****

**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** |

ÍNDICE

[INTRODUCCIÓN 1](#_Toc137578381)

[CAPÍTULO 1. DISEÑO DE HARDWARE 2](#_Toc137578382)

[1.1. Descripción detallada de los componentes del drone. 2](#_Toc137578383)

[1.1.1. FRAME DJI-F450 3](#_Toc137578384)

[1.1.2. Adafruit Feather STM32F405 4](#_Toc137578385)

[1.1.3. MPU6050 5](#_Toc137578386)

[1.1.4. BMP280 6](#_Toc137578387)

[1.1.5. FlightSky i6 7](#_Toc137578388)

[1.1.6. LiPo 8](#_Toc137578389)

[1.1.7. ESCs, motores y propellers. 9](#_Toc137578390)

[1.2. Esquemas y diagramas que muestren la conexión de los componentes 12](#_Toc137578391)

[CAPÍTULO 2. DISEÑO DE SOFTWARE 13](#_Toc137578392)

[2.1. Arquitectura del software 14](#_Toc137578393)

[2.1.1. Tareas de inicialización 15](#_Toc137578394)

[2.1.2. Bucle principal 16](#_Toc137578395)

[2.2. Desarrollo del software 18](#_Toc137578396)

[2.2.1. Inicialización **Error! Bookmark not defined.**](#_Toc137578397)

[2.2.2. Reference\_computation() 23](#_Toc137578398)

[2.2.3. Read\_units() 24](#_Toc137578399)

[2.2.4. Controllers() 29](#_Toc137578400)

[2.2.5. Actuators() 31](#_Toc137578401)

[CAPÍTULO 3. FUNCIONAMIENTO 34](#_Toc137578402)

[CAPÍTULO 4. CONCLUSIONES 35](#_Toc137578403)

[DEAD TEXT 36](#_Toc137578404)

# **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.

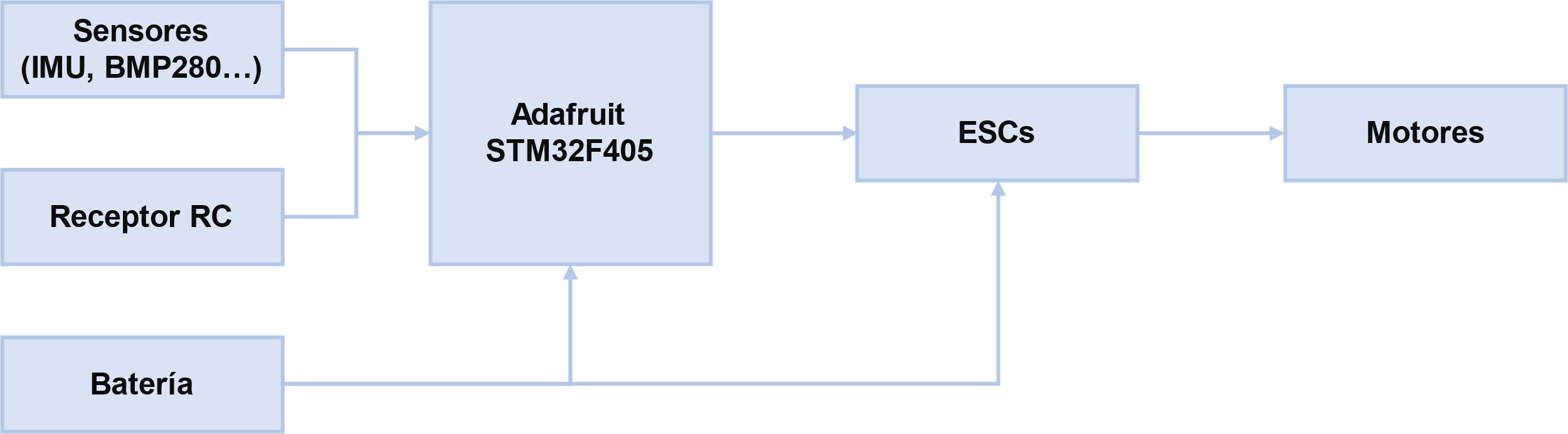
En el contexto de este proyecto, el desarrollo del software representa el componente central y de mayor importancia. El enfoque principal radica en la creación de un sistema de control inteligente y eficiente, que permita al microcontrolador STM32F405 interactuar de manera óptima con los sensores y motores 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 1. DISEÑO DE HARDWARE

## Descripción de los componentes del drone.

Los componentes de hardware seleccionados para un drone tienen que satisfacer las necesidades de un algoritmo de control para realizar el control adecuado del sistema. Hay muchos elementos que pueden formar parte de los drones, a continuación, se pueden encontrar los que han sido necesarios implementar en el proyecto para llevar a cabo el vuelo controlado del drone.

* El microcontrolador: Es posiblemente la pieza más importante de un dron debido a su función central de controlar y coordinar todas las operaciones del vehículo aéreo no tripulado. Es responsable del control de vuelo, procesamiento de datos, comunicación y seguridad del dron. Controla los motores y actuadores, procesa la información de los sensores, toma decisiones en tiempo real, facilita la comunicación entre componentes y con dispositivos externos, y garantiza la seguridad y protección del dron.
* Los sensores: Los elementos sensoriales de un dron desempeñan un papel fundamental en su funcionamiento. Dentro de estos elementos podemos encontrar la IMU o el barómetro, entre otros. La IMU garantiza la estabilidad y orientación proveyendo al microcontrolador de lecturas de aceleración y velocidad angular. El barómetro se emplea para proveer lecturas de altura basadas en la presión lo que permite realizar controles de altitud.
* Receptor RC: El receptor de control remoto es el componente que actúa como el enlace de comunicación entre el control remoto del operador y el dron. El receptor RC recibe las señales de control enviadas por el control remoto y las transmite al sistema de control del dron.
* La batería: El sistema de alimentación es el componente encargado del suministro de energía necesario para su funcionamiento. Proporciona la electricidad requerida para alimentar los motores, los sistemas electrónicos y los componentes del dron, asegurando una fuente confiable y constante de energía durante el vuelo.
* Las ESCs: Los controladores electrónicos de velocidad (ESC, por sus siglas en inglés) se encargan de regular y controlar la velocidad de los motores. Los ESC reciben las señales de control provenientes del sistema de control del dron y las utilizan para ajustar la velocidad de los motores de forma individual. Esto permite al dron realizar maniobras precisas, cambios de dirección y ajustes de velocidad según las instrucciones del piloto.
* Los motores: Los motores son los componentes encargados de la actuación de la fuerza propulsora necesaria para el vuelo. Los motores convierten la energía eléctrica suministrada por la batería en energía mecánica, que se utiliza para hacer girar las hélices del dron. Los motores permiten al dron desplazarse verticalmente y cambiar de dirección mediante el control de su velocidad de rotación.



La elección de los componentes de hardware ha sido directamente proporcionada por la universidad por lo que no se han realizado tareas de optimización en lo que se refiere a la selección de componentes electrónicos y estructurales del drone.

