GRADO EN INGENIERÍA AEROESPACIAL EN VEHÍCULOS AEROESPACIALES

Escuela de Ingeniería de Fuenlabrada

Curso académico 2023-2024

Trabajo Fin de Grado

Diseño y validación de la aviónica de un CubeSat con una carga de pago de emisión de microondas

Autor: Amin Y. Elamin Moreno

Tutor: Pablo Solano López

Agradecimientos

En primer lugar, me gustaría agradecer el esfuerzo realizado por mi tutor, Pablo Solano, quien ha luchado por mejorar la docencia del grado desde el primer día que llegué a la universidad, además de permitirme embaucarme en el reto que suponía este trabajo a su lado. También me gustaría agradecerle su revisión y consejos a Gemma Ramos, cuya contribución ha sido vital para asegurar la calidad del trabajo.

Por otro lado, me gustaría reservar unas palabras a mis amigos y compañeros de la carrera. Ellos han compartido mi camino estos últimos años, haciendo de esta una época que jamás olvidaré. En especial, al grupo de C5 y a Alberto Colás, quienes me han acompañado a lo largo de todas las noches largas de estudio, así como en todos los momentos que merece la pena recordar. Sin vosotros, todo habría sido mucho más complicado.

También me gustaría agradecer todo el apoyo a mis amigos de toda la vida, esas personas con las que he crecido y me he convertido en la persona que soy hoy. Los años han traído muchos cambios en nuestras vidas, pero lo que se ha mantenido ha sido su amor y compañía.

Quiero agradecerle el apoyo a mi familia. A mi hermana Sara, mi padre y mi madre, quienes han confiado en mí en todo momento y me han apoyado cuando más lo he necesitado. Todos mis logros son suyos.

Por último, me gustaría acordarme de mi primo Diego. Recuerdo cuando entré en la carrera el celebrar que fuese a haber otro ingeniero en la familia, y hacer la promesa de celebrarlo juntos el día que terminase. El destino ha querido que esa celebración la tenga que realizar solo, pero miro al cielo sabiendo que, donde estés, estás compartiendo este momento conmigo. Muchas gracias.

Resumen

El diseño y validación de la aviónica para un CubeSat con una carga de pago de emisión de microondas ha sido el núcleo de este trabajo de fin de grado. La iniciativa comenzó con la identificación y comprensión de los requisitos y el entorno específico de la misión. Se consideraron las condiciones únicas del vuelo, incluidos los desafíos generados por el vehículo lanzador o la propia misión, para establecer criterios claros para el subsistema de aviónica.

Con los requisitos en mano, se ha procedido a la fase de diseño del hardware. Se comienza con el desarrollo de la estructura general de la aviónica, para dar paso después a una arquitectura específica para el subsistema. Durante esta fase, se han tomado decisiones cruciales sobre componentes clave, como el sistema de transmisión de microondas o el sistema de despliegue estructural del satélite.

Una vez definida la arquitectura del hardware, el enfoque se trasladó al diseño e implementación del software de aviónica. Se establecieron modos específicos para la misión, y se llevó a cabo un análisis exhaustivo de riesgos y fallos para garantizar la robustez del sistema. El desarrollo del código se realizó con un enfoque en la verificación y validación, asegurando que el software cumpliera con los requisitos establecidos y funcionara de manera óptima en el entorno de la misión.

A lo largo de este proyecto, se ha demostrado que, con un diseño cuidadoso y una implementación metódica, es posible desarrollar un sistema de aviónica eficiente y confiable para un CubeSat con una carga de pago específica. Las soluciones y decisiones tomadas durante este proceso no solo son relevantes para este proyecto en particular, sino que también tienen el potencial de informar futuros desarrollos en el ámbito de los CubeSats y la exploración espacial.

Abstract

The design and validation of avionics for a CubeSat with a microwave emission payload have been at the core of this project. The initiative began with the identification and understanding of the mission's specific requirements and environment. Unique flight conditions were considered, including challenges posed by the launch vehicle or the mission itself, to establish clear criteria for the avionics subsystem.

With the requirements in hand, the hardware design phase was undertaken. It started with the development of the general avionics structure, leading to a specific architecture for the subsystem. During this phase, crucial decisions were made regarding key components, such as the microwave transmission system or the satellite's structural deployment system.

Once the hardware architecture was defined, the focus shifted to the design and implementation of the avionics software. Specific mission modes were established, and a thorough risk and failure analysis was conducted to ensure system robustness. Code development was approached with an emphasis on verification and validation, ensuring the software met the set requirements and operated optimally within the mission environment.

Throughout this project, it has been demonstrated that, with careful design and methodical implementation, it is possible to develop an efficient and reliable avionics system for a CubeSat with a specific payload. The solutions and decisions made during this process are not only relevant to this particular project but also have the potential to inform future developments in the realm of CubeSats and space exploration.

Acrónimos

OBC *On Board Computer*

SW *Software*

HW *Hardware*

GPIO *General Purpose Input/Output*

VRM *Voltaje Inverso Máximo*

ISA *International Standard Atmosphere*

NO *Normally Open*

NC *Normally Closed*

FCEM *Fuerza Contra Elecro Motriz*

I2C *Inter-Integrated Circuit*

SCL *Serial Clock*

SDA *Serial Data*

RF *Radio Frecuency*

CSM *Misión del CubeSat*

MSN *Misión*

REQ *Requisito*

VEH *Vehículo*

PWR *Power*

CAM *Cámara*

LCT *Lectura*

STR *Estructura*

SNS *Sensor*

ALT *Altímetro*

INA *INA260*

LSR *Láser*

GEN *Generador*

SOL *Solenoide*

WP *Work Package*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1. Motivaciones

Dentro del marco del estudio científico, así como en la aplicación de soluciones de ingeniería para distintas tecnologías, uno de los grandes retos a los que se ha enfrentado la humanidad es el de escapar de la atmósfera terrestre. Los retos tecnológicos que supone dicho hito son muy variados, pero sin duda, el más característico es el de disponer de un medio de inyección que sea capaz de aportar un incremento de energía cinética y potencial tales que permitan vencer a la gravedad terrestre.

Una vez obtenido un medio de inyección en dicho entorno, el siguiente reto es el de enfrentarse al ambiente espacial, el cual se presenta como un entorno complicado tanto a nivel térmico, como a nivel de radiación o presión. Dicho ambiente es completamente distinto al que se encuentra a un nivel cercano al nivel del mar en la corteza terrestre, lo que obliga a buscar soluciones de ingeniería que permitan una misión espacial.

En este contexto, surge el concepto de CubeSats, pequeños satélites modulares y de bajo costo que han revolucionado la industria espacial. Los CubeSats han permitido a universidades, empresas y gobiernos desarrollar misiones espaciales de forma más asequible y accesible, lo que ha llevado a un aumento en la investigación y el desarrollo de nuevas tecnologías en el campo de la exploración espacial.

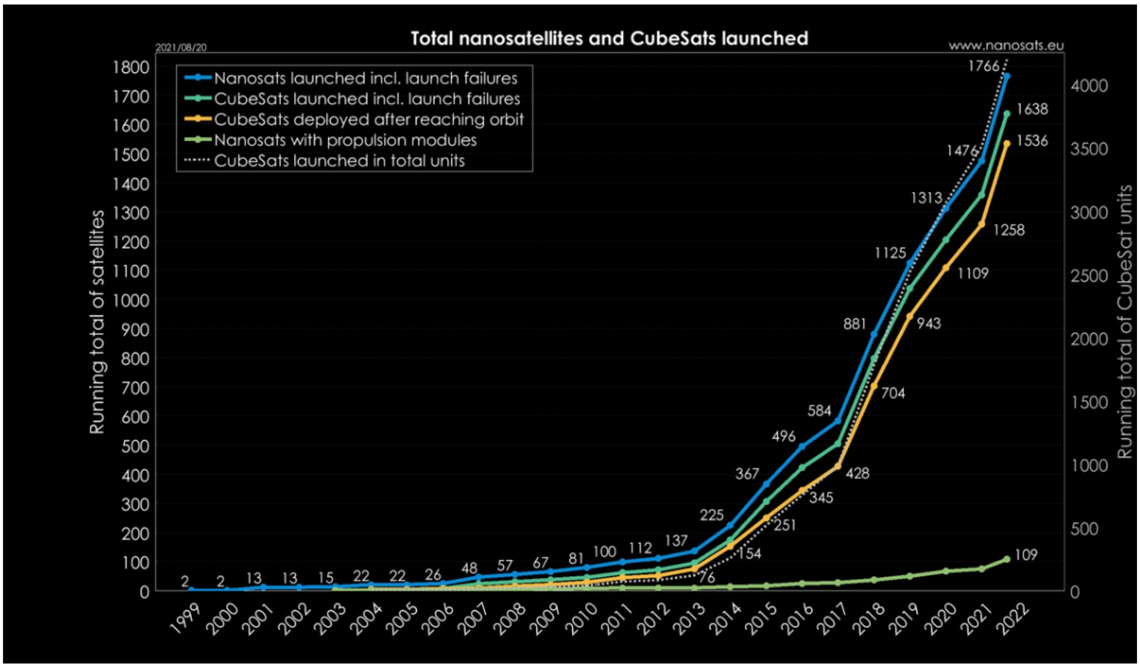

Figura 1.1: Evolución de lanzamiento de nanosats y CubeSats. Imagen de [1]

En los últimos años, los CubeSats han cobrado relevancia en el ámbito espacial. Estos pequeños satélites modulares, normalmente con un volumen de 1U, 2U o 3U (unidades cúbicas de 10x10x10 cm), han demostrado ser una solución eficiente y rentable para la realización de misiones espaciales. Gracias a su tamaño reducido y a la estandarización en su diseño, los CubeSats permiten una mayor accesibilidad al espacio y un menor costo en comparación con los satélites tradicionales. La evolución de los CubeSats ha sido posible gracias a los avances en miniaturización de componentes electrónicos y la aparición de tecnologías de bajo costo que han permitido su fabricación y puesta en órbita a precios cada vez más reducidos. Este auge en el desarrollo de CubeSats ha abierto nuevas oportunidades para la investigación científica, la observación de la Tierra, las comunicaciones y la exploración espacial.

La evolución de los CubeSats también ha permitido el desarrollo de nuevas tecnologías en el espacio. Se han creado nuevas tecnologías para el lanzamiento y la recuperación de los CubeSats, así como para la gestión y la comunicación de los mismos en el espacio. Estas nuevas tecnologías han permitido a las organizaciones y empresas crear soluciones innovadoras para problemas que antes eran imposibles de resolver.

Por ejemplo, los CubeSats se han utilizado para medir la radiación en el espacio, lo que ha permitido una mejor comprensión de los efectos de la radiación en el cuerpo humano y ha

permitido el desarrollo de nuevas tecnologías para proteger a los astronautas en el espacio. Los CubeSats también se han utilizado para estudiar los campos magnéticos del espacio, lo que ha llevado a una mayor comprensión de la física del espacio.

Otro avance en la evolución de los CubeSats ha sido su capacidad para trabajar juntos en grupos para llevar a cabo misiones más complejas. Esto se ha logrado mediante el desarrollo de tecnologías de acoplamiento y enlace que permiten que los CubeSats trabajen juntos en grupos. Esta capacidad ha permitido a las organizaciones y empresas crear soluciones más complejas y completas para una amplia variedad de problemas en el espacio.

Figura 1.2: ESA- RACE double CubeSat mision. Imagen de [2]

Uno de los desafíos más prometedores en la exploración espacial es el de la transmisión de energía mediante microondas. Esta tecnología tiene el potencial de revolucionar la forma en que se gestiona la energía en el espacio, permitiendo la transferencia de energía de forma inalámbrica y eficiente entre sistemas espaciales. La transmisión de energía por microondas se basa en la conversión de energía eléctrica en ondas electromagnéticas en la banda de frecuencias de microondas, que luego son transmitidas a través del espacio a un receptor que convierte las microondas de nuevo en energía eléctrica. Esta tecnología puede ser aplicada en diversas misiones espaciales, desde el suministro de energía a satélites en órbita hasta la alimentación de vehículos espaciales en misiones más allá de la órbita terrestre.

La demostración de esta tecnología en un entorno espacial se plantea como una oportunidad para avanzar en la comprensión de sus aplicaciones y beneficios en el ámbito espacial. La utilización de CubeSats en esta tarea se presenta como una solución ideal debido a su simplicidad, bajo costo y accesibilidad al entorno espacial. La demostración de la tecnología de transmisión de energía mediante microondas en un CubeSat permitiría validar su funcionamiento en el espacio y sentar las bases para futuras aplicaciones en misiones de mayor envergadura.

Para llevar a cabo esta misión, se propone utilizar un vehículo inyector basado en un globo aerostático que permita elevar el CubeSat a una altura en la que las condiciones sean similares a las del entorno espacial. La elección de un globo aerostático como vehículo inyector se debe a su capacidad para alcanzar altitudes elevadas y simular condiciones espaciales sin la necesidad de un lanzamiento espacial convencional, lo que reduce los costos y la complejidad de la misión. Además, este tipo de lanzamiento permite un mayor control sobre las condiciones de la prueba y facilita la recuperación del CubeSat tras su descenso, permitiendo la reutilización del satélite y el análisis de los resultados obtenidos.

En resumen, este trabajo busca demostrar la viabilidad y el potencial de la tecnología de transmisión de energía por microondas en un entorno espacial utilizando un CubeSat como plataforma de prueba. La combinación del enfoque práctico y económico que ofrecen los CubeSats con la prometedora tecnología de transmisión de energía mediante microondas, tiene el potencial de abrir nuevas posibilidades en la exploración y explotación del espacio, permitiendo una mayor eficiencia energética y una mayor flexibilidad en la gestión de recursos en misiones futuras.

