UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FIN DE GRADO

Grado en Ingeniería Electrónica, Industrial y Automática

DISEÑO Y MONTAJE DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA UN CUADRICÓPTERO

Memoria Técnica

Autor: Eric Vergara Samsó
Director: Manuel Andrés Manzanares Brotons

Convocatoria: Junio 2020

Resum

Aquest document té com a objectiu el disseny, tant mecànic com electrònic, d'un dron com a plataforma d'adquisició de dades de forma remota. Per tal d'aconseguir aquest objectiu s'ha investigat les diferents opcions a l'hora de muntar un dron, així com els components amb una millor relació qualitat/preu, com la interconnexió entre els diferents elements del vehicle i la seva corresponent programació. A més a més, s'utilitzarà el quadrotor com a una plataforma d'adquisició de dades. Els paràmetres que es rebin dels diferents sensors s'enviaran per *Bluetooth* a un telèfon personal, on es podran visualitzar les dades obtingudes a partir d'una aplicació mòbil.

Així doncs, durant el desenvolupament d'aquest document es tractarà d'exposar tot el que envolta als diferents aspectes d'aquests vehicles no tripulats i es tractarà d'explicar amb claredat i pas a pas la creació d'un dron com a plataforma d'adquisició de dades tenint en compte el temps que s'ha d'invertir per a poder-lo realitzar de forma òptima i posant en pràctica els coneixements que s'han adquirit al llarg de tota la carrera universitària.

Resumen

Este documento tiene como objetivo el diseño, tanto mecánico como electrónico, de un dron como plataforma de adquisición de datos de forma remota. Para conseguir este objetivo se ha investigado las diferentes opciones a la hora de montar un dron, así como los componentes con una mejor relación calidad/precio, como la interconexión entre los diferentes elementos del vehículo y su correspondiente programación. Además, se utilizará el cuadricóptero como una plataforma de adquisición de datos. Los parámetros que se reciban de los diferentes sensores se enviarán por *Bluetooth* a un teléfono personal, donde se podrán visualizar los datos obtenidos a partir de una aplicación móvil.

Así pues, durante el desarrollo de este documento se tratará de exponer todo lo que rodea a los diferentes aspectos de estos vehículos no tripulados y se tratará de explicar con claridad y paso a paso la creación de un dron como una plataforma de adquisición de datos teniendo en cuenta el tiempo que se debe invertir para poderlo realizar de forma óptima y poniendo en práctica los conocimientos que se han adquirido a lo largo de toda la carrera universitaria.

Abstract

This document aims to design, both mechanically and electronically, a drone as a remote data acquisition platform. To achieve this objective, the different options when mounting a drone have been investigated, as well as the components with a better quality / price ratio, such as the interconnection between the different elements of the vehicle and their corresponding programming.
In addition, the quadcopter will be used as a data acquisition platform. The parameters received from the different sensors will be sent by Bluetooth to a personal phone, where it can be viewed the data obtained from a mobile application.

Thus, during the development of this document it will try to expose everything that surrounds the different aspects of these unmanned vehicles and it will try to explain clearly and step by step the creation of a drone as a data acquisition platform taking into account the time that has to be inverted to be able to realize it optimally and putting in practice the knowledge that has been acquired along all the university career.

Agradecimientos

Quisiera hacer una especial mención a las personas que me han ayudado de una forma u otra a realizar este proyecto.

A mi tutor del proyecto Manuel Andrés, que me ha orientado a la hora de encarar el proyecto y ha sido vital en el desarrollo del mismo gracias a sus consejos y apoyo.

Finalmente, a mis padres y a mi pareja, que no solo han sido imprescindibles en la realización de este trabajo si no que sin su apoyo y ayuda no hubiera sido capaz de concluir mi etapa universitaria.

Glosario

Abreviatura	Significado
ESC	*Electronic Speed Controller*
IMU	*Inertial Motion Unit*
CPU	*Central Processing Unit*
LCD	*Liquid Crystal Display*
LED	*Light-Emitting Diode*
PWM	*Pulse-Width Modulation*
I2C	*Inter-Integrated Circuit*
SCL	*System Clock*
SDA	*System Data*
PDB	Placa de Distribución de Potencia
USB	*Universal Serial Bus*
PID	Controlador Proporcional, Integral y Derivativo
BOE	Boletín Oficial del Estado
AESA	Agencia Estatal de Seguridad Aérea
RoHS	*Restriction of Hazardous Substances*
UE	Unión Europea

Índice

1. Prefacio

1.1. Origen del trabajo

Se parte de la idea de diseñar un dron primeramente por la popularidad de estos equipos en la actualidad, además de que supone un reto para demostrar los conocimientos adquiridos durante la carrera en un trabajo de fin de grado, puesto que, desde la selección de componentes hasta la programación del mismo se ponen en práctica mucho de lo que se ha aprendido a lo largo de la carrera. Además, se intentará expandir dichos conocimientos con la creación de una plataforma de adquisición de datos.

1.2. Motivación

El trabajo surge de la necesidad de crear un proyecto desde cero y de demostrar que este proyecto solo es posible realizarlo si se tienen los conocimientos adquiridos a lo largo de la carrera de Ingeniera Electrónica, Industrial y Automática y de si se es capaz de resolver los problemas que surjan durante el transcurso del proyecto de forma solvente.

1.3. Requerimientos previos

Los requerimientos previos necesarios para realizar satisfactoriamente el proyecto son los conocimientos adquiridos durante la carrera de Ingeniera Electrónica, Industrial y Automática, ya que se necesita la capacidad de seleccionar componentes económicos y compatibles entre ellos, la autosuficiencia de realizar un esquema eléctrico con la interconexión entre sus componentes que cumpla de lo que se precise y la capacidad de realizar un programa de Arduino con su PID de control lo suficientemente rápido para que corrija los movimientos del dron durante el vuelo, entre otros conocimientos previos.

2. Introducción

Antes de empezar a explicar la parte técnica de la solución se necesita saber reconocer qué es realmente un dron, de que partes se compone, sus funciones o aplicaciones y su funcionamiento.

Con lo que respecta sobre que es un dron se trata de un vehículo aéreo que vuela sin tripulación que funcionan gracias a unas hélices o rotores que les permite propulsarse y mantenerse en el aire, cabe destacar además que, a pesar de su popularidad actualmente, el dron lleva existiendo desde hace mucho tiempo pese a su fabricación, algo costosa.

Los drones pueden ser utilizados en infinidad de tareas, a continuación, se listan algunas de ellas:

- En eventos donde se congregan multitudes de personas pudiendo así obtener los mejores planos y ángulos en fotografías.
- En situaciones de emergencia pueden ser empleados para acceder a áreas de difícil acceso debido a su versatilidad.
- Para la búsqueda de personas mediante una cámara de alta definición que transmite la información en tiempo real.

En cuanto al uso de éstos en el futuro, Amazon una gran empresa dedicada al comercio electrónico está barajando la posibilidad de utilizar a éstos para la entrega de paquetes, otro ejemplo sería utilizarlos como satélites para crear redes de internet donde éste no alcanza, funcionando con energía solar.

Aunque la mayoría de los cuadricópteros, como bien indica el nombre, se componen de cuatro hélices también hay de seis y hasta ocho hélices. Para que el dron se mantenga estable dos de éstas han de girar en un sentido y las otras dos en el otro. Los drones pueden realizar diferentes movimientos (en los que se profundizará en el siguiente apartado) controlando la propulsión de cada hélice, el componente que se dedica a la gestión de éstas es el llamado controlador de vuelo, que en éste caso será el Arduino UNO.

Este controlador también es capaz de transmitir información a los ESC (*Electronic Speed Controller*) que es el que componente que hace que cada motor gire con unas revoluciones determinadas.

Para alimentar todo el circuito se suele utilizar una batería LiPo, ya que son fáciles de recargar, pesan poco y son las que determinarán la autonomía del dron.

Finalmente, para controlar el dron a distancia se suele utilizar un mando de radio o, actualmente, incluso se puede utilizar el móvil.

2.1. Objetivos del trabajo

La razón por la que se realiza este proyecto es por la ambición de crear algo desde cero para poder demostrar que durante la carrera se ha adquirido una serie de conocimientos que antes no se poseían.

Cabe destacar además que la realización de este proyecto es ampliable, es decir, una vez acabado el proyecto de fin de grado también se puede profundizar e incluir más mejoras en el proyecto, lo que lo hace ideal para probar y experimentar los conocimientos de electrónica que se adquieran en un futuro.

Los objetivos del proyecto se han dividido en pequeños retos a superar para conseguir una sensación de progresión que se puede visualizar en el Diagrama de Gantt sobre el que más adelante se profundizará.

Estos objetivos son, por ejemplo:

- Elaborar un diagrama en donde se distribuya el tiempo que se dedica al proyecto en cada una de las tareas.
- Seleccionar componentes factibles a nivel de precio y que cumplan los requisitos para que el proyecto se lleve a cabo de forma satisfactoria.
- Diseñar un esquema eléctrico desde cero.
- Distribuir y fijar los componentes de una manera óptima.
- Programar la CPU de forma que realice lo que el programador precise.
- Diseñar una aplicación móvil que reciba datos del controlador.
- Resolver los problemas que surjan durante el desarrollo del proyecto.

2.2. Alcance del proyecto

Se fija la meta de realizar un cuadricóptero desde cero con todas las partes que eso conlleva, desde decidir los componentes hasta la programación del conjunto.

Es decir, al finalizar el proyecto, se ha de obtener un dron que sea estable y que responda bien a las consignas del mando. También se plantea la opción de no poner hélices hasta la finalización y entrega del proyecto, ya que en las primeras pruebas quizá el dron no funciona como es debido y podría chocar, lo que provocaría la pérdida total de todo el *hardware*. La estabilidad del vehículo se demostrará mediante gráficas en tiempo real que mostrarán que reacciona como debería a las consignas recibidas.

2.3. Historia

Para ponerse en contexto, la idea del dron no es nueva, ya que en el año 1849, el ejército austríaco montó alrededor de doscientos globos aerostáticos no tripulados con el objetivo de dejar caer bombas explosivas en la ciudad de Venecia en el que se conoce como el primer bombardeo aéreo de la historia.

Posteriormente, en 1896 Samuel P. Langley desarrolló una serie de aeronaves a vapor sin piloto que, posteriormente serían utilizadas para realizar la primera fotografía de reconocimiento aéreo de la historia en 1898.

Otros momentos clave en la historia de los cuadricópteros fueron de carácter bélico, concretamente entre el principio de la Primera Guerra Mundial y la actualidad donde realizaron diversas funciones:

- A principios del siglo XX fueron utilizados como blanco de prácticas para fuerzas militares.
- Durante la Segunda Guerra Mundial fueron diseñados para ser una especie de bomba volante.
- Durante la Guerra Fría y actualmente en la lucha contra el terrorismo se usan como una plataforma de vigilancia. (1)

2.4. Clasificación

Existen diferentes tipos de drones que convergen en otros tipos, una forma más sencilla de apreciarlo es observando el siguiente esquema:

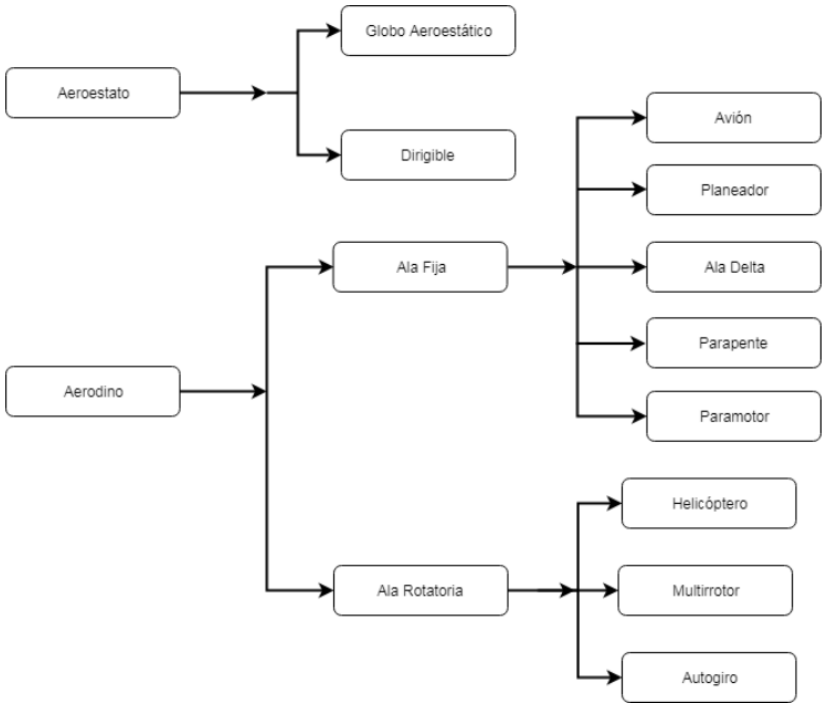

Figura 2.1. Esquemático con la clasificación de los distintos tipos de drones (Fuente: (2))

En él se aprecia que los dos grandes grupos son:

- Aerostato: Aeronaves cuya suspensión en el aire se debe al empleo de un gas más ligero que el aire.
- Aerodino: Aeronaves más pesadas que el propio aire.

Los aerodinos se dividen en los siguientes grupos:

- Drones de ala fija: Son aquellas aeronaves cuyas alas están pegadas con el resto de elementos y no poseen movimiento propio.
    - Ala alta: Drones con el ala ubicada en la parte superior del fuselaje, lo que provoca una mayor estabilidad per una más compleja maniobrabilidad.
    - Ala media: Drones con el ala ubicada en la parte media del fuselaje, lo que aporta un equilibrio entre maniobrabilidad y estabilidad
    - Ala baja: Drones con el ala ubicada en la parte inferior del fuselaje, lo que los hace de fácil maniobrabilidad.
    - Ala volante: Drones en los que el ala conforma la mayor parte del fuselaje, lo que provoca una baja resistencia aerodinámica y una elevada maniobrabilidad.
- Drones de ala rotatoria: Drones en los que las alas giran alrededor de un eje consiguiendo así la sustentación.

- Drones con un rotor principal y un rotor de cola: En este tipo de drones la sustentación es generada por el rotor principal, que está ubicado en la parte superior de la aeronave mientras que el rotor de cola compensa el par de torsión que provoca el primero.
- Drones con único rotor: Drones con único rotor y unos alerones para compensar el par del rotor, lo que los convierte en difíciles de manejar.
- Drones con dos rotores en configuración coaxial: Esta configuración posee dos rotores, uno encima del otro, y cada uno con un sentido de giro diferente para que, a partir de la diferencia generada de la velocidad angular de ambos rotores consiga emprender el vuelo.
- Drones con dos rotores en configuración tándem: Drones con dos rotores en la parte superior, uno en cada esquina, que giran en direcciones opuestas para compensar el par generado.
- Multirrotores: Aeronave que posee tres o más rotores, que dependiendo de cuántos sean y su disposición pueden dividirse en más subgrupos.

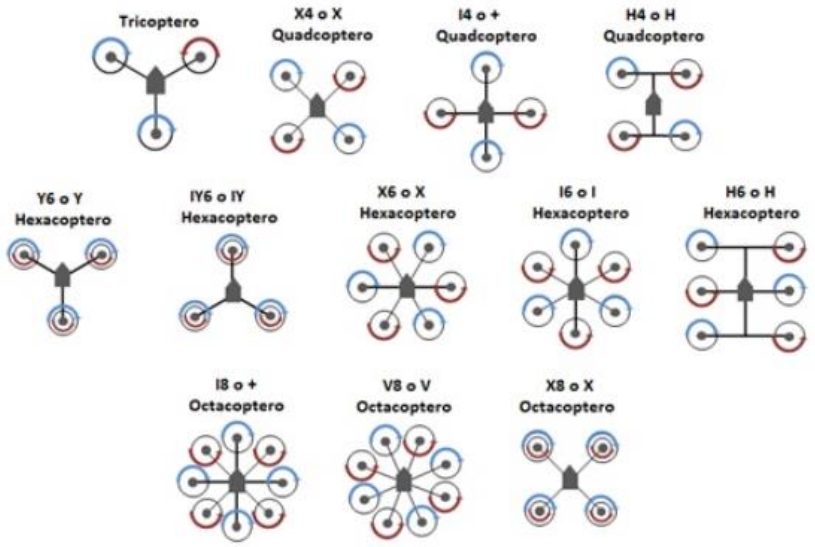

Figura 2.2. Diferentes configuraciones de multirrotores (Fuente: (2))

- Drones híbridos: Estos drones son capaces de despegar y aterrizar de forma vertical, como los drones de ala rotatoria, y de alcanzar grandes velocidades, como los drones de ala fija. (2)

2.5. Conceptos generales

Para entender el funcionamiento de un dron primero se han de revisar los principios básicos de vuelo y los fenómenos físicos que lo explican.

En este caso al ser un multirrotor su principio de sustentación en el aire se basa en crear una fuerza de empuje en dirección contraria a la gravedad y para ello se basan en rotores con hélices que, dependiendo de su tamaño y velocidad, ejercerán una determinada fuerza consiguiendo así que se mantengan en el aire.

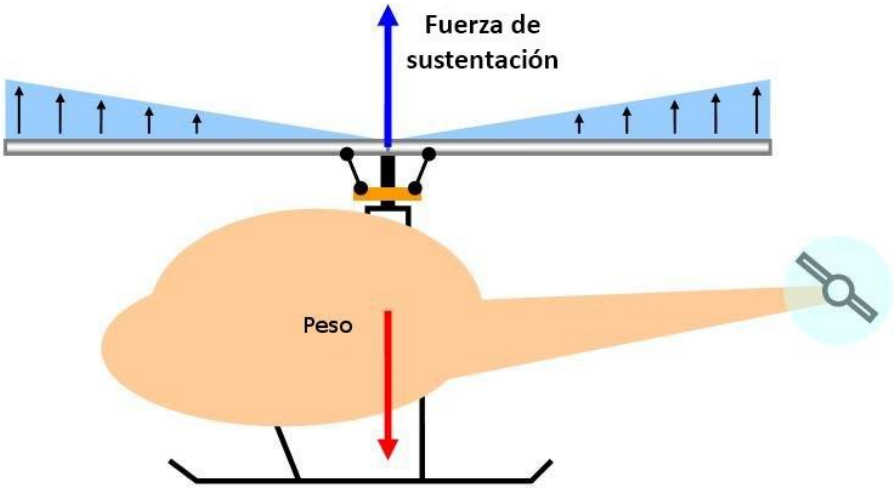

Figura 2.3. Principio de sustentación de un dron con ala rotatoria (Fuente: (3))

Una de las grandes ventajas de este tipo de aeronaves es que pueden permanecer inmóviles en el aire, aunque el problema viene dado cuando todas las hélices intentan girar en el mismo sentido generando un par que provocará que el dron se mueva de forma descontrolada.