A continuación, se explican de forma detallada los componentes que constituyen el drone.

### 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.



### 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.

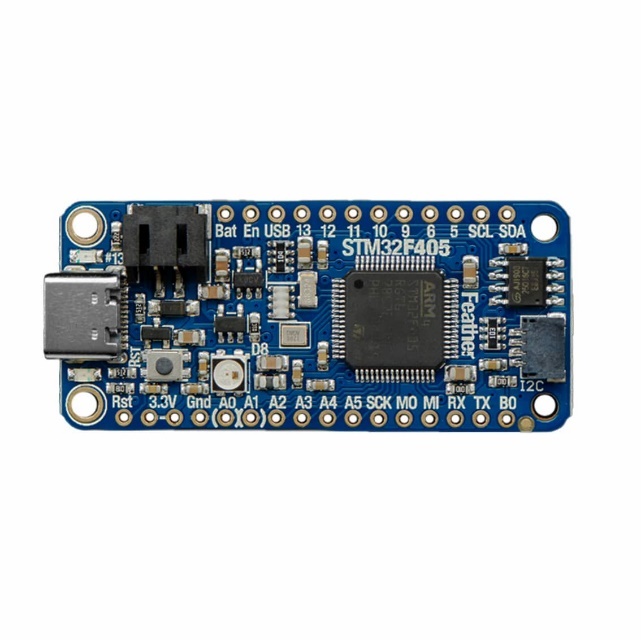
La placa de desarrollo de Adafruit se caracteriza por su facilidad de uso y su amplia gama de características, lo que la hace adecuada para una variedad de proyectos. En su núcleo, el STM32F405RG es un microcontrolador ARM Cortex-M4 de 32 bits, que opera a una frecuencia de 168 MHz y tiene una memoria 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.



### 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.



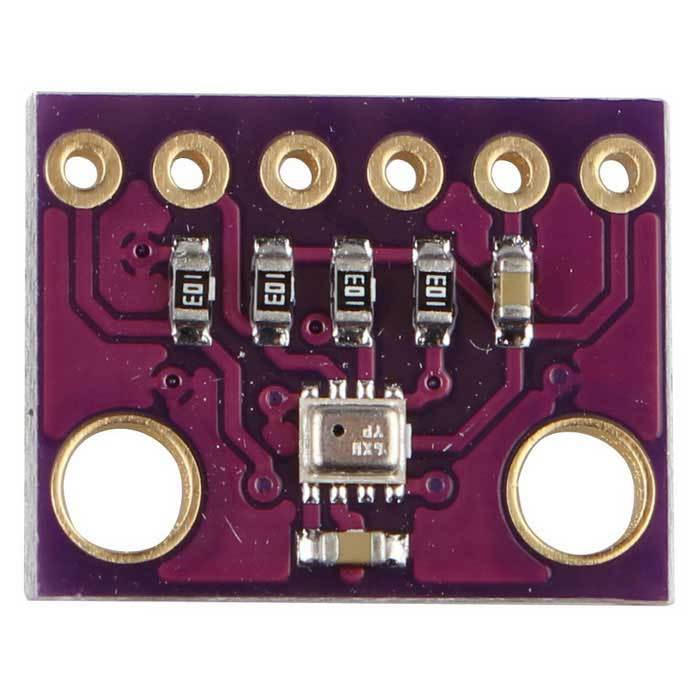
### BMP280

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.



### 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.



### 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.



### 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.

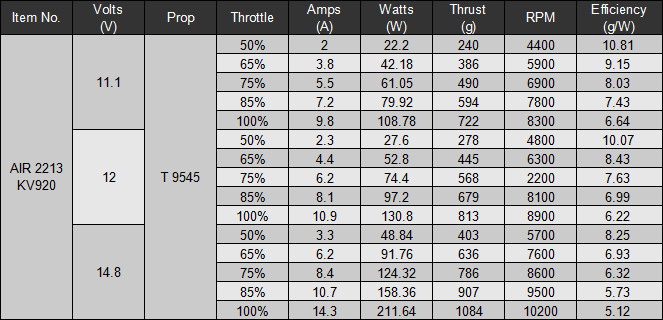
Los propellers escogidos son los T9545, son motores de 9,5 pulgadas de diámetro y con un paso de 4,5 pulgadas. Estos propellers, vienen con una cabecera de metal de auto-bloqueo la cual está hecha para asegurar que, durante el vuelo, no se aflojan los propellers. Esta cabecera de auto-bloqueo es simplemente 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.



El peso del cuadricóptero es de aproximadamente 1050 gramos, por lo que es crucial seleccionar motores y hélices que sean capaces de generar la fuerza de empuje necesaria para sostener y volar el dron. A continuación, se presenta la tabla de especificaciones seleccionada para esta combinación de motores y hélices:



Como se puede observar en la tabla de especificaciones, el empuje generado para un 65% de throttle a 11.1 V supera el peso total del dron, lo que indica que los motores y las hélices seleccionadas son más que adecuados para realizar el vuelo sin problemas. Esto garantiza una capacidad de empuje suficiente para mantener el dron en el aire y maniobrar de manera segura.

Con un thrust-to-weight ratio de:

Además, podemos calcular el tiempo de vuelo estimado del cuadricóptero para este nivel de throttle y la capacidad de la batería utilizada:

