que define al control mientras que las redes neuronales artificiales agregan adaptabilidad de los parámetros del sistema difuso mediante el entrenamiento de retropropagación [2].

En la tabla 3.1 se observan las principales comparaciones que tienen la lógica difusa y las redes neuronales artificiales.

Lógica Difusa	Redes Neuronales
Permite utilizar el conocimiento disponible para optimizar el sistema directamente	No existe un método sencillo que permita modificar u optimizar la red, ya que esta se comporta como una "caja negra"
Permite describir el comportamiento de un sistema a partir de sentencias "si - entonces"	La selección del método apropiado de red y el algoritmo de entrenamiento requiere de mucha experiencia
Permite utilizar el conocimiento de un experto	Permite hallar soluciones a partir de un conjunto de datos
El conocimiento es estático	Son capaces de aprender y de auto-adaptarse
Existen muchas aplicaciones comerciales	Su aplicación es mayormente académica
Permiten encontrar soluciones sencillas con menor tiempo de diseño	Requieren un enorme esfuerzo computacional

Tabla 3.1 Comparación de sistemas difusos y neuronales

Sistemas difusos

Las decisiones humanas son muy difíciles de realizar mediante un sistema de cómputo debido a las diversas condiciones y circunstancias que afectan el procesamiento. La experiencia de un humano no puede ser modelada ya que ese conocimiento fue adquirido de forma empírica, para que una computadora pueda replicar las acciones de un humano debe aprender de forma empírica tales conocimientos.

Los sistemas difusos son una forma de lograr este cometido, se basan en la teoría de conjuntos difusos y lógica difusa para captar y aplicar el conocimiento.

Nuestras vidas se estructuran en sistemas donde tenemos clasificaciones de los eventos que nos rodean. Estas clasificaciones nos proporcionan tener un conocimiento estructurado acerca de nuestro entorno. Las clasificaciones que aplicamos son las definiciones de conjuntos, hay una situación que debemos aclarar, ¿Qué pasa cuando un evento tiene cierto grado de pertenencia a un conjunto?, por ejemplo, se puede hablar del enunciado "Los condones que ofrecen la máxima protección". Estamos seguros que ciertas marcas ofrecen mayor protección que otras pero no todas ofrecen el

100% o el 0% de protección por lo que existe un gradiente que indica que tanto un marca de condones satisface la premisa. Así pues definimos los conjuntos difusos como aquellos en que sus elementos tienen un grado de pertenencia siendo una generalización de los conjuntos determinados.

La lógica difusa es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada vaga, ambigua, imprecisa, con ruido o incompleta. Para poder aplicar un sistema difuso a un problema físico es necesario hacer varias transformaciones a la información que nos proporciona el problema. Primero se tiene que definir cuál es el dominio de las variables de entrada, es decir, cual es el rango de valores que adquieren las distintas variables. El universo del discurso consiste en definir los límites de las variables de entrada.

El conocimiento que adquiere un humano es siempre cualificado por lo que es necesario asignar "etiquetas" al grupo de valores que adquieren las variables. No hay límite para el número de etiquetas; entre más etiquetas tenga una variable mejor asertividad se logra pero con un costo de procesamiento mayor. Generalmente con tres etiquetas es suficiente para iniciar.

Ahora hay que establecer una manera de poder asignar la experiencia (información cualificada) en datos determinísticos (información cuantificada). Este proceso se hace mediante una función de membresía la cual define el grado de pertenencia que tiene un elemento del universo del discurso sobre un conjunto difuso. El conjunto difuso viene siendo la etiqueta que definimos previamente y el elemento es uno de los valores que puede adquirir la variable. Un elemento pude estar perteneciendo a más de un conjunto difuso pero con diferente grado.

La función de membresía es una función que tiene como imagen el rango [0,1] siendo cero ninguna pertenencia y uno total pertenencia al conjunto. Estas funciones tienen parámetros que configuran el rango del dominio. Cada etiqueta tiene asociada una función de membresía [3].

Funciones de membresía

Las primeras funciones de membresía son la triangular y trapezoidal. Estas funciones son fáciles de calcular y se basan en el uso de segmentos de recta para definir la forma. La función triangular (ecuación 3.1) se define mediante tres puntos; el primero indica inicio de la pendiente ascendente, el segundo es el pico del triángulo y el tercero el término de la pendiente descendente. La función trapezoidal (ecuación 3.2) define un punto medio adicional para trazar una meseta en la parte superior.

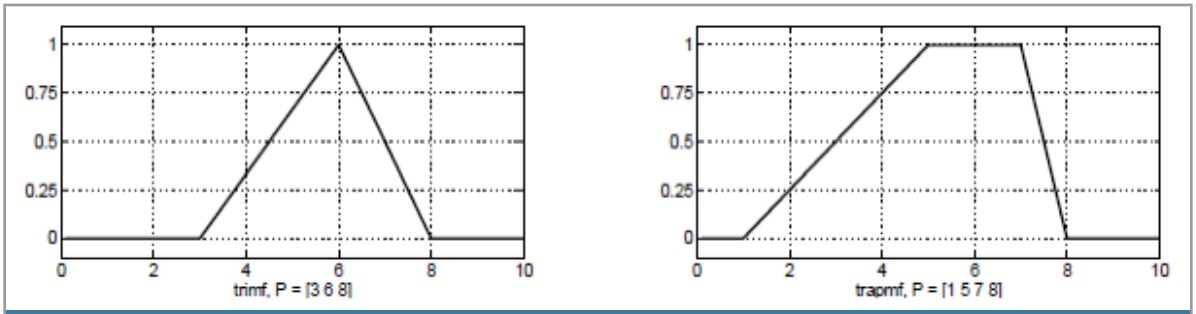


Fig. 3.1 Funciones triangular y trapezoidal [3]

$$trimf = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

Ecuación 3.1

$$trapmf = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

Ecuación 3.2

Basados en la distribución gausiana, en la figura 3.2 se muestra la primera función de membresía, esta es simétrica teniendo como parámetros de construcción la desviación estándar y valor promedio. La segunda función está definida por dos funciones gausianas, una por la izquierda y otra por la derecha las cuales, se unen en el punto central sin sobrepasar el valor de 1. La última es una distribución de campana con los parámetros de desviación estándar, ancho de la meseta y promedio. Se muestran las funciones en las ecuaciones 3.3 y 3.4.

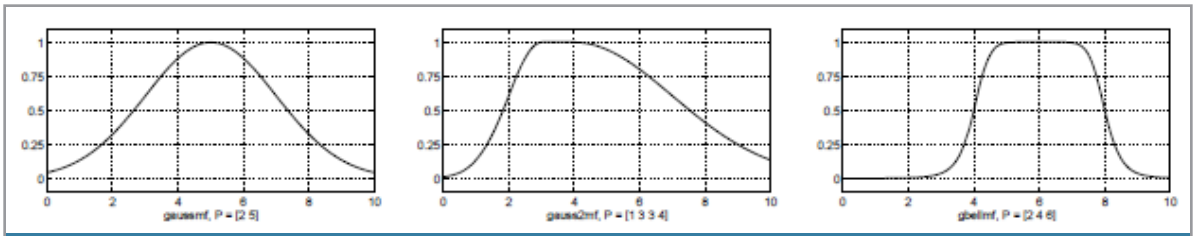


Fig. 3.2 Funciones gausianas [3]

$$gaussmf = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

Ecuación 3.3

$$gbellmf = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

Ecuación 3.4

Las funciones sigmoidales son asimétricas (figura 3.3), las últimas dos son variantes de la primera. La segunda función es la diferencia de dos sigmoidales y la tercera es el producto de ellas. Se muestra en la ecuación 3.3.

La función sigmoidal se define mediante el punto de inflexión y la desviación estándar.

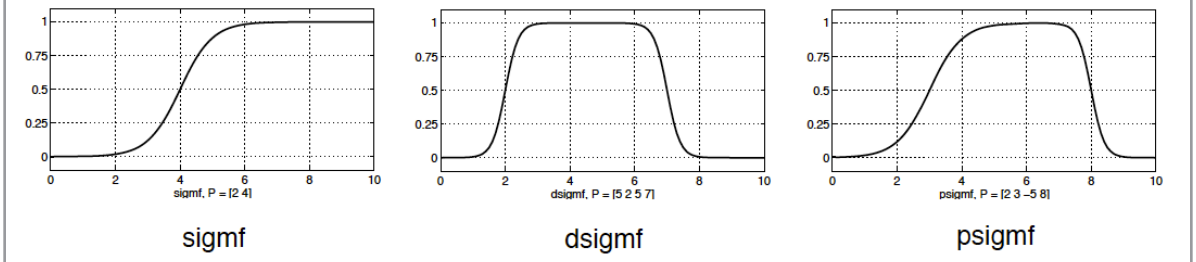


Fig. 3.3 Funciones sigmoidales [3]

$$sigmf = \frac{1}{1+e^{-a(x-c)}}$$

Ecuación 3.5

Todo lo anterior se engloba en un término llamando variable lingüística la cual se define mediante el vector de la ecuación3.6.

$$L=[\,x,\ T,\ X,\ M\,]$$

Ecuación 3.6

De la ecuación anterior, cada componente indica:

- *x* es el nombre de la variable lingüística que es el equivalente del nombre de la variable de entrada.
- *T* son las etiquetas en las que se representa los valores que adquiere la variable de forma cualificativa.
- *X* es el universo del discurso el cual contiene el dominio de la variable x el cual puede ser continuo o discreto.
- *M* son las funciones de membresía que asocia a las etiquetas T con los valores X del universo de discurso.

Todo el anterior proceso se conoce como fuzzificación que en resumidas palabras es la tarea de convertir los valores determinísticos a difusos en donde cada valor que adquiere la variable tiene un cierto grado de pertenencia a los conjuntos difusos definidos para tal variable. El grado de pertenencia para cada conjunto difuso lo definen las reglas de membresía. Estas funciones deben ser continuas y se definen sobre el dominio de la variable.

Para procesar las variables de entrada es necesario crear reglas que indiquen que acciones se tienen que tomar dada una serie de condiciones llamado antecedente. Las reglas tienen la siguiente estructura:

SI ANTECEDENTE ENTONCES CONSECUENTE

La formulación de la regla es mediante la condición SI, y está compuesta por el antecedente que consiste de las condiciones formuladas con operadores difusos sobre elementos pertenecientes a los conjuntos difusos. El resultado de la evaluación del antecedente indica el grado de cumplimiento de la regla y por lo tanto la influencia que tendrá el consecuente en el valor de salida del sistema.

El número de reglas está en función de las variables de entrada (m) y de los conjuntos difusos definidos (n), la relación es m^n [4].

Las operaciones de los conjuntos difusos definidos en el antecedente se realizan mediante los operadores lógicos difusos que son en esencia los mismos que los booleanos pero con una definición de aplicación más generalizada:

AND. Busca la intersección de dos conjuntos, esta intersección está definida de tres formas

Algebraico. Multiplicación de los valores

Mínimo

Máximo

OR. Realiza la unión de dos conjuntos,

Probabilístico.

Mínimo

Máximo

NOT. Devuelve el complemento del conjunto

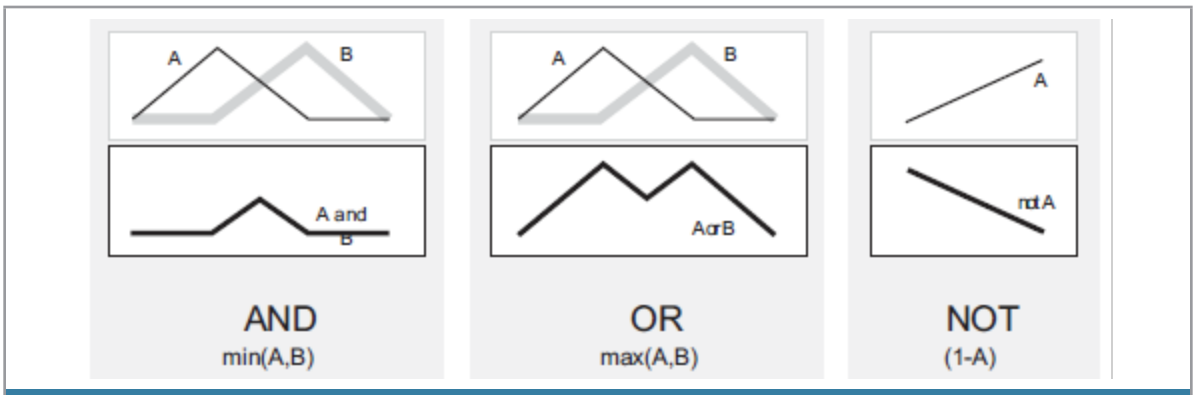


Fig. 3.4 Operadores lógicos difusos [3]

El consecuente está compuesto por acciones que se realizan si se cumplen las condiciones del antecedente. La implementación del consecuente depende del tipo de la máquina de inferencia; hablemos pues del tipo Takagi-Sugeno. Sugeno establece usar funciones que dependen de las variables de entrada, estas funciones pueden ser lineales o no lineales con la singularidad de poder usar en vez de funciones, un valor para cada regla; estos valores se conocen como *singletons* y su dominio está sobre el dominio de la variable de salida.

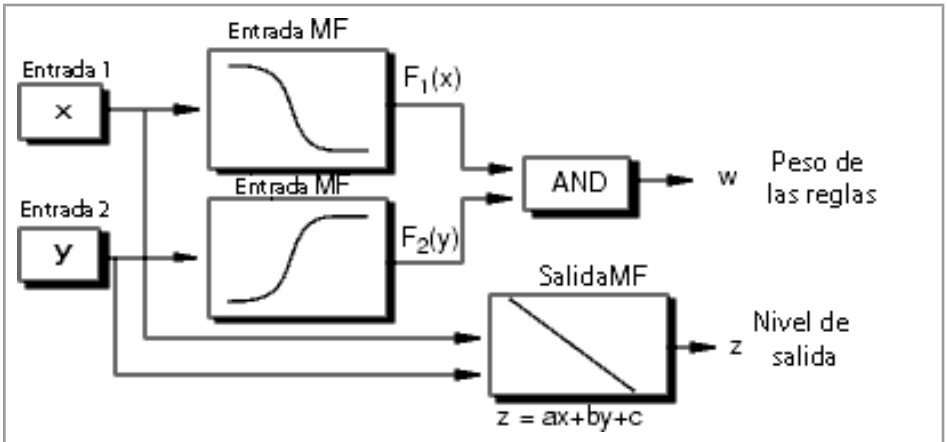


Fig. 3.5 Sistema Takagi-Sugeno [3]

En este tipo de sistemas no existe la desfuzificación porque los *singletons* tienen valores determinados, así que la etapa de agregación solo se realiza mediante el promedio ponderado de las reglas con los *singletons*.

$$salida = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \quad \begin{array}{l} donde\ w_i\ es\ la\ evaluación\ de\ la\ regla \\ z_i\ es\ el\ valor\ del\ singleton \end{array}$$

Ecuación 3.7

Por lo tanto el número de *singletons* es igual al número de reglas definidas.

El uso de *singletons* es muy conveniente para implementar un sistema difuso en un microcontrolador porque no se requiere hacer un procesamiento fuerte.

Redes Neuronales Artificiales (RNA)

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

	Computadora	RNA
Procesador	- Complejo - Alta Velocidad - Uno o pocos	- Simple - Baja velocidad - Un gran numero
Memoria	- Separada del procesador - No es direccionable por el contenido	- Integrada en el procesador - Direccionable por el contenido
Computación	- Centralizado - Secuencial - Programas Almacenados	-Distribuido paralelo - Auto-aprendizaje
Confiabilidad	- Muy vulnerable	- Robusta
Punto fuerte	- Manipulaciones numéricas y simbólicas	- Problemas de percepción

	Computadora	RNA
Ambiente operativo	- Bien definido - Bien limitado	- Pobremente definido - Ilimitado

Tabla 3.2 Computadora Von Neumann vs Sistema Neuronal

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por tres funciones [5]:

1.- Una función de propagación, que por lo general consiste en la suma de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina excitatoria; si es negativo, se denomina inhibitoria.

2.- Una función de activación, que modifica a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.

3.- Una función de transferencia, que se aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son la función sigmoidal (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]).

Ventajas

- Aprendizaje adaptativo. Capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
- Auto organización. Puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
- Tolerancia a fallas. La destrucción parcial de una red conduce a una degradación de su estructura; pero, algunas capacidades de la red se pueden retener.
- Flexibilidad.
- Tiempo real. Se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

Clasificación

Las redes neuronales se clasifican en los siguientes aspectos:

1. Topología
    - Capas
    - Conexiones
2. Mecanismo de aprendizaje
    - Supervisado
        - Corrección de error
        - Por refuerzo
        - Estocástico
    - No supervisado
        - Hebbiano
        - Competitivo
3. Asociación y representación de la información
    - Heteroasociativas
    - Autoasociativas
4. Constitución del las señales de entrada y salida
    - Discreta
    - Continua

Esta clasificación nos indica que son propiedades inherentes de una RNA. La topología de una red consiste en la organización y disposición de las neuronas formando capas, los parámetros fundamentales son el número de capas, número de neuronas por capa, grado y tipo de conexiones entre neuronas.

Es importante destacar que a mayor número de neuronas se tiene más posibilidades de encontrar una solución, en particular, una red con una capa solo puede resolver problemas que sean linealmente separables y que las redes con más de cuatro capas generan ruido en el procesamiento.

El aprendizaje es el proceso por el cual la red modifica los pesos para que la salida esté en función de la información de entrada. La regla de aprendizaje es un algoritmo que describe como realizar estas modificaciones. Esta regla puede ser supervisada o no supervisada.

En el caso supervisado existe un agente externo a la red que controla su desempeño de aprendizaje y le indica cuales son las modificaciones. Una forma de hacerlo es ajustar los pesos de los vectores de entrada en función del error que hay entre los valores deseados y obtenidos de la red.

Otra opción es solo indicar si la salida es correcta con una señal de refuerzo, esto es útil cuando no se tiene el patrón completo deseado pero es más lento que el aprendizaje por corrección del error.

Para ambos casos, la red tiene que realizar el proceso de generalización, dada una entrada que no fue parte del entrenamiento, la red pueda hacer una generalización en la salida aproximando la entrada hacia una categoría.

La información que va aprendiendo la red se registra de forma distribuida en los pesos asociados a las conexiones entre neuronas, la forma de recuperar esta información se hace mediante una hetero-asociación donde la red ha aprendido a identificar parejas de entrada ->salida; para este tipo de redes es preciso que haya como mínimo dos capas, una para captar y retener la información y otra para mantener la salida.

Las redes autoasociativas aprenden cierta información de tal forma que cuando se presente un dato de entrada realizará una autocorrelación, respondiendo con el dato almacenado más parecido al de la entrada.

Por último, las salidas pueden presentarse en forma continua o discreta, es continua cuando tenemos como función de activación la función sigmoidal, lineal o tangente hiperbólica y es discreta cuando es la función escalón.

La red de retropropagación

Es una RNA formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables como el reconocimiento de patrones. Esta red puede ser total o localmente conectada. En el primer caso cada salida de una neurona de la capa "i" es la entrada de todas las neuronas de la capa "i+1", mientras que el segundo, cada neurona de la capa "i" es la entrada de una serie de neuronas (región) de la capa "i+1".

Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

El entrenamiento de esta red es realizado por el algoritmo delta generalizada que utiliza una función de error asociada a la red, buscando el estado estable de mínima energía a través del camino descendente de la superficie del error. La modificación de los pesos se basa en el valor proporcional al gradiente decreciente de dicha función de error [1].

$$w_{ji}(t+1) = w_{ji}(t) + \left[\Delta w_{ji}(t+1) \right]$$
$$w_{ji}(t+1) = w_{ji}(t) + \left[\alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t) \right]$$

Ecuación 3.8

En la ecuación 3.8:

$\delta_{pj} = (d_{pj} - y_{pj}) f'(net_j)$ si U_j es una neurona de salida y

Ecuación 3.9

$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) f'(net_j)$ si U_j no es una neurona de salida

Ecuación 3.10

w es la matriz de pesos

α es el factor de aprendizaje, e indica que tan rápido queremos que cambien los valores de los pesos. Está acotado de cero a uno. Es remarcable que con un factor de aprendizaje alto podemos llegar a un mínimo local de la función ó no converger al error mínimo. En el caso contrario los pesos se modifican ligeramente haciendo que la convergencia sea lenta

β es la cantidad de momento el cual evita oscilaciones cuando el factor de aprendizaje es alto. También hace que el número de iteraciones se reduzca ya que toma en cuenta el gradiente actual haciendo que aumente la velocidad de cambio de pesos cuando el gradiente actual es del mismo signo que el nuevo cambio de pesos o que disminuya si es que son de signo contrario

j representa la neurona de la capa de actual.

i representa la neurona de la capa de anterior a la actual

p es cada elemento del vector de entradas

k es la neurona de alguna capa oculta

El proceso de aprendizaje se desarrolla en dos fases, primero, se presenta un patrón en la entrada de la red y propagando a través de las capas hasta llegar a la capa de salida.

Una vez obtenidos los valores de salida se inicia la segunda fase, comparando estos valores con la salida esperada para obtener el error.

Las funciones de transferencia de los elementos de procesado (neuronas) han de ser derivables. La ecuación 3.11 es la derivada de la función de transferencia.

$$f'(net_j)$$

Ecuación 3.11

Las más comunes son la lineal (ecuación 3.12) y la sigmoidal logística (ecuación 3.13).

$$f'(net_j) = net_j \qquad f'(net_j) = 1$$

Ecuación 3.12

$$f'(net_j) = \frac{1}{1+e^{-net_j}} \qquad f'(net_j) = f(net_j)\left(1 - f(net_j)\right)$$

Ecuación 3.13

Estas funciones son de las formas mostradas en las figuras 3.8 a 3.10.

