Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Control de Posicionamiento de un Cuadricóptero

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Aitor Cortés Sáez

Tutor: Pascual Pérez Blasco

2014-2015

Resumen

El principal objetivo de este proyecto consiste en la construcción de un vehículo aéreo no tripulado de cuatro rotores y su control de estabilidad utilizando microcontroladores no especializados en este campo de la plataforma Arduino. El diseño del vehículo se ha realizado a partir de elementos básicos y sin utilizar componentes preparados para este tipo de trabajo, tratando de probar la capacidad y el potencial de estos dispositivos para el desarrollo de esta clase de proyectos.

Para la realización de este trabajo se han estudiado todos los componentes a utilizar y el procedimiento de montaje, los posibles métodos de prueba del vehículo, el sistema de comunicaciones por radio y el sistema de control de la estabilidad del aparato. Las fuentes de información han sido obtenidas la mayoría de páginas especializadas en este tipo de vehículos, documentos, manuales y esquemas de los componentes y conocimientos propios.

Los resultados han sido positivos y los objetivos del proyecto se han cumplido casi en su totalidad. Se ha concluido satisfactoriamente que con los elementos antes mencionados se puede desarrollar perfectamente este tipo de vehículos de una forma eficiente y sencilla.

Palabras clave: cuadricóptero, cuadrirotor, drone, informática, mecatrónica, microcontrolador, Arduino, control, PID, vehículo, aéreo, radio, estabilización.

Abstract

The main objective of this project is the construction of a four-rotor unmanned aircraft vehicle using for its stability control a microcontroller unspecialized in this field provided by the Arduino platform. Vehicle design was made from basic elements and without using any prepared components for this kind of work, trying to probe the capability and the potential of these devices for the development of this projects.

We have studied all the components to be used and its assembly procedure, the possible testing methods, the radio communications system and the stability control system. Most of the information sources were obtained from specialized web pages in this kind of vehicles, documents, manuals and datasheets from the components and own knowledge.

The results were positive and the project objectives were accomplished almost entirely. It has been concluded that this kind of vehicles can be developed in an efficient way with the components mentioned above.

Keywords: quadcopter, quadrotor, drone, computing, mechatronics, microcontroller, Arduino, control, PID, vehicle, aircraft, radio, stabilization.

Índice de Figuras

Glosario

Arduino - Una plataforma que ofrece dispositivos que permiten utilizar microcontroladores de una forma fácil, sencilla y bastante eficiente, sin disminuir la potencia de uso del microcontrolador.

Ardupilot - Plataforma basada en Arduino especializada en el control de vehículos radio control, ofrece hardware preparado para funcionar con vehículos tanto terrestres como aéreos ('*Arducopter*') y sus distintas configuraciones.

Cuadricóptero - Vehículo aéreo no tripulado, impulsado por cuatro rotores con hélices situados normalmente con una configuración en equis o en cruz, el cual es controlado variando la velocidad relativa de cada rotor respecto a los demás.

ESC - '*Electronic Speed Controller*' o controlador electrónico digital, es un dispositivo que se utiliza para controlar la velocidad de un motor sin escobillas de tres fases, generando para ello una serie de pulsos que simulan una corriente trifásica. Este dispositivo también separa las tensiones que alimentan los circuitos de control de la tensión que alimenta los motores.

I2C - Bus digital desarrollado por Philips bastante eficiente y que utiliza únicamente dos cables para la transmisión de los datos (aparte de la alimentación y la masa).

IMU - '*Inertial Measurement Unit*' o unidad de medida inercial, es un dispositivo electrónico capaz de detectar y leer las fuerzas inerciales que se apliquen sobre éste.

Kalman - El filtro de kalman es un algoritmo que se utiliza para descartar los ruidos que se puedan haber producido en una señal determinada. Para este cálculo se ayuda de otra señal que pueda representar los mismos datos que la anterior.

KV - Unidad de medida que utilizan los fabricantes de motores sin escobillas para indicar distintas características del motor, como pueden ser la tracción y la velocidad.

PID - Algoritmo de control por retroalimentación que se basa en el cálculo del error entre un valor obtenido y uno deseado. Explicado de forma detallada en la sección "*4.2 Algoritmo de control PID*".

PPM - Un tipo de modulación para enviar valores mediante un canal digital. Explicado en más detalle en la sección "*2.9 Emisora y receptor de radio*".

PWM - Es un tipo de modulación que permite simular un valor analógico mediante una señal digital. Explicado en la sección "*2.5 Electronic Speed Controller*".

Índice de contenidos

1. Introducción

Actualmente se utilizan muchas y muy diversas palabras para denotar varios tipos de vehículos aéreos no tripulados. Desgraciadamente, aún no existen definiciones claras para cada uno de ellos. Y es por esto que no siempre se utilizan los términos correctamente a la hora de hablar de alguno de estos aparatos, muchas veces confundiendo significados y relacionando palabras como si fueran sinónimos.

Ningún organismo español oficial ha regulado ni definido con propiedad cómo llamar a cada uno de los diferentes tipos de vehículos aéreos no tripulados que existen. Este problema preocupa a los expertos, periodistas y público en general, pues se deja a criterio del autor el uso arbitrario de estas definiciones (1).

Todavía no existe definición propia de '*cuadricóptero*' en el Diccionario de la Real Academia Española (DRAE). En octubre de 2014 se publicó la vigesimotercera edición de este diccionario, la cual acogía el término "dron" (del inglés '*drone*') para referirse a un vehículo aéreo no tripulado (2). Pero sigue siendo una definición muy general para su uso en cuestiones más técnicas.

Así pues, dejando a criterio del autor de este documento la definición de '*cuadricóptero*', se entenderá éste por un "*vehículo aéreo no tripulado, impulsado por cuatro rotores con hélices situados normalmente con una configuración en equis o en cruz, el cual es controlado variando la velocidad relativa de cada rotor respecto a los demás*".

1.1 Descripción general del proyecto

El proyecto descrito en el documento consiste en la implementación de un sistema de control para cuadricópteros que permita un uso adecuado para principiantes. Para ello se debe realizar la construcción del aparato y su sistema de estabilización.

1.2 Justificación del proyecto

El principal motivo de la elección del trabajo ha sido el de aprender y profundizar en el campo de la mecatrónica y el control informatizado de diversos elementos a bajo nivel. El hecho de realizar el proyecto sobre un cuadricóptero se debe al auge que está teniendo este tipo de instrumentos y de los cuales quedan bastantes aspectos por investigar. Y conocer las bases de esta tecnología y haberlas aplicado en la realidad posibilita varias ampliaciones en un futuro.

1.3 Objetivos principales

Los objetivos principales a conseguir en este proyecto son los siguientes:

- Construcción y diseño del cuadricóptero.
- Establecer un sistema de comunicación por radio para el envío y recepción de datos al cuadricóptero.
- Implementación de un sistema de estabilización.
- Diseño y definición de restricciones a la libertad de vuelo para el aprendizaje de usuarios inexpertos.
- Creación de una plataforma de pruebas adecuada.
- Conseguir una correcta estabilización del cuadricóptero mediante el ajuste del sistema de estabilización implementado.

1.4 Estado del arte

En la actualidad, casi todos los cuadricópteros que se encuentran en el mercado, o bien están dedicados a un uso amateur para usuarios nóveles, o bien están enfocados a un uso profesional (ya sea para competición como para la grabación de eventos, etc.) y poseen diversos modos de control que ayudan a los usuarios menos expertos y que posibilitan a los más experimentados una gran libertad de acciones y movimientos.

Cabe destacar que el control acrobático de un cuadricóptero se basa en las velocidades de giro de sus ejes, es decir, si inclinamos la palanca del eje de alabeo unos 45 grados, un cuadricóptero acrobático no se inclinaría esos 45 grados sino que recibiría una señal de velocidad de giro en ese eje que, para usuarios con experiencia, posibilita un mayor control del aparato.

Así pues, se ha decidido colocar en este apartado una serie de proyectos que, además de poseer los sistemas de estabilización propios de cualquier cuadricóptero (los cuales se tratan de desarrollar en el proyecto), poseen unos sistemas de control enfocados a una amplia gama de usuarios, unas ampliaciones que se querrían realizar en un futuro en este trabajo.

Estos cuadricópteros han sido realizados la mayoría por empresas especializadas, las cuales cuentan con un historial bastante amplio de trabajos sobre el campo de los vehículos radio-control.

1.4.1 LaTrax Alias Quad-Rotor

LaTrax es una empresa fundada en 1974 por Jim Jenkins la cual fue pionera e innovadora en los sistemas de radio-control durante la década de 1970. Esta empresa se dedica a la construcción especializada de vehículos radio-control de altas prestaciones (3).

El modelo Alias Quad-Rotor de esta empresa implementa unas rutinas de control que se ejecutan en segundo plano y que no afectan a la experiencia de vuelo, las cuales permiten que el cuadricóptero recupere su posición de estabilidad en cualquiera que sea la situación en

la que se encuentre. Su sistema de control también ayuda a usuarios inexpertos a que se familiaricen rápido con el modelo y puedan realizar diversas maniobras acrobáticas (4).

1.4.2 Horizon Hobby

Horizon Hobby es una empresa fundada en Octubre de 1985 por Rick Stephens especializada en el hobby de los vehículos radio control. Posee varias sucursales alrededor del mundo: USA, UK, Francia, Alemania y China entre otros. Esta compañía ha desarrollado varias tecnologías que ayudan a mejorar el control de estos vehículos y potenciar la destreza con la que se puedan llegar a manejar (5).

Entre estas tecnologías hay una que destaca en lo que atañe al futuro de este proyecto, la cual es SAFE™ (Sensor Assisted Flight Envelope) (6). Esta tecnología implementa varias funciones para facilitar el control y evitar posibles accidentes. De estas funciones hay algunas que se asemejan a las posibles ampliaciones que se querrían realizar en un futuro para este trabajo:

- Control de altitud.
- Limitación de los ángulos de cabeceo y balanceo.
- Recuperación de la posición de vuelo estable.
- Aterrizaje automático.

2. Construcción, diseño y componentes

El cuadricóptero ha sido construido siguiendo un diseño que facilite la consecución de los objetivos del proyecto y teniendo en consideración los costes económicos totales de los elementos a utilizar.

2.1 Configuración

El diseño y configuraciones de los 'multi-rotores' varían dependiendo el número de rotores que tengan éstos. Para el caso de un cuadricóptero, cuyo número de rotores es cuatro, existen dos configuraciones posibles a la hora de repartir la potencia entre sus motores y el control del mismo.

2.1.1 Configuración en equis '×'

La configuración en equis es la que se utiliza en este proyecto. Esta configuración consiste en colocar los motores en forma de equis, de manera que en las cuatro caras del cuadricóptero siempre estén presentes dos motores:

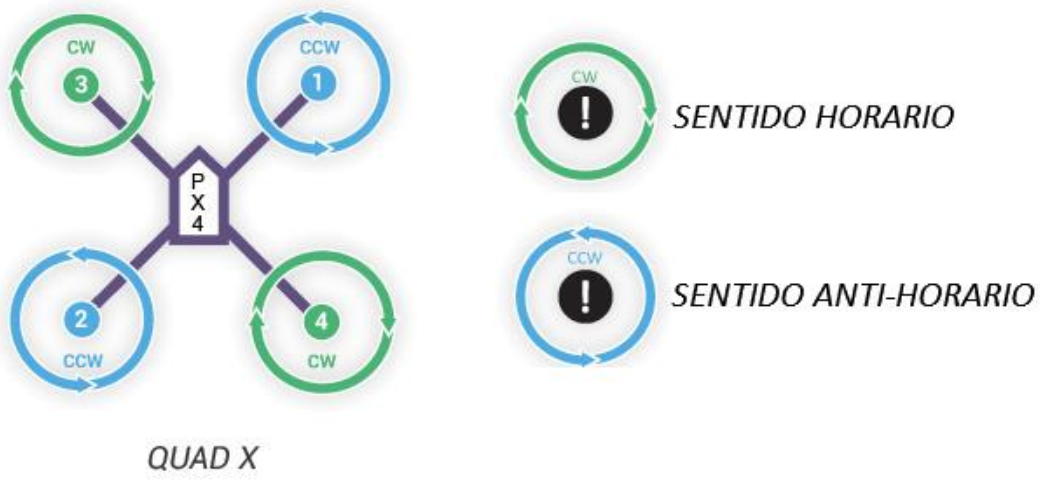

Figura 1 - Configuración en equis

El mayor beneficio que proporciona esta configuración es una cuestión física de repartición de potencia entre los motores, de forma que la carga individual que tenga que soportar cada uno es menor que la soportada en la configuración en cruz. Por otro lado, el algoritmo que controla la diferencia de potencia entre los motores es algo más complejo.

El hecho de que exista menor carga individual en cada motor en esta configuración que en la configuración en cruz se debe principalmente a cuestiones estadísticas del uso de los cuadricópteros.

Las direcciones que más repercusión van a tener en el movimiento del aparato van a ser las direcciones ortogonales “adelante/atrás” e “izquierda/derecha”, pues a la hora de controlarlo van a ser las más utilizadas. Esta configuración sitúa los motores de forma que en esas direcciones se encuentren dos de ellos, de manera que los cuatro motores (un par aumentando la potencia y el otro disminuyéndola) se repartirán la carga necesaria para hacer avanzar el cuadricóptero.

2.1.2 Configuración en cruz ‘+’

Esta configuración sitúa un motor en cada dirección ortogonal “adelante/atrás” e “izquierda/derecha”. Las características son las opuestas a las mencionadas en la configuración en equis.

El algoritmo que controla la diferencia de potencia entre los motores es un poco más simple pero a la hora de repartir la carga entre los motores es peor, puesto que un solo par de motores ha de proveer la potencia necesaria para mantener la inclinación del cuadricóptero en esas direcciones.