## Esquemas y diagramas que muestren la conexión de los componentes

# CAPÍTULO 2. 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.

## Arquitectura y desarrollo del software

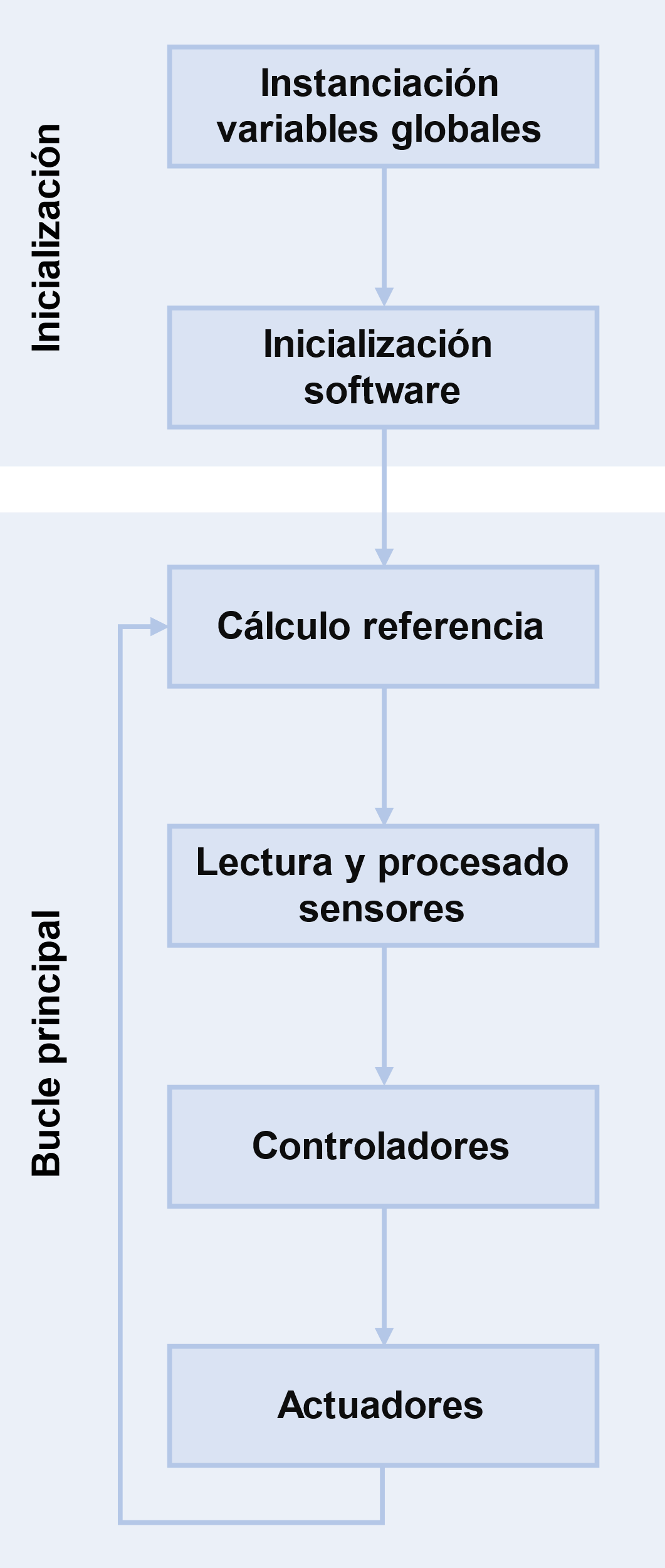
Es fundamental establecer una estructura bien definida para implementar el algoritmo de control del dron, asegurando que los cálculos internos se realicen de manera procedimental y ordenada. Esto resulta de gran utilidad a la hora de identificar posibles errores, fallos de implementación u otras situaciones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



A continuación, se detalla de manera procedimental y general, la estructura del código:

### Tareas de inicialización

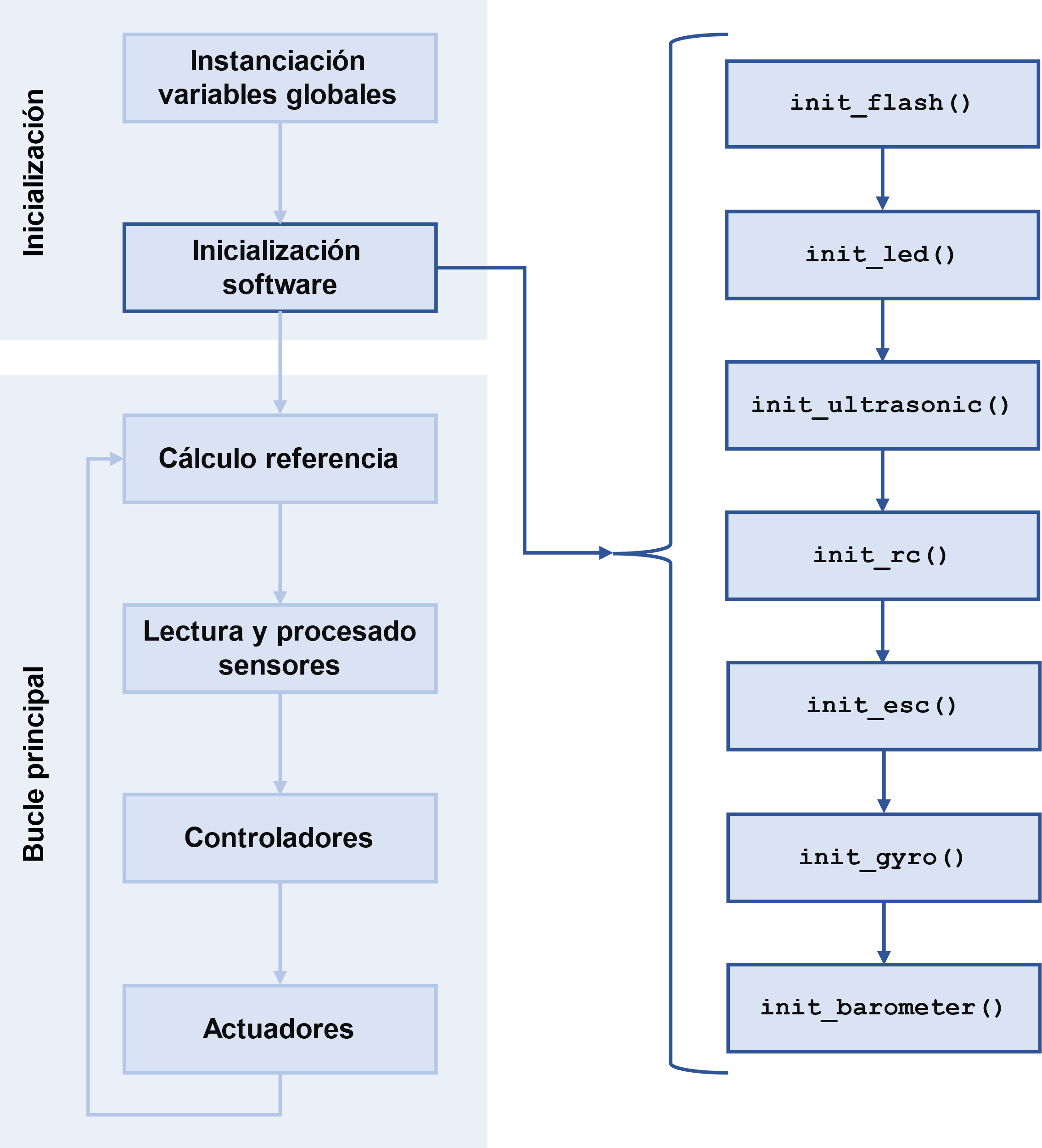
#### Declaración de variables globales utilizadas por el código.

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.

#### 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 inicio. 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.

Dentro de la función setup, hallamos la llamada a la subrutina init\_components. Esta función se encarga de ejecutar los diferentes módulos necesarios para la inicialización de los componentes del drone. 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()**: Prepara el pin conectado al LED para poder controlarlo con el microcontrolador.

**init\_ultrasonic()**: Inicializa el pin correspondiente a los ecos del ultrasónico y vincula el método para asignar el comportamiento cuando se detecte una interrupción.

**init\_rc()**: Este método se encarga de inicializar la conexión con el dispositivo receptor de la radio. 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 calibrado 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 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 configurar ciertos parámetros.

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.

