UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

Grado en Ingeniería en Tecnologías Industriales

DISEÑO DE UN VEHÍCULO AÉREO CUADRIROTOR DE BAJO COSTE CON PLATAFORMA RASPBERRY PI

AUTOR: Alejandro Javier Torres Arroyo

TUTOR: José Luis Navarro Herrero

Curso Académico: 2020-2021

AGRADECIMIENTOS

Quisiera empezar agradeciendo el esfuerzo, la atención y dedicación de mi tutor José Luis Navarro Herrero. Por responder a mis dudas y preguntas, y por transmitirme sus valiosos conocimientos.

Quisiera agradecer a mi familia todo el apoyo recibido, en especial, a mis padres por darme la oportunidad de estudiar una carrera; a mi padre, Roque, por transmitirme sus conocimientos técnicos y pasión por la tecnología; a mi madre, Marta, por enseñarme el valor del trabajo y del esfuerzo; y a mi hermano, Guille, por mostrarme su amor y apoyo incondicional.

También quisiera agradecer a mis amigos su interés y apoyo constante, en especial, a mis compañeros de piso, por permitirme realizar pruebas en casa y ayudarme en ellas; y a mi amigo Salva por ayudarme con su conocimiento del lenguaje.

Por último, me gustaría gradecer a la UPV y a todos sus profesores y demás trabajadores la encomiable labor que realizan en el ámbito educativo.

RESUMEN

Este proyecto consiste en diseñar y fabricar un vehículo aéreo cuadrirotor haciendo uso de los conocimientos adquiridos a lo largo de la carrera, más concretamente los aprendidos en las asignaturas de electrónica, mecánica, informática y, sobre todo, automatización y control, ya que uno de los objetivos primarios de este proyecto es dotar de estabilidad al dron gracias al uso de un controlador PID.

Para lograr esto, se han seguido los siguientes pasos, siempre con la idea de crear un producto económico, versátil y practico. Primero, se realizó un exhaustivo estudio del mercado y de los drones existentes, posteriormente se esbozó un primer diseño de los componentes del dron y del software con el que este funcionaría, programado en Python. Una vez hechos los cálculos y elegidos los componentes, se procedió a la adquisición de estos, para más tarde practicar el montaje del cuadricóptero. Para finalizar, se hicieron pruebas de vuelo en un espacio controlado y bajo supervisión para proceder a los ajustes pertinentes.

RESUM

Este projecte consistix a dissenyar i fabricar un vehicle aeri cuadrirotor fent ús dels coneixements adquirits al llarg de la carrera, més concretament els apresos en les assignatures d'electrònica, mecànica, informàtica i, sobretot, automatització i control, ja que un dels objectius primaris d'este projecte és dotar d'estabilitat al dron gràcies a l'ús d'un controlador PID.

Per a aconseguir açò, s'han seguit els següents passos, sempre amb la idea de crear un producte econòmic, versàtil i practique. Primer, es va realitzar un exhaustiu estudi del mercat i dels drones existents, posteriorment es va esbossar un primer disseny dels components del dron i del programari amb què aquest funcionària, programat en Python. Una vegada fets els càlculs i triats els components, es va procedir a l'adquisició d'estos, per a més tard practicar el muntatge del cuadricóptero. Per a finalitzar, es van fer proves de vol en un espai controlat i baix supervisió per a realitzar les modificacións necesaries.

ABSTRACT

This project consists of designing and manufacturing a quadrotor aerial vehicle, making use of the knowledge acquired throughout the degree course, more specifically that learned in the subjects of electronics, mechanics, computer science and, above all, automation and control, as one of the primary objectives of this project is to provide the drone with stability thanks to the use of a PID controller.

To achieve this, the following steps have been followed, always with the idea of creating an economical, versatile and practical product. First, an exhaustive study of the market and existing drones was carried out, then a first design of the drone's components and the software with which it would work was sketched, programmed in Python. Once the calculations had been made and the components chosen, they were purchased, and then the quadcopter was assembled. Finally, flight tests were carried out in a controlled space and under supervision in order to make the necessary adjustments.

ÍNDICE

1. INTRODUCCIÓN

1.1. Motivación

La evolución de la tecnología implica muchos sectores, el sector de la aeronáutica es uno de ellos, los drones viven esta evolución de cerca. Las grandes marcas de drones, como DJI, han conseguido crear drones autónomos, con la ayuda de la inteligencia artificial, pero con un coste, a veces, excesivo. Estos drones reúnen muchas materias y sectores de la tecnología: tecnología de materiales, ya que hay que utilizar los materiales más resistentes y ligeros para la estructura o las hélices; dinámica de fluidos, para analizar la aerodinámica del dron; electrónica, para el uso de los sensores y la controladora de vuelo; mecánica para la dinámica del cuadricóptero; informática, para la programación de los drones; automática y control, para el controlador PID encargado de la estabilidad; o energía, para encontrar las mejores baterías. Todas las materias nombradas, han sido estudiadas en el grado y ponerlas en práctica es un gran reto y motivación para un estudiante del grado como yo. La posibilidad de hacer esto con un coste reducido y asequible es otra de las principales motivaciones de este proyecto.

1.2. Objetivos

El alcance de este proyecto contiene todo lo referido a la construcción de un cuadricóptero de bajo coste desde el diseño del dron hasta su construcción y la implementación del software. Ahora bien, para lograr esto será necesaria la consecución de diversos objetivos específicos. Entre los cuales destacan los siguientes:

- Investigar el mercado de los drones y su normativa
- Mantener el bajo coste
- Seleccionar la controladora de vuelo
- Seleccionar los motores y controladores electrónicos de velocidad
- Integrar la Raspberry-Pi y el sistema operativo
- Crear el programa para la Raspberry-Pi en lenguaje Python 3
- Diseñar e implementar el algoritmo del controlador PID
- Lograr la comunicación de la Raspberry-Pi con los diferentes sensores
- Realizar pruebas de vuelo para constatar el correcto funcionamiento de los componentes y su programación

Se considerará alcanzado el objetivo global siempre y cuando se hayan ido satisfaciendo los objetivos específicos durante el transcurso del proyecto.

2. ESTADO DEL ARTE

2.1. Historia de los drones y cuadricópteros

El uso de drones, como forma de vehículo aéreo no tripulado, se remonta hasta 1849, en Austria. Los austriacos fueron pioneros en este sector, con fines bélicos. La ciudad de Venecia, que había sido cedida a Austria por Napoleón Bonaparte en 1797, decidió rebelarse y declararse independiente. Ante esta revuelta, los austriacos decidieron asediar la ciudad, cortando todos los suministros a esta. Franz Von Uchatius, un militar austriaco, propuso bombardear la ciudad desde el aire, de forma remota. Y así fue, desplegaron una flota de globos no tripulados, cargados de bombas, que sobrevolarían la ciudad y estallarían, soltando las bombas que destruirían la ciudad. Esta misión fue un fracaso, ya que los globos volaban de forma incontrolada, y muchos de ellos fueron arrastrados por el viento de vuelta a los austriacos. A pesar de este descalabro, se demostró el potencial que podían tener los vehículos aéreos no tripulados.(Arena, 2015)

No es hasta 1907 cuando los hermanos Louis y Jacques Bréguet inventan el primer cuadricóptero. Aunque este primer acercamiento tenía la gran deficiencia de tener que ser tripulado por 4 personas, una por cada motor, por lo que no podría ser considerado como un dron. El vehículo de los hermanos llego a levantarse un par de pies del suelo. Un invento que claramente sirvió de base para los siguientes trabajos realizados en este sector de la aeronáutica.(Oliver, 2018)

Unos años más tarde, en 1917, se realizó el primer vuelo de un avión no tripulado, otra vez con intenciones bélicas. Este avión podía ser controlado remotamente gracias a la novedosa tecnología de radio control, inventada 20 años antes por Nikola tesla. Este avión fue concebido con la idea de ser una bomba teledirigida, pero finalmente no fue utilizado en ningún combate. Pero de nuevo, este avión sirvió de cimiento para los futuros drones militares.(Oliver, 2018)

En 1920 por fin un cuadricóptero consiguió volar de forma prolongada, de hecho, batió el récord de distancia recorrida por ningún helicóptero hasta la época, recorriendo 360 metros. Se trata del cuadricóptero inventado por Étienne Oehmichen. Aunque este vehículo es considerado como un cuadricóptero, realmente poseía ocho hélices: cinco para mantener el vehículo estable, dos para impulsarse hacia adelante y hacia atrás y una última para cambiar de dirección. Las 8 hélices eran impulsadas por un motor central de 120 caballos. Esta invención fue el impulso definitivo para seguir desarrollando los cuadricópteros hasta lo que son hoy en día. (Krossblade, s. f.)

En los años venideros los drones evolucionaron de forma gigantesca, tanto para uso civil como para uso militar, hasta los drones que conocemos hoy en día. Actualmente los drones están muy integrados en nuestra sociedad, aparte del uso recreativo, se usan en múltiples sectores profesionales, desde la agricultura hasta el salvamento de personas en emergencias.

En la agricultura son ampliamente usados, sobre todo para el control de cultivo, los drones pueden recorrer distancias muy grandes, de forma muy rápida, recolectando información en el proceso gracias a sus sensores. Pueden recolectar información sobre la hidratación o temperatura del cultivo, incluso detectar plagas y enfermedades de forma prematura, aumentando la eficacia de la plantación. Ahora mismo hay múltiples empresas en el mercado que lo hacen por una muy buena relación eficacia/precio.(Bejerano, s. f.)

En el sector no civil también son muy usados, por ejemplo, para el rescate de personas, muchos equipos de emergencias disponen de drones para llegar hasta zonas de difícil acceso, pudiendo localizar personas desaparecidas o incluso llevando suministros a las personas que lo puedan necesitar. También son usados para vigilancia por parte de la policía, en España se han estado usando para vigilar playas de difícil acceso, para evitar el uso ilegal de estas. Durante la pandemia, en China, también se usaron drones para controlar que la gente portase la mascarilla, incluso advirtiendo a la gente cuando no cumplían las normas.

Los drones también pueden ser usados por personas civiles para el ocio. Estas personas pueden comprar todo tipo de drones, algunos de estos para grabar videos o para volarlos por simple diversión. También existe una modalidad de drones, llamada FPV, donde los pilotos controlan el dron viendo lo que el dron está haciendo, en tiempo real, a través de unas gafas conectadas a una cámara incorporada en el dron. En la modalidad de FPV hay carreras profesionales, donde los distintos pilotos compiten entre ellos por ser el más rápido.

Existen múltiples empresas en el mercado que se dedican a la fabricación de drones de todo tipo, siendo la más reconocida DJI, una empresa China. Esta empresa tiene drones de todo tipo, drones profesionales, de recreo, incluso militares. Esta empresa también destaca por el uso de la inteligencia en sus drones, siendo muchos de ellos totalmente autónomos. Muchos de estos drones están a la venta para todo tipo de público, pero antes de comprar o fabricar un dron es muy importante conocer la normativa del país donde se vaya a usar para evitar problemas legales.

2.2. Normativa en España

Como se ha comentado en el apartado anterior, conocer la normativa de los vehículos aéreos no tripulados es muy importante para evitar problemas legales. Puesto que este proyecto se ha llevado a cabo en la ciudad de Valencia, se va a comentar un poco la normativa de relativa a drones en España. La Agencia Estatal de Seguridad Aérea (AESA) es el organismo encargado de regular el uso de drones en España. Hasta ahora los drones estaban sujetos al Real Decreto 1036/2017, pero solamente estará vigente hasta el 1 de enero de 2022, cuando entrará en vigor el nuevo reglamento europeo RE 2019/947, por lo que el resumen se centrará en este nuevo reglamento.

Este reglamento establece unas normas generales a todos los drones y también distingue entre tres tipos de categorías: categoría abierta, para vuelos de bajo riesgo sin necesidad de autorización;

categoría específica, para vuelos con riesgo medio con necesidad de autorización en algunos casos; y categoría certificada para vuelos de riesgo alto que requieren autorización.

Entre las normas generales hay que destacar las siguientes: ”el dron ha de estar siempre al alcance visual del piloto; el dron no puede superar los 120 metros de altura de vuelo, medidos desde la superficie donde despegue; no se puede volar un dron en un radio mínimo de 8km de cualquier aeropuerto o espacio aéreo controlado; el dron deberá llevar una placa identificativa ignífuga fijada en la estructura que contendrá datos como el nombre del fabricante, el modelo, y los datos de contacto del piloto; proteger el derecho a la intimidad de los individuos que pudieran aparecer en las imágenes captadas por el dron, y tener especial cuidado con su divulgación pública para no vulnerar la Ley de Protección de Datos; también se recomienda disponer de un seguro de responsabilidad civil.”(OnAir, 2021)

3. FUNDAMENTOS TEÓRICOS DE LOS CUADRICOPTEROS

La dinámica de los cuadricópteros, a diferencia del resto de las aeronaves, es un poco especial, aunque hay conceptos que son comunes a toda la aeronáutica y se aplican también a los drones, como, por ejemplo, los giros y los ejes. Los ejes en la aeronáutica son denominados: roll, pitch and yaw, o en español alabeo, cabeceo y guiñada, que corresponden a los ejes X, Y, Z de un eje tradicional. Los giros alrededor de estos ejes portan el mismo nombre, en la siguiente imagen (figura 1) se pueden ver estos ejes más claramente.