En lo que respecta al movimiento del dron se ha de tener en cuenta que una vez esté en el aire se utilizarán los conocidos ángulos de navegación, que son los siguientes:

- *Pitch*: Rotar sobre este eje hará que el dron avance o retroceda.
- *Roll*: Moverse en este eje hará que el dron vaya hacia la izquierda o derecha.
- *Yaw*: Rotar sobre este eje hará que se mueva sobre su propio eje vertical.

Además, hay que asignar estos ángulos de navegación al mando, ya que hay diferentes tipos de configuración. Con lo que respecta a este proyecto se ha optado por seguir la configuración que marca la siguiente foto:

Figura 2.4. Configuración optada en el mando para los ejes en este proyecto (Fuente: (4))

Este modo de vuelo es el más extendido en el mundo de los drones.

En cuanto a la configuración que el cuadricóptero va a presentar, ésta será en 'equis', que se caracteriza por asignar dos motores a la parte frontal del cuadricóptero, cuyo sentido de giro apunta directamente a la parte delantera del centro del dispositivo, lo que provoca que sea mucho más fácil de maniobrar. (3)

Otra de las consideraciones que hay que tener en cuenta es la orientación del acelerómetro en un dron. Siempre ha de estar orientando con uno de los ejes de movimiento, orientando la 'Y' que aparece en el sensor con lo que se considere la parte delantera del dron, ya que cuando se mueva el eje *Pitch* hacia arriba, el dron se moverá hacia delante, es decir, hacia la dirección que marca la flecha 'Y'.

Cabe destacar, que esta orientación del sensor marcará además el número que se le asigna a cada motor, en este caso el motor de arriba a la derecha será el número 1, el de arriba a la izquierda el número 2, el de abajo a la izquierda el número 3 y finalmente, el de abajo a la derecha el número 4, tal como muestra la siguiente ilustración. (4)

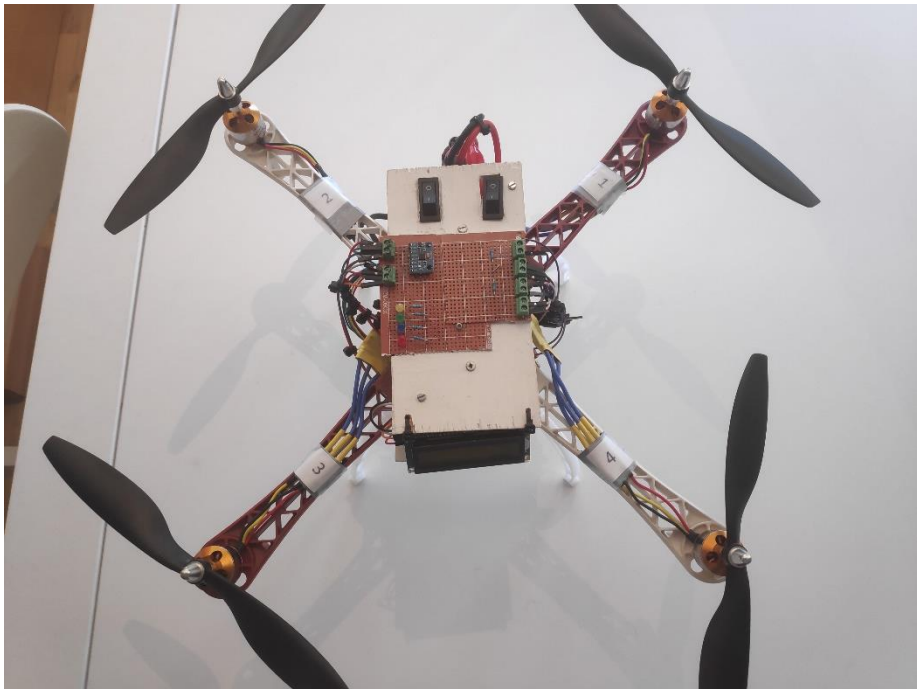

Figura 2.5. Asignación de numero de motores (Fuente: Propia)

Uno de los aspectos primordiales a la hora de montar un dron es el colocar bien las hélices ya que de lo contrario si están mal colocados puede ocurrir lo siguiente:

- El dron no se eleva.
- Solo se eleva la parte delantera o trasera.
- El dron se desplaza hacia delante o hacia atrás pero no se eleva. (5)

El giro de cada hélice ha de ser según muestra la imagen:

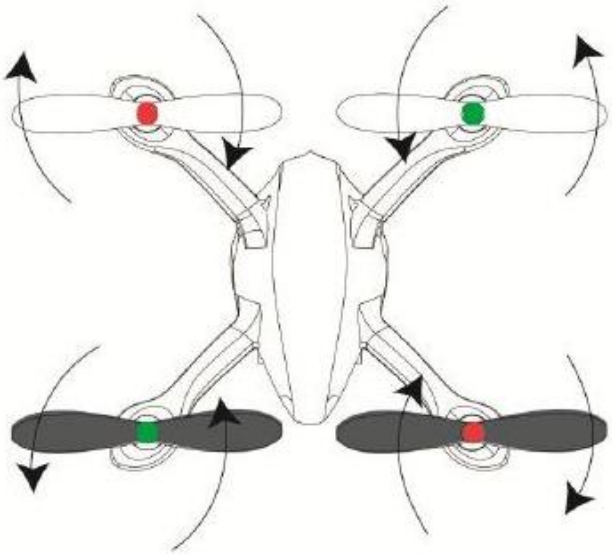

Figura 2.6. Sentido al cual ha de girar cada hélice (Fuente: (5))

Como se aprecia en la imagen las hélices correspondientes al motor 1 y 3 han de girar en sentido anti-horario, mientras que los motores 2 y 4 lo han de hacer en el horario, de esta forma se asegura que el dron se eleve y mantenga la estabilidad durante el vuelo.

3. Diseño del Dron

A la hora de montar el dron se han tenido una serie de consideraciones como la elección de componentes, el diseño mecánico y eléctrico o la función de cada uno de estos componentes en el sistema.

En este apartado se ahondará en el proceso de diseño y montaje final del cuadricóptero para obtener una visión más precisa de este largo proceso.

3.1. Elección de Componentes

La elección de los componentes del sistema es una de las partes más cruciales del diseño de cualquier sistema, dependiendo de la calidad, función o proporciones de los elementos se pueden a llegar a obtener unos resultados insatisfactorios o inesperados.

Por lo que respecta a este sistema, se ha tenido especial cuidado de que cada componente sea compatible con el anterior y que tampoco sea especialmente caro, teniendo en cuenta que normalmente el montaje de un dron suele ser costoso.

En los siguientes sub-apartados se han organizado la elección de los componentes en función de si están más encarados a la parte mecánica o electrónica del dron.

3.1.1. Mecánica

Chasis: Es la estructura donde se montará el dron, en este caso se trata de una estructura de unos 450 mm. La elección del chasis del dron determinará el comportamiento a nivel mecánico del sistema, como su aerodinámica, su rigidez o su inercia; además el material del cual está hecho, en este caso, fibra de carbono, determinará su resistencia ante un impacto.

La fibra de carbono suele ser el material más extendido en cuanto a un chasis del dron, ya que posee una serie de ventajas que otros materiales no poseen, como que el material es barato y es relativamente fuerte, rígido y ligero. Sin embargo, posee una desventaja con respecto a sus competidores, y es que bloquea las señales de radio, por eso es de vital importancia que el transmisor de radiofrecuencia no quede oculto por el chasis.

El chasis también determinará el tamaño de las hélices, en este caso las hélices tendrán que ser de 7''.

Figura 3.1. Chasis que será incorporado en el dron (Fuente: Propia)

Motores 1000 kV: Se necesitan mínimo 4 motores para hacer volar al dron. Es el material más importante a la hora de hacer girar el dron. Para el caso de un cuadricóptero normalmente se tratan de motores sin escobillas y, en este caso de una constante de 1000 kv. La constante kv se refiere a el número de revoluciones por minuto que será capaz de ofrecer el motor cuando se le aplique 1 V (6). En este caso como la batería es de 11,1 V el número máximo de revoluciones por minuto que será capaz de ofrecer el motor es de 11100 rpm. Este tipo de motores se caracterizan por recibir un tren de pulsos de corriente generado en sincronía para excitar sus bobinas de una forma concreta y de éste modo hacerlos girar, la gran ventaja que presentan con respecto a los motores con escobillas es que éstos, al disponer de escobillas, suelen tener un ciclo de vida más corto, ya que estas piezas normalmente se desgastan.

El tamaño del motor dependerá primeramente del tamaño del chasis, ya que debe encajar mecánicamente, y del peso total del conjunto del sistema para garantizar que el dron consiga volar de forma estable, como mínimo la fuerza total de los motores al 100% debería ser el doble de lo que pesa el dron.

En este caso el motor sin escobillas es un A2212, lo que indica que presenta 22 mm de diámetro de estator y 12 mm de altura del estator; cuanto más alto sea la altura del estator a más rpm conseguirá girar el motor y a cuanto más diámetro de estator más par generará el motor a menos rpm. (7)

Figura 3.2. Motores que serán utilizados en el montaje del dron (Fuente: Propia)

ESC de 30 A: Se necesitan 4 ESC, uno para cada motor. Los ESC o *Electronic Speed Controller* se encargan de controlar la velocidad de los motores y son imprescindibles en aplicaciones de este tipo. Estos componentes reciben la señal de consigna de velocidad del controlador del sistema y llevan al motor a la velocidad óptima proporcionando la corriente necesaria para alcanzar esta consigna. De manera más técnica, este tipo de componentes reciben una tensión continua de la batería y convierte esta tensión en una tensión alterna regulable en función de una señal PWM que reciba del controlador. Esta señal PWM deberá ser de frecuencia constante y de ancho de pulsos variable en función de la velocidad de giro que se quiera obtener.

A la hora de seleccionar el tipo de ESC es vital que la corriente que los motores consuman sea menor que la que los ESC puedan proporcionar, ya que si no el ESC podría calentarse en exceso. También hay que asegurarse de que el controlador de velocidad sea compatible con el voltaje de entrada de la batería, ya que si el voltaje de la batería es mayor podría quemar el componente.

Las conexiones entre los ESC y los motores deberá ser una conexión robusta y sin malos contactos para asegurar que durante el vuelo no se desconecte ninguno de los motores. Los conectores más habituales entre estos componentes son los conectores bala. (8)

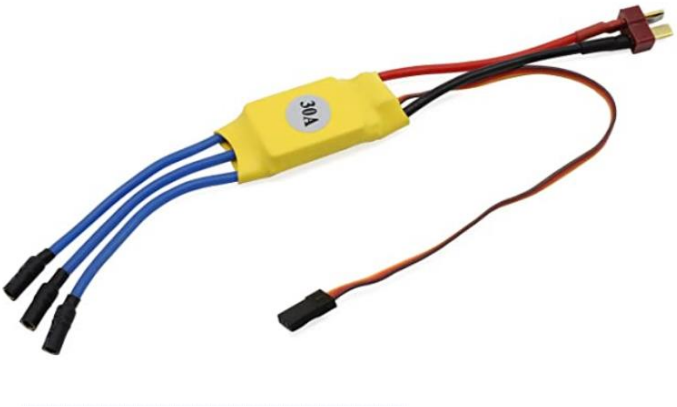

Figura 3.3. ESC que serán utilizados en el montaje del dron (Fuente: Propia)

Hélices: Se necesitan 4, una para cada motor y son elementos necesarios para conseguir que la estructura mecánica se eleve en el aire, en este caso se tratan de hélices 10x4,5R; que son compatibles con la estructura de chasis ya escogida.

Este material va acoplado al eje del motor sin escobillas y generan un empuje al mover al aire cuando giran, hay que tener en cuenta que cuanto más grande es la hélice mayor empuje puede generar al conjunto, pero a cambio consume más corriente de la batería.

En todos los cuadricópteros siempre hay dos de las hélices que giran en sentido horario y otras dos que giran en sentido anti-horario, es importante asegurar que los ejes de los motores giran en el sentido previsto antes de fijar las hélices, para ello simplemente con cambiar dos de los cables de los ESC a los motores se puede conseguir que el motor gire en sentido contrario. (9)

Figura 3.4. Hélices que serán incorporadas en el dron (Fuente: Propia)

Baquelita: Placa sobre el cual se soldarán los componentes electrónicos con estaño e irán fijadas en el centro del dron.

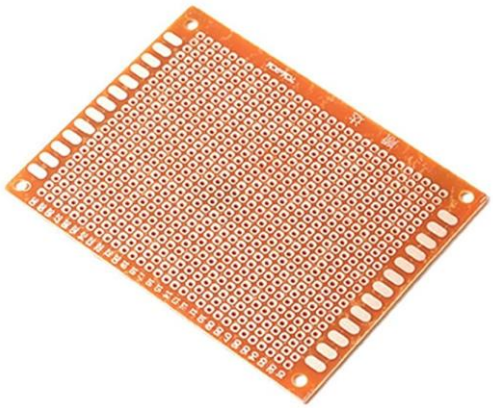

Figura 3.5. Placas que serán utilizadas para la electrónica del dron (Fuente: Propia)

Conector tornillo de 2 pines: Conectores para facilitar la conexión de las salidas de la placa del Arduino con los diferentes periféricos como las señales PWM de los ESC o los 5 V de alimentación de los sensores.

Figura 3.6. Conectores que serán utilizadas para conectar las señales de la CPU del dron a los sensores (Fuente: Propia)

3.1.2. Electrónica

Arduino UNO: La CPU donde se ejecutará el programa de control de vuelo, así como donde se tratarán las diferentes señales. El Arduino UNO es una placa de desarrollo que se basa en un chip de ATMEL llamado ATMEGA328 de 16 MHz, dispone de 14 entradas/salidas digitales, 6 entradas analógicas y un conector USB hembra entre otros conectores.

Esto a priori, es suficiente para controlar el dron ya que interesa que la placa disponga de las suficientes entradas/salidas para monitorizar los parámetros importantes mientras vuelta y necesita un chip lo suficientemente potente y rápido para que el cuadricóptero responda con rapidez a nuestros comandos. (10)

Esta placa se puede programar a través del entorno de programación Arduino IDE, que es un *software* de código abierto basado en Java que permite compilar el código en la placa y dispone de herramientas útiles como un monitor serie para visualizar las variables en tiempo real o un generador de gráficas. (11)

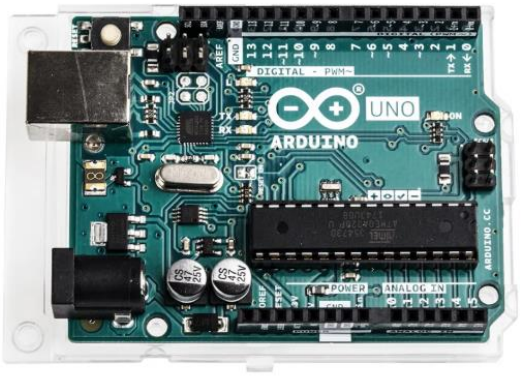

Figura 3.7. CPU utilizada para controlar el dron (Fuente: Propia)

Mando y Receptor RC: Necesario para controlar el vuelo del dron, es imprescindible conectar el receptor al Arduino para poder controlarlo.

Las emisoras de radio control son un tipo de mando inalámbrico que permite controlar el vuelo del cuadricóptero. En función del número de canales que disponga el emisor, más variables se podrán enviar a el control. Para este caso se emplearán cuatro canales, cada uno será necesario para que se reciban las señales de *pitch*, *roll*, *yaw* y *throttle* que se enviarán a través del mando. (12)

Para el caso de este mando, opera en una banda de 2.4 GHz, cómo es lo común en la gran mayoría de mandos de RC, ya que es una frecuencia libre, y tiene una sensibilidad de 10 bits.

Figura 3.8. Mando (derecha) y receptor RC (izquierda) utilizado para controlar el dron (Fuente: Propia)

Batería 3S 2200 mAh: Fuente de energía del dron. Estas baterías se caracterizan por poseer una alta densidad de energía, alta velocidad de descarga y pesan relativamente poco, lo cual las hace ideales para ser utilizadas como fuentes de alimentación para un dron, sin embargo, este tipo de baterías suelen ser bastante inestables, por tanto, hay que tener especial precaución a la hora de almacenarlas y utilizarlas, ya que si no se hace podrían llegar incluso a explotar.

Este tipo de baterías poseen dos tipos de cable, el cable principal, que es el que se utiliza para alimentar el circuito y el cable de balance.

La batería utilizada en el proyecto es 3S, lo que indica que dispone de tres celdas, cada celda tiene un voltaje nominal de 3,7 V; así pues, la batería es de 11,1 V. Este voltaje afecta directamente a la velocidad de los motores y, por tanto, a mayor voltaje mayor velocidad girarán los motores.

Otro dato importante es la capacidad de la batería, que en este caso es de 2200 mAh, que es básicamente una indicación de cuánta corriente se puede extraer de la batería en una hora hasta que quede inoperativa. (13)

Figura 3.9. Batería utilizada para alimentar el dron y su conjunto (Fuente: Propia)

Cargador de Baterías 3S: Cargador específico para recargar la batería 3S. Las baterías LiPo suelen ser cargadas por un cargador especial y hay que seguir un procedimiento concreto. El cable de balance que se ha nombrado anteriormente se conecta a este cargador para asegurarse que todas las celdas se cargan por igual, además de permitir monitorizar el voltaje de cada celda.