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():** Este método, se encarga de realizar el calibrado necesario para preparar el barómetro BMP280 para su uso. Para hacerlo, utiliza la librería Wire para realizar la conexión I2C con el chip, junto 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 por lo 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 se emplean para correcciones de temperatura y 9 para correcciones 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 aplicar 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**;**

Finalmente, ajustamos los registros de configuración del sensor para prepararlo para el tipo de operación a realizar. Esta configuración se lleva a cabo de una manera muy similar a la que se hace con la IMU. Los registros modificados son el 244 y el 245, los cuales permiten ajustar los siguientes parámetros:

* Temperature oversampling: x2.
* Pressure oversampling: x2.
* Operational mode: Normal mode.
* Normal mode standby time duration: 0,5 ms.
* IIR filter time constant: 16.

### Bucle principal

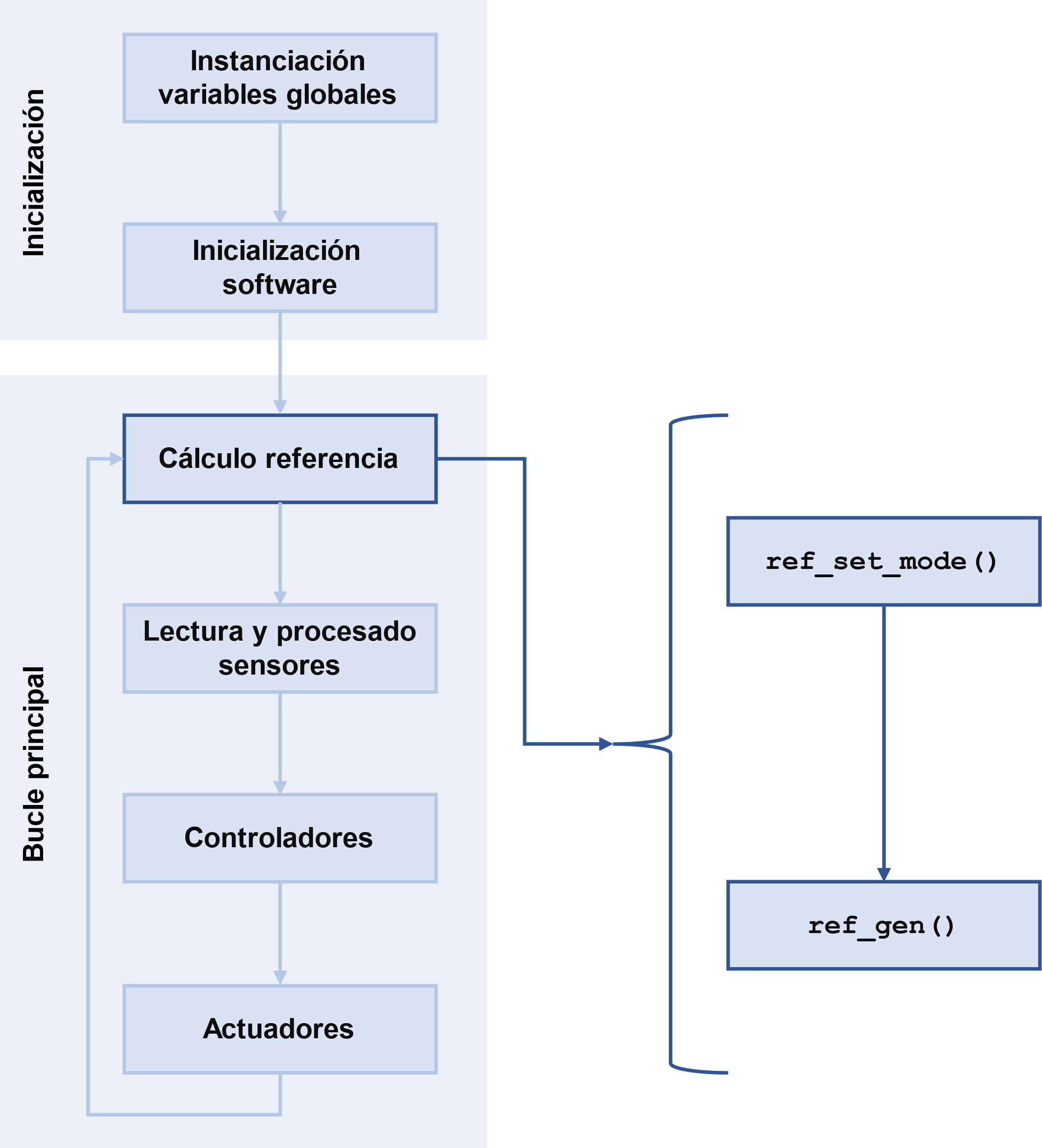
En el bucle principal encontraremos la ejecución reiterativa de las funciones principales que segmentan el código en sus respectivos módulos. En estas funciones se encontrarán las tareas de cálculo de referencias, obtención de medidas de los sensores, cálculo de la señal de error y posterior cálculo 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.

#### Cálculo de referencia.

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.

El método que se encarga de realizar estas funciones es el denominado reference\_computation. 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 prácticamente distinciones relacionadas con el 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 “Loiter” o “RTH”.

Este módulo encargado de participar en la generación de la referencia tiene las siguientes subrutinas anidadas:



**ref\_set\_mode()**: Este primer método es el encargado de contener toda la lógica para discernir cual es el modo de vuelo en el que opera. La estructura de este método 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 trabaja con una enumeración de Arduino, la cual es muy beneficiosa para establecer asignaciones ordenadas y para atribuir etiquetas donde sea necesario.

**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 en generar las referencias y contener la parametrización de vinculada a cada uno de los modos de vuelo. A continuación, en forma de tabla, se pueden ver los diferentes comportamientos en función del modo de vuelo.