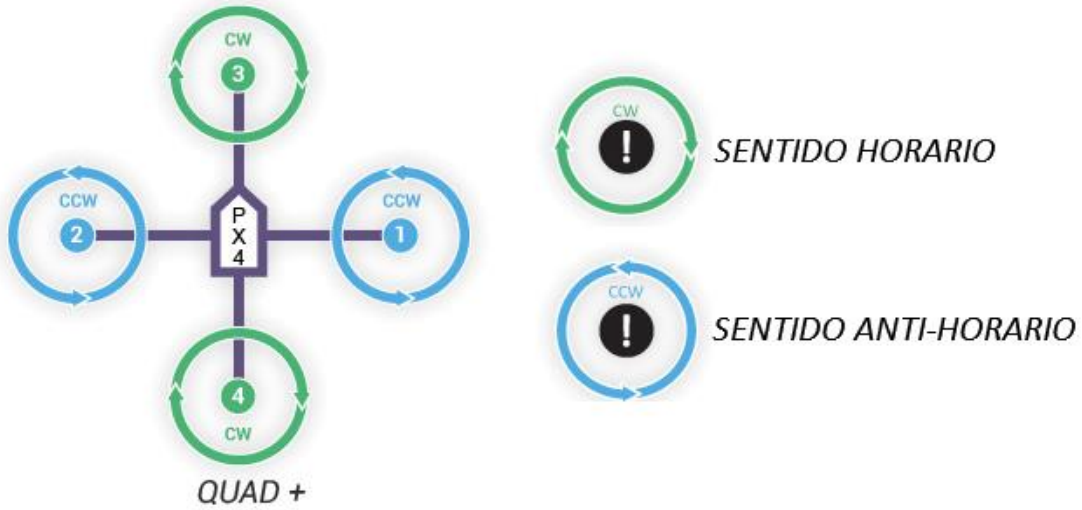

Figura 2 - Configuración en cruz

2.1.3 Sentido de giro de los motores

Tal y como se puede observar en las figuras “*Figura 1 - Configuración en equis*” y “*Figura 2 - Configuración en cruz*”, el sentido de giro de los motores es diferente dependiendo de su posición. Más concretamente, cada par de motores opuestos giran en el mismo sentido y cada par gira en sentido contrario al otro par.

Cuando se eleva un motor con una hélice en el aire, el estator del motor tiende a girar sobre sí mismo en el sentido contrario en el que gira el rotor con la hélice. En un helicóptero, este efecto se contrarresta colocando otra hélice en el lugar más alejado de la cola (para ejercer mayor fuerza por la “ley de la palanca”) de modo que ésta se encargue de mantener la posición del aparato en el eje de guiñada.

En el caso de un 'multi-rotor', este efecto se puede contrarrestar con otra hélice girando en el sentido contrario a la primera. Para un cuadricóptero, al poseer cuatro rotores, el efecto se evita tal y como se ha descrito en el primer parágrafo: haciendo girar los motores de una diagonal (en el caso de la configuración en equis) en el sentido opuesto a los de la otra.

2.2 Estructura

Para la estructura principal del cuadricóptero se ha elegido un armazón ('*frame*' en inglés) que cumpla con varias características importantes. Tales como evitar la rotura de algún componente o del propio armazón, evitar costes elevados o que tenga una buena aerodinámica de vuelo.

A la hora de elegir un armazón apropiado se debe tener en cuenta el tamaño y peso de éste. Si se opta por una estructura bastante grande existe el problema del peso, habría que utilizar motores más potentes y hélices más grandes, lo que aumentaría los costes de producción. Si por el contrario se elige una estructura más pequeña, la estabilización de la misma y la poca resistencia que ofrece a los embates del viento se convierten en un problema a la hora de implementar el control del aparato.

Por otro lado, se deben tener en cuenta aspectos como la flexibilidad del material y la resistencia que ofrezca a los golpes. En este caso, el armazón elegido cuenta con unos brazos de nylon poliamida y un cuerpo fabricado con fibra de vidrio. Estos brazos son lo suficiente rígidos como para soportar golpes y lo bastante flexibles como para no partirse. Y el cuerpo del armazón protege de forma eficiente los elementos electrónicos que se encargan del control del cuadricóptero.

Con estas características en mente, se ha optado por el armazón Q450 de la marca Diatone, la cual es bastante popular entre los fabricantes de cuadricópteros caseros. Este armazón de 450 mm, brazos de nylon poliamida y cuerpo de fibra de vidrio cumple bastante bien los requisitos de tamaño intermedio, resistencia a los golpes, poco peso y de relativamente bajo coste económico.

Figura 3 - Diatone Q450, vista superior.

Figura 4 - Diatone Q450, vista inferior.

2.3 Motores

La elección de los motores es muy importante a la hora de construir un cuadricóptero ya que es sobre éstos donde se sustenta el aparato y sobre los que hay que aplicar las acciones de control, si los motores no responden adecuadamente el trabajo se dificulta y es muy probable que el control no se pueda llevar a cabo.

Hay que tener muchos aspectos en cuenta, los cuales influirán también en la elección de otros componentes que se complementan o son requeridos por los motores para funcionar correctamente. Estos aspectos son básicamente la velocidad del motor, la potencia, el consumo, el tipo de motor y el precio.

Los motores elegidos deben ser lo suficientemente veloces como para, con las hélices que se elijan (las cuales dependerán de esta velocidad), elevar el cuadricóptero en el aire y permitir una acción de control de calidad.

Por otro lado, la potencia que utilicen los motores y el amperaje que consuman se verán reflejados en la batería que se utilice, la cual se adoptará en función de estas dos características, que determinarán la duración de la batería, el precio y el peso de ésta. Todos los factores son determinantes entre ellos por lo que cada uno afecta a los demás, se ha de llegar a un convenio para conseguir la configuración más adecuada.

Otro de los aspectos que es muy importante es el tipo de motor. En la actualidad existen dos tipos de motores de corriente continua para su uso en cuadricópteros: los motores con escobillas y los motores sin escobillas (del inglés, '*brushless*').

La diferencia entre estos motores radica precisamente en el uso o no de escobillas. En los motores con escobillas, éstas son usadas para transmitir la electricidad a la bobina, la cual se encuentra en el rotor y lo hace girar; mientras que los imanes se encuentran en el estator y se mantienen fijos.

El hecho de tener escobillas provoca el rozamiento de éstas con las placas que llevan la corriente, generando calor y deteriorando progresivamente el material. Es por esto que estos motores requieren de mantenimiento y son menos eficientes que los motores sin escobillas.

En el caso de los motores sin escobillas los imanes se encuentran en el rotor y las bobinas en el estator, de forma que no se necesitan escobillas para transmitir la electricidad al rotor. Los primeros motores de este tipo funcionaban con corriente alterna pero gracias a los circuitos electrónicos se puede simular un patrón que alimente en el momento adecuado las bobinas adecuadas para conseguir el giro.

El encargado de simular este patrón es un ESC ('*Electronic Speed Controller*') el cual proporcionará los pulsos en el orden correspondiente. A la hora de elegir un ESC se debe prestar atención al amperaje soportado por éstos, que ha de ser mayor que el amperaje que consuma el motor.

En cuanto al precio, los motores con escobillas son mucho más simples y más baratos que los '*brushless*' pero su rendimiento es menor y merece la pena decantarse por estos últimos. También hay que tener en cuenta que por cada motor ha de existir un ESC.

Figura 5 - Motor DYS BE2208

Tras sopesar todas las características se han elegido los motores BE2208 del fabricante DYS ("*Figura 5 - Motor DYS BE2208"*). Estos motores son de 1400 KV, lo cual nos indica que tienen poca tracción pero consiguen una gran velocidad. Dado que el elemento a mover es pequeño (las hélices) y se necesita una velocidad lo suficientemente alta como para elevar el aparato, estos motores son perfectos para este trabajo. Conociendo los KV, una batería de 11.1 V (que es la que se utilizará) el motor alcanzará aproximadamente unas 15.540 revoluciones por minuto (rpm). Por otro lado, el motor puede llegar a consumir hasta 19 A por lo que se escogerá una ESC capaz de resistir un amperaje mayor, en este caso 25 A.

2.4 Hélices

Las hélices serán las encargadas de propulsar el cuadricóptero en el aire y se han de elegir de modo que cumplan una serie de características. Son las que más golpes van a sufrir de entre todos los elementos que conforman el aparato y las más frágiles.

Las consideraciones a tener en cuenta sobre las hélices son el peso, el material con el que han sido fabricadas, el ángulo de la curvatura desde el centro hasta el extremo de la hélice, el tamaño y el número de aspas.

El peso influirá en la fuerza que deban utilizar los motores para moverlas. El material nos proporcionará la resistencia y flexibilidad que necesitamos e influirá sobre el peso de la hélice. El ángulo de curvatura nos indica la función para la que han sido fabricadas, si el ángulo es pequeño, la hélice ofrecerá menos resistencia al aire y por lo tanto menos propulsión, de forma contraria sucede si el ángulo es más grande, la propulsión ofrecida será mayor pero la resistencia al aire también y al motor le costará más moverla.

Por otro lado, el tamaño de la pala de la hélice también influye en el peso y en el impulso que proporciona la hélice. En el aire, unas palas demasiado anchas no ejercerán la propulsión necesaria con un motor normal como para que cualquier aparato se eleve, puesto que el aire tiene muy poca densidad.

Continuando con el impulso, otro de los factores que influye en éste es el número de aspas de la hélice. El impulso de la hélice es inversamente proporcional al número de palas que ésta posea pero, en cambio, cada pala soportará individualmente una carga mucho menor cuanto más elevado sea el número de éstas. La mejor relación entre propulsión y carga para helicópteros se obtiene con dos aspas. Y hay que tener en cuenta que el número de aspas también afectará al peso de la hélice.

Figura 6 - Par de hélices Gemfan 1045

Las hélices 1045 de la marca Gemfan cumplen las características recomendadas por el fabricante de los motores. Son unas hélices de dos aspas que tienen un tamaño de 10 pulgadas con un ángulo de torsión de 4.5 grados. Están fabricadas en fibra de carbono con filamentos de nylon, una mezcla de materiales que les proporciona poco peso y bastante flexibilidad para evitar roturas.

2.5 *Electronic Speed Controller*

Un '*Electronic Speed Controller*' o controlador de velocidad electrónico, es un circuito que se utiliza para variar la velocidad de un motor eléctrico, generalmente '*brushless*', proporcionando una corriente trifásica de poco voltaje para mover el motor.

Figura 7 - ESC, Emax 25 A

En un ESC se pueden observar ocho cables distintos que son los necesarios para poder conectar el motor, una fuente de alimentación y los cables por los que el ESC recibirá los datos del control ("*Figura 7 - ESC, Emax 25 A*").

En este caso se ha optado por un variador de la marca Emax que soporta hasta 25 A ya que el motor alcanza a consumir una potencia máxima de 19 A y es necesario un pequeño margen para evitar posibles sobrecorrientes.

Los tres cables que salen en el extremo derecho de la "*Figura 7 - ESC, Emax 25 A*" son los que se conectarán al motor sin escobillas y que portarán la corriente trifásica simulada por el variador. Los dos cables de mayor tamaño de la izquierda (rojo y negro) son los cables de alimentación del ESC, a donde irá conectada la batería que le proporcionará la potencia al motor.

Por último, los tres cables restantes (rojo, naranja y marrón) son los que se utilizan para el control de la velocidad del motor. El cable marrón es la masa, que se ha de conectar a la masa del microcontrolador que nos proporcione los datos de control. El cable rojo es un suministro de voltaje que ofrece el ESC para dar corriente a otros circuitos. En este caso, el ESC posee un elemento (BEC) que se encarga de limitar el voltaje que recibe de la batería a 5 V para que el sistema de control no se vea afectado por el voltaje de ésta, el cual es de 11.1 V.

De todas formas, dado que el sistema de control está alimentado por otra batería para evitar posibles sobretensiones provenientes de los motores y las ESCs, no es necesario conectar el cable rojo a ningún pin.

El cable naranja es el que se utiliza para enviar los pulsos que indicarán al ESC la velocidad a la que deben girar los motores. Para ello se utiliza lo que se conoce como modulación por ancho de pulso o '*Pulse Width Modulation*' (PWM). Este sistema consiste en utilizar un valor digital (0 o 1) para representar un valor analógico en el tiempo.

Para ello se coloca a '1' o a '0' la señal durante un tiempo que determina el valor analógico que la señal va a tener en un ciclo. Es decir, si se supone un tiempo de ciclo de 2 ns, y el tiempo que está la señal a '1' (5 V) es de 1 ns (el 50%), el valor representado es la mitad del valor de referencia usado (5 V -> 2.5 V).

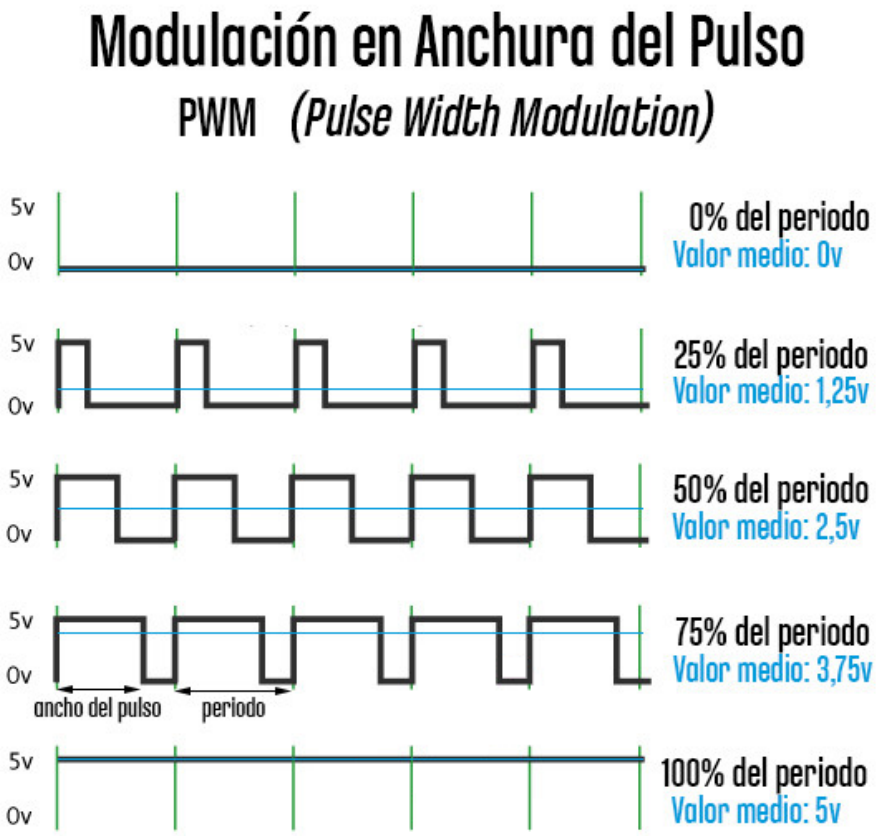

Figura 8 - Ejemplos de modulación por anchura de pulso

Para no alterar el funcionamiento del sistema que consume esta señal, los ciclos han de tener una longitud lo suficientemente pequeña como para que el sistema aprecie la señal de forma analógica y lo más suavizada posible.