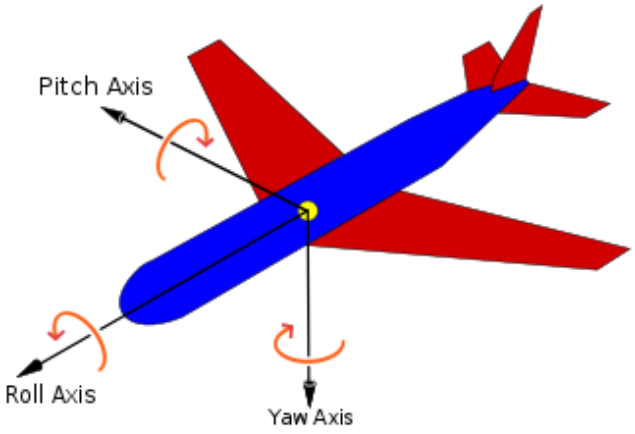

Figura 1: Nombre de los ejes en la aeronáutica(Wikipedia, 2021)

La configuración de los motores del dron se puede establecer de dos maneras, en forma de X (equis) o en forma de (cruz), en el caso de este proyecto, se ha establecido en X. Al girar las hélices en una dirección, un momento es generado en el dron con sentido opuesto al giro de estas. Este momento es debido a la tercera ley de Newton, “acción reacción”, cuando las hélices giran, el contacto con el aire genera el momento en sentido contrario. Por esto mismo los motores se han de instalar en el dron de una forma específica, los motores contiguos han de girar en sentidos contrarios, esto se hace para que los momentos generados por los pares de reacción se contrarresten y no provoquen que el dron gire de forma descontrolada. La configuración de los motores se puede hacer de dos maneras, en forma de X (equis) o en forma de + (cruz), para este proyecto se van a colocar en forma de X, como se puede observar en la imagen:

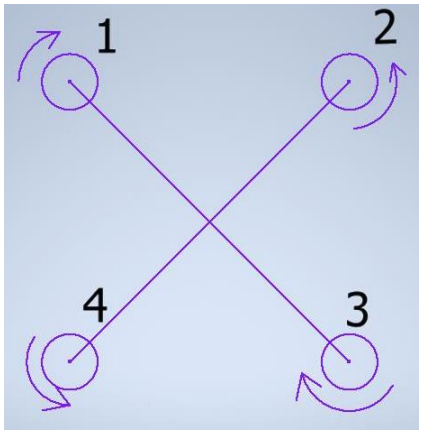

Figura 2: Números de los motores y sentido de giro

Los giros y movimientos del dron del dron también se consiguen de una forma especial, esto es debido a que los cuatro motores solo pueden generar empuje en una dirección. Para conseguir que el dron se mueva en una dirección, hay que hacer que el dron gire en el eje correspondiente, esto provocará que la componente vectorial del empuje no sea completamente vertical. Al estar el dron girado, el vector de empuje tendrá una componente vertical y otra horizontal, la suma de la componente vertical del empuje, de los cuatro motores, ha de seguir siendo igual o mayor al peso del dron, para que este pueda seguir volando. La componente horizontal del vector de empuje será el movimiento lineal que seguirá el dron.

Para hacer que el dron gire respecto a alguno de sus ejes, hay que combinar la acción de los motores de una forma determinada. Para conseguir que el dron gire sobre su eje roll positivo, por ejemplo, se ha de aumentar la velocidad, y por lo tanto el empuje, de los motores 1 y 4 y disminuir la de los motores 2 y 3 (Figura 2), esto provocara que el dron gire sobre su eje roll. Y como el empuje dejará de ser solamente vertical, el dron también se moverá linealmente en la dirección hacia la que está girando, en el caso de la imagen (Figura 2), hacia la derecha. Para que el dron avance hacia adelante, deberá girar sobre su eje pitch y, por lo tanto, deberá aumentar la velocidad de los motores 3 y 4, y disminuir la de los motores 1 y 2. Y por último para que el dron rote sobre sobre su eje yaw, simplemente habrá que aprovecharse de la ley de Newton, antes mencionada, aumentando la velocidad de motores opuestos, por ejemplo, el 1 y el 3, y disminuir la de los motores 2 y 4, esto provocará que el momento, provocado por la reacción del aire, sea mayor en un sentido que en el contrario y hará que el dron rote.

4. COMPONENTES DEL DRON

4.1 Componentes mecánicos

4.1.1 Estructura

La estructura es la base del cuadricóptero, es el componente rígido que le dará la forma al dron y donde irán montados todo el resto de los elementos del dron. Es muy importante la elección de este elemento, pues el peso y tamaño de la estructura condicionará el resto de los elementos mecánicos del dron.

Se estudió la posibilidad de crear una estructura propia para este proyecto, esto daba mucha libertad para la elección del tamaño, forma y material. La idea era crear esta estructura con la ayuda impresora 3D, pero finalmente se descartó esta idea, ya que los materiales disponibles, a un precio razonable, no eran viables. Algunos no tenían la rigidez necesaria para aguantar un hipotético golpe, mientras que otros, siendo más rígidos, no tenían la elasticidad requerida para soportar las vibraciones de los motores por lo que aparecerían microgrietas que acabarían creciendo y fisurando la estructura. Se optó por examinar el mercado, en el que existen estructuras ya preparadas, para proyectos similares, de todos los tamaños y materiales. Puesto que la base de este proyecto es crean un cuadricóptero de bajo coste, teniendo en cuenta el tamaño de la controladora de vuelo y pensando en futuros ampliaciones, se tomó la decisión de iniciar la búsqueda de estructura, bajo estos parámetros. Tras una exhaustiva búsqueda, se encontraron varias opciones interesantes:

- QAV250 drone frame de fibra de carbono (Figura 3)

Figura 3: Estructura QAV250 (Amazon, s. f.)

- BETAFPV Pavo30 Pusher (Figura 4)

Figura 4: BETAFPV Pavo30 Pusher (Amazon, s. f.)

- QWinOut F450

Figura 5:Foto de la estructura QWinOut F450 (Amazon, s. f.)

Finalmente se optó por descartar las dos primeras por varias razones, la primera, aunque parecía buena idea, por estar hecha de fibra de carbono, tenía precio un poco excesivo para el marco de este proyecto (unos 30€). La segunda, aunque tenía un precio más asequible se descartó por el tamaño, una vez elegida la controladora de vuelo. Esta estructura medía unos 150mm entre ejes de motor y el espacio para la controladora de vuelo era muy pequeño. Finalmente, la estructura elegida es la tercera, la estructura F450 del fabricante QWinOut.

Esta estructura fue diseñada por la empresa DJI para uno de sus drones más vendidos, muchos otros fabricantes decidieron tomar este diseño para vender estructuras montables para proyectos caseros. Esta estructura es lo suficientemente grande para albergar la controladora de vuelo y los demás elementos electrónicos del dron. Es una estructura cuadrada, en forma de X, con un ancho de 363mm, una distancia entre ejes opuestos de 455mm y un peso de 282g. La estructura está hecha de un material super resistente, llamado PA66+30GF. Es un material compuesto por un tipo de poliamida (PA66), un material termoplástico, y reforzado con un 30% de fibra de vidrio. Un material con excelentes propiedades mecánicas, preparado para aguantar tanto los posibles golpes que pueda sufrir el dron como las vibraciones producidas por los motores.(Alser, s. f., p. 66)

4.1.2 Motores

Los motores son los actuadores de nuestro dron, son los encargados de mover las hélices para que estas puedan generar el empuje necesario para que el dron vuele. Existen muchos tipos de motores y aunque muchos vehículos aéreos utilizan motores de combustión, la gran mayoría de drones utilizan motores eléctricos de corriente continua. La gran mayoría de motores eléctricos disponen de unas escobillas para cambiar la polaridad de la corriente que los recorre y generar el campo magnético necesario para que giren, y aunque estos motores son muy económicos y comunes, los motores utilizados para los drones son particulares, ya que no disponen de escobillas, son conocidos como "brushless motors", por su traducción al inglés. Se utilizan estos motores puesto que son mucho más eficientes y generan más potencia que los que si llevan escobillas. Esto se debe a que los motores con escobillas están en constante contacto con el rotor, la parte que gira del motor, este contacto genera una fricción bastante grande, que, por un lado, degrada los motores más rápidamente, acortando su vida útil; y, por otro lado, provoca una pérdida de calor muy importante que repercute en la potencia aprovechable y por lo tanto disminuye la eficiencia del motor.

Los motores brushless están compuestos por dos partes principales el stator y el rotor. El stator es la parte fija que no se mueve, y está compuesto por bobinas, tantas como polos tenga el motor, estas bobinas son alambres de cobre enrolladas. El rotor es la parte que gira del motor, esta parte está hecha por un material ferromagnético, es básicamente un imán permanente. Las bobinas, al ser recorridas por una corriente generan un campo magnético, que, dependiendo de la polaridad de la corriente, atrae o repele el imán, haciendo que este gire. Alternando el sentido de la corriente que recorre las bobinas, cambia la polaridad de los imanes, haciendo este proceso de forma correcta, se consigue que el rotor gire de forma indefinida. Estos motores, a diferencia de los que si tienen escobillas, alternan la polaridad de la corriente de una forma muy particular. Las bobinas del stator están agrupadas en tres fases distintas, por las cuales pasará una corriente con distinto sentido, para alternar la polaridad del imán, y así conseguir el giro del rotor, como podemos observar en la figura 6.(Dejan, s. f.)

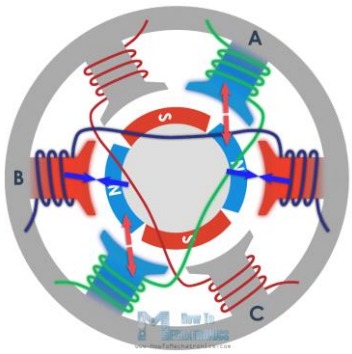

Figura 6: Esquema del interior de un motor Brushless (Dejan, s. f.)

De las tres fases, una será recorrida por una corriente positiva, generando un campo magnético del mismo signo, otra fase será recorrida por una corriente negativa y la última fase no será recorrida por ninguna corriente. En la imagen 6 está representado un motor de 6 polos, podemos observar como hay dos bobinas por cada fase, estando estas opuestas, si el motor fuera de 12 polos habría 4 bobinas por cada fase. En este caso, cuando la fase A es recorrida por una corriente negativa, un campo magnético negativo es generado en la bobina, atrayendo el polo positivo del imán y repeliendo el negativo; la fase B, a su vez, es recorrida por una corriente positiva que atrae el polo negativo y repele el positivo dándole más fuerza al rotor; la fase C no es recorrida por ninguna corriente. Cuando el polo negativo pasa por delante de la fase B, la polaridad de esta fase, y la del resto de fases, cambia de sentido para que esta repela al polo que acaba de pasar y el rotor siga girando. El resto de las fases también alterna el sentido de la corriente, pasando una corriente positiva por C y la fase A es la que no es recorrida por ninguna corriente. Si este proceso es repetido continuamente se consigue que el rotor siga girando. Realmente las fases son recorridas por una única corriente, que entra por una fase y sale por otra, de esta forma se consigue la corriente positiva por una fase y negativa por otra. Lo complicado de este proceso es saber cuando alternar la polaridad de las fases y esto se consigue gracias al efecto hall, el stator tiene unos sensores que se aprovechan de este efecto para detectar este campo magnético y enviar una señal para alternar la corriente. Esta señal es interpretada por el controlador electrónico de velocidad y alterna la polaridad gracias a los transistores MOSFET que posee, el funcionamiento de este elemento será explicado en profundidad en el apartado 4.2.2. (Dejan, s. f.)

Para la elección de los motores es muy importante conocer el empuje que estos pueden generar y el empuje que necesitamos. En el mercado, los motores brushless tienen un parámetro para medir las revoluciones por minuto de forma teórica y, conociendo las características de las hélices poder calcular el empuje teórico. Este parámetro se mide en KV, que son revoluciones por voltio, por ejemplo, con una batería de 3 celdas, aportando cada una, 3,7 voltios y un motor de 1000KV se obtiene un valor teórico máximo de 11000 rpm.

$$3\ celdas\ \times 3{,}7\ voltios\ \times 1000KV = 11000\ rpm\ (1)$$

Conociendo las revoluciones por minuto y las características de la hélice se puede calcular el empuje teórico con la ayuda de fórmulas de aerodinámica. Este proceso es bastante complejo puesto que se necesita determinar algunas constantes de forma práctica para tener una estimación precisa, por lo que se va a recurrir a los datos suministrados por los fabricantes. Aun así, es necesario hacer una estimación de peso de nuestro dron para saber el empuje requerido por los motores:

Elemento	***Peso***
Estructura	282g
ESC	4 × 29,1g = 116,4g
Motores	4 × 54,7g = 218,8g
Batería	181g
Raspberry-Pi	50g
IMU	4g
Driver PWM	5g
Hélices	4 × 9,5g = 38g

Total: 894,2g

Tabla 1: Suma de los pesos de los componentes del dron

En el mercado existen motores de diferentes KV, un motor con un KV alto generará un par menor, pero girará a mayor velocidad, mientras que un motor con un KV bajo generará más par, pero girará a menos velocidad, consumiendo menor energía. Para este proyecto y la estructura elegida, interesa un motor de KV bajo. También hay que tener en cuenta que el empuje generado por los 4 motores ha de ser mayor que dos veces el peso del dron.

