Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG:

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: León Enrique Prieto Bailo

DIRECTOR: Ramon Casanella Alonso

DATA: 7 de juliol del 2023

Contents

CAPÍTULO 1. INTRODUCCIÓN

En los últimos años, la industria de los drones ha experimentado un crecimiento exponencial, revolucionando diversos sectores y generando un impacto significativo en múltiples aspectos de nuestra sociedad. Los drones, también conocidos como sistemas aéreos no tripulados (UAS, por sus siglas en inglés), son dispositivos voladores controlados de forma remota que han evolucionado desde su uso militar inicial hasta convertirse en una tecnología accesible y versátil en manos de empresas, investigadores y aficionados.

Este crecimiento acelerado se ha visto impulsado por avances tecnológicos clave, como la miniaturización de componentes, la mejora de la autonomía de vuelo, la calidad de las cámaras y sensores, y la optimización de los sistemas de control. Estos avances han permitido que los drones se utilicen en una amplia gama de aplicaciones, desde la captura de imágenes y videos aéreos hasta la entrega de paquetes, inspecciones industriales, mapeo topográfico, agricultura de precisión y mucho más. En este contexto, el presente proyecto se enfoca en el diseño y desarrollo de un drone utilizando el frame DJI-F450 como plataforma y programando el microcontrolador Adafruit STM32F405 empleando el entorno de Arduino.

El drone desarrollado cuenta con una estructura robusta y liviana, fabricada en nylon y fibra de vidrio, lo que permite una mayor estabilidad y maniobrabilidad en vuelo. El drone incluye una variedad de sensores para hacer posible la operabilidad del cuadricóptero, entre ellos, un giroscopio, un acelerómetro, un barómetro, entre otros. La interoperabilidad de estos sensores juntamente con el microcontrolador, permiten volar el drone de manera estable y controlada lo cual cumple con el propósito inicial.

El propósito de esta tesis es construir un drone funcional mediante la cooperación activa de hardware y software de los componentes electrónicos, como sensores y motores, junto con el microcontrolador STM32F405. Esta combinación es fundamental para garantizar un vuelo estable y eficiente, donde los sensores recopilan datos del entorno y el microcontrolador controla los movimientos del drone a través de los motores u otros actuadores de los que pueda disponer.

En el contexto de este proyecto, el desarrollo del software representa el componente central y de mayor importancia. El enfoque principal radica en la creación de un sistema de control inteligente y eficiente, que permita al microcontrolador STM32F405 interactuar de manera óptima con los sensores y motores del drone. A través de la implementación de algoritmos y programas específicos, se busca lograr un vuelo estable, la recopilación precisa de datos y la capacidad de respuesta en tiempo real. El desarrollo del software constituye así la piedra angular de este proyecto, con el objetivo de garantizar el funcionamiento óptimo y seguro del drone.

CAPÍTULO 2. DISEÑO DE HARDWARE

2.1. Descripción detallada de la estructura del drone y los componentes utilizados en su montaje

En este proyecto de diseño de un drone, se han seleccionado los componentes necesarios para habilitar el vuelo del cuadricóptero. La elección del DJI F450 como estructura o frame se basa en su capacidad para personalizaciones extensas y su conveniente implementación de la electrónica. Esta elección permitirá alcanzar un equilibrio óptimo entre flexibilidad y eficiencia en el diseño del drone.

El microcontrolador escogido, el STM32F405 es un microcontrolador con una alta capacidad de procesamiento y que, por ello, es utilizado comúnmente como controladora de vuelo para aplicaciones de drones. En este caso, viene integrada en la placa Adafruit STM32F405. El papel principal del microcontrolador es el procesado digital de las señales que obtiene de los diferentes sensores electrónicos que posee el drone y, a partir de ellos, hacer los cálculos pertinentes para la operabilidad adecuada del drone. Este microcontrolador permite el desarrollo en el entorno de Arduino lo cual simplifica la programación de este sintácticamente y nos facilita el uso de librerías, tarjetas, etc.

La alimentación del sistema del cuadricóptero consta de una batería LiPo de 2250 mAh y 60C, la cual proporciona la energía suficiente para propulsar el drone un tiempo prolongado, así como para alimentar todo el sistema electrónico del drone. Este tipo de batería es ampliamente utilizado en aplicaciones de UAS debido a su alta densidad de energía y bajo peso. Es muy importante disponer de los conocimientos electrónicos para trabajar con este tipo de baterías ya que pueden proporcionar una enorme cantidad de corriente, lo cual puede dañar los elementos del circuito. Adicionalmente, es importante informarse sobre otras cuestiones de seguridad relacionadas a este tipo de batería, ya que una mala praxis puede conllevar a que la batería explote, pudiendo dañar los elementos del drone o incluso causando lesiones a quien esté operando con ella.

El sistema de estabilización del drone se basa en el uso de diferentes sensores. En particular, se ha incluido un sensor inercial MPU6050, que es capaz de medir la aceleración y la velocidad angular del drone. Este sensor permite al controlador de vuelo calcular la inclinación y la velocidad de rotación del drone en tiempo real, lo que es esencial para mantener el drone estable en vuelo. Además, se ha utilizado un barómetro BMP280, que mide la presión atmosférica y la temperatura ambiente. Estos datos son utilizados por el controlador de vuelo para ajustar la altura del drone y mantener una altitud

constante durante el vuelo lo cual es empleado cuando queramos realizar un control de tipo altitude hold.

Para el control del drone se ha utilizado una radio Flysky FS-i6. Esta radio se comunica con el controlador de vuelo mediante señales de radiofrecuencia y permite al usuario controlar el drone en vuelo. El control de la radio es esencial para realizar un control adecuado del drone, es el canal de comunicaciones entre el piloto y el drone y lo que permite al microcontrolador entender, en todo momento, la intención del piloto.

Los actuadores escogidos para este proyecto son los Tmotor AIR20A y los Tmotor 2213 los cuales se distribuyen como un kit y son compatibles con el frame seleccionado. Esta combinación de ESCs y motores permiten la propulsión del cuadricóptero y aseguran la potencia necesaria para poder realizar el vuelo del drone. Es importante que siempre que trabajemos con el drone y estemos probando funcionalidades e implementaciones, lo hagamos sin las hélices puestas para asegurarnos de que realmente tenemos el control de la aeronave y evitemos posibles lesiones en caso de descontrol.

2.1.1. FRAME DJI-F450

El Frame DJI-F450 es un frame creado por la empresa DJI diseñado ser utilizado en aplicaciones de cuadricópteros. Este frame tiene infinidad de aplicaciones como la fotografía y videografía aérea, el mapeo o la vigilancia entre otros. El marco está fabricado con la aleación PA66+30GF, la cual es una mezcla de poliamida 66 (nylon 66) con un 30% en peso de fibra de vidrio como refuerzo. La poliamida 66 es un polímero termoplástico que posee una alta resistencia mecánica, rigidez y resistencia al calor. La fibra de vidrio, por otro lado, se utiliza como material de refuerzo para mejorar las propiedades mecánicas del polímero.

El marco DJI-F450 está diseñado para ser fácilmente ensamblado y desmontado, lo que permite un fácil mantenimiento y reparación en caso de ser necesario además de ser económico y fácil de adquirir. Su tamaño compacto lo hace fácil de transportar y almacenar.

También es compatible con una amplia gama de componentes electrónicos, incluyendo controladores de vuelo, motores, ESC, baterías y otros equipos de radiocontrol. La compatibilidad con diferentes componentes hace que sea fácil personalizar y actualizar el drone según las necesidades específicas del usuario.

Además de las características mencionadas anteriormente, es importante destacar que el DJI-F450 es también un marco espacioso que permite añadir una multitud de componentes electrónicos. Este espacio adicional ofrece la flexibilidad necesaria para personalizar el drone según las necesidades del

usuario, permitiendo la integración de sistemas de posicionamiento global (GPS), sensores de distancia, iluminación LED y otros componentes que pueden mejorar la funcionalidad del drone.

Otra ventaja del Frame DJI-F450 es su diseño integrado de PCB, que permite la conexión segura y ordenada de los componentes electrónicos. Este diseño optimizado no solo simplifica el cableado de los ESC y la batería, sino que también proporciona un aspecto más limpio y profesional al drone.

2.1.2. Adafruit Feather STM32F405

El Adafruit STM32F405 es una placa de desarrollo que se basa en el microcontrolador STM32F405RG de STMicroelectronics. Este microcontrolador está diseñado para ofrecer un rendimiento óptimo y un bajo consumo de energía. Pertenece a la familia STM32F4 de la serie STM32, que está basada en la arquitectura ARM Cortex-M.

La placa de desarrollo de Adafruit se caracteriza por su facilidad de uso y su amplia gama de características, lo que la hace adecuada para una variedad de proyectos. En su núcleo, el STM32F405RG es un microcontrolador ARM Cortex-M4 de 32 bits, que opera a una frecuencia de 168 MHz y tiene una memoria RAM total de 192KB. Además de su capacidad de punto flotante, también incluye numerosos periféricos integrados.