2.6 Baterías

Las fuentes de alimentación que se utilizarán para el cuadricóptero serán baterías de polímero de litio ('*LiPo*' Baterías) las cuales son ampliamente utilizadas en el mundo del radio-control por su capacidad y ligereza.

Se utilizarán dos tipos de baterías para el aparato. Una para el sistema de ESCs y motores y otra para los componentes electrónicos que serán los encargados de llevar a cabo el control de la máquina y la recepción de los datos de la emisora.

Las características a tener en cuenta a la hora de elegir las baterías son la capacidad, el ratio de descarga y el número de celdas. La capacidad de la batería depende del número de amperios que sea capaz de descargar en una hora. Se especifica en mA/h.

El ratio de carga/descarga se especifica mediante la letra 'C' y es una constante creada por los fabricantes de baterías para indicar más fácilmente la intensidad a la que debe cargarse o descargarse una batería sin que ésta sufra daños. Esta constante es el producto de dividir los mA/h de la batería entre 1000. De esta forma, si tenemos una batería de 30C y cuya capacidad es de 5000 mA/h, la batería no se debería de cargar o descargar con una intensidad mayor de 150 A.

El número de celdas de una batería determina el voltaje de ésta. Cada celda es una batería que suele rondar los 3.7 V y que está conectada en serie con otras, de modo que estos voltajes se suman para obtener uno mayor. La mayoría de las baterías tienen entre una y tres celdas pero para los casos en los que se requiera más potencia serán necesarias baterías con más celdas o trabajar con más de una batería. El número de celdas de una batería está representado por un número seguido de la letra 'S'.

Figura 9 - Batería ZOP Power 3S 5000 mA/h 30C

Como batería principal para alimentar los motores y ESCs se utilizará la mostrada en la "*Figura 9 - Batería ZOP Power 3S 5000mA/h 30C*" la cual pertenece al fabricante ZOP. Esta batería posee tres celdas que proporcionarán un voltaje de unos 11.1 V, cuenta con una capacidad de 5000 mA/h y no se debe de cargar o descargar a más de 150 A. Con esta batería se cubre perfectamente las necesidades de los motores y ESCs y alcanza para una duración de vuelo de entre 10 o 15 minutos.

Figura 10 - Batería ZOP Power 3S 800 mA/h 25C

Para alimentar los circuitos electrónicos que realizarán el control no se precisa de una batería muy potente puesto que estos componentes consumen muy poco. Por ello se ha optado por una batería del mismo fabricante que la anterior, de tres celdas también, pero que tiene una capacidad de 800 mA/h y una constante de carga/descarga de 25C.

2.7 Unidad de Medida Inercial

Una unidad de medida inercial o en inglés "*inertial measurement unit*" (IMU) es un componente electrónico que permite medir las fuerzas de inercia que actúan sobre éste. Una IMU consta de un acelerómetro y un giroscopio aunque en la mayoría de los casos también suele llevar un magnetómetro. Este conjunto de aparatos nos permite medir fuerzas como la gravedad, la inercia aplicada por el movimiento no-gravitatorio, el ángulo y velocidad de giro y la posición respecto a los polos.

Para las primeras pruebas se utilizó una IMU GY-85 la cual poseía un acelerómetro ADXL345 de tres ejes (XYZ), un giroscopio ITG3205 también de tres ejes y un magnetómetro HMC5883L del mismo número de ejes.

Figura 11 - IMU GY-85 de 9 ejes

Dado que esta primera IMU tenía fallos en el eje Z del acelerómetro se optó por utilizar una MPU 9250, de la cual se tendría que adaptar o encontrar una librería portada a Arduino, ya que el modelo de esta MPU era bastante reciente.

Figura 12 - MPU 9250

Estos dispositivos funcionaban ambos mediante el bus I^2C desarrollado por Philips, el cual consta únicamente de dos cables de datos más la alimentación y la masa, y al que se le pueden añadir más dispositivos sin la necesidad de más cables. Es una interfaz ampliamente utilizada, además de por el bajo número de cables, por ser multi-maestro y permitir una gran cantidad de dispositivos conectados simultáneamente.

Este bus es perfecto para utilizarlo con el microcontrolador elegido dadas sus características, pues el microcontrolador es de un tamaño reducido y tiene pocos pines digitales para la cantidad de conexiones que puede llegar a requerir un cuadricóptero de uso profesional.

La MPU 9250 ("*Figura 12 - MPU 9250*") es un modelo reciente del cual no existen muchas referencias en cuanto a su uso en proyectos por lo que es difícil encontrar librerías para Arduino que hayan sido revisadas por muchos usuarios y sean fiables. Aun así, se ha optado por el uso de la librería de InvenSense MPU/DMP 5.1, la cual ha sido portada a Arduino por el usuario "*gregd72002*"(7) en los foros oficiales de Arduino y que se puede encontrar en GitHub (8).

Por otro lado esta IMU cuenta, además de con los elementos básicos (Acelerómetro, Giroscopio y Magnetómetro), de un barómetro para medir presiones atmosféricas y de un sensor de temperatura. Estos elementos, al estar implementados dentro del mismo componente, evitarán el uso de más cableado y proporcionarán unos datos bastante útiles para realizar el control de altitud (barómetro) y recogida de muestras de temperatura (sensor de temperatura).

2.7.1 Acelerómetro

Un acelerómetro es un dispositivo que mide las fuerzas inerciales que se le apliquen, como por ejemplo la gravedad. Esta funcionalidad es muy útil para comprobar la inclinación del cuadricóptero respecto al eje vertical. Al medir la aceleración en los tres ejes de coordenadas (XYZ) y tener una referencia constante como es la gravedad, se puede conocer la dirección en la que se encuentra el suelo y saber la inclinación respecto a éste.

Existen muchos tipos de acelerómetros pero los utilizados en electrónica suelen ser acelerómetros capacitivos, estos acelerómetros miden las fuerzas inerciales utilizando dos placas de un condensador, una de ellas fija y la otra expuesta a la inercia del movimiento. La diferencia de capacidad producida por el movimiento de las placas del condensador es recogida y amplificada para obtener un voltaje de salida útil en un circuito digital (9).

2.7.2 Giroscopio

El giroscopio permite medir la velocidad angular que posea el objeto en el que se encuentra. De esta forma, con los datos de la velocidad angular se puede calcular el error de posición que tenga el aparato respecto a una posición inicial y comprobar así su posición actual.

2.7.3 Magnetómetro

Los magnetómetros sirven para medir la fuerza de los campos magnéticos y su dirección. Existen magnetómetros escalares y vectoriales, los magnetómetros escalares miden la fuerza del campo magnético y los vectoriales calculan la fuerza del campo magnético en una dirección.

Los magnetómetros que se van a usar en el proyecto son estos últimos puesto que miden varias componentes del campo magnético de forma electrónica. Con tres

magnetómetros (o magnetómetro de tres ejes) situados ortogonalmente se puede averiguar las componentes de un campo magnético en un punto por lo que se podría averiguar la posición y dirección de ese punto en relación a la magnetosfera terrestre. (10)

2.7.4 Barómetro y sensor de temperatura

El barómetro es utilizado para cuantificar la presión atmosférica, con esta presión, que varía en función de la altura a la cual se esté midiendo, se puede obtener la altura a la que se encuentra el sensor.

Para la medida de la presión atmosférica se utilizan varios métodos, en concreto se utilizan un conjunto para los barómetros electrónicos, los cuales han de tener un tamaño reducido. Estos métodos se basan en la deformación o el movimiento de un pequeño material por el efecto de la presión, que es detectada mediante cambios en la resistencia, conductividad o capacidad de un material.

Existen barómetros resistivos, los cuales funcionan mediante una pletina flexible a la cual hay enganchada unas cintas extensiométricas, estas cintas cambian su resistencia cuando son estiradas de forma que cuando la presión deforma la pletina, estas se estiran y cambian su resistencia. Y barómetros piezoresistivos, los cuales siguen el mismo funcionamiento pero en este caso es la propia membrana flexible la que cambia su resistencia al estar hecha de un material piezoresistivo.

También existen barómetros capacitivos, en los cuales la presión mueve una de las paredes de un condensador, alterando por tanto su capacidad y permitiendo la medida de la presión.

Por otro lado existen materiales piezoeléctricos (ej. cristales de cuarzo) que cuando se comprimen se produce una polarización eléctrica de sus caras opuestas. Esta compresión produce una descolocación de la estructura cristalina y genera un momento dipolar dependiente de la dirección en la que se ejerce la presión. La intensidad de la carga producida por el material es proporcional a la presión aplicada sobre éste y el momento dipolar indica su dirección. Cabe recalcar que este método de medida es solo apto para presiones dinámicas y que el uso de estos sensores en la actualidad está limitado a aplicaciones especiales. (11)

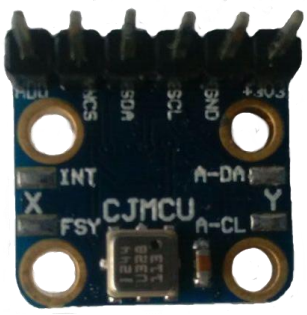

Figura 13 - Sensor BMP 180 de Bosch incrustado en la parte trasera de la MPU9250

El sensor BMP 180 de Bosch es el que se encuentra disponible en la MPU 9250 y, aparte de la presión atmosférica, permite medir la temperatura. Este sensor puede funcionar a tan solo 3 µA y en el modo de alta resolución es capaz de medir la altitud con un ruido de poco más de 17 centímetros (12).

2.8 Microcontrolador

Un microcontrolador es un dispositivo electrónico compuesto básicamente por un procesador, una memoria RAM, una memoria ROM donde almacenar el programa a ejecutar y una serie de periféricos para interactuar con el exterior como lectores analógicos, salidas digitales PWM, etc.

A estos dispositivos se les carga el programa a ejecutar en la memoria ROM y actúan exactamente como cualquier ordenador, el procesador ejecuta las instrucciones del programa y carga y lee datos de la memoria RAM, interactúa con los periféricos, recibe interrupciones, etc.

Se denominan microcontroladores por ser de tamaño reducido y disponer únicamente de los elementos básicos para su funcionamiento, evitando componentes innecesarios, los cuales se pueden añadir en caso de necesitarlo de forma externa. El microcontrolador es el elemento básico y central de todo aquel sistema que se utilice para controlar algo electrónicamente (13).

2.8.1 Arduino

Arduino es una plataforma que permite y facilita en gran medida el trabajo con microcontroladores (en concreto de la familia Atmel) a usuarios nóveles, proveyendo componentes sencillos pero capaces de ofrecer una gran funcionalidad.

Arduino es un proyecto de código abierto que está basado en el uso fácil del hardware y el software. Nacido en el Instituto de Diseño de Interacción Ivrea (‘*Ivrea Interaction Design Institute*’), fue concebido en un principio para estudiantes y usuarios con un nivel bajo de conocimientos en electrónica y programación. A medida que la comunidad fue creciendo, Arduino fue adaptándose para cubrir nuevas necesidades y desarrollar hardware más potente y enfocado a proyectos más complejos, pero siempre desde el punto de un uso cómodo y fácil.

Al ser tanto software como hardware de código abierto, Arduino permite que los usuarios creen y modifiquen sus productos para adaptarlos a sus necesidades. Y, gracias a la gran comunidad existente de contribuyentes, el proyecto recibe una gran retroalimentación que mejora y hace avanzar el producto. (14)

Figura 14 - Logotipo de Arduino

Así pues, la elección de Arduino como microcontrolador para el proyecto se debe a su bajo costo y a su uso rápido y relativamente fácil, puesto que muchos de los problemas que se encontrarían con un microcontrolador normal vienen ya solventados de antemano gracias al diseño y los componentes que conforman el hardware; y a las librerías y el entorno de programación gratuito que se ofrecen que, por otra parte, son multiplataforma.

2.8.2 Arduino Nano

La plataforma Arduino ofrece una gran variedad de productos (de ahora en adelante 'arduinos') a sus usuarios, cada uno enfocado en un tipo de uso o una función en concreto, aunque brindando una amplia variedad de posibilidades.

Existen desde productos de aprendizaje, como son el Arduino Uno o el Arduino Pro, pasando por arduinos de más alto nivel y capacidad, como son el Arduino Mega o Due, arduinos dedicados al internet de las cosas como el Arduino Yún, hasta arduinos enfocados en el uso de 'wearables' (Arduinos Lilypad o Gemma) e impresoras 3D (kit Materia 101). (15)

Aunque enfocados hacia un uso y un público, las características y especificaciones de cada arduino deben ser consideradas a la hora de hacer una elección. En este caso se ha escogido el Arduino Nano que, aunque pertenezca a la zona de productos dedicados a la toma de contacto con la plataforma, sus atributos posibilitan su uso en este tipo de proyectos.

Arduino Nano es pequeño y con un número de pines lo suficientemente alto para este proyecto. Posee un microcontrolador ATmega328 en su versión 3.x (que es la utilizada) y cuya velocidad de reloj son 16 MHz, funciona a un voltaje de 5V recibiendo de la fuente de 7 a 12 voltios como voltaje recomendado y utilizando una corriente de 40 mA por cada pin en uso.

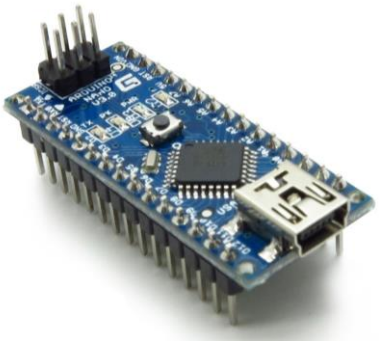

Figura 15 - Arduino Nano

Tiene suficiente memoria estática como para almacenar el programa que requerirá el cuadricóptero y suficiente capacidad de proceso y memoria RAM para ejecutar los algoritmos de control, tanto de estabilidad como de las funcionalidades de posicionamiento y lectura de radio, y mantener todos los datos que se irán utilizando sin problemas. (16)

2.9 Emisora y receptor de radio

La emisora utilizada para el envío de señales por radio al cuadricóptero es el modelo FS-TH9XB de la marca FlySky ("*Figura 16 - Emisora FS-TH9XB del fabricante FlySky*"). Esta emisora nos permite hasta 8 canales por los que enviar información al aparato.

El número de canales básicos para utilizar en este trabajo serían 4 de ellos: *aceleración*, *cabeceo*, *alabeo* y *guiñada*. Los canales restantes se pueden utilizar para diversas funciones de control como cambiar el modo de vuelo, realizar alguna foto o vídeo, etc.