- *Sigmoide logística*

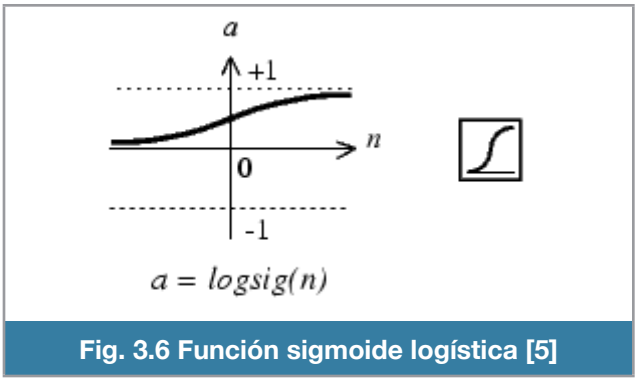


Fig. 3.6 Función sigmoide logística [5]

- *Sigmoide tangencial*

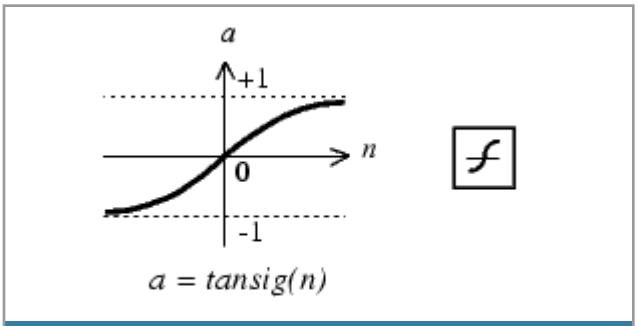


Fig. 3.7 Función sigmoide tangencial [5]

- *Lineal*

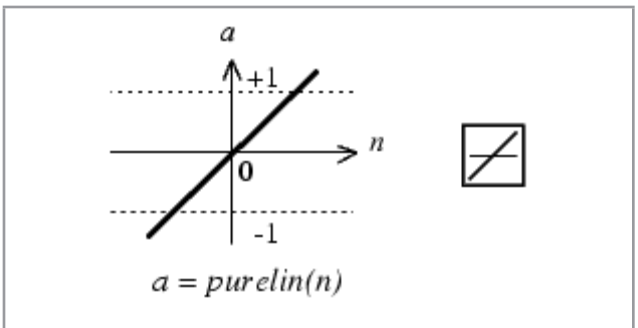


Fig. 3.8 Función lineal [5]

La representación de una capa de red neuronal de retropropagación se aprecia en la figura 3.14.

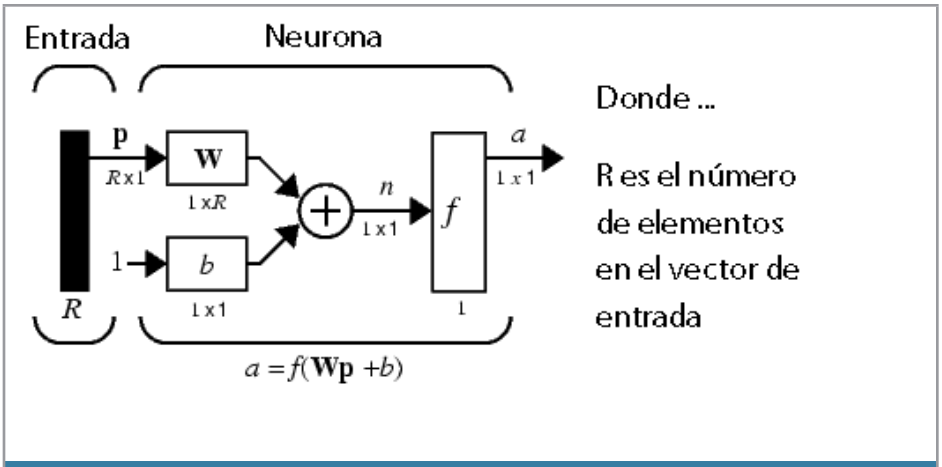


Fig. 3.9 Representación de una capa de RN backpropagation [5]

Y esta unidad básica de diseño se puede expandir a muchas capas de varias entradas con distinto número de salidas. En la práctica solo hay mejoras notables con redes de hasta cuatro capas.

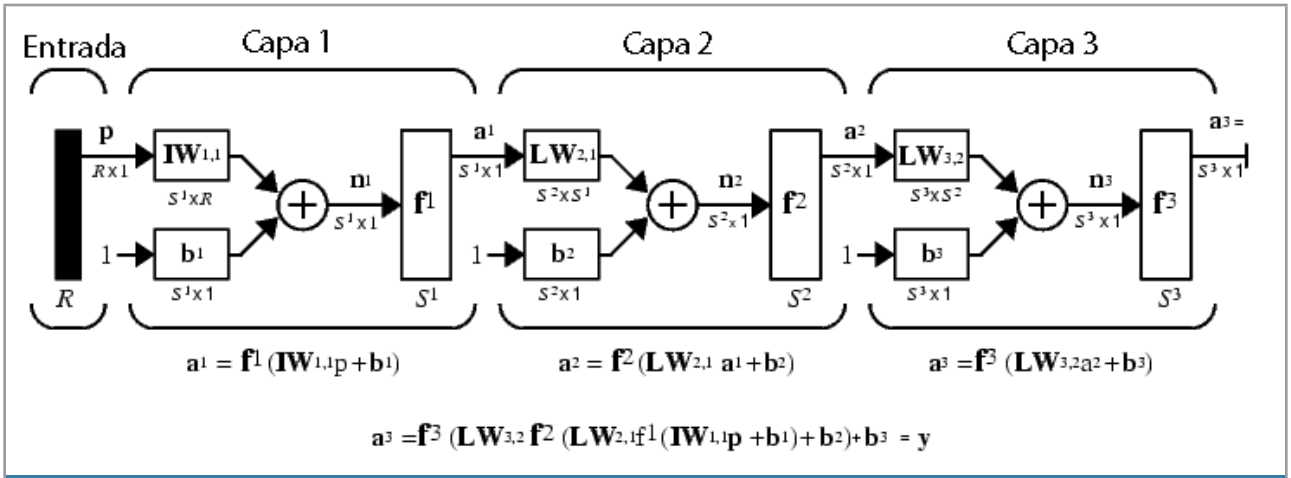


Fig. 3.10 Arquitectura de la red neuronal [5]

La red utiliza una matriz de pesos que afecta a las entradas y que a su vez tiene un elemento de umbral lo que permite que se desplace sobre el eje ordenado. La información es introducida en una función de transferencia y el resultado se interpreta o se usa como las entradas para una siguiente capa.

El modo de cómo se entrena la red neuronal varía dentro de estos dos ámbitos

- Regresión de datos
- Reconocimiento de patrones

Es importante para este tipo de red neuronal tener una buena base con patrones que permitan hacer una generalización a la neurona.

Diseño del sistema neuro - difuso en Matlab

El uso de *Matrix Laboratory* (*MatLab*) nos provee una gran gama de herramientas para poder realizar la selección, análisis y procesamiento de la información que obtuvimos previamente con *LabView*. Nos facilita el cálculo de los parámetros del sistema neuro-difuso y despliegue de la simulación del sistema.

El trabajo se divide en tres secciones, la primera sección tiene por finalidad captar el archivo de datos generado por LabView identificando la siguiente información:

- Tiempo de simulación
- Ángulo sobre el eje *roll*
- Ángulo sobre el eje *pitch*

- Señal comandada al servomotor del cíclico lateral
- Señal comandada al servomotor del cíclico longitudinal

Estos datos deben dividirse a su vez en tres bloques de datos, el primer bloque se usa para el entrenamiento de la red neuronal, el segundo bloque para la validación del entrenamiento y el tercero para probar mediante simulación.

La división de estos datos se realiza de forma aleatoria conservando la siguiente proporción:

- Entrenamiento: 60%
- Validación: 20%
- Prueba: 20%

Además, se grafican los ángulos *roll*, *pitch* y las señales comandadas a los servomotores de cíclico lateral y longitudinal. Esta gráfica se almacena en el disco duro para su referencia posterior.

La segunda sección realiza el procesamiento de los datos. Matlab ofrece una herramienta llamada *ANFIS* (*adaptive neuro-fuzzy inference system*) el cual ajusta los parámetros de las funciones de membresía de un FIS (*fuzzy inference system*) usando una RNA con entrenamiento de retropropagación. Se muestra un ejemplo de código de generación del sistema neuro-difuso en la figura 3.11.

```
----------//Generacion sistema neuro-difuso//----------
anfis2e1slat=genfis1(vtrla,nummf,tipomf,'constant');
anfis2e1slon=genfis1(vtrlo,nummf,tipomf,'constant');
[maqlat elat st chklat
chkelat]=anfis(vtrla,anfis2e1slat,epoch,[],vtela);
[maqlon elon st chklon
chkelon]=anfis(vtrlo,anfis2e1slon,epoch,[],vtelo);
----------//-----------------//----------
```

Fig. 3.11 Código de generación del sistema neuro-difuso

Para usar la herramienta es necesario primero crear un modelo *fis* tipo *Takagi-Sugeno* con salida a *singleton* el cual debe definir el número de entradas, una sola salida, el número y tipo de funciones de membresía a la entrada. Las reglas del sistema se generan mediante la combinación de la funciones de membresía de todas las entradas.

En nuestro caso definimos dos sistemas neuro difusos, uno para cada salida debido a que *anfis* impone solo una salida por sistema. Para ambos sistemas tienen las mismas entradas que son los ángulos de roll y pitch. Para cada entrada existen tres funciones de membresía las cuales no tienen alguna representación ya que la red neuronal establece los parámetros de las funciones en base a los datos de entrenamiento.

Las reglas del sistema son generadas en función del número de entradas y el número de funciones de membresía que tiene cada entrada, entonces, si tenemos dos entradas y tres funciones de membresía por entrada nos da como resultado 3^2 reglas y por lo tanto 9 *singletons* también.

La decisión de tomar tres funciones de membresía por variable de entrada es debida a que no hay un mejor rendimiento notable y porque en la codificación en el microcontrolador hace que crezca el número de operaciones de forma no lineal.

Hay varias formas de generar un sistema difuso en Matlab, la primera es mediante el entorno gráfico el cual no nos sirve porque no estamos asignando los parámetros; la otra forma es mediante la línea de comando.

La función *genfis1* nos permite generar la estructura esencial del sistema difuso tipo sugeno, pasamos la matriz de datos entrada – salida, indicamos el número de funciones de membresía por entrada, la forma de las funciones de membresía y el tipo de salida que es *singleton*.

La red neuronal artificial es creada a través de *anfis* usando como base el sistema difuso creado anteriormente, adicionalmente se especifica el bloque de datos para usarse como entrenamiento, y el bloque de datos para validación. Opcionalmente especificamos el número de iteraciones para el entrenamiento.

También con *anfis* se inicia el entrenamiento de la red neuronal, consiste en adaptar los pesos de cada capa de la red neuronal.

La conclusión del entrenamiento se hace cuando suceden los siguientes eventos:

- El parámetro de tolerancia al error es mayor que el error del sistema
- Se ha llegado al máximo número de iteraciones
- Se ha alcanzado el mínimo valor del cambio de gradiente (retropropagación)
- Se ha llegado al máximo número de validaciones inválidas

El tercer archivo nos muestra una simulación del sistema neuro difuso mediante la función *evalfis*. Esta función evalúa a los sistemas difusos con los bloques de datos de pruebas. Desplegamos una gráfica con los resultados de la evaluación y la comparación con la información de entrada.

Los parámetros ajustados de los sistemas difusos ajustados por la red neuronal son guardados en un archivo de texto plano para su posterior utilización en el microcontrolador. Los datos que se requieren son los siguientes:

- Singletons del servomotor para el cíclico lateral y longitudinal
- Los parámetros de las funciones de membresía (triangulares) para los ángulos roll y pitch.

En las figuras 3.12, 3.13 y 3.14 se muestra al sistema difuso antes de ser ajustado mediante *anfis*. Vemos que la función *genfis1* nos genera la estructura de dos variables de entrada por una variable de salida utilizando takagi-sugeno como máquina de inferencia. Por defecto las funciones de membresía para la variable roll son configuradas equitativamente a lo largo del universo del discurso; de igual forma sucede con pitch.

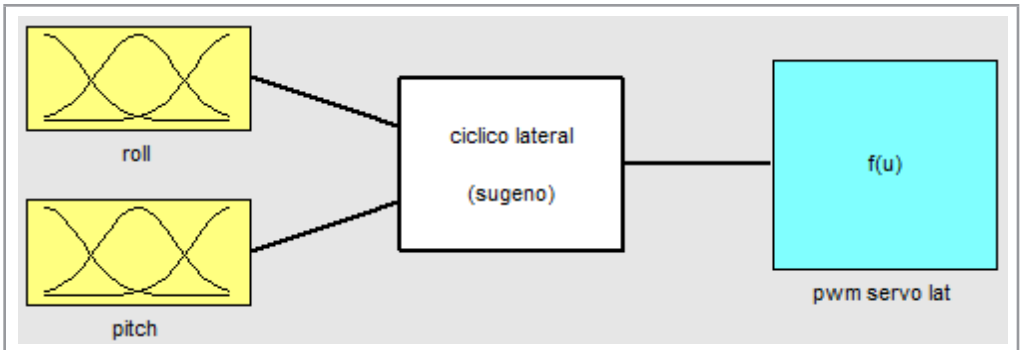


Fig. 3.12 Modelo del controlador difuso para el control lateral

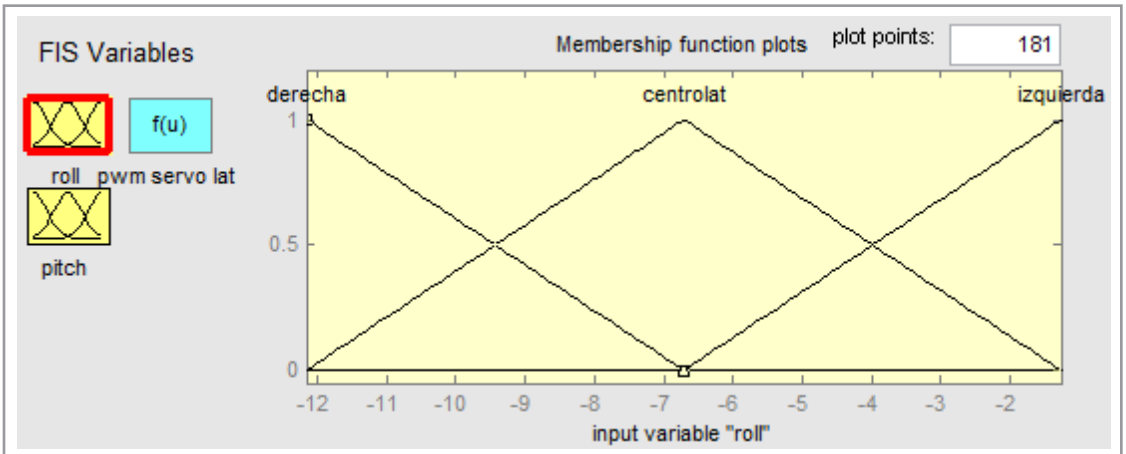


Fig. 3.13 Funciones de membresía para la entrada Roll del control lateral

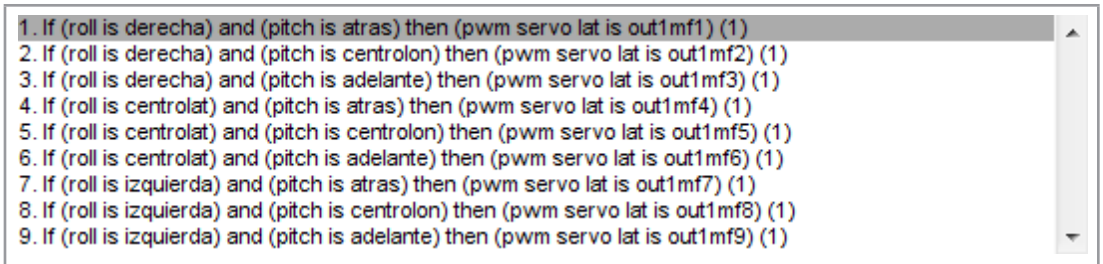


Fig. 3.14 Reglas del control lateral

Las reglas para el controlador son formados como una combinación del número de las variables de entradas con las funciones de membresía que tienen. Ambas variables tienen el mismo número de funciones de membresía. En la variable de salida el número de *singletons* es igual al número de reglas existentes, pero estos aún no tienen una representación conceptual del sistema.

Las figuras 3.15 y 3.16 despliegan las funciones de membresía para ambas entradas, los parámetros de cada función han sido optimizados mediante el resultado del entrenamiento de la red neuronal en base a los datos de vuelo obtenidos previamente.

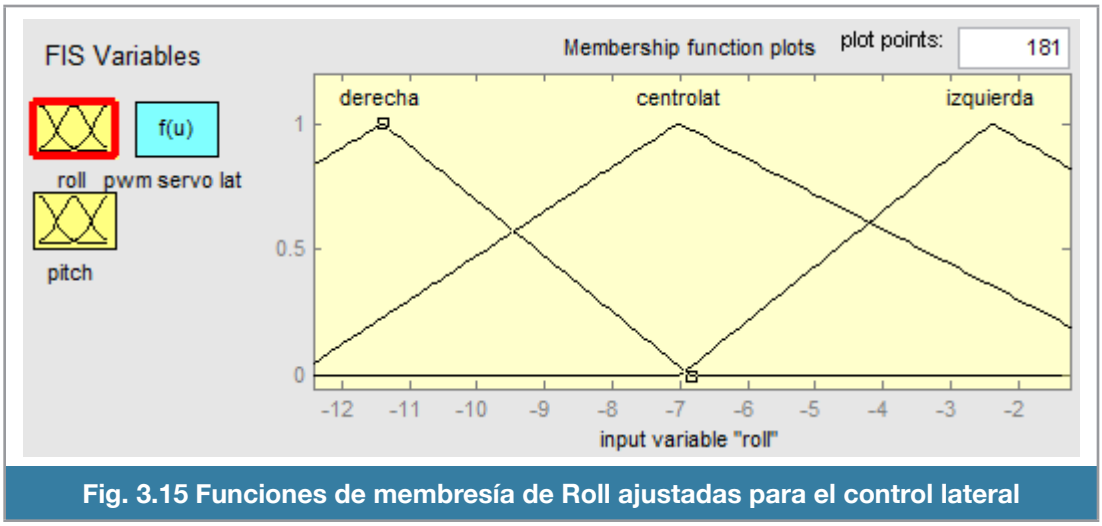


Fig. 3.15 Funciones de membresía de Roll ajustadas para el control lateral

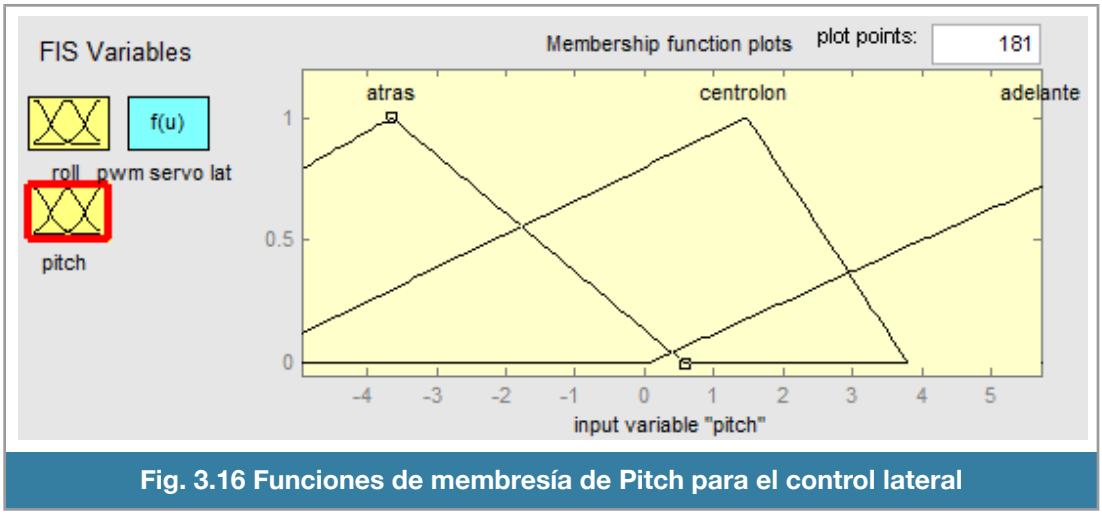


Fig. 3.16 Funciones de membresía de Pitch para el control lateral

La figura 3.17 nos muestra la superficie de control, en la que se observa como cambia el valor de la variable de la salida en función de los ángulos roll y pitch.

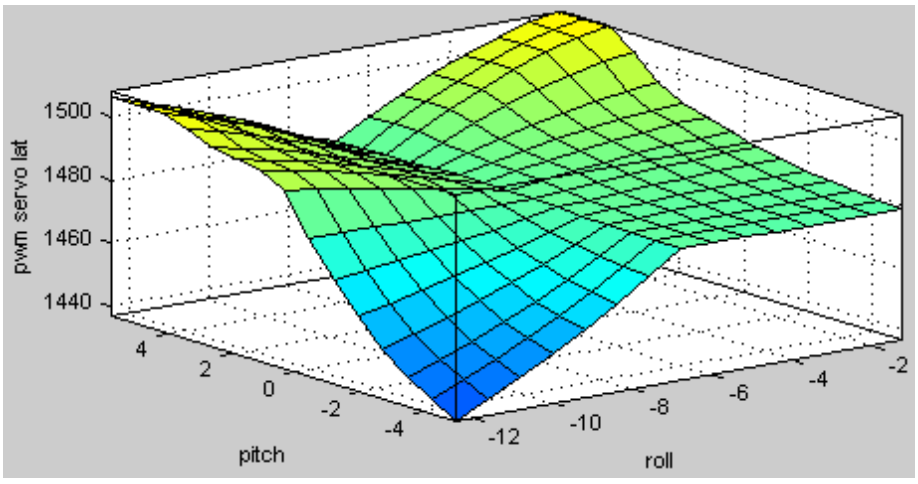


Fig. 3.17 Superficie de control para el servo lateral

En las figuras 3.18, 3.19 y 3.20 apreciamos lo consecuente para el control longitudinal.