La conectividad es una de las fortalezas del Adafruit STM32F405. La placa cuenta con puertos USB-C, UART, SPI, I2C y GPIO, que permiten una fácil comunicación con otros dispositivos. Los pines operan a una tensión de 3,3 V aunque la mayoría también admiten hasta tensiones de 5 V. También incluye un conector microSD para almacenamiento externo, brindando opciones adicionales para la expansión de memoria.

Una de las ventajas de utilizar el Adafruit STM32F405 es su compatibilidad con el entorno de desarrollo Arduino. Adafruit ha desarrollado una biblioteca que permite programar el microcontrolador utilizando el lenguaje de programación y las herramientas familiares de Arduino. Esto simplifica el proceso de desarrollo

y programación, especialmente para aquellos que ya están familiarizados con el ecosistema de Arduino.

Además de las características principales, la placa Adafruit STM32F405 incluye componentes adicionales como un chip SPI Flash de 2MB y un led RGB. Estos componentes ofrecen funcionalidades adicionales y expanden las posibilidades de proyectos que se pueden desarrollar con la placa.
Con una amplia variedad de pines de entrada/salida (E/S), el Adafruit STM32F405 brinda una gran flexibilidad para conectar y controlar dispositivos y periféricos externos. Esto permite la integración de una amplia gama de sensores, actuadores y otros dispositivos en los proyectos.

2.1.3. MPU6050

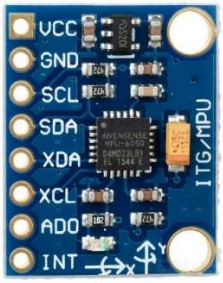
El MPU6050 es un módulo de sensor de movimiento utilizado para medir la aceleración y la velocidad angular en dispositivos electrónicos. Este módulo es especialmente popular en aplicaciones de robótica, drones, control de movimiento y realidad virtual.

El MPU6050 integra un acelerómetro de tres ejes y un giroscopio de tres ejes en un solo chip. Esto permite medir la aceleración lineal y la velocidad angular en tres direcciones diferentes: X, Y y Z. El acelerómetro mide la aceleración lineal en cada uno de los ejes, mientras que el giroscopio mide la velocidad angular o la tasa de cambio del ángulo en cada eje.

Una de las características destacadas del MPU6050 es su capacidad para proporcionar mediciones en tiempo real con alta precisión y sensibilidad. Esto permite detectar movimientos y cambios de orientación con gran precisión, lo que resulta útil en aplicaciones como la estabilización de vuelo de drones, la detección de movimientos en juegos de realidad virtual y la navegación inercial en robótica.

Además de sus capacidades de medición, el MPU6050 también incluye características adicionales como un sensor de temperatura incorporado o la capacidad de ajustar diferentes rangos de medición y tasas de muestreo según las necesidades del proyecto.

Para utilizar el MPU6050 con el microcontrolador Adafruit Feather STM32F405, se requiere la programación y configuración adecuada. Para establecer la conexión entre el MPU6050 y el microcontrolador, se utilizará el puerto I2C (Inter-Integrated Circuit). El puerto I2C permite una comunicación sencilla y eficiente entre dispositivos, y es compatible con el MPU6050. Mediante la programación adecuada, el microcontrolador podrá recibir datos del MPU6050 y realizar las acciones necesarias en función de las mediciones de aceleración y velocidad angular proporcionadas por el sensor. Esto permite aprovechar las capacidades del MPU6050 en aplicaciones como estabilización de vuelo, control de movimiento y detección de orientación precisa.



2.1.4. BMP280

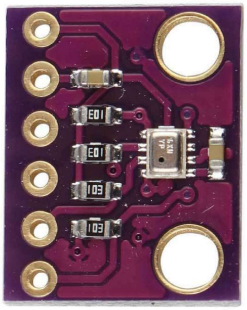
El sensor BMP280 es un sensor de presión y temperatura de alta precisión fabricado por Bosch Sensortec. Está diseñado para medir la presión barométrica y la temperatura en una amplia gama de aplicaciones, incluyendo la navegación, los drones y la meteorología. El BMP280 utiliza un principio de medición piezo-resistivo para medir la presión atmosférica con una precisión de hasta ±1 hPa, lo que lo hace ideal para aplicaciones en las que se necesita una medición precisa de la presión.

En cuanto a su aplicación en drones, el BMP280 se utiliza para medir la altura y la altitud del drone. Al medir la presión atmosférica, el BMP280 puede calcular la altitud del drone con una precisión razonable. Esta información complementada con un algoritmo de control de altitud se puede emplear para desarrollar modos de vuelo como el “altitude hold”, el cual permite maniobrar el drone a una altura constante, o el control de vuelo por GPS el cual permite mantener una posición en el espacio de manera constante.

Para los drones contemporáneos es prácticamente obligatorio que dispongan de lecturas de presión para aplicar este tipo de control sobre la altitud del drone. Todo esto tiene aplicaciones para cuadricópteros que realizan misiones de vigilancia, mapeo y fotografía aérea, ya que les permite mantener una altitud constante y controlar su posición con mayor precisión.

En este proyecto en cuestión el sensor BMP280 se utiliza en combinación con la MPU6050, con el propósito de implementar, de manera viable, un modo de

vuelo “altitude hold” que permita al cuadricóptero mantenerse a una altura constante y moverse, en un plano horizontal, utilizando el joystick que controla roll, pitch y yaw. Para hacerlo, el controlador de vuelo dispondrá, en su algoritmo de control, de diferentes controladores PID que reaccionaran de manera independiente a las perturbaciones obtenidas por los sensores con el fin de realizar las correcciones pertinentes.



2.1.5. FlightSky i6

La Flysky i6 es un radiocontrol de 6 canales diseñada para el pilotaje de modelos de radiocontrol, como aviones, helicópteros, drones y otros dispositivos RC. Es una radio muy popular entre los entusiastas de los vuelos de radiocontrol debido a la simplicidad de su manejo, versatilidad y precio asequible.

La Flysky i6 cuenta con una pantalla LCD retroiluminada y un diseño ergonómico que la hace cómoda de sostener y utilizar durante largos períodos de tiempo. Ofrece una interfaz intuitiva con botones de fácil acceso y un menú de navegación sencillo, lo que facilita la configuración y personalización de los ajustes.

Una de las principales características de la Flysky i6 es su capacidad de operar usando el protocolo AFHDS 2A, que proporciona una comunicación de radio con frecuencia estable y confiable. Esto garantiza una respuesta precisa y rápida entre la emisora y el receptor, lo que resulta crucial para un control preciso del UAS.

El radiocontrol Flysky i6 ofrece una amplia gama de funciones y ajustes, como la asignación y mezcla de canales, la configuración de puntos de ajuste y límites, la selección de modos de vuelo (modo estabilizado, modo altitude hold, etc.) y la programación de diferentes modelos.

Adicionalmente, la Flysky i6 es compatible con varios receptores Flysky, lo que brinda flexibilidad para adaptarse a diferentes modelos y configuraciones. También ofrece la posibilidad de actualización de firmware, lo que permite agregar nuevas funciones y mejoras a medida que estén disponibles.

Respecto al procesado de la señal desde el microcontrolador, la Flysky es una radio que utiliza la modulación PPM (Pulse Position Modulation) para transmitir la información de control desde la emisora al receptor. La modulación PPM es un método eficiente que permite enviar múltiples señales de control a través de un solo cable, lo que ahorra el uso de pines del microcontrolador en el receptor. En lugar de tener un cable separado para cada canal de control, la Flysky utiliza una única señal PPM que lleva consigo la información de todos los canales de control en forma de pulsos de diferentes longitudes. Esto simplifica la conexión entre la emisora y el receptor, liberando pines en el microcontrolador para otros usos. Además, la modulación PPM permite una transmisión rápida y precisa de la información de control, lo que contribuye a una mayor eficiencia y capacidad de respuesta en el sistema de radiocontrol.

2.1.6. LiPo

Las baterías LiPo (Lithium Polymer) son una fuente de energía comúnmente utilizada en los drones y otros dispositivos electrónicos portátiles. Se caracterizan por tener una alta densidad de energía y de dimensiones y peso reducido, lo que significa que pueden proporcionar una gran cantidad de energía en un paquete pequeño y ligero.

En el caso específico del proyecto de drone que estamos discutiendo, utiliza una batería LiPo de la marca SUNPADOW de 3 celdas (3s), lo que significa que contiene tres celdas de batería en serie. Cada celda proporciona una tensión nominal de 3.7V, lo que significa que la batería completa tiene una tensión nominal de 11.1V. Esta batería tiene una capacidad de 2250 mAh, lo cual es una cantidad de energía suficiente para operar el drone.

La tasa máxima de descarga de la batería es de "60C", este valor hace referencia al factor máximo de corriente que podemos llegar a pedir a la batería, esto que significa que puede descargar energía a una tasa de hasta 60 veces su capacidad nominal (en este caso, 60 x 2.25 A = 135 A) sin sufrir daños significativos. Esto es realmente importante y es uno de los motivos por

los cuales este tipo de baterías se utilizan tanto en el sector de los drones ya que la batería debe ser capaz de suministrar suficiente corriente a los motores para mantener el vuelo.

La batería es enchufada al drone utilizando el conector XT60. Este es un tipo de conector diseñado específicamente para baterías LiPo y son muy habituales junto con los conectores XT30, JST o XT90. Se caracteriza por ser fácil de conectar y desconectar, seguro y resistente a altas corrientes.