1.2. Objetivos y Misión

El primer paso a la hora de desarrollar este proyecto será el de conocer sus objetivos, así como la misión que se llevará a cabo para poder completarlos. Combinando los objetivos que se quieren alcanzar junto a la misión (que definirá las limitaciones y el entorno), se podrán desarrollar unos requisitos a distintos niveles para la arquitectura del subsistema.

El objetivo del proyecto es claro: el desarrollo teórico de un subsistema de aviónica para un

CubeSat con el fin de probar la tecnología de transmisión de energía por método de microondas. De este modo, a falta de conocer y dar forma a la arquitectura del subsistema, se puede comenzar a intuir que este contará con un sistema de antenas que trabajen con microondas. Si bien el análisis y especificación de la transmisión recaería a otro subsistema (en este caso, el de payload), el subsistema de aviónica tendrá como objetivo principal el de suministar potencia y control a la payload, haciendo de interfaz entre esta y el resto del satélite a nivel de funcionamiento.

La aviónica a desarrollar contará con un ordenador de a bordo, un módulo de batería que suministrará potencia al satélite, y un conjunto de componentes electrónicos que permitirán llevar a cabo las funcionalidades del satélite. En este trabajo se afrontará el diseño de dicha electrónica, de modo que a partir de la definición de los requisitos extraídos al subsistema (a partir de la misión y todas las peculiaridades que giran en torno a ella) se pueda conformar una arquitectura final. Una vez obtenida la arquitectura que definirá el hardware, se dearrollará un software que permita la correcta ejecución de las funciones del satélite a lo largo de la misión.

En teoría, cualquier frecuencia de radio podría usarse para transferir energía a través de ondas electromagnéticas. Sin embargo, las ondas de radio de baja frecuencia se atenúan rápidamente a medida que se propagan, lo que significa que se necesitaría una gran cantidad de energía para transferir una cantidad útil de energía a través de grandes distancias.

En general, las ondas de radio de alta frecuencia, como las microondas, son más efectivas para transferir energía a través de grandes distancias. La frecuencia óptima para la transferencia de energía por microondas depende de varios factores, como la distancia de transmisión, la cantidad de energía que se desea transferir y la eficiencia de las antenas utilizadas.

En la práctica, las aplicaciones de transferencia de energía por microondas suelen utilizar frecuencias en el rango de varios gigahertz (GHz), que se encuentran en el espectro de las microondas. Sin embargo, la elección de la frecuencia óptima dependerá de las especificaciones de la aplicación y del equipo disponible.

La misión que se llevará a cabo para desarrollar el proyecto vendrá limitada en gran medida por el vehículo lanzador elegido, pues este delimitará completamente el entorno al que se exponga el satélite. Cabe recalcar desde el principio que el satélite no realizará ningún tipo de control sobre el vehículo lanzador, de modo que, si bien este vehículo será elegido por su capacidad de transportar al satélite a una situación equivalente (o muy parecida) a la espacial, la actuación del vehículo estará definida desde el inicio, cerrando el entorno al que se verá expuesto el satélite desde el primer momento (y sin capacidad de alterarlo por necesidades de

desarrollo del satélite).

En el caso de este proyecto, se parte de un globo aerostático como vehículo lanzador. De este modo, quizá sería más apropiado hablar de vehículo de transporte en vez de vehículo lanzador o inyector, pues el vuelo que realizará el satélite será unido solidariamente al vehículo. La idea detrás de la elección de un globo aerostático como vehículo elegido parte de la capacidad de estos para elevarse a alturas a las cuales las condiciones atmosféricas son muy parecidas a las espaciales. A 20 km de altura respecto al nivel del mar, la densidad del aire es de en torno a un 7 % de la densidad a nivel del mar. A 33 km, esta cifra baja por debajo del 1 % (estimaciones realizadas a partir del modelo ISA de atmósfera). En dichas condiciones se podrá experimentar, en primera aproximación, la tecnología a probar por el satélite.

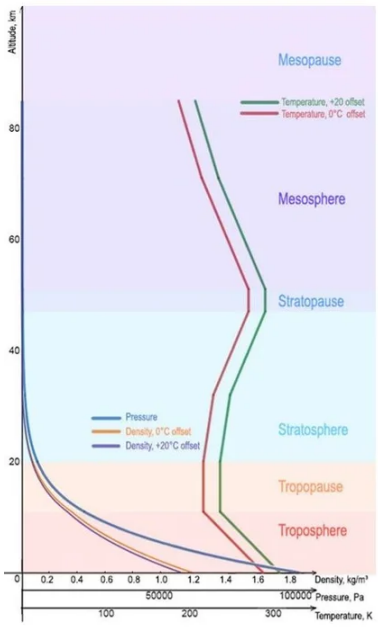

Figura 1.3: Perfiles de las variables presión, densidad y temperatura en la atmósfera ISA e ISA+20. Imagen de [3]

Como primer paso para estudiar la tecnología de transmisión de energía en el espacio a través de microondas se estudiará dicha transmisión en las condiciones que posibilita el globo como primera aproximación. Una vez definida la misión a realizar, se pueden establecer los objetivos principales del proyecto que se está realizando.

1.3. Estado del arte

En las últimas décadas, los CubeSats han emergido como una herramienta revolucionaria en la exploración espacial. Originalmente concebidos como herramientas educativas, estos pequeños satélites han evolucionado hasta convertirse en plataformas versátiles para aplicaciones científicas y comerciales. La estandarización de su diseño y la miniaturización de componentes han sido factores clave en su proliferación y éxito.

El campo de la comunicación y gestión de energía ha visto avances significativos. En particular, la tecnología de transmisión de energía por microondas ha ganado atención los últimos años por su potencial para mejorar la eficiencia energética en misiones espaciales, si bien dicha tecnología aun no se encuentra extendida en el sector espacial. Estas tecnologías no solo mejoran la autonomía de los CubeSats sino que también abren posibilidades para nuevas aplicaciones, como la transferencia de energía inalámbrica entre naves espaciales.

En los siguientes subapartados, se explorarán tanto el estado del arte actual, como las futuras tendencias en cuanto a software y hardware en la aviónica de Cubesats. Como referencia, se ha tomado el último informe realizado por la NASA acerca de la aviónica de pequeños satélites [25].

1.3.1. Estado del Arte y Futuras Tendencias en Hardware de Cubesats

En los primeros modelos de CubeSats, el diseño de hardware estaba limitado por la disponibilidad y el tamaño de los componentes electrónicos. Estos modelos iniciales a menudo utilizaban componentes de tamaño estándar, lo que restringía su capacidad para albergar sistemas avanzados en un espacio limitado. Las estructuras eran más pesadas y menos eficientes en términos de uso del espacio interno.

Los factores clave en cuanto al estado del arte del hardware de la aviónica de satélites pequeños son la escala del satélite y el diseño arquitectónico. Con respecto a la escala del satélite, los pequeños satélites tienen sistemas aviónicos de baja escala en tamaño, peso, energía y costo, adaptándose a misiones con mayor tolerancia al riesgo y menores requisitos de fiabilidad.

Por el lado del diseño arquitectónico, este puede ser centralizado o descentralizado, simple o tolerante a fallas, y modular o monolítico. La tendencia es hacia diseños más centralizados y

con enfoques en la tolerancia a la radiación.

En la figura 1.4, se muestran los principales componenentes de un CubeSat actual que se encuentra a la vanguardia en cuanto a tecnología, donde destaca su configuración modular y sus modos de ensamblaje.

Figura 1.4: Vista detallada de los componentes de un CubeSat de 6U contemporáneo. Imagen de [4]

Con respecto a las futuras tendencias en cuanto al hardware, se encuentra:

- **Arquitecturas modulares:** Federadas (cada subsistema es autónomo) e integradas (funcionalidad distribuida y compartida).
- **Redes de constelaciones:** Nuevas oportunidades con la sincronización y comunicaciones entre satélites.
- **Tecnologías emergentes:** Procesadores y FPGA resistentes a la radiación, memoria y bloques de función electrónicos, interfaces eléctricas CDH, y esquemas de mitigación de radiación.

Si bien estas tendencias futuras presentan los principales caminos en cuanto a la investigación actual, ya se han probado versiones iniciales de dichas tecnologías, de modo que se espera que en los próximos años se integren en mayor medida dentro del sector espacial.

1.3.2. Estado del Arte y Futuras Tendencias en Software de Cubesats

Por el lado del software, inicialmente, este era bastante básico, centrado principalmente en funciones operativas simples como la recopilación de datos y la comunicación básica con la Tierra. La capacidad de procesamiento era limitada y las funcionalidades de software estaban restringidas por el hardware disponible.

Con respecto al estado del arte, actualmente la elección de entornos de desarrollo y sistemas operativos depende de los recursos de memoria y computacionales, presupuesto y experiencia previa. La complejidad y fiabilidad aumentan con la capacidad de procesamiento de data handling y otros procesadores. Se han utilizado comunmente lenguajes como C, C++, Python y Arduino.

Respecto a las futuras tendencias por el lado del software en CubeSats, estas vienen dominadas por:

- **Tendencias de software de vuelo:** Incluyen sistemas operativos para procesadores multicore, inteligencia artificial, y middleware para constelaciones de SmallSats.
- **Autonomía del sistema:** Avances en sistemas autónomos para la gestión, dirección y control de todos los subsistemas y funciones.
- **Integración y digitalización:** La aviónica de los pequeños satélites no se considera un componente aislado, sino como parte de un ecosistema aviónico integrado que incorpora todas las tecnologías digitales.

1.4. Planificación Temporal

Para la realización de este proyecto, se ha dividido el trabajo a realizar en distintos paquetes de trabajo (Work Package, del inglés). Dichos paquetes de trabajo, han sido desarrollados desde el 15 de septiembre de 2022, hasta el 24 de octubre de 2023. Si bien no se ha llevado a cabo un registro de las horas dedicadas al proyecto, el dato aproximado debe situarse en torno a 450+-50h.

La descripción de los objetivos de cada paquete de trabajo se detalla en la tabla 1.1

WP	OBJETIVOS
1	Analizar el entorno de la misión y los requisitos que se le imponen al subsistema de la aviónica
2	Desarrollar la estructura de la aviónica, definiendo las comunicaciones a nivel de datos y potencia
3	Dimensionado del balance de potencias del sistema de transmisión de microondas
4	Elección de los componentes que conforman la electrónica del subsistema
5	Diseño de la arquitectura final del hardware
6	Análisis de los modos del satélite y requisitos del software
7	Análisis de riesgos y fallos del software
8	Desarrollo de código para la misión
9	Validación y test del código

Tabla 1.1: Descripción de los paquetes de trabajo del proyecto.

Dichos paquetes de trabajo, han sido realizados siguiendo el esquema temporal mostrado en la figura 1.5.

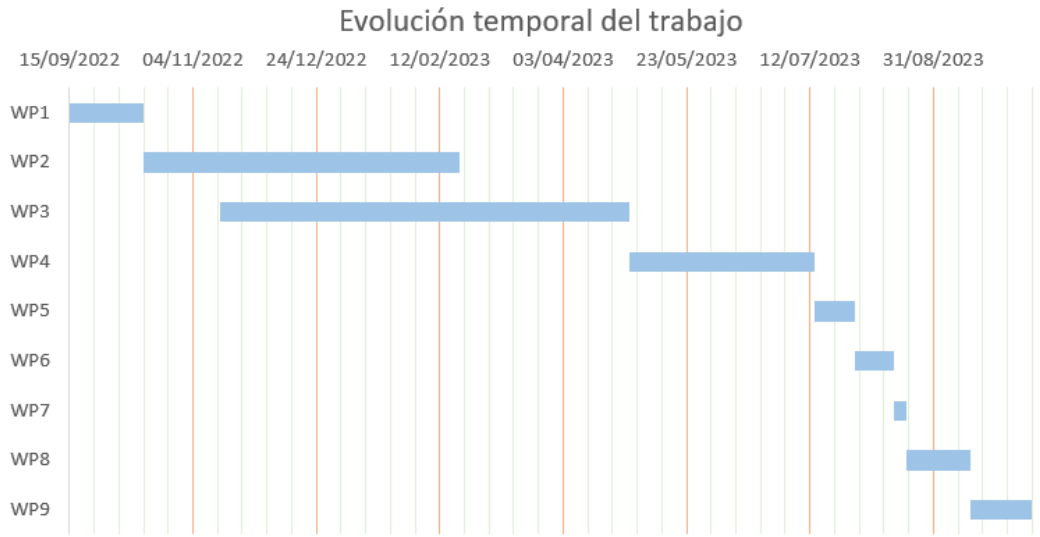

Figura 1.5: Diagrama Gantt del proyecto

1.5. Estructura de la Memoria

La estructura que va a seguir esta memoria es la siguiente:

- **Capítulo 1:** En este capítulo se recogerán los objetivos y motivaciones que dan lugar a la misión, así como la organización temporal con la que se ha desarrollado el trabajo.
- **Capítulo 2:** Una vez se ha definido la misión, se estudian todas las limitaciones y requisitos que se extraen de ella para realizar el diseño de la aviónica del satélite. En este apartado se consideran desde las funcionalidades que debe aportar la aviónica, hasta los requisitos extraídos a partir del propio vehículo lanzador del satélite.
- **Capítulo 3:** Tras la definición de los requisitos de la aviónica, se desarrolla la arquitectura del hardware que la conforma. Para ello, se comienza con una distribución de los componentes de la aviónica, para después realizar la elección de cada uno de los componentes y obtener una arquitectura final. En este capítulo, se aborda también el coste energético del satélite durante la misión, el cual es vital para realizar la elección del sistema de potencia.
- **Capítulo 4:** En este capítulo, se lleva a cabo el desarrollo del software de la aviónica. Para ello, se consideran todos los modos en los que se encontrará el satélite a lo largo de la misión, así como los principales riesgos a los que se enfrentará. Tras el desarrollo del código que conforma el software embebido, se pone a prueba mediante la validación del código por medio de un test que simula la misión.
- **Capítulo 4:** Finalmente, se obtienen las principales lecciones aprendidas mediante este proyecto, así como se proponen los posibles trabajos futuros y vías de mejora del diseño de la aviónica desarrollado.