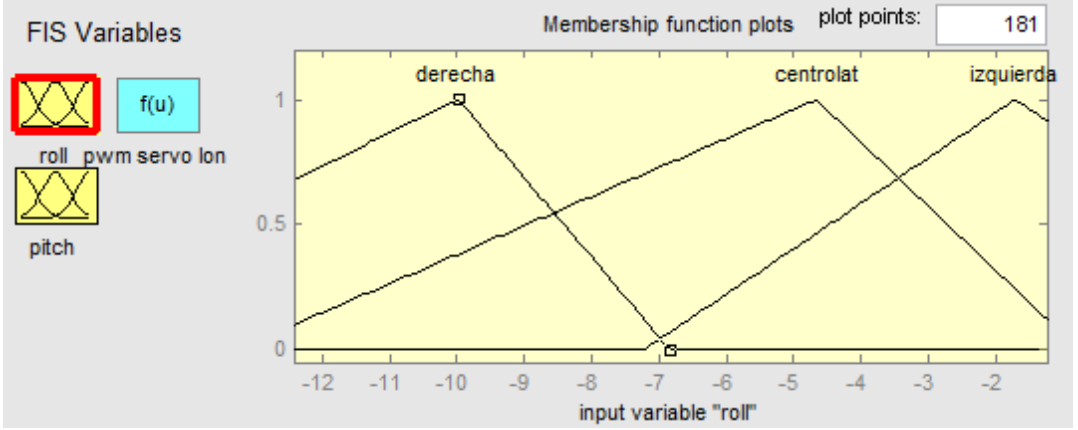


Fig. 3.18 Funciones de membresía de Roll ajustadas para el control logitudinal

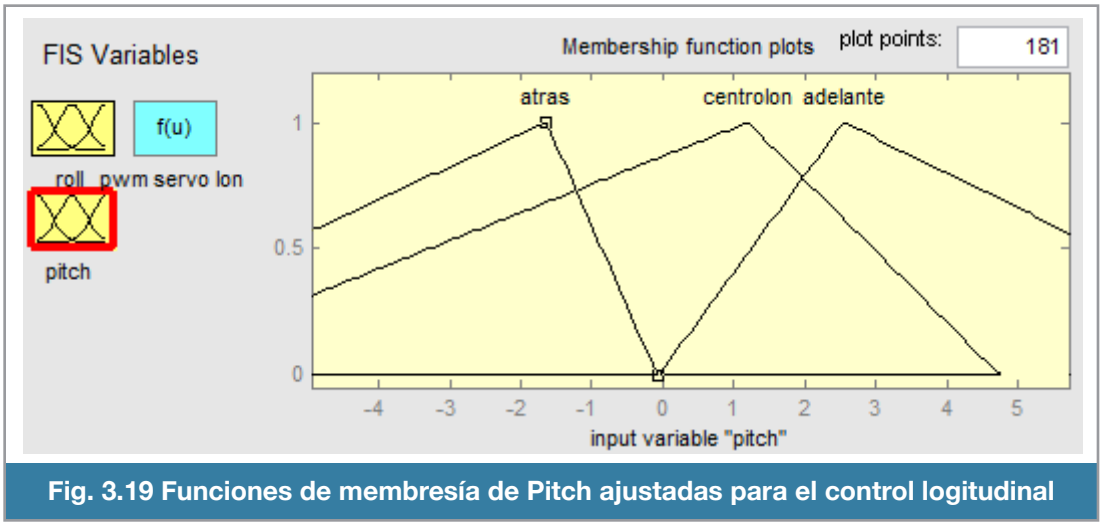


Fig. 3.19 Funciones de membresía de Pitch ajustadas para el control logitudinal

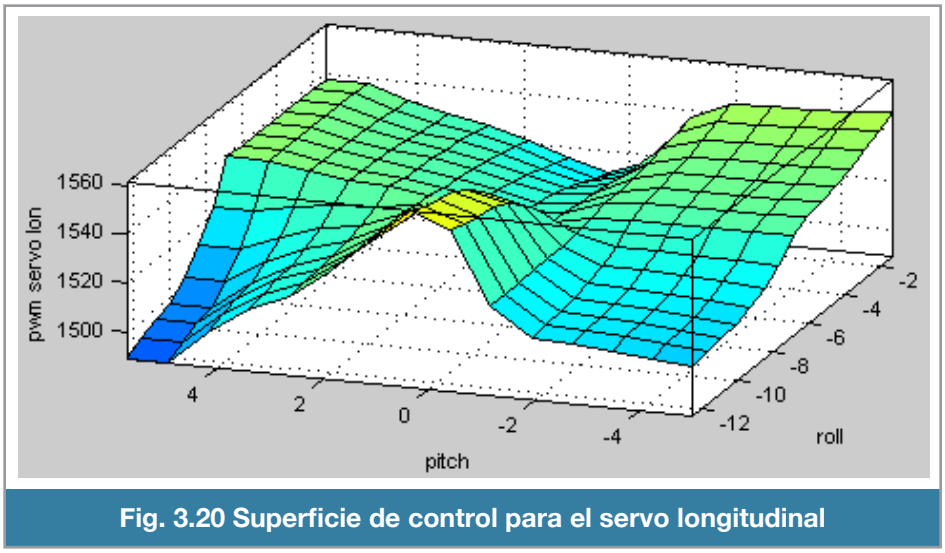


Fig. 3.20 Superficie de control para el servo longitudinal

Desarrollo del sistema de control difuso

El software del control implementado en el microcontrolador *ATMEGA328* de la plataforma *Arduino* realiza las siguientes actividades:

- Comprobar el estado del interruptor que selecciona el modo manual y modo automático
- Si el modo es manual, simplemente escribe los datos previamente leídos del canal PPM a los servomotores; si el modo es automático, calculará los pulsos de control apropiados para estabilizar el helicóptero

Se muestra el diagrama de flujo del programa de control en la figura 3.21.

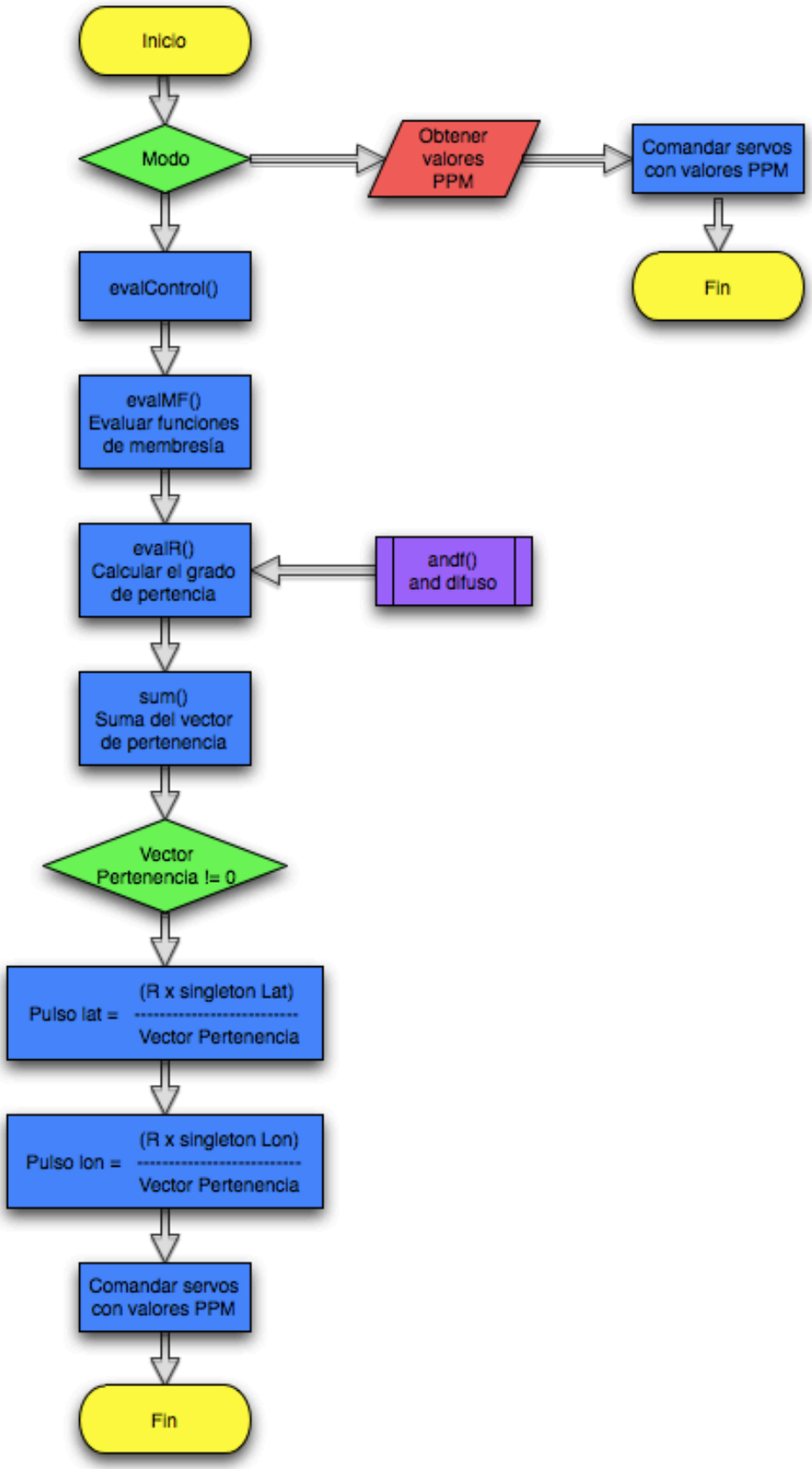


Fig. 3.21 Diagrama de bloques del software de control

En el código de la figura 3.22 se puede apreciar que la función *evalControl* manda llamar a las funciones *evalMF* y *evalR*, además de hacer el calculo final del pulso lateral y del pulso longitudinal; para estos cálculos se efectúa la suma del vector R (vector de reglas que contienen el grado de pertenencia), se comprueba que tal suma no sea cero para evitar problemas de cálculo con el microcontrolador; se hace una multiplicación de vectores entre R y el vector de los *singletons* (uno corresponde al lateral y el otro al longitudinal) y esta multiplicación se divide entre la suma de R.

```
----------//Evaluar Control Difuso//----------
void evalControl(){
  float sumRlat;
  float sumRlon;

  evalMF();
  evalR();

  sumRlat=sum(rlat);
  sumRlon=sum(rlon);

  if(sumRlat!=0){
    lat= (int)(mul(rlat,singlelat) / sumRlat);
  }
  if(sumRlon!=0){
    lon= (int)(mul(rlon,singlelon) / sumRlon);
  }
}
----------//------------------//----------
```

Fig. 3.22 Código de evaluación de control difuso

Un ejemplo de un vector *singleton* se aprecia en la figura 3.23.

```
float singlelat[] = {1370.80, 1426.81, 1556.36, 1453.91, 1452.87, 1415.95, 1312.26,
1423.49, 1497.57};
```

Fig. 3.23 Ejemplo de vector singleton

Cabe resaltar que estos valores son generados por el software de Matlab, mediante el sistema neuronal.

Cada renglón del arreglo *mfroll* en la figura 3.24 corresponde a una función triangular, en donde se define un punto mínimo, un central y un máximo.

```
float mfroll[] = {-39.38, -18.40, -11.87,
                  -26.37, -9.24 ,  16.9 ,
                  -9.88 , -6.94 ,  20.59 };
```

Fig. 3.24 Ejemplo del vector de las funciones de membresía

En la figura 3.25 se muestra la función *evalMF* que es la encargada de evaluar las funciones de membresía que nos genera el sistema neuronal.

```
----------//Evaluar Funciones de membresía//----------
void evalMF(){
  int j=0;
  for(int i=0;i<numM;i++){
    mfvalues[i]=mft(ToDeg(xkaldata.angle),mfroll[j],mfroll[j+1],mfroll[j+2]);
    mfvalues[i+numM]=mft(ToDeg(ykaldata.angle),mfpitch[j],mfpitch[j+1],mfpitch[j+2]);
    j+=3;
    if(j==numM*3)
      j=0;
  }
}
----------//------------------//----------
```

Fig. 3.25 Código de evaluación de las funciones de membresía

La función *mft* que se muestra en la figura 3.26 es la que se encarga de evaluar y entregar los datos de cada función triangular que se utiliza en las funciones de membresía.

```
----------//Evaluación de las funciones triangulares//----------
float mft(float x,float xmin, float xc, float xmax){
  return max(min((x-xmin)/(xc-xmin),(xmax-x)/(xmax-xc)),0);
}
----------//------------------//----------
```

Fig. 3.26 Código de la función de membresía triangular

La función *evalR* que se aprecia en la figura 3.27 se encarga de calcular el grado de pertenencia de cada función de membresía mediante el uso de una función llamada *andf* (figura 3.28) que

corresponde al *and* difuso, el cual hemos programado los tres tipos de *and* difusos, el de mínimo, producto y el de máximo. El *and* producto es el que se utiliza.

```
----------//Grado de pertenencia de las funciones de membresía//----------
void evalR(){
int k=0;

  for(int i=0;i<numM;i++){
       rlat[i]=mfvalues[i];
       rlon[i]=mfvalues[i+numM];
  }
}
----------//------------------//----------
```

Fig. 3.27 Código del calculo del grado de pertenencia

```
----------//And difuso//----------
float andf(float mx, float my, int opcion){
  float opf;
  switch (opcion){
    case 1: opf=min(mx,my);
    break;
    case 2: opf=mx*my;
    break;
    case 3: opf=max(0,mx+my-1);
    break;
  }
  return opf;
}
----------//------------------//----------
```

Fig. 3.28 Código de la función *and* difuso

Las operaciones vectoriales que se ocupan, no están implementadas en ninguna librería ANSI C, por lo que tuvimos que implementar la operación suma vectorial y la operación de multiplicación de vectores, se muestran en las figuras 3.29 y 3.30 respectivamente.

```
----------//Operación suma de vectores//----------
float sum(float x[]){
  int i=0;
  float resul=0;
  for(i=0;i<9;i++)
    resul+=x[i];
  return resul;
}
----------//------------------//----------
```

Fig. 3.29 Código del calculo de suma de vectores

```
----------//And difuso//----------
float mul(float x[], float y[]){
  int i=0;
  float resul=0;
  for(i=0;i<9;i++)
    resul+=x[i]*y[i];
  return resul;
}
----------//------------------//----------
```

Fig. 3.30 Código del calculo de la multiplicación vectorial

El código completo se muestra en el apéndice. Las partes más relevantes se mostraron en este capitulo.

4.- Pruebas

El sistema diseñado fue sometido a la siguiente serie de pruebas a lo largo de su desarrollo. Se verificó la parte de instrumentación asegurando que la lectura de los ángulos y de los servos fuera correcta.

Posteriormente se probó el sistema de control el cual tiene dos entradas que son los ángulos de alabeo (*roll*) y cabeceo (*pitch*); y dos salidas correspondientes a la señal de los servos de los cíclicos lateral y longitudinal. Primero se configuró el sistema neuro - difuso calibrando sus parámetros para tener la mejor respuesta. Por último se diseñaron varias estructuras de sistemas difusos e implementó en código de Arduino la estructura con el mejor resultado para que fuera evaluada por el microcontrolador.

Pruebas de instrumentación

Se realizaron varias pruebas para determinar que configuración de instrumentación era la más adecuada, se usó la interfaz para poder determinar la mejor opción.

En las pruebas se evitó que el helicóptero se elevara colocándolo en una mesa en donde se anclaba con cinchos de seguridad. El ángulo de ataque de las palas del rotor fue calibrado en el rango de 0º a 5º para evitar la autosustentación y minimizar la vibración.

Se recopilaban las muestras de los ángulos de alabeo (*roll*) y cabeceo (*pitch*) provenientes de la *IMU* conectada al Arduino, hacia la interfaz de LabView vía puerto serial en tiempo real. La prueba consistió en tener lecturas de ángulos cercanos a los 0º ya que el helicóptero estaba en una superficie plana.

Esta prueba fue un éxito solo cuando el motor principal estaba apagado. Cuando estuvo en funcionamiento, las lecturas de los sensores no fueron estables, afectando considerablemente la medición de los ángulos.

Para tal caso, se utilizó la tarjeta de adquisición USB-6009 de LabView para obtener las señales de los sensores sin ser procesadas por el microcontrolador Arduino. Las lecturas se hicieron con el motor principal en funcionamiento para leer la señal y obtener el espectro de frecuencia e identificar de forma gráfica las componentes del ruido.

Con el archivo de ejemplo "*Dual channel espectral measurement of a filter design.vi*" proporcionado por el software de Labview, observamos que el ruido está mas acentuado en los acelerómetros, radicando en la banda desde los 30hz hasta los 40hz. Para suprimir tal ruido se recurrió al uso de la técnica de alisado exponencial.

A la interfaz de *LabView* se agregó la función de almacenar la información de los ángulos de alabeo (*roll*), cabeceo(*pitch*) y las dos señales de control para los servomotores del cíclico en un archivo de texto plano para que en Matlab se haga el entrenamiento de la red neuro-difusa (fuera de línea).

Para asegurar el buen funcionamiento del sistema de instrumentación y dado que en la prueba el helicóptero se encuentra fijo en una superficie plana, los ángulos *roll* y *pitch* deben de ser cero o muy cercanos al cero.

En la figura 4.1 se aprecia que los ángulos *roll* y *pitch* tienen valores muy alejados de lo que deberían de ser, cercanos a cero. La guiñada (*yaw*) no la tomamos en consideración. La prueba se realizó a mitad de aceleración total. por la parte del software se utilizó el filtro DCM (Direction Cosine Matrix) que tiene una respuesta retrasada en comparación el filtro Kalman.

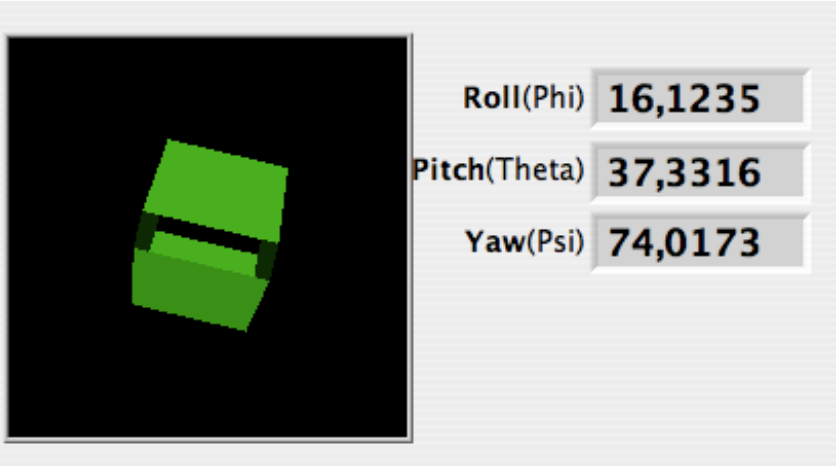


Fig. 4.1 Problemas de vibración filtro DCM sin bandas elásticas

La figura 4.2 muestra los resultados de la misma prueba con la diferencia del cambio del filtro DCM por el filtro Kalman. Se aprecia que funciona mejor que el anterior ya que el ángulo de cabeceo está más cercano a cero.

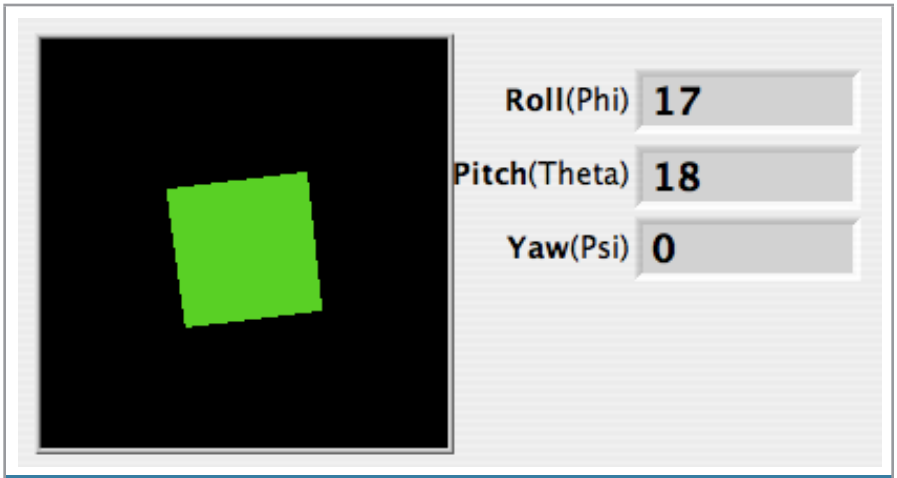


Fig. 4.2 Problemas de vibración filtro Kalman sin bandas elásticas

El filtro Kalman, a pesar de ser computacionalmente más riguroso, se comporta de mejor manera que el filtro DCM.

Los ángulos en ambos casos son erróneos, aunque en el filtro Kalman trabaja de mejor manera debido a que pierde la orientación en menor medida que el filtro DCM; además, la respuesta del filtro DCM es más lenta que la del filtro Kalman.

La figura 4.3 ilustra los resultados de la instrumentación con el filtro DCM y con la instrumentación aislada mediante bandas elásticas. Se aprecia que ha disminuido de manera considerable los errores de medición producidos por la vibración gracias al uso de las bandas elásticas.

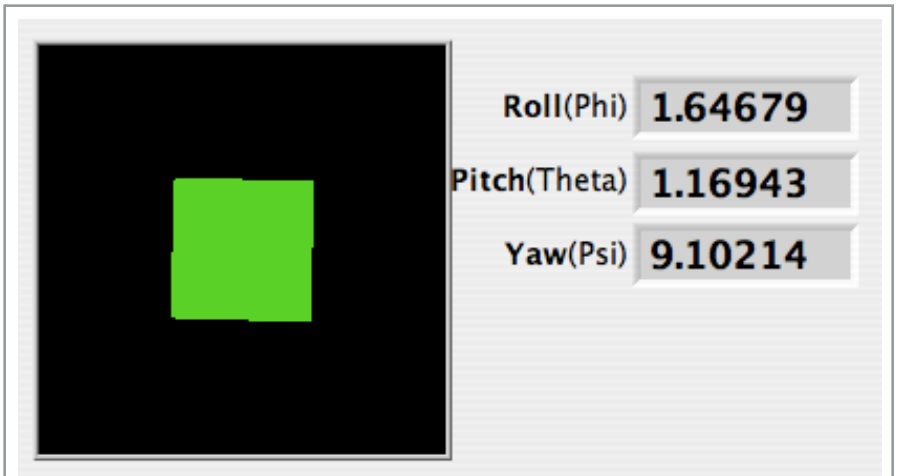


Fig. 4.3 Problemas de vibración filtro DCM y bandas elásticas

La figura 4.4 corresponde a la prueba usando el filtro Kalman, y con la instrumentación aislada mediante bandas elásticas. Se aprecia que el comportamiento de la instrumentación es el adecuado ya que, a pesar de la vibración,los ángulos se mantienen más cercanos a cero.

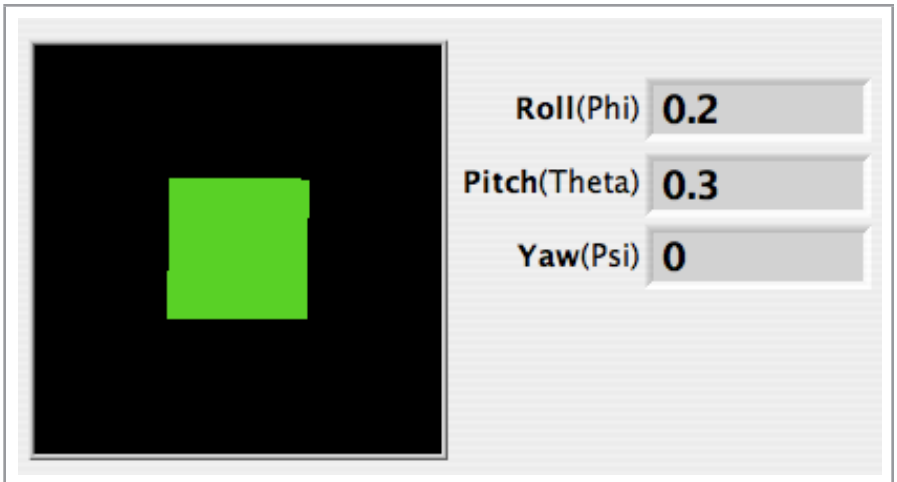


Fig. 4.4 Vibración con filtro Kalman y bandas elásticas

Al final de estas pruebas se elucidó a la configuración final de la instrumentación que fue el uso de un filtro Kalman, un alisado exponencial que tiene la función de un filtro paso bajas, instrumentación montada con velcro a una placa de estireno entre 2 bases de foam y todo montado al tren de aterrizaje mediante bandas elásticas.