Además, es importante tener en cuenta ciertas precauciones cuando se opera con este tipo de baterías ya que una mala praxis puede inutilizar la batería o producir lesiones en el usuario. Una de las pautas a seguir es realizar las cargas con un cargador que tenga funciones de seguridad incorporadas como detección de voltaje máximo y capacidad de apagado automático y realizar una supervisión de la carga. También es conveniente realizar inspecciones regulares en busca de hinchazones, daños o ver si se calienta en exceso después de utilizarla en operación.

2.1.7. ESCs, motores y propellers.

Las ESCs (Electronic Speed Controllers) son dispositivos electrónicos que se utilizan para controlar la velocidad de los motores eléctricos en los drones. Las ESCs se conectan directamente a la batería del drone y a los motores, y utilizan señales de control provenientes del controlador de vuelo para ajustar la velocidad de los motores y así controlar la altura, la velocidad y la dirección del drone. Las ESC, son esencialmente, conversores DC-AC regulables, estos se encargan de convertir la tensión continua de las baterías en tensión alterna, utilizando una señal que proviene de la controladora de vuelo. Esta señal recibida del controlador de vuelo afecta al ancho de pulso de la señal generada por las ESC, lo que regula la velocidad de los motores para realizar el control de vuelo.

En este proyecto se han escogido las Tmotor AIR20A, estas son ESCs de alta calidad diseñadas específicamente para vehículos multirrotores sin BEC (Battery Elimination Circuit). Estas ESCs son capaces de proporcionar una alta

corriente de salida y están diseñadas para ser muy eficientes en términos de energía. Además, las AIR20A son compatibles con una gran variedad de controladores de vuelo y software de programación, lo que las hace muy versátiles y fáciles de integrar en cualquier proyecto de drone.

Los motores escogidos para este proyecto son los Tmotor AIR 2213. Estos motores eléctricos son ideales para cuadricópteros de 1200 g a 1500 g. Son motores de respuesta rápida y de poco ruido, son compatibles con el frame y fáciles de instalar en conjunto con las ESC. Dos de ellos deben girar en sentido CW y los otros dos en sentido CCW. Estos motores son de 920 KV lo que significa que por cada voltio que apliquemos a los motores obtendremos 920 revoluciones por minuto, en combinación con la LiPo de 12,6 V de tensión máxima (4,2 V/celda * 3 celda) los motores pueden llegar a girar a un ritmo de 11592 RPM.

Los propellers escogidos son los T9545, son motores de 9,5 pulgadas de diámetro y con un paso de 4,5 pulgadas. Estos propellers, vienen con una cabecera de metal de auto-bloqueo la cual esta hecha para asegurar que, durante el vuelo, no se aflojan los propellers. Esta cabecera de auto-bloqueo es simplemente el diseño de una rosca que se ajusta en la dirección contraria a la que giran los motores lo que prevee de la posibilidad de que la rosca se salga.
Los propellers escogidos son los T9545, que son propellers de 9,5 pulgadas de diámetro y tienen un paso de 4,5 pulgadas. Estos propellers vienen equipados con una cabecera de metal de auto-bloqueo, diseñada para garantizar que los propulsores no se aflojen durante el vuelo. La cabecera de auto-bloqueo consiste en una rosca que se ajusta en sentido contrario al giro de los motores, evitando así que la rosca se desenrosque.

Todos estos elementos se pueden obtener en forma de kit y son compatibles entre ellos además de serlo con los demás elementos del drone como el frame y la controladora de vuelo.

2.2. Esquemas y diagramas que muestren la conexión de los componentes

CAPÍTULO 3. DISEÑO DE SOFTWARE

La parte de software en sistemas de control es crucial para el correcto funcionamiento de cualquier sistema, incluyendo drones. El algoritmo de control es la pieza central del software, ya que es el encargado de calcular la señal de control que se enviará a los actuadores para lograr que el drone actúe en la dirección deseada.

El algoritmo de control típicamente se compone de varias partes: generación de referencias, adquisición de datos, procesamiento de datos, cálculo de errores, cálculo de la señal de control, y envío de la señal de control a los actuadores. Cada una de estas partes es importante y debe estar diseñada para trabajar en conjunto de manera eficiente y precisa.

La generación de referencias se encarga de determinar la trayectoria deseada del drone, basándose en los objetivos del sistema. Esta trayectoria puede ser definida en términos de velocidad, aceleración, posición, orientación, entre otros. Es importante tener en cuenta que las referencias pueden cambiar constantemente en tiempo real, por lo que se requiere de un algoritmo que pueda actualizarlas de forma rápida y precisa.

La adquisición de datos es la parte del software que se encarga de leer los sensores, en el caso del drone, la MPU6050 y el BMP280. Estos sensores proporcionan información sobre la posición vertical, velocidad, orientación, aceleración, entre otros parámetros, que son necesarios para el cálculo de la señal de control. Es importante que los datos adquiridos sean precisos y se actualicen a una tasa adecuada para evitar errores en el cálculo del control.
El procesamiento de datos es la parte del software que se encarga de analizar los datos adquiridos y prepararlos para su uso en el cálculo del control. Esto puede incluir la conversión de unidades, el filtrado de señales, la eliminación de ruido, la calibración de los sensores, entre otros.

El cálculo de errores es una parte fundamental del algoritmo de control. Esta parte se encarga de comparar las referencias con los datos adquiridos, y determinar la diferencia entre ellos. Esta diferencia se conoce como error, y es la base del cálculo de la señal de control. Es importante que el cálculo del error sea preciso y actualizado constantemente.

El cálculo de la señal de control es la parte del algoritmo que se encarga de determinar la señal que se enviará a los actuadores para lograr la trayectoria deseada. Este cálculo se realiza mediante algoritmos de control, como los controladores PID, que ajustan la señal de control en función del error calculado y de otros parámetros del sistema.

Finalmente, la señal de control es enviada a los actuadores, que son los encargados de mover el drone en la dirección deseada. En el caso del drone

mencionado, los actuadores son los motores que varían su velocidad en función de señal de control procesada por las ESCs .

El software de un sistema de control es una parte fundamental para el correcto funcionamiento del sistema. Requiere de algoritmos de control precisos y eficientes, así como de una arquitectura adecuada para asegurar que todas las partes trabajen en conjunto de forma óptima.

3.1. Arquitectura del software

Es fundamental establecer una estructura bien definida para implementar el algoritmo de control del dron, asegurando que los cálculos internos se realicen de manera procedimental y ordenada. Esto resulta de gran utilidad a la hora de identificar posibles errores, fallos de implementación u otras situaciones similares, ya que facilita el proceso de diagnóstico gradual del código.

Desde la perspectiva del desarrollador, una arquitectura de software bien diseñada agiliza tanto la implementación inicial como la posterior modificación de funcionalidades en el código. En el caso de que otros desarrolladores muestren interés en el proyecto o deseen colaborar en él, una sólida arquitectura de software facilita considerablemente la comprensión del código y permite que desarrolladores externos se familiaricen rápidamente con él.

La arquitectura de software seguida para implementar el algoritmo de control se basa en una estructura modular. Trabajar con estructuras modulares presenta múltiples ventajas, ya que permite al desarrollador incorporar nuevas funcionalidades en el sistema de manera sencilla. Una estructura modular sigue el concepto de poder añadir y mover módulos con facilidad cuando sea necesario. En este caso, se realizará una llamada a los módulos principales, que consisten en funciones ubicadas en archivos separados. Estos módulos, a su vez, se dividen en subrutinas adicionales las cuales se llaman desde la ejecución interna del módulo principal.

Por ejemplo, existe un módulo que se encarga de recopilar toda la información de los sensores y realizar el procesado necesario de las señales para poder obtener información. El bucle principal llama a la función ubicada en el archivo correspondiente que a su vez llama a otras subrutinas ubicadas en este mismo archivo que se encargan de, por ejemplo, leer y procesar la IMU. Esto, como se puede ver, es realmente ventajoso a la hora de implementar nuevos sensores u otras funcionalidades ya que solo hace falta añadir las variables globales en la cabecera del archivo principal y añadir una función que simplemente se encargue de realizar la lectura y el procesado de los datos necesarios, sin afectar al resto del código.

Esta arquitectura no solo proporciona ventajas al implementar nuevas funcionalidades, sino que también facilita la comprensión del funcionamiento del código en general y la segmentación de las operaciones realizadas en el algoritmo de control.

[AQUÍ QUIERO AÑADIR ALGO MAS]
A continuación, se detalla de manera procedimental y general, la estructura del código:

3.1.1. Tareas de inicialización

3.1.1.1. Declaración de variables globales utilizadas por el código.

En general, con proyectos basados en Arduino C, y para este tipo de proyectos donde cada método requiere de bastantes inputs y outputs diferentes y se opta por una estructura modular, lo más sencillo es trabajar con variables globales donde se tiene acceso a las variables desde cualquier punto del código. Adicionalmente, el hecho de tener variables globales facilita la implementación de los módulos de código ya que estas son accesibles en cualquier método. Por lo tanto, se ha decidido usar este tipo de instanciación para las variables del código.