Sabiendo esto, se optó por buscar un motor de unos 1000KV, con suficiente fuerza para levantar bastante peso, y con la velocidad necesaria para que las acciones de control funcionen, se encontró el motor A2212 de 1000KV, las indicaciones del fabricante indican un empuje de aprox. 475g para una hélice de 09x4,5 y un empuje de aprox. 800g para una hélice de 10x4,5, el significado de los parámetros de las hélices será explicado en el siguiente apartado.

$$Hélice\ 09x4{,}5: 475g\ \times 4\ motores\ = 1900g\ \geq 894{,}2\ \times 2 = 1788{,}4\ g\ (2)$$

$$Hélice\ 10x4{,}5: 800g\ \times 4\ motores\ = 3200g\ \geq 894{,}2\ \times 2 = 1788{,}4\ g\ (3)$$

Figura 7: Motores A2212 seleccionados (Amazon, s. f.)

4.1.3 Hélices

La hélice es el último elemento mecánico, junto a los motores es la forma de conseguir el empuje y, por lo tanto, lo que hará que el cuadricóptero vuele. Hay hélices de muchas clases, fabricadas con distintos materiales y distintas características, que dependiendo el dron en el que se vayan a instalar serán más o menos eficaces. Entre las características se pueden destacar, el diámetro, el paso, el peso y el número de palas. El diámetro de una hélice es la longitud de la hélice, y por consecuente el diámetro del circulo que forma la hélice al girar, se suele medir en pulgadas. Cuanto mayor sea el diámetro, mayor cantidad de aire podrá impulsar, generando un empuje mayor, pero cuanto mayor sea el diámetro de una hélice, mayor será su inercia y, por lo tanto, mayor será el consumo de energía necesario para moverla. El paso es la distancia que avanza la hélice por vuelta y también se mide en pulgadas, por ejemplo, una hélice que tiene un paso de 4,5” avanza 4,5 pulgadas por cada vuelta completa que de la hélice. Al igual que con el diámetro, cuanto mayor sea el paso, más avanzará el dron, a costa de un mayor consumo energético. Lo mismo pasa con el número de hélices, cuanto mayor sea, mayor superficie de contacto con el aire habrá y por lo tanto mayor empuje generarán, pero el peso e inercia serán mayores y por consecuencia el consumo de energía. Las hélices comerciales se

suelen denominar en función del diámetro y del paso, una hélice de 10x4,5, es una hélice de diámetro 10” y 4,5”.

El empuje creado por las hélices se puede calcular aproximadamente con las ecuaciones de la aerodinámica, pero como se ha explicado antes, son muy complejas y son necesarios algunos datos que no suelen ser proporcionados por los fabricantes Lo que si suelen proporcionar los fabricantes, son los empujes generados en función de lo motores utilizados. Como se ha comentado en el apartado anterior (4.1.2), para el motor seleccionado de 1000KV, unas hélices de 09x4,5 generan un empuje aproximado de 1900g y unas hélices de 10x4,5 generan un empuje aproximado de 3200g. Como el peso del dron es aproximadamente es de 900g, los motores han de ser capaces de levantar el doble del peso, y previendo futuras ampliaciones, lo que conllevaría más peso, se ha decidido escoger las hélices de 10x4,5 de dos palas.

Figura 8: Hélice 10x4,5 seleccionada (Amazon, s. f.)

4.2 Componentes electrónicos

4.2.1 Controladora de vuelo

La controladora de vuelo es el “cerebro del dron”, se encarga de todas las gestiones de vuelo, recibir órdenes de un operador externo, como puede ser un control remoto o una estación de tierra encargada del plan de vuelo de dron. La controladora también será la que reciba las señales de los sensores, ejecute el programa de control del dron y aplique la acción de control sobre los motores, para mantener el cuadricóptero estable durante el vuelo.

En esta sección, se presentarán distintas opciones de hardware que se estudiaron antes de adquirir el definitivo, bajo unos criterios fijados, y elegir la que mejor se adapta al proyecto. Los criterios de comparación serán los siguientes: Peso, precio, velocidad del procesador y capacidad de generación de PWM (necesario para controlar los controladores electrónicos de velocidad). Los hardware que se van a comparar son placas programables llamadas Arduino Uno y Rasberry-Pi.

4.2.1.1 Arduino Uno:

ARDUINO UNO es una placa de desarrollo de bajo coste de la empresa Arduino. Es una placa electrónica basada en un microcontrolador Atmega 328 de Amtel. Esta placa contiene todas las facilidades para conectar periféricos a ella, cuenta con 14 salidas digitales (6 de ellas generan PWM) y 6 entradas analógicas. Esta placa es ampliamente utilizada en proyectos caseros de robótica o automatización.(Arduino, s. f.)

Podemos enumerar varias ventajas de Arduino, una de ellas puede ser su facilidad de programación, se programa en un lenguaje propio de medio nivel, llamado Arduino. Este lenguaje es muy sencillo e intuitivo, muy similar a C++. Otra ventaja del uso de Arduino es la extensa comunidad, que comparte proyectos y librerías sobre las que podemos apoyarnos para este proyecto. Estas librerías serán muy útiles, para interactuar con el sensor inercial, barómetro u otros sensores que se requieran usar.

Arduino Uno dispone de un procesador de 8bits con una frecuencia de 16Mhz, características que a priori parecen un poco modestas, pero que son suficientes para interpretar ordenes exteriores y ejecutar el bucle de control en el periodo necesario. Aunque, para futuras extensiones se queda un poco corto, ya que la reducida velocidad de procesamiento no permitiría añadir más periféricos como un GPS o una cámara en tiempo real. Además, Arduino Uno no tiene WIFI por lo que sería bastante necesario añadir un módulo WIFI.(Fernández, 2020)

Para hacer una comparación equitativa entre todos los hardware se añade a esta opción la compra de una IMU y un módulo WIFI. Como se justificará en el apartado 4.2.2, la IMU seleccionada es la MPU-9250.

El precio de la placa Arduino Uno ronda los 20€ y se puede comprar en la página de web de Arduino o en un gran distribuidor como Amazon. Además, puesto que es un producto de hardware libre, existen muchas imitaciones, más baratas, con las mismas prestaciones que la original, estas imitaciones las podemos encontrar desde 5€. El precio de la MPU-9250 es de 9€ y el del módulo WIFI es de 10€. Lo que hace un total de 39€. En cuanto al peso, la placa Arduino Uno pesa unos 25g, la IMU pesa 4g y el módulo WIFI pesa 5g, esto hace un peso total de 34g.

4.2.1.2 Raspberry Pi:

Raspberry-Pi es una serie de ordenadores de placa reducida de bajo coste, creada por la Raspberry Pi Foundation. Salvo un modelo en particular, del cual hablaremos, los microrordenadores de Raspberry-Pi requieren de un sistema operativo, normalmente se usa el nativo de Raspberry, el Debian, una adaptación de Linux, pero acepta otros sistemas operativos, como Windows. En este apartado, nos vamos a centrar en tres modelos en particular, la Raspberry-pi 3B+, la Raspberry-Pi Pico y la Raspberry-Pi Zero W. Estas dos últimas serán mencionadas por ser opciones interesantes y la primera, será vista más al detalle y la que será comparada al resto de hardware…

Empezaremos por la Raspberry-Pi Pico, esta placa es muy similar a la placa de Arduino, no dispone de sistema operativo, se puede cargar un único programa en ella que se irá ejecutando de forma indefinida. Esta placa fue concebida como una placa de bajo costes para pequeños proyectos caseros o su acrónimo en inglés “DIY” (Do It Yourself). Dispone de un microcontrolador creada por la propia Raspberry-Pi Foundation, llamado RP2040, dispone de 30 puertos GPIO (acrónimo den ingles de “General Purpose Input/Output” o Entrada/Salida de Propósito General en español), incluyendo 4 entradas analógicas y 6 puertos con capacidad de generar PWM. La placa lleva incorporada un módulo WIFI, por lo que no sería necesario incorporar uno extra. Es una opción interesante, pero que se queda un poco corta para este proyecto, ya que la velocidad de procesamiento no es demasiado elevada y esto podría afectar al bucle de control, lo que resultaría en una catástrofe. (Pastor, 2021)

La siguiente placa de esta serie es la Raspberry-pi Zero, es una placa muy interesante ya que es una placa intermedia entre la Pico y la 3B+. Esta placa ya es en todos sus aspectos un miniordenador, no es tan potente como la 3B+, pero ya dispone de un sistema operativo. Dispone de un procesador de 1Ghz y una memoria RAM de 512MB. También dispone, como sus hermanas, de una salida HDMI y un par de puertos USB, además de 40 puertos GPIO, de los cuales solo dos pueden generar una señal PWM. Es una opción bastante interesante, sobre todo por su bajo coste y peso, aunque para este proyecto se ha optado por usar su hermana mayor, la 3B+, por su mayor capacidad de procesamiento y facilidad para utilizar periféricos.(Penalva, 2017)

La Raspberry-Pi 3B+ es la opción más interesante para este proyecto, se trata de una de las placas más potentes de la empresa. Esta placa, como su hermana pequeña pero mucho más potente, es un miniordenador. Entre sus características destaca un procesador mejorado, de 4 núcleos con una frecuencia de reloj de 1,4Ghz; una memoria RAM mayor que su predecesora de 1GB; cuenta con todas las facilidades para conectar periféricos, 4 puertos USB, un puerto HDMI, un puerto Ethernet y un puerto para conectar la cámara nativa de la empresa, esta cámara no será usada en este proyecto, pero es muy interesante para futuras ampliaciones. Además, cuenta con una conectividad más que envidiable, incluyendo Bluetooth 4.2 y WIFI a doble banda de 2.4 y 5Ghz. (Foundation, s. f.)

Figura 9: Foto de la placa Raspberry-Pi 3B+ (Foundation, s. f.)

La placa, como el resto de las placas de la empresa, cuenta con 40 pines GPIO, y aunque en las especificaciones se puede entender que dispone de 4 puertos de PWM, realmente tiene 2, puesto que

dos de ellos están clonados. Esto es un problema para este proyecto, porque la capacidad de generar PWM de la placa es vital para el control de los motores. En un principio se pensó en la posibilidad de generar la señal PWM por software, aunque parecía una solución factible, tras las pruebas realizadas se descubrió que era inviable y que había que buscar una solución alternativa. El problema de generar la señal por software es que la Raspberry maneja muchos procesos al mismo tiempo, y aun dándole máxima prioridad a este proceso, de vez en cuando sufría pequeñas interrupciones, que dejaban los motores parados. Estas interrupciones eran cortas, de alrededor de 100ms, pero para un bucle de control que tiene que actuar cada 20 ms, se trata de un 500%, es una interrupción demasiado larga que haría perder la estabilidad del dron y que este se estrellase. La solución que se ha encontrado es un hardware que se comunica a con la placa a través de I2C y genera la señal por su cuenta, no viéndose afectado por las interrupciones que se puedan generar, se explica más a fondo este elemento en el apartado 4.2.4.

La elección de esta placa por delante de sus hermanas y Arduino está más que justificada, aunque su precio y peso es un poco mayor, se podrá, en futuras ampliaciones, llegar mucho más lejos de lo que se podría con las otras. El precio de la Raspberry-Pi 3+ es de 35€, en su página oficial, como se ha comentado antes, la IMU cuesta 9€ y el controlador para generar PWM 7€, esto es un total de 51€. En cuanto al peso, la Raspberry pesa 50g, la IMU 4g y el controlador 12g, un total de 66g.

4.2.2 Unidad de medida inercial

Una unidad de medida inercial (IMU de ahora en adelante, por sus siglas en inglés), es una unidad con diferentes sensores electrónicos que recolecta datos de los movimientos inerciales en un sistema de referencia, tales como, velocidad angular, aceleración lineal y campo magnético terrestre. Estos datos pueden ser utilizados para obtener la posición, velocidad y/o aceleración del dispositivo donde se instala. Este dispositivo es vital en un cuadricóptero, ya que es necesario conocer en todo momento la inclinación del dron, en los tres ejes, para poder calcular el error respecto a la referencia y aplicar la acción de control correctora para mantener el vehículo estable. En este proyecto, se utilizará un dispositivo IMU MEMS (micro-electromechanical system), que básicamente es una IMU de tamaño y peso muy reducido.

¿Cómo se mide cada dato?

Acelerómetro:

El acelerómetro es el sensor encargado de medir la aceleración lineal del dispositivo. Funciona con una combinación de una masa móvil, gracias a muelles, y unas placas de silicio estacionarias. Con el movimiento del dispositivo, la masa se mueve solidaría al movimiento, generando un cambio de los valores de capacitancia de C1 y C2, este cambio es interpretado por un circuito integrado, resultando en valores de aceleración.(Dejan, s. f.)