Para terminar, notamos que la vibración producida tiene relación con el grado de aceleración del rotor principal. Al principio del arranque vemos que se torna muy fuerte la vibración llegando a un punto máximo a 1/4 del total de velocidad y reduciendo considerablemente cuando está después de los 3/4 de la velocidad total. La máxima velocidad de giro que alcanza el rotor está alrededor de las 2300rpm.

Pruebas de generación del sistema difuso con Matlab

En los primeros vuelos de prueba con el helicóptero instrumentado, los datos se guardaban en un archivo de texto. Estos datos eran los correspondientes al cíclico lateral y longitudinal del piloto, los ángulos alabeo (*roll*) y cabeceo (*pitch*) del helicóptero; fueron capturados mediante la HMI creada en LabView para después ser procesados en Matlab.

En la figura 4.5 se muestra un ejemplo de los datos obtenidos en las pruebas.

En la figura 4.6 se pueden apreciar gráficamente los datos obtenidos, tales como los ángulos de alabeo (*roll*), cabeceo (*pitch*), el *PWM* del cíclico lateral y el PWM del cíclico longitudinal, todos los datos son dentro del lapso de vuelo sostenido o hovering.

Las pruebas consistían en elevar al helicóptero del suelo aproximadamente unos 30 cm y mantenerlo en vuelo sostenido o *hovering*. Cuando el piloto tiene el helicóptero estable le indica al operador de la interfaz a ejecutar la captura y almacenamiento de los datos guardados.

Se procedió a realizar el entrenamiento de la red neuronal y la generación del sistema difuso y después una simulación con los mismos datos de ángulos

# Muestra	Roll	Pitch	S.Lat	S.Lon
76126	-7.770000	-4.640000	1446	1501
76297	-14.61000	-8.510000	1471	1517
76443	-18.13000	-10.02000	1467	1544
76615	-9.670000	-6.260000	1467	1544
76728	-13.07000	-5.290000	1469	1536
76900	-7.020000	-1.550000	1488	1592
77046	-7.880000	-3.200000	1482	1592
77200	-11.63000	-4.980000	1401	1454
77347	-11.72000	-4.440000	1391	1433
	Grados	Grados	uS	uS

Fig. 4.5 Ejemplo de datos capturados

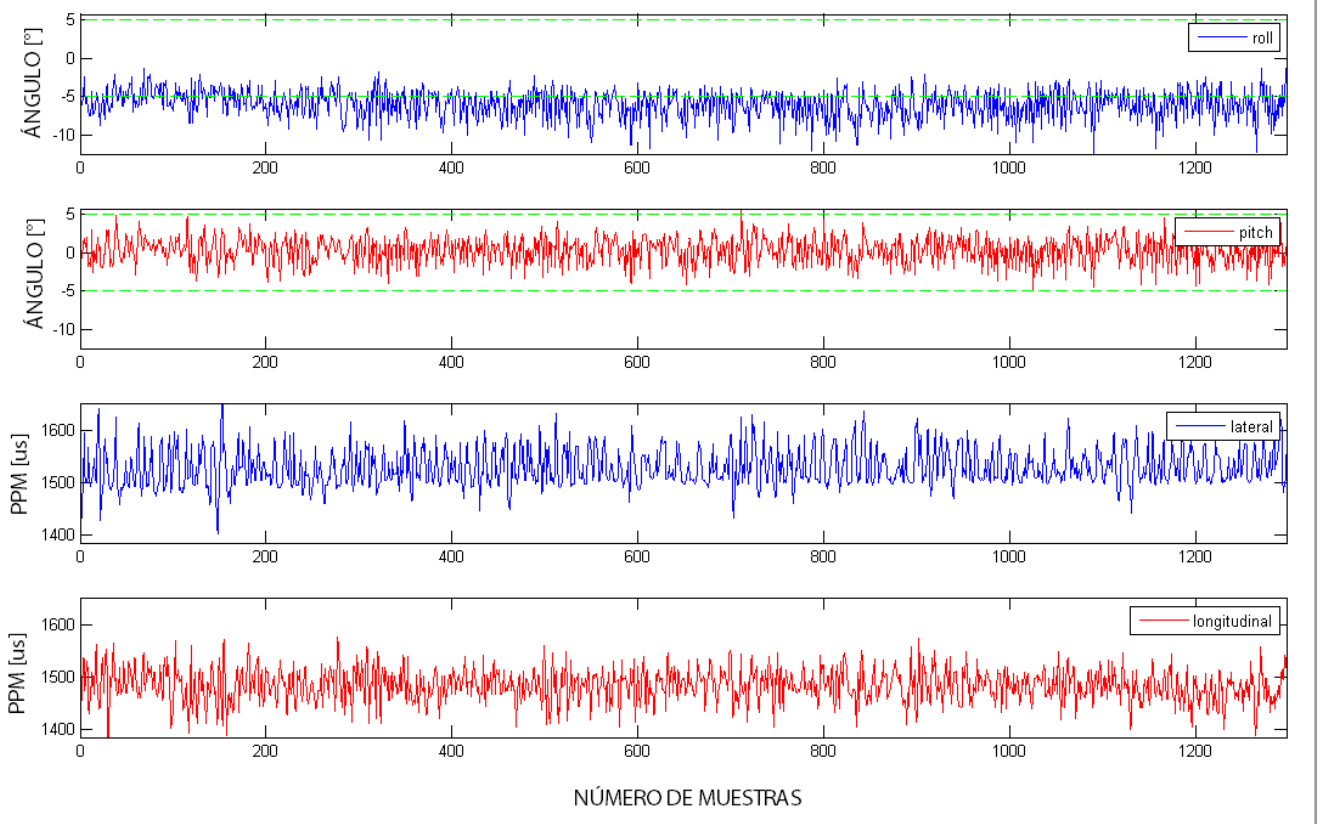


Fig. 4.6 Datos obtenidos

alabeo (*roll*) y cabeceo (*pitch*) y hacer la comparación de las acciones del piloto con las acciones del control difuso, tal como se muestra en la figura 4.7.

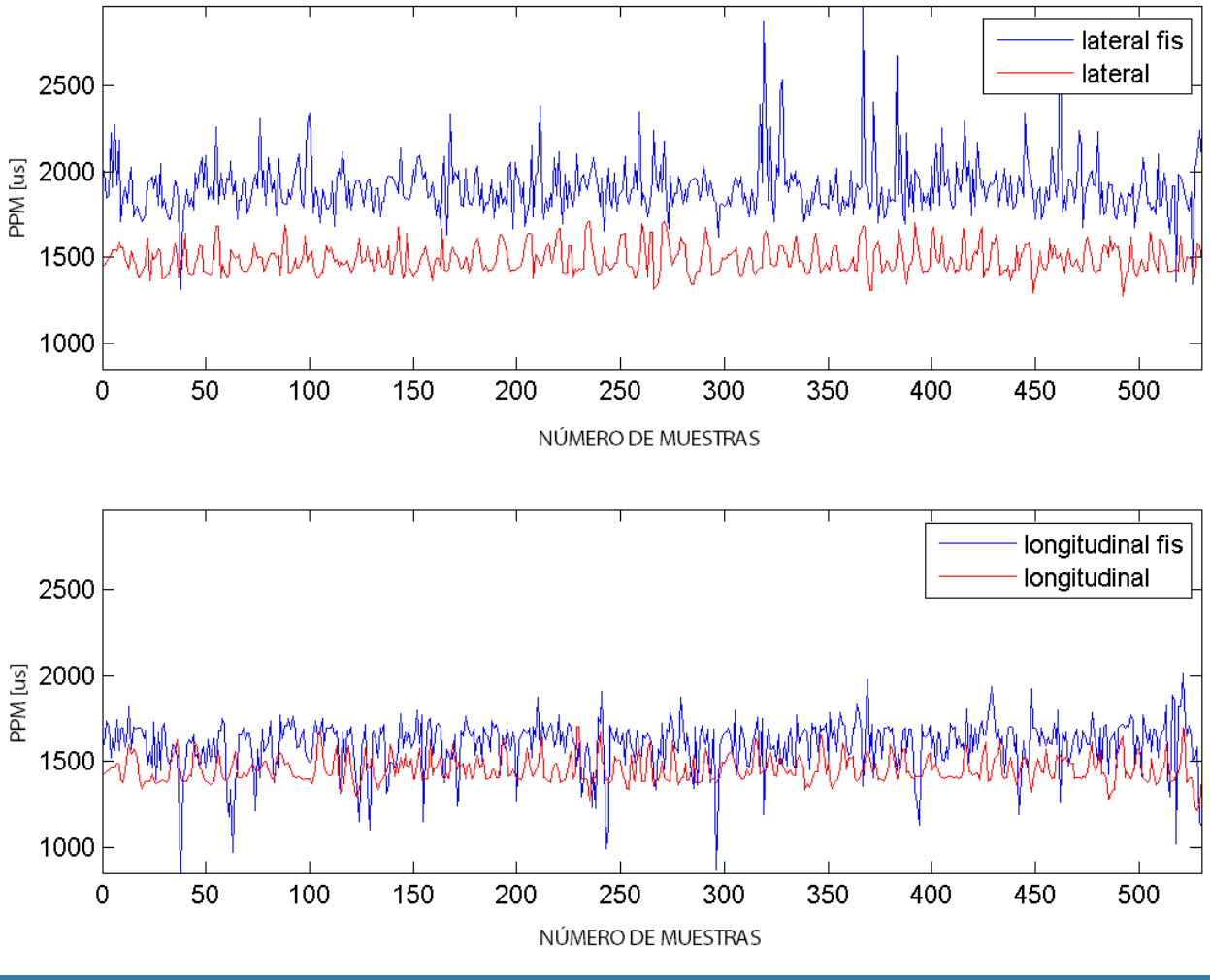


Fig. 4.7 Piloto VS Control Difuso

En la figura 4.7 se aprecia la comparación entre el sistema neuro - difuso (línea azul) y las acciones del piloto (línea roja). Como sabemos, la estabilidad se logra con las acciones del piloto, se obtiene mejor eficiencia en el control difuso si este se aproxima al comportamiento del piloto, en otras palabras, que las gráficas sean básicamente las mismas.

Se puede apreciar que el entrenamiento de la red neuronal en la prueba de la figura 4.7 no fue nada adecuado, ya que el control difuso no ajustó las acciones realizadas por el piloto para mantener estable el helicóptero.

A su vez, en el entrenamiento de la figura 4.8 se aprecia que el control difuso está "debilitado", podemos pensar en agregar una ganancia para aumentar la amplitud de la señal de control pero esto no garantiza que siga de forma adecuada el control del piloto. En ambos casos no se tiene una satisfactoria respuesta para el control que buscamos.

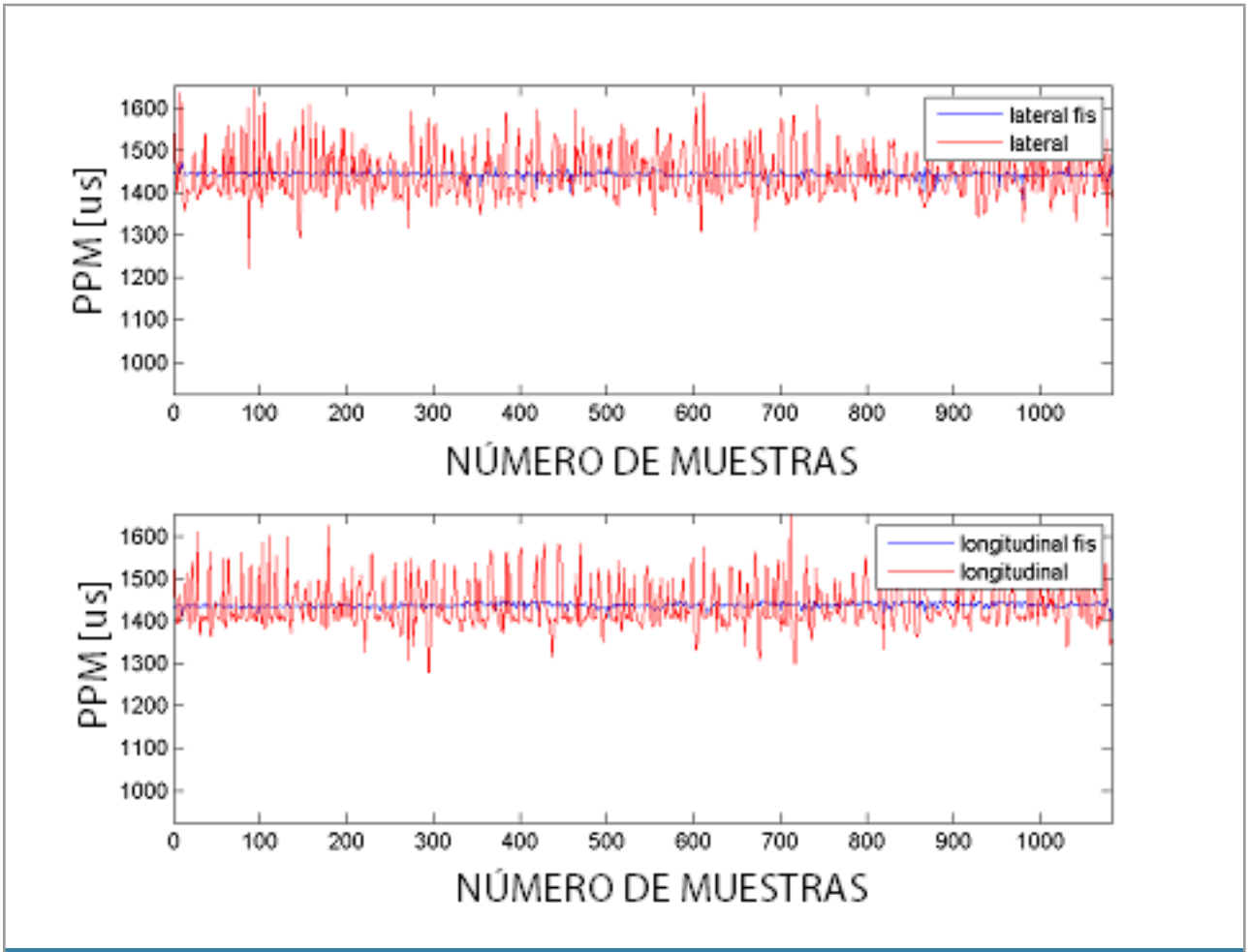


Fig. 4.8 Piloto VS Control Difuso 2

Conforme se hicieron más pruebas, se buscó el mejor ajuste de los parámetros de la red neuronal pero no se obtuvo resultados exitosos.

Control difuso de 2 entradas y 2 salidas

A raíz del pobre resultado obtenido con el sistema neuro - difuso se decidió eliminar la parte neuronal y sintonizar de forma manual el sistema difuso. Hacerlo de forma manual consiste en definir cuantas funciones de membresía tendrá cada variable de entrada, cuantos *singletons* se van a

crear por cada variable de salida y las reglas de inferencia que asocian el antecedente con el consecuente.

El diseño del sistema difuso se acotó por la implementación realizada en el microcontrolador Arduino. Se planteó generar el mismo número de funciones de membresía para cada variable de entrada así como el de *singletons* para cada variable de salida.

El primer sistema que se probó aparece en la figura 4.9. Se ilustra el diagrama general del sistema difuso que es de tipo Sugeno, cuenta con dos entradas (ángulos *roll* y *pitch*) y 2 salidas (actuador lateral y actuador longitudinal).

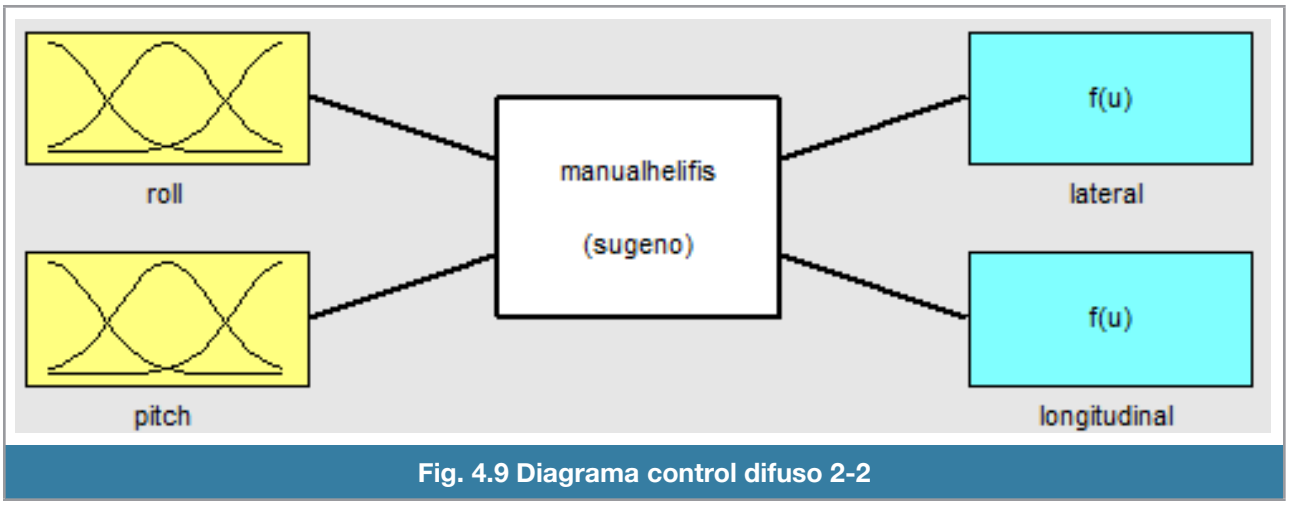


Fig. 4.9 Diagrama control difuso 2-2

Cuanta con 5 funciones de membresía por cada entrada. Ellas se aprecian en las figuras 4.10 y 4.11.

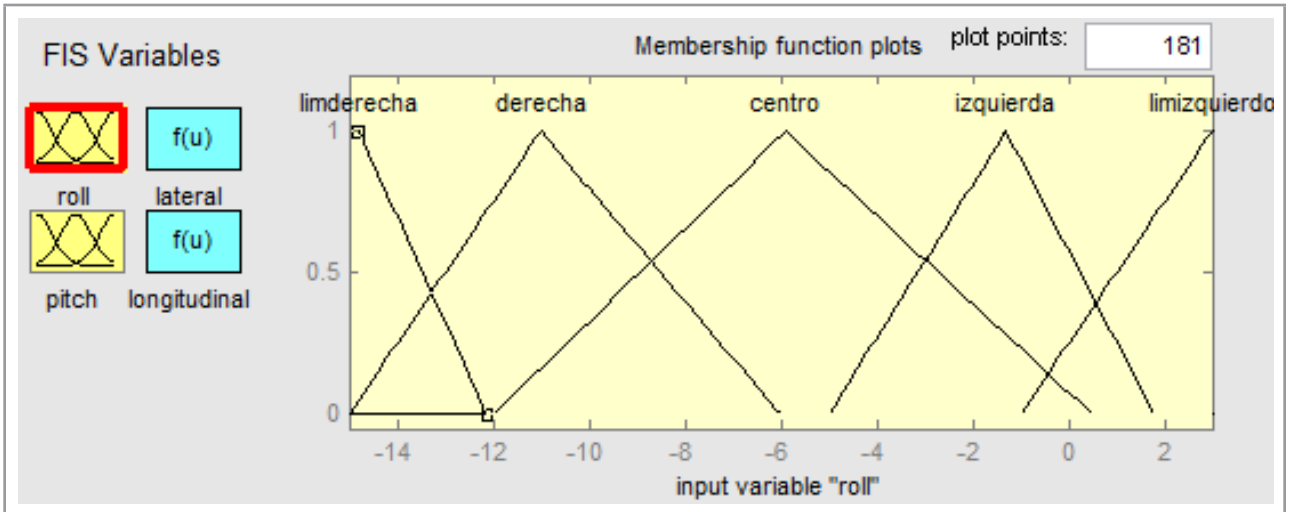


Fig. 4.10 Funciones de membresía para entrada Roll

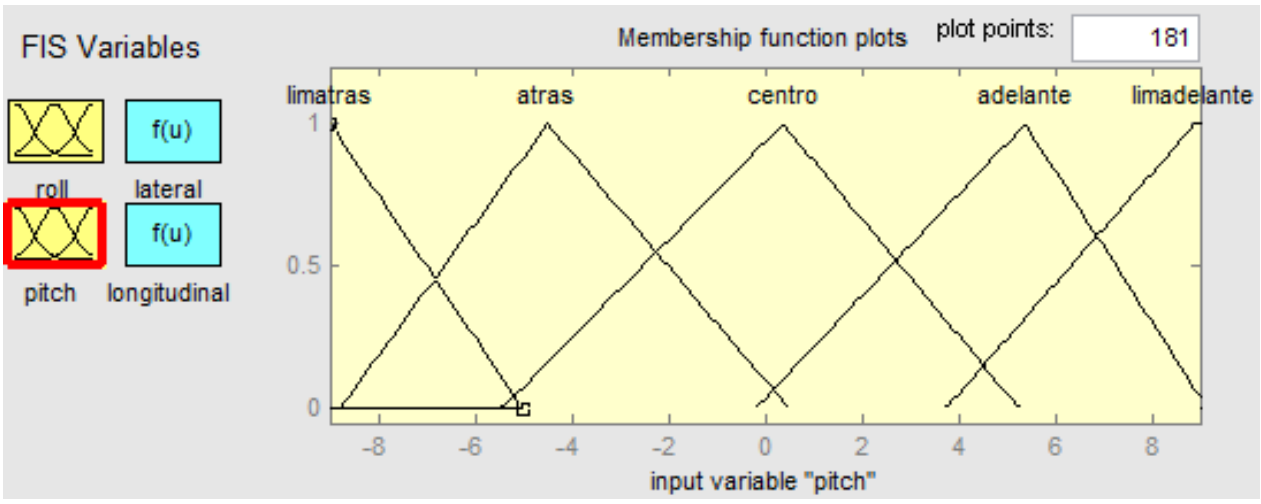


Fig. 4.11 Funciones de membresía para entrada Pitch

Las reglas estrictamente deben de ser 25 (5 X 5 funciones de membresía), pero los casos se llegan a repetir, así que quedan se reducen a 13 reglas, se muestran en la figura 4.12.