Figura 16 - Emisora FS-TH9XB del fabricante FlySky

Junto a la emisora se ha escogido el receptor FS-R9B del mismo fabricante ("*Figura 17 - Receptor de radio FS-R9B de la marca FlySky*"). El receptor produce una señal PPM ("*Pulse Position Modulation*" o modulación por posición de pulso) por cada canal. Esta señal será recogida por un Arduino Nano que hará de receptor de todas las señales de radio ("*Sección 3.1.1*").

Figura 17 - Receptor de radio FS-R9B de la marca FlySky

La modulación por posición de pulso consiste en codificar un valor en el tiempo, de forma que se detecte por el instante en el que se recibe un pulso. La señal comienza a enviarse a un nivel '*alto*' y, cuando el tiempo de envío alcanza el valor codificado, se envía un pulso a

nivel ‘bajo’. El emisor envía los 8 canales en una misma señal en la que viajan en orden y el receptor se encarga de dividirlos y enviarlos por cada uno de sus 8 pines.

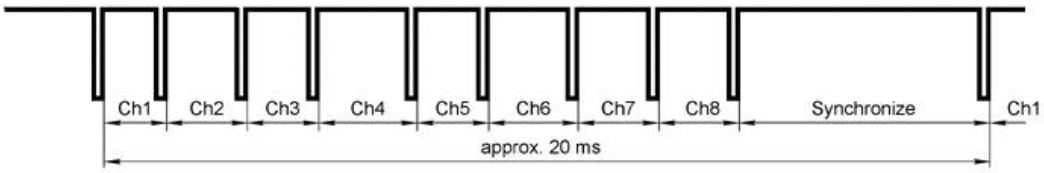

Figura 18 - Ejemplo de modulación PPM

Esta modulación permite juntar varios canales en una misma señal, de forma que cuando se detecta el pulso que indica el valor de un canal, acto seguido comienza el siguiente canal, hasta detectar el pulso de ese canal, continuando con el siguiente y de ese modo sucesivamente.

3. Esquema de conexión

A continuación se muestra el esquema de conexiones entre los diferentes componentes del proyecto. Éste ha sido realizado con la herramienta Fritzing (17), la cual es gratuita y bastante completa, es la utilizada por Arduino en los tutoriales de su página web.

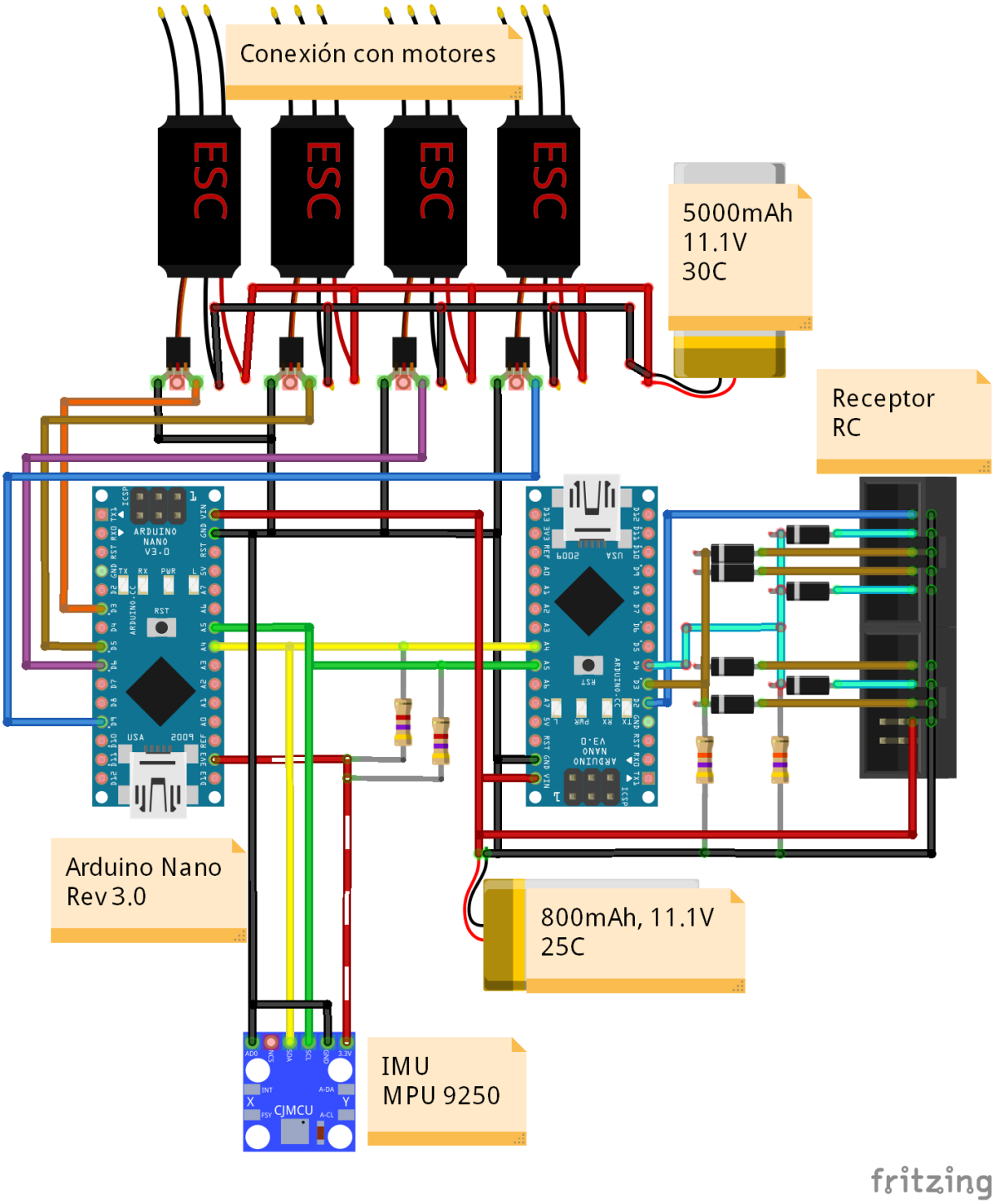

Figura 19 - Esquema de conexiones del proyecto

Como se puede observar en la "*Figura 19 - Esquema de conexiones del proyecto*", el esquema de conexiones no es del todo complejo, consta de conexiones con los controladores electrónicos de los motores, de un bus I^2C que conecta dos arduinos Nano y la MPU9250, fuentes de alimentación y un sistema de diodos para proteger el receptor de corrientes en sentido contrario.

3.1 Arduinos

Comenzando con los arduinos, este sistema ha sido diseñado con dos de ellos por motivos de rapidez en la ejecución de instrucciones, intentando minimizar todo lo posible los retardos por comunicaciones.

El receptor de radio divide una frecuencia PPM ("*Figura 18 - Ejemplo de modulación PPM*") en sus ocho canales y éstas llegan en un orden, es por ello que el total de los datos del receptor tardan en recibirse mucho más que lo que tarda en leerse una variable de otro arduino por I^2C.

Así pues, uno de los arduinos es el encargado de controlar el cuadricóptero, obteniendo para ello datos del receptor y de la IMU lo más rápido posible. Y el otro arduino se centrará en la obtención de los datos provenientes de la radio, su almacenamiento en variables y su envío cuando lo necesite el otro arduino. El código de ambos microcontroladores se explicará con más detalle en la sección "*4. Software/Firmware*".

3.1.1 Arduino receptor

El arduino que se encarga de la lectura de datos del receptor contiene un programa que ejecuta instrucciones más específicas que las que se ejecutan en el otro microcontrolador. Esto quiere decir que no son instrucciones básicas que ofrece la API de Arduino como podría ser leer un valor determinado en un pin. Con estas instrucciones se puede acceder al nivel más bajo del microcontrolador y leer y escribir registros, los cuales ofrecerán unas posibilidades mucho más amplias que el uso de instrucciones básicas.

El motivo por el cual se hace esto es el tiempo que lleva realizar alguna instrucción básica comparada con una instrucción específica. Para una lectura de un pin digital con la instrucción básica '*digitalRead()*', arduino ejecuta internamente una docena de instrucciones, lo cual puede llegar a consumir un tiempo de unos 4,8 µs. En cambio, para realizar la misma lectura, utilizando una instrucción que acceda a un registro del microcontrolador, ésta puede tardar únicamente unos 0,06 µs (18).

Sin estas instrucciones para acceder directamente a los registros del microcontrolador no se podría hacer una buena lectura de los canales de la radio puesto que el tiempo que tardarían podría impedir la lectura de alguno de los canales, con los consecuentes fallos de lectura de los canales restantes.

3.1.2 Arduino controlador

En este arduino es en el cual se ejecutará el código principal para el control del cuadricóptero. Será el encargado de leer los valores de los sensores y la radio, de ejecutar las diversas rutinas de control para la estabilización del aparato y de proporcionar la salida adecuada a los controladores de los motores.

En este microcontrolador se encuentran las diversas librerías que facilitan la lectura de los sensores y todo el código desarrollado en el trabajo para el control del cuadricóptero y el acceso a los datos del arduino receptor.

3.2 Receptor de radio

El receptor de radio control proveerá la señal del emisor dividida en ocho señales, cada una por un canal. La emisión por cada canal de estas señales PPM se realiza en orden, en el caso de este receptor el orden es el siguiente: 1, 3, 2, 4, 5, 6, 7, 8 (con los canales 3 y 2 intercambiados en el orden natural).

Estos canales han sido conectados a tres de los pines del arduino receptor para ahorrar espacio en pines y, dado que hay varios canales conectados entre sí (3, 4, 6 y 8 a pin 3; 2, 5 y 7 a pin 4), todos los canales excepto el 1 (el cual está conectado directamente al pin 2) se encuentran protegidos por diodos y resistencias de pull-down que impedirán que reciban corrientes en sentido contrario.

Cuando el arduino receptor reciba la señal del primer canal, comenzará a leer las señales de los demás canales en orden, alternando únicamente entre los pines 3 y 4. La disminución en el número de pines utilizados no afecta a la rapidez de la lectura de datos pero proporciona la posibilidad de una ampliación que requiera pines digitales en el futuro.

3.3 MPU 9250

La MPU 9250 se encuentra conectada mediante el bus I^2C al arduino controlador y proveerá las lecturas de inclinación y velocidad de giro. Estas lecturas vienen facilitadas gracias a la librería de InvenSense MPU/DMP 5.1 que portó a Arduino el usuario "*gregd72002*"(7), la cual realiza un filtrado de los valores de forma que evita la implementación aparte de un filtro de kalman o alguno similar.

Hay que destacar que esta MPU funciona con valores de voltaje de 3.3 V por lo que la totalidad del bus I^2C se encuentra a este voltaje. En un principio se realizó una conversión de voltajes con un convertidor de 5 V a 3.3 V (19) siguiendo la configuración recomendada en el documento "*AN10441 Level shifting techniques in I2C-bus design*" (20) de *NXP Semiconductors* (antiguamente *Philips Semiconductors*), los desarrolladores de este bus.

En cambio, esta configuración no produjo los resultados esperados por lo que se decidió que el voltaje del bus fuera de 3.3 V para todos los componentes, de forma que lo pudieran soportar tanto la MPU como los Arduinos.

3.4 ESCs

Los controladores de velocidad de los motores están conectados por su entrada de PWM a los pines 3, 5, 6 y 9 del arduino de control y por su masa a la masa total del circuito. La salida de voltaje que poseen las ESCs proveniente del BEQ que regula el voltaje (situada entre su entrada PWM y su masa) no se encuentra conectada puesto que el circuito de control ya está alimentado por la batería de 800 mA/h. A su vez, los controladores reciben corriente de la batería de 5000 mA/h para alimentar con ésta los motores.

3.5 Baterías

Como se ha remarcado varias veces a lo largo del trabajo, el sistema que proporciona la alimentación de corriente al circuito se encuentra dividido entre dos baterías. Una de ellas, de 800 mA/h, es la encargada de proporcionar corriente al circuito electrónico de control de forma que, como el consumo de este circuito es bastante bajo, su duración sea lo suficientemente alta. Esta batería está conectada al receptor de radio y a los dos arduinos.

Por otro lado, la batería de 5000 mA/h se encuentra conectada a la entrada de potencia de los controladores de los motores, de esta forma el voltaje y el amperaje que consumen los motores no afecta al circuito de control.

4. Software/Firmware

En esta sección se expondrá el código programado sobre la plataforma Arduino, tanto del arduino de control como del arduino receptor, y se explicarán los procedimientos utilizados para la realización del sistema de estabilización del aparato, las comunicaciones por el bus I^2C entre los arduinos, el algoritmo de recepción de los datos del receptor de radio, los algoritmos PID y las señales que se envían a los controladores de los motores.

4.1 Recepción de datos por radio

Se comenzará a explicar el código de la recepción de los datos de la radio ya que es un procedimiento que se podría aislar de lo que es el control de estabilidad del cuadricóptero, este sistema únicamente se encarga de obtener los datos de la radio y ofrecérselos al arduino que realiza el control de una forma rápida y sencilla. Esta forma de obtención de los datos es perfectamente reemplazable por otra que sea igual de válida y ello no afectaría al control del aparato.

Como se ha revelado en las secciones "*3.1.1 Arduino receptor*" y "*3.2 Receptor de radio*", el arduino receptor se encarga de obtener los datos del receptor de radio únicamente por tres de sus pines y con una serie de instrucciones que acceden a los registros del microcontrolador ATmega328 para mayor rapidez.

El código que se ha utilizado para ello es una modificación del código proporcionado por Carlos Porta (18) ('*Cheyenne*' en los foros oficiales de Arduino). Este código permite realizar una lectura del tiempo que tarda cada canal en introducir un pulso a '0', almacenar el dato y enviarlo por el bus I^2C cuando sea requerido.

El programa comienza esperando un pulso del pin 2 (canal 1) a nivel alto ('1'), de esta forma se detecta cuando ha empezado a enviarse la señal. Es en ese momento cuando el arduino contará el tiempo que tarda la señal en dar el pulso a '0'. A partir de ese instante comenzará la lectura del tiempo de los demás canales alternando entre los pines 3 y 4 del mismo modo en el que se realizó la primera.

Estos tiempos son almacenados en variables, que serán las que se transmitirán mediante el bus I^2C. Por otra parte, mientras que se realiza la lectura hay un tiempo límite que, en caso de ser excedido, significará que ha ocurrido un error en la lectura. Si esto ocurre, se almacenará junto a las anteriores variables, otra variable booleana indicando esta situación y también será enviada junto con las otras al arduino controlador. Si se ha detectado un error, se desestiman las lecturas que aún no se han realizado y se vuelve a esperar una señal proveniente del pin 2.