Capítulo 2
Requisitos y entorno de la misión

Una vez conocidos los objetivos de la misión, se puede proceder a desarrollar los requisitos de esta misma, de modo que el diseño del hardware y software de la aviónica responderá al cumplimiento de los requisitos definidos.

La función de este capítulo es delimitar completamente el entorno a partir del cual se debe realizar el diseño del subsistema de aviónica, de modo que este finalice con los requisitos a partir de los cuales distribuir y conocer los componentes que conformen la electrónica, así como las características que los deben de regir y, por tanto, criterios para realizar la elección entra las distintas opciones que se encuentren en el mercado.

A la hora de abordar el diseño de la arquitectura de la aviónica, algunas de las constraints serán consecuencia directa de la elección del vehículo lanzador, pues este no solo definirá la altura máxima de vuelo, sino también la velocidad ascensional, tiempo de vuelo, velocidad de caída, trayectoria recorrida... Se estudiará entonces la respuesta operacional del vehículo lanzador elegido y sus consecuencias en el diseño de la aviónica.

2.1. Vehículo Lanzador y Condiciones del Vuelo

El uso de globos aerostáticos como plataforma de lanzamiento y transporte de satélites ha tenido una notable evolución en los últimos años. Estos globos, rellenos de gases más ligeros que el aire, como el helio, pueden elevarse a grandes altitudes y proporcionar una plataforma estable y de bajo costo para transportar cargas útiles, como los CubeSats, al espacio cercano.

Un globo aerostático opera mediante principios físicos muy simples. Cuando se llena un globo con un gas menos denso que el aire, el globo experimenta una fuerza ascendente llamada

flotabilidad. Esta fuerza puede ser suficiente para llevar una carga útil a altitudes de hasta 30 km o más, en la estratosfera, donde las condiciones se asemejan mucho a las del espacio exterior.

A estas altitudes, el aire es muy poco denso y la temperatura es muy baja, similares a las condiciones que se encontrarían en el espacio. Estas condiciones hacen que los globos estratosféricos sean ideales para muchas aplicaciones que normalmente requerirían un lanzamiento espacial costoso y complicado.

Los globos aerostáticos son generalmente controlados desde el suelo. Los operadores pueden ajustar la altitud del globo liberando gas para bajar o liberando lastre para subir. Sin embargo, una vez que el globo ha alcanzado la estratosfera, puede flotar a una altitud constante durante varias horas o incluso días, proporcionando una plataforma estable para las observaciones científicas o la prueba de tecnologías espaciales.

Los globos aerostáticos son generalmente controlados desde el suelo. Los operadores pueden ajustar la altitud del globo liberando gas para bajar o liberando lastre para subir. Sin embargo, una vez que el globo ha alcanzado la estratosfera, puede flotar a una altitud constante durante varias horas o incluso días, proporcionando una plataforma estable para las observaciones científicas o la prueba de tecnologías espaciales.

En el contexto de este proyecto, el globo aerostático se utilizará como un "vehículo de transporte"para un CubeSat. El CubeSat estará anclado al globo y seguirá la misma trayectoria que este. Esto permitirá que el CubeSat sea expuesto a condiciones muy similares a las del espacio, pero sin los costos y riesgos asociados con un lanzamiento espacial.

Este método de "lanzamiento"tiene varias ventajas. En primer lugar, es considerablemente más barato y más fácil de organizar que un lanzamiento espacial tradicional. Por otro lado, proporciona una forma de probar y validar tecnologías espaciales en un entorno muy similar al espacio, antes de comprometerse con un lanzamiento espacial completo.

En el caso de estudio desarrollado por este proyecto, se partirá de un modelo de globo aerostático modelizado computacionalmente y se heredarán sus limitaciones y restricciones durante un vuelo. El modelo utilizado se nutre del trabajo expuesto en Perdiguero García, B. Diseño y simulación de un modelo físico-numérico para un globo estratosférico [5]. En él, el autor presenta un código desarrollado para simular y reproducir la trayectoria de un globo de helio. En el trabajo se presentan distintos modelos, pero como referencia para el diseño del

CubeSat que aborda esta memoria, se tomará el modelo 3D con vientos implementados, el cual se presenta como el modelo más complejo (y semejante a un vuelo real).

A partir del código mostrado en el trabajo, se pueden graficar distintas figuras de interés a cerca del vuelo de referencia. El vuelo se realizará, según comenta el autor, con la condición de mayores vientos que se presenta entre los modelos. Esto se consigue al no acotar muy finamente las variables aleatorias, de modo que prácticamente a cada segundo se genere viento con una magnitud, dirección y sentido diferentes.

La primera gráfica a analizar, 2.1, es la de altitud respecto al tiempo, de la cual se pueden extraer tanto la altura máxima a la que se someterá al satélite durante el vuelo, como las mayores velocidades ascensionales presentadas en él.

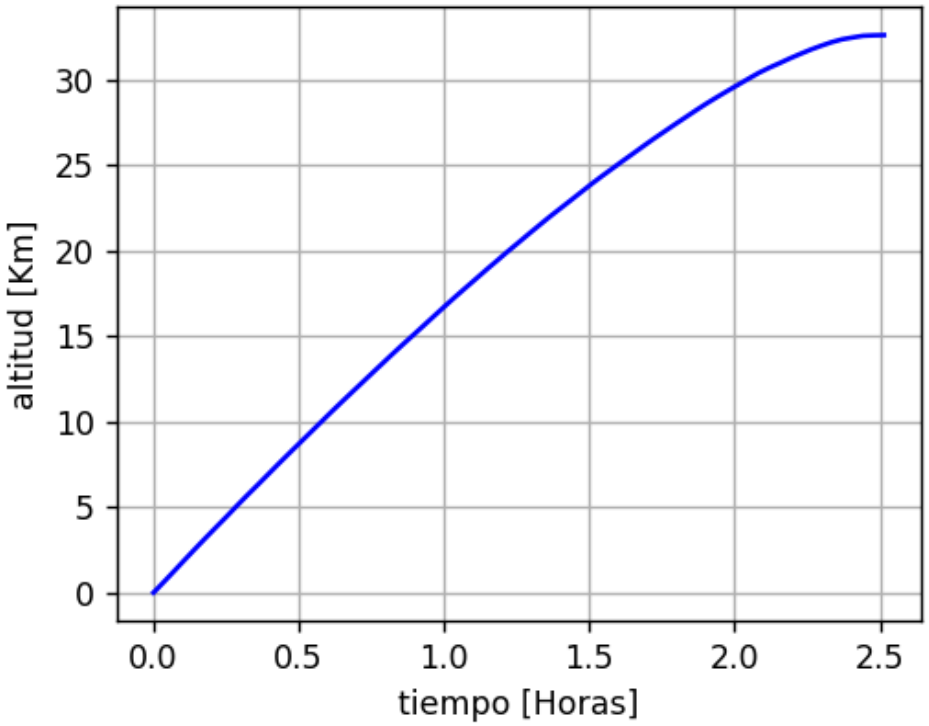

Figura 2.1: Representación de la altitud respecto al tiempo en el vuelo de referencia (Elaboración propia a partir de [5])

En la figura anterior, se muestra como la condición de flotabilidad del globo (en la cual el globo alcanza un equilibrio y mantiene su altitud de vuelo, sin ascender ni descender), se da pasadas 2,4 horas, a 33 kilómetros del suelo aproximadamente. Se tomará dicha altura como altura máxima de vuelo para el diseño de la aviónica del satélite, de modo que los

componentes elegidos para la arquitectura final de la aviónica deberán ser capaces de trabajar a dicha altura. Este será el caso, por ejemplo, del altímetro barométrico, cuyo rango de medición deberá permitir superar los 33 kilómetros.

Por otro lado, se muestra como la mayor velocidad ascensional se da al inicio del vuelo, alcanzando una velocidad de en torno a 5 m/s en su punto de mayor velocidad.

La siguiente gráfica a estudiar, 2.2, será la de la velocidad total del vehículo con respecto al tiempo, la cual mostrará la mayor velocidad a la que se expondrá al vehículo, y, por ende, al satélite.

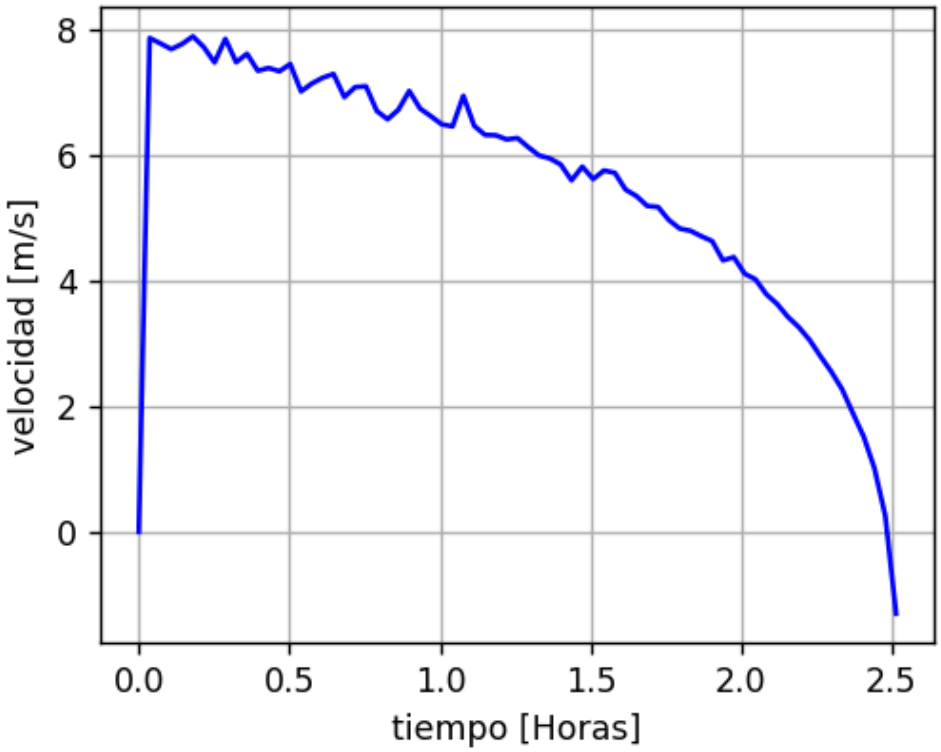

Figura 2.2: Representación de la velocidad total respecto al tiempo en el vuelo de referencia (Elaboración propia a partir de [5])

Analizando la figura, se extrae una velocidad máxima durante el vuelo cercana a 8 m/s. Dicha velocidad máxima se dará al inicio del vuelo, de forma coincidente con la velocidad ascensional máxima analizada anteriormente.

Por último, se presenta la trayectoria completa en 3 dimensiones del vehículo 2.3. Dicha

trayectoria modelizada, se presenta como la referencia de vuelo y punto de diseño para la aviónica del satélite.

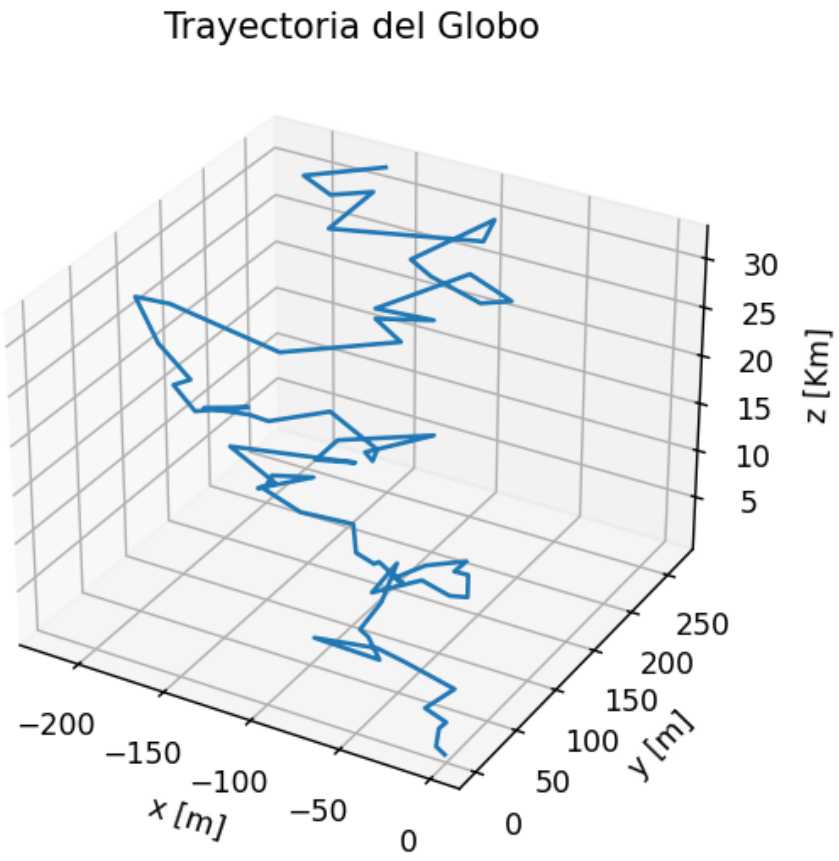

Figura 2.3: Representación de la trayectoria final del vuelo de referencia (Elaboración propia a partir de [5])

En este apartado se ha estudiado un entorno similar al que se expondrá al satélite que se plantea diseñar en este proyecto durante su misión, de modo que, a la hora de diseñar, se partirá de las estimaciones obtenidas a partir del modelo computacional propuesto.

