Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Ingeniería de Computadores

Estudio y Desarrollo de un Sistema de Control de un Cuadricóptero

---

Omar Luque Rodríguez

Septiembre 2014

informatika fakultatea
facultad de informática

RESUMEN

Los *drones* o aeronaves no tripuladas (*UAV,* según las siglas inglesas de *Unmanned Aerial Vehicle*), son uno de los temas tecnológicos más actuales, gracias a los avances logrados en los últimos años, y a su amplia comercialización más allá del sector militar o de grupos minoritarios como los aficionados al aeromodelismo.

Debido a su creciente popularidad, usándose mayoritariamente para grabaciones aéreas, tanto privadas como en eventos públicos, manifestaciones, etc. el Gobierno ha aprobado una ley para regular su uso, organizándolos en categorías en base al peso (por medidas de seguridad), estableciendo la formación necesaria para los pilotos (carné incluido), y declarándolos legalmente como aeronaves, lo que conlleva el no poder sobrevolar núcleos urbanos sin autorización previa.

Dentro del segmento de aeronaves denominadas *drones*, los que más popularidad han alcanzado los últimos años han sido los "Multirotores" (helicópteros de más de 2 rotores de sustentación), y entre ellos mayoritariamente los de 4 hélices (denominados comúnmente "Cuadricópteros"), gracias a su bajo coste, su facilidad de construcción (gracias a su simple mecánica) y su gran maniobrabilidad.

En este proyecto, se han estudiado los fundamentos motrices y matemáticos de los "Cuadricópteros", y se ha desarrollado un sistema de control básico, utilizando un microcontrolador de 16 bits de la gama PIC24H de Microchip, más concretamente el PIC24HJ256GP610A. El sistema de control tiene como objetivo calcular la posición angular de la aeronave, para así facilitar el manejo del *drone* mediante una emisora de radiocontrol estándar (con un mínimo de 4 canales), pudiendo así desarrollar controles automáticos de estabilidad y dirección.

ÍNDICE DE CONTENIDO

ÍNDICE DE IMÁGENES

PLANIFICACIÓN Y GESTIÓN DEL PROYECTO

OBJETIVOS

Aunque el objetivo final del proyecto es lograr un cuadricóptero estable y fácilmente manejable, el desarrollo del mismo es interesante, ya que conlleva el aprendizaje de muchas cosas, sintetizadas en los siguientes objetivos:

- Conocer las Unidades de Medición Inercial (*IMU*), que sensores llevan y como funciona cada uno.
- Profundizar en la programación de microcontroladores PIC.
    - Aprender a usar el módulo I2C por interrupciones.
- Conocer los fundamentos de vuelo de las aeronaves, principalmente de los cuadricópteros.
- Comprender y utilizar la teoría de la Matriz de Cosenos Directores para calcular la orientación espacial de nuestra aeronave.
- Comprender los sistemas de Control de Estabilidad y Dirección, necesarios en los cuadricópteros.
- Comprender y saber ajustar un mando y un receptor de radiocontrol, diseñado específicamente para aeronaves.
- Volar un cuadrícoptero programada y ajustado por uno mismo.

FASES

Para completar con éxito el proyecto, este se divide en tareas que deben ser completadas para su correcto desarrollo. En la siguiente EDT (Estructura de Desglose de Trabajo) se muestran gráficamente:

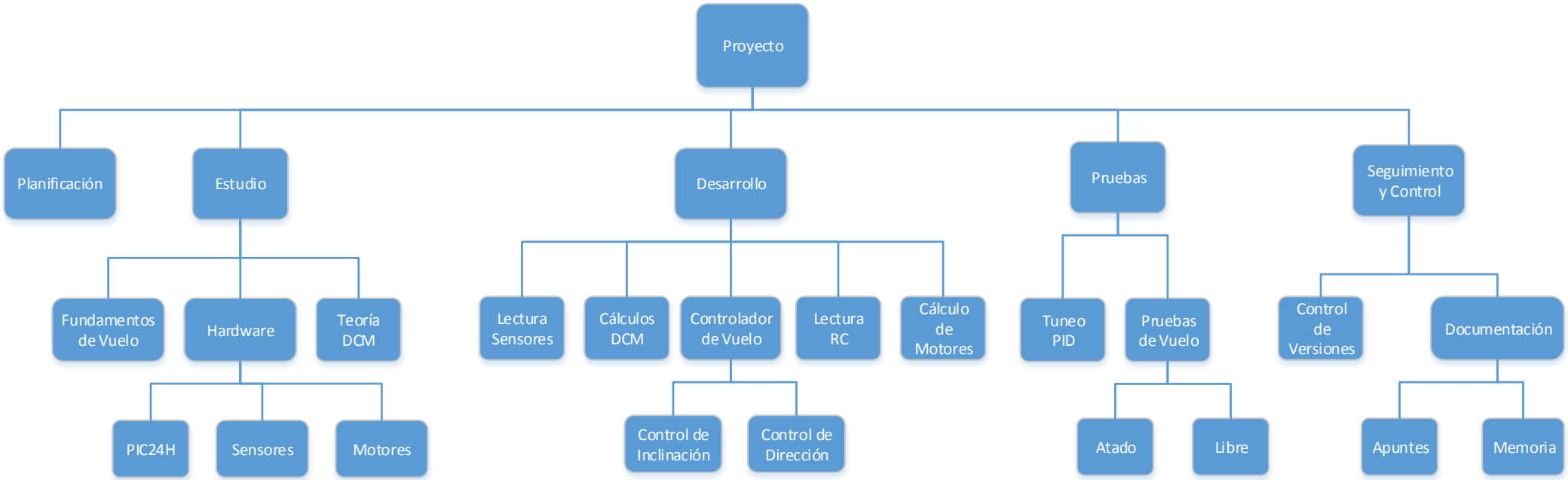
Proyecto
Planificación
Estudio
Desarrollo
Pruebas
Seguimiento y Control
Fundamentos de Vuelo
Hardware
Teoría DCM
Lectura Sensores
Cálculos DCM
Controlador de Vuelo
Lectura RC
Cálculo de Motores
Tuneo PID
Pruebas de Vuelo
Control de Versiones
Documentación
PIC24H
Sensores
Motores
Control de Inclinación
Control de Dirección
Atado
Libre
Apuntes
Memoria

Imagen 1: EDT

La descripción de las tareas principales es la siguiente:

- **Planificación:** Planificación de las etapas del proyecto, y puesta a punto de todo lo necesario para empezar con él.
- **Estudio:** Tarea dedicada a buscar información y entender las diferentes tecnologías, fundamentos y teorías necesarias para el proyecto.
- **Desarrollo:** Diseño e implementación del código del proyecto.
- **Pruebas:** Fase destinada a perfeccionar el funcionamiento del cuadricóptero, en base a el comportamiento mostrado en las pruebas.
- **Seguimiento y Control:** Parte destinada a revisar el desarrollo del proyecto, gestionar las versiones del código, y tomar apuntes para la documentación.

HITOS

Para poder planificar y organizar mejor los tiempos del proyecto, se creado unos puntos de control o hitos, que se alcanzan al completar algunas de las tareas designadas. Los hitos en cuestión son los siguientes:

- **H1:** Fin de la planificación
- **H2:** Fin del estudio de sensores de la *IMU*
- **H3:** Lectura correcta de sensores de la *IMU*
- **H4:** Fin del estudio de la teoría de la Matriz de Cosenos Directores
- **H5:** Comprobación del correcto cálculo de los ángulos de vuelo
- **H6:** Fin del estudio del Sistema de Control de Vuelo
- **H7:** Fin del diseño del Controlador de Vuelo
- **H8:** Fin de la implementación del Controlador de Vuelo
- **H9:** Fin de la implementación de Motores por I2C
- **H10:** Fin de la implementación del Sistema de RC
- **H11:** Pruebas de Vuelo
- **H12:** Fin de la documentación

DIAGRAMA DE GANTT

En el diagrama de Gantt podemos ver cuándo se han realizado las tareas expuestas en la EDT, y que dependencias hay entre ellas:

Id.	Nombre de tarea	Comienzo	Fin
1	Planificación	14/10/2013	20/10/2013
2	**Estudio**	**21/10/2013**	**26/07/2014**
3	Fundamentos de Vuelo	21/10/2013	23/10/2013
4	**Hardware**	**24/10/2013**	**26/07/2014**
5	PIC24H	24/10/2013	28/10/2013
6	Sensores	28/10/2013	10/11/2013
7	Motores	24/07/2014	26/07/2014
8	Teoría DCM	14/04/2014	18/05/2014
9	**Desarrollo**	**12/11/2013**	**25/08/2014**
10	Lectura de Sensores	12/11/2013	07/04/2014
11	Cálculo de ángulos mediante DCM	15/07/2014	29/07/2014
12	**Controlador de Vuelo**	**04/08/2014**	**25/08/2014**
13	Control de Inclinación	04/08/2014	25/08/2014
14	Control de Dirección	04/08/2014	25/08/2014
15	Controlador de Motores	11/08/2014	19/08/2014
16	Lectura de RadioControl	10/08/2014	21/08/2014
17	**Pruebas**	**02/08/2014**	**27/08/2014**
18	Tuneo PID	02/08/2014	25/08/2014
19	**Pruebas de Vuelo**	**20/08/2014**	**27/08/2014**
20	Atado	20/08/2014	27/08/2014
21	**Seguimiento y Control**	**14/10/2013**	**19/06/2015**
22	Control de Versiones	12/11/2013	23/09/2014
23	**Documentación**	**14/10/2013**	**19/06/2015**
24	Apuntes	14/10/2013	25/08/2014
25	Final	28/08/2014	19/06/2015

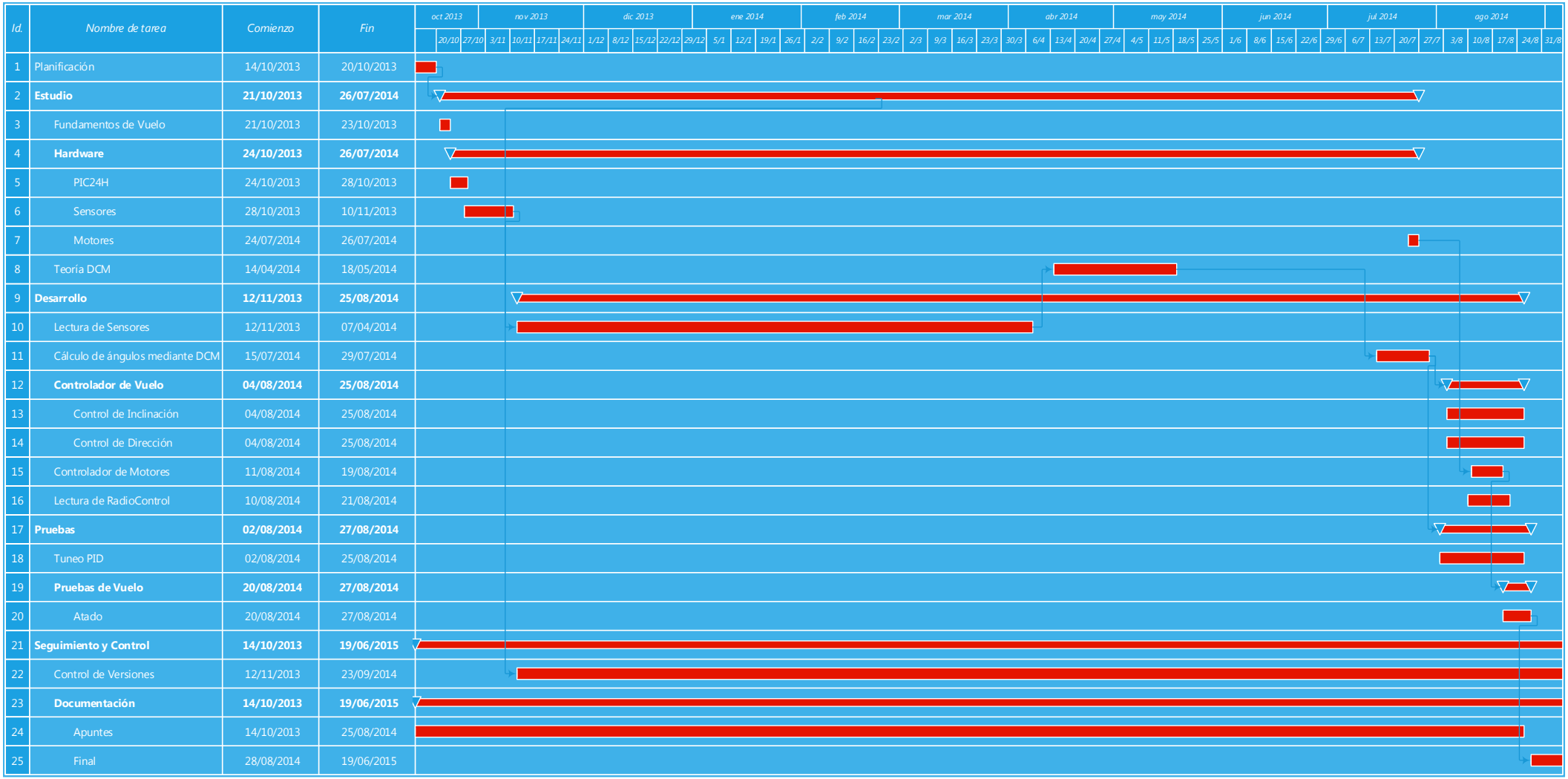

Imagen 2: Diagrama de Gantt

No se esperaba tardar tanto en la tarea de lectura de los sensores, pero daba problemas de lecturas erróneas en el bus I2C y el código estaba bien. Al final se solucionó cambiando el cable de conexión por uno blindado, pues todo era debido a interferencias electromagnéticas externas.
El parón sufrido en el desarrollo de mayo a julio ha sido por motivos externos al proyecto.

PLAN DE COMUNICACIÓN

La comunicación con el director del proyecto se ha planificado de la siguiente manera:

- Para dudas o comentarios generales que no requieren de una respuesta inmediata, la comunicación será por correo electrónico.
- Para dudas que requieran de una respuesta inmediata, la comunicación será por vía telefónica.
- Para el cambio entre fases del desarrollo (cada vez que se alcance uno de los hitos marcados), la puesta al día del trabajo realizado y replanificación de la siguiente tarea a realizar se hará mediante una reunión presencial con el director.

PLAN DE RIESGOS

Los principales riesgos detectados son los siguientes:

- Pérdida total o parcial del código desarrollado.
- Rotura de alguna pieza del cuadricóptero.

Y el plan de contingencia para los dos casos anteriores consiste en:

- Llevar un control de versiones mediante Git, y crear un repositorio remoto para su seguridad.
- Tener repuestos de las piezas más débiles, como las hélices, los cables, los motores y el microcontrolador.

GESTIÓN DE CAMBIOS

Se ha acordado que el alumno puede realizar los cambios necesarios en el código, pero siempre es mejor consultarlos antes con el director.

Una vez realizados los cambios debe comunicárselos al director, bien sea por correo electrónico o en persona en la siguiente reunión.