3.1.1.2. Inicialización del software.

Lo primero que se hace al ejecutar cualquier software basado en Arduino es ejecutar la función setup. Esta función, principalmente, se encarga de todas las tareas que hay que ejecutar una sola vez al principio de la ejecución. Aquí encontraríamos tareas como el calibraje de la imu, la detección de la radio, la vinculación de interrupciones de hardware o la conexión serial para realizar tareas de depuración.

3.1.2. Bucle principal

En el bucle principal encontraremos la ejecución reiterativa de las funciones principales que segmentan el código en sus respectivos módulos. En estas funciones se encontrarán las tareas de cálculo de referencias, obtención de medidas de los sensores, cálculo de la señal de error y posterior cálculo de los PID y finalmente adaptación de la señal para los actuadores del cuadricóptero. Adicionalmente también hay un módulo encargado de realizar tareas de diagnóstico para depurar y un sistema de control de tiempo de ejecución de bucle.

3.1.2.1. Cálculo de referencia: reference_computation()

En este primer módulo, se encuentra toda la información relacionada con las referencias que se deben utilizar para llevar a cabo la ejecución adecuada del algoritmo de control. Esto incluye la definición de los modos de vuelo del cuadricóptero y la lógica necesaria para realizar las transiciones entre ellos.

3.1.2.2. Lectura de los sensores y procesado de la señal: read_process_units()

Este módulo se encarga de leer las señales provenientes de los sensores y de los elementos de hardware externos para llevar a cabo el procesamiento necesario. Su función principal consiste en establecer las conexiones adecuadas con los sensores y obtener las lecturas necesarias para adaptar las señales recibidas a la ejecución del algoritmo de control.

3.1.2.3. Cálculo del error de control y PIDs: controllers()

Este módulo se encarga de comparar las señales correspondientes a la referencia y la medida, calcular el error de control y aplicar los algoritmos PID. Es el núcleo del algoritmo de control, ya que determina cómo responder a los errores en las magnitudes que deseamos controlar.

3.1.2.4. Generación de la señal para los actuadores: actuators()

En este último módulo del algoritmo de control se integran las salidas de los PIDs para lograr el control deseado. En esta etapa, se aplica la lógica que determina qué señales deben utilizarse en función del modo de vuelo del dron, y se generan las señales correspondientes para los actuadores.

3.1.2.5. Herramientas de depuración y diagnóstico: diagnostics()

Este módulo, independiente del algoritmo de control, se encarga de varias tareas relacionadas con la depuración. Su funcionalidad resulta útil para detectar fallos en el algoritmo de control, visualizar señales y analizar el comportamiento de sensores y actuadores, entre otras funcionalidades.

3.1.2.6. Sistema de control de duración del bucle.

Aunque la ejecución del bucle principal del código siempre involucre la repetición de los mismos módulos de código, no siempre se ejecuta con la misma rapidez. Esto puede deberse a situaciones específicas en las que se realizan cálculos más complejos o se requiere un tiempo adicional para acceder a los registros de alguno de los sensores.

No obstante, es crucial que los algoritmos de control se ejecuten a una frecuencia constante para garantizar la estabilidad del drone. Para abordar esta necesidad, al final del bucle principal se incluye un fragmento de código encargado de mantener la duración del bucle de manera uniforme.

Este fragmento de código se implementa con el propósito de ajustar el tiempo de espera o realizar acciones adicionales para compensar cualquier variación en la duración del bucle principal. Su objetivo es mantener una frecuencia de ejecución constante, independientemente de las circunstancias que puedan afectar la velocidad de ejecución del código.

Esta práctica asegura que el algoritmo de control se ejecute de manera consistente y predecible, lo que resulta fundamental para lograr una operación estable y segura del drone.

[AÑADIR ESQUEMA?]

3.2. Desarrollo del software

3.2.1. Inicialización

EXPLICACIÓN DEL SOFTWARE (DISCLAIMER): Todo el código empleado para el desarrollo de esta tesis se podrá hallar anexo a esta memoria. Se presupone que el lector de esta memoria tiene nociones básicas de programación de microcontroladores y desarrollo en el entorno Arduino. Esto significa que en la descripción del software a continuación no se explicará cada una de las líneas que aparecen en el código si no las funcionalidades principales y cuál es la lógica operacional detrás de ellas. Adicionalmente, se recomienda la lectura paralela de esta sección juntamente con el código para entender adecuadamente la funcionalidad, el orden de ejecución y la arquitectura del software.

El primer paso que debe realizar el algoritmo de control una vez conectado es configurar, a nivel de software, todos los parámetros e inicializar todas las variables adecuadamente, a lo largo de esta sección revisaremos detalladamente el proceso que se sigue para realizar el calibraje del drone. Todas estas operaciones se ejecutan una sola vez y están situadas fuera del bucle principal, en el código desarrollado estas operaciones pertenecen a la función `void setup() {}`.

Dentro de esta función, hallamos la función `init_components()`. Esta función se encarga de ejecutar los diferentes módulos necesarios para la inicialización del software. Dentro de ella encontramos llamadas, de forma secuencial, a las siguientes funciones:

- `init_flash()`: Esta función, esencialmente, se encarga de iniciar la comunicación con el adaptador de tarjetas SD disponible en la placa desarrollada por Adafruit. Este método no tiene ningún impacto directo sobre la funcionalidad del algoritmo de control y, durante el desarrollo del proyecto, se empleó para realizar tareas de diagnóstico / almacenaje de datos.
- `init_led()`: método que, simplemente, prepara el PIN conectado al LED para poder controlarlo con el microcontrolador.
- `init_ultrasonic(): // DEPRECATED?`
- `init_rc()`: Este método se encarga de inicializar la conexión con el dispositivo receptor de la radio. Simplemente configura el PIN de entrada de la señal PPM de la radio y le adjunta una interrupción de hardware que llama al método read_PPM, encargado de leer y procesar la señal de la radio.
- `init_esc():` Este método inicializa las señales PWM que reciben las ESC para realizar el control de velocidad de rotación de los motores. Para controlar adecuadamente los timers del microcontrolador, se utiliza la librería HardwareTimer de stm32duino. Es necesario el uso de los timers del microcontrolador ya que esto permite ahorrar el tiempo que supone generar las señales PWM de manera manual.

- init_gyro(): Este método, se encarga de realizar el calibraje necesario para preparar la MPU6050 para su uso. Para hacerlo, utiliza la librería Wire para realizar la conexión I2C con el chip. Las primeras líneas del metodo se encargan de iniciar la comunicación entre el microcontrolador y el dispositivo utilizando una dirección de 8 bits que identifica al dispositivo.
  Una vez la conexión se ha demostrado que es satisfactoria, se configuran los valores de cuatro registros diferentes de 8 bits del sensor. Esta acción permite configurar ciertos parametros.
  Registro 107 (0x6B): Nos permite activar y configurar el modo de operación del sensor.

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Configuramos cada bit siguiendo la descripción del datasheet:

Bit	Descripción	Valor
7	Reinicia el dispositivo	0
6	Modo de suspensión de bajo consumo	0
5	modo de ciclo desactivado	0
4	-	0
3	Termómetro deshabilitado	0
2	Oscilador de 8 MHz seleccionado.	0
1		0
0		0

Escribimos el registro mediante la librería wire:

```
Wire.beginTransmission(gyro_address);
Wire.write(0x6B);
Wire.write(0x00);
Wire.endTransmission();
```

Y repetimos el mismo proceso para el resto de los registros:
Registro 27 (0x1B): Nos permite establecer la configuración del giroscopio.

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]				

Bit	Descripción	Valor
7	Prueba autónoma del giroscopio en el eje X deshabilitada	0
6	Prueba autónoma del giroscopio en el eje Y deshabilitada	0
5	Prueba autónoma del giroscopio en el eje Z deshabilitada	0
4	Rango completo del giroscopio de 500 dps	0
3		1
2	-	0
1	-	0
0	-	0

Registro 28 (0x1C): Nos permite establecer la configuración del acelerómetro.

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				

Bit	Descripción	Valor
7	Prueba autónoma del acelerómetro en el eje X deshabilitada	0
6	Prueba autónoma del acelerómetro en el eje Y deshabilitada	0
5	Prueba autónoma del acelerómetro en el eje Z deshabilitada	0
4	Rango completo del giroscopio de +-8g	1
3		0
2	-	0
1	-	0
0	-	0

Registro 26 (0x1A): Nos permite activar el filtro paso bajo digital integrado en el sensor:

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Bit	Descripción	Valor
7		0
6		0
5	External Frame Synchronization not used	0
4		0
3		0
2	Frecuencia de corte del filtro paso bajo ~43 Hz	0
1		1
0		1

Una vez se ha configurado adecuadamente el sensor, es hora de calibrarlo. Para hacerlo, se obtienen 2000 lecturas del sensor y se hace un promedio para, posteriormente, extraer estos valores de las futuras lecturas. Esto se hace para fijar como valores de referencia los valores que nos provee el acelerómetro al principio de la ejecución del código.