2.2. Requisitos del Subsistema de Aviónica

En este apartado, se abordará la definición de los requisitos de la aviónica para la misión del CubeSat. Los requisitos son especificaciones o criterios claros que describen las capacidades,

características o cualidades esperadas de un sistema. En el contexto de la aviónica, estos requisitos determinan qué funcionalidades debe tener el sistema, cómo debe desempeñarse y bajo qué condiciones debe operar.

Al establecer los requisitos, se sientan las bases para el diseño posterior del subsistema. Cada requisito tiene una implicación directa en la forma en que se diseñará el sistema de aviónica:

- **Funcionalidad:** Los requisitos determinan qué debe hacer el sistema, es decir, cuáles son sus funciones primordiales. Esto puede incluir, por ejemplo, lectura de altitud, comunicación de telemetría o monitorización de sistemas de la aeronave.
- **Desempeño:** Estos criterios especifican cómo debe actuar el sistema en términos de velocidad, precisión, capacidad de respuesta y confiabilidad. Por ejemplo, puede establecerse un requisito de que el sistema de navegación actualice su posición cada segundo con una precisión de un metro.
- **Condiciones Operativas:** Los requisitos también establecen bajo qué condiciones debe funcionar el sistema. Esto puede incluir aspectos como temperaturas extremas, altitudes elevadas o interferencias electromagnéticas.

Una vez explicada la necesidad de una buena definición de requisitos para realizar el diseño de la aviónica, se presentan en la siguiente tabla los requisitos recogidos para este proyecto 2.1. Dichos requisitos recopilan la información de la misión, así como de las tareas que debe cumplir la aviónica para hacer viable la misión.

No.	Título	Requisito	Explicación	Método de verificación
CSM-MSN-REQ-1	Trasmisión de energía a distancia	Potencia eléctrica debe ser transferida de un sub-satélite a otro	La tecnología a demostrar es la trasmisión de energía a distancia en condiciones similares a las orbitales	Lectura de potencia trasmitida al sub-satélite receptor
CSM-MSN-REQ-2	Trasmisión de potencia por microondas	La trasmisión de potencia se debe realizar en frecuencia de microondas	La demostración de tecnología busca probar la viabilidad de la trasmisión en esta frecuencia	Inspección

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-VEH-REQ-1	Dimensiones del subsistema	El subsistema de aviónica no debe ocupar un volumen superior a 2U en el caso del cuerpo grande, y de 1 U en el pequeño	La aviónica se encuentra embebida dentro de la estructura del satélite, luego los componentes deben poder albergarse en su interior	Inspección
CSM-VEH-REQ-2	Altitud máxima de vuelo	Los componentes elegidos para la aviónica deben poder trabajar a una altitud de vuelo superior a 33 km	Durante el vuelo, la altitud máxima que alcanza el vehículo lanzador puede sobrepasar los 33 km	Obtención de lecturas y mediciones ininterrumpida
CSM-PWR-REQ-1	Módulo de batería	Se debe integrar un módulo de batería capaz de suministrar energía a la aviónica y a la payload	El módulo de batería debe tener una capacidad máxima superior a la energía demandada por la aviónica y la payload en conjunto a lo largo de toda la misión	Cálculo de coste energético
CSM-PWR-REQ-2	Control de voltaje	Se debe integrar un controlador de voltaje entre la batería y el OBC	La tensión de suministro del OBC debe mantenerse estable y con el valor nominal que indique el fabricante	Inspección
CSM-CAM-REQ-1	Imágenes	El satélite debe tomar imágenes cada 20 minutos durante el vuelo	Una de las misiones secundarias del proyecto es la de obtener imágenes del vuelo	Recopilación de imágenes
CSM-CAM-REQ-2	Vídeo	El satélite debe realizar un video en el momento del despliegue de la estructura	Se busca poder analizar el despliegue de la estructura tras el vuelo.	Recopilación de vídeo
CSM-LCT-REQ-1	Lectura de Potencia	Se debe realizar una lectura de la potencia trasmitida	La lectura de la potencia trasmitida es esencial para comprobar la viabilidad de la tecnología a probar	Recopilación de datos
CSM-LCT-REQ-2	Continuidad en la lectura	La lectura de potencia se debe realizar a lo largo de toda la trasmisión	Para permitir un posterior análisis de la tecnología, la lectura de potencia debe ser pseudo-continua durante toda la trasmisión	Recopilación de datos

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-STR-REQ-1	Despliegue estructural	Se debe incluir un componente que permita realizar el despliegue de la estructura	La estructura se encontrará retraída al inicio de la misión, y el despliegue debe ser mediante una orden del OBC a un componente actuador	Inspección
CSM-SNS-REQ-1	Medición de separación estructural	Se debe incluir un componente que permita realizar una medición de la distancia que separa las subpartes de la estructura	La trasmisión de energía debe darse una vez se haya verificado que el despliegue estructural ha sido exitoso y la distancia entre los cuerpos del CubeSat es de 50 cm	Recopilación de datos
CSM-ALT-REQ-1	Lectura de altura de vuelo	Se debe incluir un altímetro con la capacidad de hacer un muestreo de la altura durante el vuelo	Se busca, como una misión secundaria, el conocer cómo se desarrolla el ascenso del vuelo y la velocidad ascensional del vehículo lanzador durante la misión.	Recopilación de datos
CSM-ALT-REQ-2	Lectura continua	La lectura del altímetro debe ser pseudo-continua durante toda la misión	Para poder reconstruir el ascenso durante la misión, se requiere una recogida de datos constante a lo largo de la misión	Recopilación de datos
CSM-SW-REQ-1	Bucle principal	El software debe contener un bucle principal que se ejecute durante toda la misión	El bucle principal será el encargado de llamar a todas las funciones de data-handling del software, por lo que se debe ejecutar toda la misión	Inspección del código
CSM-SW-REQ-2	Almacenamiento de datos	Se deben almacenar los datos recogidos por los componentes de lectura a lo largo de la misión	El análisis de la misión se realizará a partir de la información almacenada del vuelo	Inspección de datos almacenados

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-SW-REQ-3	Periodicidad de lecturas	Se deben establecer periodos de lectura que permitan una recogida de datos suficiente para cada componente	Cada componente tiene un fin en la aviónica, así como un funcionamiento propio. Se deben establecer periodos de lectura propios para cada componente, de modo que el análisis de los datos no se vea afectado por la cadencia de las muestras.	Inspección del código

Tabla 2.1: Requisitos del subsistema de aviónica

A partir de los requisitos, se comienza a esbozar la arquitectura del sistema de aviónica. Esta arquitectura se basará en asegurar que todos los requisitos se cumplan. Por ejemplo, si un requisito estipula que el sistema debe poder realizar una medición de la distancia que separa los 2 cuerpos del CubeSat, la arquitectura deberá incluir hardware y software capaces de gestionar dicha lectura.

Capítulo 3
Diseño de la arquitectura y Hardware de la aviónica

Como se introdujo en los capítulos anteriores, el primer paso para poder definir la arquitectura de la aviónica era el delimitar completamente cuales serían los criterios de diseño a partir de los cuales trabajar. Mediante el estudio del vehículo lanzador y la misión, en el 2 se ha realizado dicho estudio, el cual ha culminado en la definición de los requisitos del subsistema de aviónica del CubeSat.

En este apartado, se abordará el proceso completo de diseño del hardware de la aviónica. Para ello, se partira de las consignas detalladas en los capítulos anteriores para construir un esquema de la arquitectura, el cual contenga que componentes se deben añadir a la electrónica (en base a las funcionalidades que deba desarrollar el satélite).

Tras la definición del esquema de la arquitectura, se procederá a realizar la elección de cada uno de los componentes electrónicos de la aviónica. Para ello, se compararán los pincipales y más extendidos modelos que se encuentren en el mercado, en base a las necesidades establecidas por los requisitos. Una vez elegidos todos los componentes que conformen la arquitectura, se habrán definido a su vez todas las interfaces que se han de comunicar en la electrónica, de modo que se propondrá un modelo de arquitectura final que integre tanto los componentes como las conexiones entre estos.

Como punto a destacar, el cálculo del coste energético de la misión se realizará como uno de los últimos apartados de este capítulo. La razón de ser de esta decisión es que, para aproximar al máximo el consumo de potencia eléctrica durante la misión, se debe conocer el consumo específico de cada uno de los componentes que conforman la electrónica, de modo que estos deben haber sido seleccionados con anterioridad. En todos los apartados relacionados con la selección de un componente, se valorará el criterio de buscar componentes que reduzcan al

máximo el consumo eléctrico durante la misión, pues dicha decisión alargará la vida útil del satélite.

Finalmente, tras el cálculo del coste energético, se podrá realizar la elección del módulo de baterías que suministre energía electrica a la electrónica del satélite.

3.1. Esquema de la Arquitectura

Conocidos los requisitos asignados al subsistema de aviónica, se debe construir una arquitectura que los satisfaga. Dichos requisitos permiten conocer las características principales que deben presentar la mayoría de los componentes, pero se debe definir como se efectuará la distribución de potencia y de datos en el satélite.

Se partirá de una base en la cual se conoce una distribución de componentes a grandes rasgos (definida por la función de cada uno). Dicha representación se llevará a cabo mediante un diagrama de bloques, donde tras la elección de cada componente, se extrapolará la arquitectura final del subsistema.

Cabe resaltar la importancia de las interfaces entre los distintos componentes, pues si bien algunas vendrán definidas como premisa para el propio diseño, otras se extraerán a partir de la elección de los componentes (pues, por ejemplo, un sensor óptico puede utilizar una comunicación I2C con el ordenador de a bordo, pero se pueden encontrar soluciones en el mercado que utilizan otro tipo de comunicaciones como una simple a un puerto GPIO del ordenador).

Debido a las limitaciones en el volumen disponible para el subsistema de aviónica y a obtener una mayor simplicidad en la arquitectura, se utilizará una única computadora que realizará las tareas de data-handling asignadas al OBC. De este modo, dicho ordenador será el centro de procesamiento de la estructura de datos, y de él colgarán el resto de los componentes.

Otra opción que se encuentra ampliamente en proyectos de satélites es una distribución modular, donde existen varios procesadores comunicados entre sí y designados a funciones distintas. En este tipo de sistemas, el data-handling se efectúa de manera conjunta entre todos los procesadores.

En cuanto la misión principal del satélite, se busca demostrar la trasmisión de energía mediante microondas, de modo que el control de las antenas de microondas será muy limitado. Se buscará el emitir y recibir potencia trasmitida a las antenas, de modo que se realice una lectura de potencia aguas abajo y se pueda analizar la calidad de la trasmisión. La comunicación con el OBC será muy simple, este dará la orden de cuando trasmitir y recibirá los datos de la lectura de potencia aguas abajo.

Por último, cabe definir como se dará la distribución de potencia dentro del satélite. Se contará con un módulo de potencia que suministrará corriente a todo el satélite. Dicha tarea será realizada a partir de conectarse a nivel de potencia con el ordenador de a bordo, el cual, a su vez, distribuirá energía al resto de componentes. Por otro lado, se utilizará una segunda ruta, la cual trabajará a un voltaje superior y se conectará al sistema de despliegue estructural (el cual contará con un solenoide, componente que demanda un voltaje superior para su utilización).

El diagrama de bloques que se obtiene de los requisitos de potencia y del subsistema de aviónica se conforma en la figura 3.1.

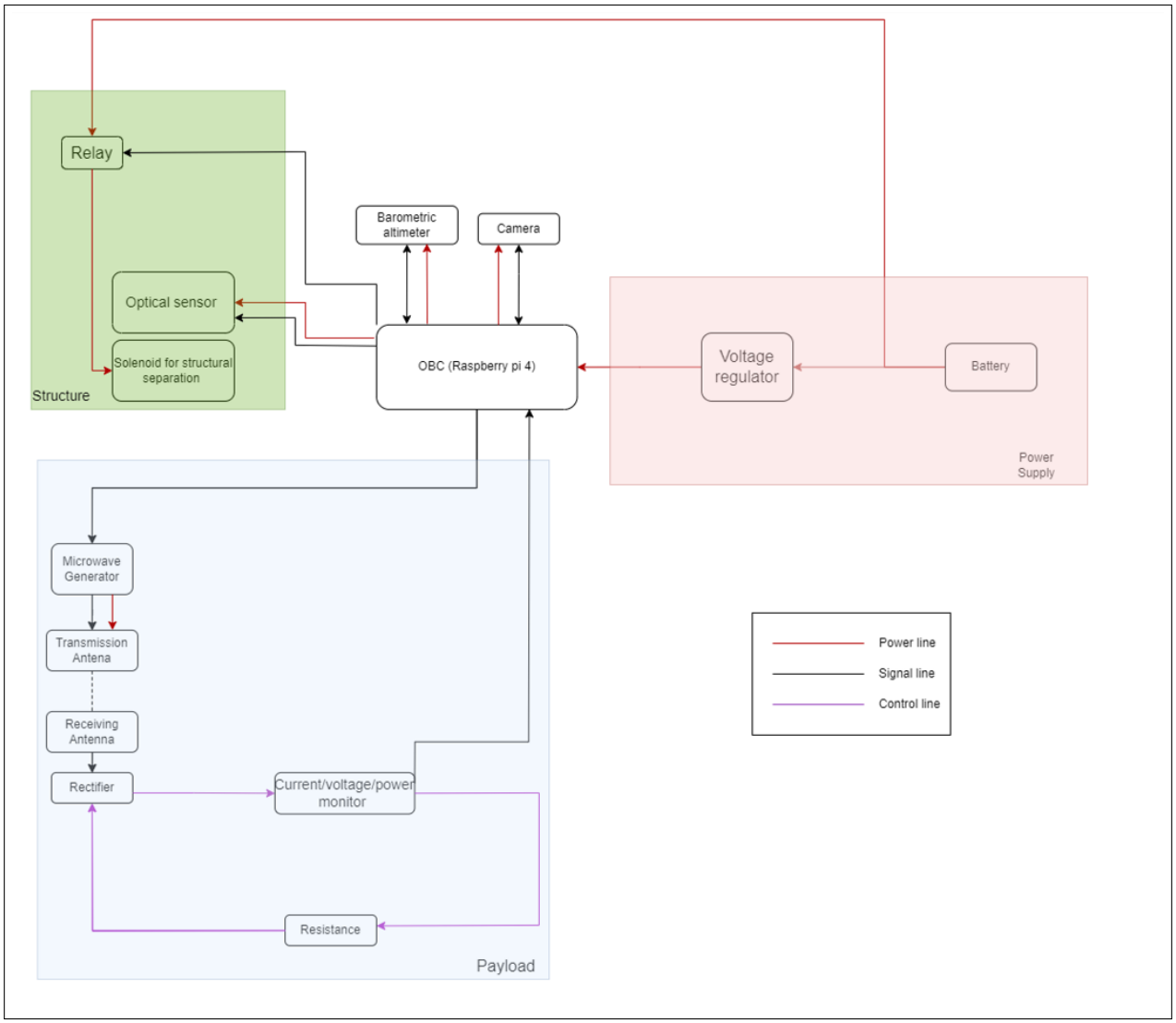

Figura 3.1: Diagrama de bloques de la aviónica del satélite (Elaboración propia)

En la figura 3.1, se muestran todos los componentes electrónicos que conforman la aviónica del satélite. En los siguientes apartados, se abordará la elección y diseño de cada uno de los bloques que conforman el diagrama, de modo que finalmente se obtenga una arquitectura final con todas las conexiones necesarias para conformar la aviónica.