INTRODUCCIÓN A LOS CUADRICÓPTEROS

DEFINICIÓN

Un Cuadricóptero o Cuadrirotor, es una aeronave de ala giratoria, propulsada por cuatro rotores. Normalmente éstos se encuentran colocados simétricamente, formando cada uno un vértice de un cuadrado imaginario.

Al ser un tipo de helicóptero, tanto la sustentación como la propulsión se basan en el aire impulsado por las hélices de sus rotores. En cambio, a diferencia de los helicópteros que varían el ángulo de incidencia (inclinación) de las palas de las hélices para maniobrar (tanto las palas del rotor principal como las del rotor de cola), los cuadricópteros (como los demás multicópteros modernos) basan sus maniobras en el cambio de revoluciones de sus distintos motores, usando así hélices con palas de paso fijo que simplifican en gran medida la mecánica de este tipo de aeronaves.

CONFIGURACIONES

Mayormente, hay dos tipos o configuraciones de cuadricópteros, los de tipo **x** (en aspa o equis) y los de tipo **+** (en cruz). Estas denominaciones se deben a lo que consideremos como la parte delantera de la aeronave.

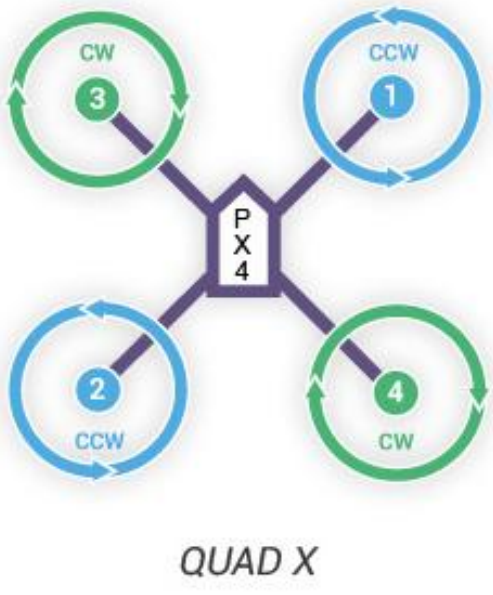

Imagen 3: Configuración en X

En un cuadricóptero configurado en **x**, tendremos dos rotores delanteros y dos rotores traseros (y redundantemente, dos rotores izquierdos y dos rotores derechos). Por hacer una analogía podríamos compararlos con el posicionamiento de las ruedas de un automóvil.
Este tipo de montaje es más popular para llevar cámaras en ellos, pues la cámara no tiene

ningún brazo en su campo de visión (contando que las cámaras suelen apuntar hacia adelante, aunque hay algunas rotatorias).

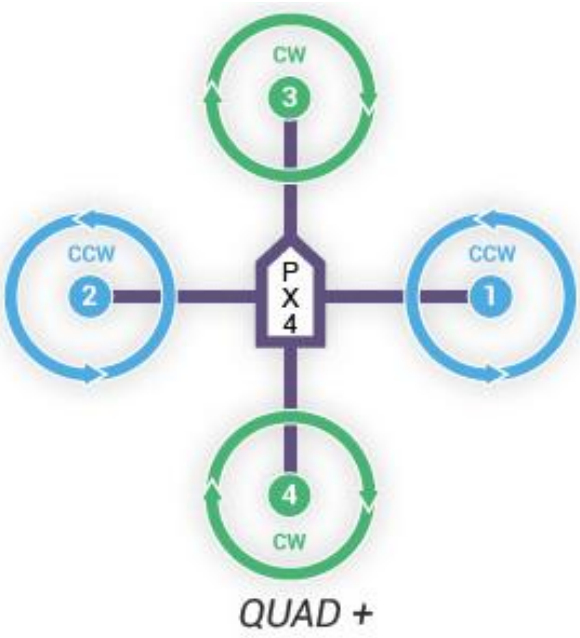

Imagen 4: Configuración en +

Por otra parte, la configuración en **+** conlleva tener un rotor delantero, uno trasero, uno izquierdo y uno derecho. Podemos comparar este tipo de configuración con las partes de un avión siendo el rotor delantero la cabeza, el rotor trasero la cola y los rotores laterales las alas del avión. Este tipo de montaje es más popular entre los aficionados al aeromodelismo y las acrobacias, sobre todo para los que tienen aviones de radiocontrol por su similitud con estos, y por ser más sencillo de programar y entender.

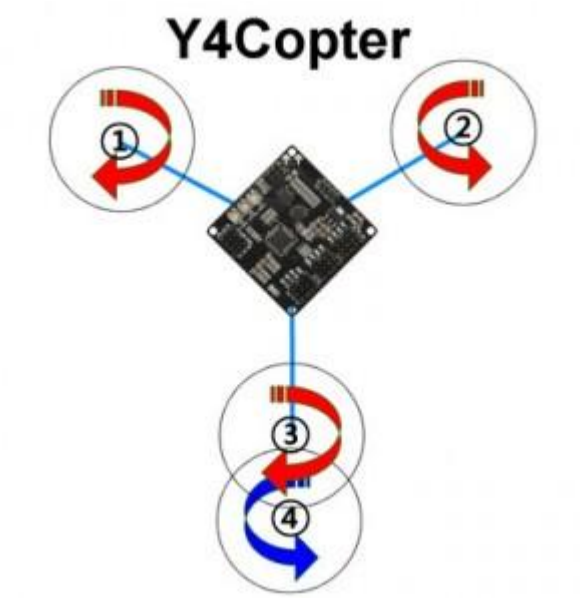

Imagen 5: Configuración Y4

Últimamente, han aparecido otros dos tipos de cuadricópteros, cuyos rotores no están colocados formando un cuadrado, los del tipo Y4 y los VTail. Por definición son cuadricópteros (pues son aeronaves de ala móvil y propulsadas por cuatro rotores), pero su comportamiento y forma es similar al de los tricópteros (multicópteros de 3 rotores), aunque aumentan la estabilidad de estos últimos gracias a tener un número par de rotores.

La configuración Y4 consiste en una estructura en forma de Y, teniendo en su parte delantera dos rotores (uno en cada punta de los brazos de la Y) y otros dos rotores coaxiales en la cola (dos rotores sobre el mismo eje vertical, uno encima del otro).
Esta forma es una evolución de los tricópteros convencionales pero evitando el servo necesario para el giro sobre el eje vertical (guiñada o *yawing*), usando los dos rotores coaxiales, girando en dirección contraria el uno del otro. Con esto se logra alrededor de 1/3 más de fuerza de sustentación (pues el peso no varía mucho al ser la estructura similar al tricóptero), y una mayor estabilidad y fiabilidad, pues no se depende de ningún servo.

Imagen 6: Configuración VTail

La configuración VTail corresponde a una mezcla entre los cuadricópteros en **x** y los tricópteros. Como los cuadricópteros, tiene dos rotores delanteros y dos traseros, pero estando los delanteros colocados cada uno sobre un brazo (como en un la configuración **x** convencional), los traseros se encuentran sobre un único brazo trasero (a modo de cola) colocado uno a cada lado de su extremo, he inclinados alrededor de él en un ángulo determinado.
Esta configuración no es muy popular, debido a que es menos eficiente que la Y4 (puesto que el aire propulsado por los rotores de cola no se entrecruza y no se destina todo a la sustentación debido a su inclinación) y a que su control de motores es algo más complicado. Pero por otra parte, es una configuración más estable (gracias a la parte de empuje lateral de sus rotores de cola), y su orientación es más reconocible a distancia gracias a su clara distinción entre cabeza y cola.

En este proyecto usaremos una configuración en **+**, porque no pensamos incluir una cámara, porque la programación y comprensión de su funcionamiento es más sencilla, y porque que el cuadricóptero del que disponemos viene preparado para el vuelo en cruz, siendo el brazo del rotor delantero de color rojo y todos los demás negros.

El vuelo de los Cuadricópteros se basa (como hemos mencionado anteriormente) en la sustentación obtenida mediante el impulso del aire por sus hélices, y en las maniobras logradas mediante el cambio de velocidad de sus rotores.

Podemos ver los cuadricópteros como máquinas voladoras con 6 Grados de Libertad, ya que se pueden desplazar y rotar sobre los 3 ejes del espacio tridimensional. Aunque esto sea así, en un instante de tiempo dado, sólo podemos indicarle al cuadricóptero 4 movimientos diferentes.

Como toda aeronave, para conocer su orientación espacial sólo necesitamos tres valores, los **Ángulos de Euler** (o ángulos de vuelo). Estos ángulos son la diferencia de orientación que hay entre el sistema de referencia de tierra (eje X apuntando al Norte, eje Y al Este y eje Z hacia abajo) o *NED*, y el sistema de referencia de vuelo (eje X apuntando a la cabeza del aeronave, eje Y a la derecha, y eje Z, perpendicular a ambos, apuntando hacia la parte de abajo del aeronave).

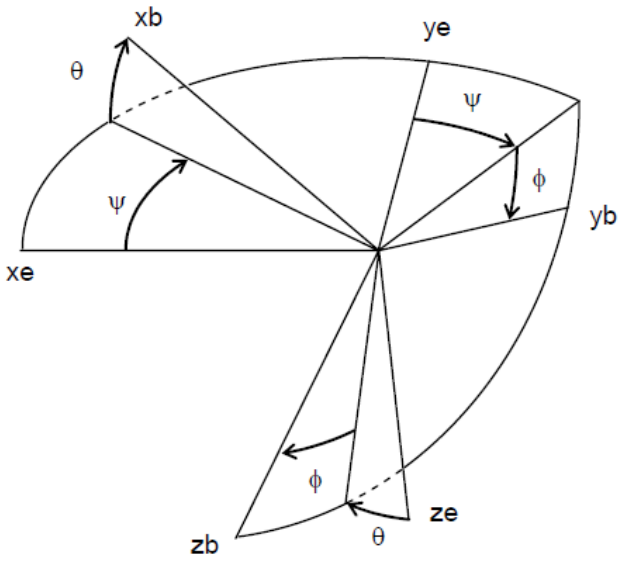

Imagen 7: Ángulos de Euler

Para entender los denominados ángulos de vuelo, supongamos una situación inicial en la que los dos sistemas de referencia (tierra y vuelo) estas alineados.

El ángulo Yaw (ψ) es el ángulo girado sobre el eje Z (de ambos sistemas pues están alineados), logrando girar los ejes X e Y del sistema de referencia de vuelo respecto al de tierra. Esto nos lleva a tener los ejes X_{v1} e Y_{v1}.

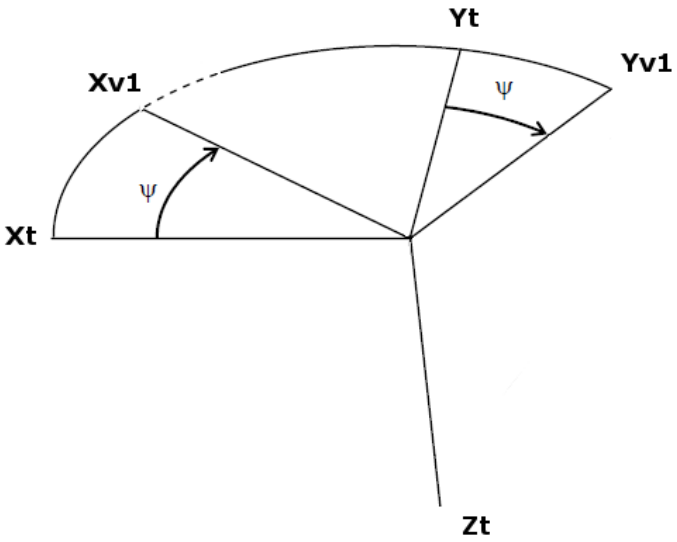

Imagen 8: Giro en ángulo Yaw

Ahora procedemos a girar sobre el eje Y_{v1}, dando como resultado un determinado Pitch (θ) y logrando girar los ejes X y Z del vuelo, dando como resultado los ejes X_{v2} y z_{v2}.

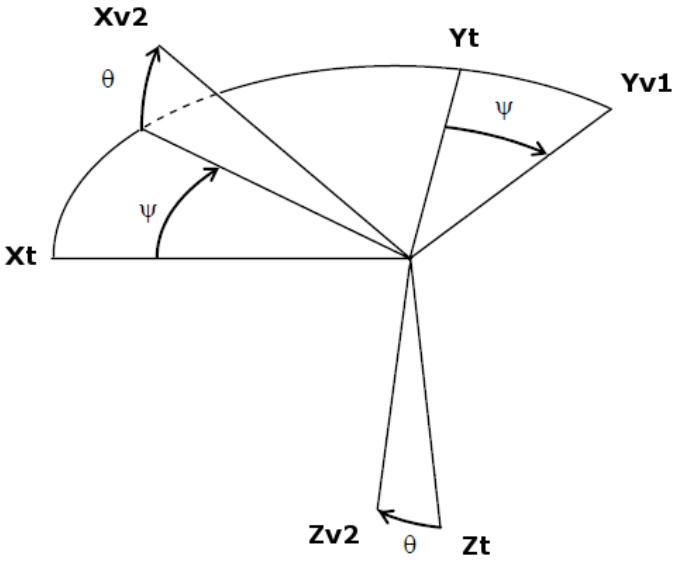

Imagen 9: Giro en ángulo Pitch

Por último giramos alrededor del eje X_{v2} dando como resultado un determinado Roll (ϕ), y los ejes Y_{v3} y z_{v3}.

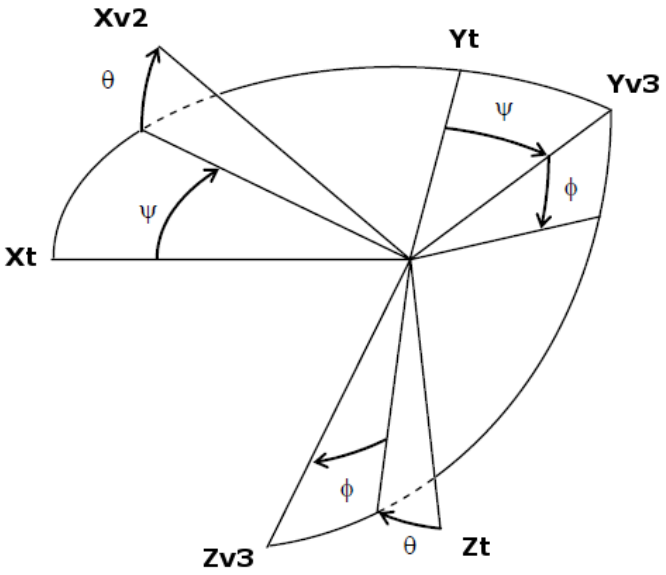

Imagen 10: Giro en ángulo Roll

Las cuatro maniobras básicas de un cuadricóptero son las siguientes:

- Desplazamiento vertical:

  Como su nombre indica, consiste en la traslación sobre el eje Z del espacio tridimensional (el eje vertical). Se logra incrementando o disminuyendo la velocidad de todos los rotores a la vez, siempre y cuando mantengamos la misma velocidad de rotación en los cuatro rotores.