- `init_barometer_v2()`: Este método, se encarga de realizar el calibraje necesario para preparar el barómetro BMP280 para su uso. Para hacerlo, utiliza la librería Wire para realizar la conexión I2C con el chip, en conjunto con la MPU6050. Las primeras líneas del método se encargan de iniciar la comunicación entre el microcontrolador y el dispositivo utilizando una dirección de 8 bits que identifica al dispositivo. Una vez la conexión se ha demostrado que es satisfactoria, se configuran los valores de cuatro registros diferentes de 8 bits del sensor. Esta acción permite hacer lecturas de los parámetros de compensación y configurar el sensor.
  Como se ha mencionado anteriormente, el sensor es tremendamente sensible a los cambios de temperatura y es por eso que es necesario

hacer correcciones de las lecturas del sensor para poder utilizarlas como referencia para el algoritmo de control. Para realizar las debidas correcciones se emplean unos parámetros de compensación que vienen integrados en los registros de lectura del microcontrolador.

Estos parámetros de compensación son 12 valores de 16 bits, 3 correcciones de temperatura y 9 de presión. El datasheet del BMP280 nos dice que estos valores se encuentran a partir del registro 136 (0x88) en adelante por lo que se hace una solicitud de 24 bytes y se almacenan en las variables para luego utilizarlos para las correcciones.

```
Wire.beginTransmission(BMP280_ADDRESS);
Wire.write(0x88);
Wire.endTransmission();
Wire.requestFrom(BMP280_ADDRESS, 24);
dig_T1 = Wire.read() | Wire.read() << 8;
dig_T2 = Wire.read() | Wire.read() << 8;
dig_T3 = Wire.read() | Wire.read() << 8;
dig_P1 = Wire.read() | Wire.read() << 8;
dig_P2 = Wire.read() | Wire.read() << 8;
dig_P3 = Wire.read() | Wire.read() << 8;
dig_P4 = Wire.read() | Wire.read() << 8;
dig_P5 = Wire.read() | Wire.read() << 8;
dig_P6 = Wire.read() | Wire.read() << 8;
dig_P7 = Wire.read() | Wire.read() << 8;
dig_P8 = Wire.read() | Wire.read() << 8;
dig_P9 = Wire.read() | Wire.read() << 8;
```

Adicionalmente, aplicamos la configuración del barómetro para el tipo de operación que vamos a realizar. Esto se hace de la misma manera que se hacia con la MPU6050, modificando los registros de escritura destinados a la configuración.

Registro 244 (0xF4): Nos permite configurar el modo de operación del barómetro y el oversampling de temperatura y presión:

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
F4	244	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]	

Bit	Descripción	Valor
7	Temperature oversampling: x2	0
6		1
5		0
4	Pressure oversampling: x16	1
3		0
2		1
1	Mode: Normal	1
0		1

Registro 245 (0xF5): Nos permite configurar el rate, filtro y las interfaces del dispositivo.

Registo (Hex)	Registo (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
F5	245	t_sb [2:0]			filter[2:0]			-	spi3w_en[0]

Bit	Descripción	Valor
7	Normal Mode standby duration: 0,5 ms	0
6		0
5		0
4	IIR filter time constant: 16	1
3		0
2		0
1	-	0
0	No SPI	0

Los parámetros de compensación

Esta acción permite configurar ciertos parametros.

-

Explicar en detalle la inicialización del código, aquí hay que añadir toda la información acerca de calibraje, detección de rc…
¿Cuestiones a tener en cuenta? Posiblemente sea interesante hablar del failsafe también entre otras cosas, a nivel de prevención.
Podría ser interesante hablar de la enum de los modos de vuelo. Good idea!
Checar código en general

3.2.2. Reference_computation()

El método “reference_computation()” se encarga de generar las referencias necesarias para el cuadricóptero. La idea general de este módulo es segmentar la generación de las referencias en función del modo de vuelo que se opere y que en el resto de ejecución del algoritmo de control, no aparezcan distinciones en función del modo de vuelo.

Hacer esto permite que sea más fácil añadir modos de vuelo a posteriori como por ejemplo modos de vuelo basados en GPS como el “Loiter” y el “RTH”.
Este modulo encargado de participar en la generación de la referencia, tiene dos métodos anidados. Estos son ref_set_mode() y ref_gen().

3.2.2.1. Ref_set_mode()

Este primer metodo es el encargado de contener toda la lógica para discernir cual es el modo de vuelo en el que opera. La estructura de este metodo es bastante sencilla y con unas pocas condiciones, permite implementar las transiciones necesarias entre los modos de vuelo del cuadricóptero.
El método en si está basado en una enum de Arduino, estas variables son realmente útiles para establecer asignaciones ordenadas y no ordenadas para atribuir etiquetas donde sea necesario.

3.2.2.2. Ref_gen()

Este método, se encarga de generar la referencia en función del valor del modo de vuelo establecido en el anterior método.
El método consiste simplemente en generar las referencias y contener los elementos vinculantes a cada modo de vuelo. A continuación, en forma de tabla, se pueden ver los diferentes comportamientos en función del modo de vuelo.

Flight Mode	Description	
Disabled	En este modo de vuelo, el drone permanece quieto, sin capacidad para actuar los motores. Este modo de vuelo es en el que siempre se debe trabajar el cuadricóptero para evitar posibles lesiones.	Led de seguridad encendido
Mounting	Modo de transición entre Disabled y los modos de vuelo de operación del drone.	Led de seguridad apagado Reset de los PIDs
Stable	Modo de vuelo que permite volar el drone en modo estabilizado, donde el usuario controla la altura con el	Throttle proviene de la radio. Se registra la presión actual para realizar la

	throttle, y el roll pitch y yaw.	transición a Altitude Hold”
Altitude hold	Modo de vuelo que permite volar el drone en modo altitud constante, donde la altura se regula automáticamente y el usuario controla roll pitch y yaw.	Throttle proviene del hover throttle (hallado experimentalmente).

3.2.3. Read_units()

La función "read_units" tiene como propósito procesar las señales de entrada que son necesarias para el correcto funcionamiento del sistema de control de vuelo. Estas señales pueden provenir de distintas fuentes, como por ejemplo sensores, baterías, entre otros elementos del sistema. Es importante que estas señales sean procesadas y acondicionadas correctamente, para poder obtener información útil y precisa acerca del estado del drone y así poder generar las referencias necesarias para el control de este.

Esta función es esencial en la arquitectura de software del sistema de control de vuelo, ya que es el punto de entrada para todas las señales de entrada que son necesarias para el correcto funcionamiento del drone. Por lo tanto, es fundamental que esta función sea desarrollada cuidadosamente y de forma robusta, para asegurar que todas las señales de entrada sean adquiridas y procesadas de forma correcta.

La llamada a read_units es simplemente la llamada una serie de funciones anidadas, sirviendo como modulo. Estas funciones son las siguientes:

3.2.3.1. Read_battery()

La función read_battery() es una función esencial en el controlador de vuelo del drone, ya que se encarga de medir la tensión de la batería. Es importante tener en cuenta que la batería utilizada en el drone es de unos 11.1V nominales y los pines del microcontrolador operan entre 3.3-5 V, lo que significa que, si intentáramos leer directamente la tensión de la batería con el microcontrolador, se produciría un cortocircuito y se dañaría el microcontrolador.

Para evitar esto, se utiliza un divisor de tensión de dos resistencias en serie para reducir la tensión de la batería a un nivel seguro y legible por el microcontrolador. De esta manera, se puede medir la tensión de la batería con precisión y utilizar esta información para ajustar la potencia del drone y garantizar que no se agote la batería durante el vuelo.

[HACER REFERENCIA AL ESQUEMA ELECTRICO?]
La función read_battery() se encarga de leer los valores de voltaje a través del divisor de tensión y realizar los cálculos necesarios para convertir esos valores

en una lectura de tensión precisa de la batería. Conocer en todo momento la tensión de la batería es importante para diferentes aspectos, como asegurarse que no agotamos la batería completamente o regular la potencia de los motores.

El hecho de añadir un divisor de tensión a la batería, lo que produce es
La implementación de esta función es muy básica simplemente se encarga de hacer la lectura analógica del pin al que está conectado el divisor de tensión y hacer un escalado para recuperar el valor de la tensión real de la batería.

$$V_{PIN} = V_{BAT} \cdot \frac{R_1}{R_1 + R_2}$$

La tensión que lee el PIN es un valor de 8 bits donde 0 equivale a 0 V y 255 equivale a 3,3V. Por lo tanto, para recuperar el valor de la tensión del divisor de tensión V_{PIN} debemos mapear este valor V_{AR} de 8 bits al rango correspondiente:

$$V_{AR} = \frac{255}{3{,}3\ V} \cdot V_{PIN}$$

Por tanto, para recuperar el valor de V_{BAT} debemos:

$$V_{AR} = V_{BAT} \cdot \frac{255}{3{,}3\ V} \cdot \frac{R_1}{R_1 + R_2} \rightarrow V_{BAT} = V_{AR} \cdot \frac{3{,}3V}{255} \cdot \frac{R_1}{R_1 + R_2}$$

Sustituyendo $R_1 = X$ y $R_2 = Y$:

$$V_{BAT} = X2 \cdot V_{AR}$$

Y, finalmente aplicando el filtro complementario

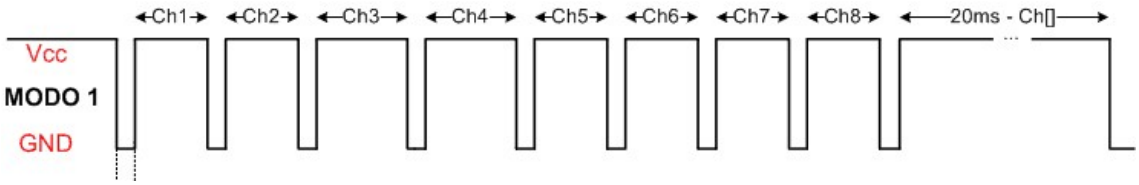
$$V_{BAT}(z) = 0{,}92 \cdot V_{BAT}(z-1) + 0{,}08 \cdot V_{BAT}(z)$$

3.2.3.2. Read_rc()

La función read_rc no es realmente la función encargada de la lectura de la lectura de la radio, pero si es la encargada del procesado. Como esta estipulado anteriormente en esta memoria, se utiliza una interrupción de hardware para ir registrando las lecturas de los pulsos PPM de la radio, sin embargo, se utiliza esta función para guardar en un vector la duración de los pulsos recibidos por la radio, los cuales se emplearán en el algoritmo de control para realizar el control de la aeronave.