3.2. Elección del Sistema de Transmisión de Microondas

3.2.1. Composición del Sistema de Generación de la Señal

En este apartado, se expondrá la configuración de un sistema de microondas prototipo, el cual servirá como modelo para dimensionar todos los inputs necesarios por parte de la aviónica que se está diseñando. La misión de la aviónica es la de proporcionar todos los requisitos que tenga el sistema de trasmisión de microondas, tanto a nivel de potencia como a nivel de control del mismo.

A la hora de realizar la elección del sistema, se considerarán los requisitos: CSM-MSN-REQ-1, CSM-MSN-REQ-2, CSM-LCT-REQ-1 y CSM-LCT-REQ-2. Como consecuencia, el sistema elegido debe ser capaz de transmitir energía a distancia en el espectro de las microondas y realizar una lectura de la potencia transmitida a través de un lector que realice una medición pseudo-continua a lo largo de toda la misión.

El primer paso para llevar a cabo un dimensionado del sistema de trasmisión será el de conocer que componentes los conforman. Se comenzará por el componente encargado de generar la señal. Para ello, se estudiará el método de generación más conveniente para la misión contando con las limitaciones que presenta el propio CubeSat que se está diseñando.

El reto a la hora de seleccionar un método de generación se encuentra las limitaciones de tamaño y potencia del satélite. Los generadores convencionales no respetan dichas limitaciones, por lo que se debe investigar otro método para la generación de la señal.

Los generadores de señal convencionales, a menudo empleados en entornos de laboratorio y pruebas de campo, tienden a ser grandes y consumen cantidades significativas de energía. Esto es incompatible con el diseño de un cubesat, que debe mantenerse compacto y eficiente desde el punto de vista energético. Los generadores de señal convencionales son demasiado grandes para alojarlos dentro de la estructura limitada de un cubesat y su consumo de energía podría agotar rápidamente las baterías del satélite.

La opción más viable para esta aplicación es un enfoque de dos partes: un chip generador de señal y un amplificador de señal. El chip generador de señal, en su forma más básica, es un componente de circuito integrado diseñado para producir una señal de microondas. Este chip es pequeño y consume una cantidad mínima de energía, lo que lo convierte en un candidato ideal para su uso en un cubesat.

Figura 3.2: Ejemplo de chip emisor de señal de microondas. Imagen de [6]

Por otro lado, la señal generada por el chip será de muy baja potencia, insuficiente para lograr la transmisión de energía de uno de los cuerpos del satélite al otro. De este modo, se añade al sistema de generación de la señal un amplificador. La función del amplificador es la de surtirse de la batería para amplificar la señal generada, de modo que la señal que se envíe a las antenas tenga la potencia suficiente como para cumplir con la misión principal de la operación.

Dado que el amplificador se alimentará de las baterías para amplificar la señal, durante la trasmisión de microondas el consumo energético del satélite aumentará notablemente, lo que obligará a transmitir energía en momentos específicos y designados de la operación, y no mediante todo el vuelo.

Por el lado del amplificador, este debe ser capaz de cumplir ciertos requisitos. En primer lugar, debe ser capaz de trabajar en consonancia con el chip generador de señal. En segundo lugar, debe poder amplificar la señal a un nivel que permita la transmisión de energía a pesar de las pérdidas inherentes a la transmisión a través del espacio.

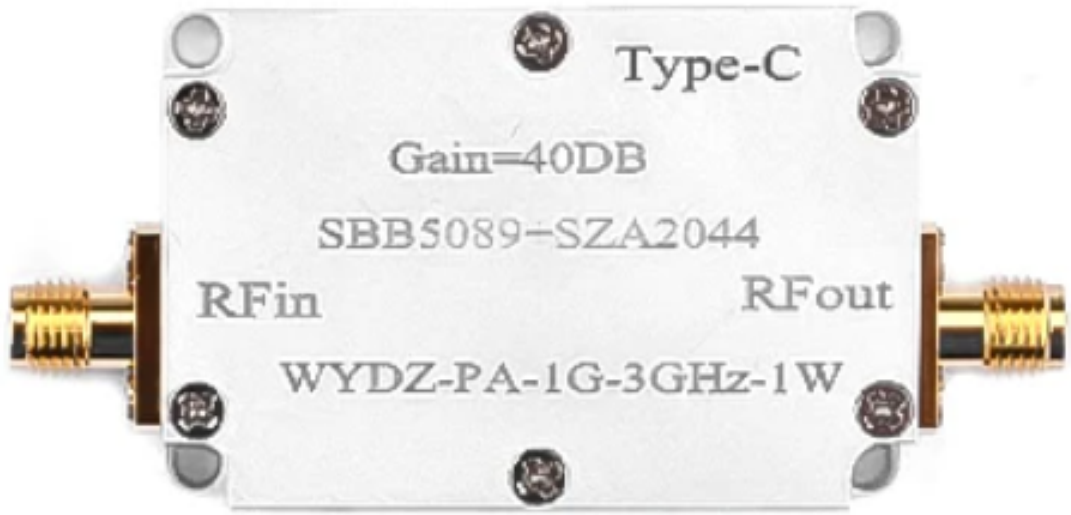

Figura 3.3: Modelo de amplificador de señal de referencia. Imagen de [7]

Es importante destacar que esta solución también tiene potencial para ser optimizada en función de las necesidades específicas de la misión. Por ejemplo, podrían seleccionarse diferentes modelos de chips generadores de señal y amplificadores de señal, dependiendo de las frecuencias de operación requeridas, los niveles de potencia necesarios y las limitaciones de tamaño y potencia del cubesat.

3.2.2. Definición de Antenas de Microondas

En este caso, se busca seleccionar antenas de microondas que permitan la transmisión de energía a través de un espacio de 50 centímetros, una distancia relativamente pequeña, pero que implica un reto técnico importante debido a las pérdidas de energía inherentes a la transmisión de microondas.

Las antenas de microondas son dispositivos que permiten la transmisión y recepción de señales de radio en el espectro de las microondas, es decir, frecuencias que van desde los 300 MHz hasta los 300 GHz. Estas antenas tienen múltiples aplicaciones, desde la comunicación satelital hasta la detección de objetos y la transmisión de energía.

En el caso del CubeSat que se está diseñando, se busca utilizar estas antenas para la transmisión de energía entre dos partes de la estructura del satélite, separadas por una distancia

de 50 cm. Una de las antenas se colocará en el cuerpo de mayor tamaño (2U) y la otra en el cuerpo más pequeño (1U). Al inicio de la misión, los 2 cuerpos se encontrarán unidos solidariamente. A lo largo de la misión, el ordenador central dará la orden de realizar el despliegue de la estructura, de modo que los 2 cuerpos (que hasta ese momento se encontraban unidos) serán separados con una distancia de 50 cm mediante los raíles del despliegue estructural.

Las antenas funcionan convirtiendo las señales eléctricas proporcionadas por el amplificador en ondas electromagnéticas que se propagan en el espacio libre. Para ello, están conectadas al amplificador mediante un puerto SMA, un tipo de conexión de RF comúnmente utilizada por su confiabilidad y rendimiento en altas frecuencias. Este puerto permite un acoplamiento efectivo de la señal amplificada desde el amplificador a la antena, minimizando las pérdidas de transmisión.

Para simplificar la arquitectura del sistema, se ha optado por considerar el mismo modelo de antena tanto para la emisión como para la recepción. Esta decisión tiene varias ventajas, entre ellas, la homogeneidad en las características de las antenas, la simplificación del diseño y la reducción de los posibles problemas de compatibilidad.

Es importante tener en cuenta que las antenas no son puramente isótropas, es decir, no irradian la señal de manera uniforme en todas las direcciones. Por esta razón, presentan una ganancia, lo que significa que la señal se propaga con mayor intensidad en determinadas direcciones. Esta característica es esencial para asegurar una transmisión de energía eficiente y compatible energéticamente hacia la dirección deseada.

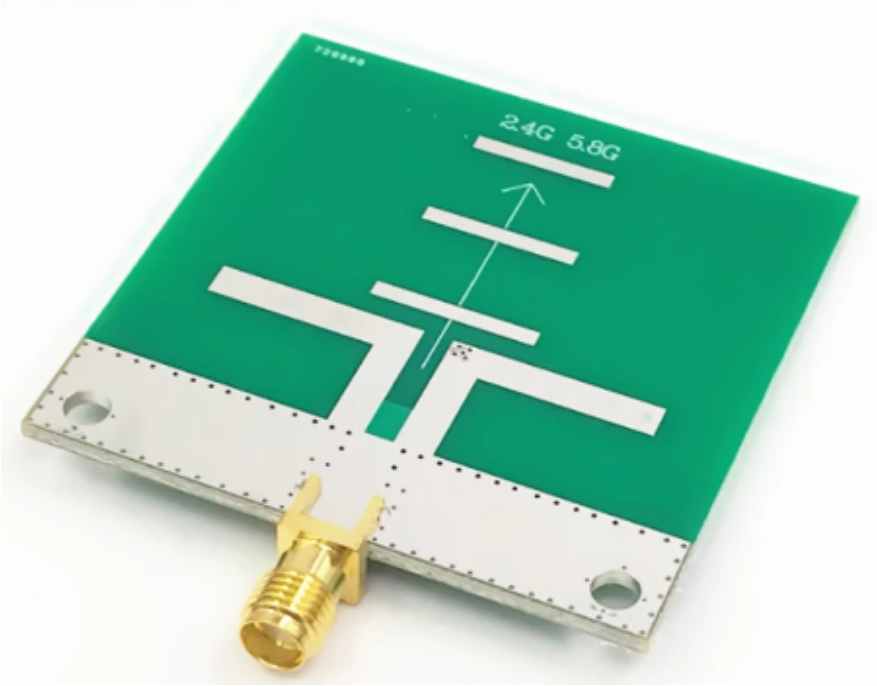

Figura 3.4: Ejemplo de antena de trasmisión/recepción de microondas. Imagen de [8]

En cuanto a la ubicación de las antenas en el CubeSat, se van a situar en paralelo dentro de la estructura. Este diseño permitirá minimizar las pérdidas de señal durante la transmisión y recepción, especialmente cuando la estructura se despliega. Esta disposición garantiza que las antenas se mantengan alineadas, optimizando la comunicación entre ellas y maximizando la eficiencia de la transmisión de energía.

La antena receptora de energía, denominada rectena cuando cuenta con un rectificador, estará conectada a un monitor de corriente. Este dispositivo permitirá medir los parámetros eléctricos de la señal recibida y, de esta manera, evaluar la calidad y eficacia de la transmisión. Este es un aspecto fundamental de la misión, ya que el principal objetivo es demostrar la viabilidad de la transmisión de energía mediante microondas en el espacio.

3.2.3. Definición de Rectificador

La señal recibida por la rectena será procesada por un rectificador. Un rectificador es un dispositivo que convierte la corriente alterna (CA) en corriente continua (CC). En este caso, se utilizará para convertir la señal de microondas recibida por la antena en una señal de corriente continua, que se pueda medir con facilidad por el monitor y que constituirá la señal de control a lo largo de la misión.

El funcionamiento de un rectificador se basa en la propiedad de los diodos de permitir el paso de la corriente en una sola dirección. En un rectificador de media onda, por ejemplo, solo se utiliza una parte del ciclo de la corriente alterna, mientras que en un rectificador de onda completa se utilizan ambas partes. Esto resulta en una señal de corriente pseudo-continua con un nivel de voltaje constante, que es más fácil de manejar y puede ser usada de manera más eficiente en los sistemas electrónicos.

El circuito rectificador que se propone para el sistema de trasmisión de microondas consta de las siguientes subpartes:

- **Red de acoplamiento y filtrado:** Esta red transforma la impedancia para que coincida con la del diodo y bloquea las armónicas de orden superior de la radiación capturada por la antena receptora. Su objetivo es optimizar la transferencia de potencia y suprimir las frecuencias no deseadas.
- **Diodo Rectificador:** Este diodo se encarga de convertir la energía de radiofrecuencia (RF) en corriente continua (DC). Además, produce señales fundamentales y armónicas de RF debido a su no linealidad.
- **Filtro DC de salida:** Este componente, que incluye un condensador, permite pasar el voltaje DC de salida y bloquea las armónicas de orden superior. Junto con la red de acoplamiento y filtrado, se utiliza para confinar la energía de RF.