  Para elevar el cuadricóptero se requiere incrementar la rotación, y para descenderlo, disminuirla.

- Alabeo:

  Esta maniobra consiste en la rotación sobre el eje longitudinal de la aeronave (eje que va de la parte delantera a la parte trasera del cuadricóptero).

  En nuestro caso (configuración en **+**) se logra aplicando un determinado incremento en la velocidad del rotor contrario al lado hacia el que se quiere girar, y disminuyendo el mismo nivel de velocidad en el rotor contrario (el rotor del lado hacia el que queremos girar).

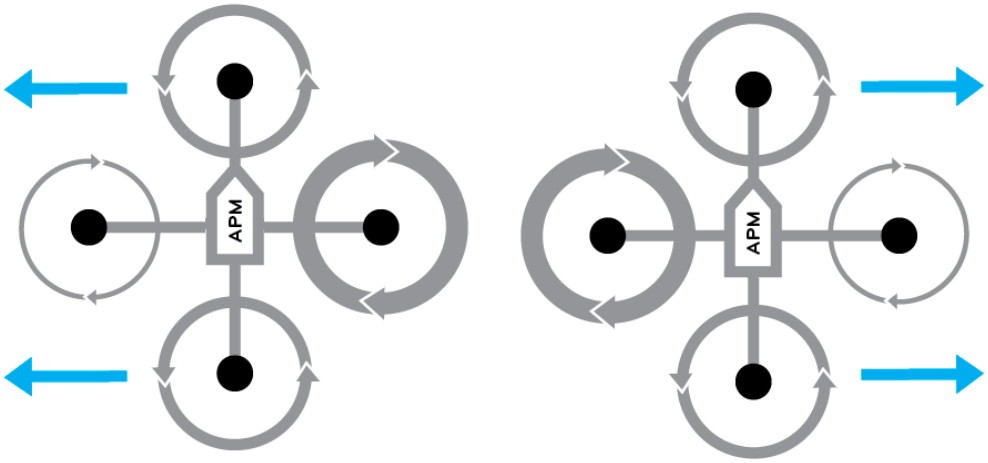

Imagen 11: Desplazamiento Lateral

- Cabeceo:

Esta maniobra consiste en la rotación sobre el eje transversal de la aeronave (eje lateral, que va de izquierda a derecha de la aeronave, aunque en el caso de un cuadricóptero al ser simétrico, sólo se distingue del eje longitudinal por lo que consideremos como parte delantera del aparato), o lo que es lo mismo, elevar o bajar la cabeza (parte delantera) del cuadricóptero.

En nuestro caso (configuración en **+**) se logra elevar la cabeza incrementando en un nivel determinado la velocidad del rotor delantero y disminuyendo en el mismo nivel la del trasero; y viceversa para hacerla descender.

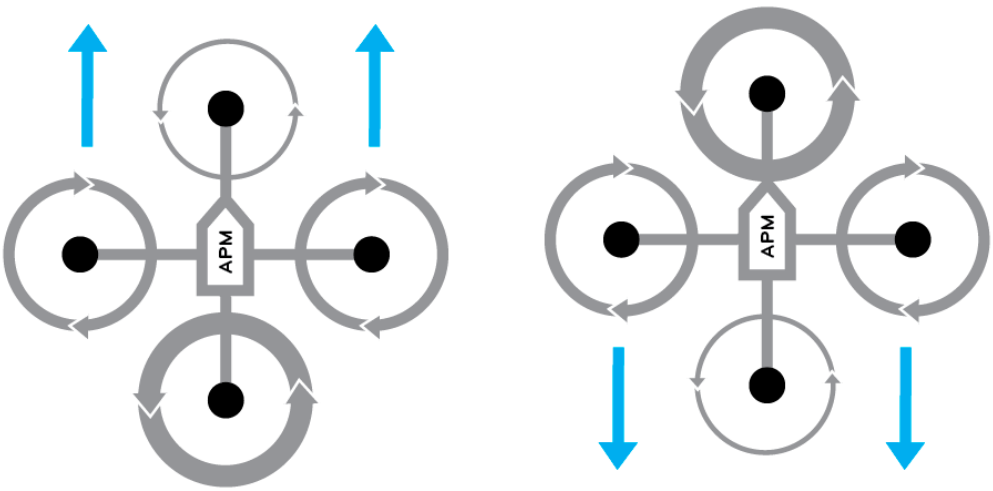

Imagen 12: Desplazamiento Longitudinal

- Guiñada:

Esta maniobra consiste en rotar la aeronave sobre el eje vertical, haciéndola girar sobre sí misma a izquierda o derecha.

En nuestro caso (y en el de la configuración en x también), se logra esta maniobra gracias a que los rotores del eje longitudinal giran en un sentido y los del eje lateral en otro. Mientras giran a la misma velocidad, el efecto torque (o momento de fuerza) a derechas creado sobre el eje por los motores a izquierdas, es contrarrestado por el torque a izquierdas creado sobre el eje por los motores a derechas, logrando un momento de fuerza nulo sobre el conjunto del cuadricóptero.

Por eso, para girar a la derecha, debemos incrementar la velocidad de los rotores que giran a la izquierda (pues los ejes, y con ellos el conjunto, tienden a girar al lado contrario) y disminuir en la misma medida los rotores que giran a la derecha. Y viceversa para girar a la izquierda.

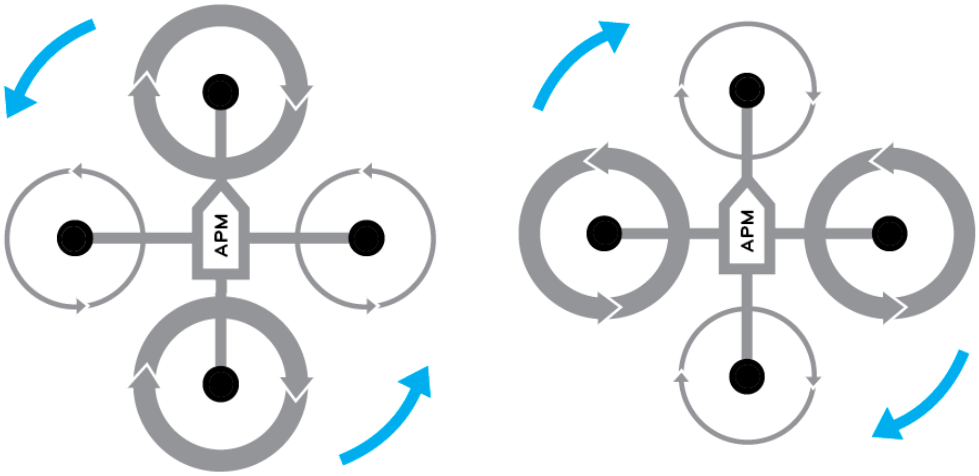

Imagen 13: Guiñada o *Yawing*

Estos son los tres giros y el desplazamiento que podemos indicarle al cuadricóptero que haga en un determinado instante de tiempo, pero ¿cómo hacemos para desplazarnos hacia adelante, atrás o hacía los lados? No tenemos más que mantener inclinado el aparato durante un espacio de tiempo para lograrlo.

Para el desplazamiento sobre el eje longitudinal (adelante/atrás), debemos mantener un cabeceo en el sentido que queramos desplazarnos (cabecear hacía abajo para avanzar, hacía arriba para retroceder), y volver a una posición horizontal cuando deseemos detener el desplazamiento. Lo mismo ocurre con el desplazamiento sobre el eje transversal (izquierda/derecha), aunque en este caso, lo que debemos mantener es un alabeo a izquierdas o a derechas, depende del sentido en el que queramos desplazarnos.

Con todas las maniobras anteriormente comentadas, podemos desplazar el cuadricóptero con total libertad por el aire.

HISTORIA

A principios del siglo XX se idearon los primero modelos de cuadricópteros. Estas naves eran grandes, pilotadas (por supuesto no había tecnología de radiocontrol), y poco eficientes, pues no había motores (ni eléctricos ni de combustión) como los de hoy en día.

A continuación se comentan los distintos proyectos conocidos que llegaron a avanzar algo en el desarrollo de estas formidables máquinas:

- Giroplano de Breguet-Richet (1907):

  Fue el primer proyecto de aeronave de ala rotatoria capaz de elevarse del suelo por sí sola. Diseñada por los hermanos Breguet de Breguet Aviation (registrada a nombre de Louis Breguet) con la ayuda del profesor Charles Richet, esta aeronave fue capaz de elevarse unos 60 cm del suelo por primera vez el 29 de septiembre de 1907. No tenía ningún sistema de control ni de direccionamiento, por lo que era estabilizada por cuatro hombres que la sujetaban.

  El cuerpo estaba formado por cuatro brazos de tubos de acero, tensados por cables, con un rotor en la punta de cada brazo, y dos hélices superpuestas en cada rotor, una en la parte superior y otra en la parte inferior del brazo. Las hélices estaban provistas de cuatro palas cada una, y giraban cuatro en una dirección y las otras cuatro en otra para contrarrestar la rotación generada por el efecto *torque*. Tenía una altura de 3,7 m, un peso de 500 kg estando vacío, y capacidad para cargar al piloto y un poco de combustible, llegando a un peso bruto de 578 kg.

  El piloto se sentaba en el centro, encima del motor de combustión de 34 kW que movía los rotores. En 1908 se mejoró creando el Giroplano Nº 2 que llevaba un motor de 41 kW, y hélices de dos palas en vez de cuatro. Según los creadores consiguieron elevarlo hasta una altura de 1,52 m, pero no era maniobrable, y acabo destruido en un aterrizaje fallido.

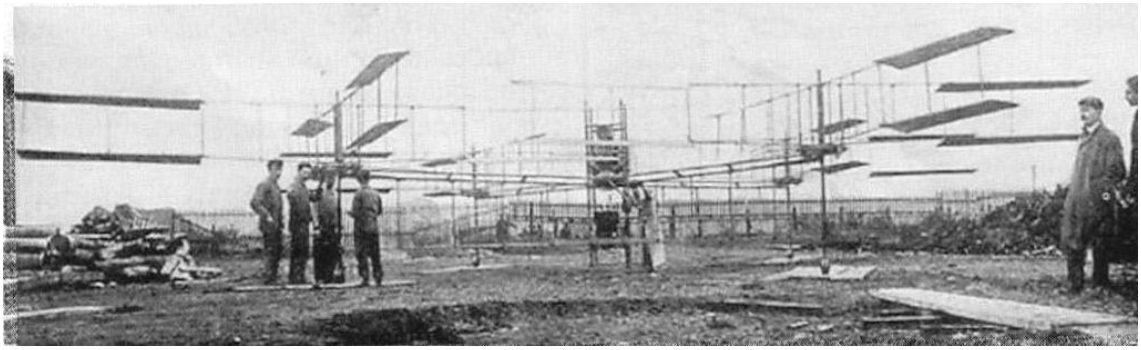

Imagen 14: Giroplano de Breguet-Richet

- Oeminchen Nº 2(1920):

El ingeniero francés Étienne Oeminchen, estuvo experimentando con diferentes diseños de aeronaves de ala móvil, siendo la mejor su Oeminchen Nº 2.

Esta máquina disponía de un cuerpo de tubos de acero, con cuatro brazos (siendo el delantero más largo que los demás), 4 hélices principales de sustentación y otras 8 hélices para el control de estabilidad y dirección. Las hélices eran todas de dos palas, pero las principales no eran de paso regulable mediante unos cables atados a las palas. Todas las hélices eran propulsadas por un motor central de 135 kW.

De las hélices de control, cinco de ellas eran horizontales para mejorar la estabilidad, una vertical en el brazo delantero para dirigir la nave (idea que serviría para el rotor de cola que sirve de timón en los helicópteros actuales), y las dos últimas colocadas en vertical en los brazos laterales a modo de propulsores.

Llevó a cabo unos mil vuelos exitosos, en estableció el primer record de vuelo de una aeronave de ala rotatoria (360 m), y después logró completar una vuelta a un circuito cerrado de 1 km en 7 minutos y 40 segundos, ganando un premio de 90.000 francos. El tiempo de vuelo máximo era de 14 minutos.

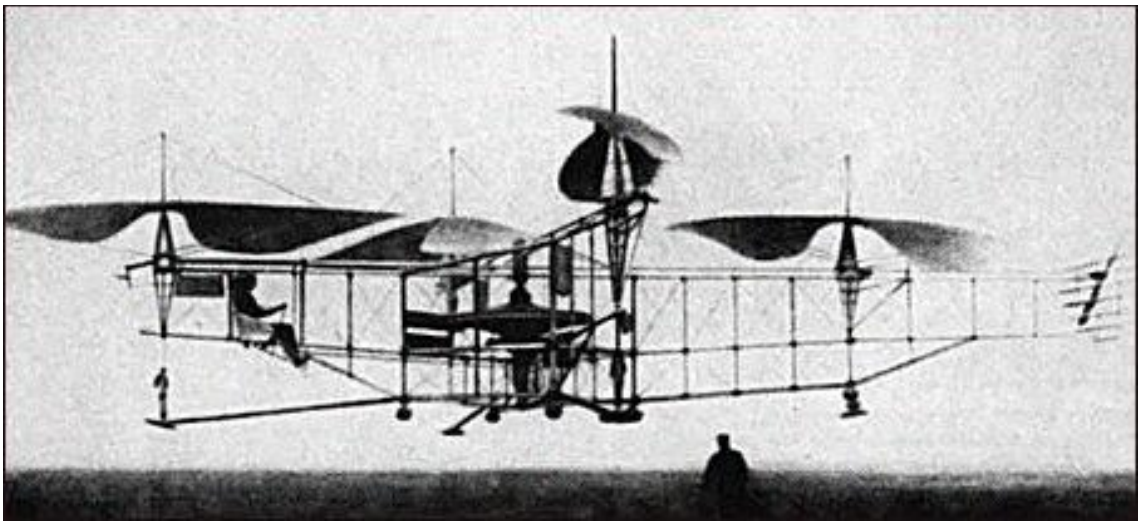

Imagen 15: Oeminchen Nº2 volando

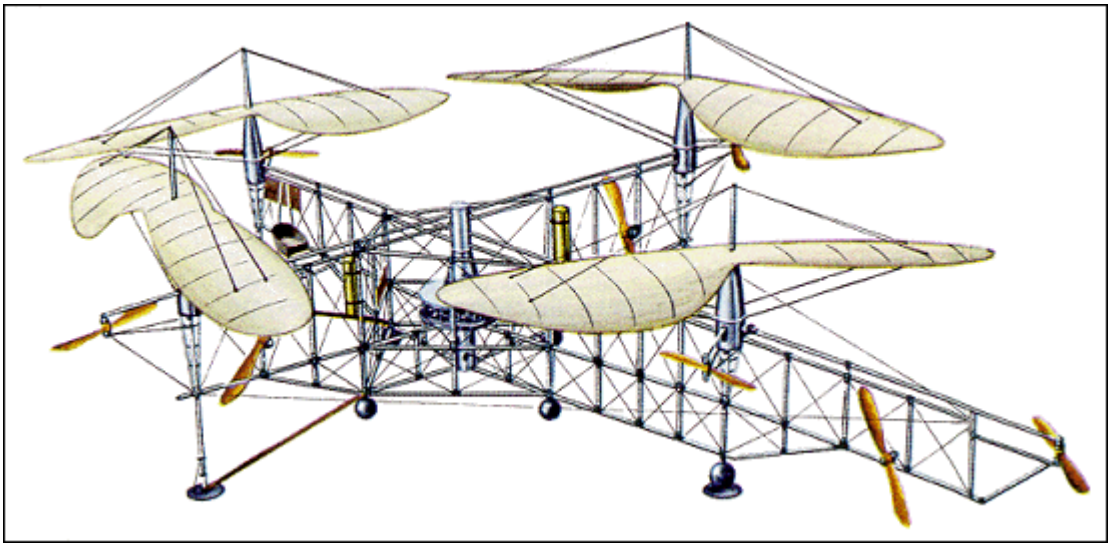

Imagen 16: Modelo 3D del Oeminchen Nº2

- Helicóptero de Bothezat (1922):

Este modelo fue diseñado por el doctor George de Bothezat e Iván Jerome para el ejército estadounidense. Contaba con un cuerpo de tubos de acero con forma de X, cuatro rotores con hélices de seis palas, y dos pequeños propulsores que servían para desplazarse y dirigirlo.