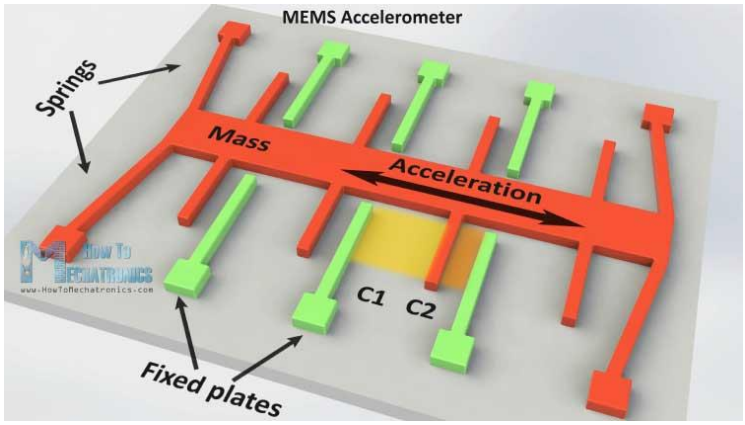

Figura 10: Esquema de un acelerómetro (Dejan, s. f.)

Estos valores de aceleración se pueden usar para conocer la inclinación del dron, ya que la aceleración de la gravedad siempre estará presente. Combinando las medidas en los tres ejes y aplicando un poco de trigonometría se obtiene la posición, las fórmulas trigonométricas son las siguientes:

$$roll = \text{atan}\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \quad (4)$$

$$pitch = \text{atan}\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right) \quad (5)$$

$$yaw = \text{atan}\left(\frac{\sqrt{A_y^2 + A_x^2}}{A_z}\right) \quad (6)$$

Con estas ecuaciones se puede conocer una estimación de la inclinación, pero surgen dos grandes problemas: el primero es que si el dron está en su posición natural, por ejemplo, apoyado en una mesa y se gira el dron alrededor de su eje yaw (eje z), el dispositivo no detectará ningún cambio en la gravedad, ya que esta solo está presente en el eje z, por lo que el resultado de la tercera ecuación siempre será el mismo y la orientación en este eje no variará; el segundo problema es que el acelerómetro es muy susceptible a las vibraciones y aceleraciones externas, como el viento o la aceleración de los propios motores, esto provocará un error en las medidas y por consecuencia en la estimación de la posición. Estos problemas se pueden solucionar combinando las medidas del acelerómetro con las del giroscopio y aplicando un filtro adecuado.

Giroscopio:

Un giroscopio en un sensor electrónico encargado de medir la velocidad angular en los tres ejes del sistema de referencia del dispositivo donde se aloje. El giroscopio funciona de forma similar al acelerómetro, calculando la velocidad angular gracias a los cambios de capacitancia.

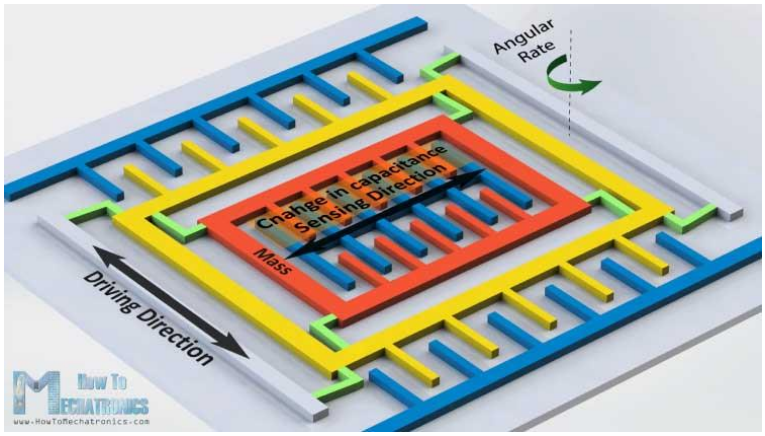

Figura 11: Esquema de un giroscopio (Dejan, s. f.)

La posición en el caso del giroscopio se estima de forma diferente, ya que las medidas que obtenemos son velocidades angulares, integrando estas medidas respecto el tiempo que pasa entre una lectura podremos obtener una estimación de la posición:

$$roll = roll + G_x \times \Delta T \quad (7)$$

$$pitch = pitch + G_x \times \Delta T \quad (8)$$

$$yaw = yaw + G_x \times \Delta T \quad (9)$$

Pero como no podia ser todo perfecto, vuelve a surgir un gran problema, y esque el giroscopio tambien es susceptible a un pequeño ruido de medida. A pesar de que este ruido es pequeño, al integrar los valores de las medidas, este error se va acumulando y acaba convirtiendose en un error muy grande y hace de la estimación de la posición un valor inservible. Esto se soluciona, como se ha nombrado antes, combinando los valores del acelerometro y del giroscopio y tratandolo con un filtro, por ejemplo el complementario. El filtro complementario añade los valores del acelerometro y del giroscopio, multiplicados por un coeficiente en tanto por 1, dandole más importancia a una medida u otra:

$$roll = 0{,}98 \times roll_{gyro} + 0{,}02 \times roll_{accel} \quad (10)$$

$$pitch = 0{,}98 \times pitch_{gyro} + 0{,}02 \times pitch_{accel} \quad (11)$$

$$roll = 0{,}98 \times yaw_{gyro} + 0{,}02 \times yaw_{accel} \quad (12)$$

Siendo roll_gyro y roll_accel, las posiciones estimadas por el giroscopio y el acelerometro respectivamente. Se puede observar que el filtro complementario no es otra cosa que un filtro paso alto para el giroscopio y un filtro paso bajo para el acelerometro. Se ha utilizado valores de 0,98 y 0,02 pero se puede variar dependiendo de cada caso y las pruebas realizadas.

Con el filtro complementario se obtiene una estimación bastante buena de la posición, pero sigue apareciendo un problema ya mencionado antes. La estmación de la inclinación yaw del acelerometro

no es fiable, y no siendo esta medida fiable, de nada sirve el filtro complementario porque recae todo en la medida del giroscopio, que como se ha mencionado antes, acumula error. Este problema se puede solucionar combinando estas medidas con las del magnetometro y aplicando un filtro nuevo.

Magnetómetro

Un magnetómetro o un compás es un dispositivo que permite medir la densidad de flujo magnético gracias al efeto hall, la unidad de esta medida en el SI es Tesla. Estas medidas pueden ser utilizadas para conocer la dirección o inclinación del dispositivo respecto al campo magnético terrestre. Las medidas del magnetómetro pueden ser combinadas con la del acelerómetro y giroscopio para obtener una estimación de la posición mucho más precisa, sobre todo en el eje yaw, que como se ha explicado antes, había una deficiencia de precisión. En este caso habrá que combinarlas utilizando un filtro Kalman.(Ltd, s. f.)

El filtro Kalman es un algoritmo, apoyándose en la estadística, para estimar variables, en este caso la posición. Es un algoritmo bastante complejo, que se sale del alcance de este trabajo, por lo que se va a explicar de forma resumida. El algoritmo está diseñado para tener en cuenta el ruido de los sensores y corregirlo. El algoritmo se divide en dos fases, una primera fase de predicción y una segunda fase de corrección de la predicción. En la fase de predicción se hace uso de la media y de la varianza para estimar el estado siguiente a partir del estado actual, con una distribución normal. En la fase de corrección se usan los errores anteriores para intentar predecir el error futuro y corregirlo. Usando las dos fases de forma adecuada se consigue una estimación bastante precisa de la posición. Por sencillez y por alcance de este proyecto, se hará uso de una librería de Python, ya creada por otro autor, para facilitar el uso de este Filtro, esto se explicará en el apartado 5.2.(Gluón, 2019)

En este trabajo se ha decidido usar una IMU MEMS, de la empresa de electrónica Adafruit, en concreto la llamada “MPU 9250”, que consiste en una combinación de un sensor “MPU 6050” de 6GDL (acelerómetro y giroscopio), y un magnetómetro “AK8963” de 3GDL.

Figura 12: Foto de la IMU elegida (Amazon, s. f.)

Se ha elegido este sensor por su extendido uso en aeromodelismo, por su tamaño y peso (es muy pequeño y ligero) y su bajo coste (unos 9€).

La comunicación con el dispositivo se realizará a través del protocolo I2C, un bus de campo que permite conectar varios dispositivos a la vez a un mismo bus de campo.

4.2.3 Controladores Electrónicos de Velocidad

Los controladores electrónicos de velocidad (ESC de ahora en adelante, por su traducción inglesa "Electronic Speed Controller") son los elementos que controlan la velocidad de los motores, recibiendo una señal PWM y enviando una corriente por la fase correspondiente del motor para que estos puedan girar. Esto se consigue activando los transistores MOSFETs adecuados. Los ESC, poseen las siguientes conexiones: un cable rojo y uno negro que se conectan al polo positivo y negativo de la batería, respectivamente; un cable triple para transmitir la señal PWM; y tras cables que se conectan al motor, uno por cada fase (Figura 13).

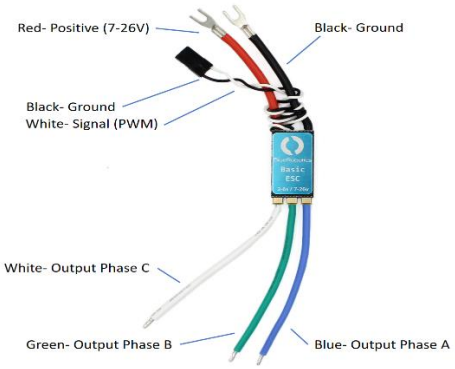

Figura 12: Esquema conexiones ESC (BlueRobotics, s. f.)

En la siguiente imagen (Figura 13) se puede observar como funcionan los transistores para mandar la corriente por la fase adecuada. Cuando el microcontrolador recibe la señal para alternar la fase, la corriente, que viene del polo positivo de la batería, pasa por el transistor Q2, esta entra por la fase B, generando un campo magnético positivo; y sale por la fase C, generando un campo magnético negativo en esta.

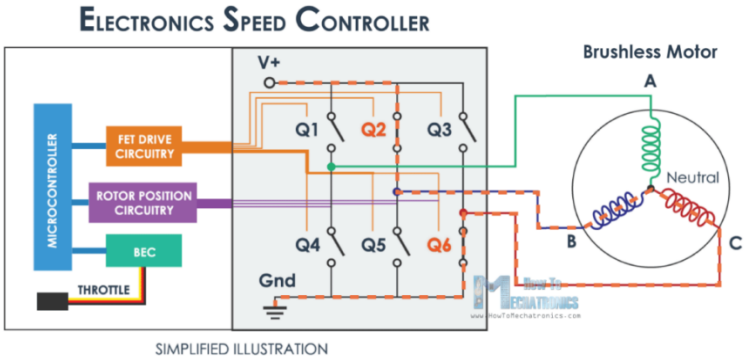

Figura 13: Esquema del interior de un ESC (Dejan, s. f.)

4.2.4 PWM driver

La señal PWM o modulación de ancho de pulso, es una forma de generar una señal analógica con una señal digital, esto se consigue alternando la señal digital entre on/off durante un periodo de tiempo. Cuanto más tiempo esté la señal en on, mayor será el voltaje efectivo en la salida

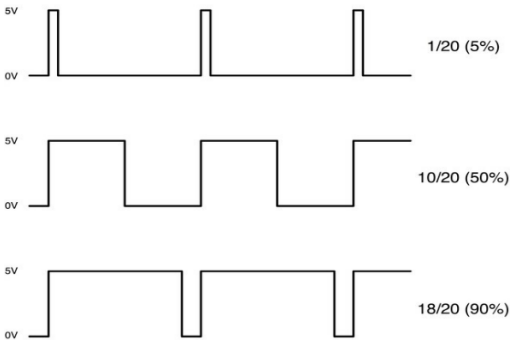

Figura 14: Grafica de tres señales PWM según el pulso (Ergosum, s. f.)

El duty cycle es el porcentaje de tiempo que está en ON respecto al periodo completo, por lo tanto, el voltaje que se mide a la salida será el ese porcentaje del voltaje total disponible (en este caso, cuando las salidas están en ON (o en 1), el voltaje es de 5V). Siendo el duty cycle calculable con la siguiente formula (13):

$$Duty\ cycle = \frac{t_{on}}{T} \times 100 \quad (13)$$

La generación de una buena señal PWM es muy importante en este proyecto porque, como se ha explicado antes, los motores pueden regular su velocidad gracias a los controladores electrónicos de velocidad que pueden ser controlados regulando la frecuencia de una señal PWM entre 1000 y 2000 ms.

El hardware para generar la señal PWM es necesario porque la controladora no tiene capacidad suficiente para generar esta señal por ella misma por hardware y generarla por software sería muy arriesgado, podría acabar con el dron destruido. Tras buscar en el mercado se ha elegido un generador

de PWM de la empresa de electrónica Adafruit, más concretamente el PCA9685, se ha elegido este por su buena relación peso-precio/calidad. Este componente se comunica con las Raspberry-Pi a través de I2C, es un bus de comunicación en serie, bastante sencillo de usar. Además, Adafruit proporciona una librería en Python para facilitar su uso.

Figura 15: Foto del driver PCA9685 (Amazon, s. f.)

4.2.5 Baterías

La elección de las baterías es una parte vital del proyecto, porque de ello dependerá el peso y la autonomía del dron. Aunque las baterías están evolucionando rápidamente, con la tecnología, sigue existiendo una gran paradoja, y es que cuanto mayor capacidad tienen las baterías, mayor autonomía tienen, pero más pesan. Y cuanto más pesan, mayor empuje han de aportar los motores, en consecuencia, mayor es el gasto de energía y por lo tanto baterías de mayor tamaño son necesarias para mantener la autonomía, y así sucesivamente. Por eso es muy importante la elección de la batería ideal, con una buena relación capacidad/peso.