Cargar bien la batería es importante para asegurar su correcto funcionamiento y para alargar su período de vida, para ello hay que realizar dos tipos de carga. El primero de ellos es la carga balanceada, es la carga que se realizará antes de utilizar el dron, ya que monitoriza el voltaje que hay en cada celda y mientras realiza el proceso de carga mantiene constante el voltaje en todas las celdas. El segundo es la carga de almacenaje, éste se utilizará cuando se deje de usar el dron para, como su propio nombre indica, almacenar la batería; durante este proceso el cargador cargará cada celda a 3,80 o 3,85 V; que es un valor de voltaje que asegura el buen funcionamiento de la batería la siguiente vez que se utilice. (13)

Figura 3.10. Cargador de baterías específico para las baterías LiPo (Fuente: Propia)

LCD 16x2: Pantalla utilizada para visualizar alarmas y variables durante el vuelo.

En este caso se utilizará un LCD para señalar estados del dron, como la calibración de los ESC o sensores. Éste dispone de un adaptador I2C que facilita la conexión y programación del *display*, ya que, si no, se invertiría una gran cantidad de pines para conectarlo y de esta forma se ahorra conexionado. Además, con los pines del Arduino SDA y SCL es sumamente sencilla la programación de la pantalla. (14)

Figura 3.11. LCD utilizado para las señales de control del dron (Fuente: Propia)

Modulo GY-521: Es un módulo que incorpora el giroscopio y acelerómetro MPU6050 y un regulador de voltaje que permite ser alimentado directamente a 5 V/3.3 V en vez de los 3.3 V que únicamente admite el MPU-6050.

Es el IMU más utilizado, ya que ofrece una buena calidad a un precio asequible. Éste proporciona lecturas de la velocidad de rotación y aceleración en los tres ejes de movimiento. Se comunica a través del bus I2C de Arduino y el rango del acelerómetro puede ser ajustado a ±2 g, ±4 g, ±8 g, y ±16 g y el del giroscopio a ±250, ±500, ±1000, and ±2000 °/sec. (15)

Es el sensor más importante del proyecto, ya que sin él no sería posible hacerlo volar de manera controlada.

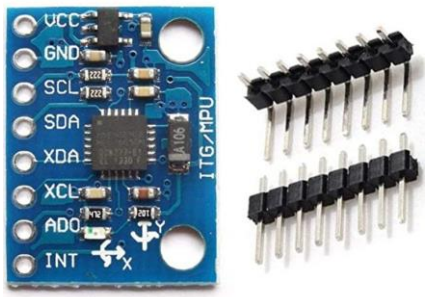

Figura 3.12. Placa que hace la función de acelerómetro del conjunto (Fuente: Propia)

Interruptor de 20 mA: Interruptor de dos posiciones para controlar la entrada de alimentación al circuito general y otro para el circuito de control.

Placa de distribución de potencia (PDB): PCB que se utiliza para distribuir la potencia de la batería y facilitar las conexiones.

En este caso se utilizará para soldar los cuatro ESC y que éstos reciban los 11,1 V de la batería y para alimentar el Arduino.

Figura 3.13. Placa que distribuye la potencia (Fuente: Propia)

Módulo HC-06: Módulo *Bluetooth* utilizado para transmitir o recibir datos del controlador a otro dispositivo que también disponga de *Bluetooth*.

Se ha optado por utilizar este tipo de comunicación puesto que actualmente cualquier dispositivo tiene integrado de fábrica *Bluetooth* y gracias a ello, cualquier móvil o *tablet* podrá utilizarlo para adquirir datos de forma remota del dron siempre y cuando tenga la aplicación instalada. El uso de este módulo es similar al uso del puerto serie del Arduino, siendo el mayor problema la necesidad de emparejar el módulo con el dispositivo que se utilice para la adquisición de datos.

Éste módulo en concreto solo puede actuar como *Slave*, lo que quiere decir que solo puede conectarse a un *Master* y, por tanto, tendrá que esperar a que se conecten a él para establecer la comunicación. Esté, se ha de conectar a través de los pines RX y TX del Arduino, que sirven para recibir y transmitir los datos respectivamente. (16)

Figura 3.14. Módulo para comunicarse por *Bluetooth* al Arduino (Fuente: Propia)

3.2. Diseño mecánico

En el diseño mecánico de la aplicación se ha tenido en cuenta diferentes aspectos, como el peso del conjunto o la configuración de hélices del propio dron.

Primeramente, se ha montado el chasis en forma de 'equis' garantizando así un vuelo estable y de forma robusta. Así es como ha quedado la estructura del dron tras montar el chasis:

Figura 3.15. Chasis del dron tras escoger el tipo de estructura del dron, en forma de 'equis' (Fuente: Propia)

Como se aprecia en la imagen este chasis dispone de agujeros donde es posible conectar los motores, a los extremos de los brazos, así como en el centro de la estructura, que posteriormente será utilizado para fijar de forma robusta las baquelitas con los diferentes componentes electrónicos. Además, la propia estructura incluye unas patas de apoyo para que el dron pueda aterrizar sin dañar la estructura.

El siguiente paso será fijar los motores en los extremos de los cuatro brazos que sobresalen del dron, es imprescindible que queden bien fijados para asegurar el correcto vuelo del vehículo. La estructura queda de la siguiente manera:

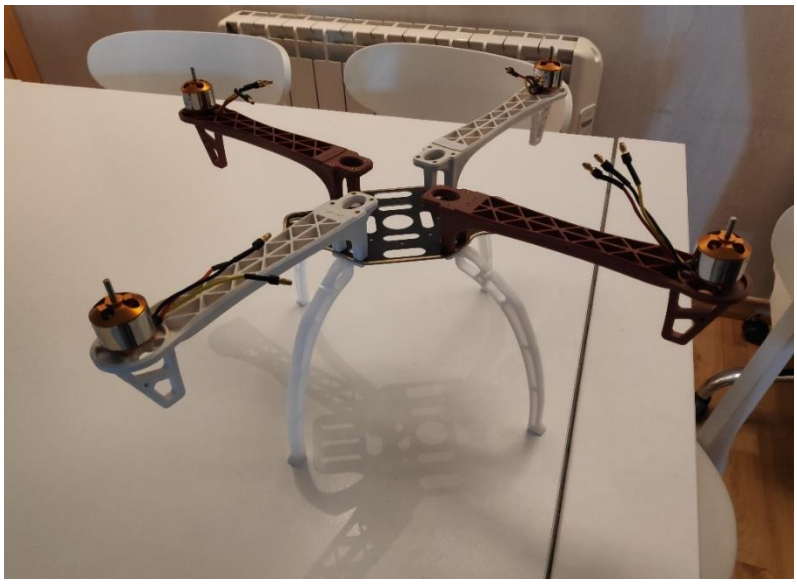

Figura 3.16. Chasis del dron tras fijar los motores en los extremos de los brazos. (Fuente: Propia)

Tras fijar los motores hay que conectar los ESC (*Electronic Speed Controller*) a éstos, para ello se ha usado cinta adhesiva para garantizar que la conexión entre los motores y los ESC no se interrumpa, ya que podría provocar la caída del dron y el posterior daño de toda la estructura.

Una vez se han fijado los ESC a los motores, hay que empezar a soldar los componentes electrónicos a las placas según marca el esquema eléctrico del apartado 3.3.

El siguiente paso será verificar las conexiones con el multímetro para verificar si las soldaduras se han realizado correctamente, también se puede probar a alimentar el Arduino con el cable USB y comprobar si el circuito de control se alimenta correctamente.

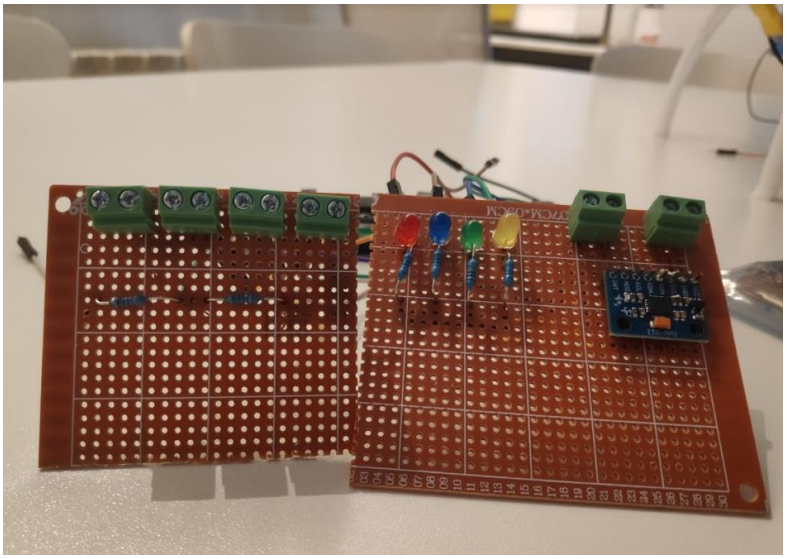

Figura 3.17. Placa de potencia con los conectores tornillo para los *ESC* (izquierda) y placa de control con el IMU y LED's (derecha) (Fuente: Propia)

Posteriormente y probadas las placas, con respecto a la fijación de los componentes en el dron, en este proyecto se ha optado por utilizar columnas de unos 10 mm para realizar dos pisos, abajo se situará el Arduino y la PDB y en el piso de arriba irán las placas donde van soldados el GY-521, los LED's, etc. Entre pisos se colocará el LCD, que quedará fijado en un extremo para que visualizar los estados del dron sea más cómodo.

Para ello se ha optado por usar una tabla de madera y cortarla de forma que la que iría situada en el piso de abajo sea más larga que la de arriba, de esta forma se garantiza una mayor estabilidad y que estéticamente el resultado sea más vistoso. Sobre esta tabla de madera se han hecho los agujeros necesarios para fijar los diferentes equipos y se ha pintado de blanco para que se respete los colores del chasis. Debajo del piso inferior se situará la batería, que es el componente con más peso y así se asegurará que el momento de inercia del conjunto sea menor y la estabilidad mayor; el emisor de RC y el módulo de *Bluetooth* también se situarán en el piso inferior ambos fijados con bridas.

Sobre estas tablas de madera, además, se harán unos agujeros para que quepan los interruptores del conjunto y se fijarán con silicona

Finalmente se cableará todos los componentes y con bridas se juntarán los cables de forma que el conjunto quede más ordenado y para asegurarse de que ninguna hélice rompa ningún cable accidentalmente o de que ningún cable se suelte durante el vuelo. Se adjuntan unas fotos del resultado final.

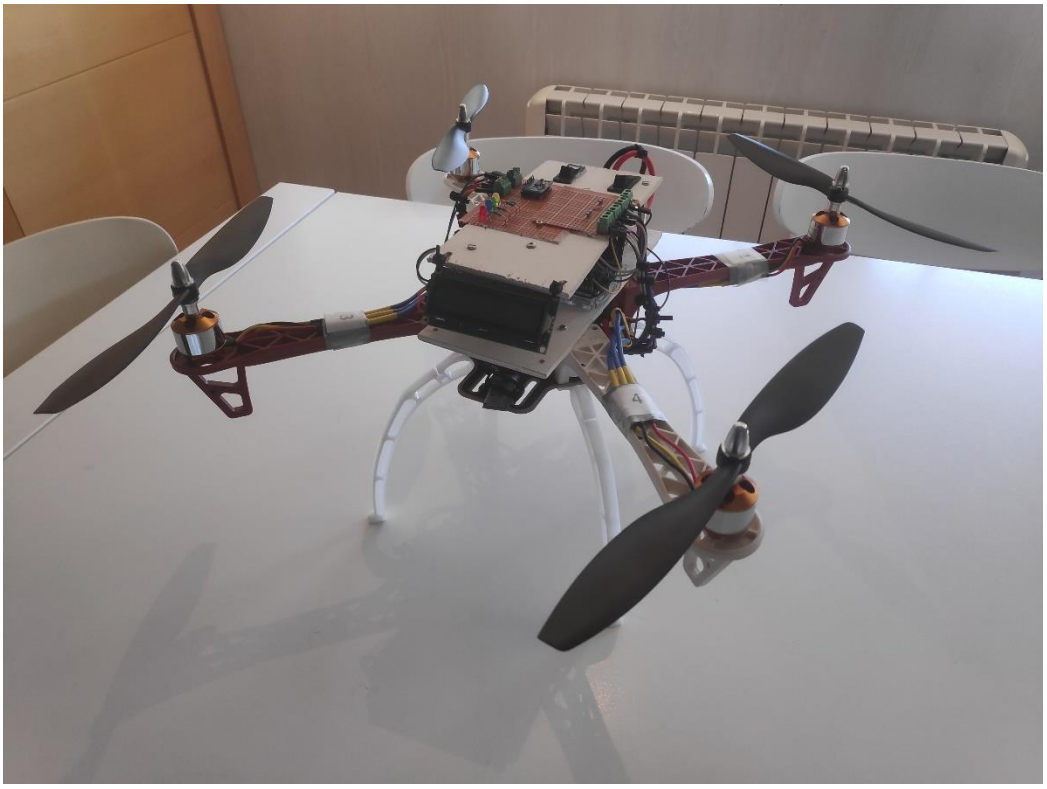

Figura 3.18. Dron del proyecto tras finalizar el montaje mecánico. (Fuente: Propia)

Figura 3.19. Dron del proyecto tras finalizar el montaje mecánico (Fuente: Propia)

3.3. Diseño del *hardware*

En éste apartado se trata de explicar el esquema eléctrico diseñado para la aplicación en concreto.

Éste ha sido realizado utilizando la plataforma *EasyEda*, una página web que permite editar esquemáticos *online* sin necesidad de descargar ningún s*oftware* adicional, sin embargo, carece de algunos modelos de componentes que se han utilizado en el proyecto, así pues, algunos componentes han sido modelizados como genéricos.

Durante el desarrollo del esquema final se han ido modificando algunas partes del mismo, por ejemplo, en un principio se planteó que la CPU fuera un Arduino Nano pero finalmente se optó por sustituirla por un Arduino UNO debido a su mayor velocidad de procesador, además de que el cableado entre los componentes era más sencillo.

Así pues, el esquema adjuntado es la última versión que se realizó y el que físicamente se montó en el dron.

En el esquemático que se plantea el elemento central es el Arduino UNO, el cual hace de controlador del sistema.

Antes de él la batería LiPo alimenta a la PDB, la cual no está modelizada en el esquema puesto que no está incluida en ninguna librería del programa, está alimentación es habilitada por un interruptor principal que habilita o no la entrada de potencia. La PDB es la encargada de distribuir la potencia a los ESC que controlan a los motores y, además, tiene una salida de 12 V que alimenta al Arduino a través de un interruptor secundario. La PDB tiene un conector XT60 mientras que la batería posee un conector tipo T macho, así que, se ha optado por colocar un adaptador entre ambos para realizar la conexión entre ambos componentes.

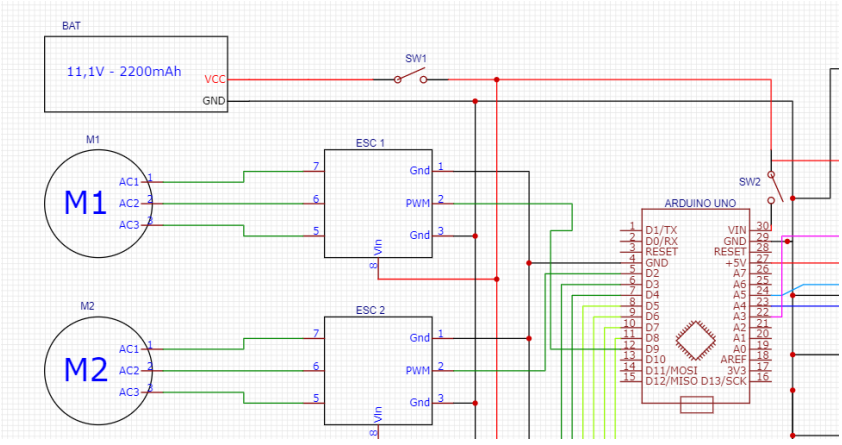

Figura 3.20. Conexión entre la batería LiPo y el Arduino (Fuente: Propia)

Con lo que respecta a los ESC y a los motores, los cables de alimentación de los ESC van conectados a una baquelita con cuatro conectores tornillo y esos a su vez van conectados a la PDB, ya que de esta forma se facilitan las conexiones, mientras que las señales PWM encargadas de controlar la velocidad a la cual girarán los motores van conectadas a una serie de entradas digitales del Arduino. De estas señales PWM simplemente se conectan dos cables, el de tierra y el de datos, éstos se llevan a unos conectores tornillos que van conectados directamente a las entradas digitales del controlador.

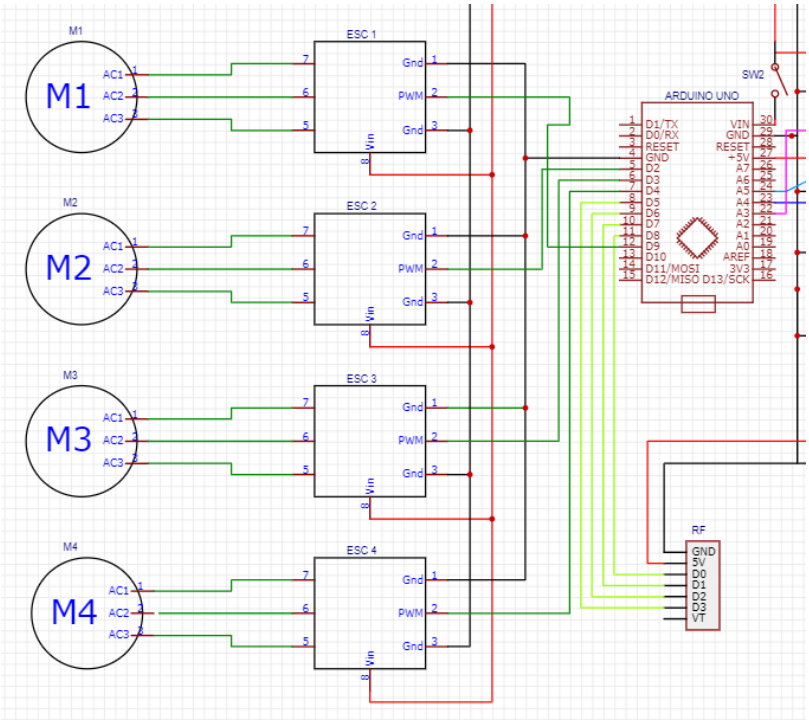

Figura 3.21. Conexión entre el Arduino y los ESC (Fuente: Propia)

Se han incorporado, además, un conjunto de LED's y un LCD para visualizar y monitorizar en todo momento los estados del sistema, así de cómo alertar en caso de que hubiese cualquier fallo. El LCD se ha optado por usarlo de forma orientativa en el caso de que el piloto del dron lo arranque por primera vez con estados como "Calibrando Giroscopio" o "Listo para volar", de esta forma el piloto es consciente de en qué fase del arranque está el cuadricóptero. Los LED's, sin embargo, se usan para avisar al programador de si el tiempo de ciclo se excede o de si está listo para volar, en este proyecto se ha optado por usar solo dos LED's y los otros dos quedan libres de uso. Además, se ha añadido un divisor de tensión con dos resistencias, una de 220 Ω y otra de 330 Ω, de esta forma se dimensiona para 5 V (lo máximo que puede leer una entrada analógica de Arduino) la tensión de la batería de 11,1 V y el Arduino sabrá siempre cual es el estado de la batería. Para conocer el voltaje de la batería se necesita entonces conectar un cable entre las dos resistencias que vaya a una entrada analógica del controlador.