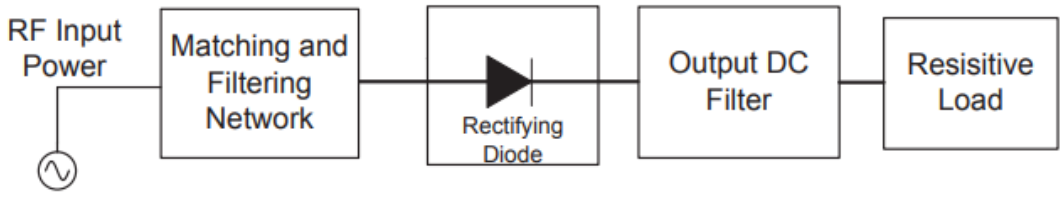

Figura 3.5: Diagrama de bloques del circuito rectificador. Imagen de [9]

En el mercado no existe una gran oferta de modelos de rectificadores de RF para el uso que se busca en esta memoria, donde la eficiencia del proceso de rectificación es crucial al obtener de input una señal cuya potencia ha sido muy atenuada por las pérdidas en la trasmisión

inalámbrica.

Sin embargo, existen algunos modelos experimentales que pueden ser utilizados como referencia para este trabajo a la hora de cuantificar su rendimiento.

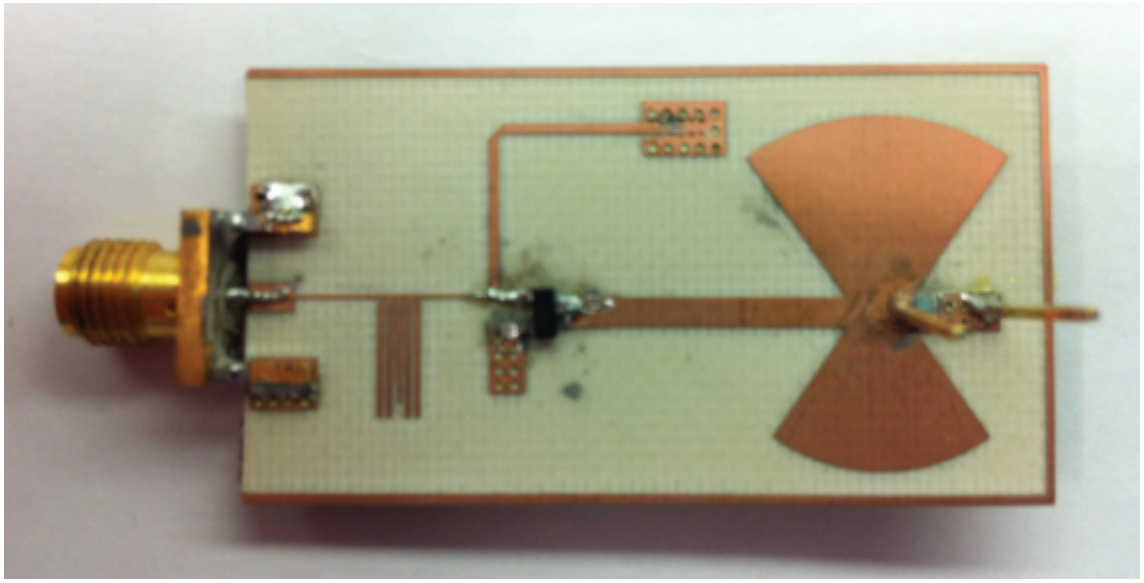

Figura 3.6: Ejemplo de rectificador construido para aplicación de trasmisión de energía por medio de microondas. Imagen de [9]

Dado que, durante la trasmisión de energía mediante microondas, uno de los parámetros a estudiar será la potencia trasmitida en el sistema de antenas, se debe contar con las pérdidas de potencia generadas por el rectificador aguas arriba de la lectura eléctrica que realizará el monitor a la señal de control.

3.3. Elección de Monitor de Corriente

A la hora de estudiar la calidad de la trasmisión de energía, se debe contar con algún elemento de monitorización de dicho proceso. Para ello, la aviónica debe contar con un componente que haga una lectura de la intensidad y voltaje del circuito de lectura, es decir, el circuito aguas abajo de la rectena.

El estudio de los valores obtenidos a lo largo de la misión queda fuera de este proyecto, pues lo que se busca es el diseñar un modelo de aviónica que permita realizar exitosamente la misión del CubeSat. Por tanto, en este apartado se abordará la elección de un monitor de corriente que permita capturar el estado de la trasmisión a lo largo del vuelo.

La matriz de trade-off entre algunas de las opciones más comunes y comercializadas en el mercado (por distintos proveedores) se muestra en la tabla 3.1:

Modelo	Rango de medición de corriente	Resolución	Precisión	Rango de voltaje de alimentación	Interfaz	Tamaño
INA260	± 10A	1.5µA	± 1 %	2.7-5.5V	I2C, SMBus	5mm x 4.4mm
MAX471	± 3A	9.5µA	± 1 %	2.7-7V	Voltaje de salida	5mm x 6.4mm
ACS723	± 5A a ± 20A	40mA a 80mA	± 1.5 %	3-5.5V	Voltaje de salida	21mm x 12.7mm
PZWTN3A0XX	± 3A	10mA	± 1 %	4.5-18V	I2C	4mm x 4mm
LT6106	± 1A	1.5µA	± 2.5 %	2.7-36V	Voltaje de salida	2mm x 2mm

Tabla 3.1: Matriz de trade-off de lectores de potencia

En base a la posibilidad de que la corriente en la lectura del circuito de control sea muy reducida, se busca: un monitor de corriente con la menor resolución posible; una buena precisión; un rango de medición de corriente que incorpore la corriente de trabajo; un rango de voltaje de alimentación que integre el de los pines del OBC; un tamaño reducido; y ,por comodidad, una interfaz I2C, de modo que se pueda integrar junto a otros componentes en un mismo BUS, con distintas direcciones asignadas.

Teniendo en cuenta todos estos campos, sobre todo en base a la resolución, su interfaz y su tamaño, la opción más interesante entre las propuestas es la Ina260.

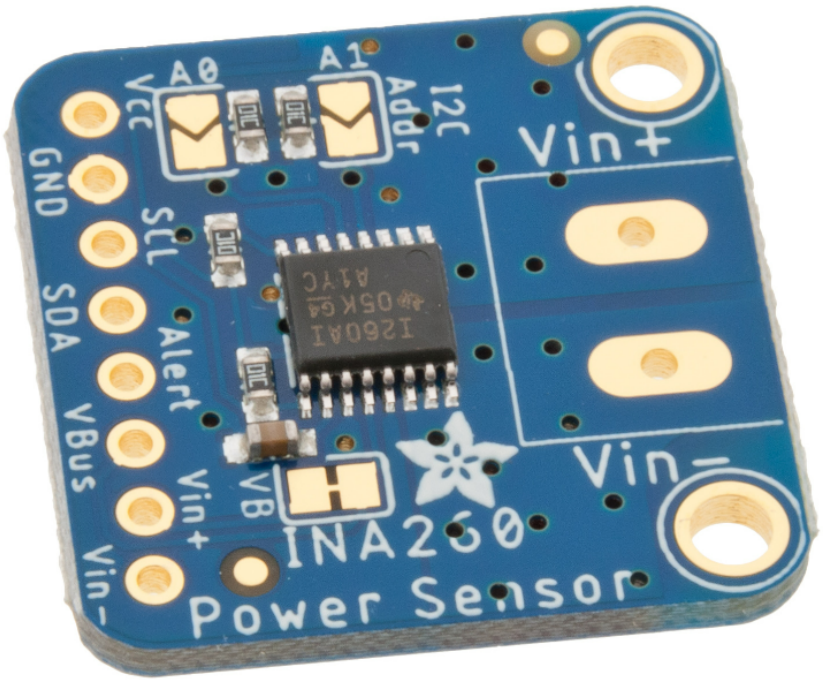

Figura 3.7: Imagen de INA260. Imagen de [10]

Es importante remarcar que el modelo de monitor de corriente que se ha seleccionado para la arquitectura de la aviónica es capaz de trabajar únicamente con una señal en corriente continua, luego el desempeño de la rectificación de su señal de entrada es esencial para el correcto funcionamiento del componente.

3.4. Balance de Potencias del Sistema de Trasmisión de Microondas

En este apartado se va a realizar una estimación del coste energético que supondrá el sistema de trasmisión de microondas a lo largo de la misión. Dicho cálculo del coste energético es crucial para realizar el dimensionado de las baterías que deberá portar el CubeSat.

Por otro lado, es esencial realizar un estudio de la viabilidad de la trasmisión inalámbrica que se pretende plantear en este trabajo en términos de potencia, pues dicha trasmisión representa la misión principal de la payload. La aviónica está encargada de suministrar la potencia y el control al sistema de trasmisión, lo que implica que esta debe estar preparada para suministrar la potencia necesaria para realizar satisfactoriamente la trasmisión durante el vuelo.

Para realizar el dimensionado de las baterías, se debe conocer el balance de potencias del sistema de trasmisión de microondas. Si bien este sistema queda fuera del ámbito de actuación de la aviónica y, por tanto, de este trabajo, sí se debe realizar una estimación del balance de potencias para asegurar que la aviónica estará preparada para cumplir con los requisitos mínimos que demande la payload.

En este caso, la trasmisión busca probar el transporte de energía por medio de una lectura de potencia aguas abajo del sistema de trasmisión, de modo que, en este apartado, se buscará estudiar la viabilidad del suministro de potencia suficiente para que, una vez realizada la trasmisión, la potencia leída por el monitor sea superior a la sensibilidad de lectura del propio componente.

De este modo, para el cálculo del balance de potencias se partirá del dato de potencia emitida por el generador de señal, al que se le aplicarán todas las ganancias y pérdidas a lo largo del sistema de trasmisión hasta llegar a la potencia suministrada al lector de potencia. En el caso de que la potencia suministrada aguas abajo del sistema de trasmisión sea superior a la sensibilidad del lector, la trasmisión de energía se dará por viable y se podrán extraer los datos de potencia requeridos por los componentes para realizar el dimensionado (aproximado) del coste energético.

Se parte de una sensibilidad de 1,5 mW (o 1,76 dBm) por parte del monitor de corriente. Por el lado del chip emisor de la señal de microondas, supondremos (en consonancia con modelos que se encuentran en el mercado, como el mostrado en la bibliografía) que genera una señal de 2,4 GHz a 5 dBm.

Dicha señal pasará a ser amplificada por el amplificador, el cual, de nuevo tomando como ejemplo y referencia un modelo de mercado como el mostrado en la bibliografía, aportará una ganancia estimada de 35 dB.

Con respecto a la antena de emisión y recepción, un valor razonable de ganancia como punto de partida para la estimación del balance de potencias es el de 2 dB, el cual es un factor relativamente conservador con respecto a las ganancias de las antenas que se encuentran en el mercado.

Para modelizar las pérdidas que se darán en la comunicación, se acudirá al modelo de

propagación en espacio libre, planteado mediante la ecuación de Friis. Dicho modelo se ha elegido siguiendo la Recomendación UIT-R P.525-4. De este modo, el modelo propone la ecución 3.1:

$$L_{bf}(dB) = 20 \cdot \log_{10}(d) + 20 \cdot \log_{10}(f) + 20 \cdot \log_{10}\left(\frac{4\pi}{c}\right) \tag{3.1}$$

Donde L_{bf} es la pérdida de trayectoria en dB; d es la distancia entre las dos antenas en metros; f es la frecuencia de la señal en Hertz; c es la velocidad de la luz en metros por segundo (aproximadamente $3 \cdot 10^8$); y log_{10} representa el logaritmo base 10.

Sustituyendo con los valores conocidos para la trasmisión que se está planteando, se obtiene el resultado mostrado en la ecución 3.2:

$$L_{bf} = 20 \cdot \log_{10}(d) + 20 \cdot \log_{10}(f) + 20 \cdot \log_{10}\left(\frac{4\pi}{c}\right) \rightarrow 34,03\,dB \rightarrow \boxed{L_{bf} \approx 34\,dB} \tag{3.2}$$

Donde d es 0,5m; f es 2,4 Ghz; y c es $3 \cdot 10^8$ m/s

Dado que, en el momento de la trasmisión, la atmósfera terrestre presentará una densidad muy reducida, se supondrán como despreciables las pérdidas atmosféricas, las cuales son causadas por el vapor de agua, oxígeno y otros precipitados y gases que se puedan encontrar en suspensión en el aire.

Por otro lado, la eficiencia del rectificador definirá las pérdidas de potencia en dicho componente. Dicha eficiencia se obtendrá a partir de un modelo de rectificador de microondas diseñado para una trasmisión de energía, el cual se tomará como referencia. En dicho modelo, las curvas que caracterizan el desempeño del componente se muestran en la figura 3.8.