Existen varias variedades de baterías en el mercado del aeromodelismo, hay variaciones de baterías basadas en Níquel, a destacar las baterías NiCd (Níquel-Cadmio) y baterías NiMH (Níquel-Metal hidruro). También existen variaciones de baterías basadas en Litio, a destacar las baterías de Iones de Litio y las baterías LiPo (batería de Polímero de Litio)

Las baterías de Níquel-Cadmio eran muy usadas en el sector doméstico e industrial, actualmente se usan mucho menos porque el cadmio es muy contaminante y tienen efecto memoria. El efecto memoria se provoca cuando se carga una batería sin haber sido descargada del todo y se forman unos cristales en el interior, por una reacción química, el efecto memoria reduce la capacidad de las baterías. Las baterías NiCd tienen una larga vida útil, alrededor de 1000-1500 ciclos y cada celda aporta unos 1,2 Voltios. («Batería de níquel-metalhidruro», 2021)

Las baterías de Níquel de Metal Hidruro utilizan el mismo ánodo que las baterías de Níquel-Cadmio, pero en vez de utilizar cadmio, utilizan una aleación de hidruro metálico como cátodo. Están baterías son menos toxicas, al eliminar el Cadmio, además, poseen mayor capacidad de carga y menor efecto memoria. Estas baterías tienen una vida útil entre 500 y 1500 ciclos, y proporcionan 1,2 Voltios

por celda. Estas baterías presentan una mayor tasa de auto descarga que las anteriores por lo que al ser almacenadas se descargaran antes.(«Batería de níquel-metalhidruro», 2021)

Las baterías de Iones de Litio emplean una sal de litio como electrolito. Estas baterías tienen una vida útil más corta que las dos anteriores, unos 300-1000 ciclos, también son más costosas que las anteriores. Por otro lado, estas baterías son muy ligeras, y tienen un tamaño bastante reducido. Además, cada celda aporta mucha más tensión, 3,7 voltios, frente a los 1,2 voltios de las baterías de Níquel. Estas baterías también consiguen reducir el efecto memoria. Pero estas baterías tienen un gran riesgo, y es que, si se sobrecalientan mucho, corre el riesgo de que se auto inflamen o incluso exploten, por lo que es necesario añadirles medidas de seguridad que las hacen más costosas.(Wikipedia, 2021)

Las baterías de polímero de Litio funcionan de forma muy similar a las de Iones de Litio, pero en estas baterías, el electrolito es sólido en vez de ser líquido. Eso es una ventaja ya que disminuye el peligre de que se derrame el electrolito y se inflame la batería. Aunque tiene una vida menor, estas baterías son más pequeñas, más robustas y flexibles. Estas baterías son ideales para aeromodelismo, sobre todo por su bajo su peso y flexibilidad. (330ohms, 2020)

Tras analizar las opciones mencionadas, se decidió optar por las baterías de polímero de Litio, más comúnmente conocidas como baterías LiPo. Estas baterías se tienen que cargar de una forma específica, ya que tienen que cargarse con una corriente continua y constante, por lo que también es necesario comprar un cargador inteligente de baterías, se optó por comprar uno de bajo coste. Ya se disponía de una batería LiPo de proyectos anteriores, ya que estaba en buen estado, se optó por utilizar esta y evitar comprar una nueva. Se trata de una batería LiPo de 2600mAh, con 3 celdas de 3,7 voltios, lo que hace una batería de 11,1 voltios. Para calcular la autonomía aproximada, primero hay que conocer el consumo medio, Este es la suma del consumo de la Raspberry-Pi, los sensores y los motores.

Elemento	**Consumo medio**
Raspberry-Pi	6 W
Sensores	0,5 W
Motores	120 W
TOTAL:	**126,5 W**

Tabla 2: Consumo energético de los distintos elementos del dron

Con un consumo medio de 126,5W, solo falta averiguar los Watios-hora que puede suministrar nuestra batería, esto lo haremos con la siguiente formula:

$$Wh = V \times Ah \quad (14)$$

Con una batería de 11,1 V y 2,6 Ah, obtenemos un valor:

$$Wh = V \times Ah = 11{,}1V \times 2{,}6\,Ah = 28{,}86Wh \quad (15)$$

Si ahora dividimos este valor entre el consumo medio, obtenemos la autonomía en horas, y si lo multiplicamos por 60, obtenemos el valor de la autonomía en minutos:

$$28{,}86Wh \div 126{,}5W = 0{,}228\,h \quad (16)$$

$$0{,}228h \times 60 = 13{,}69\,\text{min} \quad (17)$$

Se obtiene un valor medio de autonomía de 13,69 minutos, lo cual es un valor bastante bueno en cuenta la autonomía media del resto de drones que existe en el mercado.

5 IMPLEMENTACIÓN Y PRUEBAS

5.1 Implementación hardware

En este apartado se va a explicar todo el proceso de montaje del dron, desde el ensamble de la estructura y de los motores hasta las soldaduras de los ESC a la PCB conectada a la batería. Es un proceso bastante sencillo por lo que se va a explicar de forma bastante breve. Se hizo uso del manual de instrucciones proporcionado por el fabricante de la estructura. El proceso seguido fue el siguiente:

1: Se atornillan las cuatro patas (nº1 en la imagen (Figura 16)) a las placas de unión (nº2 en la imagen);

2: Se atornillan los motores (nº4 en la imagen) a las patas (nº2);

3: Se sueldan los cables positivo y negativo de los ESC (no mostrados en la imagen) a la PCB integrada en la placa de unión inferior (nº2 en la imagen);

4: Se atornillan las hélices (nº3 en la imagen) a los ejes de los motores (nº4 en la imagen);

5: Por último, se comprueba con un multímetro que no está cortocircuitado ninguno de los cables o soldaduras

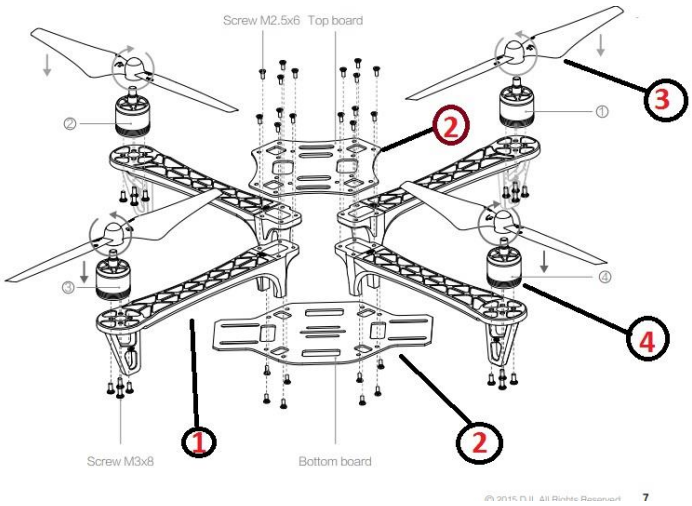

Figura 16: Esquema de montaje del dron (Amazon, s. f.)

Los siguientes pasos que siguieron, fueron: conectar los puertos SDA, SCL (estos puertos son para la comunicación I2C), alimentación positiva y tierra común de la Raspberry-Pi al driver para generar PWM y al sensor inercial, esta conexión se hizo en paralelo. También se conectan los cables de conexiones de PWM de los cuatro ESC a cuatro de los canales del driver PCA9685. Los ESC's fueron anclados a las patas de la estructura con la ayuda de unas bridas de plástico.

Una vez montada la estructura y acoplado todo el hardware se pasa a la parte más compleja de este proyecto, que es la implementación del software, la creación y ajuste del código.

Figura 17: Foto del dron montado

5.2 Implementación software

La implementación del software por la configuración de la Raspberry-Pi, como se ha explicado antes, esta requiere de un sistema operativo. Aunque existen otras opciones como Windows, se ha optado por usar el sistema operativo nativo de Raspberry, el Debian, un sistema operativo de software libre basado en Linux. El sistema operativo se puede descargar desde la página web oficial de Debian, este habrá que instalarlo en una microSD formateada, con la ayuda de un programa llamado NOOBS. Una vez instalado, solo hay que introducir la microSD en la Raspberry, conectar esta a la alimentación y a un monitor, con HDMI, y ya podremos ver el escritorio del miniordenador. Para poder controlar la Raspberry, habrá que conectar también un teclado y un ratón a los puertos USB instalados en la placa. El siguiente paso será actualizar el software y las bibliotecas básicas, desde el terminal de la Raspberry-Pi, una vez hecho esto, está todo listo para empezar a programar el programa del dron.

Para escribir el programa del dron se han utilizado dos entornos de programación, uno instalado por defecto en la Raspi, Thonny y otro instalado en el ordenador Windows, Spyder de Anaconda, este último es para poder leer el programa sin necesidad de tener la Raspberry-Pi conectada. Para no

alargar demasiado este apartado se van a explicar las funciones y clases más importantes del programa, obviando las partes triviales.

Una de las partes más importantes del programa y del proyecto es el bucle de control de estabilidad del dron, este es el encargado de corregir los ángulos de inclinación del dron. El objetivo del controlador es el vuelo estacionario. Para hacer esto primero se leerán los tres ángulos de inclinación del dron (pitch, roll y yaw), se compararán con la referencia, que en este caso será 0º para los tres ángulos, y la diferencia de cada ángulo pasará por un controlador PID para calcular una acción correctora. Esta acción correctora se introducirá en un algoritmo que mezcla las acciones de control para enviarlas al motor correspondiente, para luego aplicarla en estos. Realmente, debería haber un cuarto PID para el control de la altura, que se mediría con un barómetro, aunque no ha sido posible instalarlo por falta de tiempo. En la siguiente imagen (Figura 18) se puede observar un esquema para entender este bucle de forma más sencilla.

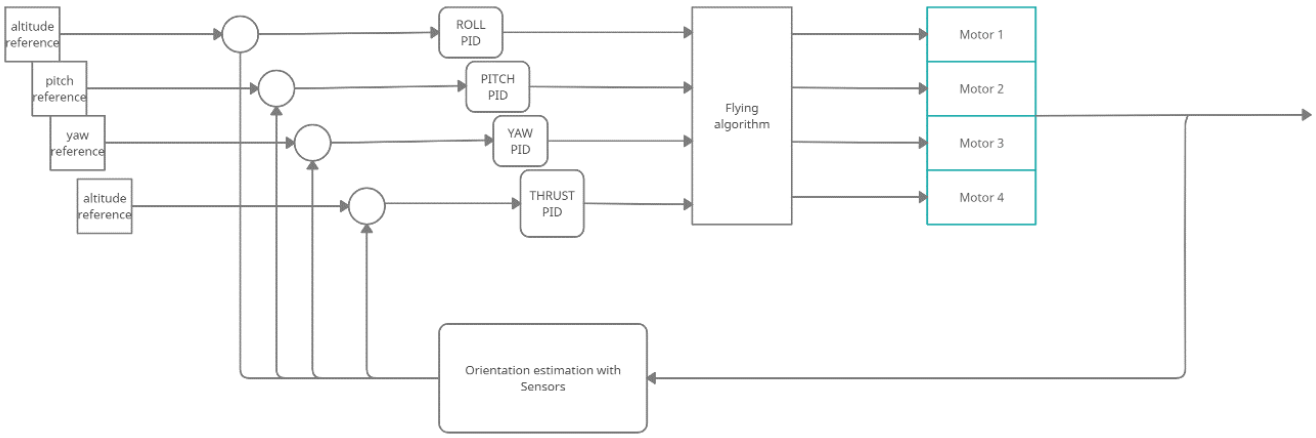

Figura 18: Esquema bucle de control

Para empezar el programa, lo primero es importar las librerías y clases necesarias para el funcionamiento del programa, entre ellas podemos destacar la librería "*Adafruit_PCA9685*" que nos proporciona el fabricante Adafruit para facilitar el uso del driver para generar PWM. También podemos destacar la importación de la clase MPU9250, para controlar la IMU de forma más sencilla, esta clase contiene funciones para leer los sensores y transformar las medidas a valores más fáciles de entender. También vamos a importar una la clase "*Kalman*" de la librería "*imusensor.filters*", para hacer uso del filtro Kalman, que ayudará a definir con precisión la orientación del dron.

```
import os
import time
from board import SCL, SDA #libreria necesaria para inicializar la PCA9685
import busio #libreria necesaria para inicializar la PCA9685
import Adafruit_PCA9685 #libreria para el driver PCA9685, para generación de pwm
import math #libreria necesaria para calcular los arctan
import smbus
import sys
sys.path.append("")
from mpu9250_jmdev.registers import * #importar modulo de la -imu
from mpu9250_jmdev.mpu_9250 import MPU9250 #importar clase de la imu
from imusensor.filters import kalman #importar clase para aplicar filtro kalman
```

Figura 19: Clases y librerías importadas

En la siguiente figura se puede observar un diagrama de flujo para entender mejor el funcionamiento del programa (Figura 20).