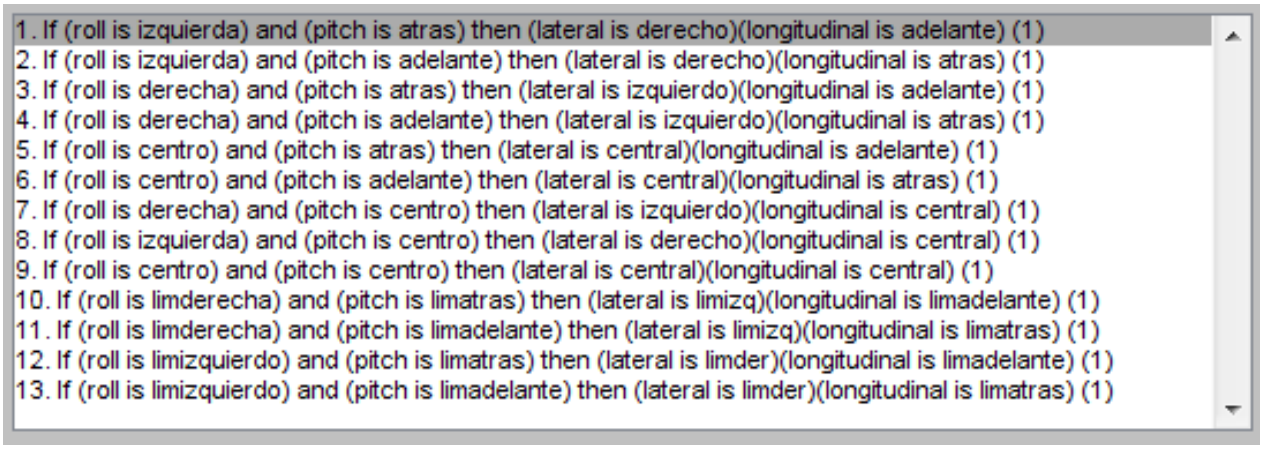


Fig. 4.12 Reglas

En la figura 4.13 se aprecian las superficies de control del sistema difuso.

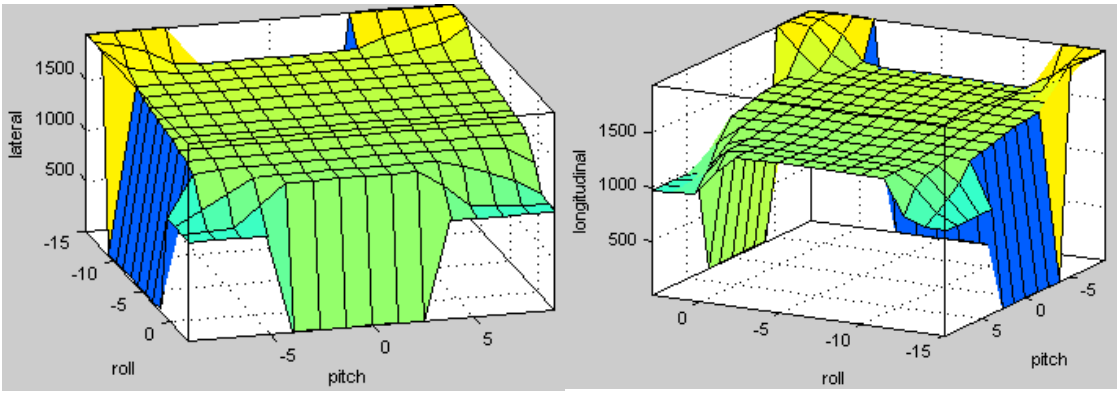


Fig. 4.13 Superficies de control

La señal de salida de cada actuador comparada con las señales de control provenientes del piloto se muestra en la figura 4.14, en la que se aprecia que el control ha evolucionado de manera considerable, comparado con las versiones pasadas.

El error calculado entre el piloto humano y el piloto difuso, se muestra en la figura 4.15 Se obtuvo un error relativo de 12.8496 % para el actuador lateral y un error de 10.9497 % para el actuador longitudinal.

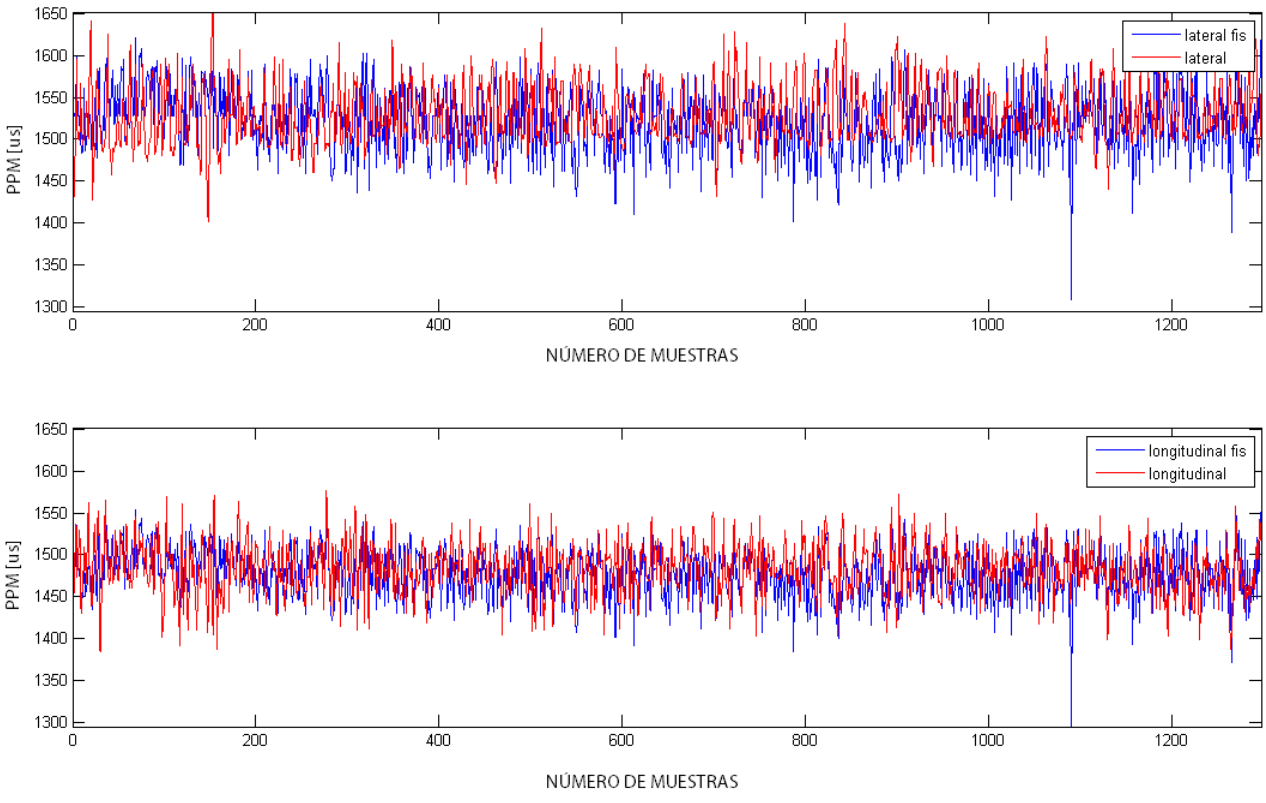


Fig. 4.14 Comparación Piloto-Control

Este control se comportó de manera adecuada pero tiene el problema de que ambas salidas para los actuadores se comportan muy similar en todos los casos, provocando que cuando el helicóptero tiene un ángulo cabeceo (*pitch*) positivo, el control genera una señal correctiva hacia el servomotor provocando que el plato longitudinal se mueva en un ángulo cabeceo (*pitch*) negativo. Lo anterior es correcto pero por la estructura del control, también hace que el servomotor lateral se mueva de manera negativa, generando un movimiento en el eje de alabeo (*roll*) negativo y de esta manera no se pueda estabilizar el helicóptero.

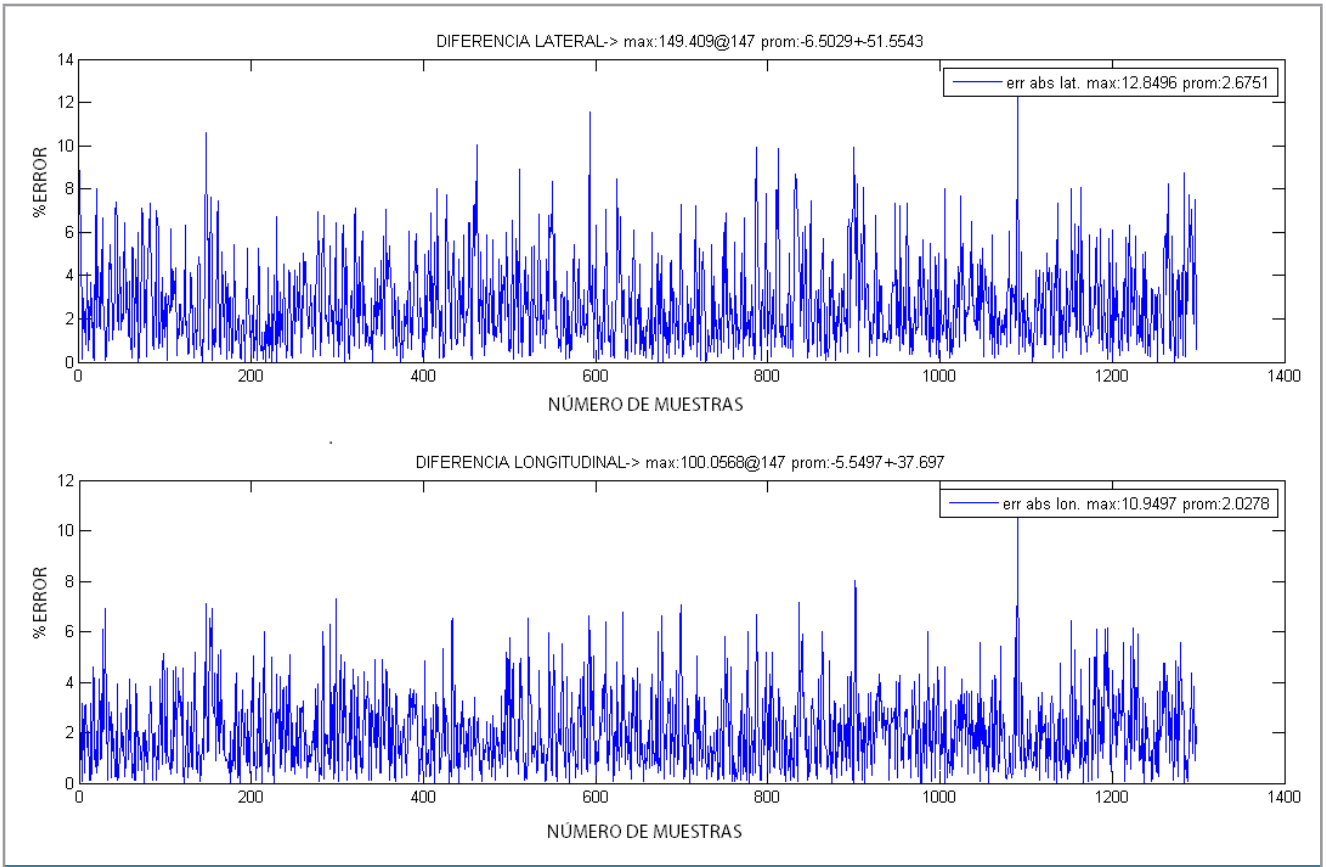


Fig. 4.15 Comparación Piloto-Control

Control difuso de 2 entradas y 1 salida

De la experiencia con el control de dos entradas y dos salidas, surgió la modificación de hacer dos sistemas, uno con salida para el actuador lateral y el otro para el actuador longitudinal. Ambos sistemas comparten las variables de entrada, como se aprecia el diagrama en la figura 4.16.

Este diseño solo fue propuesto pero nunca se llevó a la práctica debido a que por cada sistema existe una entrada que no tiene relación con la salida. De esta manera, la entrada del ángulo de cabeceo (*pitch*) no afecta a la salida lateral y por otro lado la entrada del ángulo alabeo (*roll*) no afecta a la salida longitudinal. Pero esta idea derivó en el diseño del siguiente control que involucra una entrada y una salida.

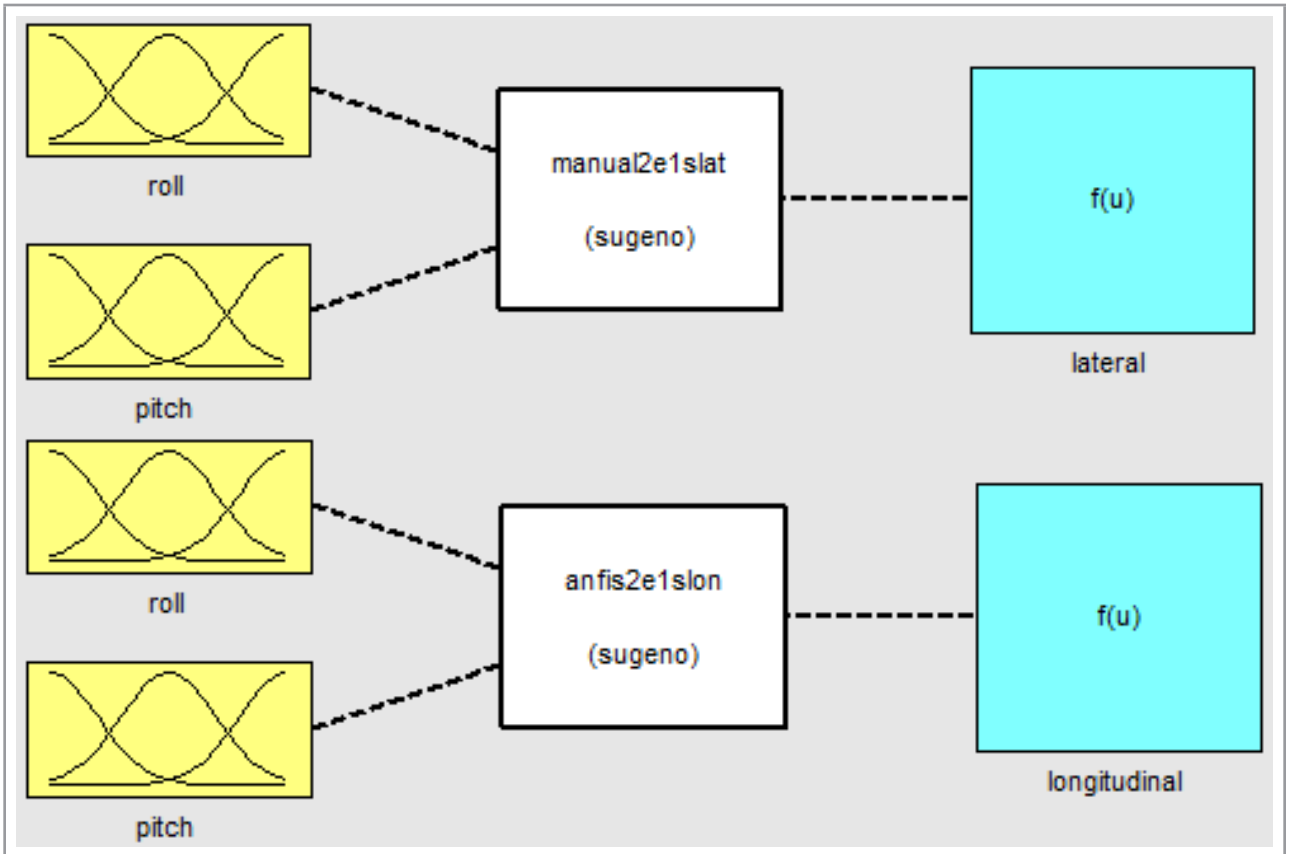


Fig. 4.16 Diagrama control difuso 2-1

Control difuso de 1 entrada y 1 salida

Con las observaciones en los dos tipos de controles anteriores se diseñó un control de una entrada y una salida. La entrada está directamente relacionada con la salida, como lo es la entrada alabeo (*roll*) con salida actuador longitudinal (figura 4.17) y entrada cabeceo (*pitch*) con salida al actuador lateral (figura 4.18).

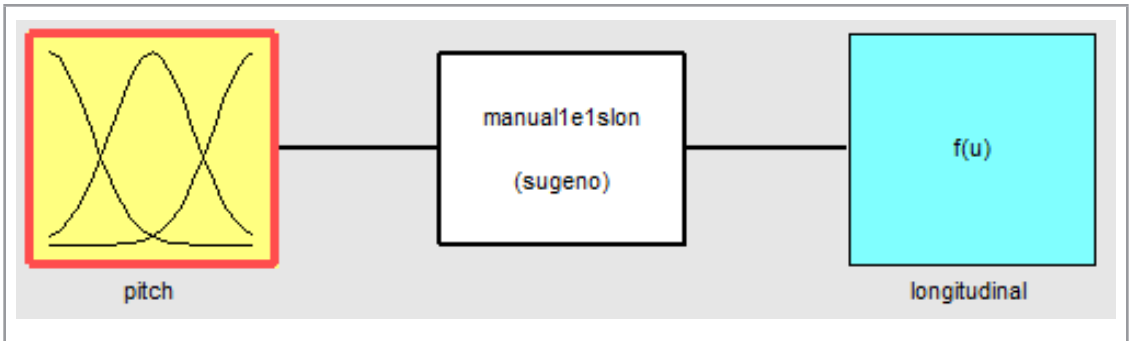


Fig. 4.17 Diagrama de control 1-1, actuador longitudinal

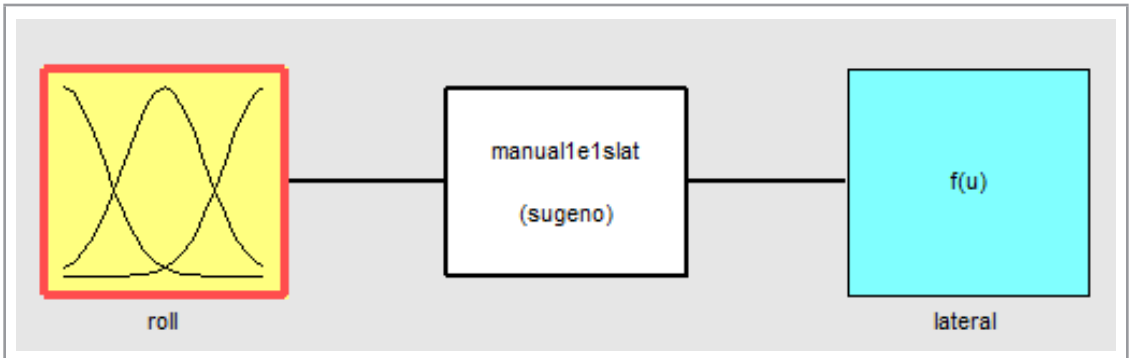


Fig. 4.18 Diagrama de control 1-1, actuador lateral

Cuenta con cinco funciones de membresía por variable de entrada de la forma triangular (figura 4.19). Se escogió ese tipo de función ya que necesita menos cálculos por parte del microcontrolador. La colocación de cada función dentro del rango de cada variable se realizó de forma heurística.

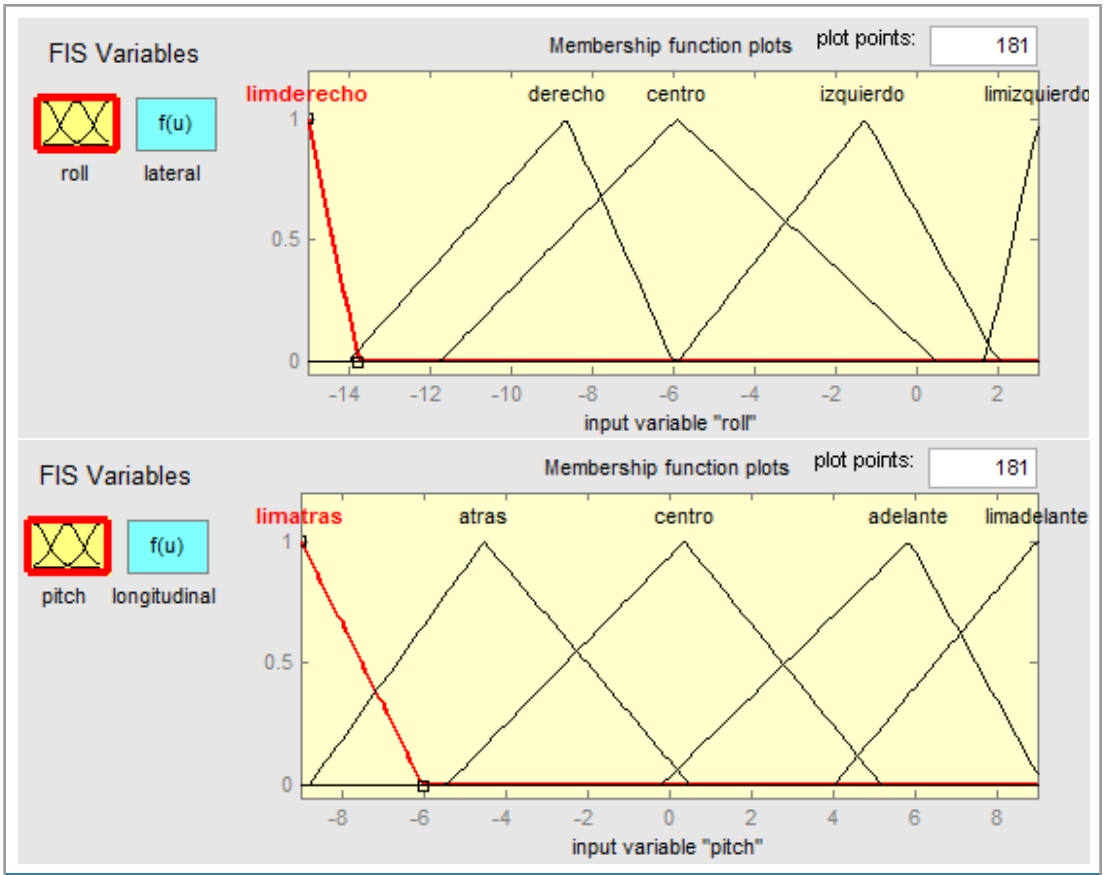


Fig. 4.19 Funciones de membresía control 1-1

El sistema se simplificó de manera considerable, tiene cinco reglas por sistema (figura 4.20).