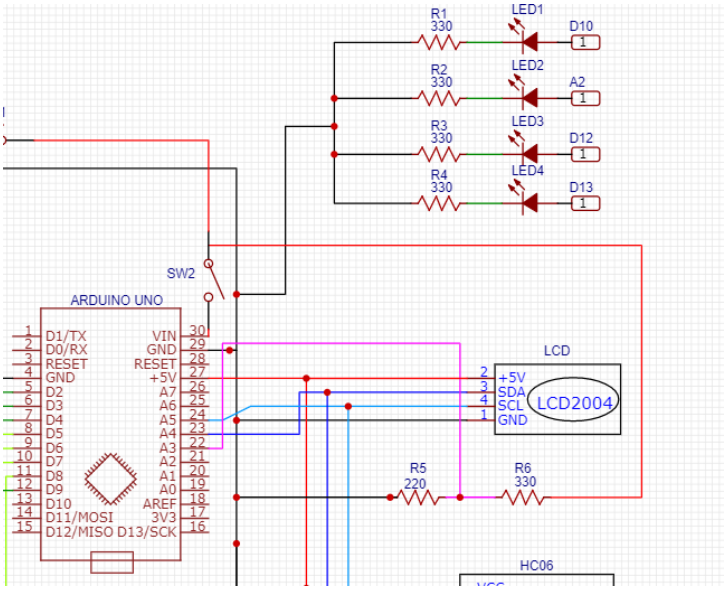

Figura 3.22. Conexión entre el Arduino los LED's, el LCD y el divisor resistivo (Fuente: Propia)

Al Arduino van conectados el sensor GY-521, que hace de acelerómetro y giroscopio, funciones imprescindibles a la hora de hacer volar un dron, y el LCD, ambos van conectados a los pines SDA y SCL del Arduino lo que indica que se utiliza el bus de campo I2C. En el Arduino UNO estos pines son el A4 y el A5, estos pines pueden variar en el caso de que se use un modelo de Arduino distinto.

Por otra parte, se incorpora el módulo *Bluetooth* HC-06 para comunicarse con el dron mientras está en funcionamiento y adquirir datos que va conectado a través de los pines 0 y 1, que son el RX y el TX. Estos pines son los que usa el controlador para volcar el programa a la CPU y, por tanto, cada vez que se quiera cargar un programa nuevo se tendrá que desconectar este sensor, ya sea su alimentación o estos dos pines

Finalmente, se conectan los cuatro canales del receptor de radiofrecuencia a entradas digitales del Arduino y se alimenta a 5 V. Si todo se realiza correctamente si se enciende el mando RC se debería encender un LED dentro del receptor que corrobore que está correctamente cableado.

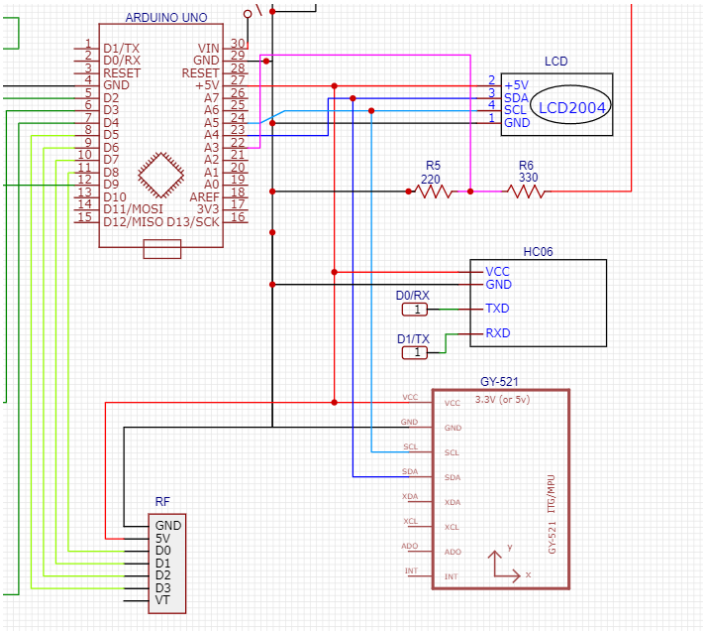

Figura 3.23. Conexión entre el Arduino y los diferentes sensores (Fuente: Propia)

3.4. Análisis del peso del conjunto

A lo largo de este apartado se tratará de calcular el peso total del conjunto para poder hacer una estimación de si con los motores que se dispone es suficiente para que el dron se eleve.

COMPONENTE	PESO [KG]	CANTIDAD	PESO TOTAL [KG]
CHASIS	0,481	1	0,481
ESC	0,018	4	0,072
MOTORES	0,081	4	0,324
HÉLICES	0,010	4	0,040
PLACA DE DISTRIBUCIÓN DE POTÉNCIA	0,040	1	0,040
ARDUINO UNO	0,045	1	0,045
DISPLAY LCD	0,080	1	0,080
BATERIA LIPO	0,184	1	0,184
TRANSMISOR RC	0,002	1	0,002
INTERRUPTOR DE DOS POSICIONES	0,033	2	0,066
SENSOR GY-521	0,002	1	0,002
MÓDULO HC-06	0,013	1	0,013
CONECTORES, RESISTENCIAS, LEDS…	x	x	0,100
TOTAL		1,449 kg	

Tabla 1.- Tabla de análisis de peso de los componentes (Fuente: Propia)

El resultado total que se ha obtenido del peso es de **1,449 kg**; teniendo en cuenta que los motores tienen una constante de 1000 kv cada uno y, como se ha comentado anteriormente, el empuje de los

motores tendría que ser como mínimo del doble de lo que pesa. Así pues, teniendo en cuenta que son motores de 1000 rpm/V se presupone que los motores tendrán la capacidad suficiente para alzar el dron, ya que comparando éste peso con otros drones del mercado de tamaño similar el peso es parecido y además los motores 2212 son los que más fuerza generan ya que son los recomendados para los chasis grandes.

4. Programación

Con respecto a la programación del controlador se han realizado una serie de programas probando cada componente y asegurando su correcto funcionamiento antes de realizar el programa final. Además, durante la programación se han utilizado librerías que facilitan la programación sobre las que se profundizará más adelante.

Las herramientas que incluye el Arduino IDE han facilitado la comprobación de los componentes, como por ejemplo el *Serial Plotter* que es un generador de gráficas en tiempo real para visualizar variables. A lo largo de este apartado se incluirán capturas de pantalla a los resultados de las pruebas utilizando esta herramienta.

En los siguientes apartados se tratará de desgranar las diferentes partes que entran en juego con respecto a la programación del proyecto y se tratará de profundizar sobre ellos.

4.1. Bus de campo I2C

Como se ha comentado con anterioridad, en este proyecto se utilizará el protocolo de comunicación I2C para enviar y recibir datos. Este protocolo de bus de campo, diseñado por *Philips Semiconductors*na principios de los 80 se basa en un bus con dos líneas de señal y masa que se utiliza de forma sincrónica y en serie. Las señales son las siguientes:

- SDA (*System Data*): Línea por la que se mueven los datos entre los dispositivos.
- SCL (*System Clock*): Línea de los pulsos de reloj que sincronizan el sistema.
- GND (Masa): Común de la interconexión entre todos los dispositivos del bus.

Las líneas SDA y SCL están a '1' (estado lógico alto) cuando están inactivas (para ello necesitan unas resistencias de *Pull-up* entre estas líneas y la alimentación), lo que permite conectar en paralelo varios dispositivos. (17)

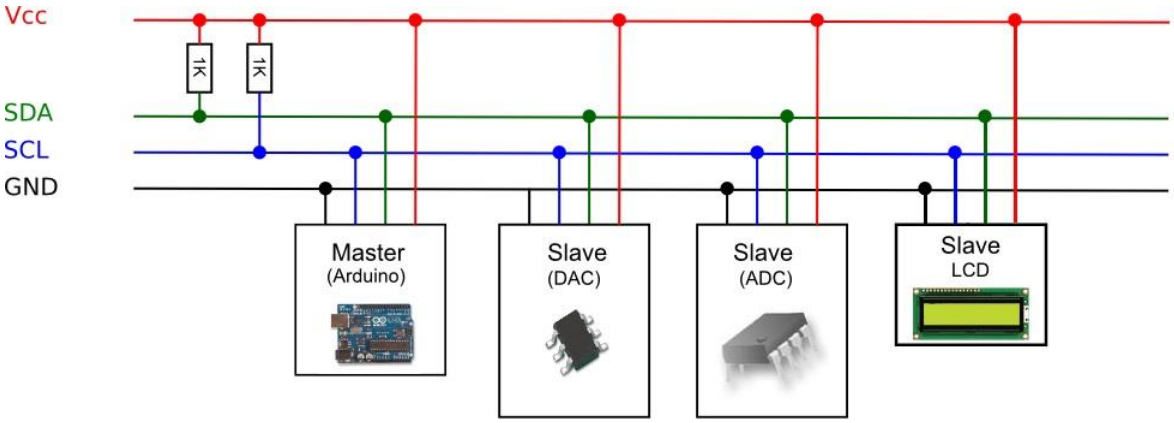

Figura 4.1. Topografía del bus de campo I2C (Fuente: (18))

Para establecer la comunicación entre equipos se necesita un dispositivo que actúe de maestro, determinando los tiempos y la dirección del tráfico del bus, en este caso el maestro será el Arduino; además, es el encargado de aplicar los pulsos de reloj en la línea SCL. Los demás dispositivos actuaran como esclavos, que son los que reciben las señales de los comandos y de reloj generados por el maestro.

Debe tenerse en cuenta que si se utiliza el Arduino UNO como maestro de bus I2C los pines de SDA y SCL son los pines A4 y A5 respectivamente y que las resistencias de *Pull-up* no son necesarias puesto que el Arduino las incorpora de forma interna.

Cada dispositivo que se conecte en este bus de campo tiene una dirección asociada que se puede encontrar en el *Datasheet* del componente o bien se puede programar un buscador de dispositivos I2C, en este caso se tiene dos direcciones: 0x68 del GY-521 y 0x27 del LCD. Para programar la mayoría de estos dispositivos se utiliza una librería de Arduino llamada 'Wire.h', la cual permite enviar datos desde el controlador (enviar mensajes que se escriban en el LCD) o bien recibir datos de alguno de los dispositivos (recibir los datos del acelerómetro / giroscopio). (18)

4.2. Estrategias de control de vuelo para drones

Las estrategias de vuelo más comunes a la hora de pilotar un dron son la acrobática y la estable.

En el modo de control acrobático solo se utilizarán las lecturas de velocidad angular, que se miden en °/s. En este modo el dron no volverá a su posición inicial de 0º, simplemente se contrarrestará la rotación debido a una ráfaga de aire o a que uno de los motores tiene más potencia, por lo que solo se compensará la velocidad de rotación a 0 °/s y no la de inclinación a 0º.

Hacer esto es posible gracias al PID o lazo de control. Este sistema se basa en tres elementos, el elemento proporcional, el integral y derivativo; el objetivo de estos tres elementos es alcanzar el estado de salida deseado, para este caso el PID de esta aplicación necesita que el error entre la consigna de velocidad y la velocidad real sea de 0 º/s, es decir, que la velocidad real sea exactamente igual a la consigna. (19)

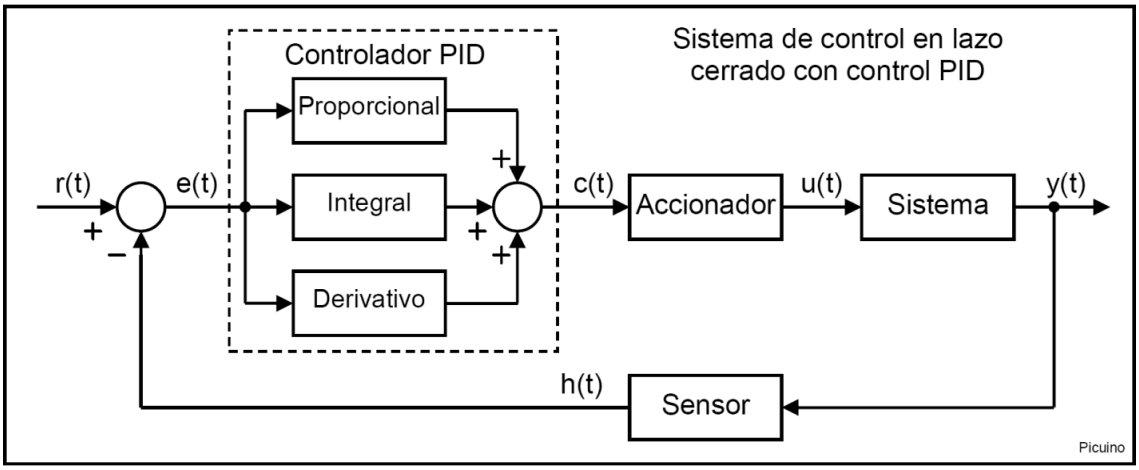

Figura 4.2. Sistema de control en lazo cerrado (Fuente: (20))

De esta forma, el objetivo del PID ha de ser de corregir el error que se ocasione entre la consigna que proporcione el mando RC y la que dé el sensor GY-521. Una vez que el PID reciba el error lo compensará acelerando o decelerando los motores correspondientes hasta contrarrestar la perturbación.

Los parámetros del PID tienen la siguiente función:

- El elemento proporcional (Kp) aumenta la velocidad del sistema y a su vez la inestabilidad del mismo.
- El elemento integral (Ki) disminuye el error del sistema en régimen permanente aunque aumenta ligeramente la inestabilidad del mismo.
- El elemento derivativo (Kd) aumenta la estabilidad del sistema, pero lo hace más lento. (20)

Ajustando estos parámetros se puede conseguir un sistema lo suficientemente estable y rápido para la aplicación que se desarrollará en este proyecto. Cabe destacar que a la salida del PID se obtendrá en milisegundos, esto es debido a que estos milisegundos afectan directamente a la duración del pulso de PWM en estado alto, y, cuanto más dure este pulso, a mayor velocidad girarán los motores.

Finalmente, se enviará una señal para cada motor y estas señales nunca deben superar los 2 ms, que es la consigna máxima de potencia para los motores.

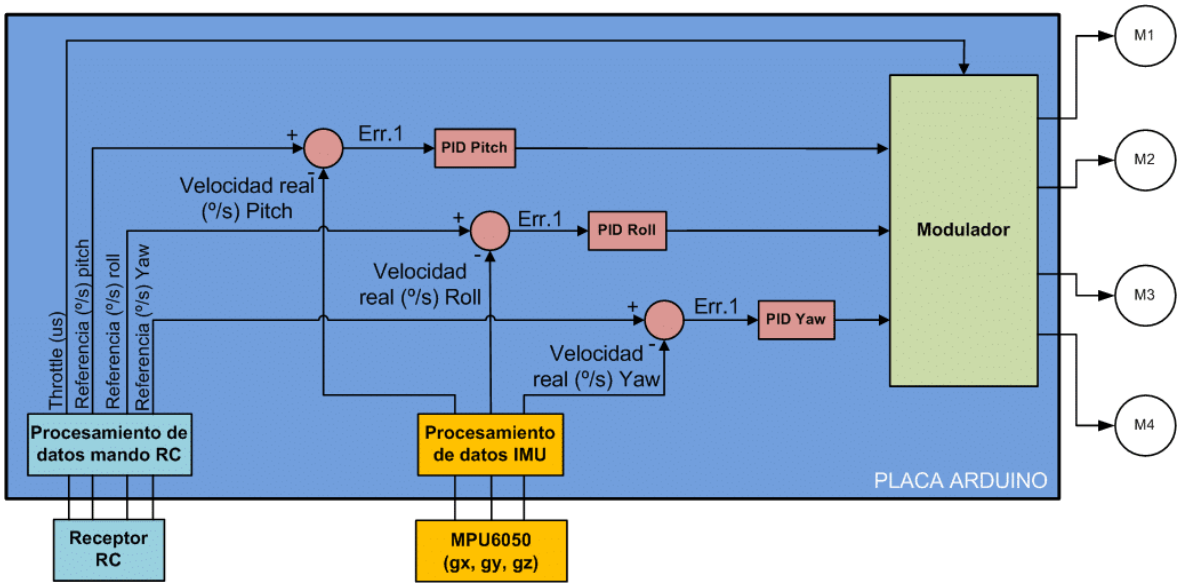

Figura 4.3. Estrategia de control para la aplicación en modo acrobático (Fuente: (21))

Volviendo a las estrategias de control nombradas anteriormente, ambos modos utilizan esta estrategia de lazo de control. La principal diferencia entre los dos modos es que el acrobático no vuelve a su posición inicial si se suelta la palanca del mando RC, mientras que en el modo estable siempre volverá a su posición de 0º de inclinación cuando se suelte la palanca.