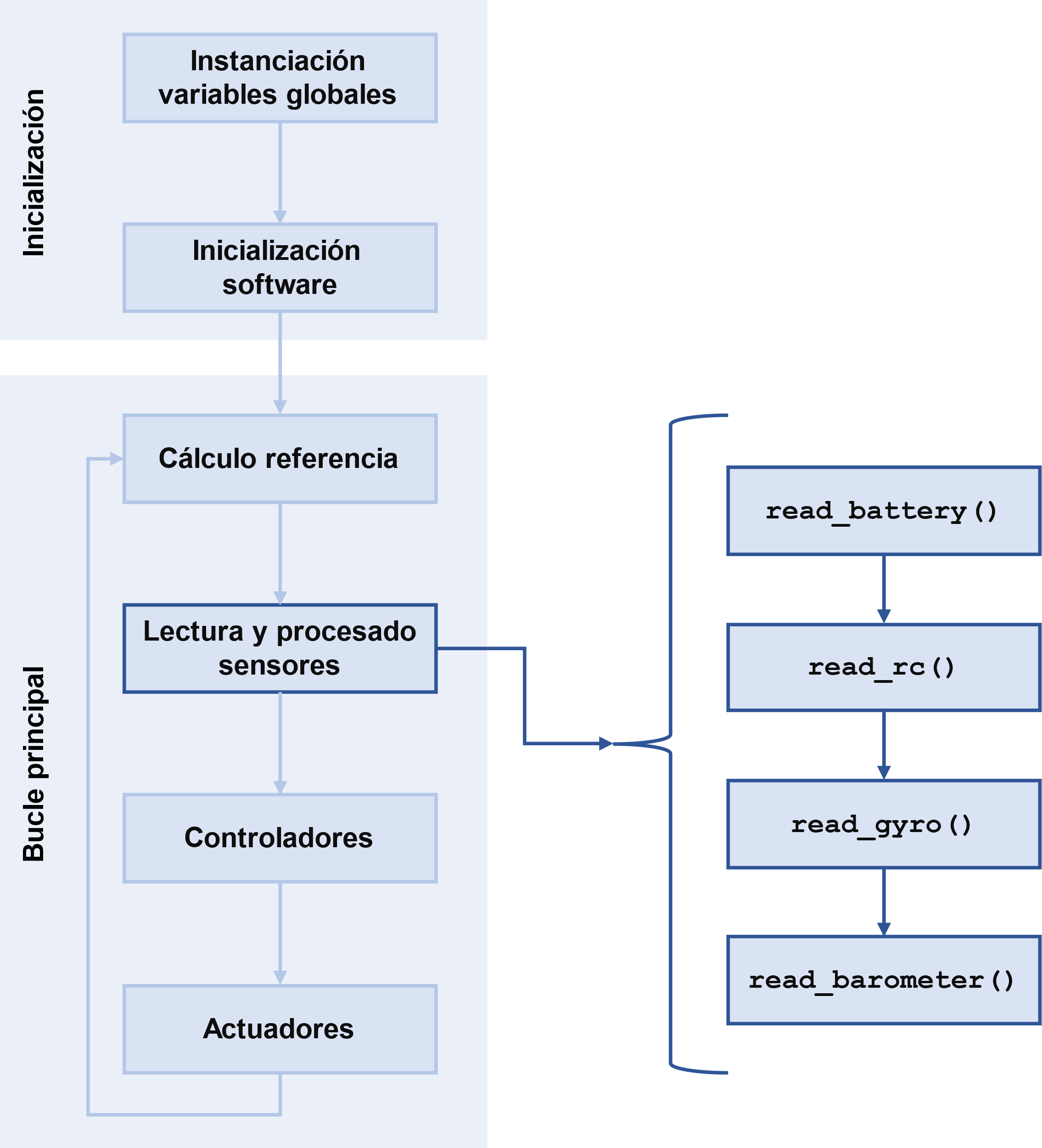
|  |  |
| --- | --- |
| **Flight Mode** | **Descripción** |
| Disabled | El drone permanece quieto, sin capacidad para actuar los motores. Este modo de vuelo es en el que siempre se debe trabajar físicamente con el cuadricóptero alimentado ya que permite una manipulación segura para evitar posibles lesiones. Por defecto este es el modo de vuelo por defecto al enchufar la batería y realizar los calibrados. |
| Mounting | Modo de transición entre Disabled y los modos de vuelo de operación del drone. |
| Stable | Modo de vuelo que permite volar el drone en modo estabilizado, donde el usuario controla la altura con el throttle, y el roll pitch y yaw. |
| 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. |

#### Lectura de los sensores y proceso de la señal.

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.

La función read\_process\_units es la que tiene el propósito de realizar esta tarea. Esta función es esencial en la arquitectura de software del sistema de control de vuelo, ya que es el origen de todas las señales en las cuales se basa el algoritmo de control. 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.

Este método, de la misma manera que los anteriores, está compuesto por la siguiente ejecución de subrutinas:



**read\_battery()**: 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 posiblemente resultaría en daños permanentes del 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 adecuadamente la potencia de los motores.

Para procesar adecuadamente la señal de tensión de la batería, debemos entender cómo afecta el divisor de tensión a las lecturas desde el microcontrolador. Implementando el divisor de tensión, sabemos que el voltaje que habrá en el borne del pin sigue la siguiente expresión:

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, el valor de tensión convertido por el ADC del microcontrolador corresponderá a la siguiente expresión:

Una vez disponemos de este valor de tensión digitalizado, podemos recuperar la tensión de la batería combinando ambas expresiones:

Sustituyendo y :

Adicionalmente, se aplica un filtro complementario para evitar posible ruido de alta frecuencia:

**read\_ppm()**: Este método se encarga de demodular la señal recibida por la radio. Esta emplea la modulación por posición de pulso (PPM) donde la señal consiste en un tren de pulsos ubicados en instantes de tiempo variables los cuales permiten, a partir de medidas temporales, obtener el ancho de los canales de la radio. Este método se implementa mediante una interrupción de hardware desde la inicialización por lo que no contiene una llamada explicita desde read\_process\_units.

Para demodular correctamente esta señal, es necesario establecer un sincronismo entre microcontrolador y receptor RX. Para hacerlo, se emplea el denominado pulso de sincronismo, el cual se identifica por tener una duración destacablemente mayor al resto. Es por eso por lo que cada vez que el microcontrolador detecta la presencia de un pulso con una duración mayor, se reinicia la lectura de los pulsos:

**if** **(**micros**()** **-** pulso\_instante**[**contador\_flaco **-** 1**]** **>** 2500**)** contador\_flaco **=** 0**;**

Estos instantes de tiempo se almacenan en un vector y se van actualizando a medida que el receptor RX envía nuevas señales al microcontrolador.

**read\_rc()**: Este método se encarga de procesar los datos obtenidos por la función anterior. Se encarga de restar los valores almacenados en el vector para obtener las posiciones de los pulsos PPM las cuales oscilan entre 600 µs y 1600 µs. Posteriormente se realiza un mapeo de estos valores a una escala que oscile entre 1000 µs y 2000 µs, este mapeo no es obligatorio ni imprescindible pero facilita la comprensión y es más fácil operar con estos valores.

**read\_gyro()**: Este método, se encarga de realizar las lecturas de la MPU6050. El procedimiento a seguir es bastante similar que el empleado en la inicialización del sensor, mediante la librería Wire y el protocolo I2C, realizaremos conexiones de lectura y escritura a los diferentes registros de interés para obtener las medidas del sensor.

Los registros de interés 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 todos directamente para ahorrar tiempo en la consulta y almacenarlos en las variables correspondientes.

**process\_gyro()**: Este método se encarga de realizar el procesado de los datos obtenidos en la subrutina anterior, esto se encarga de transformar los bits en lecturas útiles de aceleración y velocidad angular.

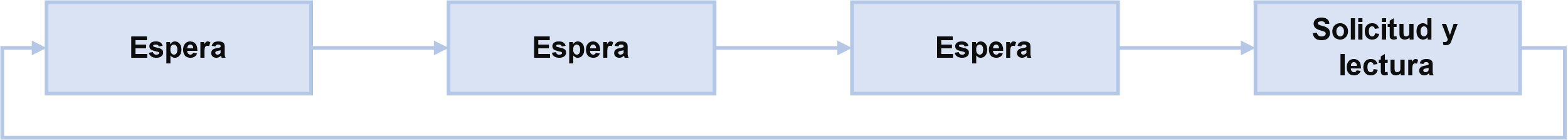
Lo primero que se computa es la conversión de los valores de velocidad angular de 16 bits a º/s y a la aplicación de un filtro complementario que permite reducir posibles señales de ruido. Estas señales corresponden a las medidas que debemos controlar mediante el uso del algoritmo de control.