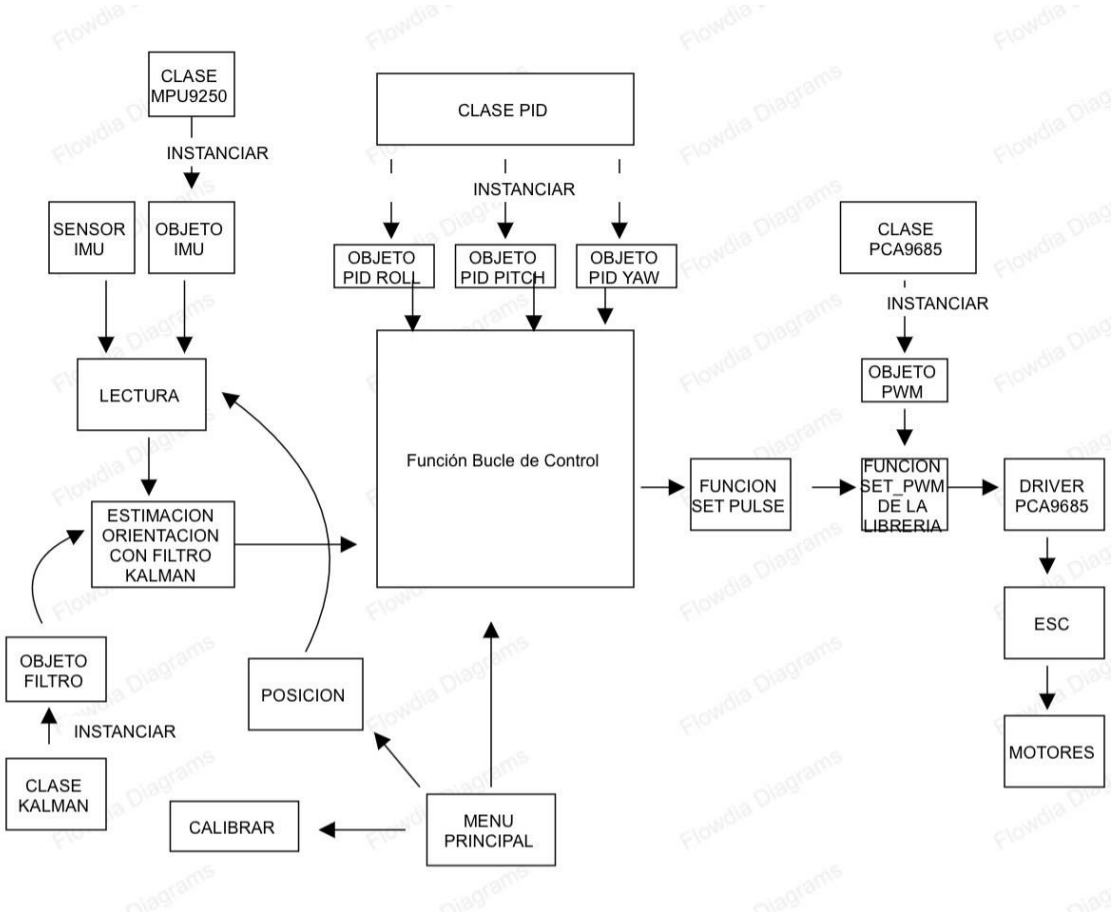

Figura 20: Diagrama de flujo de funcionamiento del programa

Lo siguiente es definir una clase para el controlador PID, esta clase nos servirá para instanciar los objetos de los tres controladores PID que vamos a utilizar, uno por cada ángulo. Esta clase contiene dos funciones, una inicial, que recibirá los parámetros P, I y D a la hora de instanciar los objetos, para utilizarlos a la hora de calcular la acción de control; y una segunda función, llamada "*calculo_pid*", que calcula la acción de control. Esta función recibe la referencia deseada, normalmente 0º, en la variable

"*ref*", y la inclinación actual del dron, en la variable "*pos*". La función también devuelve una variable, llamada "PID", que es la acción de control.

La función primero calcula la diferencia entre estas dos variables y con la diferencia, que es el error.

$$err = referencia - posicion \quad (18)$$

Con este error se calculan tres acciones:

-La acción proporcional (Up), que como su nombre indica es una parte proporcional del error, se multiplica el error por el parámetro P.

$$Up = P \times err \quad (19)$$

-La acción integral (Ui) proporciona una corrección para compensar las perturbaciones basándose en la acción integral anterior. Multiplica el error por el parámetro I y le añade la acción integral anterior.

$$Ui = Ui_{ant} + (I \times err) \quad (20)$$

-La acción derivativa (Ud) intenta anticipar la acción de control para estabilizar más rápidamente la variable controlada y así, intentar reducir al máximo la sobre oscilación. Se calcula la diferencia entre el error actual y el error inmediatamente anterior, y se multiplica esta diferencia por el parámetro D.

$$Ud = D \times (err - err_{ant}) \quad (21)$$

Por último, se añaden estas tres acciones para obtener la variable "PID", que retornará la función.

$$PID = Up + Ui + Ud \quad (22)$$

```
14  #creamos una clase PID, con los parámetros del PID, y una función que cálcula la acción de control
15  #con esta clase, crearemos los objetos para cada PID
16  class PID:
17      def __init__(self, P, I, D):
18          self.P=P
19          self.I=I
20          self.D=D
21          self.err_ant=0
22          self.Ui_ant=0
23      def calculo_pid(self, ref, pos):
24          global Ui_ant
25          P=self.P
26          I=self.I
27          D=self.D
28          Ui_ant=self.Ui_ant
29          err_ant=self.err_ant
30          err=ref-pos
31          Up=P*err
32          Ui=Ui_ant+I*err
33          Ud=D*(err-err_ant)
34          PID=Up+Ui+Ud
35          #Anti-windup
36          if(PID>=550 or PID<=-450):
37              Ui=Ui_ant
38              PID=Up+Ui+Ud
39          self.err_ant=err
40          self.Ui_ant=Ui
41          return PID
```

Figura 21: código de la clase "PID"

Como se puede ver en la figura anterior (Figura 20), en el código también se ha añadido, entre la línea 36 y 38, un antiwindup, que es básicamente una función condicional, que en caso de que la acción de control salga de unos límites, se deja de añadir la acción integral para evitar que esta quede saturada. Esta clase, como se ha explicado antes, será usada para instanciar los objetos de cada PID,

llamados: "*pidroll*", "*pidpitch*" y "*pidyaw*", esta parte del código será mostrada en el anexo ya que no es relevante.

La siguiente función importante es el bucle de control, donde se usará la clase PID y su función para calcular la acción de control. En esta función, se seguirá el bucle anteriormente explicado en el esquema de la figura 18. Lo primero que se hace es definir las variables globales "*posición*" y "*thrust*", que es el empuje en ms. Se usa la función try-except para englobar el bucle while, esta función permite detectar errores y, en caso de que ocurra alguno, parar los motores con la función stop(). En el bucle while, el primer paso es calcular el tiempo de ejecución, necesario para calcular la posición, esta se calcula con la función "*posición_filtro_kalman*", que hace uso de la librería, antes nombrada, usando el filtro Kalman, la posición se almacena en un vector de tres variables llamado "posición". Después se calculan las acciones de control para cada PID, usando la función "*calculo_pid*", vista en la figura nº, a esta función hay que pasarle el vector "*ref*" y "*posicion*" y devuelve la acción de control, que se almacena en las correspondientes variables. Una vez calculada la acción de control, se introducen en el algoritmo de mezclado, que calcula el pulso a introducir en cada motor dependiendo la posición y el sentido de giro, como se ha explicado en los fundamentos teóricos de los drones. Por último, se envía la señal al driver PCA9685, para que genere la señal PWM, que irá a los ESC y finalmente a los motores. Esto último se hace con la función "*set_pulse*", que convierte el valor decimal en milisegundos a un valor en 16 bits, y haciendo uso de la librería proporcionada por Adafruit, se envía la señal al driver.

```
114  def bucle_control(ref):
115        global posicion
116        global thrust
117        tiempo_ini=time.time()
118        thrust=1400
119        try:
120            while True:
121                tiempo=time.time()-tiempo_ini
122                posicion=posicion_filtro_kalman(tiempo)
123        #se calculan las acciones de control
124                ac_roll=pidroll.calculo_pid(ref[0], posicion[0])
125                ac_pitch=pidpitch.calculo_pid(ref[1], posicion[1])
126                ac_yaw=pidyaw.calculo_pid(ref[2], posicion[2])
127        #se calcula la señal pwm a aplicar a partir de las acciones de control calculadas
128                vel_m1 = thrust + ac_roll - ac_pitch + ac_yaw
129                vel_m2 = thrust - ac_roll - ac_pitch - ac_yaw
130                vel_m3 = thrust - ac_roll + ac_pitch + ac_yaw
131                vel_m4 = thrust + ac_roll + ac_pitch - ac_yaw
132        #se envia la señal pwm deseada a los actuadores
133                set_pulse(m1, vel_m1)
134                set_pulse(m2, vel_m2)
135                set_pulse(m3, vel_m3)
136                set_pulse(m4, vel_m4)
137                tiempo_ini=time.time()
138        except:
139            print("error en el bucle de control")
140            stop()
```

Figura 22: Bucle de control

Se han explicado las funciones más importantes, pero en el anexo nº2, se puede ver el código completo, con ciertos comentarios para explicar las diferentes funciones. Tras escribir el código, se realizaron pruebas para comprobar el buen funcionamiento de la IMU, del driver y ajustar los Parámetros del PID.

5.3 Pruebas

Las primeras pruebas que se realizaron fueron para comprobar el funcionamiento de la IMU y del filtro Kalman. Se modificó el código para almacenar los datos de posición en un fichero de texto para posteriormente trazar una gráfica con la ayuda de Matlab y ver los datos de forma más clara. Se hizo una prueba, moviendo el dron 90º en su eje pitch, devolviéndolo a su posición original, moviéndolo 90º en su eje roll y devolviéndolo a su posición original de nuevo. Se almacenaron la posición con los ángulos medidos por el giroscopio (gyro), los ángulos medidos por el acelerómetro (accel) y los ángulos filtrados con el filtro Kalman.

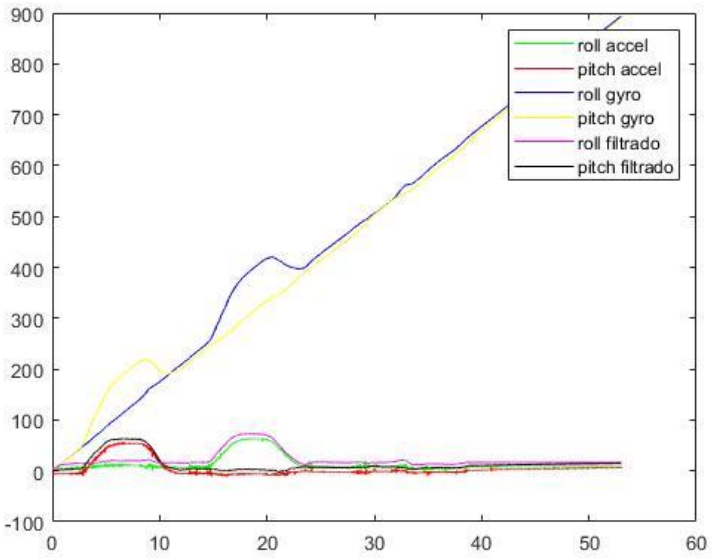

Figura 23: Gráfica de los datos medidos

En la gráfica se puede observar como la posición calculada integrando los datos del giroscopio va acumulando error (drifteo) y se vuelve un valor inutilizable. En cuanto a la posición calculada con los datos del acelerómetro, este no acumula error, pero se puede observar que no es del todo precisa y además, contiene bastante ruido. Sin embargo, la posición calculada con el filtro Kalman es muy precisa y no contiene ruido.

La siguiente prueba que se realizó fue para comprobar la señal PWM generada por el driver PCA9685, esta prueba se realizó con la ayuda de un osciloscopio digital portátil. Se conectó el positivo del osciloscopio al puerto PWM de uno de los canales del driver y el negativo al “ground” del mismo canal. Se comprobó que la señal generada por el driver es la señal que se buscaba generar, por lo que ya se podía probar a conectar los ESC al driver y a los motores y probar estos.

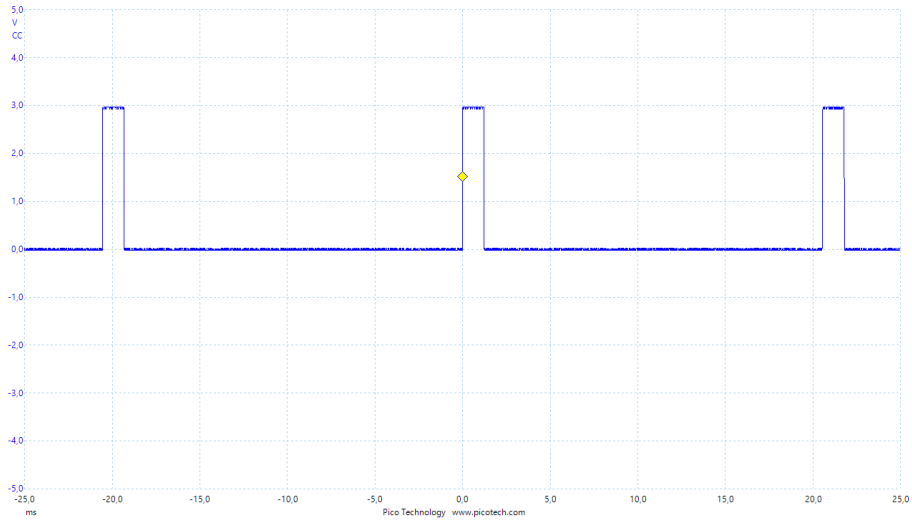

Figura 24: gráfica señal PWM medida con el osciloscopio digital

Una vez comprobada la señal PWM, conectados los ESC y los motores, se probó a controlar los motores con un programa sencillo, que simplemente establecía un pulso PWM. Se comprobó que los motores funcionaban, se dejaron los motores funcionando durante un largo periodo de tiempo para comprobar que no había interrupciones y se observó que todo iba bien. Además, se utilizó el osciloscopio para comprobar la señal que enviaba el ESC a los motores, al tener un osciloscopio simple, solo se pudo conectar a dos de las fases. Se hizo esta prueba con un pulso de 1100 ms, 1500ms y 2000ms. Se puede comprobar que conforme el pulso es mayor, la frecuencia de la señal enviada tensión enviada al motor también es mayor.