Además, la estrategia de control del modo estable varia con respecto al acrobático. En este caso, lo primero que se hace es comparar la consigna de inclinación que llega desde el mando RC con la lectura que da el sensor GY-521. La primera gran diferencia con respecto a este modo de control es que aquí la consigna es la inclinación en [º], y no la velocidad de rotación en [º/s]. La variable de salida que da el primer PID al comparar la consigna con la inclinación real será la entrada de otro PID, que será el mismo utilizado en el modo acrobático, de esta forma se indica al dron que si está inclinado ha de aumentar o reducir la velocidad de los motores para volver a la posición inicial de 0º.

A modo de resumen y como se aprecia en la siguiente imagen, el primer PID compara la consigna que recibe del mando RC con la que proporciona el sensor y, si hay una desviación entre estos valores, este error será la entrada del segundo PID, que acelerará o decelerará los motores para que el dron pueda volver a su posición inicial de 0º. (21)

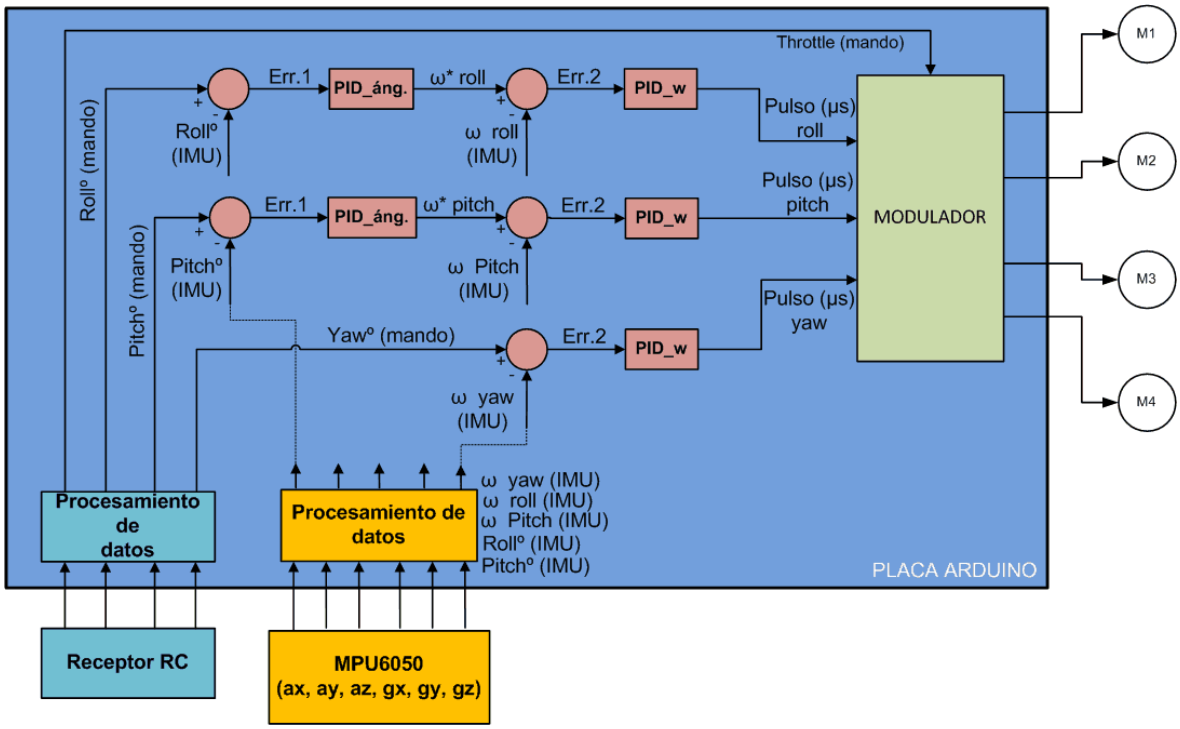

Figura 4.4. Estrategia de control para la aplicación en modo estable (Fuente: (21))

4.3. Calibración de los ESC

Antes de hacer volar el cuadricóptero lo primero que hay que hacer es calibrar los ESC para que éstos conozcan los valores mínimos y máximos de PWM que el controlador de vuelo enviará.

Es importante realizar este paso con las hélices quitadas y alimentando el Arduino con el cable USB.

Así pues, para calibrar los ESC hay que realizar los siguientes pasos:

1. Lo primero que hay que hacer es conectar los ESC a los motores.
2. Encender el mando de RC y subir el *Throttle* al máximo.
3. Habilitar el interruptor de potencia de la batería.
4. En este paso se han de escuchar dos pitidos.
5. Si se han escuchado, bajar el *Throttle* al mínimo.
6. Si se escuchan tres pitidos es que la calibración se ha realizado correctamente.

Cabe destacar que este paso solo se ha de realizar una vez y, por tanto, una vez se ha completado este paso, ya se podrían mover los ejes de los motores con total libertad.

4.4. Depuración del programa

Antes de hacer volar al dron hay que hacer una serie de comprobaciones de los distintos elementos del dron para corroborar su correcto funcionamiento.

De estas pruebas se considera primordial que funcione lo siguiente:

- El controlador lee las consignas del mando RC.
- El controlador lee los valores que da el sensor GY-521.
- El PID responde correctamente ante la variación de ángulo del dron.

4.4.1. Comprobación mando RC

Para comprobar que el Arduino lee correctamente las consignas que proporciona el mando primeramente hay que definir a que entrada del Arduino se asocia a cada movimiento del dron. Según muestra el esquema eléctrico, se ha realizado de la siguiente forma:

- La entrada digital 5 corresponde al *Yaw*.
- La entrada digital 6 corresponde al *Throttle* o Potencia.
- La entrada digital 7 corresponde al *Pitch*.
- La entrada digital 8 corresponde al *Roll*.

Una vez cableado el receptor hay que entender el funcionamiento de tanto emisor como receptor. El emisor envía ondas de radio en forma de señales PWM de 50 Hz que pueden variar de 1 ms a 2 ms en función de la posición del *stick* del mando, mientras que el receptor, una vez recibe estas ondas de radio genera pulsos de 3,3 V de forma secuencial al Arduino.

El programa que ejecuta una placa de desarrollo Arduino es secuencial, es decir, se ejecuta línea por línea y por ello la forma de programar correctamente un emisor es mediante interrupciones, que básicamente, cada vez que reciben un flanco del emisor en alguno de los pines detienen la ejecución secuencial del Arduino y ejecutará la parte de código asociada a dicha acción. En la siguiente foto se puede apreciar cómo se ha realizado. Si en el Arduino se instala la librería 'EnableInterrupt.h' se puede convertir cualquier pin en interrupción, como se aprecia en la siguiente imagen se han de declarar cuatro interrupciones, una por cada canal.

```
void setup() {
  pinMode(5, INPUT_PULLUP);                  // YAW
  enableInterrupt(5, INTyaw, CHANGE);
  pinMode(6, INPUT_PULLUP);                  // POTENCIA
  enableInterrupt(6, INTpotencia, CHANGE);
  pinMode(7, INPUT_PULLUP);                  // PITCH
  enableInterrupt(7, INTpitch, CHANGE);
  pinMode(8, INPUT_PULLUP);                  // ROLL
  enableInterrupt(8, INTroll, CHANGE);

  Serial.begin(115200);
  delay(100);
}
```

Figura 4.5. Declaración de las interrupciones en un programa de Arduino (Fuente: Propia)

Una vez se han declarado los pines como interrupciones hay que asociar cada interrupción a una acción. El código funciona de la siguiente manera: Si el Arduino lee que ha habido un flanco positivo, éste activa un contador y sigue ejecutando esa parte de código hasta que le llega el flanco negativo que es cuando detiene el contador, de esta forma se puede saber de cuanto es el intervalo de tiempo que ha estado activa la onda PWM.

```
volatile long contYawInit;        // YAW
volatile int PulsoYaw;
void INTyaw() {
  if (digitalRead(5) == HIGH) contYawInit = micros();
  if (digitalRead(5) == LOW) PulsoYaw = micros() - contYawInit;
}

volatile long contPotenciaInit;   // POTENCIA
volatile int PulsoPotencia;

void INTpotencia() {
  if (digitalRead(6) == HIGH) contPotenciaInit = micros();
  if (digitalRead(6) == LOW)PulsoPotencia = micros() - contPotenciaInit;
}

volatile long contPitchInit;      // PITCH
volatile int PulsoPitch;
void INTpitch() {
  if (digitalRead(7) == HIGH) contPitchInit = micros();
  if (digitalRead(7) == LOW) PulsoPitch = micros() - contPitchInit;
}

volatile long contRollInit;       // ROLL
volatile int PulsoRoll;
void INTroll() {
  if (digitalRead(8) == HIGH) contRollInit = micros();
  if (digitalRead(8) == LOW) PulsoRoll = micros() - contRollInit;
}
```

Figura 4.6. Activación de las interrupciones para cada acción asociada al mando (Fuente: Propia)

Para comprobar si realmente el controlador está leyendo estas señales se grafica aprovechado el *Serial Plotter* del Arduino y se obtiene lo siguiente:

```
void loop() {
  while (micros() - loop_timer < 10000);

  tiempo_ejecucion = (micros() - loop_timer) / 1000;
  loop_timer = micros();

  Serial.print(PulsoPotencia);
  Serial.print("\t");
  Serial.print(PulsoPitch);
  Serial.print("\t");
  Serial.print(PulsoRoll);
  Serial.print("\t");
  Serial.println(PulsoYaw);
}
```

Figura 4.7. *Loop* principal del código para graficar las diferentes señales del mando RC (Fuente: Propia)

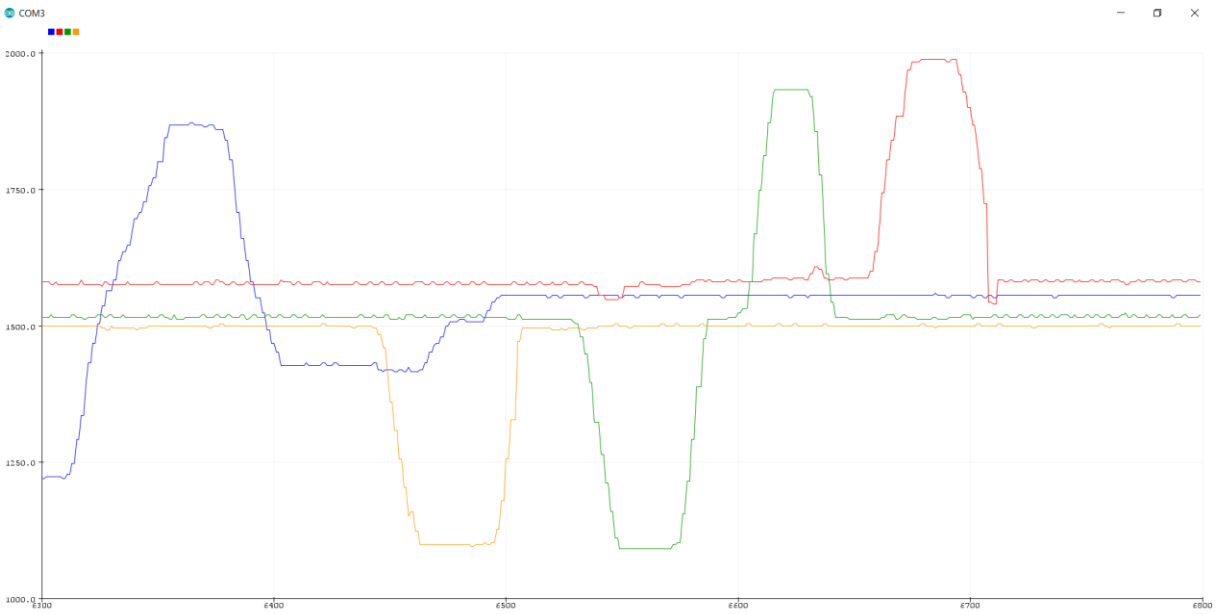

Figura 4.8. Señales que recibe el controlador del mando: *Throttle* en azul, *Pitch* en rojo, *Roll* en verde, *Yaw* en amarillo (Fuente: Propia)

Como se aprecia en la imagen, el controlador reacciona a todas las señales generadas por el mando RC y además se puede apreciar que estos valores se mueven entre 1000 µs y 2000 µs, corroborando así que el mínimo y máximo intervalo de tiempo recibido para cada señal PWM es 1 ms y 2 ms respectivamente.

En el código final se tendrá que indicar al Arduino la conversión de esos milisegundos a grados, puesto que con el mando indicaremos cuantos grados se quiere que se incline el dron hacia una cierta dirección.

4.4.2. Comprobación sensor GY-521

Otra parte crucial para que funcione el cuadricóptero es que el sensor GY-521 ha de proporcionar datos fiables y reales, puesto que el controlador necesita saber en todo momento la inclinación y la velocidad de rotación de los tres ejes.

Hay que tener en cuenta que la dirección de los ejes está marcada en el propio sensor.

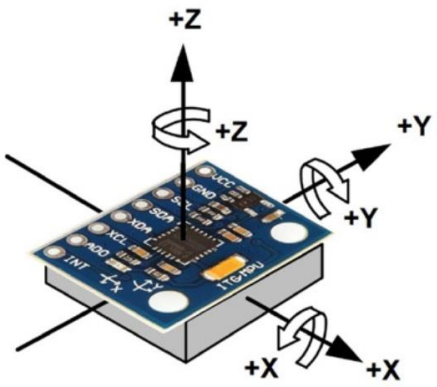

Figura 4.9. Dirección que sigue cada uno de los tres ejes del acelerómetro/giroscopio (Fuente: (22))

Para comunicarnos con este sensor cabe destacar que utiliza el protocolo de comunicación I2C con la dirección asociada 0x68 lo cual facilita bastante la programación del mismo, puesto que la librería 'Wire.h' es ideal para este tipo de dispositivos con comunicación I2C. (22)

Ahora bien, para tratar de entender los valores que proporciona este sensor hay que hablar de la gravedad. Puesto que es un acelerómetro y siempre mide la aceleración de los ejes, aunque esté quieto uno de esos ejes, siempre está sometido a la aceleración de la gravedad, lo cual, lo hace una ventaja para saber con seguridad que los datos del sensor son fiables. Puesto que se conoce que la aceleración es siempre 9,81 m/s^2 si se dejara el dron quieto sobre el plano horizontal se sabrá que uno de los ejes ha de dar igual aproximadamente ese valor.

No obstante, aunque hacer esta validación serviría para comprobar si el sensor proporciona medidas veraces, mientras el dron esté funcionando interesa que proporcione la velocidad de rotación en [º/s] y la inclinación del dron en [º]. Para ello hay que convertir los valores que da el sensor a los deseados mediante ecuaciones matemáticas. (23)

Para el caso de velocidad de rotación basta con utilizar los valores que proporciona el sensor y ajustarlo a la configuración que se esté utilizando en el giroscopio. En este caso se ha utilizado la sensibilidad de

±8 g, lo que, mirando el *datasheet* de éste equivale a 4096 LSB/g, lo que quiere decir que por cada 4096 unidades que dé el sensor equivaldrán a 1 g de aceleración o 9,8 m/s^2.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	

Figura 4.10. Especificaciones del acelerómetro del sensor MPU-6050 (Fuente: (24))

Para la velocidad de rotación en cambio, para una sensibilidad de ±500 dps, cada 65,5 LSB equivaldrá a 1 °/s de rotación, según el *datasheet*. Cabe destacar que el sensor utilizado para el proyecto, el GY-521 tiene integrado el sensor MPU-6050 y por tanto el *datasheet* que se mirará será éste:

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	

Figura 4.11. Especificaciones del giroscopio del sensor MPU-6050 (Fuente:(24))

Conociendo esta información se tendrá que dividir los valores que proporcione al sensor por 65,5 para así obtener la velocidad angular en [°/s].

```
// Cálculo Velocidad Angular
pitchGyro =    (gx - gyro_X_cal) / 65.5;   //  gx, gy y gz son los valores que proporciona el sensor
rollGyro =     (gy - gyro_Y_cal) / 65.5;   //  giro_X_cal, giro_Y_cal y giro_Z_cal son los valores medios
yawGyro =      (gz - gyro_Z_cal) / 65.5;   //  de cada uno de los ejes para saber el "offset" del sensor
```

Figura 4.12. Cálculo de la velocidad angular en el programa de Arduino (Fuente: Propia)

Como se aprecia en la figura anterior, es necesario además conocer los valores medios de cada uno de los ejes para poder corregir la desviación del sensor con el valor actual, para ello se recogerán 3000 muestras del sensor antes de iniciar el vuelo y se señalizará con la activación de un LED, una vez se han realizado todas esas muestras se saca el valor medio para cada eje y se desactiva el LED. De esta forma ya se conoce la desviación del sensor.

```
void setup() {
  Wire.begin();
  Serial.begin(115200);
  pinMode(10, OUTPUT);      // LED
  init_gyro();              // Inicializar sensor

  digitalWrite(10, HIGH);   // Encender un Led durante la calibracion

  //Calibrar giroscopio
  for (cal_int = 0; cal_int < 3000 ; cal_int ++) {
    MPU_6050();            // Leer los datos del sensor 3000 veces y sacar el valor medio para obtener el offset
    gyro_X_cal += gx;
    gyro_Y_cal += gy;
    gyro_Z_cal += gz;
    delayMicroseconds(1000);
  }
  gyro_X_cal = gyro_X_cal / 3000;       // Valor medio de las 3000 muestras
  gyro_Y_cal = gyro_Y_cal / 3000;
  gyro_Z_cal = gyro_Z_cal / 3000;

  //Calibrar acelerometro
  for (cal_int = 0; cal_int < 3000 ; cal_int ++) {
    MPU_6050();
    angle_pitch_acc_cal += ax;
    angle_roll_acc_cal += ay;
    angle_yaw_acc_cal += az;
  }
  angle_pitch_acc_cal = angle_pitch_acc_cal / 3000;
  angle_roll_acc_cal = angle_roll_acc_cal / 3000;
  angle_yaw_acc_cal = angle_yaw_acc_cal / 3000;
  accCalibOK = true;

  digitalWrite(10, LOW);
  loop_timer = micros();
}
```

Figura 4.13. Calibración del acelerómetro/giroscopio en el programa de Arduino (Fuente: Propia)

Una vez se ha calculado la velocidad angular solo falta el ángulo. Para ello se ha de multiplicar el valor que se ha obtenido previamente de la velocidad angular por el tiempo que el dron ha estado rotando a esa velocidad.

```
// Cálculo Ángulo
angulo_pitch += pitchGyro * tiempo_ejecucion / 1000 ;
angulo_roll += rollGyro * tiempo_ejecucion / 1000 ;
```

Figura 4.14. Cálculo del ángulo en el programa de Arduino (Fuente: Propia)

A éstas fórmulas de cálculo de ángulo se le podría añadir además una fórmula que intente compensar la deriva del sensor. La idea es conseguir eliminar el ruido o deriva para que el sensor no cambie de ángulo al detectar otra fuerza que no sea de la gravedad.