En el tiempo que se tarda en observar los valores en un pin mediante una instrucción básica como '*digitalRead()*' puede ocurrir que se pierda parte o la totalidad de una señal de algún canal, produciendo como consecuencia un error en las posteriores lecturas. Es en este caso en el que entra a tomar parte la instrucción de acceso a los registros del microcontrolador '*bitRead()*', la cual tarda mucho menos tiempo.

Esta instrucción se encarga de acceder al registro del microcontrolador en el cual están ubicados los biestables que almacenan constantemente los valores recibidos por cada pin, de un conjunto de ocho pines. Acceder a este registro, el cual varía dependiendo el microcontrolador (por lo que hay que fijarse en el esquemático de éste), requiere un tiempo mucho menor que la instrucción básica '*digitalRead()*' puesto que ésta ejecuta unas 12 instrucciones más. En el caso del Arduino Nano Rev 3.0 con microcontrolador ATmega328, este registro se denomina '*PIND*'.

4.1.1 Modificaciones realizadas en el código

El código anterior es el encargado de leer los datos de la radio. Para enviar los datos por el bus I^2C de forma rápida y sin complicaciones se requiere de una modificación adicional que consuma el menor tiempo de proceso posible.

Hay que recalcar que la comunicación por buses se realiza a través de datos planos y que éstos se reciben como simples bytes, que deberán ser interpretados para obtener los valores correctos. Como todos los buses, el bus I^2C no se diferencia en este aspecto y los datos que se reciben de la radio se han de pasar y recibir como tal.

Es por ello que se ha diseñado una estructura '*union*' en el lenguaje *C++* que, sin recurrir a operaciones de conversión de datos, permite acceder a éstos de varias formas distintas, desde acceder como bytes a acceder como enteros, coma flotante, etc. Para explicarlo mejor se incluye un fragmento de código que contiene esta estructura:

```
//Estructura que contiene los 8 canales del receptor y el error
typedef struct paquete{
unsigned int canal1;
unsigned int canal2;
unsigned int canal3;
unsigned int canal4;
unsigned int canal5;
unsigned int canal6;
unsigned int canal7;
unsigned int canal8;
boolean radio_error;
}Paquete;

//Estructura para la facilitar la la conversion de tipos entre bytes, enteros y booleanos (2 bytes
<-> 1 int;
// 1 byte <-> 1 boolean)
typedef union {
Paquete canales;
byte arrayBytes[17];
```

Figura 20 – Estructuras para facilitar la conversión entre tipos de datos

Como se muestra en la "*Figura 17 - Estructuras para facilitar la conversión entre tipos de datos*" se han creado dos tipos de estructuras, una estructura normal que contiene un conjunto de 8 enteros sin signo y un booleano y una estructura de tipo '*union*' que contiene un tipo de estructura como la anterior y un *array* de 17 bytes.

Antes de crear una estructura como esta es necesario conocer el número de bytes que ocupa cada tipo de dato, en este caso los enteros sin signo y los booleanos. Para Arduino, cada entero sin signo ocupa dos bytes de memoria y cada booleano ocupa uno. Al tener 8 enteros sin signo y 1 booleano con el error de la radio, en total se ocuparán 17 bytes de memoria.

Con esto se puede crear una estructura '*union*' que englobe la estructura anterior y un *array* de 17 bytes de memoria. Esta estructura nos permitirá acceder a esos 17 bytes tratándolos como tales si los accedemos desde '*Radio.arrayBytes*' o como otro tipo definido si los accedemos desde '*Radio.canales*'. Así pues, si se desea acceder al valor de un canal tratándolo como un entero sin signo, se deberá utilizar '*Radio.canales.canalX*' (donde '*X*' es el número del canal) y si se desea acceder al error se utilizará '*Radio.canales.radio_error*'.

Cabe decir también que el orden de las variables en la estructura y el *array* de bytes se respeta, dando como resultado que el '*canal1*' se encuentre en los dos primeros bytes del *array* y el error en el último.

Una vez se tiene esta estructura, se programa el método '*requestEvent()*' que utiliza la librería '*wire.h*' de Arduino. En este método ("*Figura 18 - Evento de petición del arduino receptor*") se guardan las variables de los tiempos de cada canal en la estructura utilizando las variables de los tipo entero y booleano y se escriben en el bus accediendo a la estructura mediante el *array* de bytes.

```
void requestEvent(){
  estructura.paquetito.canal1 = TiempoMedioCanalesRadio [0];
  estructura.paquetito.canal2 = TiempoMedioCanalesRadio [1];
  estructura.paquetito.canal3 = TiempoMedioCanalesRadio [2];
  estructura.paquetito.canal4 = TiempoMedioCanalesRadio [3];
  estructura.paquetito.canal5 = TiempoMedioCanalesRadio [4];
  estructura.paquetito.canal6 = TiempoMedioCanalesRadio [5];
  estructura.paquetito.canal7 = TiempoMedioCanalesRadio [6];
  estructura.paquetito.canal8 = TiempoMedioCanalesRadio [7];
  estructura.paquetito.error = ErrorRadio;

  Wire.write(estructura.arrayBytes,17);
}
```

Figura 21 - Evento de petición del arduino receptor

4.2 Algoritmo de control PID

Antes de continuar con el código presente en el arduino de control se ha de explicar claramente en qué consiste un algoritmo de control Proporcional Integral Derivativo (PID). En este algoritmo se basará el control de la estabilidad del cuadricóptero por lo que en este apartado se detallará su funcionamiento y los efectos de aumentar o disminuir sus parámetros de control.

Así pues, un algoritmo de control PID es un algoritmo que, mediante realimentación, determina el error entre un valor obtenido y uno deseado, proporcionando una salida que trata de ajustar un sistema o proceso (del cual se mide ese valor obtenido) hasta obtener un error mínimo entre los dos valores. (21)(22)

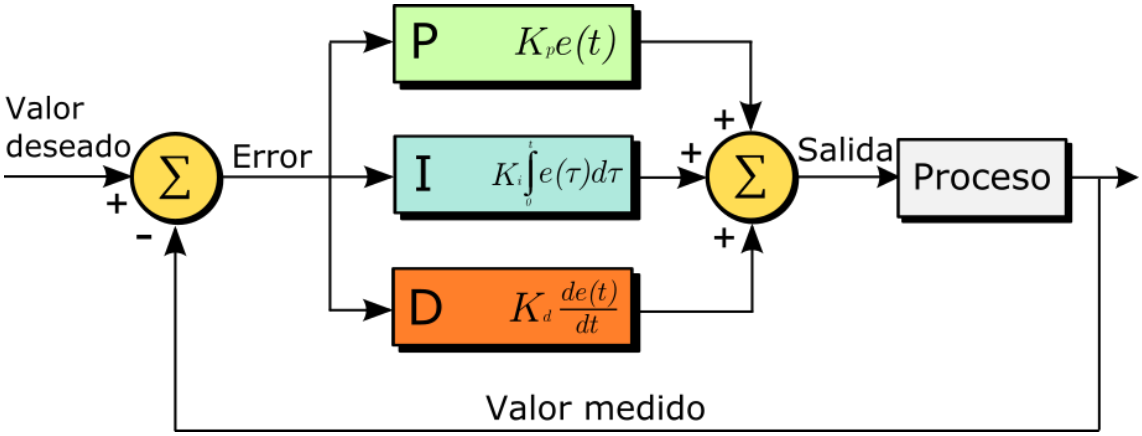

Figura 22 - Esquema de un algoritmo de control PID

Como se muestra en la "*Figura 22 - Esquema de un algoritmo de control PID*" la diferencia entre el valor deseado y el valor medido del proceso es el valor que utilizarán las tres partes del algoritmo (proporcional, integral y derivativa) para calcular su salida. Las tres salidas de los componentes se sumarán para proporcionar la salida final de control.

El peso que cada una de las partes ejerza sobre la suma del algoritmo viene dado por las constantes asociadas a éstos: *Kp*, *Ki* y *Kd*. Se ha de ajustar el valor de cada constante hasta encontrar los valores que proporcionen la respuesta deseada.

A continuación una breve explicación de cómo afecta cada constante a la respuesta del algoritmo. Se ha de tener en cuenta que esta respuesta es dependiente del sistema por lo que siempre habrá que realizar algunos ajustes.

4.2.1 Parte proporcional

La función de esta parte es bastante simple, consiste en la multiplicación del *error* por la constante proporcional *Kp*. Esto genera una respuesta que se irá acercando al valor deseado de forma oscilatoria.

Es decir, si entre el valor deseado y el medido hay una diferencia de 10 unidades y la constante proporcional *Kp* es 5, la salida generada tendrá un valor de 50. Es muy probable que tras varias iteraciones el nuevo *error* medido sea un valor negativo, por ejemplo -6, ya que el sistema tenderá a sobrepasar el valor deseado, de forma que la salida esta vez tendrá un valor de -30. Esto produce una respuesta oscilatoria que va disminuyendo el *error*.

Incrementar este valor producirá que la respuesta del sistema a los cambios sea mucho más rápida. Ahora bien, si el valor de *Kp* es demasiado alto, la salida generada podría oscilar constantemente y no disminuir el *error*, incluso podría llegar a producir una señal que oscilase sin control, aumentando cada vez más este *error*.

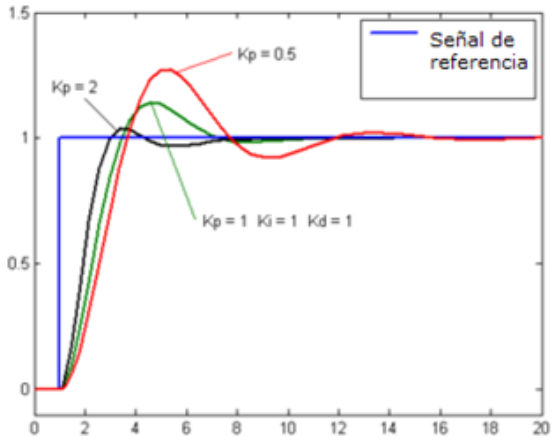

Figura 23 - Efectos de diferentes valores para la constante proporcional en un sistema dado

Como se observa en la "*Figura 23 - Efectos de diferentes valores para la constante proporcional en un sistema dado*", un valor pequeño en la constante proporcional (Kp = 0,5) producirá una respuesta más lenta que un valor más alto (*Kp* = 2).

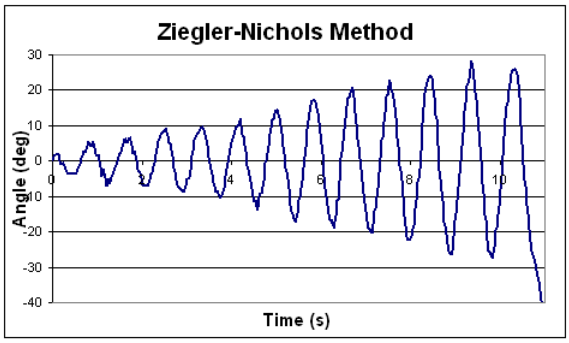

Figura 24 - Respuesta oscilatoria con una constante proporcional demasiado elevada

Pero en caso de aumentar la constante a un valor demasiado elevado se podría producir la respuesta de la "*Figura 24 - Respuesta oscilatoria con una constante proporcional demasiado elevada*" (*), una oscilación que no dejaría de aumentar.

(*) La respuesta de la "Figura 24 - Respuesta oscilatoria con una constante proporcional demasiado elevada" se ha sacado de una prueba del método Ziegler-Nichols, en este método se ha de buscar una constante Kp que produzca una respuesta oscilatoria lo más nivelada posible, de forma que con los parámetros integral y derivativos se alcance la respuesta total adecuada.

4.2.2 Parte integral

En esta parte, el *error* es integrado en el tiempo para tratar de disminuir la oscilación producida por la parte proporcional en el estado estacionario. La respuesta integral va a ir incrementándose en el tiempo si el *error* es distinto de cero. Esta respuesta es multiplicada por la constante *Ki* para hacerla más o menos significativa y luego se suma a la respuesta de la parte proporcional.

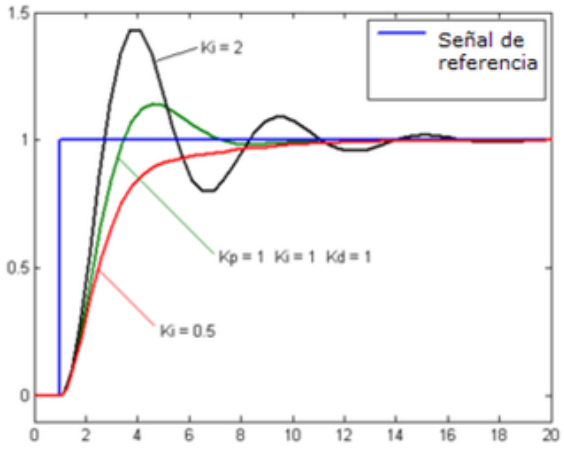

Figura 25 - Efectos de diferentes valores para la constante integral en un sistema dado

Conforme la constante integral vaya tendiendo al infinito, el efecto de la integración se irá perdiendo hasta ser nulo. Si la constante integral es muy pequeña comienzan a aparecer oscilaciones y el sistema puede llegar a ser inestable.

Una constante integral baja producirá una respuesta más lenta pero impedirá al sistema sobrepasar el valor deseado, eliminando los valores que generan el montículo inicial al solo utilizar el algoritmo proporcional. Una constante más alta hará que el proceso actúe con más rapidez pero con la consecuencia de una mayor oscilación antes de estabilizarse.

4.2.3 Parte derivativa

Derivando el *error* el algoritmo puede predecir la tendencia que tomará éste. La derivada actúa cuando el *error* está cambiando, mientras se encuentre en un estado estacionario no actuará. Es decir, la derivada trata de estabilizar el sistema después de que se haya producido el *error* disminuyendo las oscilaciones que pueda generar éste.

Añadiendo esta parte al algoritmo de control se disminuirá el valor de la salida cuando ésta tienda a aumentar y viceversa, de esta forma se suaviza el control ante los cambios que se produzcan en el sistema.