Posteriormente, se calculan las contribuciones de la velocidad angular al cambio en la orientación del drone, asumiendo comportamientos lineales para cada iteración del bucle.

**read\_barometer()**: Este método, se encarga de realizar las lecturas del BMP280. La metodología seguida para obtener las medidas de este sensor a partir de los registros es la misma que la IMU, mediante el protocolo I2C y la librería Wire. Esta función provee de lecturas de presión al algoritmo de control las cuales serán usadas posteriormente para realizar el control de altitud mediante el modo de vuelo “altitude hold”.

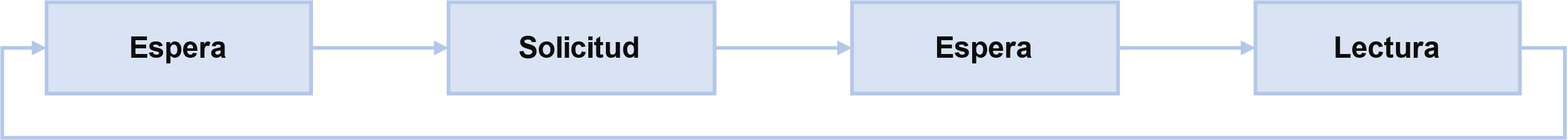
Durante el desarrollo y la implementación de la función aparecieron una serie de problemáticas que imposibilitaban la lectura adecuada de los datos y la correcta ejecución del bucle principal. El primer problema que apareció fue que la frecuencia de actualización de registros del barómetro era inferior 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 lo que resultaba en una pérdida de rendimiento en la ejecución.

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 eliminaba los valores repetidos de las señales por lo que la frecuencia de solicitud y lectura se redujo a una vez cada cuatro ciclos.



Otro problema que apareció fue que los ciclos del bucle principal en los cuales se hacia la consulta y lectura de los registros del sensor eran demasiado lentos. 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. Es de vital importancia que la frecuencia de ejecución de los controladores sea prácticamente constante ya que puede dar problemas de estabilidad derivados de los controladores si esta se ve alterada.

Para solucionar este problema, se decidió cambiar la manera en la que se hacia la consulta y la lectura al barómetro. Se optó por, en vez de realizar la lectura y consulta del barómetro una vez cada cuatro ciclos, separar consulta y lectura en ciclos diferentes.



Como se ha mencionado anteriormente en la inicialización, el BMP280 es un sensor muy sensible a cambios de temperatura, tanto es que mover el sensor de presión de la sombra al sol, ya corresponde a un error en la lectura de presión equivalente a varios metros de altura. Es por ello por lo que es necesario realizar las correcciones necesarias para evitar este tipo de alteraciones en la señal a controlar. Dichas correcciones emplean los registros cargados durante la inicialización y se aplican directamente con unas funciones de código que provee el fabricante, las cuales han sido transcritas a Arduino C para su uso para este proyecto.

La señal resultante de presión la cual provee el sensor es una señal prácticamente inmune a cambios de temperatura razonables para el tipo de operación en cuestión, pero es una señal irregular y difícilmente útil para emplearla en un algoritmo de control. Para solucionar este inconveniente se han aplicado una serie de filtros los cuales ajustan la señal para un control más adecuado.

El primer filtro que se le aplica es un filtro complementario, los filtros complementarios son útiles para eliminar ruido sin embargo se ven penalizados cuando se requiere de una respuesta rápida del drone, aplicar filtros con coeficientes mayores incrementa la lentitud en la respuesta del drone.

Para solucionar este problema de cambios grandes de presión en un intervalo de tiempo pequeño, se ha optado por añadir un sistema que detecta dichos cambios para darle prioridad a la señal y pasar el filtro complementario a un segundo plano.

Con esto se obtiene un sistema que mitiga los ruidos de manera eficaz pero que también es capaz de reaccionar de manera rápida cuando existen cambios de presión repentinos.

#### Cálculo del error de control y PIDs.

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.

Estas tareas se realizan dentro de la subrutina principal denominada controllers. Aquí se encuentran todos los métodos que se encargan de aplicar los PID al algoritmo de control. Este módulo, se encarga de recuperar 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.

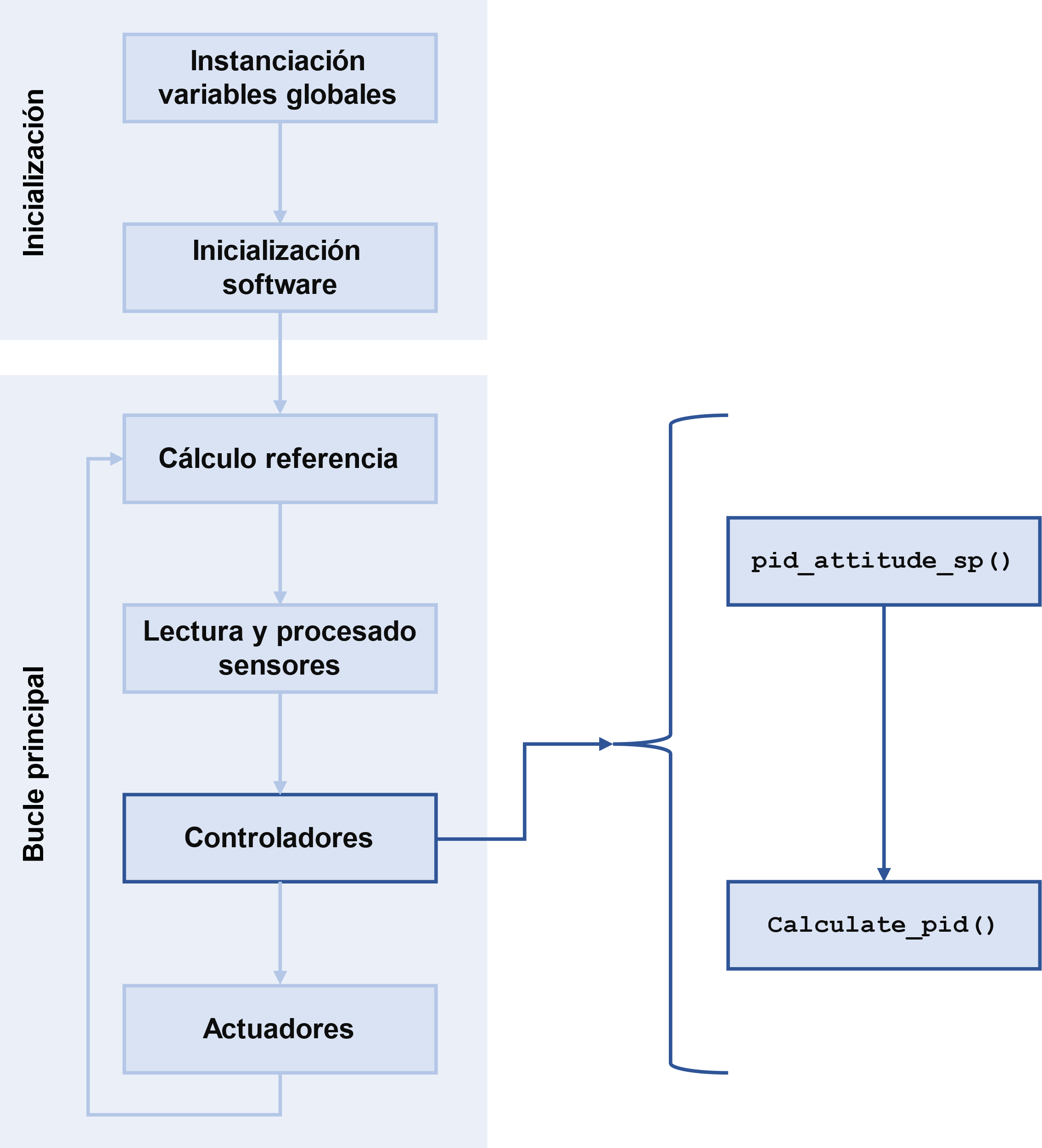
Todos los controladores que se han empleado en este proyecto son de tipo PID (Proportional-Integral-Derivative). Un PID es una función de transferencia empleada para controlar un sistema de lazo cerrado y es el controlador más popular para este tipo de aplicaciones.