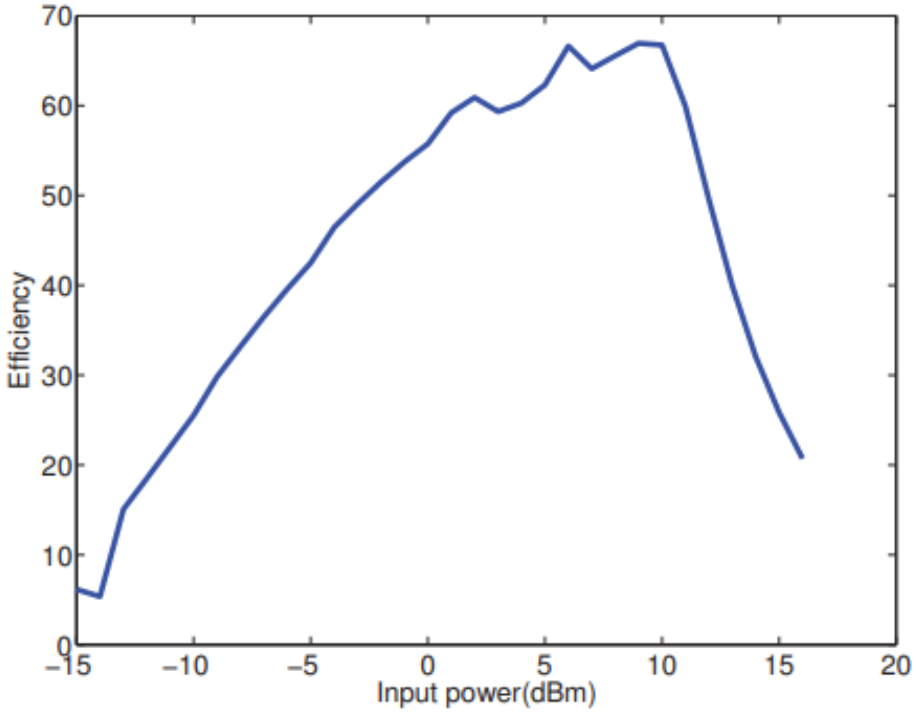

Figura 3.8: Eficiencia de conversión medida frente a la potencia de entrada por el rectificador a 2,45 GHz. Imagen de [9]

Para conocer el dato de potencia de entrada al rectificador, se deben aplicar las pérdidas y ganancias previas a dicha señal, partiendo de la señal de entrada emitida por el generador de señal.

La potencia en dBm se refiere a la potencia relativa en milivatios (mW) en una escala logarítmica, y se calcula usando la relación mostrada en la ecuación 3.3.

$$P(dBm) = 10 \cdot \log\left(\frac{P(mW)}{1mW}\right) \tag{3.3}$$

Donde P(dBm) es la potencia en milivatios.

Cuando se tienen ganancias o pérdidas en decibelios (dB), se pueden sumar o restar directamente a la potencia en dBm para obtener la potencia final. Si se tienen múltiples ganancias o pérdidas, simplemente se suman o restan en serie. Si se desea convertir de nuevo la potencia en dBm a mW, se puede utilizar la relación inversa, mostrada en la ecuación 3.4.

$$P(mW) = 10^{\frac{P(dBm)}{10}} \tag{3.4}$$

De este modo, el dato de potencia de entrada al rectificador se obtiene en la ecuación 3.5.

$$\begin{aligned} P_{\text{rectificador,in}} &= P_{\text{generador}} + G_{\text{amplificador}} + G_{\text{Tx}} - L_{\text{bf}} + G_{\text{Rx}} \\ &\rightarrow \boxed{P_{\text{rectificador,in}} = 9,97\,\text{dBm} \approx 10\,\text{dBm} = 10\,\text{mW}} \end{aligned} \tag{3.5}$$

Donde $P_{\text{generador}}$ es 5 dBm; $G_{\text{amplificador}}$ es 35 dB; G_{Tx} es 2 dB; L_{bf} es 34,03 dB; y G_{Rx} es 2 dB.

En la figura 3.8 se muestra la eficiencia de conversión medida frente a la potencia de entrada por el rectificador a 2,45 GHz. Dicha gráfica se utilizará como referencia para obtener las pérdidas de potencia en la señal de control tras ser rectificada. A partir del dato de potencia de entrada al rectificador obtenido, la eficiencia en dicho componente será del 67 %.

Las pérdidas en dB a las que se somete la señal en dicho componente se pueden obtener a partir del dato de la eficiencia utilizando la relación entre la potencia de salida y la potencia de entrada. En este caso, la potencia de salida es el 67 % de la potencia de entrada, es decir, 0.67 veces la potencia de entrada.

$$L_{rectificador}(dB) = 10 \cdot \log\left(\frac{P_{in}}{P_{out}}\right) = 10 \cdot \log(0,67) \rightarrow \boxed{L_{rectificador} = -1,74\,dB} \tag{3.6}$$

Una vez conocidos todos los datos de ganancias y pérdidas a lo largo de todas las etapas del sistema de trasmisión de microondas, se puede realizar el cálculo de la potencia recibida por el lector de potencia en base al modelo calculado en este apartado. El resultado obtenido se muestra en la ecuación 3.7.

$$\begin{aligned} P_{\text{lector,in}} &= P_{\text{generador}} + G_{\text{amplificador}} + G_{\text{Tx}} - L_{\text{bf}} + G_{\text{Rx}} - L_{\text{rectificador}} \\ &\rightarrow \boxed{P_{\text{lector,in}} = 8,26\,\text{dBm} = 6,7\,\text{mW}} \end{aligned} \tag{3.7}$$

Donde $P_{\text{generador}}$ es 5 dBm; $G_{\text{amplificador}}$ es 35 dB; G_{Tx} es 2 dB; L_{bf} es 34,03 dB; G_{Rx} es 2 dB; y $L_{\text{rectificador}}$ es 1,74 dB.

El resultado obtenido es una potencia de 6,7 mW a la entrada del lector de potencia. Dado que el estudio de viabilidad requería que la potencia de entrada en el lector fuese superior a 1,5 mW, queda demostrada la viabilidad de la trasmisión inalámbrica en términos de potencia.

En la figura 3.9, se muestra la evolución de la potencia eléctrica a lo largo del sistema de transmisión de energía por medio de microondas, aguas arriba del lector de potencia. Las barras horizontales muestran el valor de potencia de la señal de salida en cada etapa del sistema de trasmisión, así como las barras verticales muestran las variaciones ocasionadas por cada etapa (ganancias en caso de que se aumente la potencia, y pérdidas en caso de que se disminuya). Por último, se muestra en rojo el valor de la sensibilidad del lector de potencia.

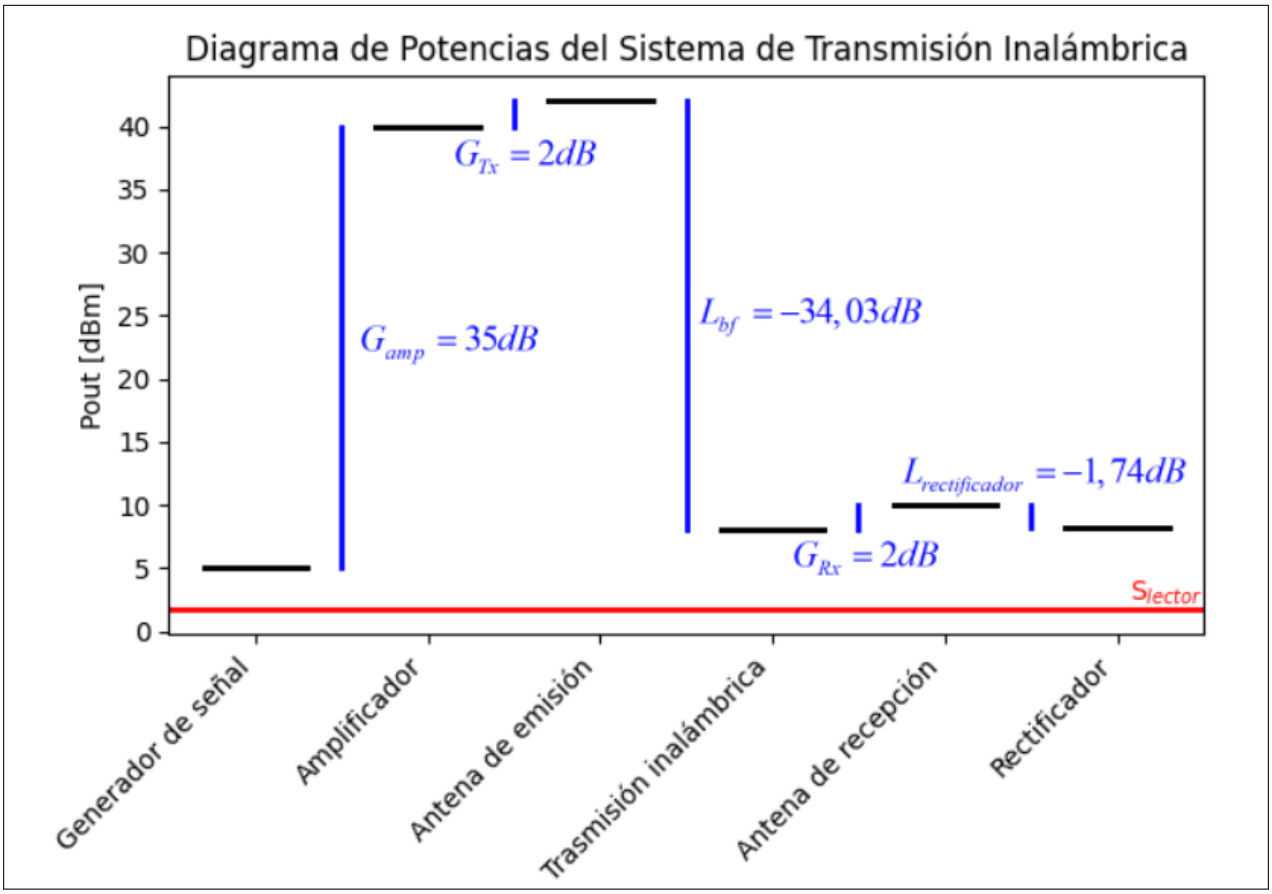

Figura 3.9: Diagrama de Potencias del Sistema de Transmisión Inalámbrica (Elaboración propia)

Como se analizó anteriormente, la potencia de entrada en el lector de potencia es superior a la sensibilidad del propio componente, confirmando así la viabilidad a nivel energético de la trasmisión de potencia, basándose en los modelos utilizados en esta memoria.

Una vez conocida la viabilidad de la misión principal del satélite, el cálculo del coste energético se realizará a partir del consumo energético de la suma de todos los componentes a lo largo de la misión.

3.5. Elección de OBC

Como ya se explicó en el apartado 3.1, la configuración de ordenador central es una configuración ampliamente extendida entre los CubeSats. Presenta ciertas ventajas respecto a una distribución modular, como puede ser un grado menor de complejidad al concentrar todos los trabajos de procesamiento de datos y comunicación con el resto de componentes. Por contraparte, en una distribución modular el fallo de un ordenador de a bordo puede ser menos crítico, además de que en caso de escalar el sistema en futuras iteraciones permite la actualización de ciertos módulos sin la necesidad de cambiar todo el conjunto.

El ordenador central será el encargado de ejecutar el código preparado para la misión, por lo que deberá ser capaz de cumplir con los requisitos: CSM-SW-REQ-1, CSM-SW-REQ-2 y CSM-SW-REQ-3. Dichos requisitos especifican como debe funcionar el código ejecutado por el ordenador, pero no son especialmente restrictivos en cuanto a la elección de un modelo de ordenador del mercado, pues prácticamente cualquier modelo puede cumplir con ellos.

Para este diseño del subsistema de aviónica, se va a optar por una distribución central de OBC. De este modo, se deberá elegir un ordenador entre los que se encuentran en el mercado para realizar las funciones de "cerebro de la misión".

Existen muchos criterios a comparar en la elección de un OBC entre las opciones de mercado, como pueden ser:

- **Coste:** Si bien en la mayoría de misiones es un campo crítico, en este caso será un parámetro de menor importancia (siempre que las distintas opciones barajadas se encuentren en un marco de precios que presenten un orden de magnitud similar).
- **Consumo de Potencia:** La potencia eléctrica es un recurso limitado en el diseño de CubeSat que se está realizando. En el dimensionado de las baterías se tendrán en cuenta el consumo de todos los componentes de la aviónica, no obstante, siempre se buscará reducir en cierta medida el consumo de estos componentes, pues permiten alargar el tiempo de vida de la misión.

- **Tamaño:** El OBC se encontrará en el cuerpo mayor del CubeSat, es decir, en el cuerpo cuyo volumen será de 2 U. Dicho volumen es bastante reducido, luego se buscará que el OBC sea comedido en cuanto a dimensiones.

- **Potencia de procesamiento:** En general, una mayor potencia de procesamiento permite ejecutar códigos más exigentes a nivel computacional. Este parámetro se buscará maximizar entre las opciones de mercado que se presenten, pues de este modo se dispone de más libertad a la hora de desarrollar el código de trabajar con grandes cantidades de datos.

- **Memoria Flash y memoria RAM:** Ambas relacionadas con la capacidad de trabajo y rendimiento a la hora de ejecutar el código, luego se buscará maximizarlas, al igual que ocurría con la potencia de procesamiento.

Conocidos los parámetros a estudiar para la comparación de los principales y más extendidos OBC que se encuentren actualmente en el mercado, se presenta la matriz de trade-off 3.2.

OBC	Coste (€)	Consumo de Energía (W)	Tamaño (mm)	Potencia de Procesa-miento (GHz)	Memoria Flash (GB)	Memoria RAM (GB)
Raspberry Pi 4 (2 GB)	60	5	88 x 58 x 19	1.5	32	2
Raspberry Pi 4 (4 GB)	75	5	88 x 58 x 19	1.5	32	4
BeagleBone Black (1 GB)	55	6	102 x 63 x 22	1	8	1
BeagleBone Black (4 GB)	65	6	102 x 63 x 22	1	8	4
NVIDIA Jetson Nano (4 GB)	129	10	70 x 45 x 21	1.43	32	4
NVIDIA Jetson Nano (8 GB)	139	10	70 x 45 x 21	1.43	32	8

Tabla 3.2: Matriz trade-off de OBC

Analizando los resultados obtenidos a partir de la matriz de trade-off 3.2 se pueden alcanzar varias conclusiones. En cuanto a tamaño, los modelos de Raspberry y NVIDIA son los más interesantes, así como en cuanto a potencia de procesamiento. Por el lado de la memoria Flash y la memoria RAM, las opciones de Raspberry y NVIDIA son suficientes en todos los casos, no obstante, el modelo de Raspberry Pi 4 de 2 GB de memoria RAM es más limitada. Finalmente,

por el lado del coste y del consumo, la opción de Raspberry de 4 GB de RAM supera con creces a los modelos de NVIDIA.