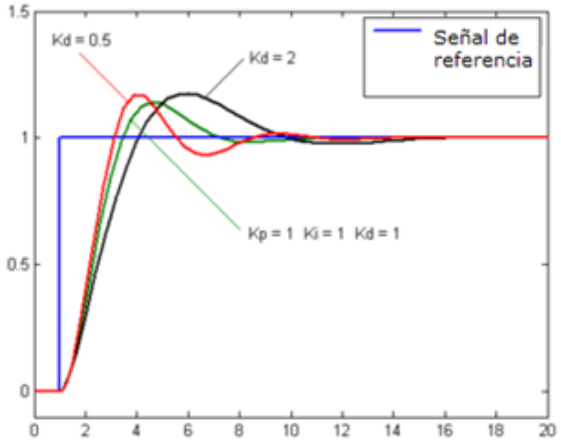

Figura 26 - Efectos de diferentes valores para la constante derivativa en un sistema dado

Aumentar la constante derivativa del sistema dará como consecuencia una acción derivativa mayor por lo que la curva de salida será más suave pero más tardará en estabilizarse. En cambio, con un valor pequeño de la constante se pierde la acción derivativa y el sistema tendrá mayores oscilaciones.

Como en todos las partes, lo ideal es encontrar un valor para la constante que corrija lo más rápido posible el *error* reduciendo las oscilaciones, aunque siempre dependiendo de la respuesta que se quiera obtener.

4.3 Control del cuadricóptero

En este apartado se describe el código que se introducirá en el arduino controlador, este código contiene las rutinas que se llevarán a cabo para la correcta estabilización del cuadricóptero. Se explicará la recopilación de los datos, su tratamiento con los algoritmos de control PID y cómo se genera la salida adecuada que recibirán los controladores de los motores.

4.3.1 Librerías, variables y constantes

A continuación se explicarán las librerías, las variables y las constantes que más influencia directa tendrán sobre el control del cuadricóptero. En este apartado se encuentran librerías, variables, constantes, estructuras y objetos, los cuáles son esenciales para el desempeño de la recepción y obtención de datos, la ejecución de los algoritmos de control y el envío de las señales propicias a los controladores de los motores.

Comenzando con las librerías, en el código actual se utilizan en total cuatro librerías para realizar la estabilización del aparato. Éstas son la librería '*Servo.h*' para el envío de señales PWM, la librería '*Wire.h*' para la implementación del bus I^2C (ambas de la API de Arduino), la librería '*Nuestro_PID.h*' la cual es una modificación de la librería '*AP_PID.h*' encontrada en el proyecto Copter de Ardupilot (23) y que se encarga de realizar el algoritmo PID, y la librería portada de InvenSense (7) para el funcionamiento de la MPU 9250.

Para una futura realización del control de altura y del aterrizaje automático se incluirían las librerías para la utilización del barómetro BMP 180 que tiene la MPU 9250 y se trabajaría con sensores de distancia como pueden ser láser o de ultrasonidos.

Continuando con las variables y constantes definidas, se han definido constantes para indicar y definir las posiciones de cada motor y el pin en el que se encuentran, así como para los canales de radio, los cuales llegarán en una estructura que se montará a partir de los datos recibidos por el bus I^2C de la forma que se explica en el apartado "*4.1.1 Modificaciones realizadas en el código*".

Por otro lado, se han definido también las constantes proporcionales, integrales y derivativas para los algoritmos PIDs encargados de controlar los ejes de cabeceo, alabeo y guiñada. También se ha creado un *array* con los objetos PIDs que llevarán a cabo la ejecución del algoritmo y las constantes que indican su posición en éste.

A su vez se han creado también variables, una por cada motor, para contener los objetos de la librería '*Servo.h*' que serán los que controlen el envío de señales PWM a los controladores electrónicos.

Los datos de la MPU 9250 se han guardado en variables para el control de errores, lecturas fallidas y éxitos así como para cada uno de los ejes de ésta, tanto para grados de inclinación como para velocidades de giro.

Se han definido constantes para la restricción de grados en cada eje del cuadricóptero para facilitar su control y evitar inclinaciones excesivas que posibiliten, con el añadido de alguna perturbación como una ráfaga de viento, que vuelque el aparato.

Por último existen una serie de constantes para el precompilador que le indican si ha de compilar los fragmentos de código de algunos componentes y las salidas de información por el puerto serial.

4.3.2 Setup: inicialización de las partes

En el método '*setup()*', el cual es el primero en ser ejecutado, se han inicializado todos los componentes necesarios para la correcta ejecución del código. En este apartado se ha inicializado la MPU 9250, se han colocado los parámetros correspondientes en cada PID y se ha dado un valor inicial a los controladores de los motores.

En el caso de los controladores de los motores, existen dos opciones viables para inicializarlos. La primera de ellas es haber programado con anterioridad las ESCs a los valores correspondientes en frecuencia PWM que es capaz de generar Arduino con la librería '*Servo.h*' y escribir un valor de cero para cada una.

La segunda es programar las ESCs cada vez que se inicializa el código, colocando el máximo valor PWM capaz de escribir la librería '*Servo.h*' y que vaya a ser utilizado. Esto producirá dos pitidos y acto seguido el usuario dispondrá de dos segundos para bajar la palanca de aceleración al mínimo, que el arduino detecte la acción y escriba un valor de cero por su salida PWM. Una vez realizado esto en el tiempo correcto se escuchará un pitido largo, después un pitido por cada celda de la batería conectada a los ESCs y por último una melodía de tres pitidos seguidos.

4.3.3 Loop: lectura de los sensores

Una vez realizado el método '*setup()*' es llamado a ejecución el método '*loop()*', el cual se ejecutará en bucle infinito hasta el reseteo del Arduino. En este bucle se tomarán los datos de los sensores, se leerán los canales de la radio y se procederá a ejecutar los algoritmos de control y generar la salida a los motores.

Al principio del método se obtendrán los datos de los sensores, esto conlleva la lectura de la MPU 9250 y, en caso de ampliar el proyecto, de cualquier sensor que se añada como podría ser el barómetro, sensor de ultrasonidos, temperatura, etc.

Cuando se obtienen los datos de la MPU 9250 mediante la librería de InvenSense se realiza un filtrado de éstos para que no se lean valores anómalos generados por ruido que puedan conllevar a un funcionamiento incorrecto del cuadricóptero.

Sobre la implementación con el acelerómetro ADXL345 y el giroscopio ITG3205, se obtenían los datos de ambos componentes y se realizaba un filtro de kalman para obtener los valores precisos. Al pasar a utilizar la MPU, este filtrado es facilitado por la propia librería que la controla.

Por otra parte, la lectura de los datos puede fallar y devolver valores erróneos en la comunicación por el bus I^2C. Es por ello que se ha de comprobar si la lectura ha sido exitosa antes de proceder a guardar los datos recabados. En este caso se lleva a cabo la cuenta de las lecturas exitosas obtenidas y se guardan los valores cada '*MAX_SUCCES_READS*' lecturas exitosas, que se decidió fueran 25 para el ámbito de pruebas.

4.3.4 Loop: lectura de los canales de radio

Una vez realizada la correcta lectura de los valores que registran los sensores se leen las órdenes que envía el piloto. Estas órdenes han sido codificadas en los ocho canales de la emisora y llegan al arduino de control a través del bus I^2C después de haber sido tratadas por el arduino receptor.

Para facilitar la conversión entre los tipos de datos que llegan desde el bus I^2C y los tipos que manejará el arduino de control se ha de recurrir a la misma estructura que se encuentra explicada en el apartado "*4.1.1 Modificaciones realizadas en el código*".

Una vez leídos los datos del bus I^2C y convertidos a los datos correspondientes mediante la estructura anterior, se ha de proceder a transformar el rango de los valores obtenidos en el rango apropiado que se utilizará para el control mediante PWM. Los rangos que se leen desde la emisora oscilan entre 1100 y 1900 aproximadamente y es necesario que estén representados por valores recluidos entre 0 y 180.

En realidad, para cada eje se ha definido una restricción en lo que se refiere al máximo valor que puede alcanzar, por lo que los valores sobre los que se transformarán las lecturas de radio serán los definidos para cada eje en las constantes *THROTTLE_RESTRICTION*, *YAW_RESTRICTION*, *PITCH_RESTRICTION* y *ROLL_RESTRICTION*.

Por otro lado, los valores enviados por radio pueden oscilar un poco, esto no tiene un impacto muy importante cuando se trata de valores que son distintos de cero, pero si se da el caso de tener la palanca del mando en esa posición, el arduino debe de interpretarla como un valor de cero. Para que esto se cumpla se deja un margen de valores, que es dependiente de cada eje, los cuales son sobrescritos por cero.

4.3.5 Loop: ejecución de los algoritmos de control PID

Cuando se han obtenido todos los datos necesarios para el control del cuadricóptero, se procede a éste mediante los algoritmos de control PID. Para realizar el control completo de todos los ejes, tanto para la estabilización como para las maniobras acrobáticas, se requiere de dos PIDs por cada eje. Uno se encargará de controlar la velocidad de giro en el eje (maniobras acrobáticas) y el otro de controlar la velocidad de giro que se desea en ese eje para que éste se mantenga estable (estabilización).

En el caso del eje de guiñada (*yaw*), no es muy normal tener un PID que se encargue de mantener una determinada posición, lo normal es que el piloto le dé una velocidad al eje y controle su posición por sí mismo. Así pues, el cuadricóptero poseerá un total de 5 PIDs, dos por cada eje de cabeceo y alabeo y uno por el eje de guiñada.

Comenzando con el PID que controla el eje de guiñada se conseguirá explicar mejor el funcionamiento conjunto de los dos PIDs que gobiernan cada uno de los ejes restantes. Tal y como se muestra en la "*Figura 27 - Algoritmo PID que controla la velocidad de giro en un eje*", se tiene un algoritmo PID que recibe la muestra de la velocidad de rotación del giroscopio de la MPU 9250 y también la velocidad de giro deseada. El algoritmo calcula la diferencia entre estas entradas y proporciona una salida, la cual se almacenará en una variable que posteriormente se utilizará para calcular la salida PWM total de cada motor.

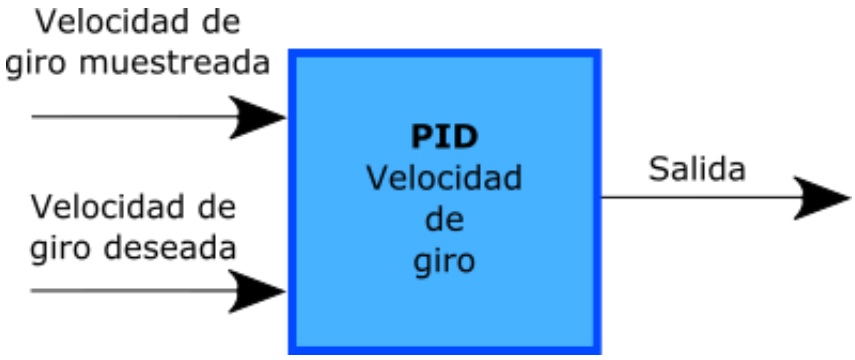

Figura 27 - Algoritmo PID que controla la velocidad de giro en un eje

El algoritmo anterior es un único PID simple que con el error entre dos variables genera una salida. Para los ejes de cabeceo y alabeo, el algoritmo que controla cada uno se vuelve más complejo, resultando en la interacción de dos PIDs. Uno de ellos se encargará de la velocidad de giro y el otro de enviar las señales a éste de forma que se consiga una estabilización.

En la "*Figura 28 - Algoritmos PID para controlar la estabilización de un eje*" se puede ver cómo la salida del PID que controla la estabilización hace de entrada para el PID que controla la velocidad de giro. De esta forma el PID que estabiliza enviará una velocidad de rotación al eje hasta que éste consiga la inclinación adecuada.

Estos dos PIDs se podrían ver como un sistema PID único, el hecho de estar separados en grados de inclinación y velocidad de giro posibilita el modo acrobático, en el que el PID de estabilización se desactivaría para dejar únicamente el PID de control de velocidad. Por otro lado, al tener dos PIDs se tiene un mejor control de cada medida y se puede ajustar cada PID por separado.

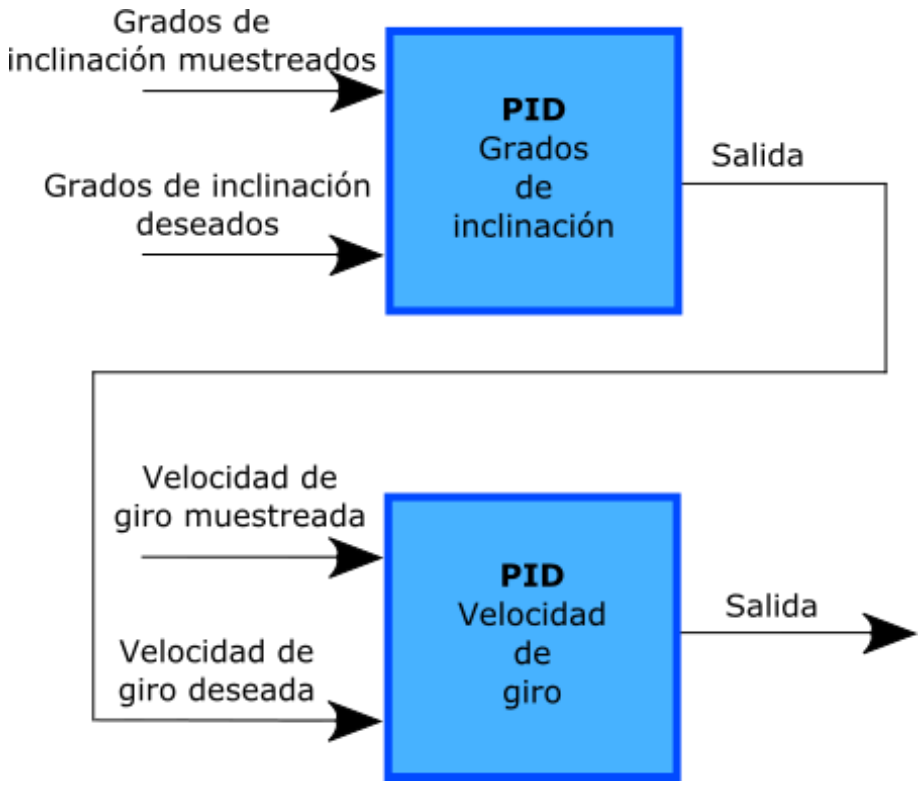

Figura 28 - Algoritmos PID para controlar la estabilización de un eje

Las salidas generadas por los PIDs de velocidad de rotación de cada uno de los ejes de cabeceo y alabeo se almacenarán, como con el del eje de guiñada, en variables que se utilizarán para proveer la potencia PWM a los controladores de los motores. En el siguiente apartado se verá cómo interaccionan entre ellas para conseguir el control correcto de cada motor.