Su primer vuelo fue sorpresivamente exitoso, teniendo en cuenta que fue construido sin prototipos previos. Hizo unos 100 vuelos exitosos y llego a elevarse a una altura de 5 m, pero era muy poco eficiente, complejo y de poca fiabilidad mecánicamente, y bastante complicado de manejar por el piloto, puesto que en estacionario había que evitar continuamente que girase sobre sí mismo.

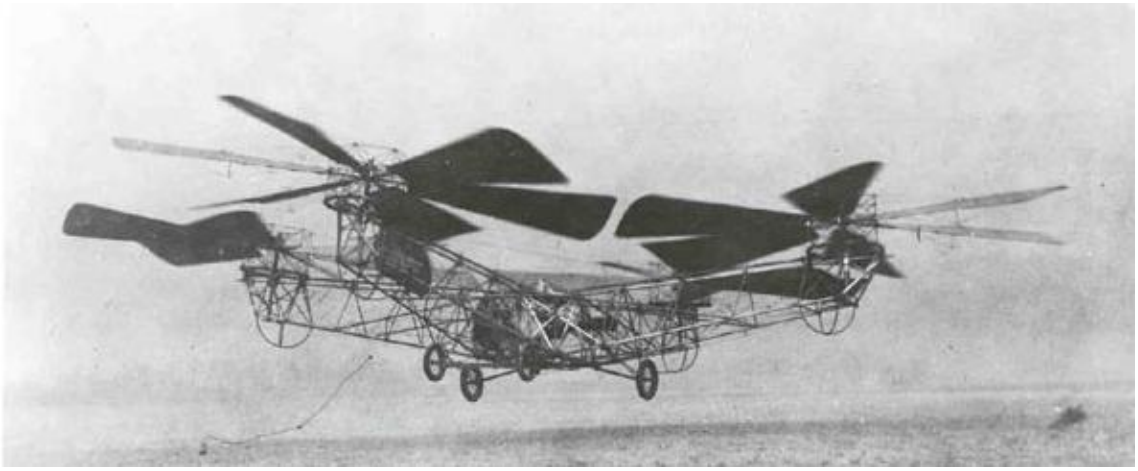

Imagen 17: Helicóptero de Bothezat

- Convertawings Model A Quadcopter(1956):

Este modelo fue un prototipo diseñado con la intención de servir de base para una nueva gama de cuadricópteros militares y civiles en EEUU.

Tenía forma de H rotada (o I), siendo la parte central o cuerpo de una estructura de tubos de acero, y los brazos que soportaban los rotores, de una aleación de aluminio. Los brazos tenían forma de V, siendo más elevados en las puntas que en el centro (donde se unían al cuerpo). Contaba con dos motores de 67 kW, y fue el primer modelo en demostrar la viabilidad de los cuadricópteros tal y como los conocemos, pues no necesitaba rotor de cola, y controlaba todas sus maniobras gracias a la diferencia de velocidad de sus rotores. También fue el primer modelo de cuadricóptero en avanzar en línea completamente recta. Pesaba unos 1000 kg

Era complejo de manejar, pues el piloto tenía que encargarse de controlar la aceleración de los cuatro motores independientemente, y aunque hizo bastantes vuelos exitosos, al no haber encargos civiles ni militares, el proyecto fue cerrado.

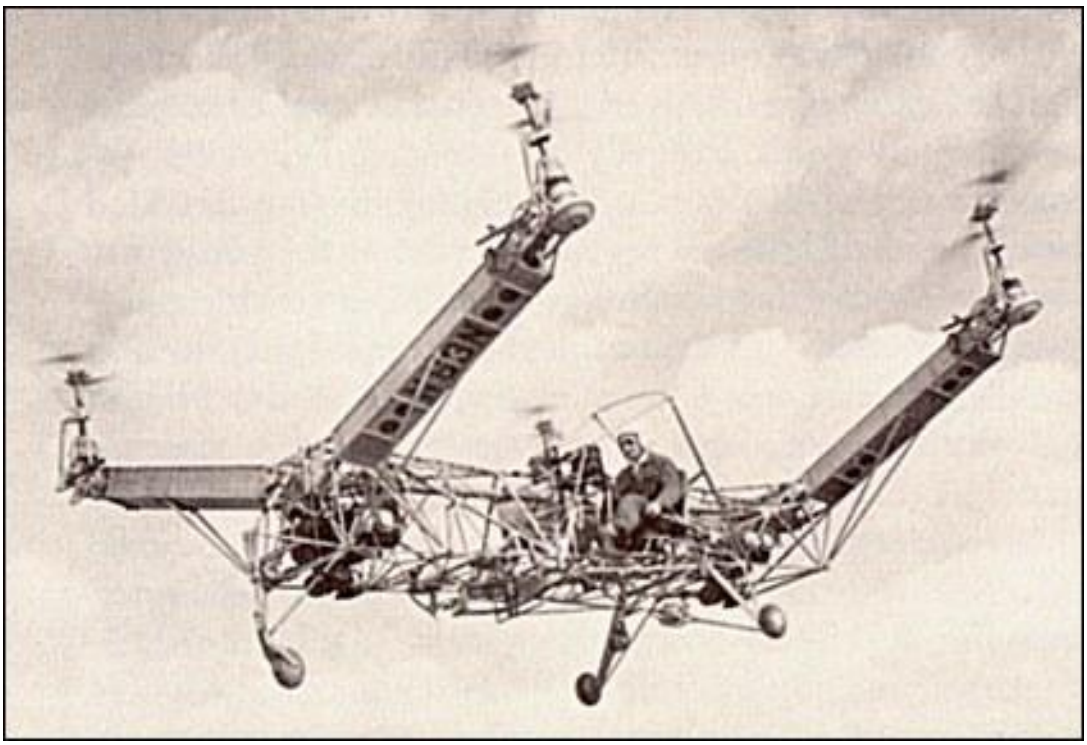

Imagen 18: Convertawings Model A Quadcopter

- Curtiss-Wright VZ-7 (1958):

Este modelo fue diseñado para el ejército estadounidense, como un prototipo de "jeep aéreo" monoplaza. Consistía en un cuerpo rectangular con dos rotores a cada lado (es decir, en configuración X), estando el asiento del piloto y los controles en la parte delantera (por delante de los rotores), y en el centro el depósito de combustible y el motor de 320 kW.

Como los modelos modernos, basaba sus maniobras en el cambio de velocidad de sus rotores, pero era más sencillo de manejar para el piloto que el Convertawings (comentado anteriormente). Sus vuelos demostraron una buena estabilidad y maniobrabilidad, pero no lograron superar los requisitos de altitud ni de velocidad, pues sólo lograron volarlo cómodamente a una altitud de 60 m y a una velocidad máxima de 51 km/h.

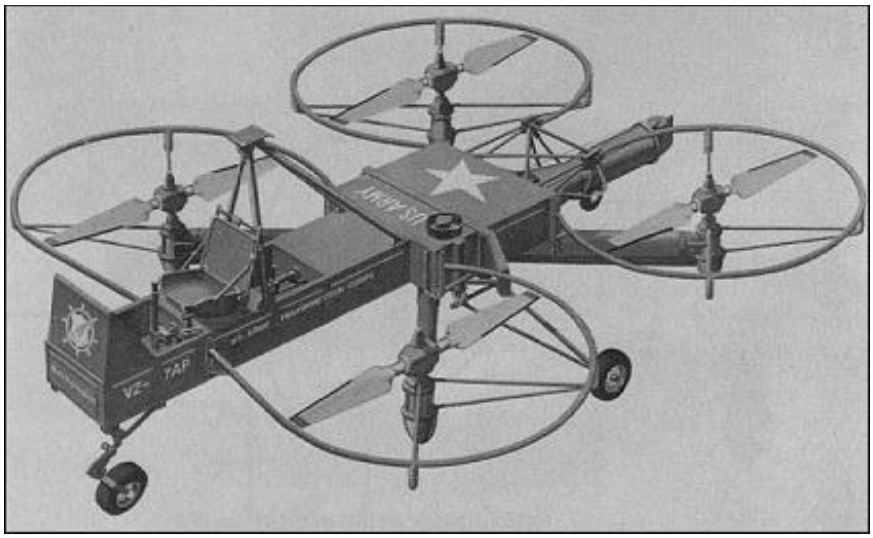

Imagen 19: Modelo 3D del Curtiss-Wright VZ-7

ACTUALIDAD

Desde hace unos años a esta parte, se han desarrollado una gran cantidad de multicópteros (mayoritariamente cuadricópteros) teledirigidos, de una multitud de tamaños que van desde unos pocos centímetros hasta más de un metro de ancho.

Imagen 20: Comparativa de tamaños de cuadricópteros

Esta miniaturización y su amplia difusión se ha conseguido gracias a la miniaturización de la electrónica, a la mejora de la informática de los mismos, a la evolución de las baterías (especialmente las LiPo utilizadas en la mayoría de estos aparatos) y la mejora y miniaturización de los motores eléctricos. Y por supuesto al abaratamiento de todo lo anterior.

Sus principales usos son la toma de imágenes aéreas (fijas o en vídeo), la investigación para futuros desarrollos, el disfrute de los aficionados al radiocontrol… pero cada vez se les están dando más usos como la entrega de paquetes (que Amazon está intentando poner en marcha), o la búsqueda de supervivientes en zonas catastróficas.

En la comunidad de software libre, hay unos cuantos desarrollos para que cualquiera con unos mínimos conocimientos pueda montarse su propio multicóptero en casa (siguiendo la filosofía *DIY*). Estas comunidades desarrollan el software necesario para el sistema de control del cuadricóptero, dejando en manos de los usuarios sólo el montaje físico del mismo, lo que es mucho más sencillo.

Entre las plataformas *open software* nos encontramos las comunidades de AeroQuad, ArduPilot/ArduCopter, OpenPilot… Cada una con sus diferencias, pero la mayoría ofrecen herramientas de software gratuitas y de código abierto (lo que permite reutilizarlo según las necesidades de cada uno), que van desde lo básico que es el controlador de vuelo a instalar en el microcontrolador de nuestro cuadricóptero, hasta software para que nuestro PC sea la "estación base" de nuestros *drones*, pudiendo controlarlos y recibir telemetría (datos, imágenes…). La mayoría de estas plataformas ofrecen su código para poder ser instalado en plataformas de hardware libre, como Arduino, aunque siempre hay placas específicamente diseñadas para su uso como controladores de vuelo (muchas basadas en Arduino).

También suelen tener tiendas online en las que venden los componentes (motores, placa con microcontrolador y sensores, hélices…) para que puedas construírtelo tu mismo sin andar buscando piezas de un lado u otro.

Imagen 21: Cuadricóptero hecho en casa (DIY)

En la actualidad hay muchos cuadricópteros comerciales: desde los más básicos y pequeños que podríamos categorizar como "juguetes" a radiocontrol, otros más avanzados tecnológicamente como el Parrot AR.Drone, con cámara integrada, para ser controlado y visualizar lo que "ve" con su cámara desde nuestro Tablet o Smartphone, u otros destinados al mercado profesional para

toma de imágenes aéreas como el DJI Phantom, que incluye navegación GPS, sistema automático de aterrizaje y, dependiendo de la configuración: cámara integrada, o bahía para colocar una cámara GoPro obteniendo imágenes de mucha más calidad.

Imagen 22: DJI Phantom 2 Vision Plus

Por supuesto, también hay desarrollos militares (aunque de esos no hay tanta información) como el Bell Boeing Quad TiltRotor, diseñado con intención de servir de vehículo de carga (para tropas o material), con un tamaño similar al de un Hércules C-130, pudiendo albergar unos 110 paracaidistas o 150 soldados de a pie. También sería capaz de cargar ocho pallets 463L cuando se destinara para transporte de materiales, pudiendo con un peso total de 16,000 kg. Por lo que sabemos todavía está en fase de prototipo.

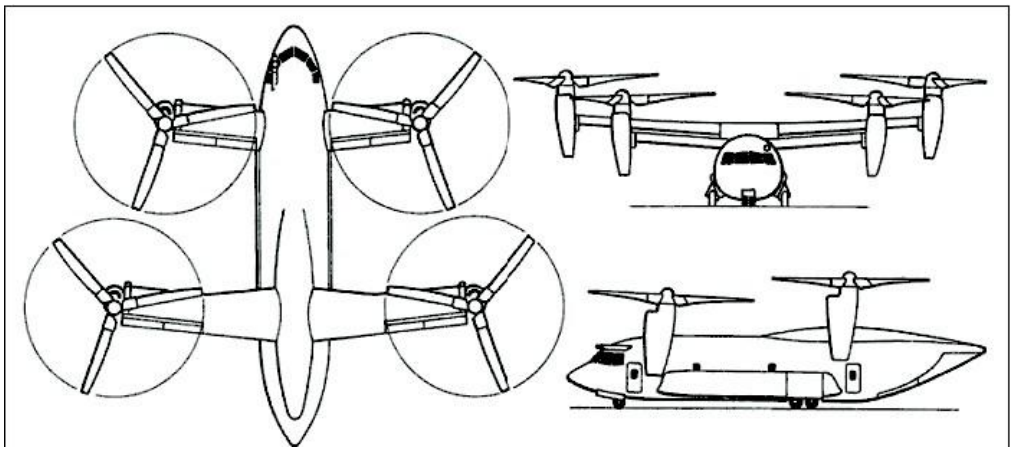

Imagen 23: Planos del Bell Boeing Quad TiltRotor

TECNOLOGÍA UTILIZADA

En este capítulo describimos la tecnología utilizada para el funcionamiento de nuestro cuadricóptero, desde la parte física (chasis, motores, hélices, batería…), hasta el software desarrollado para su control y las herramientas de desarrollo utilizadas.