La forma más sencilla de corregir este problema es con el filtro complementario, que es la unión de un filtro pasa-bajos y otro pasa-altos. Y la fórmula es la siguiente:

$$Angulo = (Angulo + AnguloGyro * \Delta t) * 0{,}98 + (AnguloAcc) * 0{,}02 \qquad \textbf{(Eq. 4.1)}$$

Donde el 'AnguloGyro' es el ángulo calculado anteriormente y el 'AnguloAcc' es el ángulo del acelerómetro calculado en forma de tangente. En el programa de Arduino quedaría de la siguiente forma:

```
angulo_pitch +=   angulo_roll * sin((gz - gyro_Z_cal) * tiempo_ejecucion  * 0.000000266);    // (tiempo_ejecucion/1000) * (1/65.5) * (PI/180)
angulo_roll -=    angulo_pitch * sin((gz - gyro_Z_cal) * tiempo_ejecucion  * 0.000000266);

acc_total_vector =  sqrt(pow(ay, 2) + pow(ax, 2) + pow(az, 2));
angle_pitch_acc =   asin((float)ay / acc_total_vector) * 57.2958;     // 57.2958 = Conversion de radianes a grados 180/PI
angle_roll_acc =    asin((float)ax / acc_total_vector) * -57.2958;

angulo_pitch =  angulo_pitch * 0.98 + angle_pitch_acc * 0.02;        // Corrige la deriva con el filtro complementario
angulo_roll =   angulo_roll * 0.98 + angle_roll_acc * 0.02;
```

Figura 4.15. Cálculo del ángulo teniendo en cuenta la deriva del sensor en el programa de Arduino (Fuente: Propia)

De esta forma se corrige el error acumulativo del ángulo en fuerzas no deseadas (25). Finalmente, para comprobar que tanto el ángulo del *Pitch* como el del *Roll* se gráfica con la herramienta *Serial Plotter* del Arduino y queda de la siguiente manera:

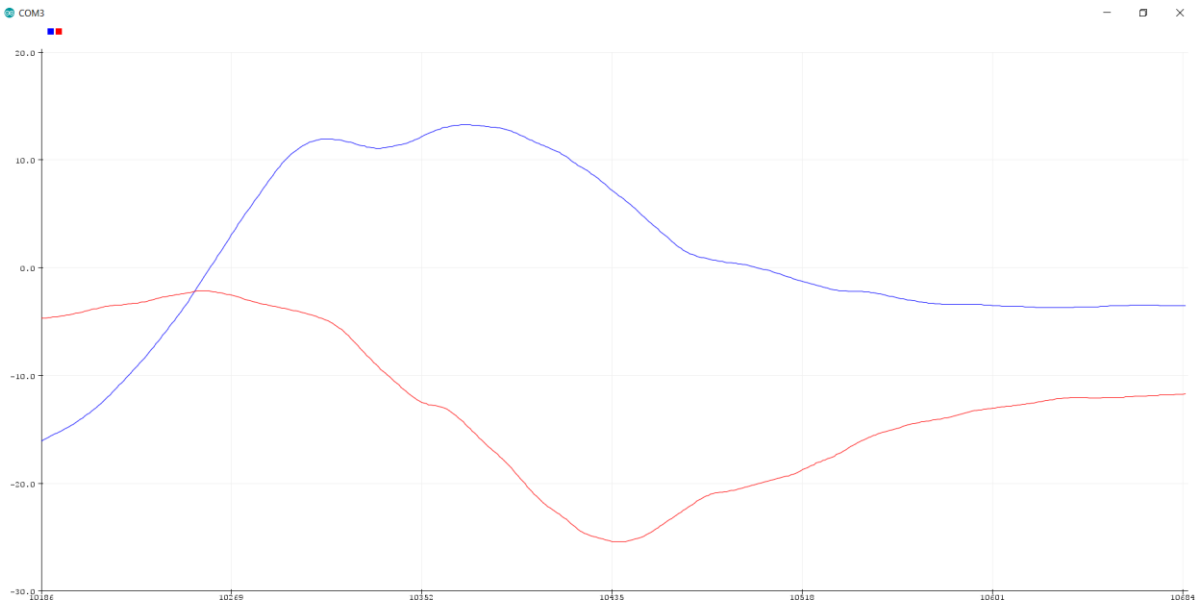

Figura 4.16. Señales en el programa de Arduino del ángulo del *Pitch* en azul y el ángulo del *Roll* en rojo (Fuente: Propia)

La forma de comprobar si los signos de los ángulos están bien es inclinar el dron hacia adelante, donde marca la flecha 'Y' y comprobar que el ángulo *Pitch* es negativo e inclinando el dron hacia la derecha, donde marca la flecha 'X' el ángulo *Roll* es positivo.

4.4.3. Comprobación PID de control

Ésta comprobación es la más importante y de ser correcta, garantizaría que el dron está listo para volar. Como se ha comentado anteriormente, el PID es el responsable de gestionar y corregir el error que se produce entre la salida del controlador y valor real, así pues, la programación se encara teniendo en cuenta que se necesita un PID para cada eje (*Pitch*, *Roll* y *Yaw*) y que lo que varía son los milisegundos que está activo el PWM en cada uno de los ESC. Se parte de las siguientes fórmulas:

$$ESC1 = Throttle - Salida\ PID\ Pitch + Salida\ PID\ Roll + Salida\ PID\ Yaw \quad \textbf{(Eq. 4.2)}$$

$$ESC2 = Throttle - Salida\ PID\ Pitch - Salida\ PID\ Roll - Salida\ PID\ Yaw \quad \textbf{(Eq. 4.3)}$$

$$ESC3 = Throttle + Salida\ PID\ Pitch - Salida\ PID\ Roll + Salida\ PID\ Yaw \quad \textbf{(Eq. 4.4)}$$

$$ESC4 = Throttle + Salida\ PID\ Pitch + Salida\ PID\ Roll - Salida\ PID\ Yaw \quad \textbf{(Eq. 4.5)}$$

Los signos de estas ecuaciones dependen de cómo esté orientado el acelerómetro en el dron y es importante que nunca ninguna de estas señales supere los 2 ms. Al cálculo de estas señales se le llama Modulador:

```
// Si el throttle es mayor a 1300us, el control de estabilidad se activa.
if (pulsoPotencia > 1300) {
  if (pulsoPotencia > 1800)pulsoPotencia = 1800;              // Limitar throttle a 1800 para dejar margen a los PID
  fESC1 = PotenciaFilt + PID_pitch_w - PID_roll_w - PID_yaw_w; // Motor 1
  fESC2 = PotenciaFilt + PID_pitch_w + PID_roll_w + PID_yaw_w; // Motor 2
  fESC3 = PotenciaFilt - PID_pitch_w + PID_roll_w - PID_yaw_w; // Motor 3
  fESC4 = PotenciaFilt - PID_pitch_w - PID_roll_w + PID_yaw_w; // Motor 4
```

Figura 4.17. Cálculo de las señales de los ESC del PID en el programa de Arduino (Fuente: Propia)

El lazo de control en este proyecto es vital y será crucial además ejecutarlo de forma constante cada cierto tiempo, en este caso se ejecutará cada 7 ms.

El siguiente paso será programar los PID, en este caso se ha tomado como referencia un programa de PID y se ha adaptado al código completo del Arduino puesto que programar de cero un PID es algo que requiere mucho tiempo y a su vez se dispone de mucha información en Internet.

```
//PITCH GYRO ang
pitch_ang_error_prop = wPitchConsigna - angulo_pitch;                                          // Error entre lectura y consigna
ITerm_pitch_ang += (Ki_pitch_ang * pitch_ang_error_prop);                                      // Parte integral (sumatorio del error en el tiempo)
ITerm_pitch_ang = constrain(ITerm_pitch_ang, salidaPI_pitch_ang_min1, salidaPI_pitch_ang_max1); // Limitar parte integral
DPitch_ang = Kd_pitch_ang * (angulo_pitch - pitch_ang_giroscopio_anterior);                    // Parte Derivativa (diferencia entre el error actual y el anterior)

PID_pitch_ang = Kp_pitch_ang * pitch_ang_error_prop + ITerm_pitch_ang - DPitch_ang;            // Salida PID
PID_pitch_ang = constrain(PID_pitch_ang, salidaPI_pitch_ang_min2, salidaPI_pitch_ang_max2);    // Limitar salida del PID
pitch_ang_giroscopio_anterior = angulo_pitch;                                                  // Error enterior
```

Figura 4.18. Cálculo del PID en el programa de Arduino (Fuente: Propia)

Como se aprecia en la anterior figura, el código se basa en leer el error entre la consigna y el valor real y aplicar a ese error los parámetros del PID para corregir de la forma más rápida y eficiente posible el error generado. Además, es importante que la frecuencia de salida de las PWM del Arduino sea igual que el lazo de control, ya que si las salidas del Arduino van cada 20 ms y la intención es que el lazo se ejecute cada 7 ms la información que proporcione el PID será ignorada hasta pasados esos 20 ms que tarda en ejecutarse las salidas PWM. Para ello se ejecuta el ciclo de la siguiente manera:

- Primero se empieza con las señales PWM en estado lógico alto debido a que se aplica el PID del ciclo anterior.
- Cuando estas señales pasan a estado lógico bajo se realiza la lectura del acelerómetro, del mando de RC y del PID del siguiente período.
- Posteriormente hay un tiempo de espera hasta llegar a los 7 ms que tarda en ejecutarse el lazo de control.
- Transcurrido este tiempo de espera las señales PWM vuelven a estado lógico alto.

Gracias a que se empieza el ciclo con las salidas de PWM en estado lógico alto esto provoca que éstas estén sincronizadas con el lazo de control del PID. (21)

Esta estrategia de control se realiza cuando el *Throttle* sea superior a cierto valor por seguridad, en este caso por encima de los 1300 µs, ya que hay que asegurarse de que el dron esté a cierta altura antes de que pueda realizar movimientos en los diferentes ejes.

La forma de comprobar si el lazo de control es efectivo y funciona como debería es, una vez más, gracias al *Serial Plotter* de Arduino, ya que si se gira el dron hacia una dirección se debería poder visualizar como el PID intenta compensar la inclinación a la que está sometida mediante la activación de los ESC correspondientes para llegar a 0^{o}, siempre y cuando esté en modo estable, ya que en el modo acrobático no se corrige el ángulo, solamente la velocidad de rotación.

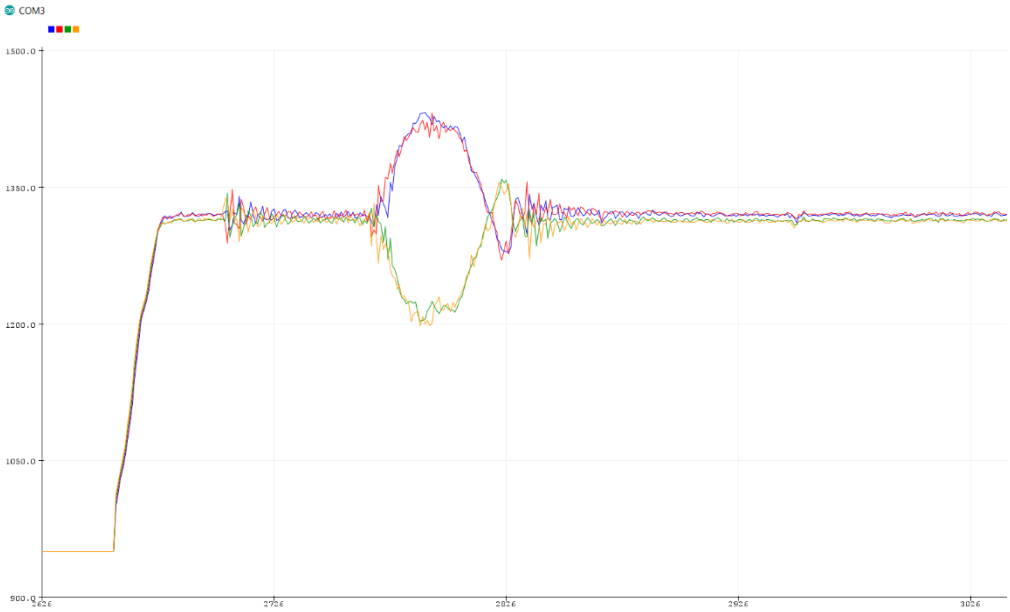

Figura 4.19. Señales en el programa de Arduino del ancho del PWM del ESC1 en azul, ESC2 en rojo, ESC3 en amarillo y ESC4 en verde sin mando de RC (Fuente: Propia)

La anterior ilustración muestra cómo se comporta el PID de control al ser inclinado el dron hacia adelante, en la dirección del eje *Pitch*, como se aprecia en la imagen los motores 1 y 2 subirían su velocidad mientras que los motores 3 y 4 la descenderían para poder compensar esa inclinación y llegar a los 0º.

También se realizan gráficas comprobando el funcionamiento del PID con respecto a las consignas que proporciona el mando de RC.

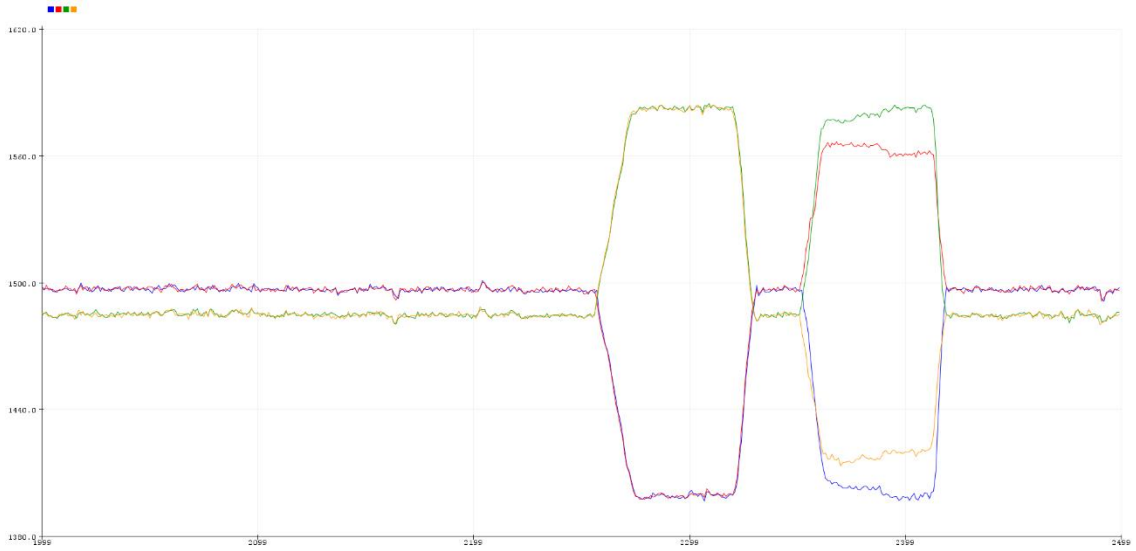

Figura 4.20. Señales en el programa de Arduino del ancho del PWM del ESC1 en azul, ESC2 en rojo, ESC3 en amarillo y ESC4 en verde con el mando de RC. (Fuente: Propia)

Como se aprecia en la imagen, también procesa correctamente las señales del PID de control, ya que, si se quisiera que el dron fuera hacia adelante en dirección del eje *Pitch*, se aumentaría la velocidad de los motores 3 y 4, mientras que se descendería el 1 y el 2, provocando que el dron se inclinase hacia adelante, forzando su movimiento.

Tras la realización de estas pruebas se dan por buenos los resultados obtenidos y ya se podría poner las hélices para que el dron empezara a volar, sin embargo, como se ha comentado con anterioridad, hasta la entrega de este proyecto no se pondrán las hélices, puesto que un movimiento inesperado o un mal funcionamiento de alguna de las partes del conjunto provocaría la pérdida del cuadricóptero, provocando que no se pudiera mostrar su funcionamiento.

4.5. Adquisición de datos

Una vez el cuadricóptero está listo para volar se realiza otra parte adyacente al proyecto y es la adquisición de datos del mismo. En esta parte del proyecto el dron ya debería ser capaz de moverse a gusto del piloto, sin embargo, es especialmente interesante usar este tipo de aeronaves para captar datos del medioambiente o del propio dron, puesto que en la actualidad las empresas valoran mucho la obtención de datos en tiempo real de sus procesos o productos.

Por lo que respecta a este proyecto, aprovechando los sensores de los que dispone el dron, se ha decidido medir los siguientes valores:

- La aceleración en [m/s^2] y velocidad angular en [$^o/s$].
- El ángulo de *Pitch* y *Roll* en [^{o}].
- El voltaje de la batería en [V].

Cabe destacar que en cuanto a adquisición de datos se refiere se pueden incluir tantos sensores como se guste y que, en el apartado de Posibles mejoras es posible encontrar otros sensores que se podrían incluir en el proyecto pero que finalmente, no han sido incluidos.

Es importante recalcar, además, que cada vez que se cargue un programa al Arduino hay que desconectar los pines RX y TX del módulo *Bluetooth*, puesto que estos pines son los que utiliza el controlador para cargar el programa a la CPU.