Antes de hablar de la función read_rc es importante hablar de la función read_PPM. Esta función se ejecuta cada vez que la señal PPM pasa de estado HIGH a LOW y viceversa. La tarea de esta función es realizar el sincronismo

con los mensajes de la modulación PPM e ir almacenando el ancho de los pulsos en un vector.



Como podemos ver en esta imagen, la señal PPM es simplemente una señal que envía una serie de pulsos cuyo ancho esta relacionado con la posición de los joysticks de la radio, cambiando los diferentes joysticks e interruptores de la radio se reflejara en la señal PPM modificando el ancho de los pulsos. Esta señal que se puede ver en la figura se repite periódicamente de manera que, una vez ha terminado el pulso largo de 20 ms, vuelve a empezar a enviar los canales de manera ordenada. Es precisamente este pulso de 20 ms el que permite realizar al controlador de vuelo el sincronismo con la señal recibida, cuando se reciba un pulso de 20 ms.

En la práctica, lo que observamos es que los pulsos que referencian a los canales de la radio tienen una duración de entre 600 us a 1600 us. En este caso, se considero el pulso de larga duración cualquier pulso recibido cuya duración fuese superior a 2500 us.

Por lo tanto, para almacenar adecuadamente la información recibida por la radio, se ha optado por seguir la siguiente estrategia:

[INSERTAR DIAGRAMA FUNCIONAL]
Una vez explicada la función read_PPM, la cual se encarga de proporcionarnos un vector con los instantes en los cuales se ha detectado un cambio en los pulsos, procedemos a procesar estos datos con la función read_RC.

Read_RC se encarga de restar estos instantes de tiempo para obtener anchos de pulso y, posteriormente, se mapean para luego emplearlos como referencia en el algoritmo de control.

Nota: El mapeo no es realmente necesario, sin embargo aquí se ha empleado para simplificar posteriores operaciones realizadas en el algoritmo de control.

3.2.3.3. Read_gyro

Este método, se encarga de realizar las lecturas de la MPU6050. Para obtener estas medidas el procedimiento a seguir es bastante similar que el empleado en la inicialización para configurar el barómetro o la IMU, utilizando la librería Wire, se puede llevar a cabo la conexión con el dispositivo mediante el protocolo I2C a través de su dirección de 8 bits.

Los registros que estamos interesados en consultar son los que nos proporcionan las aceleraciones y velocidades angulares en los tres ejes. A continuación, se puede ver el fragmento del mapa de registros de la MPU6050 que contiene los valores que nos interesan.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Se puede apreciar que los valores de aceleración, temperatura y velocidad angular son de 16 bits y se proporcionan de manera consecutiva por lo que podemos solicitarlos como sigue:

```
Wire.beginTransmission(gyro_address);
Wire.write(0x3B);
Wire.endTransmission();
Wire.requestFrom(gyro_address, 14);
```

Y podemos combinar los valores recibidos como sigue:

```
acc_x = Wire.read() << 8 | Wire.read();
acc_y = Wire.read() << 8 | Wire.read();
acc_z = Wire.read() << 8 | Wire.read();
temperature = Wire.read() << 8 | Wire.read();
gyro_pitch = Wire.read() << 8 | Wire.read();
gyro_roll = Wire.read() << 8 | Wire.read();
gyro_yaw = Wire.read() << 8 | Wire.read();
```

De esta forma, logramos almacenar estos valores para, posteriormente, realizar las operaciones necesarias para usarlo junto a la referencia para el algoritmo de control.

[FALTA AÑADIR LA PARTE DE MEAS.PROC Y UNA PARTE DE LA FUNCION READ GYRO]

3.2.3.4. Read_barometer_v2()

Este método, se encarga de realizar las lecturas de la MPU6050. Para obtener estas medidas el procedimiento a seguir es bastante similar que el empleado en la inicialización para configurar el barómetro o la IMU, utilizando la librería

Wire, se puede llevar a cabo la conexión con el dispositivo mediante el protocolo I2C a través de su dirección de 8 bits.

Este método se encarga de realizar las lecturas del sensor BMP280 para obtener las lecturas de presión que posteriormente serán empleadas para realizar el control de altitud en el modo de vuelo “altitude hold”.

El proceso para realizar las lecturas del barómetro es similar al que hemos visto anteriormente para las tareas de inicialización o la lectura de la MPU6050, recurrimos a la librería Wire para hacer la lectura.

Sin embargo, realizando el desarrollo del software, apareció una serie de problemáticas relacionada con la lectura del barómetro que imposibilitaba la lectura adecuada de los datos.

El primer problema que apareció fue que la frecuencia de actualización de registros del barómetro era inferior a la de la lectura de los mismos registros, lo cual conllevaba que la señal presión y temperatura obtenidas por el controlador de vuelo tenía valores repetidos. Para solucionar este problema la solución a implementar fue realizar las solicitudes con una frecuencia similar a la frecuencia de actualización de registros del sensor lo cual elimina o reduce al máximo los valores repetidos de las señales. Finalmente, se decidió hacer la consulta al barómetro una vez cada cuatro ciclos, como sigue:

Wait -> Wait -> Wait -> Request and read [REPEAT]

Otro problema que apareció una vez el primero fue solucionado fue que los ciclos del bucle principal en los cuales se hacia la consulta al sensor eran demasiado lento. Es decir, el solicitar los registros y proceder a su lectura conllevaba que el bucle principal del código superase el tiempo máximo de ejecución por ciclo.

Para solucionar este problema, se decidió cambiar la manera en la que se hacia la consulta y la lectura al barómetro de la siguiente manera:

Wait -> Request -> Wait -> Read [REPEAT]

Una vez diseñado el ciclo de solicitud y lectura del barómetro, se almacenan los valores de presión y temperatura que provee el barómetro de una manera similar como se hace con la MPU605.

Como se ha mencionado anteriormente, uno de los problemas principales del sensor es que el cambio de temperatura afecta directamente a las lecturas de presión. Esta sensibilidad en cambios de temperatura se puede ver manifestada, a nivel operacional, por rachas de viento frías o calidas o simplemente que el cuadricóptero pase de la sombra a la luz lo que generan errores en la presión correspondientes a varios metros de altura. Es por eso,

que el fabricante, provee unas correcciones con la temperatura para evitar estos errores en las lecturas de presión. Estas correcciones se han adaptado para el entorno Arduino y ejecutado a través de los métodos: bmp280_compensate_P_int64 y bmp280_compensate_T_int32.

Una vez se dispone de la señal presión, lo que se puede observar es que es una señal irregular y que se debe realizar un filtrado para obtener una señal mas limpia. Para ello, se han implementado unos filtros que permiten obtener una señal mas ajustada. Se ha implementado un filtro que mitiga los pequeños cambios de presión los cuales pueden ser de ruido, algún tipo de interferencia electromagnética u otros factores mediante un filtro complementario.

Uno de los problemas principales de los filtros complementarios es que generan respuestas mas lentas a las variaciones de la señal. Contra mas estricto sea el filtro complementario mas lenta será la reacción a los cambios. Esto realmente no es un gran problema cuando las variaciones de presión son pequeñas porque no pasa nada que el drone se mueva unos pequeños CMs y tardemos unos cuantos ciclos de código en procesarlo. Sin embargo, el problema se vuelve mayor cuando existe un cambio de presión mas grande por lo que es necesario adaptar esta casuística al procesado de la señal

Para hacerlo, se calcula la diferencia de presión detectada en el ciclo actual y, si esta es muy elevada se añade un termino al filtro complementario que reacciona rápidamente a estos cambios.

Esto permite mitigar fácilmente pequeñas variaciones de presión usualmente provocadas por ruido u otras interferencias y, a la vez, recibir grandes cambios de presión que requieren una respuesta rápida del drone para corregir su posición vertical.