HARDWARE

En lo que al apartado físico de nuestro cuadricóptero se refiere, la mayor parte de las piezas proceden del fabricante alemán MikroKopter (MK) puesto que hemos reutilizado un modelo suyo añadiéndole como "cerebro" del mismo un microcontrolador PIC de 16 bits y un sistema de radiocontrol ajeno al mismo, en este caso de la marca FUTABA.

- Chasis:

El chasis de nuestro cuadricóptero está formado por las piezas del modelo Quadrokopter XL de MK. A continuación se describen sus características:

-Brazos de aluminio de 290mm de longitud (el delantero rojo y los otros tres negros).

-Platos centrales de fibra de vidrio (dos, superior e inferior).

-Patas de aterrizaje, plásticas, modelo MK HiLander-26 (cuatro, una por brazo).

-Sus respectivos tornillos, tuercas y arandelas para mantenerlo todo sujeto.

- Motores:

Los motores eléctricos que utilizamos son también los del modelo Quadrokopter XL, los motores eléctricos MK3638. Estos motores son *Brushless Outrunner*. *Brushless* significa "sin escobillas", lo que conlleva un mejor rendimiento por la falta de rozamiento con las escobillas de los motores convencionales. *Outrunner* significa "rotor externo", lo que quiere decir que lo que rota es toda la carcasa exterior (menos la base), ofreciéndonos mayor *torque* (aunque menos revoluciones que los *Inrunner* en los que sólo rota el eje) lo que nos permite montar hélices mayores (que a su vez son más eficientes).

Estos motores son trifásicos y necesitan un complejo sistema electrónico de control y alimentación. Estos sistemas se denominan *ESC* (por las siglas inglesas de *Electronic Speed Control*), y gracias a la miniaturización de la electrónica, hoy en día, caben en la palma de la mano. En el apartado siguiente veremos los *ESC* utilizados en nuestro cuadricóptero.

Las características principales de estos motores son las siguientes:

-Corriente de trabajo: 8-17A (más del 75% de eficiencia a estas corrientes)

-Corriente máxima (durante 60 seg): 25 A

-Revoluciones por voltio: 770 rpm / V

-Tamaños de hélice recomendados: 10" a 14"

-Potencia máxima: 350W

-Aceleración máxima: 2200g

-Peso total (sin cable): aprox. 100g

-Dimensiones: 35 mm de altura, 38 mm de diámetro

-Diámetro del eje: 4 mm

- Controladores Electrónicos de Velocidad, *ESCs:*

Los controladores de los motores, son los ESCs BL-Ctrl V2.0 de MK. Como hemos comentado anteriormente, estos controladores electrónicos se encargan de convertir la corriente continua de la batería en corriente alterna trifásica, y de cambiar las fases a suficiente velocidad para que el motor rote.

Estos cuatros *ESCs* van montados sobre la placa *Quadro XL power distribution* de MK, que tiene la forma perfecta para encajar los *ESCs* en el chasis, apuntando cada uno a su respectivo brazo. Las características principales de estos controladores son las siguientes:

-Microcontrolador ATMEGA168 integrado con 16kB de Flash.

-Voltaje de entrada: 11,1V a 18,5V (LiPos 3s a 5s).

-Corriente sostenida: 35A, máxima 40A.

-Resolución: 11 bits (2048 velocidades)

-Dos LEDs indicadores de estado.

-Medición de voltaje y de temperatura.

-Protocolos de control: PPM, I2C y serie.

-8 direcciones internas diferentes (controlables puenteando 3 pines).

En nuestro caso los controlaremos por I2C, refrescándolos a una frecuencia de 500Hz (necesaria para los *ESCs*), y seleccionándolos mediante las cuatro primeras direcciones posibles (7 bits siguiendo el protocolo I2C, de la 0x28 a la 0x2B). Las direcciones se consiguen puenteando los pines ADR1, ADR2 y ADR3 como se muestra en la tabla siguiente:

Motores\Pines	1	2	3
Nº1(Frontal)	Libre	Libre	Libre
Nº2(Trasero)	Libre	Puente	Puente
Nº3(Derecho)	Puente	Puente	Libre
Nº4(Izquierdo)	Puente	Puente	Puente

Tabla 1: Puentes para la dirección de motores

Siguiendo el protocolo I2C, a estos controladores hay que enviarles la velocidad deseada en un Byte, es decir, de 0 a 255.

- Mando y receptor RC:

Para el manejo del cuadricóptero hemos utilizado el mando de radio control Futaba T7C. Es un mando de 2.4 GHz, con 7 canales, extremamente configurable y con varios modos de vuelo precargados (tanto para avión como para distintos tipos de helicópteros). Cuenta con 2 *sticks* analógicos con capacidad de *trimado* en cada eje, 6 palancas: 2 disponibles en los canales 5 y 7, una con tres posiciones y otra analógica. También cuenta con un potenciómetro rotatorio en el canal 6.

Imagen 24: Mando Futaba T7C

En este proyecto usaremos el mando en configuración de avión (para usar los 4 canales de los *sticks*, no 3 como en los modos de helicóptero), usando los *sticks* en Modo 2, y el interruptor (palanca) G para activar y desactivar el modo de calibrado del mando. Los controles en Modo 2 de un mando radiocontrol son los siguientes:

Imagen 25: Modo 2 de RC

Como receptor de la señal del mando usamos el módulo Futaba R617FS, totalmente compatible con nuestro mando. Este módulo recibe de 2.4 GHz codificada del mando, y la traduce a pulsos PWM estándar (pulsos de 1-2ms), que decodificamos con nuestro microcontrolador, el PIC24H que usamos como "cerebro" del cuadricóptero.

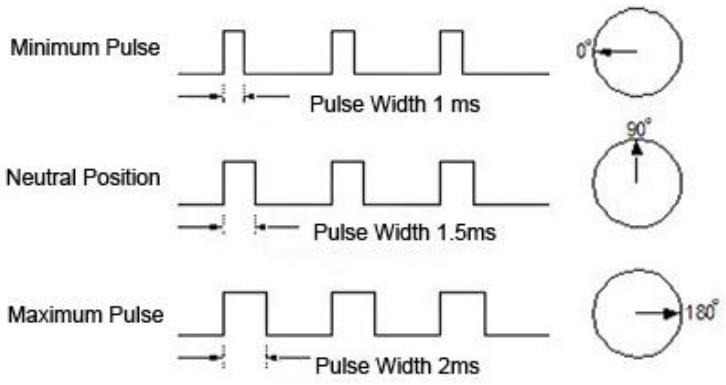

Imagen 26: PWM estandar usado en RC

- Sensores, *IMU:*

La Unidad de Medición Inercial o *IMU* (*Inertial Measurement Unit)*, es el dispositivo compuesto de diferentes sensores que se encarga de medir la velocidad, la posición y la aceleración (o fuerza gravitacional) sufrida por nuestro cuadricóptero. Estos datos nos sirven para calcular su posición espacial.

En nuestro caso hemos utilizado la placa 9DOF Stick de Sparkfun, que contiene un acelerómetro, un giróscopo y un magnetómetro (o brújula), todos tridimensionales y conectados al bus I2C para su configuración y uso.

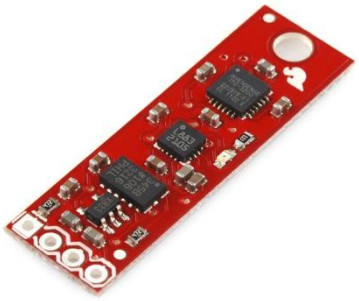

Imagen 27: 9DOF Stick de Sparkfun

Las características principales de los sensores son las siguientes:

-Acelerómetro ADXL345:

-Resolución ajustable de 10 a 13 bits.

-Lectura máxima: ±16g (con 13 bits).

-Sensibilidad de 4mg/LSB, permitiendo detectar cambios menores a 1° en la inclinación.

-Datos de salida formateados a 16bits en Complemento a 2.

-Soporta I2C a 100kHz y a 400kHz, trasmisiones simples y múltiples.

-Giróscopo ITG-3200:

-16 bits de resolución.

-Lectura máxima: ±2000°/s.

-Sensibilidad de 14,375 LSB/(°/s).

-Frecuencia de muestreo ajustable de 1 a 8kHz.

-Frecuencia de salida ajustable de 8.000 a 3,9 muestras por segundo.

-Datos de salida formateados a 16bits en Complemento a 2.

-Soporta I2C a 100kHz y a 400kHz, trasmisiones simples y múltiples, *SlewRate* necesario.

-Sensor de temperatura integrado.

-Magnetómetro HMC5883L:

-12 bits de resolución.

-Lectura máxima: ±8 Gauss.

-Sensibilidad de 2 miligauss/LSB, en grados de 1° a 2°.

-Frecuencia de muestreo máxima de 160 Hz.

-Varias muestras para calcular el valor de salida, ajustable de 1 a 8 muestras.

-Frecuencia de salida ajustable de 0.75Hz a 75Hz (por defecto 15Hz).

-Datos de salida formateados a 16bits en Complemento a 2.

-Soporta I2C a 100kHz y a 400kHz, trasmisiones simples y múltiples.

- Microcontrolador:

Como controlador de vuelo del cuadricóptero, usamos un microcontrolador PIC de 16 bits, en concreto el PIC24HJ256GP610A, con el que ya estamos familiarizados gracias a la asignatura *Diseño de Sistemas Integrados*.

Se ha diseñado un sencillo controlador de vuelo para esta arquitectura, basándonos en la teoría básica de control de los cuadricópteros, disponible gracias a las comunidades de código abierto mencionadas con anterioridad (AeroQuad, ArduCopter…). Todos los demás componentes comentados anteriormente están conectados al microcontrolador, de manera que todo el proceso de vuelo está controlado por el mismo.

Para la programación del microcontrolador y la depuración del código se ha utilizado el *debugger* REAL ICE y la placa de desarrollo Explorer 16, ambos del fabricante Microchip.

Imagen 28: Placa de desarrollo Explorer 16 junto a REAL ICE

Las características que más nos importan del microcontrolador son las siguientes:

- Velocidad de CPU: 40 MIPS.

-256 KB de memoria de programa.

-2 buses I2C.

-2 módulos RS232.

-8 módulos de captura de entradas (*Input Capture*).

-9 *timers* de 16 bits o 4 de 32 bits.

-*Prescaler* ajustable para los *timers*.

-Capacidad de programar casi todo mediante interrupciones, para agilizar la ejecución.

- Baterías:

Para alimentar todo el sistema se han utilizado baterías LiPo 4s (14.8V) de 5000mAh, principalmente. En algunas pruebas se han tenido que usar baterías LiPo 3s (11.1V) de diferente capacidad y peso (2600mAh, 3600mAh…), por no disponer en ese momento de una batería 4s cargada.

Sólo es necesaria una batería cada vez, y con ella se alimentan tanto los motores como los componentes electrónicos (*IMU, PIC…*). Se decidió utilizar las baterías 4s por disponer de 2 unidades del mismo tamaño y capacidad, puesto que al cambiar de peso hay que reajustar las ganancias de los bucles de control PID del software controlador de vuelo.

El peso hace que las inercias sean mayores o menores, por lo que las ganancias en las correcciones autónomas del sistema de vuelo deben ser diferentes para los diferentes pesos inducidos por las diferentes baterías. También habría que cambiar estas ganancias en caso de cambiar de motores o de modificar las medidas del aparato.

- PC:

Por supuesto, para todo el desarrollo ha sido necesario el uso de un PC, en nuestro caso un portátil Compaq CQ61 (en propiedad del alumno), con sistema operativo Windows 7.

SOFTWARE

El software utilizado en este proyecto lo podemos dividir en dos grupos: las herramientas de desarrollo y el software desarrollado (es decir, el proyecto en sí mismo).

- Entorno de desarrollo o *IDE:*

Hemos utilizado el *IDE* (*Integrated Development Environment*) suministrado por Microchip MPLAB X, que es una versión del *IDE* NetBeans (gratuito y de código abierto) modificada por Microchip para ajustarse a las necesidades del desarrollo en PIC.

Imagen 29: Icono MPLAB X

Como lenguaje de programación hemos usado el C (lo normal desarrollando sobre PIC), y el compilador XC16 de Microchip.

Decir que, aunque este entorno de desarrollo tiene todo lo necesario para el desarrollo de aplicaciones en la plataforma PIC, le hemos añadido el módulo de NetBeans *nbgit-0.4* para la

gestión de versiones mediante Git, pues la versión v1.90 de MPLAB X no trae esta opción por defecto (ofrece gestión de versiones mediante *CVS, Mercurial y Subversion*).

- Software de Gestión de Versiones:

El gestor de versiones elegido para mantener el código fuente del proyecto organizado y a buen recaudo ha sido *Git*. Esta decisión ha sido tomada debido al conocimiento del mismo gracias a su uso en desarrollos anteriores.

Imagen 30: Icono Git

A parte de la instalación de *Git* en nuestro PC, se ha utilizado el módulo *nbgit* para su compatibilidad con nuestro *IDE* (como hemos comentado anteriormente), el gestor visual de versiones *SourceTree* (compatible con *Git*) y el servicio online de gestión de código *BitBucket.org*, ambos gratuitos y de la compañía Atlassian.

El gestor visual *SourceTree* es una aplicación muy recomendable para el uso de Git bajo Windows, puesto que facilita en gran medida su uso, dejando atrás la línea de comandos y ofreciendo todo lo necesario a un *click* de ratón.

- Monitor serie:

Para la visualización de valores *en vivo*, para facilitar la depuración del código y comprobar valores fundamentales como los ángulos de vuelo, se ha utilizado la comunicación serie siguiendo el protocolo RS232. Como nuestro PC no dispone de puerto serie DB-9, se ha hecho uso de un convertidor USB a puerto serie.

El programa *HyperTerminal* (una versión del mismo adaptada a Windows 7) ha sido el elegido para mostrar por pantalla los datos recibido por el puerto serie. Una vez más se ha elegido este modelo por su familiaridad y facilidad de uso.

- Controlador de vuelo:

El software de control de vuelo desarrollado en este proyecto, se basa en la teoría disponible por Internet sobre los sistemas de control de los multicópteros.

Es un sistema simplificado, diseñado específicamente para nuestro hardware, cuyo principal objetivo es el de lograr que el cuadricóptero reconozca su posición espacial, la compare con la posición deseada por el piloto, e intente aproximarse lo más suavemente posible a la misma.

En caso de no haber ningún comando por parte del piloto (es decir, que los *sticks* de dirección e inclinación permanezcan en su posición central), el cuadricóptero intentará mantenerse en horizontal y mantener la dirección establecida por el piloto con anterioridad (o la dirección inicial), gracias al cálculo de los ángulos de vuelo, ángulos obtenidos mediante el procesado de los valores de la *IMU*.