Para este caso se ha optado por incluir el sensor HC-06 para la recolección de datos, puesto que el cableado es sumamente sencillo, solo hay que alimentarlo a 5 V y conectar los cables RX y TX a los pines 0 y 1 del Arduino. El HC-06 es un módulo que utiliza *Bluetooth* para la comunicación entre

dispositivos, en este caso se ha optado por que el Arduino envíe los datos por *Bluetooth*, así que actuará de *Master*, y los recibirá un móvil ya que de esta forma será siempre sencillo acceder a los datos que proporciona el dron mientras está volando. (26)

Primeramente, hay que visualizar los parámetros por el puerto serie del Arduino ya que si en este paso no se reciben los datos tampoco los recibirá el móvil. Este paso basta con introducir la instrucción '*SerialPrint*' acompañado del parámetro que queremos visualizar. La forma para adquirir los datos descritos anteriormente sería como muestra la siguiente imagen:

```
void AdquisicionDatos(){

   while (micros() - tiempo_ejecucion < 5000);  // Mandar los datos cada 5ms
   tiempo_ejecucion = micros();

     if (contador == 0)Serial.print(123456789);
     if (contador == 1)Serial.print(F("|"));

   //ACELERÓMETRO + GIROSCOPIO
     if (contador == 2)Serial.print(ax / 4096);
     if (contador == 3)Serial.print(F("|"));
     if (contador == 4)Serial.print(ay / 4096);
     if (contador == 5)Serial.print(F("|"));
     if (contador == 6)Serial.print(az / 4096);
     if (contador == 7)Serial.print(F("|"));

     if (contador == 8)Serial.print((gx - gyro_X_cal) / 16.4);
     if (contador == 9)Serial.print(F("|"));
     if (contador == 10) Serial.print((gy - gyro_Y_cal) / 16.4);
     if (contador == 11) Serial.print(F("|"));
     if (contador == 12)Serial.print((gz - gyro_Z_cal) / 16.4);
     if (contador == 13)Serial.print(F("|"));

     //ÁNGULO
     if (contador == 14)Serial.print(angulo_pitch);
     if (contador == 15)Serial.print(F("|"));
     if (contador == 16)Serial.print(angulo_roll);
     if (contador == 17)Serial.print(F("|"));

     //BATERIA
     if (contador == 18)Serial.print(fVoltBat);
     if (contador == 19)Serial.print(F("|"));

    contador ++;
    if (contador == 20)contador = 0;
```

Figura 4.21. Adquisición de datos en el programa del controlador (Fuente: Propia)

Como se puede apreciar se usa un contador que pasa la información al puerto serie a cada ciclo, esto se realiza de esta manera puesto que si enviamos todas las variables a la vez al Arduino se consumiría demasiado tiempo en el envío de los datos que podría ser invertido en la ejecución del lazo de control.

Este envío de datos se inicializa con el número identificador '123456789', en la programación de la aplicación móvil se explicará por qué se necesita el envío de este número cada vez que se envían los datos, así como por que se separan los datos con '|'.

Una vez se ha comprobado que los datos se reciben correctamente hay que empezar a plantear como se desarrollará la aplicación para móvil que permita obtener estos datos. En este caso se ha optado por usar la aplicación web *MIT App Inventor* puesto que utiliza un lenguaje de programación sencillo y es ampliamente utilizado para desarrollar aplicaciones para móvil.

La programación de una aplicación con esta herramienta se basa en dos partes: la primera de ellas es la del diseño de la propia aplicación y es como se mostrará la aplicación al arrancarla desde el móvil y la segunda se basa en la programación de la misma mediante bloques. Hay que tener en cuenta que los datos que se reciban vendrán en orden según como estén colocados en el programa de Arduino y que, por tanto, primero se recibirán los datos del acelerómetro y se acabará con el voltaje de la batería.

El diseño gráfico de la aplicación se ha realizado de forma que sea lo más sencillo posible visualizar los datos, así como que sea agradable visualmente. Ha quedado tal como muestra la siguiente imagen:

Figura 4.22. Interfaz gráfica de la aplicación móvil para la adquisición de datos (Fuente: Propia)

La interfaz gráfica consta de dos botones, uno de conectar y otro de desconectar el módulo *Bluetooth*. El botón de conectar es un *List Picker*, lo que quiere decir que cada vez que se pulsa se abre una lista de los módulos de *Bluetooth* accesibles para el móvil. El resto de elementos son *Labels* o textos sobre

los cuales se escribirán los datos que se reciben del controlador o simplemente identificadores de las variables que se están recibiendo.

La siguiente parte se basa en la programación de la misma, esta a su vez, se divide en dos grandes bloques: lo primero que hay que hacer es establecer comunicación con el móvil, para ello es necesario recibir un *feedback* de que la comunicación entre ambos dispositivos se ha realizado correctamente, posteriormente habrá que enviar los datos que anteriormente se mencionaban por el puerto serie del Arduino a la propia aplicación.

Para inicializar las variables una vez se ha ejecutado la aplicación se programa de la siguiente manera:

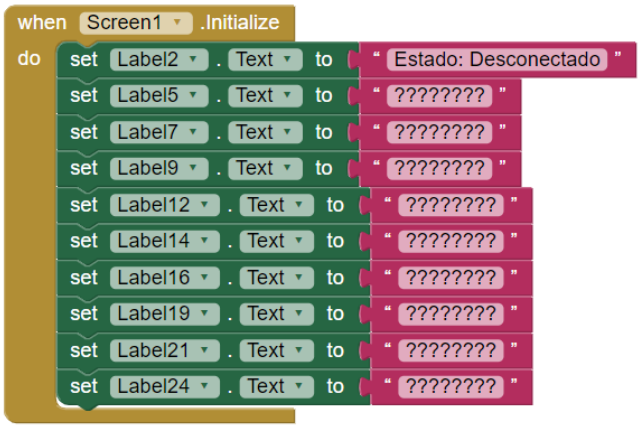

Figura 4.23. Programación en bloques de la inicialización de las variables (Fuente: Propia)

De esa forma se puede apreciar si, al arrancar la aplicación, la conexión entre el móvil y el sensor *Bluetooth* no se ha establecido correctamente y si fuera el caso muestra las variables con interrogantes.

El siguiente paso es la programación de cómo escoger el módulo *Bluetooth* con el que se quiere establecer comunicación, además si el cliente de *Bluetooth* está conectado se recibirá un *feedback* en formato de texto para saber si se ha podido establecer comunicación. También es crucial la creación de una variable que será la que recibe los datos del control y una lista vacía para que se pueda establecer a que variable corresponde el dato que se ha recibido del control.

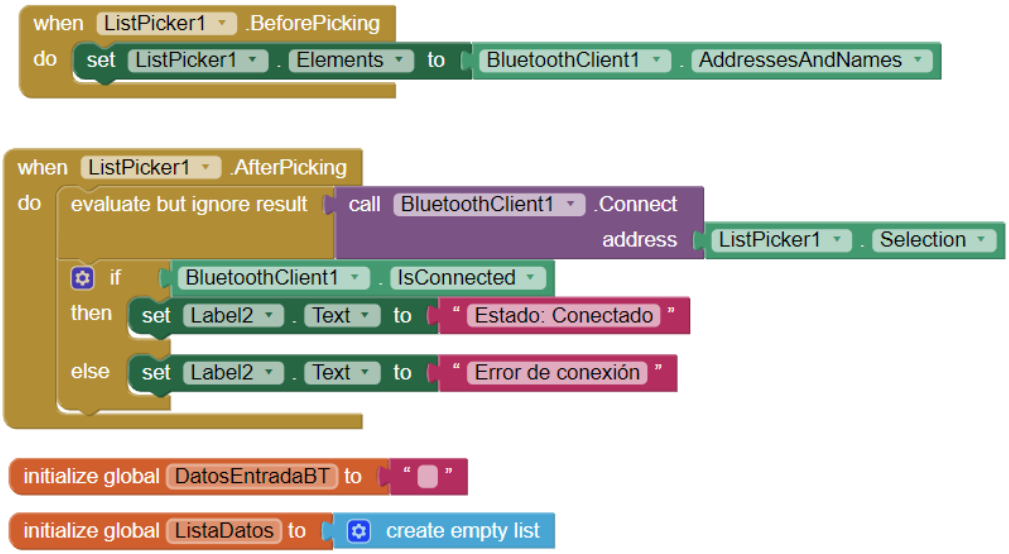

Figura 4.24. Programación en bloques de la conexión con *Bluetooth* entre el módulo y el móvil (Fuente: Propia)

Como se ha comentado anteriormente, el controlador utiliza un puntero que envía un dato a cada ciclo y eso dificulta la adquisición de datos, puesto que no se puede controlar con facilidad a que variable corresponde el dato que se recibe, ya que la aplicación está pensada para que si recibe algún dato se establece que esos datos van separados por '|' y a su vez se determina un índice en la lista creada anteriormente en la que se asocia el dato a la variable. Este sistema funcionaría si se enviarán todos los datos a la vez, puesto que cuando recibe un dato recibiría todo el conjunto y se pueden indexar cómodamente, sin embargo, al enviar un dato a cada ciclo el sistema entiende que se reciben todos a pesar de que realmente solo se reciba uno, como consecuencia de eso los datos van cambiando de índice provocando que no concuerden con sus variables.

La solución que se ha optado para solucionar este problema es la de crear una variable identificativa '123456789' que se envía primero por el monitor serie. Gracias a el envío de este variable se puede comparar cuando le corresponde a ésta el primer índice en la lista de datos y de esta forma se verifica que si el dato que llega primero es ese se puede deducir que las demás variables tienen el orden que deberían. La consecuencia de ello es que los datos se refrescan más lentos, pero sin embargo esos datos recibidos corresponden a las variables del programa. Cabe destacar además que se verifica si se reciben los datos cada vez que se ejecuta el tiempo de *Clock* que en este caso es de un segundo.

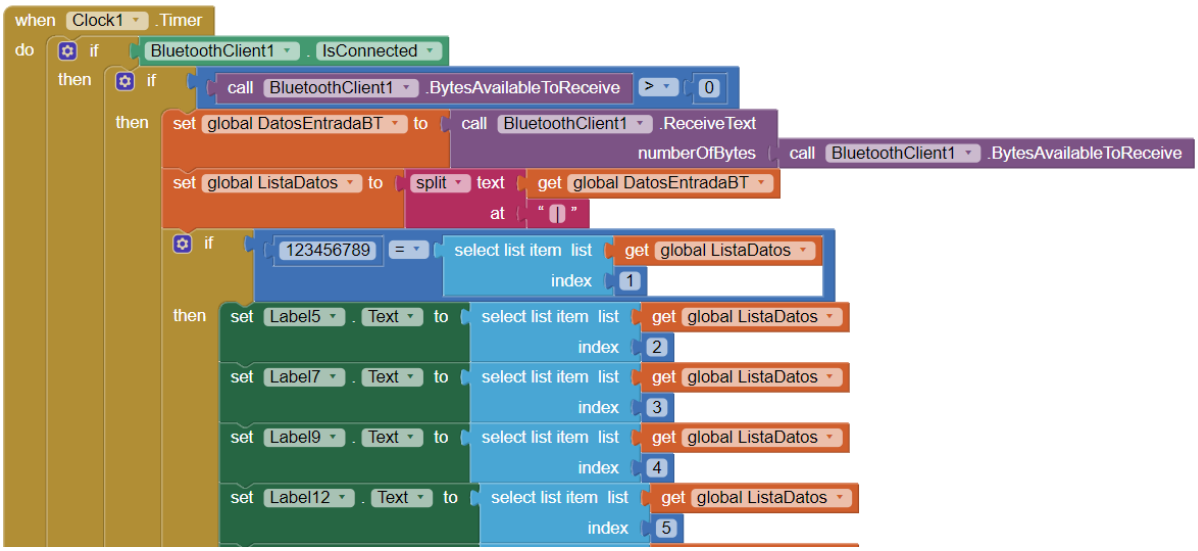

Figura 4.25. Programación en bloques de la adquisición de datos (Fuente: Propia)

En esto se basa la programación de la adquisición de datos, como se puede apreciar la programación en bloques del programa *MIT App Inventor* facilita la programación y comprensión del código de la aplicación móvil, así como crear una interfaz gráfica sencilla de forma fácil. En el caso de que se quiera hacer un cambio en el código del Arduino hay que tener especial cuidado de que el módulo HC-06 no esté alimentado, puesto que este sensor utiliza los pines 0 y 1 del Arduino para su comunicación y esos son los pines que utiliza el controlador para verter el programa a la CPU.

5. Normativa

En España hay una legislación que afecta directamente al uso de drones tanto si se usa de forma lúdica como profesional, con respecto a la realización de este proyecto el dron se usaría por ocio, y, por tanto, está sujeta a una serie de normas recogidas por el Boletín Oficial del Estado (BOE), algunas de las normas son las siguientes:

- Volar a una distancia mínima de 8 km de cualquier aeropuerto o aeródromo.
- Volar fuera del espacio aéreo controlado.
- No sobrepasar los 120 metros de altura sobre el suelo o sobre el obstáculo más alto situado dentro de un radio de 150 metros desde la aeronave.
- Volar de día y en buenas condiciones meteorológicas. Aquí hay que destacar que si la aeronave pesa menos de 2 kilogramos están permitidos los vuelos nocturnos siempre que no se superen los 50 metros de altura.
- Los vuelos siempre serán dentro del alcance visual del piloto.
- Las aeronaves de menos de 250 gramos podrán volar en ciudad y sobre aglomeraciones de personas y edificios siempre y cuando no se superen los 20 metros de altura.
- Es recomendable contar con un seguro de responsabilidad civil.
- En el caso de que el dron disponga de cámara se ha de proteger el derecho a la intimidad de los individuos que pudieran aparecer en las imágenes captadas por el dron para no vulnerar la Ley de Protección de Datos. (27)

Todo este marco normativo es vigente, ya que anteriormente en la antigua regularización de los drones en octubre de 2014 no estaba permitido el uso de drones en ciudad ni los vuelos nocturnos.

Cabe destacar que la normativa de drones que rige actualmente en España no exige ningún tipo de licencia para vuelos de drones básicos para uso recreativo o hobby, sin embargo, quienes usan drones a nivel profesional están obligados a tener una licencia certificada legalmente en España.

Existen áreas que no tienen ningún tipo de restricción para volar drones, sin embargo, hay otras en la que el uso de estos equipos está sujeto a permisos y condiciones. Y, por último, hay otras zonas en la que está totalmente prohibido su uso. Algunas de estas zonas serían por ejemplo aeropuertos, bases navales o zonas de control militar. (28)

Actualmente la nueva normativa de drones de la Agencia Estatal de Seguridad Aérea (AESA) enmarca que, en los vuelos de drones de uso recreativo, solo es necesario tener los conocimientos para pilotar la aeronave con seguridad siempre y cuando el dron no pese más de 2 kilogramos.

En el caso de que se quiera volar en zonas urbanas se han de cumplir otros requisitos:

- Se ha de mantener una distancia máxima de 100 metros hasta el dron y una altura de 120 metros sobre el edificio más alto en un radio de 600 metros.
- La aeronave no debe superar los 10 kilogramos de peso.
- El vuelo siempre se realizará dentro del alcance visual del piloto.
- Se recomienda un margen de 50 metros horizontales entre el dron y cualquier edificación.
- El dron debe de estar provisto de algún sistema de amortiguador de caídas.

Como se puede apreciar la normativa en referencia al vuelo de drones es bastante restrictiva, sin embargo, hay zonas muy amplias en la que es posible volar el dron sin tener que pedir permiso a la AESA siempre que sean vuelos diurnos y se cumplan las distancias máximas establecidas. En la actualidad existe la web ENAIRE Drones que pertenece al Ministerio de Fomento donde aparece toda la información necesaria para volar un dron con seguridad, además de aparecer un mapa indicativo donde se puede localizar donde hacer volar el dron.

En el futuro se prevé que en julio de 2020 se comenzará a aplicar un nuevo reglamento que pretende homogeneizar el marco europeo a nivel de licencias, normas y categorías operacionales. Seguramente, la norma más llamativa con referente a los drones que se aplicará en esta nueva normativa es la implementación de tres nuevas categorías para establecer las restricciones a la hora de volar, que son:

- Categoría abierta, donde se englobarían los vuelos de bajo riesgo operados por pilotos aficionados.
- Categoría específica, para las operaciones de riesgo medio que requieren de autorización.
- Categoría certificada, reservada para los vuelos de alto riesgo, para los que el piloto deberá estar certificado como operador. (29)

6. Análisis del impacto ambiental

Durante el desarrollo del proyecto se ha procurado que los componentes seleccionados cumplan la normativa RoHS, esta norma restringe ciertas sustancias peligrosas en aparatos eléctricos y electrónicos. Además, se ha comprobado si éstos componentes poseen el marcado de la conformidad europea CE, y, por tanto, el producto cumple todos los requisitos de seguridad, sanidad y protección del medio ambiente exigidos por la UE. (30)

En referencia al uso de drones en general y su impacto en el medio ambiente, es cierto que cuando estas aeronaves no tripuladas atraviesan espacios naturales pueden generar nerviosismo para los animales que viven en su hábitat natural.

Un equipo de investigadores de la universidad de Montpellier, en Francia, realizó cerca de 200 vuelos en un área costera y en el 80% de los casos no se observó ninguna incidencia en el comportamiento de especies como el flamenco común o el archibebe claro, llegando a acercar el dron a una distancia de 4 metros del animal.

Figura 6.1. Fotografía de un dron sobrevolando un espacio natural (Fuente: (24))

Parece que el color, la velocidad y la frecuencia de los vuelos no afecta al nerviosismo de estas especies, sin embargo, sí que lo hace la inclinación del mismo ya que, prácticamente en todos los ensayos cuando el dron se sitúa a 90º con respecto al suelo parece que se sienten amenazados.

Como conclusión, parece ser que siempre y cuando se respete una distancia de 4 metros y no se sobrevuele a estas especies no parece que afecte al comportamiento de flamencos y archibebes

(especies muy sensibles a la invasión de su hábitat), sin embargo, si una de estas dos condiciones no se cumple pueden llegar a confundir a los drones con un depredador.