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 en la estabilidad del sistema.

Este método se ha separado en la ejecución de las siguientes subrutinas:



**cnt\_attitude\_sp()**: Este método se encarga de convertir las lecturas proporcionadas por la radio en referencias a seguir. Lo primero que hace es establecer una banda muerta para las lecturas de los sticks de la radio, esto es útil porque así evitamos derivas producidas por fallos en la calibración de la radio o por ruido en la señal recibida de la radio. Posteriormente, se centran en cero los valores de manera que oscilan entre ±500 µs. Finalmente se obtiene el cálculo del error de control restando la señal referencia de la radio de la medida que entrega la IMU. El mismo proceso se repite con roll, pitch y yaw.

**cnt\_attitude\_pid()**: Esta función se encarga de realizar la ejecución de los PID para roll, pitch y yaw. Con la ejecución de esta subrutina se obtiene la salida del PID generada por la parte proporcional, integral y derivativa.

Al inicio del trabajo, se contemplaba en incorporar un sensor de ultrasonidos para implementar un sistema de detección de obstáculos, sin embargo, como los resultados no fueron realmente positivos se decidió a discontinuar esta implementación y, finalmente, se dejó el sensor para utilizarlo como takeoff detector.

Este takeoff detector ha sido empleado para desactivar la parte integral del controlador cuando el drone no ha despegado, esto evita que el PID funcione en lazo abierto y el integrador no se vuelva inestable. Esto sucede porque los integradores operados en lazo abierto tienden a acumular error cada ciclo y, al no tener la potencia necesaria para volar y estabilizarse, el error tiende a acumularse infinitamente. Si bien existen alternativas para que esto no suceda como limitar la salida del integrador, da mejor sensación al usuario emplear este takeoff detector, ya que la contribución del integrador es completamente nula durante el despegue.

Finalmente, el algoritmo del PID se ejecuta convencionalmente, calculando la señal de error, almacenando en memoria el resultado del integrador, generando la salida, aplicando limitaciones de seguridad y finalmente almacenando el valor del error del ciclo para compararlo con el del siguiente ciclo para aplicar la parte derivativa. Este proceso es repetido para roll, pitch y yaw.

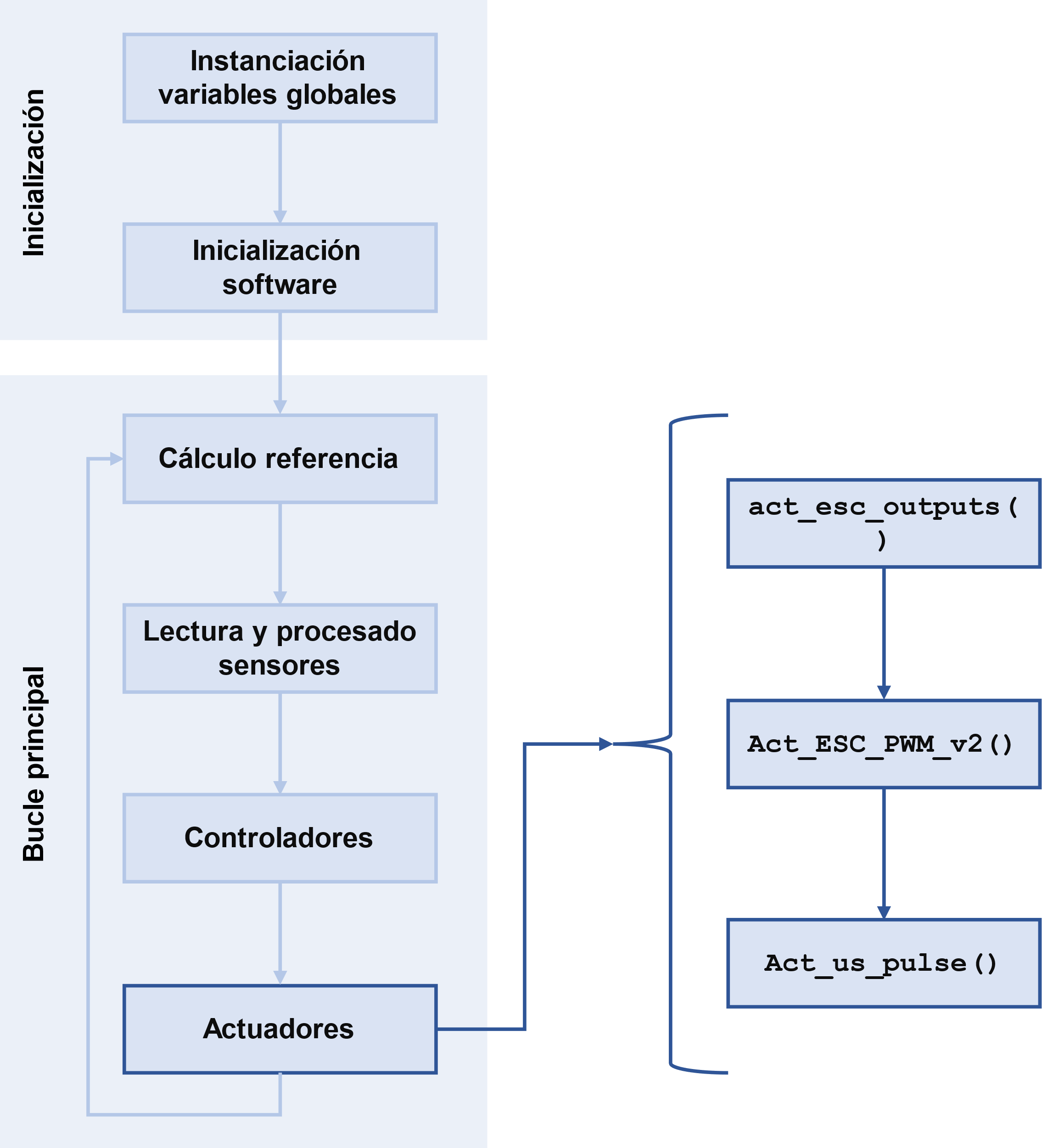
**cnt\_altitude\_pid()**: Esta subrutina se encarga de la ejecución del PID que se encarga de aplicar el control por altitud del drone. La frecuencia de este controlador es cuatro veces más baja ya que el periodo de ejecución va en fase con la disponibilidad de las lecturas del barómetro por lo que la frecuencia de ejecución es de 50 Hz en vez de 200 Hz.