SISTEMA DE CONTROL DE VUELO

En este apartado se describe el núcleo del proyecto, el Sistema de Control de Vuelo desarrollado. Se explican las diferentes secciones del algoritmo, sus funciones y sus interacciones con el resto de secciones en una abstracción teórica, evitando entrar en el código en sí.

El siguiente diagrama muestra las secciones que componen el sistema y cómo interactúan entre sí, dando lugar al algoritmo que controla el vuelo del cuadricóptero.

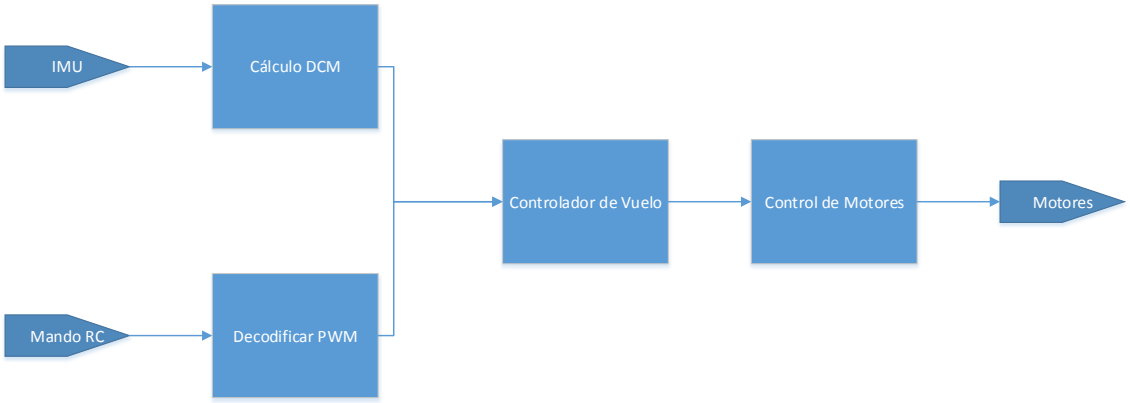

Imagen 31: Diagrama de bloques del Sistema de Control de Vuelo

Veamos a continuación las secciones, paso a paso en el orden de ejecución (orden que coincide con el orden de desarrollo):

LECTURA DE SENSORES

Esta sección (como su nombre indica), consiste en leer los sensores de la *IMU*, y convertirlos a valores entendibles por el ser humano. Como hemos comentado anteriormente, la *IMU* dispone de tres sensores principales (y uno de temperatura, pero no lo utilizamos) que son: acelerómetro, giróscopo y magnetómetro. La idea es obtener la inclinación del cuadricóptero en el espacio tridimensional gracias a las lecturas obtenidas de estos sensores.

Estos sensores los leeremos cada 20 ms (es decir, a 50Hz), dando comienzo con ello a todo el bucle de control.

El sensor principal, y sin el cual nos sería muy difícil (por no decir imposible) controlar nuestro cuadricóptero es el Giróscopo. Su importancia la corrobora el hecho de que hay cuadricópteros comerciales de gama baja que funcionan solamente con este sensor.

En la fase de inicialización del software, ejecutamos un gran número de lecturas del Giróscopo, para calcular el error que tiene cada eje no habiendo giro. Estos nos sirven para calibrar cada lectura que hagamos, pues no tendremos en cuanta esa desviación mínima que tiene cada eje por defecto.

El Giróscopo nos proporciona la velocidad de giro (en grados por segundo, °/s) alrededor de cada uno de los ejes espaciales del cuadricóptero (X,Y,Z). Esto nos lleva a que en cada intervalo de tiempo (20ms) sepamos con cierta precisión lo que ha girado, y acumulando estos giros sobre la posición inicial, podamos saber aproximadamente en que ángulo nos encontramos. ¿Pero cómo conocer la posición inicial, si el Giróscopo solamente mide el movimiento? Ahí es donde entran el juego tanto el Acelerómetro como el Magnetómetro.

El Acelerómetro, cuando estamos parados (posición inicial), la única fuerza que detecta es la fuerza gravitacional, por lo que podemos calcular fácilmente el ángulo de inclinación que tenemos con el plano horizontal (pues la gravedad es perpendicular a este). Como hemos comentado al explicar los ángulos de Euler, por convención el orden de giro de los ángulos es: Yaw, Pitch y Roll. Como el ángulo Yaw es el giro sobre el eje vertical, no podemos obtener ninguna información sobre el mismo del vector gravedad.

Para calcular el Pitch y el Roll inicial, supongamos que nuestro cuadricóptero (su sistema de referencia) empieza alineado con el sistema de referencia del mundo. Primero es girado un determinado Yaw (ψ), luego un determinado Pitch(θ) y luego un determinado Roll(ϕ). Una vez hecho esto estaríamos en la siguiente situación:

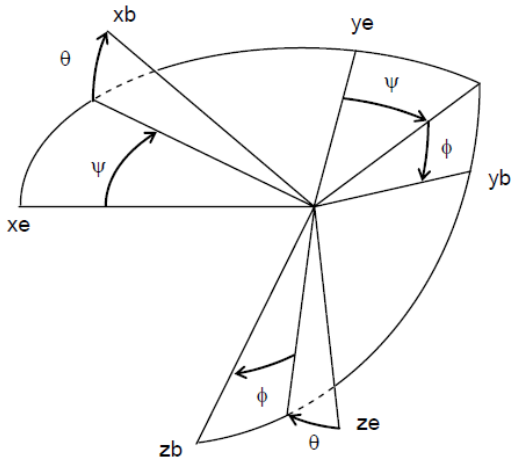

Imagen 32: Sistemas de referencia: Tierra (e de *earth*) y Vuelo (b de *body*)

Como sabemos, el acelerómetro nos da los componentes del vector de gravedad en el sistema de referencia del cuadricóptero, por lo que podemos comprobar que sen(180°+θ) es la proyección de la gravedad sobre el eje X, y cos(θ) la proyección sobre el plano YZ. Conociendo estas dos valores, podemos calcular el ángulo θ gracias al uso del *arctan*, puesto que ya conocemos el valor de tan(θ) gracias al seno y al coseno. Por lo tanto calculamos el Pitch de la siguiente forma:

$$\theta = \ arctan\left(\frac{sen(-\theta)}{cos(\theta)}\right) = arctan\left(\frac{acelX}{||(0,acelY,acelZ)||}\right)$$

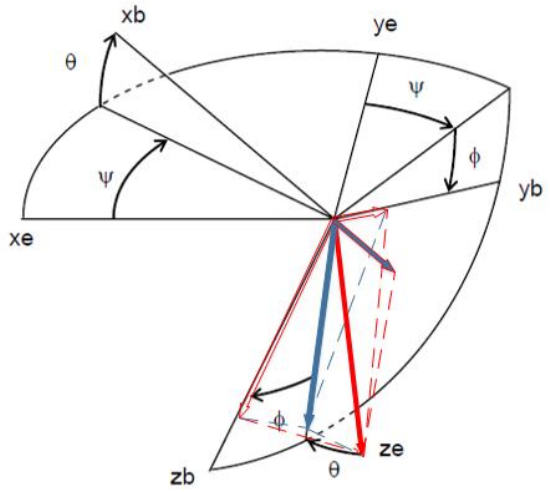

Imagen 33: Proyecciones para el cálculo del Pitch

Para el cálculo del Roll haremos algo similar, pero en este caso hemos de tener en cuenta que el giro del Roll se ha hecho sobre un plano YZ desplazado del plano vertical por la inclinación causada por el Pitch. Por lo tanto no podemos usar directamente las proyecciones del vector gravedad pues ninguna es del ángulo ϕ que queremos hallar.

Por tanto, para contrarrestar la inclinación del Pitch, usaremos la proyección del vector gravedad sobre el plano YZ (en adelante, *gravedadYZ*, cuyo módulo es cos(θ)) como vector gravedad, y podemos ver que el componente Y del vector gravedad es cos(θ)*sen(ϕ), y el componente Z es a su vez cos(θ)*cos(ϕ). Por lo tanto, calculamos el Roll como hicimos antes con el Pitch:

$$\phi = \ arctan\left(\frac{cos(\theta)sen(\phi)}{cos(\theta)cos(\phi)}\right) = arctan\left(\frac{acelY}{acelZ}\right)$$

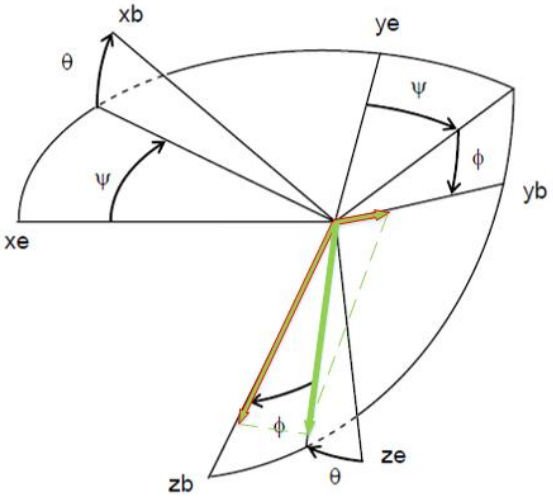

Imagen 34: Proyecciones para el cálculo del Roll

Para el cálculo del Yaw inicial, como el giro sobre el eje vertical no afecta a las proyecciones del vector gravedad en el sistema de referencia del cuadricóptero, necesitamos de la información proporcionada por el Magnetómetro. Sin embargo, aunque el magnetómetro señala al norte (pues al fin y al cabo es una brújula), lo hace en tres dimensiones, por lo que los valores que nos da (al estar en el sistema de referencia de vuelo), no nos sirven directamente para conocer nuestra desviación en el plano horizontal. Además, el norte tridimensional, no es un vector sobre el plano, si no que apunta en parte hacía abajo; por lo tanto, aún no habiendo ni Pitch ni Roll, no podríamos utilizar directamente los valores dados por el magnetómetro.

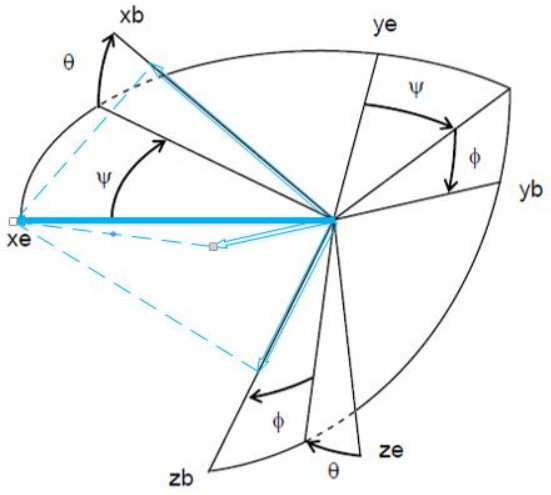

Imagen 35: Vector Norte en el sistema de referencia del cuadricóptero

Para poder usar los valores del magnetómetro, debemos proyectar el norte tridimensional sobre el plano horizontal rotado por nuestro ángulo Yaw. Es decir, debemos obtener los componentes X e Y del vector norte sobre el plano horizontal logrado al girar el de la tierra por un ángulo ψ. Para ello empezamos proyectando todos los componentes del vector norte sobre el eje X_{V1}:

$$ProyNorteX = magneX \cdot cos(\theta) + magneY \cdot sen(\phi) \cdot sen(\theta) + magneZ \cdot cos(\phi) \cdot sen(\theta);$$

A continuación procedemos a hacer lo mismo, pero sobre el eje Y_{V1}:

$$ProyNorteY = magneY \cdot cos(\phi) - magneZ \cdot sin(\phi);$$

Una vez tenemos los componentes del vector norte sobre el plano girado, podemos obtener el ángulo que hace el vector norte proyectado (que es el eje X del sistema de referencia de tierra) con nuestro eje X_{V1} mediante el uso del arcotangente. El Yaw es el negativo de este ángulo:

$$\psi = -arctan\left(\frac{ProyNorteY}{ProyNorteX}\right)$$

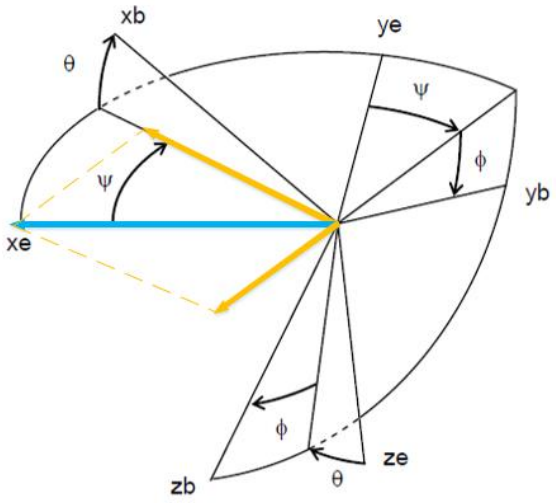

Imagen 36: Proyecciones para el cálculo del Yaw

Una vez tenemos la posición inicial, podemos empezar a movernos y a calcular nuestra orientación actual mediante la Matriz de Cosenos Directores (*DCM*).

DCM

El algoritmo de la *DCM* tiene como finalidad acumular todos los giros realizados y darnos la orientación actual a modo de Ángulos de Euler. ¿Y por qué necesitamos de un algoritmo con matrices en vez de ir acumulando el giro realizado sobre cada eje? Porque los giros no son conmutativos. No es lo mismo girar primero en Z y luego en Y, que primero en Y y luego en Z:

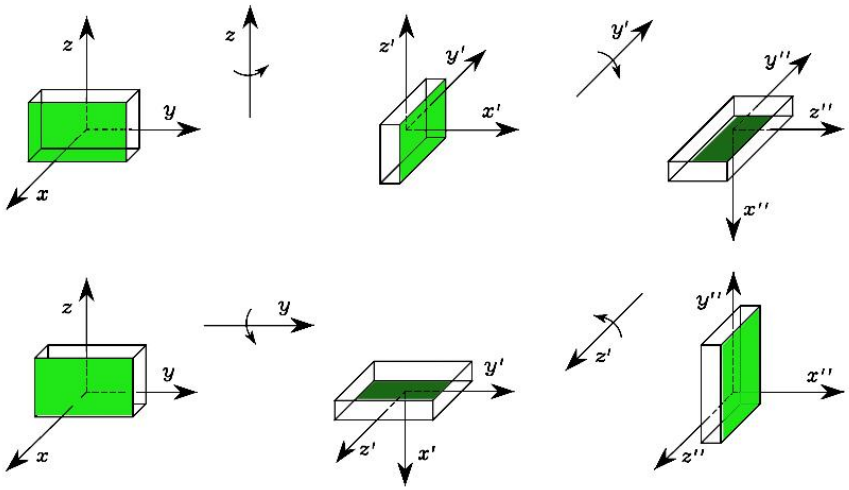

Imagen 37: Rotaciones no conmutativas

Por lo tanto hay que tratar el problema de otro modo, y es con matrices de rotación, y más concretamente con la *DCM*.