4.3.6 Loop: envío de la señal PWM a los controladores

El último paso a realizar es proveer a cada motor de una señal PWM adecuada para que el control de la inclinación del cuadricóptero sea el correcto. En este apartado se explicará el funcionamiento de un algoritmo que une las señales de los PIDs de cada eje y la aceleración deseada, así como acotar éstas y transformar sus valores a los permitidos en la señal PWM que es capaz de enviar Arduino a cada controlador electrónico.

Si se tuviese una configuración en +, el control de los motores sería más sencillo pues habría únicamente que, como ejemplo en el caso del eje de alabeo, variar las velocidades de los dos motores que controlan el eje, en esta ocasión serían los motores izquierdo y derecho. Y así con el eje de cabeceo también.

Para el eje de guiñada, se habría de generar una diferencia entre los pares de motores que giran en el mismo sentido, si se potencia los motores que giran en el sentido de las agujas del reloj y se disminuyen los que giran en sentido contrario, el cuadricóptero tenderá a girar en el sentido contrario a las agujas del reloj y viceversa.

Ya que el cuadricóptero posee una configuración en ×, se habrá de variar las velocidades de los cuatro motores para la rotación de cada eje. En el caso del eje de guiñada, la configuración es la misma.

Así pues, para rotar por ejemplo el eje de alabeo, se deberá aumentar la velocidad de los motores delantero-derecho y trasero-derecho conjuntamente y disminuir la velocidad de los motores delantero-izquierdo y trasero-izquierdo. En el cabeceo los motores a variar su velocidad serán el par delantero y el par trasero.

De esta forma, cada motor soporta las variaciones de cada uno de los tres ejes por lo que las salidas de los PIDs de todos los ejes se deben unir en el algoritmo que determinará la velocidad final del motor.

Este algoritmo no es muy complicado pues resulta en el sumatorio de todas estas salidas, cambiando el signo de cada variable según le corresponda a cada motor. Si los motores delanteros han de sumar el valor de salida del eje de cabeceo, los motores traseros deberán restarlo. Si a su vez el motor delantero-izquierdo debe sumar el valor de salida del PID que controla el alabeo, el motor delantero-derecho deberá restarlo. Lo mismo ocurre en el caso de la guiñada.

Estos valores se le sumarán o restarán al valor de la aceleración total. Para que no exista una diferencia tan grande entre la potencia de un motor y el contrario de su mismo eje, la salida de los PIDs es dividida entre un valor determinado, en este caso se ha optado por dividirla a la mitad.

```
motorFL_val = map(constrain(rc_throttle_mapped - (roll_rate_output/2) -
(pitch_rate_output/2) + yaw_rate_output, 40, 220), 40, 220, 50, 180);

motorBL_val = map(constrain(rc_throttle_mapped - (roll_rate_output/2) +
(pitch_rate_output/2) - yaw_rate_output, 40, 220), 40, 220, 50, 180);

motorFR_val = map(constrain(rc_throttle_mapped + (roll_rate_output/2) -
(pitch_rate_output/2) - yaw_rate_output, 40, 220), 40, 220, 50, 180);

motorBR_val = map(constrain(rc_throttle_mapped + (roll_rate_output/2) +
(pitch_rate_output/2) + yaw_rate_output, 40, 220), 40, 220, 50, 180);
```

Figura 29 - Algoritmo para determinar la velocidad final de cada motor

En la "*Figura 29 - Algoritmo para determinar la velocidad final de cada motor*" se pueden apreciar las sumas y restas a la aceleración que se realizan para cada uno de los motores, entendiéndose '*motorFL_val*' como motor delantero-izquierdo, '*motorBL_val*' como motor trasero-izquierdo, '*motorFR_val*' como motor delantero-derecho y '*motorBR_val*' como motor trasero-derecho. El valor de la aceleración viene representado por '*rc_throttle_mapped*' y las salidas de cada PID para cada uno de los ejes como '*X_rate_output*' tomando *X* los valores '*pitch*', '*roll*' y '*yaw*', los cuales significan respectivamente cabeceo, alabeo y guiñada en inglés.

Una vez estos valores son sumados cada uno con su correspondiente signo, se acotan los valores de forma que el mínimo que puedan alcanzar sea de 40 y el máximo de 220. Esto se hace para evitar que los PIDs, aunque puedan, envíen una señal mayor o menor a estos valores de forma que la diferencia entre los motores no sea tan alta como para desestabilizar el control del aparato y no sea tan pequeña como para detener el motor.

Por otro lado, estos valores acotados, se transforman para que abarquen un rango de 50 a 180. El motivo de esto es no producir una señal mayor que la que utiliza la librería '*Servo.h*' de Arduino para enviar su valor más alto y, por otro lado, evitar que los motores se paren con un valor menor a 50. El hecho de que los motores dejen de rotar con un valor en el controlador electrónico de menos de 50 es precisamente porque el controlador detecta los valores menores a 50 como 0 y no envía señal a los motores. Esto equivale a que una señal PWM que envíe un valor menor del 27,8% de su total (50 = 27,8% de 180) a los controladores será tomada por un 0.

Una vez se han calculado los valores, solo resta transmitirlos a los objetos '*servo*' que representan cada motor y que se encargarán de reproducir un PWM adecuado que se envíe a los controladores electrónicos. Después de esto dará comienzo de nuevo la recopilación de los datos y los valores recibidos por radio y se continuará con el cálculo de los algoritmos de control.

5. Pruebas

En esta sección se detallarán todas las pruebas realizadas a lo largo del proyecto y los resultados obtenidos de cada una de ellas. Se explicarán las pruebas sobre cada uno de los componentes que conforman el cuadricóptero por separado, las pruebas que se llevaron a cabo en el ordenador y por último las pruebas sobre el sistema real.

5.1 Pruebas de los componentes por separado

Antes de construir y poner en ejecución el sistema en el entorno real se han realizado las pruebas pertinentes sobre cada uno de los componentes por separado para comprobar su correcto funcionamiento.

5.1.1 Prueba de microcontroladores

Aunque la plataforma Arduino ofrece productos de bastante calidad hay que tener en cuenta que se trata de hardware libre y en el caso de este proyecto, los Arduinos Nano utilizados no provienen de la casa original sino que pertenecen a fabricación externa.

Esto quiere decir que los componentes que se utilizan para la fabricación de los microcontroladores pueden ser diferentes y, aunque se ha de mantener el diseño original, haber sido modificados y ofrecer peores prestaciones.

En este proyecto se han utilizado Arduinos Nano provenientes del distribuidor chino Banggood.com (24), los cuales se pueden adquirir por un precio (menos de 3€) mucho menor que lo que costaría un Arduino Nano original (sobre 20€) o de otro fabricante (puede que incluso bastante más de 20€).

No es necesario recalcar que la diferencia de precio es bastante importante y que con un precio tan bajo es probable que los componentes sean de mala calidad. No obstante, la atención al cliente es buena y en caso de fallo se envía un nuevo componente al cliente.

Así pues, se comprobaba el correcto funcionamiento de cada Arduino antes de introducir el código del cuadricóptero y de conectarlo a los demás componentes. Si bien es cierto que la mayoría de Arduinos funcionaban correctamente, hubo cuatro Arduinos que fallaron después de un corto periodo de uso. Dos de ellos alcanzaron únicamente a una ejecución.

Para realizar la prueba de los Arduinos se introducía en ellos un programa de ejemplo básico que consisten en encender y apagar un led constantemente. Con este programa se comprobaba que la memoria del microcontrolador no estuviera corrupta y que el microcontrolador proveía el funcionamiento básico. Aparte de esta prueba básica se comprobaba con un medidor de tensión que los pines indicados generaban señal PWM.

Lo cierto es que exceptuando los cuatro primeros Arduinos defectuosos con los demás no hubo ningún problema y se trabajó muy bien con ellos. Los errores obtenidos fueron la falta

de alguna que otra resistencia o condensador que se había desprendido, error de comunicación desde el puerto USB y secciones de memoria corruptas.

5.1.2 Prueba de motores y ESCs

Una vez adquiridos los motores y los controladores electrónicos se comprobó su correcto funcionamiento, tanto a la hora de un uso normal como cuando debían ser programados.

Se llevaron a cabo pruebas conectando el receptor de radio directamente al controlador electrónico y desde una plataforma Arduino con modulación PWM. Ambas pruebas dieron buenos resultados tanto en la programación de las ESCs como en su uso normal de controlar la velocidad del motor.

Durante las pruebas se ajustó la dirección de rotación de cada motor y se le asignó su posición en la estructura del cuadricóptero. Estas comprobaciones se realizaron con cada una de las ESCs y motores por separado y luego, después de implementarlas sobre la estructura del aparato, se comprobó su funcionamiento en conjunto.

5.1.3 Pruebas de las IMUs

Para realizar las pruebas de las IMUs se diseñó un programa aparte que únicamente se dedicase a acceder mediante las librerías pertinentes a éstas, calculando el número de errores de lectura y comprobando que los ejes poseían el rango adecuado y su posición neutra devolvía un valor de cero o muy cercano a éste.

Ambas IMUs funcionaron correctamente en un primer momento. Con el acelerómetro ADXL345 de la primera se realizaron muchas pruebas que involucraban un filtro de kalman hasta que se consiguieron valores funcionales. Con la MPU 9250 las pruebas fueron bastante rápidas puesto que la librería de InvenSense funcionaba perfectamente.

Como se ha comentado anteriormente en la sección "*2.7 Unidad de Medida Inercial*" el uso de la MPU 9250 vino a sustituir el mal funcionamiento del acelerómetro ADXL345, el cual en un principio funcionaba correctamente pero, después de las primeras pruebas en el sistema real, su eje *Z* dejó de proveer valores correctos. Esto afectó al filtro de kalman a la hora de calcular las inclinaciones de los ejes *X* e *Y* puesto que para ello utiliza, además del giroscopio, el eje anterior.

5.2 Pruebas de los componentes por conjuntos

Cuando se comprobó el correcto funcionamiento de los componentes por separado se procedió a las pruebas en conjuntos. Es decir, se comprobaron las distintas partes del código que para funcionar requerían de varios elementos pero no de todo el sistema.

Así pues, se probó que las conexiones por I^2C entre los dos Arduinos y las conversiones entre tipos utilizando la estructura comentada en la sección "*4.1.1 Modificaciones realizadas en el código*" trabajasen adecuadamente.

Se comprobó la recepción de los datos utilizando el método de los diodos y las funciones que acceden a los registros del microcontrolador. Por otro lado, durante estas pruebas también se ajustaron algunos parámetros del mando hasta que la configuración fue lo más idónea posible para tratar las señales.

Se hicieron pruebas utilizando únicamente las IMU para comprobar el correcto funcionamiento de los controladores PID y se ajustaron los parámetros proporcionales, integrales y derivativos de forma que la salida producida fuese lo más acertada tratando de simular el sistema real.

5.2.1 Pruebas del conjunto total de componentes de control

Una vez comprobados que todas las funcionalidades del código proporcionaban buenos resultados, se hizo la prueba con la totalidad de los componentes exceptuando los controladores electrónicos y los motores.

```
-----------------------------------
RC_THROTTLE     RC_THRTL_MAPPED
1296.00 53
RC_PITCH        RC_ROLL RC_YAW
0.11    0.19    0.00
mpu_PITCH       mpu_ROLL        mpu_YAW
12.32   10.67   18.64
mpu_gyro_PITCH  mpu_gyro_ROLL   mpu_gyro_YAW
-1.83   -0.37   -0.31
PITCH_STAB_OUTPUT:      -30.00
ROLL_STAB_OUTPUT:       25.00
YAW_STAB_OUTPUT:        45.00
PITCH_RATE_OUTPUT:      -70
ROLL_RATE_OUTPUT:       62
YAW_RATE_OUTPUT:        0
X-MOTOR_FL:     57      X-MOTOR_FR:     119
X-MOTOR_BL:     -13     X-MOTOR_BR:     49

MOTOR_FL:       62      MOTOR_FR:       107
MOTOR_BL:       50      MOTOR-BR:       56
```

Figura 30 - Ejemplo de prueba sobre el ordenador con la MPU 9250 inclinada

En la "*Figura 30 - Ejemplo de prueba sobre el ordenador con la MPU 9250 inclinada*" se pueden ver los datos recibidos de la radio, los recabados de la MPU 9250, las salidas de cada uno de los algoritmos de control PID, los valores sin acotar ni transformar que se le enviarían a cada motor y por último los valores finales que sí que serían transmitidos.

Con estas pruebas se ajustaron los parámetros de los algoritmos de control hasta encontrar una configuración bastante apropiada. Una vez obtenida esta configuración se probaría sobre el sistema real y se realizarían los últimos ajustes para su estabilización.

5.3 *Pruebas sobre el sistema real*

Habiendo realizado las pruebas de todos los componentes y comprobado su correcto funcionamiento se procedió a la construcción del sistema completo y de la plataforma de pruebas.

Para las pruebas se construyeron dos tipos de plataformas, una para las primeras pruebas que se hicieron con el sistema y otra, mejor que la anterior, que se utilizó en las pruebas siguientes.

5.3.1 Primeras pruebas y fallos en los componentes

En las primeras pruebas se comprobó que todo el sistema funcionaba correctamente y se procedió al ajuste de los parámetros de los algoritmos de control PID. Durante los primeros ajustes el sistema creó una buena respuesta oscilatoria. Éste es el primer paso para conseguir el ajuste completo modificando los parámetros integral y derivativo.

Las pruebas se realizaban sobre una plataforma que permitía al cuadricóptero girar sobre un solo eje ("*Figura 31 - Estructura inicial de pruebas*"). De esta forma se podía observar cómo trabajaba cada PID por separado y se podrían ajustar uno por uno antes de proceder a ajustarlos en conjunto.

Figura 31 - Estructura inicial de pruebas

Aun habiendo comprobado el correcto funcionamiento de todas las partes nada pudo evitar que algunas de ellas fallasen mientras se realizaban los ajustes del parámetro integral, impidiendo continuar con las pruebas.

En estas pruebas fallaron varios de los componentes que en un principio debían de funcionar correctamente. Primeramente uno de los Arduino Nano, el que realizaba el control, produjo un error de memoria corrupta, este fallo era fácilmente solucionable utilizando otro de los Arduinos de repuesto. Aunque no se esperaba el fallo en este componente no era poco probable que sucediera.

Otro de los componentes que falló y que nos impidió seguir trabajando fue el receptor de radio. El fallo sucedió cuando comprobamos el funcionamiento de las ESCs conectando directamente el receptor de radio en una de éstas. El motivo por el cual sucedió no está del todo claro pero el resultado fue un aumento de la tensión permitida por el receptor que quemó el chip regulador de tensión 6206A.