No obstante, el dron también puede llegar a ser positivo para el medioambiente, por ejemplo, pueden ser una herramienta muy importante en cualquier programa de vigilancia ambiental, en este proyecto, en el apartado de Posibles mejoras se baraja la opción de incluir un sensor que mida la calidad del aire, estos tipos de drones se suelen llamar *Ecodrones*. Además, el propio diseño de estas aeronaves suele ser no contaminantes y tienen infinidad de tareas, como:

- Control de los territorios: Pueden realizar fotografías para apreciar cómo evoluciona un terreno a lo largo del tiempo.
- Prevención y control de incendios: Podrían identificar un fuego temprano con un *software* específico que les permite identificar los colores del humo y avisar a los equipos de emergencia, así como a almacenar grandes cantidades de agua para luchar contra el incendio o identificar a personas cuando ocurre una catástrofe de esta índole.
- Control hídrico: Posibilidad de identificar vertidos ilegales o para estudiar cuencas hidrográficas.
- Vigilancia de reservas naturales: Permite sobrevolar un hábitat y controlar la aparición de cazadores furtivos, además de monitorizar a ciertas especies para estudiar su comportamiento y tratar de mantener la biodiversidad de la zona. (32)

Figura 6.2. *Ecodron* luchando contra un incendio esparciendo agua (Fuente: (31))

7. Diagrama de Gantt

El diagrama de Gantt sirve para distribuir las tareas que se han de realizar en el proyecto a lo largo del tiempo. Es una herramienta muy útil para planificar proyectos, puesto que proporciona una vista general de las tareas programadas.

Un diagrama de Gantt muestra lo siguiente:

- La fecha de inicio y finalización de un proyecto.
- De que tareas se compone un proyecto.
- Quien trabaja en cada tarea.
- La fecha programada de inicio y finalización de las tareas.
- Una estimación de cuanto llevará cada tarea.
- Como se superponen las tareas y si hay una relación entre ellas. (33)

Con respecto a este proyecto, el diagrama de Gantt es interesante para apreciar cómo se han distribuido las tareas, cuánto se ha dedicado a cada una y una fecha estimada de cuanto se acabará el proyecto.

Cabe destacar que esto se realiza al inicio del proyecto y que, por tanto, las fechas reales de finalización de cada tarea pueden acabar más tarde o más temprano, lo que provocaría que el diagrama de Gantt se alargase o se acortase.

Así pues, las actividades realizadas en el proyecto se distribuyen de la siguiente manera:

NOMBRE ACTIVIDAD	ACTIVIDADES
ACTIVIDAD A	Planificación del Proyecto
ACTIVIDAD B	Búsqueda de información acerca de los drones
ACTIVIDAD C	Redacción de la Introducción en la memoria
ACTIVIDAD D	Búsqueda y compra de los componentes
ACTIVIDAD E	Ensamblaje de los componentes mecánicos del dron
ACTIVIDAD F	Comprobación de la estabilidad del diseño mecánico
ACTIVIDAD G	Diseño de la estructura que irá fijada al dron
ACTIVIDAD H	Búsqueda y fijación de los componentes de la estructura mecánica del dron
ACTIVIDAD I	Diseño del esquema eléctrico del conjunto
ACTIVIDAD J	Soldadura de los componentes en las placas
ACTIVIDAD K	Comprobación de soldaduras
ACTIVIDAD L	Fijación de las placas y componentes en la estructura mecánica del dron
ACTIVIDAD M	Redacción del diseño mecánico y eléctrico adoptados para el proyecto
ACTIVIDAD N	Búsqueda de información con respecto a la programación del Arduino

ACTIVIDAD O	Programación y comprobación del correcto funcionamiento de todos los sensores
ACTIVIDAD P	Programación del proyecto final que irá integrado en el Arduino
ACTIVIDAD Q	Comprobación y corrección de errores
ACTIVIDAD R	Redacción de la programación en la memoria
ACTIVIDAD S	Programación de la aplicación de adquisición de datos
ACTIVIDAD T	Redacción de la adquisición de datos
ACTIVIDAD U	Redacción de toda la memoria
ACTIVIDAD V	Comprobación final y posterior revisión del Director del proyecto
ACTIVIDAD W	Preparación de la exposición
ACTIVIDAD X	Supervisión final de todo el conjunto

Tabla 2.- Tabla de distribución de las actividades del Diagrama de Gantt (Fuente: Propia)

Una vez se han definido las tareas hay que dar una fecha estimada de cuando se acabarán y cuales se superponen y pueden realizarse conjuntamente. En este caso, como se aprecia en el diagrama de Gantt que se adjunta a continuación, muy pocas tareas se superponen con otras, solamente la soldadura y comprobación de éstas, y la programación y pruebas del conjunto. Esto es debido a que la gran mayoría de tareas requiere de las anteriores y, por tanto, solo es posible realizar a la vez las tareas mencionadas anteriormente, puesto que hacer las pruebas al final de estas tareas supondría que, en el caso de que hubiera algún fallo, invertir una gran cantidad de tiempo en arreglarlo y es mucho más sencillo ir comprobándolo a cada paso.

Además, el diagrama de Gantt especifica una fecha de finalización del 22 de mayo de 2020, por tanto, entraría dentro del plazo de entrega del proyecto.

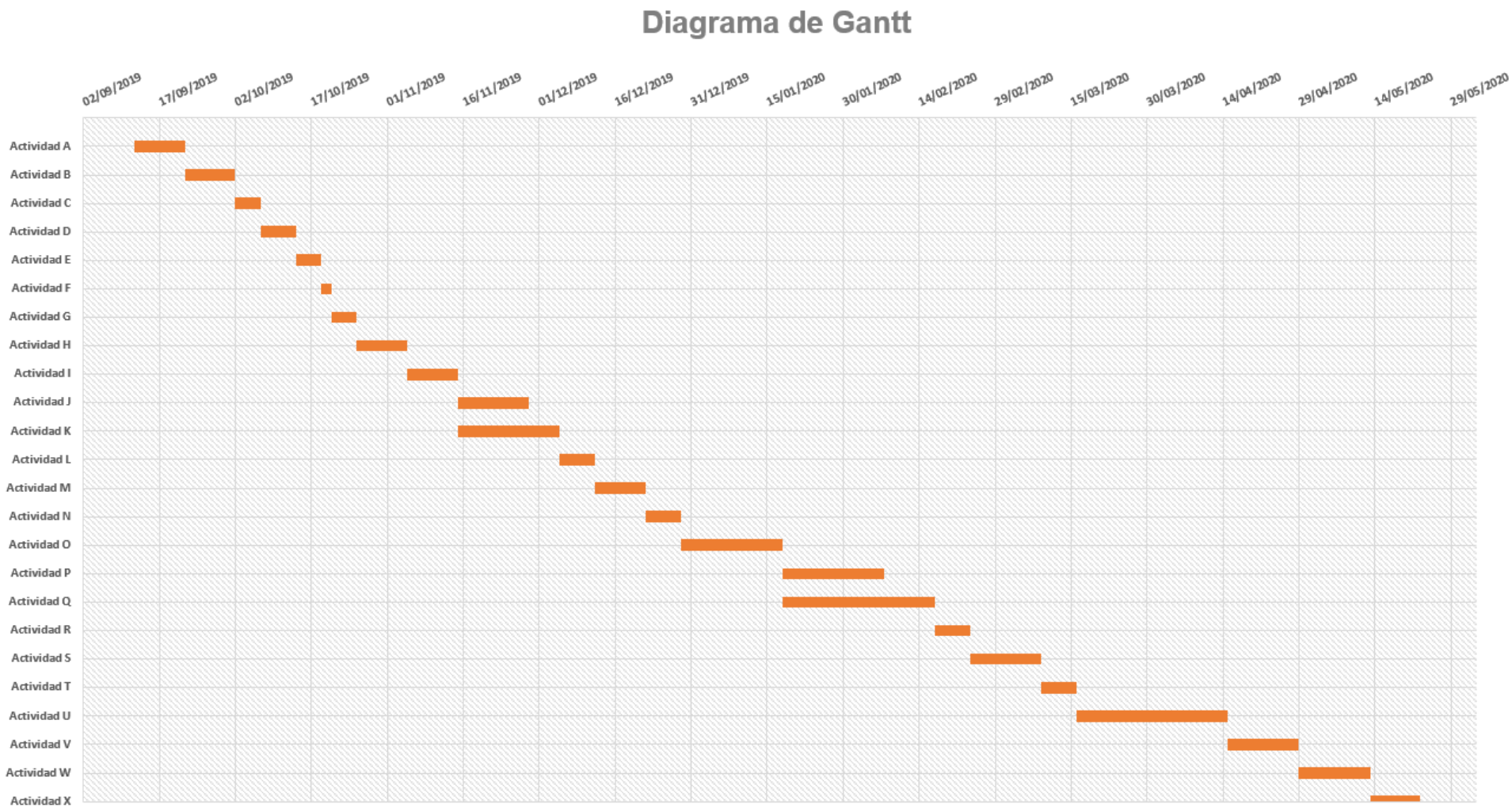

Figura 7.1. Diagrama de Gantt del proyecto (Fuente: Propia)

Conclusiones

Para concluir, la realización de este proyecto ha sido muy enriquecedora, ya que ha empezado desde la planificación inicial del trabajo pasando por tareas como el diseño del *hardware* o la programación hasta llegar a la redacción de este documento.

Durante estas fases, que están descritas en el Diagrama de Gantt se han puesto a prueba y se han ampliado los conocimientos adquiridos a lo largo de la carrera de Ingeniería Electrónica, Industrial y Automática que se imparte en la Universidad Politécnica de Barcelona.

Al ser un trabajo que requiere de la manipulación de los elementos que componen el dron, se ha podido comprobar todas las tareas de las que se requiere en el proceso de creación de un prototipo, así como lo costoso que es la corrección de los problemas que surgen durante el mismo. Estos problemas han podido ser resueltos gracias a la supervisión del proyecto y a las fuentes de información ilimitadas que supone el uso de Internet, dichas fuentes se nombran en la bibliografía.

La creación desde cero de un cuadricóptero es algo que es mucho más costoso de lo que parece, puesto que se ha de ir comprobando si los pasos anteriores se han realizado correctamente, como la soldadura de los componentes, la distribución de los mismos para que el peso esté equilibrado y la optimización del código de Arduino para que el lazo de control de PID se pueda ejecutar de forma rápida y fiable.

La realización de este proyecto no obstante no necesariamente termina aquí, como se comenta en el apartado de Posibles mejoras, se pueden incluir tantos sensores como se quiera siempre y cuando se introduzcan en los códigos de Arduino y de la aplicación móvil. Además, al utilizar en ambos casos un *software* de código abierto cualquiera puede realizar las modificaciones que crea oportunas.

Para finalizar, la realización de este proyecto ha servido para ahondar y comprender mejor la electrónica en general. El diseño y realización de un dron y una plataforma de adquisición de datos que le complementa ha supuesto un gran reto, los problemas que han surgido han sido solucionados con solvencia y se han aprendido nuevos métodos tanto de diseño electrónico como de programación.

Posibles mejoras

Una serie de mejoras que pudieran hacer del proyecto uno más completo pero que, por motivos de tiempo, no se han podido implementar se expondrán en este apartado.

Una de ellas es la inclusión de más sensores para realizar una captura de datos más completa, se barajaron varios sensores como el DHT11 que mide la temperatura y la humedad o el MQ-135, un sensor capaz de detectar la contaminación en el ambiente. A partir del punto en el que se ha dejado el proyecto, no supone demasiado coste la inclusión de más sensores, ya que, como se usa un Arduino, se pueden enviar los datos por el puerto serie de forma sencilla, tanto si los sensores usan el protocolo de comunicación I2C como si no. La implementación de más sensores sencillamente se basaría en la soldadura de estos chips en la placa de los sensores y una adaptación del código que va integrado en el Arduino. (34)

Figura 0.1. Sensor MQ-135 que mide la calidad del aire (Fuente: (27))

Otra posible solución a implementar en el proyecto es la de tapar el dron para que, en el caso de que llueva, no se estropearan los componentes electrónicos. Todo y que la normativa específica que no hay que usar el dron cuando las condiciones meteorológicas son adversas, implementar una especie de tapa, con agujeros laterales para que los componentes no se calienten en exceso sería lo ideal, aunque requeriría de más tiempo, puesto que lo normal sería realizar un modelo 3D para el dron en específico. El material seria fibra de carbono, como el chasis, puesto que es un material ligero, relativamente barato y tiene el grado de protección 'IP' suficiente para que resista a la lluvia.

Bibliografía

1. Fernández Bobadilla, H.A., Landín Torres, I.J. i Ramírez Carmona, U. *Diseño, construcción y control de una aeronave tipo dron* [en línia]. UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO, 2016. Disponible a: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/10525/TesisDron.pdf?sequence=1.

2. Santana, E. Tipos de Drones - Clasificación de Drones. A: [en línia]. 2017. Disponible a: https://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/.

3. Anónimo. Principios básicos de vuelo. A: [en línia]. Disponible a: https://vueloartificial.com/introduccion/toma-de-contacto/principios-basicos-de-vuelo/.

4. Anónimo. ¿Como elegir el mejor transmisor RC para tu drone? A: [en línia]. 2018. Disponible a: https://www.midronedecarreras.com/tutoriales/emisoras-y-transmisor-rc/.

5. Chacón, P. ¿Como colocar las hélices correctamente a tu drone? A: [en línia]. Disponible a: https://blog.juguetronica.com/como-colocar-las-helices-correctamente-a-tu-drone/.

6. Motores Brushless para drones, todo lo que necesitas saber. A: [en línia]. Disponible a: https://mobus.es/blog/motores-brushless-para-drones-todo-lo-que-necesitas-saber/.

7. Anónimo. LO QUE HAY QUE SABER PARA ELEGIR EL CHASIS PARA UN CUADRACÓPTERO. A: [en línia]. Disponible a: https://www.prometec.net/elegir-chasis-dron/.

8. Anónimo. LO QUE HAY QUE SABER PARA ELEGIR LOS ESCS PARA UN CUADRACÓPTERO. A: [en línia]. Disponible a: https://www.prometec.net/esc-para-drones/.

9. Anónimo. LO QUE HAY QUE SABER PARA ELEGIR LAS HÉLICES PARA UN CUADRACÓPTERO. A: [en línia]. Disponible a: https://www.prometec.net/elegir-helices-dron/.

10. Guerrero, J. Arduino Uno: Especificaciones y características. A: [en línia]. 2014. Disponible a: https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/.

11. Aqueel, A. Introduction to Arduino IDE. A: [en línia]. 2018. Disponible a: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html.

12. Llamas, L. CONECTAR UNA EMISORA RADIO CONTROL CON ARDUINO. A: [en línia]. 2016, Disponible a: https://www.luisllamas.es/conectar-emisora-radio-control-con-arduino/.

13. Anónimo. LO QUE HAY QUE SABER PARA ELEGIR UNA BATERÍA LIPO. A: [en línia]. Disponible a: https://www.prometec.net/elegir-bateria-lipo/.

14. Llamas, L. CONECTAR UN DISPLAY LCD HITACHI A ARDUINO POR BUS I2C. A: *2016* [en línia]. Disponible a: https://www.luisllamas.es/arduino-lcd-i2c/.

15. Latorre, E. MPU-6050 GY-521 acelerómetro y giroscopio de 6 ejes. A: [en línia]. 2017. Disponible a:

http://enrique.latorres.org/es_ES/2017/10/19/mpu-6050-gy-521-6-axis-accelerometer-gyroscope/.

16. Anónimo. MÓDULO BLUETOOTH HC-06. A: [en línia]. Disponible a: https://www.prometec.net/bt-hc06/.

17. Anónimo. Descripción y funcionamiento del Bus I2C. A: [en línia]. 2018. Disponible a: http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/.

18. García, V. Introducción al I2C Bus. A: [en línia]. 2012. Disponible a: https://www.diarioelectronicohoy.com/blog/introduccion-al-i2c-bus.

19. Hernández, R. *Introducción a los sistemas de control* [en línia]. Pearson, 2010. Disponible a: http://lcr.uns.edu.ar/fcr/images/Introduccion a Los Sistemas de Control.pdf.

20. Pardo, C. Controlador PID. A: [en línia]. 2013. Disponible a: https://www.picuino.com/es/arduprog/control-pid.html.

21. Anónimo. Control de estabilidad y PID para drones. A: [en línia]. 2018. Disponible a: https://arduproject.es/control-de-estabilidad-y-pid/.

22. Anónimo. Tutorial MPU6050, Acelerómetro y Giroscopio. A: [en línia]. Disponible a: https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Acelerómetro-y-Giroscopio.html.

23. Anónimo. USANDO EL MPU6050. A: [en línia]. Disponible a: https://www.prometec.net/usando-el-mpu6050/.

24. InvenSense. MPU-6000 Datasheet. A: [en línia]. 2013. Disponible a: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

25. Anónimo. Tutorial de Arduino y MPU-6050. A: [en línia]. 2015. Disponible a: http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/.

26. Anónimo. Tutorial de Arduino, Bluetooth y Android #2 – Crear una app con MIT inventor. A: [en línia]. 2015. Disponible a: https://robologs.net/2015/10/29/tutorial-de-arduino-bluetooth-y-android-2-crear-una-app-con-mit-inventor/.

27. Anónimo. Normativa sobre drones en España [2019]. A: [en línia]. 2019. Disponible a: https://www.aerial-insights.co/blog/normativa-drones-espana/.

28. Anónimo. NUEVA LEY DE DRONES EN ESPAÑA 2020. A: [en línia]. 2020. Disponible a: https://www.tierradedrones.com/nueva-ley-de-drones-en-espana/#la_nueva_ley_de_drones_en_espana_2019.

29. Anónimo. NORMATIVA DE DRONES EN ESPAÑA 2020. A: [en línia]. 2020. Disponible a: https://www.oneair.es/normativa-drones-espana-aesa/.

30. Anónimo. Marcado CE. A: [en línia]. 2020. Disponible a: https://europa.eu/youreurope/business/product-requirements/labels-markings/ce-marking/index_es.htm.

31. Anónimo. DRONES, LOS NUEVOS ALIADOS PARA LOS BOMBEROS. A: [en línia]. Disponible a: https://grupodeincendios.com/drones-los-nuevos-aliados-los-bomberos/.

32. Anónimo. EL DRON, UN ALIADO PARA EL MEDIO AMBIENTE. A: [en línia]. 2017. Disponible a: https://www.cerem.es/blog/el-dron-un-aliado-para-el-medio-ambiente.

33. Villanueva, C. ¿Qué es y para qué sirve un diagrama de Gantt? A: [en línia]. 2018, Disponible a: https://blog.teamleader.es/diagrama-de-gantt.

34. Anónimo. Tutorial sensores de gas MQ2, MQ3, MQ7 y MQ135. A: [en línia]. 2016. Disponible a: https://naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html.