[LA LLAMADA DEL CONTROLADOR ESTA PUESTA EN ESTE CICLO, HABRIA QUE MOVERLA AL CONTROLADOR Y EXPLICAR UN POCO LO QUE SUCEDE]
[ADICIONALMENTE, HAY QUE EXPLICAR EL CODIGO DENTO DEL /LONG TERM VARIATION]

3.2.4. Controllers()

Dentro del módulo correspondiente a los controladores, denominada controllers(), se encuentran todos los métodos que se encargan de aplicar los PID al algoritmo de control. Este módulo, se encarga de obtener los datos recibidos de reference_computation() y read_units() para obtener las señales de referencia y las señales de los sensores y realizar los cálculos del error de control.

Pid_attitude_sp() es el método que se encarga de realizar el calculo del error de control para roll, pitch y yaw y adaptar este resultado para luego, posteriormente, enviárselo a los controladores.

Los controladores empleados para el control del drone son los PID. Un PID, que significa Proportional-Integral-Derivative (Proporcional-Integral-Derivativo), es un algoritmo de control utilizado en sistemas de control automático. Es uno de los métodos más comunes y efectivos para mantener un proceso o sistema en un estado deseado.

El control PID utiliza tres componentes principales: proporcional (P), integral (I) y derivativo (D). Estos componentes trabajan juntos para ajustar y estabilizar el sistema y se hallan dentro del método calculate_pid().

El término "proporcional" se refiere a que la acción de control es proporcional al error entre el valor deseado y el valor medido del proceso. El componente proporcional responde de manera directamente proporcional al error, lo que significa que cuanto mayor sea el error, mayor será la acción correctiva.
El término "integral" se refiere a la acumulación del error a lo largo del tiempo. Este componente toma en cuenta la suma acumulada de los errores pasados y corrige el control en función de esta integral. Ayuda a eliminar el error acumulado y a reducir el error en estado estacionario.

El término "derivativo" se refiere a la tasa de cambio del error. Este componente ayuda a predecir cómo cambiará el error en el futuro y permite realizar correcciones anticipadas. Ayuda a reducir la velocidad de respuesta y la estabilidad del sistema.

3.2.4.1. Pid_attitude_sp()

Este método se encarga de convertir las lecturas proporcionadas por la radio en referencias a seguir. Como se puede ver, este método contiene la generación de las referencias para roll pitch y yaw y su funcionamento es bastante simple e intuitivo.

Como se ha explicado anteriormente en esta memoria, las señales producidas por la radio se mapean entre 1000 us y 2000 us y están centradas en 1500 us, lo que correspondería a tener el stick en el centro. Sin embargo, las radios no son perfectas determinando cual es el centro de su stick y es por ello que las señales que recibimos pueden tener error. Para solventar este problema y, para mitigar posibles derivas del cuadricóptero, se añade una “banda muerta” que hace que el drone se mantenga pasivo en caso de recibir una señal cercana a los 1500 us. En este caso, la banda muerta es de 16 us y el rango oscila entre 1492 us y 1508 us.

Posteriormente, lo que se hace, es substraer el valor del roll actual (que provee la IMU) y se hace una división entre 3, esto consiste en hacer básicamente un mapeo, donde se busca convertir los pulsos recibidos por la radio a velocidad angular deseada. En este caso, se ha establecido que un error de 500 us corresponda a una velocidad angular de referencia de 164º/s.

[REUBICAR, AMPLIAR, CORREGIR, CONCRETAR…]

3.2.4.2. Calculate_pid()

La función calculate PID se encarga de realizar la ejecución de los PID para roll, pitch y yaw. Este método, simplemente se encarga de lanzar los PID para asegurar un adecuado funcionamiento del drone.

Lo primero que se puede ver es una condición inhabilitante del integrator, si el sensor de ultrasónicos detecta que la señal es inferior a 25, el valor de la ganancia del integrator y el valor del buffer de memoria del integrator se suprimen. Esto se hace para desactivar el integrador cuando el cuadricóptero aun no ha alzado el vuelo.

Esto es necesario porque la parte integral del drone tiene un valor considerable y, cuando se le empieza a dar potencia estando en el suelo, el integrador empieza a acumular error lo cual puede provocar que se potencien unos motores mas que otros y el drone no se levante adecuadamente.

Posteriormente a eso, se realizan otros cálculos necesarios para la ejecución del PID. Se calcula la señal de error substrayendo la referencia de la medida proporcionada por el sensor. Se acumula el error en el buffer del integrator y se limita para que no se exceda demasiado, esto también sirve para controlar que la acumulación de error no se convierta en un exceso de potencia en los motores.

Posteriormente se halla la salida del PID y esta, a su vez, también es limitada a unos valores máximos para asegurarnos que el sistema no se satura.

3.2.5. Actuators()

Este es el ultimo modulo por el que pasa el algoritmo de control antes de llegar a los actuadores del drone. Los actuadores son esos dispositivos que se encargan de transmitir los cálculos generados por el algoritmo de control a los diferentes elementos del drone. Dentro de esta función, como es de esperar, aparece todo referenciado a la actuación de los motores en general pero también se puede incluir contenido relacionado con la generación de los pulsos para el sensor de ultrasónico u otros elementos.

Este módulo, como los demás, está dividido en métodos para segmentar la ejecución del código. En este caso, un total de 3 métodos se hallan en este módulo las cuales son act_esc_outputs(), act_esc_PWM_v2(), y act_us_pulse().

3.2.5.1. Act_esc_outputs()

Este método se encarga, principalmente, de acoplar los outputs de los PID para posteriormente procesarlos para su envío a las ESCs.

Es importante saber que las señales de salida deben oscilar entre 1000 us y 2000 us, esto corresponde a el ancho de pulso de la señal PWM generada para las ESCs donde los 1000 us corresponden a tener los motores apagados y 2000 us corresponde a tener los motores operando a máxima potencia.

Las salidas de los PID que deben ser acopladas dependen del modo de vuelo en el que opere el cuadricóptero. Es por ello por lo que la función es esencialmente una condición que trabaja en función del modo de vuelo de la aeronave.

Para el modo de vuelo estabilizado el acople de las salidas de los PID corresponde como sigue:

$$ESC_1 = throttle - PID_\phi - PID_\theta - PID_\varphi$$
$$ESC_2 = throttle + PID_\phi - PID_\theta + PID_\varphi$$
$$ESC_3 = throttle + PID_\phi + PID_\theta - PID_\varphi$$
$$ESC_4 = throttle - PID_\phi + PID_\theta + PID_\varphi$$

Para el modo de vuelo altitude hold, el acople de las salidas PID es esencialmente el mismo que el del modo estabilizado salvo que el throttle corresponde al throttle de equilibrio y se añade un termino que corresponde a la corrección del throttle por altitud.

$$ESC_1 = throttle_{hover} + PID_{alt} - PID_\phi - PID_\theta - PID_\varphi$$
$$ESC_2 = throttle_{hover} + PID_{alt} + PID_\phi - PID_\theta + PID_\varphi$$
$$ESC_3 = throttle_{hover} + PID_{alt} + PID_\phi + PID_\theta - PID_\varphi$$
$$ESC_4 = throttle_{hover} + PID_{alt} - PID_\phi + PID_\theta + PID_\varphi$$

Para el resto de los modos de vuelo, los cuales no operan el drone en vuelo, se limitan las salidas a 1000 us.

Finalmente, existe una condición limitante que se encarga de mantener los valores de las ESCs en el rango de 1000 us y 2000 us.

3.2.5.2. Act_esc_PWM_v2()

Este método se encarga de obtener los valores de las ecuaciones implementadas en el método anterior y generar la señal PWM. Como consta anteriormente en esta memoria, con la finalidad de ahorrar tiempo en los ciclos de ejecución, se ha decido optar por usar los timers del microcontrolador para generar las señales PWM.

Simplemente, para modificar el ancho del pulso de las señales generadas por los timers, añadimos la siguiente línea de código:

```
TIM_M1_M2->setCaptureCompare(channel_motor1, esc_1, MICROSEC_COMPARE_FORMAT)
```

Finalmente, esta corresponde a la salida final de la señal PWM generada a partir de los cálculos de todo el algoritmo de control, desde la generación de la referencia y la comparación con los inputs de los sensores a la obtención del error de control, la ejecución de los PIDs y el posterior acople de sus salidas para obtener los anchos de las señales PWM.

3.2.5.3. Act_us_pulse()

Este método, ajeno a la actuación de los motores, consiste en generar el pulso para el sensor de ultrasónicos, este pulso es generado una vez cada 7,5 ms y se emplea para medir la distancia cuando el cuadricóptero se halla cerca del suelo.

CAPÍTULO 4. FUNCIONAMIENTO

CAPÍTULO 5. CONCLUSIONES

DEAD TEXT

READ BATTERY

La función read_battery() es una función esencial en el controlador de vuelo del drone, ya que se encarga de medir la tensión de la batería. Es importante tener en cuenta que la batería utilizada en el drone es de unos 11.1V nominales, lo que significa que si intentáramos leer directamente la tensión de la batería con el microcontrolador, se produciría un cortocircuito y se dañaría el microcontrolador.

Para evitar esto, se utiliza un divisor de tensión de dos resistencias en serie para reducir la tensión de la batería a un nivel seguro y legible por el microcontrolador. De esta manera, se puede medir la tensión de la batería con precisión y utilizar esta información para ajustar la potencia del drone y garantizar que no se agote la batería durante el vuelo.