Las matrices de rotación son matrices 3x3 que multiplicadas por un vector tridimensional sirven para girarlo en el espacio. Las básicas para el giro en los ejes X, Y y Z son las siguientes:

$$G_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sen(\phi) \\ 0 & sen(\phi) & cos(\phi) \end{pmatrix}$$

$$G_Y = \begin{pmatrix} cos(\theta) & 0 & sen(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & cos(\theta) \end{pmatrix}$$

$$G_Z = \begin{pmatrix} cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Si a nuestro sistema de referencia de tierra (que sería una matriz identidad), lo giramos primero en Z, luego en Y y después en X (tal como dice la teoría de los Ángulos de Euler), obtendremos la siguiente matriz:

$$I \times G_Z \times G_Z \times G_Z = R$$

$$R = \begin{pmatrix} cos(\theta)cos(\psi) & sen(\phi)sen(\theta)cos(\psi) - cos(\phi)sen(\psi) & cos(\phi)sen(\theta)cos(\psi) + sen(\phi)sen(\psi) \\ cos(\theta)sen(\psi) & sen(\phi)sen(\theta)sen(\psi) + cos(\phi)cos(\psi) & cos(\phi)sen(\theta)sen(\psi) - sen(\phi)cos(\psi) \\ -sen(\theta) & sen(\phi)\,cos(\theta) & cos(\phi)\,cos(\theta) \end{pmatrix}$$

Esa es la Matriz de Cosenos Directores, y aunque multipliquemos más matrices de giro por ella, siempre mantiene esta estructura, pudiendo obtener los ángulos de vuelo (en cualquier momento) de la siguiente manera:

$$\psi = arctan\left(\frac{cos(\theta)sen(\psi)}{cos(\theta)cos(\psi)}\right)$$

$$\theta = -arcsin\big(-sen(\theta)\big)$$

$$\phi = arctan\left(\frac{sen(\phi)\,cos(\theta)}{cos(\phi)\,cos(\theta)}\right)$$

Esta ecuación, además, nos sirve para pasar vectores del sistema de referencia de vuelo al de tierra y viceversa; siendo las columnas de la matriz los ejes X, Y y Z de vuelo en el sistema de referencia de tierra, y las filas los eje X, Y y Z de tierra en el sistema de referencia de vuelo. Por lo tanto, para representar un vector del sistema de referencia de tierra en el sistema de vuelo, debemos multiplicarlo por la *DCM*; y en caso de querer pasar un vector del sistema de vuelo al de tierra, debemos multiplicarlo por DCM^{-1} (matriz inversa), o lo que es lo mismo en este caso DCM^{T} (su traspuesta).

Pero, ¿cómo acumulamos el giro calculado por el giróscopo? ¿Cual multiplicamos primero, el giro en X, en Y o en Z? La solución viene de nuevo de la *DCM*. Tomando las velocidades de giro leídas cada 20ms como constantes en ese intervalo de tiempo, podemos calcular una matriz

DCM con esos ángulos de giro, y multiplicarla por la de la posición actual, logrando así la nueva posición en la matriz resultante.

Pero calcular todos esas funciones trigonométricas cada 20ms puede ser demasiado para el microcontrolador. Por ello, sabiendo que los ángulos girados en 20ms no pueden ser muy grandes, usamos la siguiente aproximación (siempre utilizando los ángulos en radianes, como marca el SI):

-La multiplicación de cosenos cercanos a 0° es prácticamente 1.

-La multiplicación de senos cercanos a 0° es prácticamente 0.

-El valor de los senos para ángulos pequeños, se aproxima al ángulo en sí medido en radianes.

-El valor de los cosenos cercanos a 0° se aproxima a 1.

Teniendo en cuenta esto, la matriz *DCM* que contiene el giro en estos 20ms, se nos queda así:

$$R(dt) = \begin{pmatrix} 1 & -\alpha_Z & \alpha_Y \\ \alpha_Z & 1 & -\alpha_X \\ -\alpha_Y & \alpha_X & 1 \end{pmatrix}$$

Siendo: $\alpha_X = \omega_X \cdot dt$, $\alpha_Y = \omega_Y \cdot dt$, $\alpha_Z = \omega_Z \cdot dt$ y $dt = 20ms$.

Pero, al calcular la *DCM* con aproximaciones e intervalos de velocidad constante, vamos acumulando un error (aparte del error debido a la digitalización), por lo que nuestro sistema de coordenadas va perdiendo la ortogonalidad, dejándolo inservible. Lo bueno es que tiene fácil solución haciendo pequeñas correcciones en cada iteración del bucle.

Este proceso de garantizar la ortogonalidad se denomina **Renormalización**, y consiste en garantizar que todos los ejes sean perpendiculares entre sí, y que su módulo sea 1.

Para **obtener vectores perpendiculares,** nos basamos en la definición del producto escalar entre vectores:

$$X \cdot Y = \|X\| \cdot \|Y\| \cdot \cos(\alpha)$$

Por lo que el producto vectorial de dos ejes perpendiculares es 0. Por lo tanto usaremos el producto vectorial como medida de error. De primeras cogemos los ejes X e Y, y restamos la mitad del error obtenida a cada uno (en la dirección del vector que influye en el error):

$$X_{Ortogonal} = X - \frac{errorXY}{2} Y$$

$$Y_{Ortogonal} = Y - \frac{errorXY}{2} X$$

Con esto reducimos sustancialmente el error. Podríamos hacer lo mismo con el eje Z, pero sabiendo que ha de ser perpendicular a ambos, la mejor manera es usar el producto vectorial de ambos como Z:

$$Z = X_{Ortogonal} \times Y_{Ortogonal}$$

Ahora que ya hemos logrado la perpendicularidad entre los ejes, debemos asegurarnos de que su módulo es 1. Eso se hace dividiendo cada vector por su módulo, pero de nuevo nos encontramos con que el cálculo del módulo (al haber raíces cuadradas y divisiones) es bastante pesado para el procesador, por lo que usamos una *expansión de Taylor* de grado 1 para aproximar la división y la raíz cuadrada del cálculo:

$$V_{normalizado} = \frac{V}{\|V\|} = \frac{V}{\sqrt{V \cdot V}}$$

Por lo que aproximamos la función $\frac{1}{\sqrt{x}}$ en torno al número 1 (pues el módulo de los ejes en cada iteración cambia muy poco). Por lo tanto, nos queda la siguiente expansión de Taylor:

$$P(x) = \frac{1}{2}(3 - x)$$

Por lo que normalizamos los vectores:

$$X_{normalizado} = \frac{1}{2}(3 - X_{ortogonal} \cdot X_{ortogonal})X_{ortogonal}$$

$$Y_{normalizado} = \frac{1}{2}(3 - Y_{ortogonal} \cdot Y_{ortogonal})Y_{ortogonal}$$

$$Z_{normalizado} = \frac{1}{2}(3 - Z_{ortogonal} \cdot Z_{ortogonal})Z_{ortogonal}$$

Ahora ya tenemos nuestra matriz *DCM* ortogonal, pero aún nos queda un problema por resolver, y se debe una propiedad de los giróscopos: la deriva.

La deriva de los giróscopos produce que aun habiendo dejado de girar, durante un tiempo nos muestren un ligero giro en la salida (por algún tipo de inercia). Esto nos puede ir aumentando el error en los ángulos de vuelo, haciéndonos perder totalmente el control sobre el cuadricóptero. Para ello tenemos el algoritmo de Cancelación de Deriva, que no es más que un controlador PI que se vale de los otros dos sensores de la *IMU*, pues estos no sufren de deriva.

Primero comparamos la gravedad actual medida por el acelerómetro (comprobando siempre que no hay ningún aceleración significante producida por el movimiento del cuadricóptero), con el supuesto eje Z de la tierra (tercera fila de la *DCM*). Para ello utilizamos el producto vectorial entre ambo: será cero cuando sean paralelos, o nos dará el error de giro que hay de uno a otro (pues el resultado es un vector perpendicular a los dos, un vector de rotación, pues su módulo nos depende del ángulo girado). Este error (*errorPitchRoll*) nos sirve para el control del error inclinación.

$$errorPitchRoll = (acelX, acelY, acelZ) \times (\mathrm{DCM}_{31}, \mathrm{DCM}_{32}, \mathrm{DCM}_{33})$$

Por otra parte, calcularemos la proyección sobre el plano horizontal del vector norte que nos da el magnetómetro. Esta debería ser paralela al eje X de la tierra (primera fila de la *DCM*), así que el producto vectorial entre ambas nos dará el error de dirección (tal y como hemos hecho con la gravedad). Lo bueno en este caso, es que nos ahorramos 2/3 de los cálculos del producto vectorial, pues ya sabemos que le vector resultante va tener la dirección del vector Z de la tierra. Por lo tanto, nos vale con calcular el componente Z del error, y escalar el eje Z de la tierra con él:

$$errorYaw_Z = \mathrm{DCM}_{00} \cdot ProyNorteY \; - \; \mathrm{DCM}_{10} \cdot ProyNorteX$$

$$errorYaw = \mathrm{DCM}_{2} \cdot errorYawZ$$

Una vez calculados los dos errores, retroalimentamos un bucle PI con ellos, buscando anular el error, y la corrección resultante se la sumamos al vector de giro que nos da el giróscopo, para actualizar la *DCM* evitando (en la medida de lo posible) la deriva.

Recapitulando, el cálculo de la *DCM* se desarrolla de la siguiente manera:

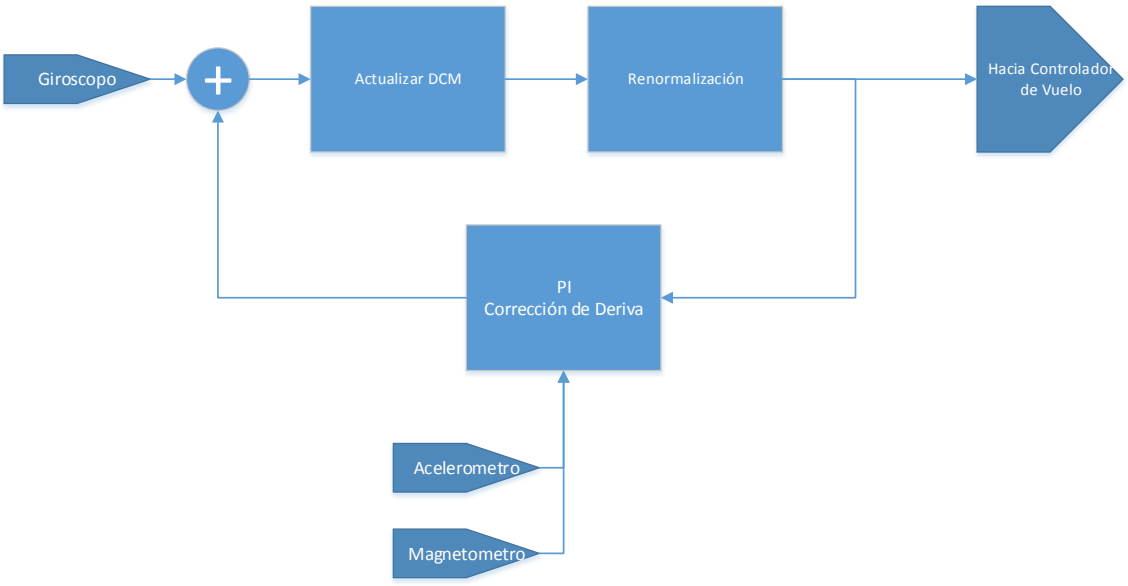

Imagen 38: Diagrama de bloques del algoritmo *DCM*

Y gracias a estos cálculos podemos conocer los ángulos de vuelo de nuestro cuadricóptero en cualquier momento.

CONTROLADOR DE VUELO

Esta parte del algoritmo es el núcleo del controlador de vuelo. Recibe como entradas los ángulos de vuelo y las ordenes del piloto, calcula los ajustes necesarios para mantener el cuadricóptero bajo control del piloto, y envía los valores necesarios para aplicar esos ajustes a los motores. Para ello cuenta (como podemos ver en el diagrama) con dos bucles PID: el de Estabilidad y el de Giro.

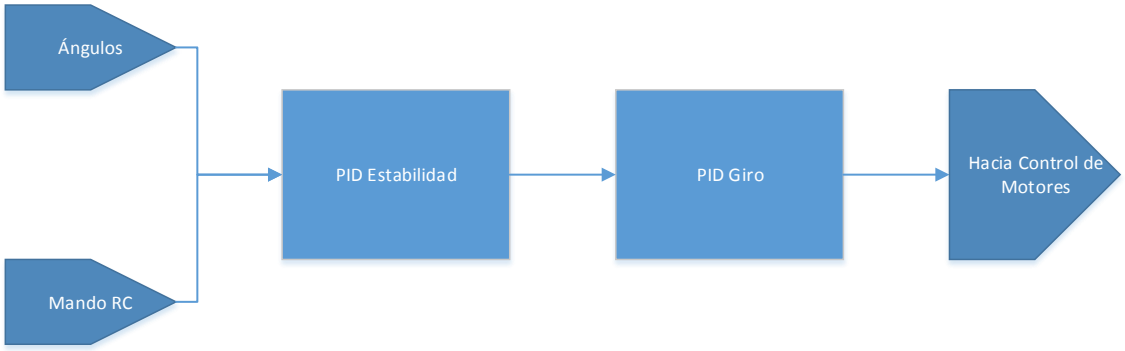

Imagen 39: Diagrama de bloques del Controlador de Vuelo

El PID de Estabilidad, recibe el ángulo que queremos conseguir, y lo compara con el actual, obteniendo la corrección de giro que necesitamos llevar a cabo (en los 20ms de iteración) para lograrlo.

Después del controlador de Estabilidad, el PID de Giro compara el giro que deseamos hacer (que es la corrección calculada en el paso anterior), con la velocidad de giro que llevamos actualmente (dada por el giróscopo), y obtiene la corrección necesaria para rotar hasta el ángulo deseado.

Como es lógico, en estos controladores PID hay tres sub-controladores, uno por cada eje de giro. Pero no todos funcionan de la misma manera, por lo que nos encontramos con dos bloques internos: el Control de Inclinación y el Control de Dirección.

- Control de Inclinación:

Como su nombre indica, se dedica a controlar que el ángulo Pitch y el Roll sean los que deseamos. Su funcionamiento depende de sí existe una orden por parte del piloto o no:

-**Existiendo una orden por parte del piloto**:

Se dedica a intentar obtener el ángulo marcado por el piloto tanto en Pitch como en Roll.

Los controladores PID del Pitch y del Roll son completamente independientes.

Se puede ajustar el ángulo máximo y mínimo posible (para evitar vuelcos). Por defecto ocupa todo el rango del Pitch y del Roll: (-90°,90°).

-No existiendo orden por parte del piloto:

El controlador intenta mantener la estabilidad del cuadricóptero; es decir, intenta mantenerlo en horizontal en base a los ángulos de vuelo estimados.

No sólo corrige inclinaciones dejadas por el piloto, sí no también las causadas por fuerzas externas, como aire o pequeños golpes.