La estructura de este controlador es prácticamente igual a la de roll, pitch y yaw, con algunas pequeñas modificaciones. La modificación más relevante es que se le añade a la salida del PID un término que modifica la parte proporcional del controlador de manera lineal cuando el error en la lectura es grande. Esto se hace para disponer de una parte proporcional suave cuando el drone está bastante cerca del valor deseado y aumentarla de manera lineal cuando el error sea mayor y se requiere de una respuesta más contundente.

#### Generación de la señal para los actuadores.

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.

El método principal que se encarga de realizar todas estas tareas es el denominado actuators y se encarga de llamar a todas las subrutinas relacionadas con la generación de la señal derivada para los actuadores.



**act\_esc\_outputs()**: Este método se encarga, principalmente, de acoplar las salidas de los PID para posteriormente procesarlos para su envío a las ESCs.

Es importante saber que las señales de salida deben oscilar entre 1000 µs y 2000 µs, esto corresponde a el ancho de pulso de la señal PWM generada para las ESCs donde enviar un ancho de pulso de 1000 µs significa a tener los motores apagados y 2000 µs significa 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 es el siguiente:

Se ha seguido una estrategia para implementar el modo de altitude hold que consiste en encontrar un ajuste para el throttle a través de un controlador PID, con el objetivo de regular la altura del cuadricóptero. La idea principal es la siguiente:

De esta manera, se puede controlar la altura del cuadricóptero empleando el PID de altitud y añadiendo la salida de este al throttle de equilibrio.

Sin embargo, uno de los problemas de esta solución es que la velocidad de los motores depende de la tensión que la batería puede proporcionar y este valor no se mantiene constante por lo que, el hover throttle que te permite lograr un equilibrio con la batería a 12,6 V no será el mismo que cuando la batería esté a 11,1 V.

Para abordar esta situación, se decidió llevar a cabo un vuelo completo en modo estabilizado, agotando por completo la batería. Durante este vuelo, se fue registrando el valor de la tensión de la batería y el valor del throttle cada vez que se detectaba un cambio de signo de la aceleración en el eje Z. Esto correspondía al thottle que generaba una fuerza nula en el drone.

Después del vuelo en modo estabilizado, se dispuso de una gran cantidad de lecturas las cuales fueron representadas gráficamente y se observó un comportamiento claramente lineal, es decir, los motores perdían potencia de manera lineal cuando la batería se iba agotando lo que conllevaba un aumento lineal del throttle.

Observando este comportamiento, se optó por obtener la recta de regresión de los datos obtenidos para estimar la dependencia del throttle de equilibrio con la tensión y usarlo como referencia para el altitude hold.

Cocluyendo con el tema de la búsqueda del throttle de equilibrio, ya podemos obtener las ecuaciones de las ESCs con el acople de las salidas de los PID para el modo de altitude hold:

**act\_esc\_PWM()**: Este método se encarga de recuperar los valores de las ecuaciones implementadas en el método anterior y generar la señal PWM. Para hacerlo, se modifican los anchos de pulso de las señales generadas con por los timers de la siguiente manera:

TIM\_M1\_M2**->**setCaptureCompare**(**channel\_motor1**,** esc\_1**,** MICROSEC\_COMPARE\_FORMAT**)**

**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.

### Otros elementos presentes

#### Herramientas de depuración y diagnóstico.

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.

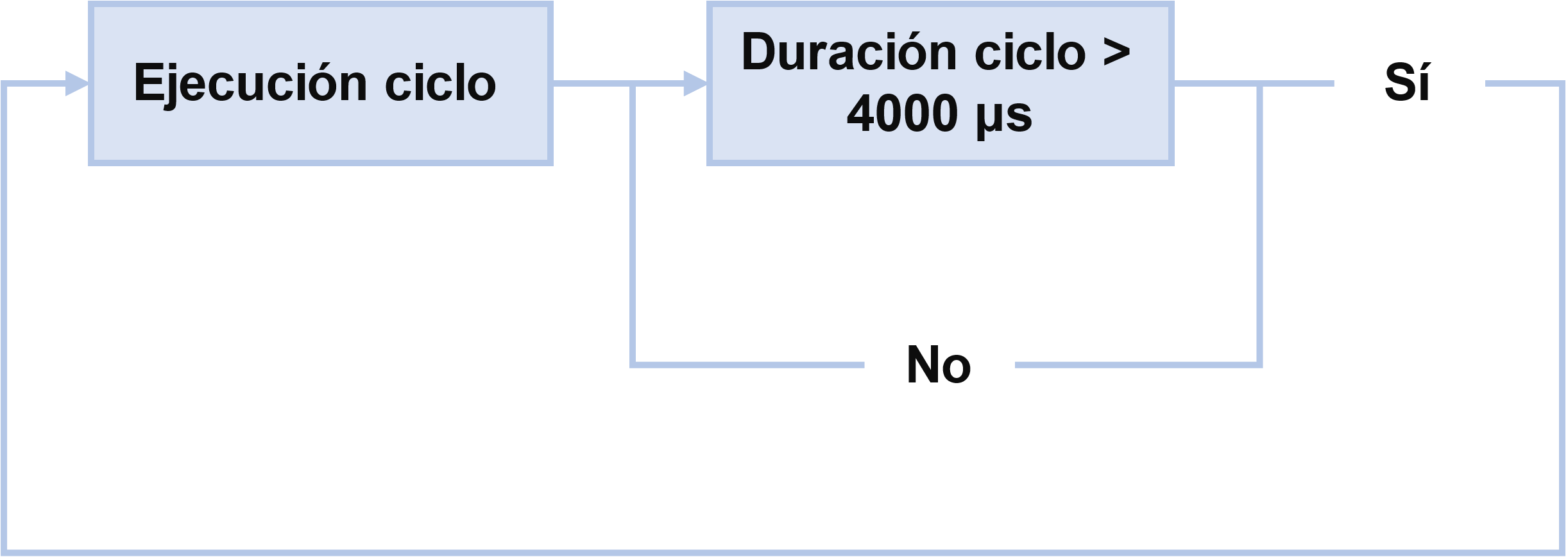
#### Sistema de control de duración del bucle.

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.



# CAPÍTULO 3. FUNCIONAMIENTO

# CAPÍTULO 4. 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.