La función read_battery() se encarga de leer los valores de voltaje a través del divisor de tensión y realizar los cálculos necesarios para convertir esos valores en una lectura de tensión precisa de la batería. Conocer en todo momento la tensión de la batería es importante para diferentes aspectos, como asegurarse que no agotamos la batería completamente o regular la potencia de los motores.

READ RC:

La función read_rc() en el código del drone se encarga de leer la señal PPM enviada por la radio y decodificarla en los diferentes canales de control que se utilizarán para el vuelo.

La modulación PPM (Pulse Position Modulation) es una técnica utilizada en la comunicación inalámbrica para transmitir datos a través de una señal de radio. En esta técnica, la información se transmite mediante la variación de la posición de los pulsos en una señal de frecuencia constante.

En el caso de la radio control, la señal PPM es utilizada para enviar las señales de control del usuario al microcontrolador del drone. La señal PPM es una señal digital que incluye todos los canales de la radio en una sola señal, permitiendo una transmisión más eficiente y reduciendo la posibilidad de interferencias.

La principal ventaja de la modulación PPM es la reducción de cables y conectores necesarios para enviar las señales de la radio al microcontrolador. Además, al enviar toda la información en una sola señal, se reduce el tiempo de transmisión y se mejora la precisión en los movimientos del drone.

READ GYRO

La función read_gyro() se encarga de leer los valores de aceleración y velocidad angular del sensor MPU6050. Para lograr esto, utiliza el protocolo I2C, que es un protocolo de comunicación serial síncrono, utilizado para interconectar circuitos integrados en un mismo circuito.

El protocolo I2C se caracteriza por ser un protocolo de dos hilos que utiliza una línea de reloj (SCL) y una línea de datos (SDA) para la transferencia de información. El dispositivo maestro, en este caso el microcontrolador, controla la comunicación mediante la generación de pulsos en la línea de reloj, mientras

que los dispositivos esclavos, como la MPU6050, responden a estas señales y envían los datos solicitados en la línea de datos.
Una de las principales ventajas del protocolo I2C es que permite la conexión de varios dispositivos en un mismo bus de comunicación, lo que lo hace muy útil en aplicaciones donde se requiere leer múltiples sensores. Además, al ser un protocolo de comunicación síncrono, la transferencia de datos es rápida y eficiente.
En cuanto a la implementación de la función read_gyro(), se utiliza la librería "Wire.h" que facilita el acceso y comunicación con dispositivos que utilizan el protocolo I2C. La función se encarga de establecer conexión con la MPU6050, solicitar los registros correspondientes y obtener los valores de aceleración, velocidad angular y temperatura del sensor. Estos valores son procesados posteriormente y luego son empleados en el cálculo del algoritmo de control.
READ_BAROMETER
La función read_barometer() es una función que se encarga de leer los valores de presión atmosférica y temperatura del sensor BMP280 utilizando el protocolo I2C. Este sensor es muy útil en los drones ya que nos permite conocer la altitud del drone y también nos proporciona información sobre la temperatura ambiente.
Uno de los problemas del BMP280 es que no puede proporcionar la información que necesitamos en cada ciclo de ejecución del microcontrolador, ya que este tarda más en preparar las lecturas para que puedan ser consultadas. Por lo tanto, probando las capacidades del chip, he decidido hacer la consulta de los datos una vez cada cuatro ciclos como sigue:
Request -> Wait -> Read -> Wait
Otra casuística a tratar del sensor BMP280 es que es muy sensible a cambios de temperatura, lo que puede afectar drásticamente a las lecturas de presión. Para solucionar esto, se pueden aplicar correcciones a las medidas de presión utilizando las funciones de corrección proporcionadas por el fabricante, las cuales han sido adaptadas para el código Arduino utilizado en el nuestro proyecto.

Reference computation y modos de vuelo:
La función ref_gen() desempeña un papel crucial en el algoritmo de control del drone, ya que se encarga de generar la referencia de vuelo para el sistema. En otras palabras, determina qué acciones debe tomar el drone en función del modo de vuelo en el que se encuentre.
Dentro de la función ref_gen(), encontramos una subfunción llamada ref_set_mode(), la cual se encarga de generar las referencias de vuelo específicas para cada modo de vuelo. Estos modos de vuelo están definidos mediante una enumeración que permite cambiar entre diferentes configuraciones de vuelo.
El primer modo de vuelo es FM_disabled, en el cual el drone se encuentra completamente deshabilitado y no responde a ningún input. Este modo es utilizado cuando se necesita interactuar físicamente con el drone de manera segura, como al añadir o quitar hélices o desconectar la batería. FM_disabled es el modo de vuelo predeterminado cuando se alimenta el drone, y se puede acceder a él moviendo el stick derecho hacia abajo y a la izquierda. Es importante tener la posibilidad de activar este modo de vuelo en caso de necesitar interrumpir el vuelo de manera segura.
El siguiente modo de vuelo es FM_Mounting, el cual actúa como una transición entre FM_disabled y FM_stable. Proporciona una capa adicional de seguridad al drone y se activa posicionando el joystick izquierdo hacia abajo y a la derecha.
Luego, pasamos al modo de vuelo FM_stable, el cual se activa cuando el joystick vuelve a la posición central después de pasar por FM_mounting. En este modo, el drone puede ser controlado de manera estable utilizando los inputs del mando de radiocontrol como referencia, mientras que la unidad de medición inercial (IMU) se encarga de realizar las correcciones necesarias para mantener la estabilidad.
El modo de vuelo FM_alt_hold se activa cuando el interruptor del mando de radiocontrol se coloca hacia abajo. Este modo permite mantener la altitud del drone utilizando tanto la información de la IMU como el sensor de presión barométrica. El control del throttle se ajusta en base a la lectura del barómetro, lo que permite al drone mantener una altitud constante.
Estos diferentes modos de vuelo proporcionan al piloto del drone una amplia gama de opciones y funcionalidades, permitiéndole adaptar el comportamiento del drone a las necesidades específicas de cada situación de vuelo.

Controllers

La función controllers() desempeña un papel fundamental en el algoritmo de control del drone, ya que se encarga de gestionar los controladores PID (Proportional-Integral-Derivative) utilizados en diferentes aspectos del vuelo. En esta función, se encuentran las implementaciones de los controladores que serán utilizados por el drone.

Una de las funciones principales dentro de controllers() es pid_attitude_sp(). Este controlador se encarga de manejar el PID relacionado con el altitude hold, es decir, se utiliza para mantener la altitud deseada del drone cuando el modo de vuelo es el altitude hold. Utiliza la información proporcionada por el sensor de presión barométrica para realizar los cálculos necesarios y generar una señal de control adecuada para mantener el drone a la altitud objetivo.

El controlador PID utiliza tres componentes principales: proporcional, integral y derivativo. El componente proporcional ajusta la señal de control en función de la diferencia entre la altitud deseada y la altitud actual del drone. El componente integral se encarga de eliminar cualquier error acumulado a lo largo del tiempo, mientras que el componente derivativo ayuda a predecir y reaccionar ante cambios rápidos en la altitud del drone.

El objetivo principal de pid_attitude_sp() es asegurar que el drone mantenga una altitud constante y estable durante el modo de vuelo de altitude hold. Esto proporciona al piloto una experiencia de vuelo más suave y controlada, ya que el drone se encarga de realizar las correcciones necesarias para mantenerse a la altitud objetivo.

La implementación de este controlador en la función controllers() demuestra la importancia del control de altitud en el vuelo del drone y cómo se utilizan técnicas de control avanzadas para lograr un rendimiento óptimo.

Actuators

La función actuators() desempeña un papel esencial en el sistema de control del drone, ya que se encarga de transferir los cálculos realizados por el algoritmo de control a los actuadores del drone, en este caso, las ESCs (Electronic Speed Controllers).

Una de las funciones clave dentro de actuators() es la función que se encarga de combinar las salidas de los controladores PID en función del modo de vuelo en el que se encuentre el drone. Dependiendo del modo de vuelo seleccionado, como el altitude hold o el modo estabilizado, se ajusta la contribución relativa de los diferentes controladores PID para generar las señales de control adecuadas. Esta combinación de salidas de los controladores asegura que el drone responda de manera apropiada y esté controlado de acuerdo con el modo de vuelo seleccionado.

Además, dentro de actuators() se implementa un mecanismo de limitación de las señales PWM generadas para evitar daños a las ESCs. Las señales PWM se ajustan para que estén dentro de los límites seguros y adecuados para las ESCs, evitando así cualquier sobrecarga o sobretensión que pueda afectar su funcionamiento.

Por último, la función act_esc_PWM_v2() es responsable de generar las salidas PWM para las ESCs utilizando los timers del microcontrolador. Los timers se utilizan para generar las señales PWM con la frecuencia y duración adecuadas para controlar la velocidad de los motores conectados a las ESCs. Estas señales PWM son esenciales para regular la velocidad de los motores y, por lo tanto, el movimiento y el vuelo del drone.

En resumen, la función actuators() se encarga de transferir los cálculos del algoritmo de control a los actuadores del drone, en este caso, las ESCs. Esto se logra combinando las salidas de los controladores PID según el modo de vuelo seleccionado, limitando las señales PWM para proteger las ESCs y generando las salidas PWM adecuadas para controlar la velocidad de los motores.