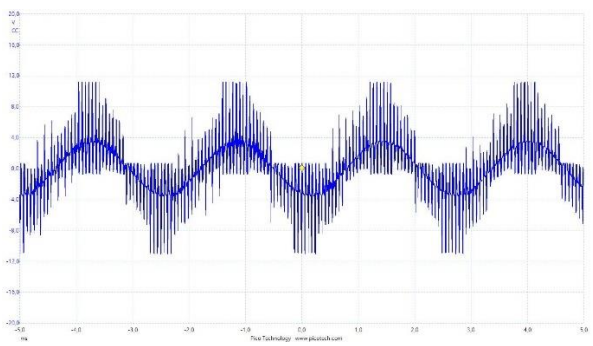

Figura 25: gráfica señal PWM 1100 ms

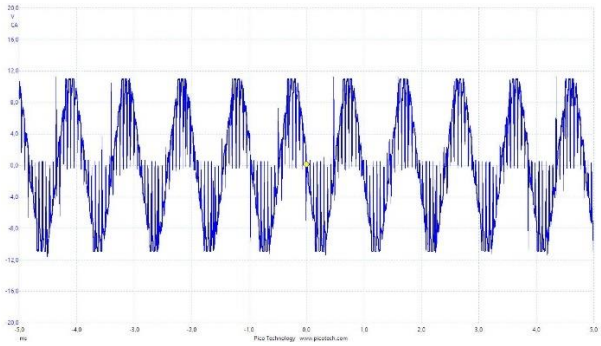

Figura 26: gráfica señal PWM 1500ms

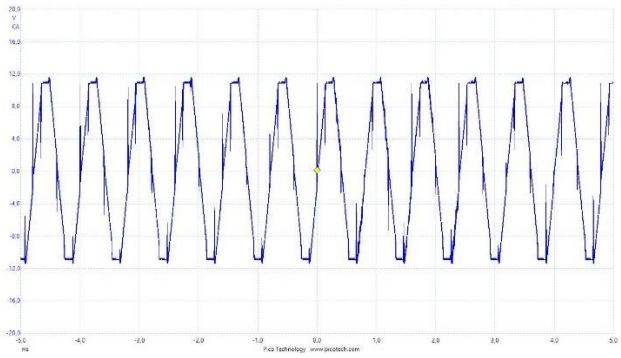

Figura 27: gráfica señal PWM 2000ms

6 CONCLUSIÓN

6.1 Futuros trabajos

Por límites de tiempos y de plazos no se han podido completar todos los trabajos deseados, por lo que, en un futuro, se seguirá trabajando en este proyecto. Lo primero en lo que se trabajará, será en el ajuste de los parámetros del PID, esto se hará con un soporte, con un solo grado de libertad, que permitirá que el dron solo gire en uno de los giros, roll, por ejemplo, y se ajustara el PID correspondiente a ese giro. Esto se repetirá con los tres giros posibles. Una vez ajustado los PID's por separado, se probará el dron en un soporte con tres grados de libertad y se comprobará que el dron reacciona bien a las perturbaciones.

El siguiente paso será añadir un barómetro para poder medir la altitud a la que se encuentra el cuadricóptero. Una vez implementado el barómetro, se podría añadir un cuarto controlador PID que controlaría el "thrust" o empuje para evitar que el dron suba sin control.

Otro plan de futuro es la integración de una cámara en el dron para poder ver en primera persona lo que el dron esté viendo en tiempo real. Además, Raspberry dispone de una cámara diseñada para la Raspberry-Pi la cual es bastante económica, pequeña y ligera. La imagen se enviaría por conexión WIFI hasta el dispositivo desde donde se controla el dron.

También se ha pensado en añadir un GPS, crear vuelos autónomos, o un sistema de detección de obstáculos. El dron se puede ir ampliando y mejorando hasta puntos impensables, las únicas limitaciones son el dinero, el tiempo y la imaginación.

6.2 Conclusión

Para finalizar, será realizado un minucioso repaso de los puntos más importantes del proyecto, así como serán revisados los objetivos planteados y analizados los resultados obtenidos, terminando con una breve conclusión personal.

Se empezó el proyecto con un exhaustivo estudio del funcionamiento de los drones, examinando los drones disponibles en el mercado y sus características. Se prosiguió con el diseño del cuadricóptero para su posterior montaje, que tuvo como resultado la obtención de un dron útil y versátil, cumpliendo el objetivo de bajo coste al seleccionar componentes económicos, pero de calidad. Respecto a la configuración de este, se ha diseñado un programa sencillo de usar, pero eficaz, con un bucle de control, con un controlador PID.

Durante el proyecto se han ido aplicando muchos de los conocimientos adquiridos a lo largo de la carrera, siendo consciente de su importancia en la aplicación práctica. Aunque no todo ha sido aplicar conocimientos, sino que también se han aprendido muchos nuevos como, por ejemplo, programación en lenguaje Python o técnicas de soldadura. Esto ha servido para extender el campo de conocimiento a otros sectores nuevos.

En cuanto a los objetivos se pueden dar todos por cumplidos ya que el proyecto se ha encuadrado en el marco económico y se han seleccionado los motores, ESC y hélices más adecuadas para el proyecto, así como la controladora de vuelo. También se ha conseguido integrar la Raspberry-Pi en el

dron, y se ha desarrollado en esta un programa para controlar el dron de forma básica, comunicándose con los distintos sensores. Asimismo, se ha logrado diseñar un algoritmo de control eficiente e implementarlo.

A nivel personal, este proyecto ha supuesto una carrera de fondo en esta última fase de mi etapa universitaria. Pese a las decenas de obstáculos encontrados y los numerosos impulsos de abandonar, ahora, una vez logrado el objetivo, echo la mirada atrás y me siento orgulloso y satisfecho del trabajo realizado. No obstante, esto no es un final, sino el comienzo para continuar trabajando en el dron y lograr un prototipo más avanzado, lo cual me va a permitir seguir aprendiendo y acercándome más a la realidad práctica de la ingeniería, pues el papel todo lo aguanta, mas cuando llega la hora de la verdad, todo cambia.

7 PRESUPUESTO

En la primera parte del presupuesto se incluirán los materiales necesarios para la creación este dron, teniendo en cuenta que todas las investigaciones y pruebas ya estarían realizadas.

Elemento	Descripción	Precio Unitario	Unidades	Precio
Raspberry-Pi	Controladora de vuelo	€ 35,00	1	€ 35,00
MPU 9250	Sensor inercial	€ 9,00	1	€ 9,00
PCA9685	PWM driver	€ 7,00	1	€ 7,00
DJI F450	Estructura del dron	€ 9,50	1	€ 9,50
A2212 1000KV	Motor brushless	€ 7,60	4	€ 30,40
QWinOut ESC 30A	Controladores electrónicos de velocidad	€ 4,50	4	€ 18,00
QWinOut 10x4,5 props	Hélices para los motores	€ 2,25	4	€ 9,00
Bridas	Bridas de plástico para amarrar los ESC	€ 0,02	100	€ 2,00
Cables	Cables macho/hembra para conexiones	€ 0,05	100	€ 5,00
Arandelas	Arandelas para ajustar las hélices	€ 0,02	10	€ 0,20
TOTAL:				€ 125,10

Se puede observar que el coste de total de los materiales asciende a 125,10€, lo cual es bastante económico teniendo en cuenta el resto de los drones que hay en el mercado con características similares.

En la siguiente parte del presupuesto se tienen en cuenta las horas requeridas para el diseño, montaje, instalación y pruebas, por parte de un ingeniero previamente formado. En esta parte no se contemplan las horas de estudio y aprendizaje precisadas.

COSTE DE INGENIERIA

Concepto	€ / hora	horas	Total
Estudio, diseño y selección de componentes	€ 80,00	10	800,00 €
Montaje de elementos	€ 60,00	5	300,00 €
Diseño de software (clases)	€ 80,00	30	2.400,00 €
Ensamblaje de funciones y clases	€ 80,00	10	800,00 €
Programación Rpi y conexión I/O	€ 80,00	5	400,00 €
Depuración y pruebas de funcionamiento	€ 80,00	5	400,00 €
TOTAL:			5.100,00 €

Si se suma el gasto en materiales y los honorarios de ingeniería se obtiene un coste final de cinco mil, dos cientos, veinticinco euros con diez céntimos (5225,10€).

8 Bibliografía

-330ohms. (2020, junio 22). ¿Qué diferencias hay entre una Li-Po y una Li-Ion? *330ohms*. https://blog.330ohms.com/2020/06/22/que-diferencias-hay-entre-una-li-po-y-una-li-ion/

-Alser. (s. f.). *PA66 30GF | Poliamida 66 con 30% Fibra de Vidrio—Plástico reciclado*. Recuperado 30 de junio de 2021, de https://www.plasticosalser.com/es/productos/pa66-30gf-poliamida-66-con-30-de-fibra-de-vidrio/

-Amazon. (s. f.). *AZDelivery PCA9685 16 Canales 12 bit PWM Controlador servo compatible con Raspberry Pi con E-Book incluido! : Amazon.es: Informática*. Recuperado 30 de junio de 2021, de https://www.amazon.es/

-Amazon. (s. f.). *BETAFPV Pavo30 Pusher Frame Kit Black PA12 Stiffener Brace of Carbon Fiber for 150X Series Motor SMO 4K Camera Pavo30 4S Brushless Whoop Drone Quadcopter: Amazon.es: Juguetes y juegos*. Recuperado 30 de junio de 2021, de https://www.amazon.es/

-Amazon. (s. f.). *QAV250 Marco, 250MM Quadcopter Aeronave Drone Frame Kit RC Accesorio para QAV250(Fibra de Carbon): Amazon.es: Juguetes y juegos*. Recuperado 30 de junio de 2021, de https://www.amazon.es/

-Amazon. (s. f.). *QWinOut F450 4-Axis Airframe 450mm Quadcopter Drone Frame Kit with 2-4S 30A RC Brushless ESC A2212 1000KV Brushless Motor 13T 1045 CW CCW Propellers: Amazon.es: Juguetes y juegos*. Recuperado 30 de junio de 2021, de https://www.amazon.es/ 9

-Amazon. (s. f.). *SparkFun IMU Breakout—MPU-9250: Amazon.es: Industria, empresas y ciencia*. Recuperado 30 de junio de 2021, de https://www.amazon.es/SparkFun-PID-13762-IMU-Breakout/dp/B01JQ79FZS

-Arduino. (s. f.). *Arduino Uno Rev3 | Arduino Official Store*. Recuperado 30 de junio de 2021, de https://store.arduino.cc/arduino-uno-rev3

-Arena, Q. (2015, marzo 29). *The history of drones and quadcopters*. Quadcopter Arena. https://quadcopterarena.com/the-history-of-drones-and-quadcopters/

-Batería de níquel-metalhidruro. (2021). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Bater%C3%ADa_de_n%C3%ADquel-metalhidruro&oldid=136619083

-Bejerano, P. (s. f.). *El uso de drones en agricultura | ToDrone*. Recuperado 30 de junio de 2021, de https://www.todrone.com/uso-drones-agricultura/

-BlueRobotics. (s. f.). *Basic ESC (Electronic Speed Controller) for Thrusters and Brushless Motors*. Recuperado 30 de junio de 2021, de https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/

-Dejan. (s. f.). *Arduino and MPU6050 Accelerometer and Gyroscope Tutorial*. Recuperado 30 de junio de 2021, de https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/

-Dejan. (s. f.). *How Brushless Motor and ESC Work—HowToMechatronics*. Recuperado 30 de junio de 2021, de https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

-Ergosum. (s. f.). *Senal-pwm.png (640×497)*. Recuperado 30 de junio de 2021, de https://www.programoergosum.com/images/cursos/255-salidas-analogicas-pwm-con-arduino/senal-pwm.png

-Fernández, Y. (2020, agosto 3). *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

-Foundation, T. R. P. (s. f.). *Buy a Raspberry Pi 3 Model B+*. Raspberry Pi. Recuperado 30 de junio de 2021, de https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

-Gluón. (2019, marzo 25). Filtro de Kalman: Deducción y ejemplos. *Lab. Gluón*. https://www.laboratoriogluon.com/filtro-de-kalman-deduccion-ejemplos/