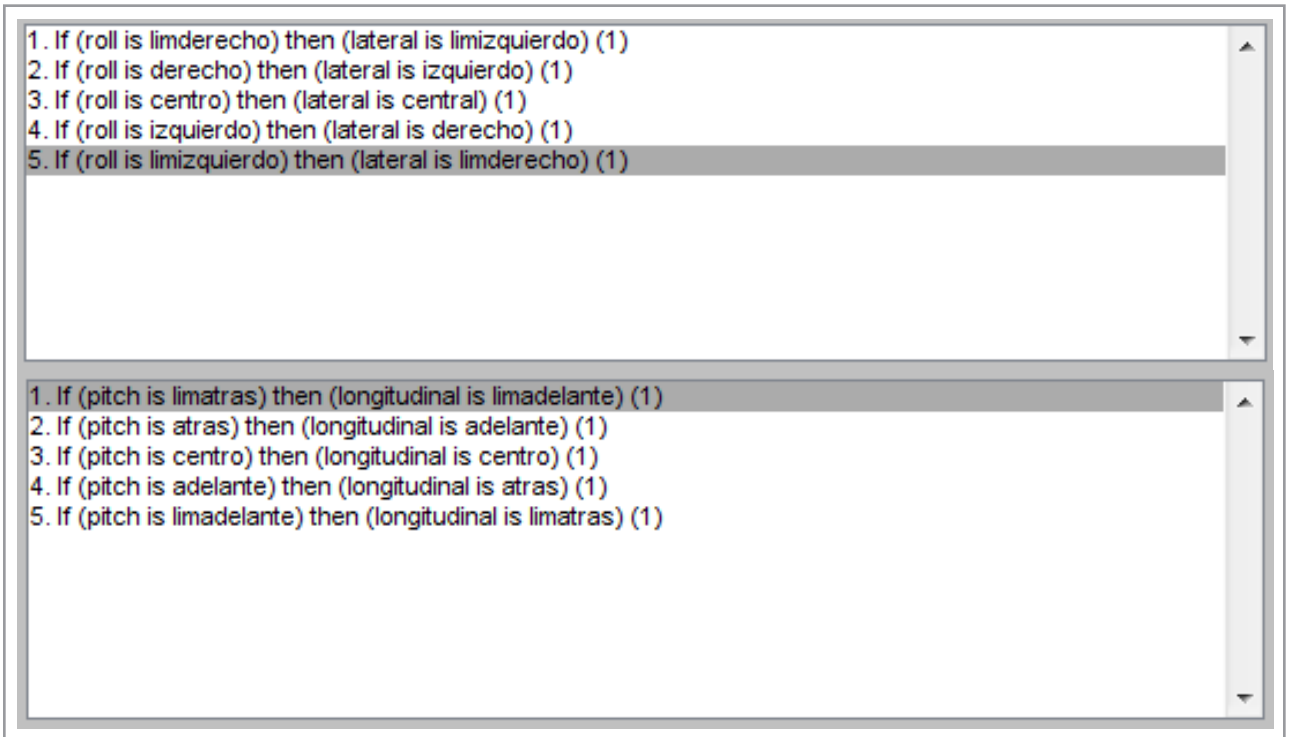


Fig. 4.20 Funciones de membresía control 1-1

En esta representación del sistema su sintonización es directa, cada ángulo afecta a las salidas de forma única.

En la figura 4.21 se aprecia que el comportamiento del control difuso es muy cercano al del piloto. El error (figura 4.22) es un poco mayor que en los controles anteriores, pero de todas maneras se sigue manteniendo en los estándares que teníamos. En el lateral un error de 11.817 % y en el longitudinal de 12.7333 %. Cabe resaltar que en la reglas se encuentra implícito la estabilidad, buscando los ángulos centrados alrededor de cero. También es importante conocer la dinámica de vuelo del helicóptero para poder diseñar las reglas.

Este tipo de control eliminó el problema de similitudes de las señales de los actuadores, permitiendo a los actuadores trabajar independientemente y de esta manera obtener la estabilidad que se desea.

En la figura 4.21 vemos que la salida del sistema lateral del control difuso está desbalanceada hacia la derecha mientras que para la salida longitudinal se encuentra aceptable la sintonización realizada.

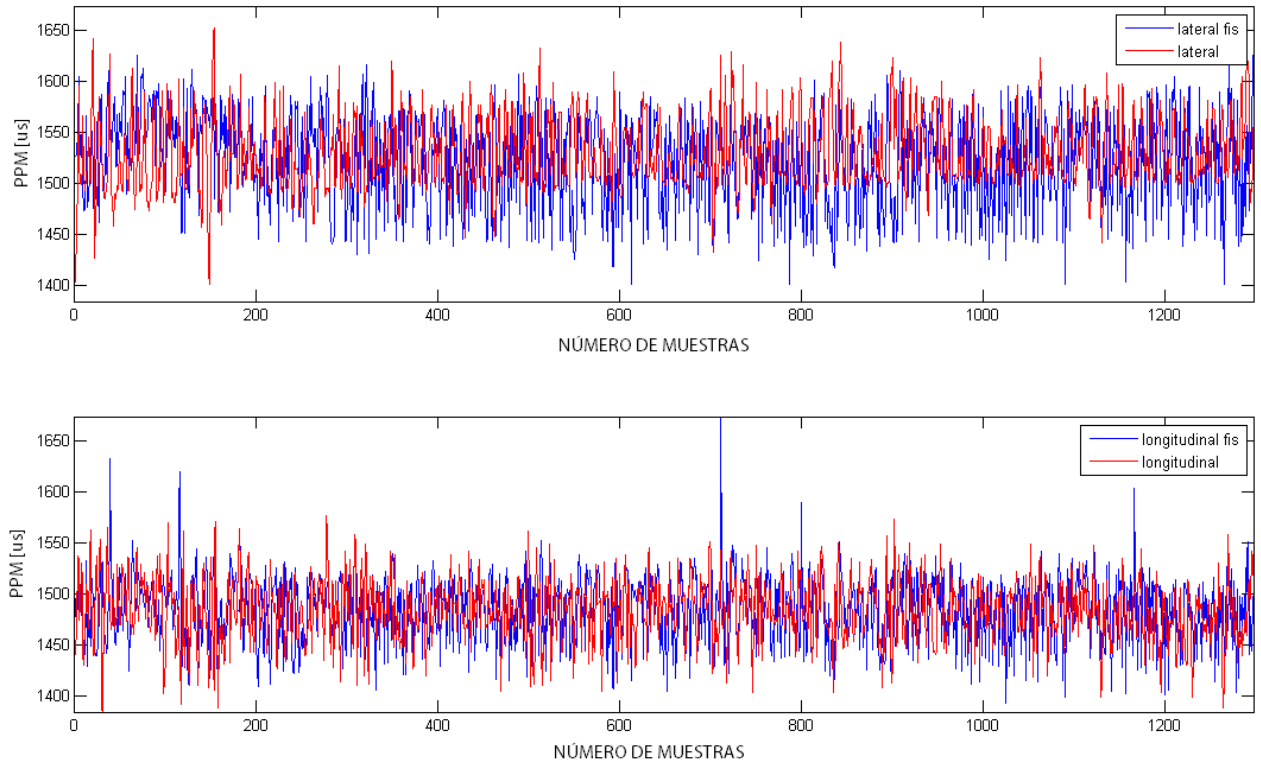


Fig. 4.21 Comparación Piloto-Control 1-1

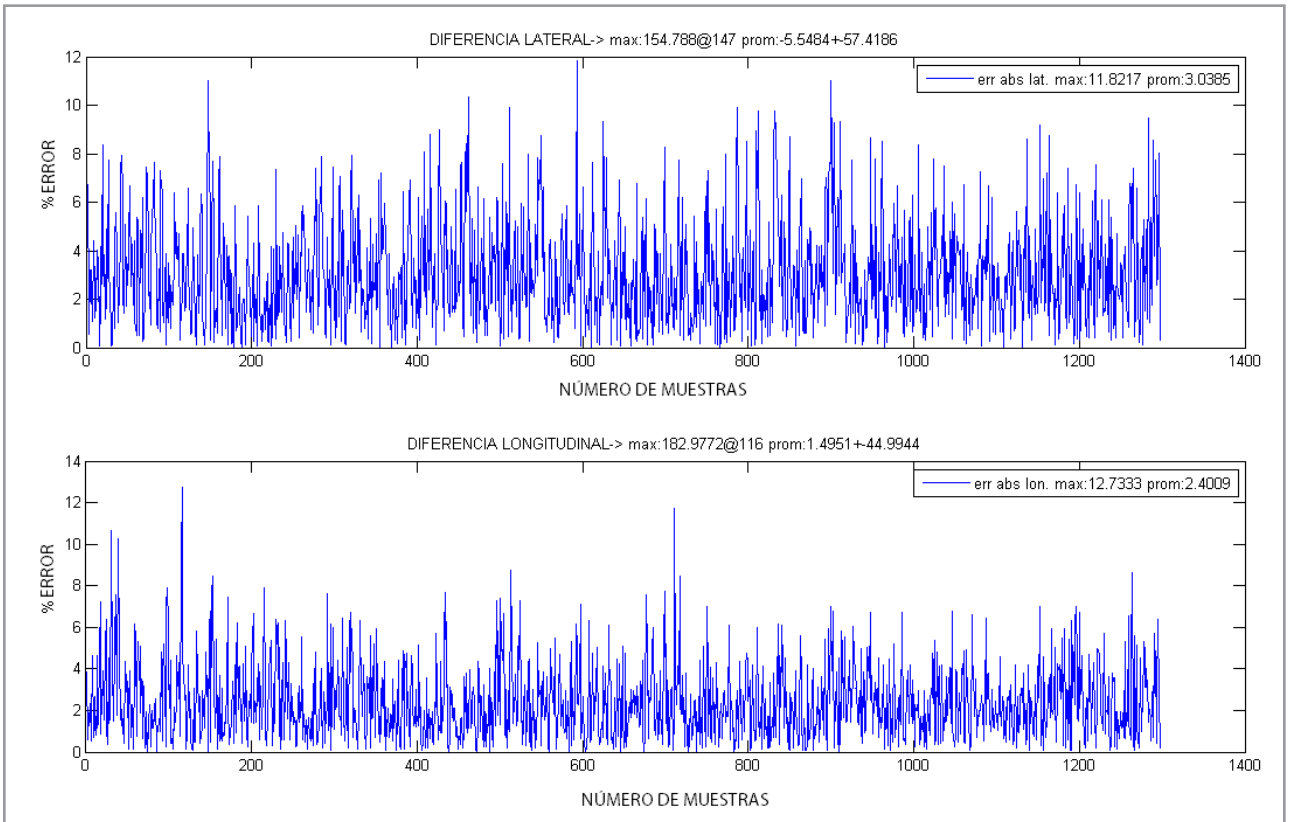


Fig. 4.22 Error control 1-1

Conclusiones

Aspectos como el adecuado diseño mecatrónico o el correcto tratamiento de las señales obtenidas por los sensores pueden condicionar el rendimiento del sistema.

El control neuro-difuso, además de evitar el problema del modelado o la obtención de parámetros de un modelo matemático, nos permite la gran facilidad de cambiar de plataforma (helicóptero) cuando se requiera. Para lograr este cometido se adapta la instrumentación a la nueva plataforma y se procede a realizar el entrenamiento de la red neuronal, generar las nuevas funciones de membresía y actualizar los parámetros en el controlador. De esta manera se ahorra tiempo y esfuerzo en la implementación de la estabilidad para nueva plataforma. Si se hace usando las maneras tradicionales (modelo matemático o parámetros) se tardaría demasiado tiempo en hacer un cambio de plataforma.

Se abandonó la parte de redes neuronales para la sintonización del sistema difuso porque las lecturas de los ángulos contenían demasiado ruido lo cual provocaba que la red neuronal no aprendiera de forma adecuada; que a su vez, generaba un ajuste incorrecto en el sistema difuso dando como resultado, que el helicóptero fuera inestable.

Se propuso dos sistemas difusos con una entrada y una salida, cada uno controla un servo del movimiento del cíclico. Se sintonizaron manualmente hasta que la diferencia entre las señales del sistema y del piloto fueran mínimas. Cada sistema difuso se comportaba de mejor manera que aquel sistema difuso sintonizado por la red neuronal.

Cabe resaltar que se usaron los mismos datos del vuelo del piloto que tenia disponibles la red neuronal para entrenarse.

Al momento de probar la eficacia del control diseñado se propuso crear una señal compuesta por el promedio ponderado del control del piloto y del sistema. En cada prueba de vuelo se fue variando la proporción del control del sistema desde cero hasta un 30%.

Después de cada vuelo realizado se entrevistó al piloto experto su experiencia del vuelo. Cada vez que se veía más involucrado el control del sistema difuso el piloto podía sentir como le ayudaba a corregir el movimiento del helicóptero para mantenerlo en vuelo estable.

Para el control del movimiento longitudinal hay una corrección total, indica que no realiza un gran movimiento para efectuar la corrección pero que aún no puede dejar que únicamente el sistema difuso haga por si mismo el trabajo.

Para el control lateral no existe una asistencia efectiva que ayude al piloto. Indica que se desvía demasiado hacia la izquierda y que es necesario realizar una acción correctiva por parte del piloto para mantener el vuelo estable.

El hecho de que este método no requiera de un modelo riguroso de la planta a controlar (helicóptero) facilita en gran medida el proceso de diseño del controlador, sobre todo cuando la determinación cuantitativa de los parámetros del sistema se hace compleja, como en nuestro caso.

El nivel de arte involucrado en el diseño del controlador no asegura un resultado óptimo inmediato, y en vista que no resulta sencillo o intuitivo introducir modificaciones al controlador difuso, se tiene una desventaja comparativa frente al control clásico.

Una posible metodología para desarrollar sistemas neuro - difusos tiene que contener las siguientes características:

- Aprendizaje rápido
- Capacidad de adaptabilidad para entrenamiento en tiempo real
- Autoajuste con el objetivo de generar el menor error global posible
- Reducida complejidad computacional

Es de suma importancia saber que la instrumentación, adquisición de datos y el pre-procesado de los mismos, requiere una atención especial ya que es la clave para el éxito de una aplicación con un sistema neuro-difuso, en especial porque en el entrenamiento es indispensable que los datos sean de primera calidad.

La sintonización manual del sistema difuso requiere conocimiento del sistema a controlar y de las capacidades del microcontrolador. Por parte del microprocesador existen restricciones de:

- Límite de memoria
- Capacidad de proceso
- Velocidad de respuesta

Por parte del sistema son puntuales, en nuestro caso intervino:

- El peso total de vuelo
- La capacidad energética de la batería
- La pericia del piloto para obtener una buena calidad de datos

Ambos conjuntos de restricciones se imponen al momento de decidir que tan fino se desea que sea el sistema difuso, impactando principalmente en el número de funciones de membresía y de las reglas.

Los objetivos planteados se han cumplido. Se puede pilotar el helicóptero con el sistema de control activado. El sistema de control proporciona una mejor respuesta en el servo longitudinal que en lateral. Para mejorar el comportamiento del servo lateral se tiene que calibrar las funciones de membresía de la variable de entrada es decir, el ángulo de alabeo (*roll*).

Una expansión de este sistema es controlar la posición del helicóptero sobre el eje de guiñada (*yaw*). Para tal propósito recomendamos la *IMU 9DOF Razor*, el cual es una mejor opción respecto al que se adquirió debido a que contiene magnetómetros, los cuales son usados para mantener una referencia respecto a los polos magnéticos en cada eje y obtener una medición de la orientación espacial mas precisa.

La técnica que usamos para alisar la señal de los sensores de los ángulos tiene una modificación que difiere de lo expuesto en el capítulo de instrumentación. En la implementación se usó el complemento del coeficiente de alisado

$$\alpha' = 1 - \alpha$$

Dando como resultado la ecuación de alisado modificada

$$y_{t+1} = (1 - \alpha')x_t + \alpha' y_t$$

En esta ecuación, el coeficiente de alisado tiene su función invertida. Cuanto esta cercano a cero deja pasar gran parte del ruido y cuando es cercano a uno deja la señal muy alisada.

Recomendaciones

Nuestro planteamiento inicial fue realizar el control con un sistema neuro-difuso. Para que se pueda usar esta técnica es necesario que los datos de entrenamiento no contengan ruido y en el peor de los casos, que este ruido pueda ser filtrado a través de un filtro paso bajas de tipo Butterworth o Chebyshev.

En el capítulo de introducción se comenta acerca del requerimiento de un piloto profesional de helicópteros a escala Es importante para poder ahorrar dinero en reparaciones así como obtener comentarios y opiniones de juicio sobre el comportamiento del control, ya que las pruebas se realizaron haciendo porcentajes entre las señales del piloto y las señales del control y eran multiplicadas por un factor de ponderación que iba incrementando conforme el piloto indicaba que el funcionamiento del control era correcto.

Se recomienda que la plataforma (helicóptero) sea más grande que la escala actual que es la 450 (Falcon 3D), ya que entre mas grande es el helicóptero; los efectos ambientales y mecánicos vibratorios disminuyen a tal grado que la tarea de estabilización es relativamente más sencilla.

Si el objetivo del proyecto es el diseño y construcción de un *RUAV* (Vehiculo Aereo No Tripulado de Ala Rotatoria), la plataforma tradicional de helicóptero de rotor principal y rotor trasero no es la adecuada, por sus capacidades de carga y problema de estabilización, es mejor que se considere una plataforma con 3 rotores (Tricoptero) ó inclusive 4 rotores (Quadcoptero). Este tipo de configuración nos ofrece una capacidad mayor de carga, mejor estabilización, menor vibración, pero como son 3 o 4 motores iguales, no contiene mecanismos complejos para controlarse, se mueve en todas las direcciones modificando la velocidad de cualquiera de los motores, el problema principal es que el tiempo de vuelo se disminuye ya que se tienen que energizar 3 ó 4 motores con la misma batería.

Contar con dos plataformas de vuelo (helicóptero) idénticas es muy recomendable porque se sufrirán accidentes y choques los cuales, muchas veces pueden retrasar la investigación y por ende, mientras se repara una plataforma la otra puede ser utilizada para continuar con las pruebas.

La plataforma de vuelo debe de encontrarse en óptimas condiciones mecánicas, las hélices principales (palas) deben de estar balanceadas, así como la barra estabilizadora (*fly-bar)*, los componentes de la cabeza del rotor principal bien ajustados. En general todo debe de estar en condiciones impecables, todo esto reduce considerablemente la vibración provocada por el helicóptero.

Cambiando la velocidad de muestreo en la lectura de los acelerómetros y giroscopios, para que coincida con las revoluciones del rotor principal puede resolver los problemas de vibración.

Los giroscopios que ofrecen un rechazo superior a la vibración son los IDG500, de la compañía InvenSense. Existe una nueva IMU son estos sensores, producto sen-09268 producido por la compañía Sparkfun.

Sería interesante conocer como puede repercutir los resultados del sistema difuso usar la máquina de inferencia tipo Mamdani, pensamos que tiene una mejor resolución en el momento de procesar los consecuentes de las reglas dando una respuesta mejor acertada.

Esta aplicación se puede escalar si se cambia de microcontrolador. Sugerimos usar uno de 32 bits, específicamente la plataforma *Maple* de leaflabs.com. Esta plataforma ofrece un microcontrolador ARM Cortex M3 a 72Mhz que ofrece compatibilidad con los códigos realizados para la plataforma Arduino. Al hacer este cambio de plataforma se pueden agregar funciones que requieran un cálculo computacional adicional, entre ellas:

Manejo de cámaras

Conectividad con otros microcontroladores

Expansión de la estructura del sistema difuso ya sea en tipo y número de funciones de membresía, número de reglas hasta la utilización de la máquina de inferencia mamdani.

Implementación de AHRS.

Apéndice

Contenido Apéndice

Motor OutRunner Brushless

Característica	Valor
Dimensiones (mm)	Ø28×32
Diámetro eje (mm)	3
Peso (gramos)	60
Potencia (W)	220
KV sin carga (rpm / V)	1200
Máxima corriente sostenida (A)	28
Corriente sin carga (A)	1.2
Resistencia Interna (mOhm)	77
Empuje (oz)	35.2
Peso recomendado de modelo (gr)	400 - 1200

ArtTech GoldenPower OutRunner Brushless motor

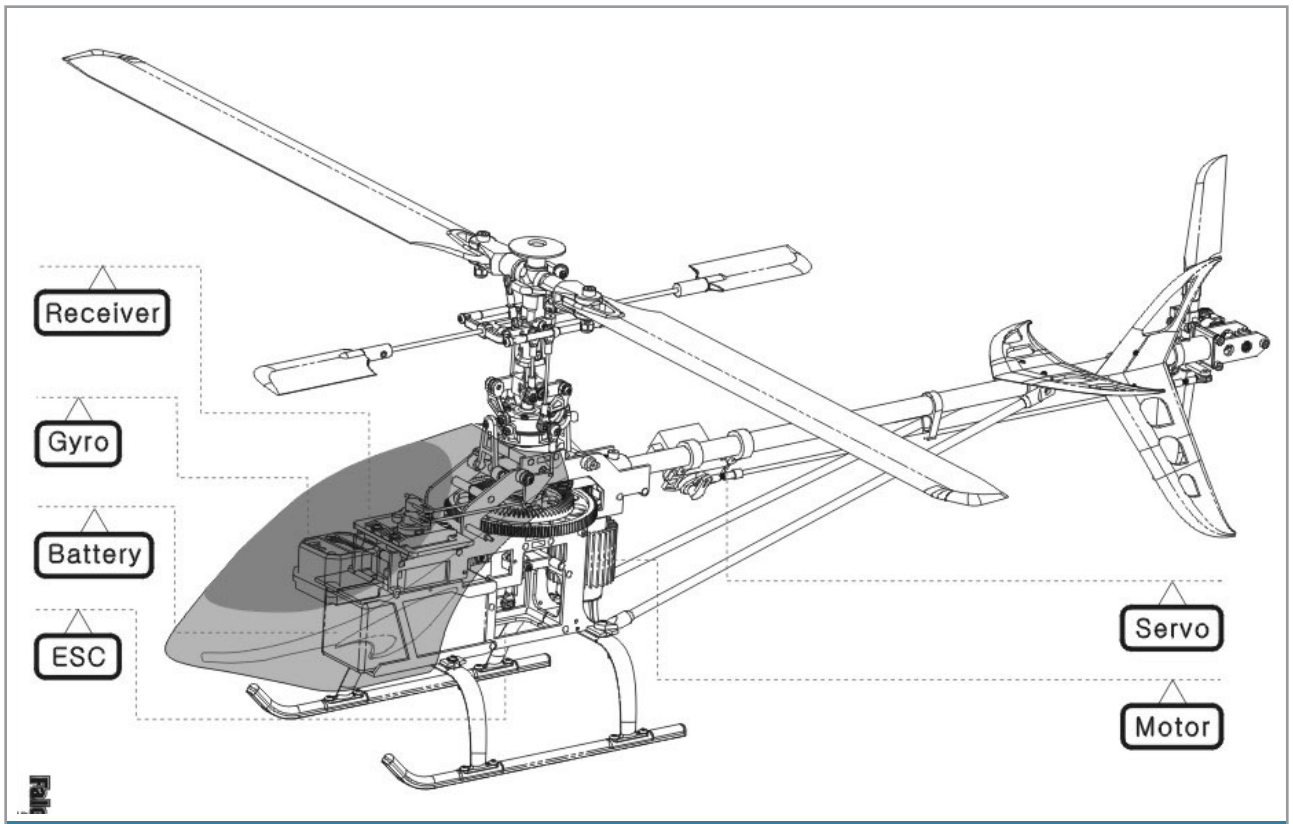
Electronic Speed Controller

Característica	Valor
Corriente (Amp)	30
Corriente Maxima >10s (Amp)	40
Modo BEC	Linear
Salida BEC	5V / 2A
Celdas LiPo	2 - 4
Programable	Si
Peso (gramos)	25
Dimensiones (mm)	34 x 24 x 11