Para la obtención de la estabilidad ante fuerzas externas es inevitable un mínimo de desplazamiento en la dirección de la corrección.

- Control de Dirección:

Como su nombre indica, este controlador se encarga de que el ángulo Yaw sea el correcto. Como el anterior, su funcionamiento depende de sí existe una orden por parte del piloto o no:

-Existiendo una orden por parte del piloto:

Se dedica a rotar el cuadricóptero en el plano sobre su eje Z.

En este caso, el piloto indica el ángulo a incrementar o disminuir, no el ángulo absoluto o dirección.

El rango de incremento por defecto es (-180°,180°). Este rango se puede ajustar fácilmente.

-No existiendo orden por parte del piloto:

El controlador intenta mantener la dirección actual; es decir, la última dirección ordenada por el piloto (o la de inicio).

No sólo mantiene la dirección, sí no que de haber un giro obligado por fuerzas externas (aire, movimiento manual, atasco con algo…), una vez la fuerza externa desaparece, vuelve a orientarse hacia la dirección guardada.

Vuelve a la dirección deseada por el ángulo más corto.

Por último, el Controlador de Vuelo también se encarga de traducir la señal que llega del acelerador y transferirla a los motores, para controlar su velocidad.

LECTURA DE MANDO RC

Como se ha comentado al explicar el hardware, disponemos de un mando y un receptor de RC (radio control).

El receptor Futaba R617FS traduce los comandos del mando a pulsos PWM, pulsos que envía por los pines conectados al PIC.

En el PIC, usamos los módulos *Input Capture* para decodificar la señal de entrad y convertirla a valores comprensible para el Controlador de Vuelo.

Los canales 1 a 4 corresponden a los movimientos de Alabeo, Cabeceo, Aceleración y Guiñada respectivamente, y van conectados a los módulos 1, 2, 4 y 3 del *Input Capture* (en ese orden).

El quinto canal, conectado al *IC5*, pertenece al *switch G* del mando, y activa el modo calibración.

- Modo calibración:

En este modo, calibramos los sticks del mando, para que el sistema pueda ganar precisión calculando los valores máximo y mínimos, y el offset de cada *stick*.

Para ello seguiremos los siguientes pasos:

1. Activar el modo calibración moviendo el *switch G* hacía nosotros (teniendo el mando en las manos).
2. Girar los dos *sticks*, llevándolos lo más al margen posible. Dar unas cuantas vueltas a cada uno.
3. Soltar todos los *sticks*. A continuación bajar el *stick* de aceleración hasta abajo.
4. Salir del modo calibración moviendo el *switch G* en dirección opuesta a nosotros.

CONTROL DE MOTORES

El control de motores se realiza por interrupciones del bus I2C. Se lanza una interrupción que envía cada 2ms (500Hz) la velocidad de los motores a los *ESC*, para evitar que se paren por falta de refresco.

El cálculo de cada motor es diferente:

- El componente principal de todos es el de la aceleración marcada por el piloto.
- Los dos motores sobre el eje X, el frontal y el trasero, reciben la corrección del Pitch generada por el controlador de vuelo:
    - Al frontal se le suma el valor de corrección del Pitch (pues el Pitch positivo eleva el morro).
    - Al trasero se le resta el valor de corrección del Pitch (pues el Pitch negativo eleva la cola).
- Los dos motores sobre el eje Y, el derecho y el izquierdo, reciben la corrección del Roll generada por el controlador de vuelo:

    - Al izquierdo se le suma el valor de corrección del Roll (pues el Roll positivo eleva el brazo izquierdo).
    - Al derecho se le resta el valor de corrección del Roll (pues el Roll negativo eleva el brazo derecho).
- Respecto al Yaw, todos los motores reciben la corrección de la dirección generada por el controlador:
    - A los motores del eje X se les resta la corrección (pues giran hacía la derecha, creando un *torque* que empuja los ejes hacía la izquierda, y el Yaw positivo es a la derecha).
    - A los motores del eje Y se les suma la corrección (pues giran hacía la izquierda, creando un *torque* que empuja los ejes hacía la derecha, y el Yaw positivo es a la derecha).

Una vez calculado el valor de cada motor, se traduce un rango de 8 bits (0-255), para enviarlo por el I2C hasta los *ESCs*.

Los motores 0, 1, 2 y 3 responden respectivamente a los motores frontal, trasero, derecho e izquierdo.

RESULTADOS Y FUTUROS DESARROLLOS

En este último apartado, exponemos los resultados obtenidos y las posibles mejoras a implementar en un futuro.

RESULTADOS

Los resultados del proyecto los podemos distribuir en dos grupos, los resultados teóricos y los prácticos:

- Teóricos:

Con resultados teóricos nos referimos a los resultados del algoritmo de la *DCM*. Sólo usando nuestro microcontrolador y la *IMU*, podemos ver en pantalla si los ángulos de vuelo calculados son los correctos o no.

Usando la aplicación (desarrollada en *Processing*) *Razor AHRS*, que podemos descargar desde https://github.com/ptrbrtz/razor-9dof-ahrs, hemos podido visualizar la orientación de la *IMU*, estimada por el PIC, en nuestro PC. La aplicación recibe los ángulos en binario a través de la RS232, y dibuja una caja que representa nuestra *IMU,* con una flecha verde señalando la parte delantera de la misma.

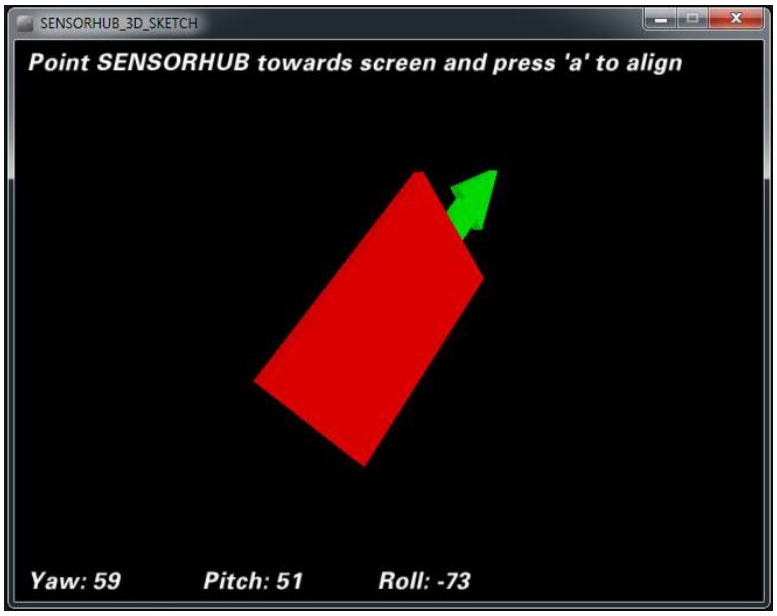

Imagen 40: Aplicación Razor AHRS en funcionamiento

Gracias a esta aplicación hemos podido tunear las ganancias del bucle PI del algoritmo *DCM*, logrando una estimación extremadamente fiel a la posición real. Los valore PI calculados como óptimos son los siguientes:

- Ganancia proporcional: 0,7
- Ganancia integral: 0,00002

Con esta ganancia proporcional logramos que la estimación se ajuste a la realidad en un muy breve espacio de tiempo, incluso habiendo grandes oscilaciones en la entrada del controlador PI (grandes variaciones de la orientación real), al instante de acabar las oscilaciones (menos de ¼ de segundo) la estimación se ajusta a la posición real, sin seguir oscilando por un tiempo.

Con la ganancia integral logramos ajustar el pequeño error estacionario que deja el ajuste proporcional, pues simplemente con él no logramos alcanzar (por muy poco), el valor real. Se ha comprobado que este valor integral no crea sobreoscilación después de un tiempo de funcionamiento.

- Prácticos:

La comprobación de los resultados prácticos es mucho más ardua y peligrosa que las pruebas teóricas realizadas.

La parte práctica, es la más entretenida y la más complicada a la vez, pues consiste en poner en marcha el cuadricóptero, y probar que responde bien a las ordenes del piloto, y que corrige adecuadamente los posibles desequilibrios producidos por fuerzas externas.

Pero para poder comprobar todo eso, hay que tomar unas medidas de seguridad, a fin de garantizar la integridad física del piloto (y otras personas que pueda haber alrededor), y la del propio cuadricóptero (para evitar romper piezas del mismo).

El primer paso a realizar es el tuneo de valores de los controladores PID del Control de Inclinación. Para evitar accidentes, se sujetaron las patas del cuadricóptero a unas gomas (permitiéndole elevarse sólo unos centímetros), y se rodearon los cuatro costados del mismo con unas cuerdas atadas del techo a la mesa, pudiendo desplazarse solamente en vertical y dentro del perímetro delimitado por las cuerdas. Con esto logramos evitar el desplazamiento lateral y longitudinal, permitiendo la inclinación y un poco de desplazamiento vertical.

Para realizar los tuneos, se usó la guía de tuneo disponible en la página web de la plataforma AeroQuad (http://aeroquad.com/showwiki.php?title=PID+Tuning). Los valores óptimos calculados son los siguientes (siempre con la batería de 5000 mAh, pues el peso influye):

- PID de Estabilidad
  - Ganancia proporcional: 1,2
  - Ganancia integral: 0,1
  - Ganancia derivativa -30,0
- PID de Giro
  - Ganancia proporcional: 30,0
  - Ganancia integral: 0,0
  - Ganancia derivativa -200,0

Con estos valores, hemos logrado una respuesta adecuada a las órdenes de inclinación (cabeceo y alabeo) del piloto; y una corrección rápida y sin sobreoscilación de inclinaciones no deseadas, logrando que el cuadricóptero mantenga la horizontalidad por sí mismo.

El siguiente paso es tunear los valores del PID del Control de Dirección. Pero es este caso, la "jaula" de cuerdas y el atar las patas no nos permite la libertad suficiente para realizar el giro sobre el eje Z (guiñada), por lo que hemos tomado otras medidas de seguridad. Se ha suspendido el cuadricóptero del techo, a muy baja altura, de tal manera que con poca aceleración no dependa de las cuerdas para mantenerse en el aire, si no que lo haga por el impulso de sus rotores. Al estar colgado del centro, nos permite realizar el movimiento de guiñada casi con total libertad (pues la cuerda siempre influye en algo a la hora de contrarrestar el giro inducido por los rotores), y poder tunera el controlador, y comprobar su correcto funcionamiento. De nuevo nos ceñimos a la guía de AeroQuad sobre tuneos (por ser la más completa que se ha encontrado). Los valores óptimos calculados para el controlador PID del Control de Dirección son los siguientes (siempre con la batería de 5000 mAh, pues el peso influye):

- PID de Estabilidad
    - Ganancia proporcional: 1,5
    - Ganancia integral: 0,0
    - Ganancia derivativa 0,0

- PID de Giro
    - Ganancia proporcional: 20,0
    - Ganancia integral: 0,0
    - Ganancia derivativa -100,0

Con estos valores hemos logrado una respuesta adecuada a la orden de guiñada del piloto, una retención de la dirección actual extraordinaria, una buena corrección del cambio de dirección causado por influencias externas. Esto último hace que vuelva tomar la dirección fijada de una manera suave, sin sobreoscilación, y siempre girando lo menos posible (es decir, gira hacía el lado más corto).

Nos gustaría haberlo probado en vuelo libre, pero debido a su tamaño (60cm de motor a motor), su peso (cercano a los 2kg), y la gran potencia de sus motores (que pueden cortar con las hélices a cualquiera si se pierde el control); necesitamos de más medidas de seguridad para hacerlo. Estaría bien meterlo en un recinto rodeado de redes (tal y como se hace en otras universidades), y probar a controlarlo desde fuera, sin posibilidad de que haga daño a nadie. Así podríamos comprobar que el control es correcto (como parecen indicar las pruebas realizadas), y después probarlo en el exterior, siempre delimitando un perímetro de seguridad para los posibles espectadores.

PROXIMOS PASOS Y MEJORAS

El principal paso a dar (que no se ha llevado a cabo por motivos de seguridad y por falta de tiempo), es el vuelo libre. Con este vuelo podríamos afinar más las ganancias de los controladores PID, logrando una mejora en la suavidad del manejo.

Como mejoras, creemos que deberíamos diseñar y fabricar un PCB que incluyese las conexiones necesarias para los diferentes dispositivos (*IMU, ESCs*, receptor RC, PIC…), pues es bastante aparatoso llevar la placa Explorer 16 montada en el centro del cuadricóptero.

Otra mejora (que requeriría de mucho esfuerzo, un proyecto aparte), sería la inclusión de un sistema de control mediante GPS, que nos permitiría el regreso a casa en caso de pérdida de señal o extravío; e incluso más adelante un sistema de vuelo completamente autónomo, que dirija el cuadricóptero hasta las coordenadas establecidas.

También sería una buena medida un control de aterrizaje automático, basado en un controlador de altitud (barómetro y sensor de distancia por ultrasonidos), que nos aterrizara el cuadricóptero suavemente sobre el suelo, pues no es una maniobra sencilla (debido a las turbulencia creadas por el aire desplazado contra el suelo, el denominado *efecto suelo*).

Por último, aunque sea más sencillo que las mejoras anteriores, vendría bien adaptar el código para que pueda ser más modulable, y poder usar otros sensores, motores o receptores de radiocontrol.

Imagen 41: Nuestro cuadricóptero en fase de pruebas

BIBLIOGRAFÍA

- https://sites.google.com/site/mikuadricoptero/
- https://github.com/ptrbrtz/razor-9dof-ahrs/wiki/Tutorial
- http://diydrones.com/profiles/blogs/dcm-imu-theory-first-draft?id=705844
    - De esta web podemos descargar los *papers* de Mahony y los de Premerlani, que contienen toda la información necesaria sobre la teoría de la *DCM*
- http://www.chrobotics.com/docs/AN-1005-UnderstandingEulerAngles.pdf
- http://www.pololu.com/file/0J589/AN-1008-SensorsForOrientationEstimation.pdf
- http://polakiumengineering.org/downloads/The_Beginner's_Guide_to_Multicopters_06292012.pdf
- http://www.fis.usb.ve/~mcaicedo/education/fisica2/rotation.pdf
- https://code.google.com/p/arducopter/wiki/AC2_StableMode
- http://technicaladventure.blogspot.com.es/2012/09/quadcopter-stabilization-control-system.html
- http://technicaladventure.blogspot.com.es/2012/03/quadrocopter-firmware-4-rotor-explained.html
- http://theboredengineers.com/2012/05/the-quadcopter-basics/
- http://www.mikrokopter.de/ucwiki/en/BL-Ctrl_Manual#Manual_BL-Ctrl
- http://www.mikrokopter.de/ucwiki/en/BL-Ctrl_Manual#Manual_BL-Ctrl
- http://blog.oscarliang.net/build-a-quadcopter-beginners-tutorial-1/
- http://en.wikipedia.org/wiki/Inertial_measurement_unit
- http://www.aviastar.org/helicopters.html
- http://en.wikipedia.org/wiki/Quadcopter