-Krossblade. (s. f.). *History of Quadcopters and Multirotors*. Krossblade Aerospace Systems. Recuperado 30 de junio de 2021, de https://www.krossblade.com/history-of-quadcopters-and-multirotors

-Ltd, Xm. (s. f.). *Magnetómetro, definición y utilidad.* Recuperado 30 de junio de 2021, de https://www.xmind.net/m/GXxk7Y/

-Oliver, E. (2018, diciembre 20). *¿Conoces la historia de los drones? ¡Te la contamos!* Digital Trends Español. https://es.digitaltrends.com/drones/la-historia-de-los-drones/

-OnAir. (2021, enero 3). Todo lo que deberías saber antes de volar tu dron. *One Air*. https://www.oneair.es/normativa-drones-espana-aesa/

-Pastor, J. (2021, enero 21). *La Raspberry Pi Pico es un microcontrolador de 4 dólares con sorpresa: Un SoC propio diseñado por la Raspberry Pi Foundation*. https://www.xataka.com/accesorios/raspberry-pi-pico-microcontrolador-4-dolares-sorpresa-soc-propio-disenado-raspberry-pi-foundation

-Penalva, J. (2017, marzo 1). *La nueva Raspberry Pi Zero W es una Zero con WiFi y Bluetooth por solo 11 euros*. https://www.xataka.com/componentes/la-nueva-raspberry-pi-zero-w-es-una-zero-con-wifi-y-bluetooth-por-solo-11-euros

-Wikipedia. (2021). Dinámica del vuelo (aeronaves de ala fija). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Din%C3%A1mica_del_vuelo_(aeronaves_de_ala_fija)&oldid=134523749

-Wikipedia. (2021). Batería de ion de litio. En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Bater%C3%ADa_de_ion_de_litio&oldid=136019962

9 Anexos

9.1 Anexo 1: Código completo

```
import os
import time
from board import SCL, SDA #libreria necesaria para inicializar la PCA9685
import busio #libreria necesaria para inicializar la PCA9685
import Adafruit_PCA9685 #libreria para el driver PCA9685, para generación de PWM

import math #libreria necesaria para calcular los arctan
import smbus
import sys
sys.path.append("")
from mpu9250_jmdev.registers import * #importar modulo de la -imu
from mpu9250_jmdev.mpu_9250 import MPU9250 #importar clase de la imu
from imusensor.filters import kalman #importar modulo para aplicar filtro kalman

#creamos una clase PID, con los parámetros del PID, y una función que calcula la acción de control.
#con esta clase, crearemos los objetos para cada PID
class PID:
    def __init__(self, P, I, D):
        self.P=P
        self.I=I
        self.D=D
        self.err_ant=0
        self.Ui_ant=0
    def calculo_pid(self, ref, pos):
        global Ui_ant
        P=self.P
        I=self.I
        D=self.D
        Ui_ant=self.Ui_ant
        err_ant=self.err_ant
        err=ref-pos
        Up=P*err
        Ui=Ui_ant+I*err
        Ud=D*(err-err_ant)
        PID=Up+Ui+Ud
        #Anti-windup
        if(PID>=550 or PID<=-450):
            Ui=Ui_ant
            PID=Up+Ui+Ud
        self.err_ant=err
        self.Ui_ant=Ui
        return PID
#creando un objeto "mpu" con la clase MPU9250, de la libreria MPU9250, para las lecturas de los sensores
mpu = MPU9250(
```

```
54      address_ak=AK8963_ADDRESS,
55      address_mpu_master=MPU9050_ADDRESS_68, # In 0x68 Address
56      address_mpu_slave=None,
57      bus=1,
58      gfs=GFS_250,
59      afs=AFS_8G,
60      mfs=AK8963_BIT_16,
61      mode=AK8963_MODE_C100HZ)
62 #iniciando el objeto "sensorfusion" para fusionar las lecturas de
   la IMU con el filtro kalman
63 sensorfusion = kalman.Kalman()
64 #iniciando el ojeto pwm "para generar la señal pwm"
65 pwm = Adafruit_PCA9685.PCA9685()
66 #iniciando los objetos para cada PID con la clase PID
67 pidroll=PID(P=3, I=0, D=0)
68 pidpitch=PID(P=1, I=0, D=0)
69 pidyaw=PID(P=1, I=0, D=0)
70 #estableciendo la frecuencia de generacion de pwm a 50
71 pwm.set_pwm_freq(50)
72 tiempo_ini=time.time()
73 #definición de variables iniciales
74 posicion_gyro=[0.0, 0.0, 0.0]
75 posicion_accel=[0.0, 0.0, 0.0]
76 posicion=[0.0, 0.0, 0.0]
77 max_val=2000
78 min_val=1000
79 tiempo_ini=0
80 #indicamos los canales de cada motor
81 m1=3
82 m2=0
83 m3=1
84 m4=2
85 #inicial: función inicial, esta función calibra la IMU y los ESC,
   para calibrar el magnetometro, mover repetidamente la IMU 360°
86 def inicial():
87      mpu.configure()
88      mpu.calibrate()#calibrar IMU
89     mpu.configure()#aplicar calibracion
90      calibrate() #calibrar esc
91 #imu(): funciÃ³n que lee datos del sensor y los asigna a tres
   variables distintas
92 def imu():
93     gyro_read= mpu.readGyroscopeMaster()
94     accel_read= mpu.readAccelerometerMaster()
95     magn_read=mpu.readMagnetometerMaster()
96     return gyro_read, accel_read, magn_read
97 #función para calcular posicion aplicandu un filtro
   complementario
98 def posicion_filtro_complementario(tiempo):
99     global posicion
100       global posicion_accel
101       global posicion_gyro
102       coef=0.98
```

```
103       gyro_read= mpu.readGyroscopeMaster()
104       accel_read= mpu.readAccelerometerMaster()
105       posicion_gyro[0]+=(gyro_read[0]*tiempo)
106       posicion_gyro[1]+=(gyro_read[1]*tiempo)
107       posicion_gyro[2]+=(gyro_read[2]*tiempo)
108
109
   posicion_accel[0]=math.degrees(math.atan(accel_read[1]/(math.sqrt
   (accel_read[0]**2+accel_read[2]**2))))
110       posicion_accel[1]=-
   math.degrees(math.atan(accel_read[0]/(math.sqrt(accel_read[1]**2+
   accel_read[2]**2))))
111
   posicion_accel[2]=math.degrees(math.atan(math.sqrt(accel_read[1]*
   *2+accel_read[0]**2)/accel_read[2]))
112
113       posicion[0]= coef*(posicion_gyro[0])+(1-
   coef)*posicion_accel[0]
114       posicion[1]= coef*(posicion_gyro[1])+(1-
   coef)*posicion_accel[1]
115       posicion[2]= coef*(posicion_gyro[2])+(1-
   coef)*posicion_accel[2]
116       return posicion
117   #funcion para calcular la posicion aplicando un filtro kalman,
   de la libreria importada
118   def posicion_filtro_kalman(tiempo):
119       gyro_read, accel_read, magn_read=imu()
120       sensorfusion.computeAndUpdateRollPitchYaw(accel_read[0],
   accel_read[1], accel_read[2], gyro_read[0], gyro_read[1],
   gyro_read[2], magn_read[0], magn_read[1], magn_read[2], tiempo)
121       tiempo_ini=time.time()
122       print ("kalman roll", sensorfusion.roll, "pitch",
   sensorfusion.pitch, "yaw", sensorfusion.yaw)
123
124   def bucle_control(ref):
125         global posicion
126         global thrust
127         tiempo_ini=time.time()
128         t_0=time.time()
129         thrust=1400
130         archivo=open("datos.csv", "w")
131         try:
132             while True:
133                 tiempo=time.time()-tiempo_ini
134                 posicion=posicion_filtro_complementario(tiempo)
135         #se calculan las acciones de control
136                 ac_roll=pidroll.calculo_pid(ref[0], posicion[0])
137                 ac_pitch=pidpitch.calculo_pid(ref[1],
   posicion[1])
138                 ac_yaw=pidyaw.calculo_pid(ref[2], posicion[2])
139         #se calcula la señal pwm a aplicar a partir de las
   acciones de control calculadas
140                 vel_m1 = thrust + ac_roll - ac_pitch + ac_yaw
```

```
141                 vel_m2 = thrust - ac_roll - ac_pitch - ac_yaw
142                 vel_m3 = thrust - ac_roll + ac_pitch + ac_yaw
143                 vel_m4 = thrust + ac_roll + ac_pitch - ac_yaw
144         #se envia la señal pwm deseada a los actuadores
145                 set_pulse(m1, vel_m1)
146                 set_pulse(m2, vel_m2)
147                 set_pulse(m3, vel_m3)
148                 set_pulse(m4, vel_m4)
149                 tiempo_ini=time.time()
150         except KeyboardInterrupt:
151             print("error en el bucle de control")
152             stop()
153         finally:
154             archivo.close()
155   #funcion para convertir la señal a 16 bits, para mandarla por
    I2C al driver de generacion de pwm,
156   #usando una funcion de la libreria PCA9685
157   def set_pulse(channel, pulse):
158       pulse *= 4096
159       pulse //=20000
160       pulse=round(pulse)
161       pwm.set_pwm(channel, 0, pulse)
162   #main(): función del menu que se verá¡ por pantalla
163   def main():
164       global tiempo_ini
165       print ("/// calibrar // controlar // armar // stop //
    posicion // auto ///")
166       inp = input()
167       if inp == "calibrar":
168           calibrate()
169       elif inp == "armar":
170           arm()
171       elif inp == "controlar":
172           control()
173       elif inp == "stop":
174           stop()
175       elif inp == "auto":
176           ref=[0.0,0.0,0.0]
177           lista=["Roll","Pitch","Yaw"]
178           print("Indica una referencia")
179           for index, item in enumerate(lista):
180               print(item)
181               ref[index]=int(input())
182           bucle_control(ref)
183       elif inp == "posicion":
184           tiempo_ini=time.time()
185           try:
186               while True:
187                   tiempo=time.time()-tiempo_ini
188                   posicion_filtro_complementario(0.1)
189                   posicion_filtro_kalman(0.1)
190                   print("roll", posicion[0], "pitch",
    posicion[1], "yaw", posicion[2])
```

```
191                     tiempo_ini=time.time()
192                     tiempo=min(0.1, tiempo)
193                     time.sleep(0.1-tiempo)
194             except KeyboardInterrupt:
195                 print("volvemos al menu")
196                 main()
197
198         else :
199             print ("Te equivocaste, vuelve a empezar")
200             main()
201     #rutina para calibrar los ESC
202     def calibrate():
203         for i in range(4):
204             set_pulse( i, 0)
205         print("Desconecta y pulsa enter Enter")
206         inp = input()
207         if inp == '':
208             for i in range(4):
209                 set_pulse( i, max_val)
210             print("Conecta la baterÃa, oiras dos pitidos, y despues
    un sonido descendente, cuando pase, pulsa Enter")
211             inp = input()
212             if inp == '':
213                 for i in range(4):
214                     set_pulse( i, min_val)
215                 print ("Un poco de musica")
216                 time.sleep(7)
217                 print ("Aguanta un poco")
218                 time.sleep (5)
219                 print ("Un poco mÃ¡s, no seas impaciente.....")
220                 for i in range(4):
221                     set_pulse( i, 0)
222                 time.sleep(2)
223                 print ("Voy a armar los ESC, espera un poco
    mÃ¡s...")
224                 for i in range(4):
225                     set_pulse( i, min_val)
226                 time.sleep(1)
227                 print ("Listo!")
228                 main()
229     #funcion para parar todos los motores
230     def stop():
231         for i in range(4):
232             set_pulse( i, 0)
233         main()
234     #funcion para probar los motores, acelerar todos a la vez
235     def control():
236         print ("¡¡¡Arrancando motores!!!")
237         time.sleep(1)
238         speed = 1200
239         inp=1100
240         print ("a: acelerar mucho // d: decelerar mucho // q:
    acelerar un poco // e: acelerar un poco // s: stop")
```

```
241         try:
242             while True:
243                 for i in range(4):
244                     set_pulse( i, speed)
245                 inp = input()
246                 if inp == "d":
247                     speed -= 100
248                 elif inp == "a":
249                     speed += 100
250                 elif inp == "d":
251                     speed += 10
252                 elif inp == "e":
253                     speed -= 10
254                 elif inp == "s":
255                     stop()
256                     break
257                 else:
258                     print ("Te equivocaste de letra!! Pulsa a,q,d,e
    o s")
259         except KeyboardInterrupt:
260             print("Interrupción, volvemos al menu")
261             stop()
262     #armar los motores para poder controlarlos
263     def arm():
264         print ("Conecta la bateria y pulsa Enter")
265         inp = input()
266         if inp == '':
267             for i in range(4):
268                 set_pulse( i, 0)
269             time.sleep(1)
270             for i in range(4):
271                 set_pulse( i, max_val)
272             time.sleep(1)
273             for i in range(4):
274                 set_pulse( i, min_val)
275             time.sleep(1)
276             print("ESC armado")
277             main()
278
279     #las primeras acciones que se ejecutan con el programa, primero
    la funcion inicial para calibrar sensor y ESC
280     #despues de la inicializacion se ejecuta el menu principal
281     inicial()
282     main()
```