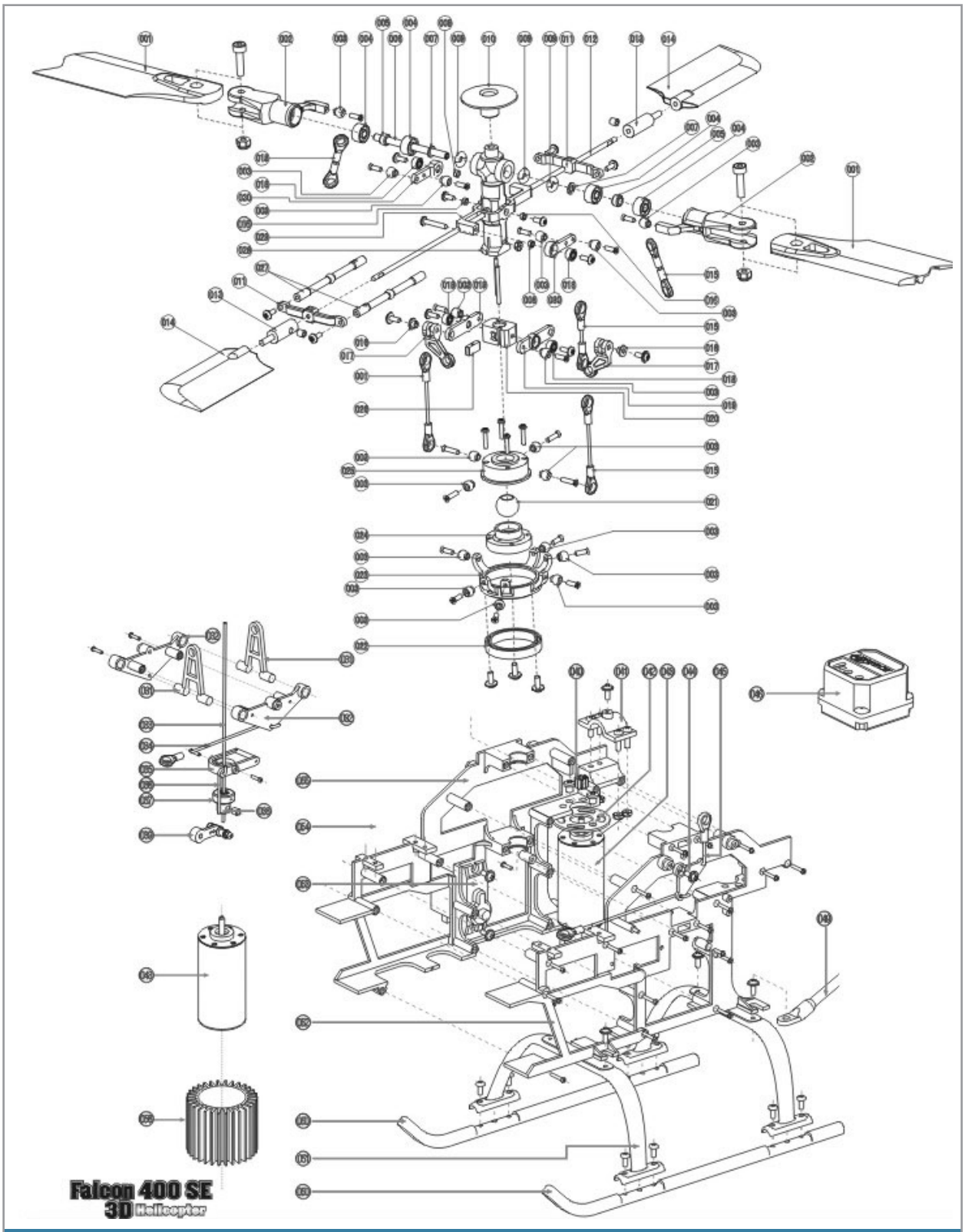
Exceed RC Volcano Series 30 Amp ESC

Helicóptero

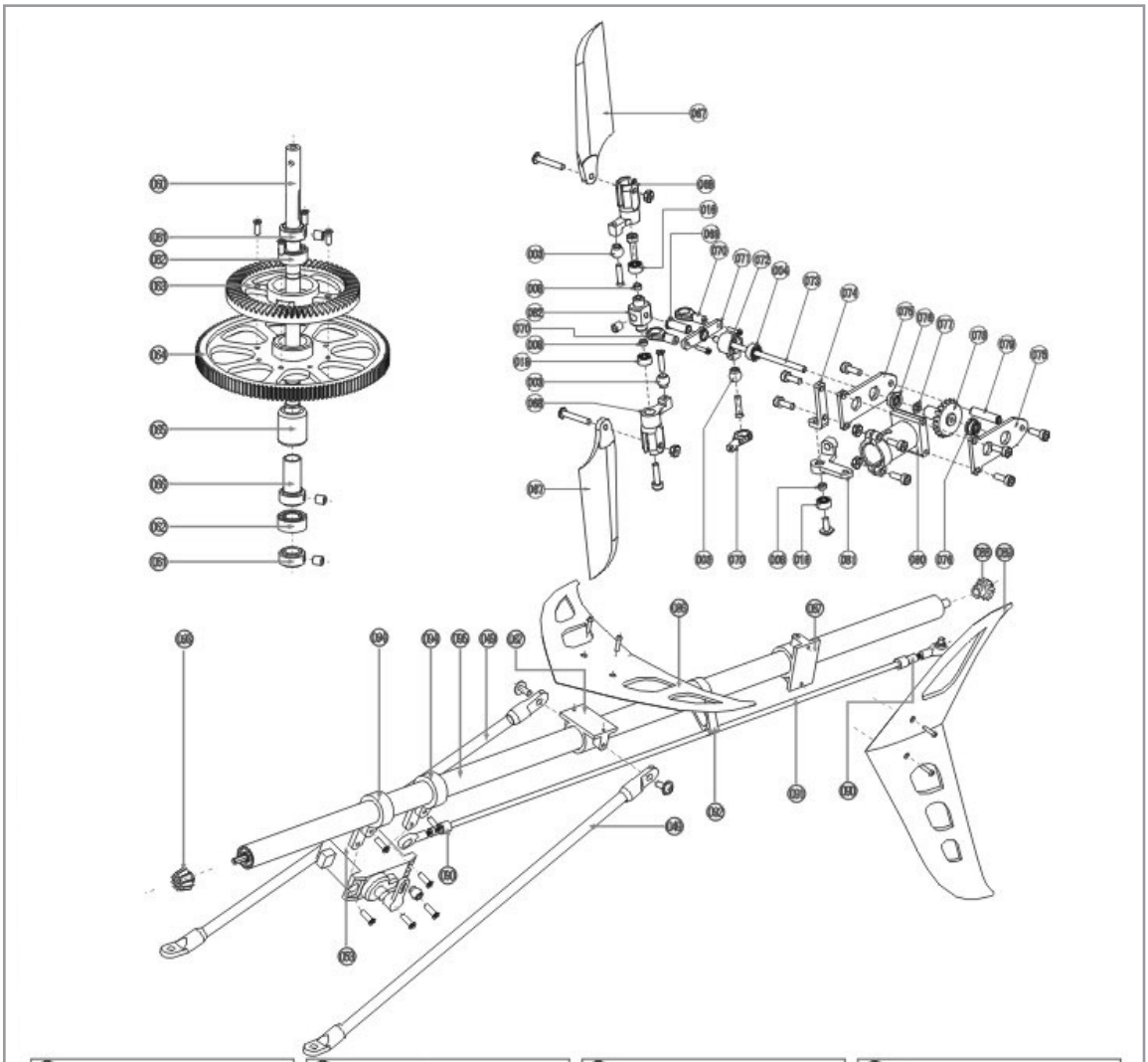
Característica	Valor
Diámetro rotor principal (mm)	630
Diámetro rotor trasero (mm)	152
Largo (mm)	670
Peso (gramos)	530
Vuelo 3D	Si



Exceed RC Volcano Series 30 Amp ESC



Explosivo 1 ArtTech Falcon 400 SE 3D



001 Main rotor blade	023 Outside inclined tray	045 Lever (φ1.2×20)	067 Sliding bush
002 Up main rotor wing grip	024 Ball fixed plate	046 Gyro	072 Nut bearing with holder
003 Ball	025 Inside inclined tray	047	073 Tail shaft (2×45)
004 Bearing(8×3×4)	026 Orienting sheet	048	074 Bracket
005 Spacing sleeve for cross nut	027 Arm	049 Stay bar (4x160)	075 Tail holder
006 Cross shaft(φ3×40)	028 Spacing sleeve for balance bar	050 Ski	076 Nut
007 Washer	029 Stabilizer frame	051 Foot rest	077 Washer
008 Washer	030 Stable wing arm (abs)	052 Left frame	078 Tail rotor gear (M0.7×Z20)
009 Oil seal gasket (6.5x2.7x1.9)	031 A arm	053 9 gram Servo	079 Bracket
010 Rotor wing control hat	032 Holder for elevator servo	054 Servo holder	080 Tail gear box
011 Stable arm(metal)	033 Lever (φ1×100)	055 Right frame	081 Tail control arm
012 Balance bar(φ1.8x20)	034 Lever (φ1.2×63)	056 Motor's radiator	082 Tail cross shaft (4.2×14)
013 Metal joint sleeve(φ6x23)	035 Total pitch arm	061 Main rotor shaft (5×98)	083 Horizontal stabilizer
014 Stable wing	036 Lever (φ1.2x16)	062 Spacing sleeve for main shait (9x4.6)	087 Holder
015 Lever (φ1.2x12)	037 Inside sleeve for arm	063 Nut (10×5×4)	088 Behind driver gear (M0.7×Z11)
016 Copper sleeve (5x2.2)	038 Main rotor bearing block (6x9.5)	064 Main (small) gear (Z66 x M0.7)	089 Vertical stabilizer
017 Triangle bell buckle	039 Arm	065 Main (big) gear (Z132 x M0.5)	090 Lever sheath
018 Nut (2X5X2.5)	040 Motor gear	066 One-way nut (6×10×2)	091 Lever (φ2x240)
019 SF arm	041 Tail pipe sheet	068 One-way nut seat (10×16.6)	092 Piper fixer
020 Orientator	042 Heat insulation plate of motor	069	093 Ahead driver gear (M0.7×Z10)
021 Ball (10×8)	043 Brushless motor	070 Tail rotor wing grip	094 Tail servo holder
022 Nut (25×20×4)	044 Triangle rocker arm	071 Tail rotor wing	095 Tail pipe (8.8×8×360)

Explosivo 2 ArtTech Falcon 400 SE 3D

Transmisor

Característica	Valor
Modulación	FM - PPM
Corriente de trabajo (mAmp)	< 250
Potencia señal (mW)	500
Voltaje de trabajo (Volts)	12
Canales	6
Modo de vuelo	A / V / H / C

Transmisor E-Fly 100C R/C

ANALOG DEVICES

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

3-axis sensing
Small, low profile package
4 mm × 4 mm × 1.45 mm LFCSP
Low power : 350 µA (typical)
Single-supply operation: 1.8 V to 3.6 V
10,000 *g* shock survival
Excellent temperature stability
BW adjustment with a single capacitor per axis
RoHS/WEEE lead-free compliant

APPLICATIONS

Cost sensitive, low power, motion- and tilt-sensing applications
Mobile devices
Gaming systems
Disk drive protection
Image stabilization
Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

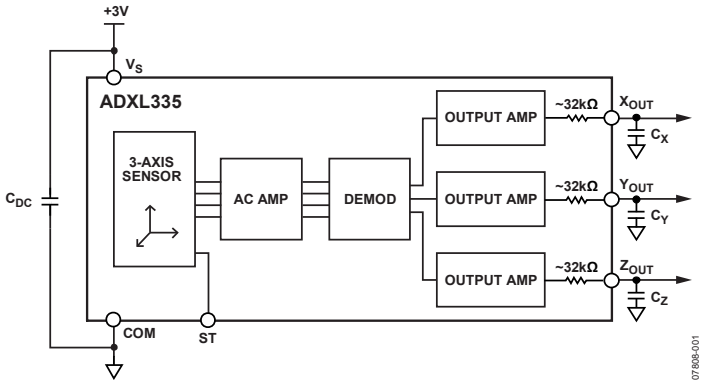


Figure 1.

Datasheet ADXL335 acelerómetro

SPECIFICATIONS

T_A = 25°C, V_S = 3 V, C_X = C_Y = C_Z = 0.1 µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		m*g*/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µ*g*/√Hz rms
Noise Density Z_{OUT}			300		µ*g*/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × C). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.
[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately 160 × C_X or C_Y or C_Z + 1 ms, where C_X, C_Y, C_Z are in microfarads (µF).

Datasheet ADXL335 acelerómetro

LPR530AL

MEMS motion sensor: dual axis pitch and roll ±300°/s analog gyroscope

Preliminary data

Features

- 2.7 V to 3.6 V single-supply operation
- Wide operating temperature range (-40 °C to +85 °C)
- High stability overtemperature
- Analog absolute angular-rate output
- Two separate outputs for each axis (1x and 4x amplified)
- Integrated low-pass filters
- Low power consumption
- Embedded power-down
- Embedded self-test
- High shock and vibration survivability
- ECOPACK® RoHS and “Green” compliant (see *Section 5*)

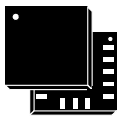
Applications

- Pointing devices, remote and game controllers
- Motion control with user interface
- GPS navigation systems
- Industrial and robotics

Description

The LPR530AL is a low-power dual-axis micromachined gyroscope capable of measuring angular rate along pitch and roll axes.

It provides excellent temperature stability and high resolution over an extended operating temperature range (-40 °C to +85 °C).



LGA-16L (5x5x1.5mm)

The LPR530AL has a full scale of ±300 °/s and is capable of detecting rates with a -3 dB bandwidth up to 140 Hz.

The gyroscope is the combination of one actuator and one accelerometer integrated in a single micromachined structure.

It includes a sensing element composed by single driving mass, kept in continuous oscillating movement and able to react when an angular rate is applied based on the Coriolis principle.

A CMOS IC provides the measured angular rate to the external world through an analog output voltage, allowing high level of integration and production trimming to better match sensing element characteristics.

ST’s gyroscope family leverages on robust and mature manufacturing process already used for the production of micromachined accelerometers.

ST is already in the field with several hundreds million sensors with excellent acceptance from the market in terms of quality, reliability and performance.

LPR530AL is provided in plastic land grid array (LGA) package. Several years ago ST pioneered successfully the usage of this package for accelerometers. Today ST has the widest manufacturing capability and strongest expertise in the world for production of sensor in plastic LGA package.

Table 1. Device summary

Order code	Temperature range (°C)	Package	Packing
LPR530AL	-40 to +85	LGA-16 (5x5x1.5)	Tray
LPR530ALTR	-40 to +85	LGA-16 (5x5x1.5)	Tape and reel

Table 2. Pin description

Pin #	Pin name	Analog function
1	GND	0V supply voltage
2	FILTVDD	PLL filter connection pin #2
3	VCONT	PLL filter connection pin #1
4	OUTY	Not amplified output
5	4xINY	Input of 4x amplifier
6	4xOUTY	Y rate signal output voltage (amplified)
7	Vref	Reference voltage
8	4xOUTX	X rate signal output voltage (amplified)
9	4xINX	Input of 4x amplifier
10	OUTX	Not amplified output
11	ST	Self-test (logic 0: normal mode; logic 1: self-test)
12	PD	Power-down (logic 0: normal mode; logic 1: power-down mode)
13	HP	High pass filter reset (logic 0: normal operation mode; logic1: external high pass filter is reset)
14,15	Res	Reserved. Connect to Vdd
16	Vdd	Power supply

2.1 Mechanical characteristics

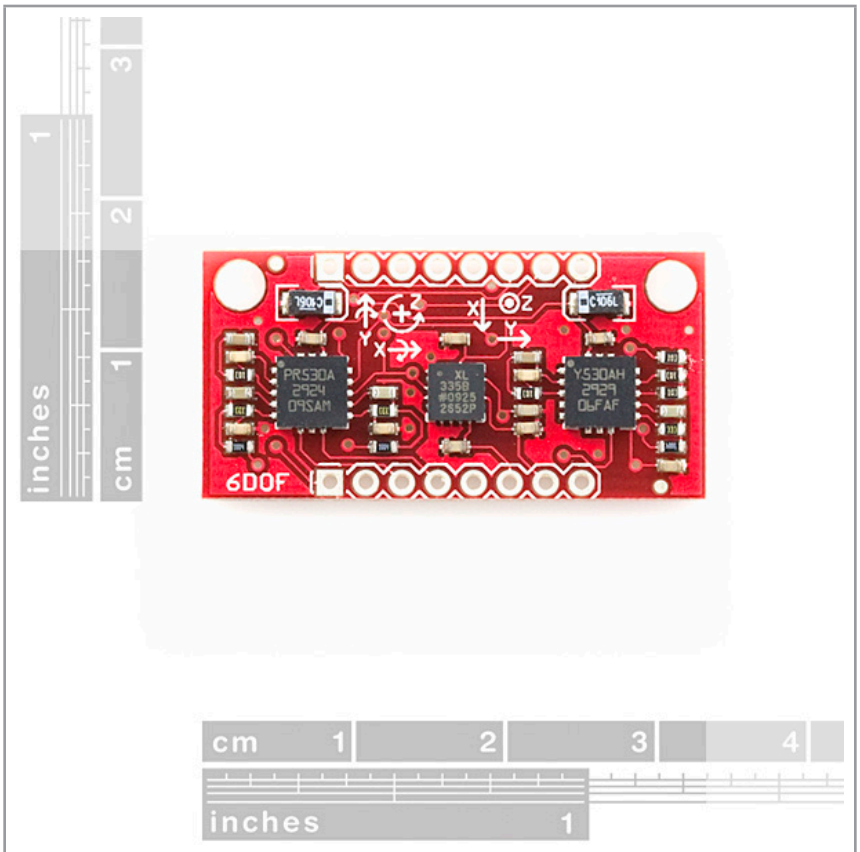
Table 3. Mechanical characteristics @ Vdd = 3 V, T = 25 °C unless otherwise noted(1)

Symbol	Parameter	Test condition	Min.	Typ.(2)	Max.	Unit
FSA	Measurement range	4x OUT (amplified)		±300		°/s
FS		OUT (not amplified)		±1200		°/s
SoA	Sensitivity(3)	4x OUT (amplified)		3.33		mV/ °/s
So		OUT (not amplified)		0.83		mV/ °/s
SoDr	Sensitivity change vs temperature	Delta from 25°C		0.03		%/°C
Voff	Zero-rate level(3)			1.23		V
Vref	Reference voltage			1.23		V
OffDr	Zero-rate level change Vs temperature	Delta from 25°C		0.05		°/s/°C
NL	Non linearity	Best fit straight line		±1		% FS
BW	Bandwidth(4)			140		Hz
Rn	Rate noise density			0.035		°/s / $\sqrt{Hz}$
Top	Operating temperature range		-40		+85	°C

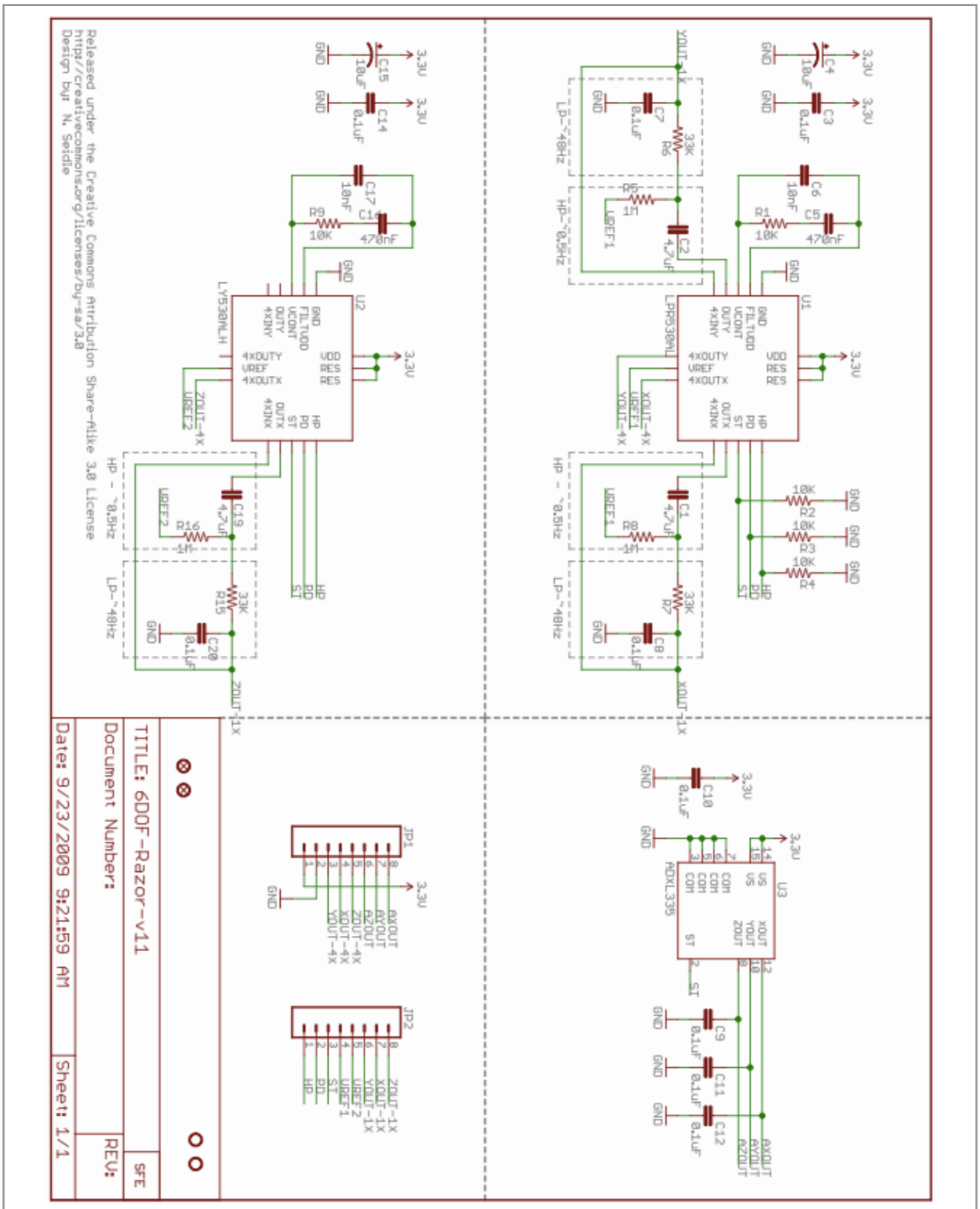
1. The product is factory calibrated at 3 V. The operational power supply range is specified in *Table 4*.
2. Typical specifications are not guaranteed
3. Sensitivity and zero-rate level are not ratiometric to supply voltage
4. The product is capable of measuring angular rates extending from DC to the selected BW.