Las ESCs disponen de un BEQ que se encarga de regular las tensiones que envía por su pin de voltaje a los receptores de radio que se conecten. Este BEQ debía enviar una tensión adecuada al receptor de 5V, lo cual no sucedió. Dado que el chip 6206A transforma tensiones de 9V a 3.3V y la batería que provee el voltaje a las ESCs tiene una tensión de 11.1 V, es presumible que ocurriera por un fallo en el BEQ de la ESC.

Después de este incidente se decidió no hacer pruebas de las ESCs sobre el receptor y utilizar un Arduino para ello, ya que el Arduino no puede recibir la tensión de alimentación por parte de uno de los BEQ, los cuales deben funcionar a 5V.

El mayor problema que produjo el fallo en el receptor fue un completo mes sin poder hacer pruebas por culpa de los largos periodos que tarda en llegar un componente desde la página.

Se podría decir que haber previsto el fallo de este componente con antelación habría solventado muchos de los problemas pero se tienen que tener varias cosas en cuenta antes de decidir tener componentes de recambio en un trabajo de este tipo.

Primeramente se trata de un componente que se supone es robusto (no sucede lo mismo con un Arduino Nano), el cual no debe de fallar puesto que ha sido diseñado para ese tipo de trabajo. Al referirme a que es un componente robusto me refiero no solo al receptor sino al BEQ de la ESC también, dado que no está claro cuál fue el fallo. Ambos dos componentes, el receptor y la ESC, están preparados para esa función y deben producir resultados correctos.

Está claro que cualquier componente por muy robusto que sea está sujeto a fallos y siempre se debe de disponer de al menos un componente de repuesto para estos casos. Pero en el ámbito de este proyecto se ha de tener en cuenta un factor muy importante que es el dinero.

Un receptor o una ESC son componentes caros de los que no se puede pedir una gran cantidad como repuesto. Por otro lado, se espera que el componente funcione bien por lo que si se piden varios repuestos es porque no se está esperando un buen funcionamiento. Si no se espera un buen funcionamiento y la tasa de fallo de los componentes es alta no merece la pena pedir repuestos si éstos van a fallar también. Y, en caso de no fallar, se ha desperdiciado bastante dinero en la compra de los recambios.

5.3.2 Siguientes pruebas y nuevos fallos en los componentes

Después de las primeras pruebas en las cuáles fallaron varios de los componentes, se volvieron a realizar pruebas sobre el sistema en el ordenador y sobre cada componente por separado para comprobar que las partes funcionaban correctamente.

Fue en estas pruebas en las que se comprobó que el eje *Z* del acelerómetro ADXL345 tampoco funcionaba correctamente por lo que se decidió comprar dos MPU 9250 de distintos fabricantes.

Una vez se hubieron repetido todos los pasos anteriores y se hubo comprobado que los componentes funcionaban correctamente, se procedió de nuevo a la implementación sobre el sistema real.

Para estas pruebas se utilizó la nueva plataforma y los primeros resultados fueron satisfactorios, el PID proporcionaba una respuesta oscilatoria como en las primeras pruebas que se realizaron por lo que se podría continuar con los ajustes.

Figura 32 - Plataforma final de pruebas

Después de varias pruebas con la nueva plataforma ("*Figura 32 - Plataforma final de pruebas*"), tal y como sucedió en las primeras pruebas, varios de los componentes empezaron a fallar después de un uso continuado.

En esta ocasión dos de los motores (delantero-derecho y trasero-izquierdo) dejaron de rotar y se quedaron en un estado en el que parecían no conseguían arrancar, haciendo temblar las hélices.

Todas las ESCs habían sido programadas sin problemas y su rango se encontraba en el adecuado. Se volvieron a reprogramar individualmente cada una comprobando también su arranque normal. De esta forma se observó que, efectivamente, todas se programaban con el rango deseado, pero por lo visto dos de ellas no conseguían mandar correctamente los pulsos a los motores o por algún motivo el motor estaba estropeado.

Para comprobar si era cuestión del arranque se le dio a los motores un impulso inicial y efectivamente el motor comenzó a rotar sin problemas, la velocidad no aumentaba pero tampoco disminuía. Para comprobar si se podía aumentar o disminuir la velocidad del motor se varió el pulso que recibía la ESC.

Esto produjo un repentino aumento de la velocidad tal que, con la propia fuerza del motor y el hecho de que éste estaba sujeto a la plataforma de pruebas y no podía desplazarse más allá de cierto ángulo, rompió las palas de la hélice y éstas se desprendieron de tal forma que partieron otra de las palas de una de las hélices adyacentes.

Estos problemas impidieron que el trabajo pudiera continuar por falta de tiempo y dejaron clara la mala calidad de los componentes utilizados. Al igual que el receptor o los ESCs, los motores se suponen unos componentes aún más simples y robustos si cabe y, a su vez, más caros.

Así pues, la estabilización del cuadricóptero no se alcanzó y se detuvo en la fase de ajuste de los algoritmos de control PID. La respuesta de los algoritmos en las pruebas dio buenos resultados pero se han de continuar regulando los parámetros hasta alcanzar la estabilización. El estado en el cual se halla el cuadricóptero en estos momentos es el de continua oscilación.

6. Conclusiones

1. Se han cumplido todos los objetivos del proyecto exceptuando la consecución del ajuste del sistema de estabilización para lograr la misma sobre el cuadricóptero.

2. Los problemas surgidos con los componentes y el tiempo que conlleva la realización de las pruebas, suponen un gran obstáculo a superar para conseguir el ajuste que permita la estabilización del sistema, y es uno de los asuntos más importantes que se deben plantear.

3. Se ha demostrado que no supone un gran problema la implementación de un sistema de control (así como de lectura de radio) sobre microcontroladores de la plataforma Arduino de menores prestaciones, partiendo únicamente de la utilización de algunas librerías y evitando el uso de microcontroladores diseñados para tal propósito (*Ardupilot* (23)).

7. *Trabajos futuros*

Las posibilidades de ampliación de este trabajo son bastante extensas. Tal y como se comentaba en el apartado "*1.4 Estado del arte*", los procedimientos que utilizan las empresas *LaTrax* y *Horizon Hobby* serían una gran línea de investigación que aplicar a este proyecto, una ampliación que probablemente requiriera un cambio en los componentes del control para mejorar su capacidad de memoria y procesamiento, así como la incorporación de nuevos componentes.

La ampliación que más posibilidades tiene de ser implantada sobre este proyecto una vez se consiga la estabilización sería un modo de control de altura o '*hover*' en inglés. Para este modo se utilizaría el barómetro BMP 180 del fabricante Bosch que viene incorporado en la MPU 9250 y se debería añadir un sensor de distancia para controlar alturas cercanas al suelo.

El código pensado consistiría básicamente en un algoritmo de control PID que se encargase de mantener una altura adecuada, implementando un filtro para las lecturas de presión del barómetro evitando ruidos, preferiblemente un filtro de kalman.

Otra de las ampliaciones que le sigue en posibilidades a la anterior es la implementación de un programa de aterrizaje automático simple. Para esta mejora se necesitaría únicamente de los elementos anteriores.

El programa se encargaría de ir ajustando la altura del cuadricóptero utilizando el PID del sistema de control de altura y, en el momento detectase la presencia de terreno con un sensor de distancia (preferiblemente varios para detectar los puntos cruciales donde se debería de posar), procedería a un control mucho más eficiente que le permitiese un aterrizaje suave.

Otra posible ampliación sería la inserción de una cámara para tomar fotografías o vídeos aéreos y, en caso de continuar la ampliación, la implementación de un sistema de visión en primera persona en tiempo real (*FPV – First Person View* en inglés).

Para este sistema habría de incluir elementos mucho más caros, cambiar el sistema de radio o añadir otro sistema aparte, e incluir un nuevo microcontrolador que se encargase de la interacción con la cámara. También se debería incluir un dispositivo de memoria donde almacenar las fotografías o vídeos. Un sistema '*gimbal*' que se encargase de estabilizar el ángulo de visión de la cámara también sería una característica a tener en cuenta.

Por otro lado, la incorporación de un sistema GPS mejoraría muchos de los aspectos del cuadricóptero, permitiéndole realizar un seguimiento de rutas u objetos. Esta ampliación requeriría la creación de un software para facilitar la introducción de rutas o de un elemento GPS externo que indicase la posición de un objeto al cual el cuadricóptero debiera seguir.

Existen muchas ampliaciones posibles para un sistema de este tipo y muestra de ello es el gran auge que, se comentaba al inicio del proyecto, estaban teniendo estos vehículos aéreos.

Muchas empresas trabajan en el desarrollo de estos aparatos desde un punto de vista industrial, económico, de ocio o militar por la gran variedad de opciones que ofrecen para muchos campos, lo que previsiblemente conducirá a una serie de ampliaciones y mejoras que aún no se conciben.

8. Bibliografía

1. Cuál es la diferencia entre un drone, un UAV y un RPA [Internet]. Onemagazine. [citado 20 de julio de 2015]. Recuperado a partir de: http://www.onemagazine.es/noticia/14713/Industria/Cual-es-la-diferencia-entre-un-drone-un-UAV-y-un-RPA.html

2. dron, adaptación al español de drone [Internet]. [citado 20 de julio de 2015]. Recuperado a partir de: http://www.fundeu.es/recomendacion/dron-adpatacion-al-espanol-de-drone/

3. admin. About LaTrax [Internet]. 2014 [citado 20 de julio de 2015]. Recuperado a partir de: https://latrax.com/about

4. admin. Alias: Quad Rotor Helicopter, Ready-To-Fly with 2.4GHz radio system, 650mAh LiPo battery, and single USB-powered charger. [Internet]. 2014 [citado 20 de julio de 2015]. Recuperado a partir de: https://latrax.com/products/alias

5. Home - Horizon Hobby LLC [Internet]. [citado 15 de julio de 2015]. Recuperado a partir de: http://www.horizonhobbycorp.com/

6. SAFE [Internet]. [citado 15 de julio de 2015]. Recuperado a partir de: http://www.flysaferc.com/

7. MPU6050/MPU6500/MPU9150/MPU9250 library over I2c [Internet]. [citado 27 de agosto de 2015]. Recuperado a partir de: http://forum.arduino.cc/index.php?topic=242335.0

8. rpicopter/ArduinoMotionSensorExample · GitHub [Internet]. [citado 27 de agosto de 2015]. Recuperado a partir de: https://github.com/rpicopter/ArduinoMotionSensorExample

8. Arenas Mas, Marta. Sistema para la adquisición y monitorización de aceleraciones mediante microprocesador. Capítulo 4. [Internet] Biblioteca de Ingeniería, Universidad de Sevilla [citado 2 de agosto de 2015]. Recuperado a partir de: http://bibing.us.es/proyectos/abreproy/11638/fichero/Capitulo+4.pdf\line

9. García García, Sergio. DISEÑO Y CONSTRUCCIÓN DE MAGNETÓMETRO TRIAXIAL PARA ANÁLISIS Y EXPERIMENTACIÓN DE AISLAMIENTOS MAGNÉTICOS [Internet]. Departamento de Ingeniería Mecánica, Escuela Politécnica Superior de la Universidad Carlos III de Madrid, Leganés [citado 8 de agosto de 2015]. Recuperado a partir de: http://webcache.googleusercontent.com/search?q=cache:2fq6Fo_Ek0cJ:e-archivo.uc3m.es/bitstream/handle/10016/17038/pfc_sergio_garcia_garcia_2013.pdf%3Fsequence%3D1+&cd=3&hl=es&ct=clnk&gl=es

11. admin. ¿Cómo funciona un transmisor de presión? [Internet]. bloginstrumentacion.com. [citado 13 de agosto de 2015]. Recuperado a partir de: http://www.bloginstrumentacion.com/blog/2010/06/28/como-funciona-un-transmisor-de-presion/

12. BMP180 [Internet]. [citado 13 de agosto de 2015]. Recuperado a partir de: http://www.bosch-sensortec.com/en/homepage/products_3/environmental_sensors_1/bmp180_1/bmp180

13. Microcontrolador [Internet]. Wikipedia, la enciclopedia libre. 2015 [citado 8 de agosto de 2015]. Recuperado a partir de: https://es.wikipedia.org/w/index.php?title=Microcontrolador&oldid=84079070

11. Arduino - Introduction [Internet]. [citado 8 de agosto de 2015]. Recuperado a partir de: https://www.arduino.cc/en/Guide/Introduction

15. Arduino - Products [Internet]. [citado 9 de agosto de 2015]. Recuperado a partir de: https://www.arduino.cc/en/Main/Products

16. Arduino - ArduinoBoardNano [Internet]. [citado 9 de agosto de 2015]. Recuperado a partir de: https://www.arduino.cc/en/Main/ArduinoBoardNano

17. Fritzing [Internet]. [citado 27 de agosto de 2015]. Recuperado a partir de: http://fritzing.org/

18. Cheyenne. Cuadricoptero DIY con Arduino: Leer emisora RC Parte 2 [Internet]. Cuadricoptero DIY con Arduino. 2012 [citado 27 de agosto de 2015]. Recuperado a partir de: http://cuadricopterodiyarduino.blogspot.com.es/2012/07/leer-emisora-rc-parte-2.html

19. 5 Pcs 3.3V 5V TTL Bi-directional Logic Level Converter For Arduino Sale-Banggood.com [Internet]. [citado 27 de agosto de 2015]. Recuperado a partir de: http://www.banggood.com/5-Pcs-3_3V-5V-TTL-Bi-directional-Logic-Level-Converter-For-Arduino-p-951182.html

20. AN10441.pdf [Internet]. [citado 27 de agosto de 2015]. Recuperado a partir de: http://www.nxp.com/documents/application_note/AN10441.pdf

21. PID Theory Explained - National Instruments [Internet]. [citado 29 de agosto de 2015]. Recuperado a partir de: http://www.ni.com/white-paper/3782/en/#toc1

22. PID controller [Internet]. Wikipedia, the free encyclopedia. 2015 [citado 30 de agosto de 2015]. Recuperado a partir de: https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=678518249

23. Copter | UAV Multirrotor [Internet]. [citado 30 de agosto de 2015]. Recuperado a partir de: http://copter.ardupilot.com/

24. Compras en línea de dispostivos, Helicóptero RC y Quadcopter, Teléfonos Móviles, Moda en Banggood.com [Internet]. [citado 31 de agosto de 2015]. Recuperado a partir de: http://www.banggood.com/es/