IMU 6 DOF Razor

Característica	Valor
Alimentación	2.7 - 3.6 volts CD
Giroscopio Roll y Pitch	LPR530AL
Giroscopio Yaw	LY530ALH
Acelerometro	ADXL355
Costo	$ 1,292 mxn



Vista frontal IMU 6DOF Razor



Esquemático IMU Razor (cortesía de SparkFun)

Arduino Duemilanove

Característica	Valor
Microcontrolador	Atmel ATmega 328
Arquitectura	RISC 8 bits
Velocidades	16 mhz
Entradas/Salidas digitales	14
Entradas Analógicas	6 (ADC 10 bits)
Memoria Flash	32 kb
Alimentación	6 - 20 volts CD
SRAM	2 kb
EEPROM	1 kb

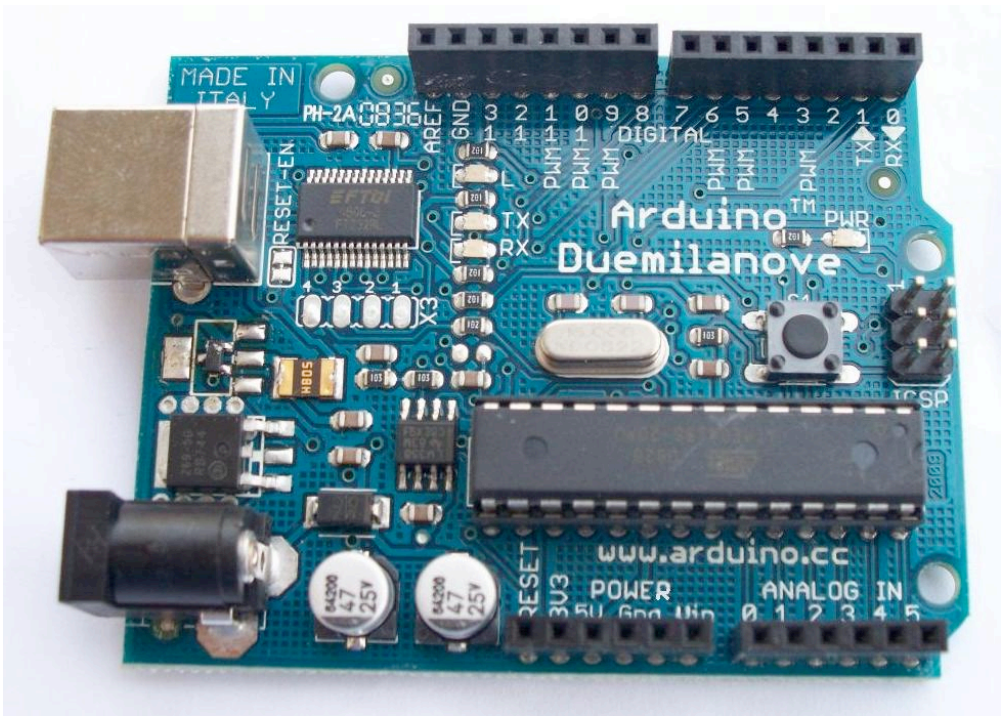


Fig. 2.32 Arduino (Cortesía SparkFun)

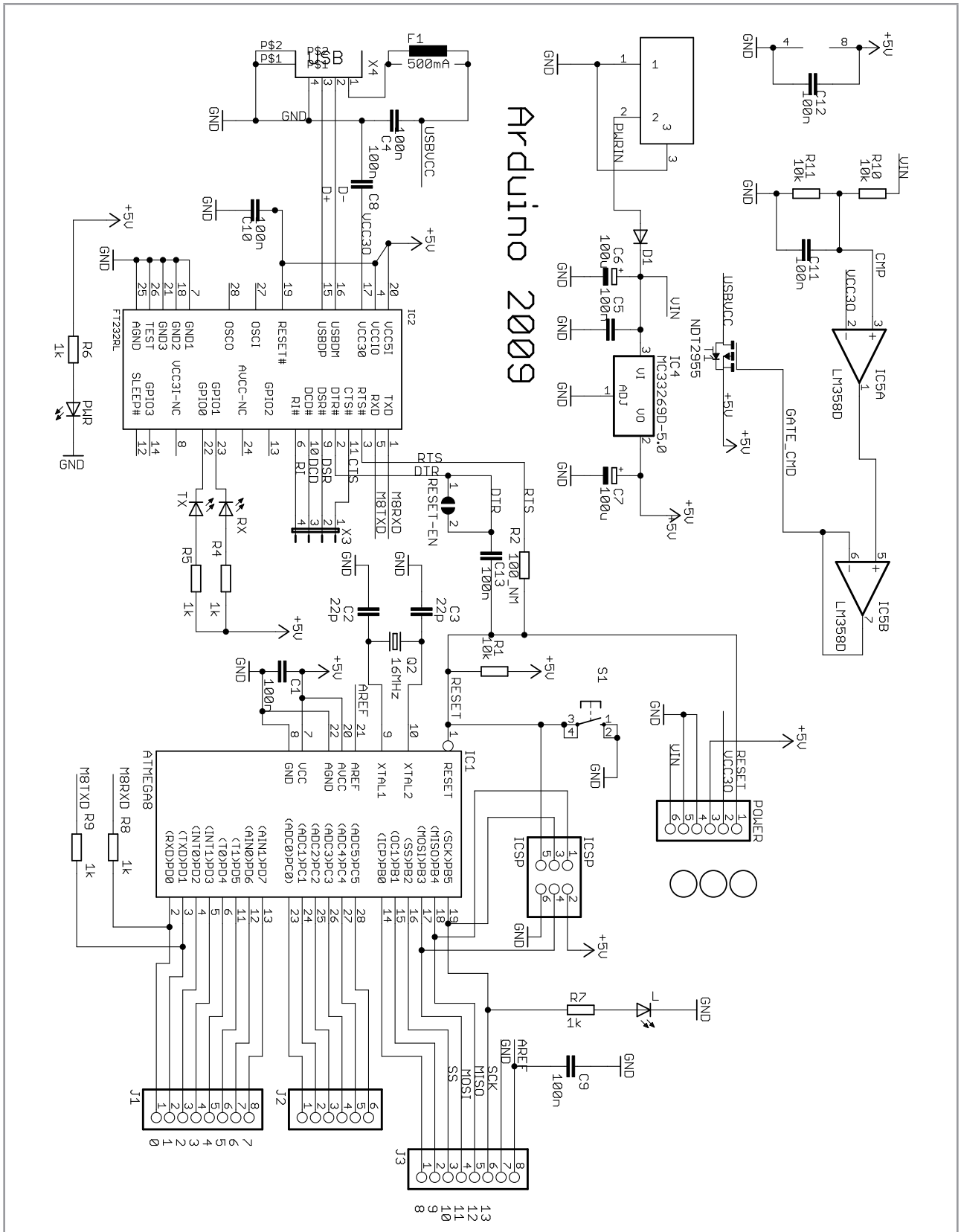


Fig. 2.33 Esquemático Arduino (cortesía de Arduino.cc)

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Fig. 2.34 Características ATmega 328

Software Arduino

Setup()

```
#include <math.h>
#include <ServoTimer2.h>
#include "kalman.h"
#include "fuzzy.h"
#include "contexto.h"


unsigned long Now = millis();
unsigned long lastread = Now;

volatile uint8_t MuxSel=0;
volatile uint8_t analog_reference = DEFAULT;
volatile int16_t analog_buffer[8];


void setup()
{
  Serial.begin(38400);
  pinMode( 8, INPUT );

  ciLat.attach(ser1out);
  ciLon.attach(ser2out);

  setup_timer1();

  Analog_Reference(EXTERNAL);
  Analog_Init();


  for(int c=0; c<75; c++)
  {
    read_adc_raw();

   delay(100);
  }

  for(int y=0; y<=5; y++)
  {
    AN_OFFSET[y]=AN[y];
    Serial.println((int)AN_OFFSET[y]);
  }
    AN_OFFSET[5]=AN[5]-GRAVITY;

    kalmanInitState(&xkaldata);
    kalmanInitState(&ykaldata);
}
```

setup()

loop() y control()

```
void loop()
{
    Now = millis();
    dt = (Now - lastread) * .001;
    read_adc_raw();

    state_update(Gyro_Scaled(read_adc(1)),&xkaldata, dt);
    state_update(Gyro_Scaled(read_adc(0)),&ykaldata, dt);

    kalman_update( (atan2(-read_adc(5),read_adc(3))-(PI/2)) ,&xkaldata);
    kalman_update( (atan2(-read_adc(5),read_adc(4))-(PI/2)) ,&ykaldata);



    if((millis()-timer24)>=100)
    {
      timer24=millis();

      printdata();
      control();

    }

    lastread = Now;
}

void control(){

   if(serinp[4] > 1500){

     evalControl();
     if (!(lat > 1990 || lat < 1030))
       ciLat.write(lat*factorControl+serinp[0]*factorPiloto);
     else
       ciLat.write(serinp[0]);

     if (!(lon > 1990 || lon < 970))
       ciLon.write(lon*factorControl+serinp[1]*factorPiloto);
     else
       ciLon.write(serinp[1]);
   }
   else{

     ciLat.write(serinp[0]);
     ciLon.write(serinp[1]);

   }

}
```

loop() y control()

ADC (analog-digital converter)

```
void read_adc_raw(void)
{
  float filterVal = 0.6;

 AN[0]= smooth(((Analog_Read(1)*.9)+(AN[0]*.1)), filterVal + 0.2 , AN[0]);//Gx
 AN[1]= smooth(((Analog_Read(0)*.9)+(AN[1]*.1)), filterVal + 0.2 , AN[1]);//Gy
 AN[2]=  1; // Gz
 AN[3]= smooth(((Analog_Read(5)*.9)+(AN[3]*.1)), filterVal , AN[3]); // Ax
 AN[4]= smooth(((Analog_Read(4)*.9)+(AN[4]*.1)), filterVal , AN[4]); // Ay
 AN[5]= smooth(((Analog_Read(3)*.9)+(AN[5]*.1)), filterVal , AN[5]); // Az

}

int read_adc(int select)
{
  return (AN[select]-AN_OFFSET[select])*SENSOR_SIGN[select];
}
void Analog_Init(void)
{
 ADCSRA|=(1<<int Analog_Read(uint8_t pin)
{
  return analog_buffer[pin];
}

void Analog_Reference(uint8_t mode)
{
      analog_reference = mode;
}
ISR(ADC_vect)
{
  volatile uint8_t low, high;
  low = ADCL;
  high = ADCH;
  analog_buffer[MuxSel]=(high << 8) | low;
  MuxSel++;
  if(MuxSel >=8) MuxSel=0;
  ADMUX = (analog_reference << 6) | (MuxSel & 0x07);
  ADCSRA|= (1<<ADSC);
}
```

ADC

Envío de datos a estación base

```
void printdata(void)
{
      Serial.print("!!!");
      #if PRINT_ANALOGS == 1
      Serial.print("AN0:");
      Serial.print(read_adc(0));
      Serial.print(",AN1:");
      Serial.print(read_adc(1));
      Serial.print(",AN2:");
      Serial.print(read_adc(2));
      Serial.print(",AN3:");
      Serial.print(read_adc(3));
      Serial.print(",AN4:");
      Serial.print(read_adc(4));
      Serial.print(",AN5:");
      Serial.print(read_adc(5));
      Serial.print (",");
      #endif
      #if PRINT_KALMAN == 1
      Serial.print("ANX:");
      Serial.print(ToDeg(xkaldata.angle));
      Serial.print(",ANY:");
      Serial.print(ToDeg(ykaldata.angle));
      Serial.print(",RTX:");
      Serial.print(ToDeg(xkaldata.rate));
      Serial.print(",RTY:");
      Serial.print(ToDeg(ykaldata.rate));
      Serial.print(",BSX:");
      Serial.print(ToDeg(xkaldata.q_bias));
      Serial.print(",BSY:");
      Serial.print(ToDeg(ykaldata.q_bias));
      Serial.print (",");
      #endif
      #if PRINT_RADIO == 1
      Serial.print("RC1:");
      Serial.print(serinp[0]);
      Serial.print(",RC2:");
      Serial.print(serinp[1]);
      Serial.print(",RC3:");
      Serial.print(serinp[4]);
      Serial.print (",");
      #endif
      #if PRINT_CONTROL == 1
      Serial.print("LAT:");
      Serial.print(lat);
      Serial.print(",LON:");
      Serial.print(lon);
      Serial.print (",");
      #endif
      Serial.println("***");
}
```

printdata()

Control Difuso

```
void evalControl(){
  float sumRlat;
  float sumRlon;

  evalMF();
  evalR();

  sumRlat=sum(rlat);
  sumRlon=sum(rlon);

  if(sumRlat!=0){
    lat= (int)(mul(rlat,singlelat) / sumRlat);
  }

  if(sumRlon!=0){
    lon= (int)(mul(rlon,singlelon) / sumRlon);
  }

}

void evalR(){
  int k=0;
  for(int i=0;i<numM;i++){
     rlat[i]= mfvalues[i];
     rlon[i]= mfvalues[i+numM];
  }
}

void evalMF(){
  int j=0;
  for(int i=0;i<numM;i++){
mfvalues[i]=mft(ToDeg(xkaldata.angle),mfroll[j],mfroll[j+1],mfroll[j+2]);
mfvalues[i+numM]=mft(ToDeg(ykaldata.angle),mfpitch[j],mfpitch[j+1],mfpitch[j+2
]);
   j+=3;
    if(j==numM*3)
       j=0;
  }
}
```

evalControl

Extras Control Difuso

```
float andf(float mx, float my, int opcion){
  float opf;
  switch (opcion){
    case 1: opf=min(mx,my);
    break;
    case 2: opf=mx*my;
    break;
    case 3: opf=max(0,mx+my-1);
    break;
  }
  return opf;
}

float mft(float x,float xmin, float xc, float xmax){
  return max(min((x-xmin)/(xc-xmin),(xmax-x)/(xmax-xc)),0);
}

float mul(float x[], float y[]){
  int i=0;
  float resul=0;
  for(i=0;i<numM;i++)
     resul+=x[i]*y[i];
  return resul;
}

float sum(float x[]){
  int i=0;
  float resul=0;
  for(i=0;i<numM;i++)
     resul+=x[i];
return resul;
}
```

extras evalControl

Captura del canal PPM

```
void setup_timer1(){

TIMSK1 &= ~( _BV(TOIE1) | _BV(ICIE1) | _BV(OCIE1A) | _BV(OCIE1B));
TCCR1A &= ~( _BV(WGM11) | _BV(WGM10) );
TCCR1B &= ~( _BV(WGM12) | _BV(WGM13) | _BV(ICNC1));
TCCR1B |= _BV(ICES1);
TCCR1B |= _BV(CS11);
TCCR1B &= ~( _BV(CS12) | _BV(CS10) );
TCCR1A &= ~( _BV(COM1A0) | _BV(COM1A1) | _BV(COM1B0) | _BV(COM1B1));
TIMSK1 |= (1<static unsigned char cserinp;

unsigned int licr;

if(ICR1>10000){
  cserinp=0;
}else{
  serinp[cserinp]=((ICR1/2));
  cserinp++;
}
TCNT1=0;
}
```

setup_timer1

Software Arduino (contexto fuzzy)

```
float mfroll[] = {-359.981  , -15.001, -13.741,
                  -14.0238 ,  -8.62 ,  -5.98 ,
                  -11.7381 ,  -5.9  ,   0.5  ,
                   -5.881  ,  -1.25 ,   2.05 ,
                    1.6905 ,   3.05 , 360.0
                };

float mfpitch[]  = {-360.0  , -9.0   ,   -6.0  ,
                     -8.8  , -4.52  ,    0.52 ,
                     -5.45 ,  0.37  ,    5.2  ,
                     -0.21 ,  5.871 ,    9.171,
                      4.0  ,  9.0   ,  360.0
                };

float singlelat[] = { 970,  1401,  1528,  1652,  1950 };

float singlelon[] = { 970,  1384,  1483,  1576,  1950 };

float mfvalues[10];

float rlat[5];
float rlon[5];

float lat;
float lon;

int numM=5;

#endif
```

contexto fuzzy

Kalman 1

```
#ifndef _KALMAN_H_
#define _KALMAN_H_


struct KALDATA {
  float P[2][2];
  float angle;
  float q_bias;
  float rate;
  float Pdot[4];
  float err;
};

void kalmanInitState(KALDATA p_kd);

void state_update(
    const float         q_m   ,
    KALDATA p_kd,
    float dt

);

void
kalman_update(
    const float         angle_m,
    KALDATA p_kd
);


#endif
```

kalman1

Kalman 2

```
#include "kalman.h"

static float            P[2][2] = {
      { 1, 0 },
      { 0, 1 },
};

float             angle;
float             q_bias;
float             rate;
#endif

static const float      R_angle     = 0.3 ;
static const float      Q_angle     = .001 * 57.2958;
static const float      Q_gyro      = .001 * 57.2958;

void kalmanInitState(struct KALDATA *p_kd){
  p_kd->P[0][0] = 1.0;
  p_kd->P[0][1] = 0.0;
  p_kd->P[1][0] = 0.0;
  p_kd->P[1][1] = 1.0;
  p_kd->angle = 0;
  p_kd->q_bias = 0;
  p_kd->rate = 0;
  p_kd->Pdot[0] = 0;
  p_kd->Pdot[1] = 0;
  p_kd->Pdot[2] = 0;
  p_kd->Pdot[3] = 0;
}

void
state_update(
    float         q_m
    struct KALDATA *p_kd,
    float dt
)
{

      float       q;

    q = q_m - p_kd->q_bias;
      p_kd->Pdot[0] = Q_angle - p_kd->P[0][1] - p_kd->P[1][0];/* 0,0 */
      p_kd->Pdot[1] = 0 - p_kd->P[1][1];        /* 0,1 */
      p_kd->Pdot[2] = 0 - p_kd->P[1][1];              /* 1,0 */
      p_kd->Pdot[3] = Q_gyro;                   /* 1,1 */
      p_kd->rate = q;
      p_kd->angle += (q * dt);
      p_kd->P[0][0] += (p_kd->Pdot[0] * dt);
      p_kd->P[0][1] += (p_kd->Pdot[1] * dt);
      p_kd->P[1][0] += (p_kd->Pdot[2] * dt);
      p_kd->P[1][1] += (p_kd->Pdot[3] * dt);
}
```

kalman2

Kalman 3

```
void
kalman_update(
      float       angle_m,
    struct KALDATA *p_kd
)
{
    float         angle_err;
    float         C_0;
    float         PCt_0;
    float         PCt_1;
    float         E;
    float         K_0, K_1;
    float         t_0;
    float         t_1;
      angle_err = angle_m - p_kd->angle;
        p_kd->err = angle_err;
      C_0 = 1;
      PCt_0 = C_0 * p_kd->P[0][0]; /* + C_1 * P[0][1] = 0 */
      PCt_1 = C_0 * p_kd->P[1][0]; /* + C_1 * P[1][1] = 0 */
      E =   R_angle     + C_0 * PCt_0;    /*    + C_1 * PCt_1 = 0 */
      K_0 = PCt_0 / E;
      K_1 = PCt_1 / E;
      t_0 = PCt_0; /* C_0 * P[0][0] + C_1 * P[1][0] */
      t_1 = C_0 * p_kd->P[0][1]; /* + C_1 * P[1][1]  = 0 */
      p_kd->P[0][0] -= (K_0 * t_0);
      p_kd->P[0][1] -= (K_0 * t_1);
      p_kd->P[1][0] -= (K_1 * t_0);
      p_kd->P[1][1] -= (K_1 * t_1);
      p_kd->angle += (K_0 * angle_err);
      p_kd->q_bias      += (K_1 * angle_err);
}
```

kalman3

Contexto (variables y constantes)

```
#define ser1out 2
#define ser2out 3
#define GRAVITY 101
#define Accel_Scale(x) x*(GRAVITY/9.78)
#define Gyro_Gain 2.5
#define Gyro_Scaled(x) x*((Gyro_Gain*PI)/360)
#define G_Dt(x) x*.02
#define ToRad(x) (x*PI)/180.0
#define ToDeg(x) (x*180.0)/PI

#define PRINT_ANALOGS 0
#define PRINT_KALMAN  0
#define PRINT_CONTROL 0
#define PRINT_RADIO   0

int serinp[6];

ServoTimer2 ciLat;
ServoTimer2 ciLon;

float SENSOR_SIGN[]={-1,-1,-1,1,1,-1};

int long timer=0;
int long timer24=0;
int AN[8];
int AN_OFFSET[8];
int EX[8];

float dt; // = .016384;

KALDATA xkaldata;
KALDATA ykaldata;

float factorControl =  0.3;
float factorPiloto  =  0.7;
```

contexto

Referencias

1.- Principio de funcionamiento del helicóptero

1.- Modelado y control de helicópteros autónomos. Béjar, M. y Ollero, A.

2.- How Stuff Works. http://www.howstuffworks.com/

3.- Wikipedia, la enciclopedia libre. http://es.wikipedia.org/

4.- Sky Technologies. http://www.skytechnologies.net/

5.- ShenZhen ArtTech LTD. http://www.art-tech.cn/english/index.asp

6.- System identification of small-size unmanned helicopter dynamics. Bernand Mettler, Mark B. Tischler, Takeo Kanade.

7.-System identification modeling of a model-scale helicopter. Bernand Mettler, Mark B. Tischler, Takeo Kanade

2.- Instrumentación

1.- Wikipedia, la enciclopedia libre. http://es.wikipedia.org/

2.- ST Microelectronics. http://st.com/gyroscopes

3.- SparkFun electronics. http://sparkfun.com

4.- Hyndman Rob, et al. Forecasting with Exponential Smoothing: The State Space Approach. 2008. Springer-Verlag. Alemania pp 9-14

5.- Atmel AVR. http://www.atmel.com/

6.- Welch, Greg. Bishop Gary. An Introduction to the Kalman Filter. Julio 2006. Universidad del Norte de Carolina. < http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf >. Pp. 1 - 16. [Consulta: 1 abr 2010].

7.- Pycke, Tom. Kalman filtering of IMU data. Mayo 2006. < http://tom.pycke.be/mav/71/kalman-filtering-of-imu-data >. [Consulta 7 abr 2010].

Chan Yupo. Location, transport and land-use: modelling spatial-temporal information. 2005. Springer-Verlag. Alemania. Pp 673 - 676

3.- Control

1.-Chahuara Q, J Carlos. Control neuro-difuso aplicado a una grúa torre. Universidad Nacional Mayor de San Marcos. Perú. <http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Ingenie/chahuara_qj/Cap2.pdf> [Consulta 28 mar 2010].

2.-Bonissone, Piero P. Adaptive Neural Fuzzy Inference Systems (ANFIS): Analysis and Applications. Schenectady, NY USA. 1997. < http://www.rpi.edu/~bonisp/fuzzy-course/99/L9/ANFIS.pdf > [Consulta: 21 mar 2010].

3.- Kovacic Zdenko, Bogdan Stjepan. Fuzzy Controller Design Theory and Applications. 2006. CRC Press. Pp 9 – 70

4.- The MathWorks. Fuzzy Logic Toolbox 2. 2008 < http://www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf > [Consulta 14 feb 2010]. Pp. 2-2 to 2-27

5.- Demuth Howard, et al. Neural Network Toolbox 6. The MathWorks. 2008. < http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf >. Pp. 2-8 a 2-19 [Consulta: 10 oct 2010].