Por estas razones, el modelo seleccionado como OBC del CubeSat a tratar en este proyecto será la Raspberry Pi 4 (4 GB).

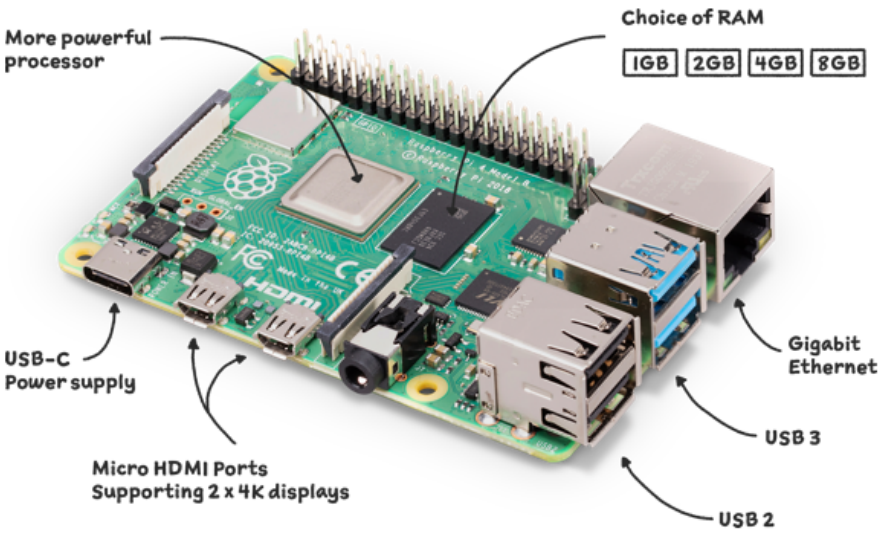

Figura 3.10: Imagen de Raspberry Pi 4. Imagen de [11]

Como acompañante al OBC, se elegirá un sistema de almacenamiento adecuado para el almacenamiento de los datos recogidos durante la misión. La Raspberry Pi 4, que ha sido seleccionada como ordenador de a bordo para el cubesat, utiliza un sistema de almacenamiento en tarjeta SD para almacenar tanto el sistema operativo como los datos generados durante la operación del cubesat. Una tarjeta SD funciona como un dispositivo de almacenamiento no volátil que mantiene los datos sin la necesidad de una fuente de alimentación constante.

Dentro de la amplia gama de tarjetas SD disponibles en el mercado, se ha optado por seleccionar una tarjeta SD de la propia marca Raspberry Pi, con el fin de garantizar la máxima compatibilidad con la Raspberry Pi 4. Las tarjetas SD de Raspberry Pi están diseñadas y probadas para funcionar de manera óptima con sus propios productos, lo que reduce la posibilidad de problemas de compatibilidad que podrían surgir al usar tarjetas SD de otros fabricantes.

En lo que respecta al espacio de almacenamiento, se ha determinado que una tarjeta SD de 64 GB sería suficiente para cubrir las necesidades del cubesat. Esta capacidad es suficiente para almacenar el sistema operativo y los datos generados durante la misión.

Figura 3.11: Imagen de targeta de memoria seleccionada. Imagen de [12]

Por lo tanto, la elección de una tarjeta SD de la marca Raspberry Pi de 64 GB para el almacenamiento de datos en el cubesat asegura una compatibilidad óptima con la Raspberry Pi 4, proporciona una capacidad de almacenamiento suficiente para la misión, y ofrece una solución coste-efectiva para las necesidades de almacenamiento del cubesat.

3.6. Elección de Sensor Óptico

El sensor óptico es un componente que trabaja emitiendo un haz de luz, el cual convierte en una señal electrónica. En el caso del sensor óptico que se busca para la aplicación de despliegue estructural, dicho sensor debe emitir un haz de luz que, al rebotar en una superficie que refleje dicho haz y lo envíe de vuelta al sensor, sea capaz de medir la distancia a la que se encuentra el objeto en el que ha rebotado. De este modo, cuando el haz viaje en el aire lo hará a velocidad constante, luego el tiempo que tarde el haz en ser devuelto al sensor será

directamente proporcional a la distancia entre cuerpo y receptor.

Actualmente en el mercado, existen diversas soluciones que, mediante esta tecnología, son capaces de medir distancias entre objetos. La aplicación que se busca en este proyecto es la de enfocar el sensor entre los 2 cuerpos del CubeSat, de modo que cuando se despliegue la estructura, la distancia que mide el sensor que separa los 2 cuerpos aumentará. Dicha lectura se utilizará como confirmación de que el despliegue se ha efectuado correctamente. Cabe recalcar que, como se explicará más adelante en el desarrollo del software embebido en la aviónica, la lectura de la distancia entre los cuerpos tendrá un carácter meramente informativo en la misión, de modo que el código no se verá condicionado en ningún momento por la lectura de este sensor. En otras palabras, la medida del sensor está enfocada al análisis de la misión y de la "performance" de los sistemas del CubeSat a posteriori de la propia misión.

El sensor seleccionado debe cumplir con el requisito CSM-SNS-REQ-1, luego debe ser capaz de realizar una lectura de por lo menos 50 cm.

Con estos conceptos claros, la elección entre los principales sensores del mercado se llevará a cabo mediante la matriz de trade-off 3.3.

Modelo de sensor	Rango mínimo (m)	Rango máximo (m)	Resolución (mm)	Tasa de actualización (Hz)	Voltaje de entrada (V)	Consumo de corriente (mA)	Interfaz	Dimensiones (mm)
VL53L0X	0.05	2	N/A	5000	2.6 - 5.5	10	I2C	25 x 12.2 x 2
TFMini Plus	0	12	5	1 - 1000	5	500	UART, I2C	35 x 18.5 x 21
TF03	0.5	180	10	1 - 1000	5	180	UART	44 x 43 x 32
TFMini	0.3	12	5	100	5	24.0	I2C	42 x 15 x 16
LIDAR-Lite v3HP	0.3	40	10	1000	4.75 - 5.5	85.0	I2C	24.5 x 53.5 x 33.5

Tabla 3.3: Matriz de trade-off de sensor óptico

Tras analizar la tabla de comparación de las especificaciones técnicas, se puede concluir que el sensor VL53L0X es el modelo que mejor se adapta a las necesidades del proyecto.

En primer lugar, el rango de medición de distancia del sensor VL53L0X, que oscila entre

0.05 y 2 metros, es suficientemente amplio para cubrir las necesidades de la aplicación, teniendo en cuenta que la separación entre los dos cuerpos del CubeSat no excederá estos límites.

Además, el VL53L0X destaca por su alta tasa de actualización, hasta 5000 Hz, lo que significa que es capaz de proporcionar medidas de distancia de manera muy rápida y continua. Esta capacidad es especialmente importante para el análisis de la misión y del rendimiento de los sistemas del CubeSat después de la misión, como se ha mencionado anteriormente.

En lo que respecta al voltaje de entrada y al consumo de corriente, el VL53L0X también resulta ser la opción más adecuada. Con un voltaje de entrada que varía entre 2.6 y 5.5 V, es compatible con la alimentación eléctrica disponible en el CubeSat, además de ser la opción más versátil en cuanto a la elección de un voltaje de alimentación (dentro de las opciones analizadas). Además, su bajo consumo de corriente, de solo 10 mA, lo convierte en la elección más eficiente desde el punto de vista energético, un parámetro crítico teniendo en cuenta las limitaciones de capacidad con las que contarán las baterías del satélite.

En términos de interfaces de comunicación, el sensor VL53L0X utiliza el protocolo I2C, que es ampliamente compatible con muchos microcontroladores y sistemas embebidos, incluyendo la Raspberry Pi 4 elegida para este proyecto.

Por último, las dimensiones compactas del VL53L0X, de solo 25 x 12.2 x 2 mm, son un factor crucial dada la limitación de espacio en el CubeSat, siendo la opción más comedida en cuanto a tamaño entre las analizadas.

Por todo ello, se concluye que el sensor VL53L0X es la opción más adecuada para este proyecto, gracias a su rango de medición, alta tasa de actualización, bajo consumo de energía, compatibilidad de interfaz y tamaño compacto. Además, a la comparativa de especificaciones técnicas del componente que se ha realizado, se le añade el amplio uso que ha tenido este componente en proyectos similares y de robótica, lo que ha permitido que la comunidad haya volcado el gran conocimiento adquirido a cerca de su comportamiento y de su programación en los códigos que lo dirigirán.

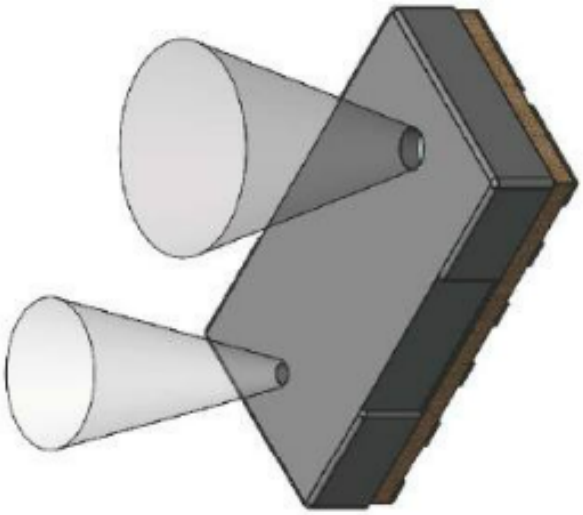

Figura 3.12: Imagen del componente de medición. Imagen de [13]

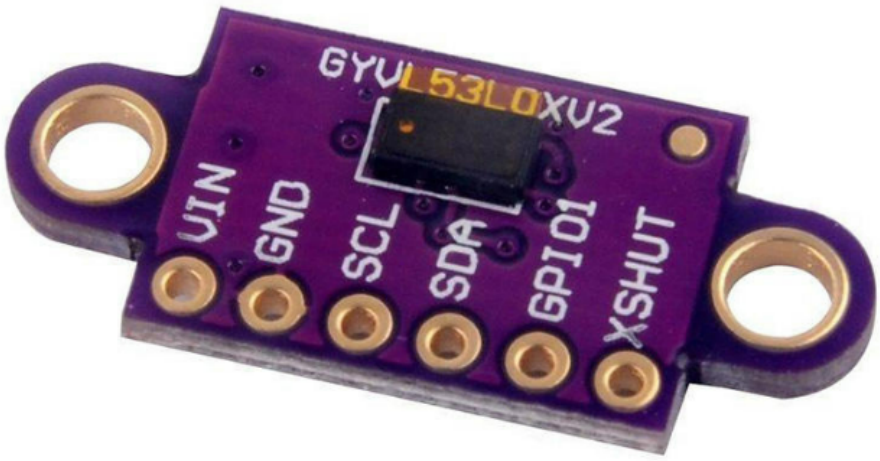

Figura 3.13: Imagen del componente de medición integrado en la placa de conexiones. Imagen de [14]

3.7. Elección de Altímetro Barométrico

Una de las misiones secundarias del satélite es la de caracterizar el ascenso durante la misión. Dicho ascenso se verá puramente determinado por el desempeño del vehículo lanzador, en este caso, el globo aerostático. De este modo, el conocimiento a tiempo real de la altitud de vuelo no influirá de ningún modo a la misión principal de transmisión de ondas, pero se utilizará para el análisis posterior de la misión.

Por esta razón, se especifica como requisito CSM-ALT-REQ-1 el incluir algún componente en la arquitectura de la aviónica capaz de realizar el muestreo de la altitud de vuelo a lo largo de la misión. Además, asociando las distintas altitudes a los momentos de lectura durante toda la misión, se podrá reconstruir un estudio pseudo-continuo de la velocidad ascensional del vehículo, como exige el requisito CSM-ALT-REQ-2. El requisito CSM-VEH-REQ-2 establece la necesidad de que el componente de medición sea capaz de trabajar a una altitud superior a 33 km sobre el nivel del mar, de modo que este pueda resultar operativo a lo largo de toda la misión.

Cabe recalcar que, a partir de un componente de lectura de altura, no será posible reconstruir la trayectoria seguida por el vehículo lanzador a lo largo de la misión, pues solo se estudiará la mecánica de dicho vehículo en una dimensión, la de la altitud de vuelo.

Una vez definida la necesidad de realizar el muestreo de la altura del satélite a lo largo de la misión, el siguiente paso es buscar el componente idóneo para realizar dicha tarea. Se considerará la altura máxima que puede alcanzar el vehículo en la misión, pues el componente más sencillo de implementar (capaz de responder al requisito a tratar) es un altímetro barométrico, cuya principal limitación es que cuenta con una altura máxima a la que es capaz de operar. Por esta razón, se busca una solución de mercado cuya altura máxima sea superior a la que puede llegar a alcanzar el vehículo.

Dado que hay varios modelos de altímetros barométricos disponibles en el mercado, se realizará una matriz de trade-off que permita comparar las principales características y especificaciones de diferentes sensores. Los criterios a considerar en esta matriz incluyen el rango de medición de presión, la frecuencia de muestreo, la precisión, el consumo de energía, el tamaño, el peso, la interfaz de conexión y el voltaje de entrada.

Antes de comparar los distintos sensores, es necesario entender cómo calcular la altitud

máxima a partir del rango de presiones del sensor. Dicha estimación no es trivial, pues la caracterización de la atmósfera es el objeto de estudio de diversas ramas de la ciencia, donde la obtención de una relación directa de la altura con la presión atmosférica muchas veces no es posible. Con el fin de entender, en rasgos generales, la morfología de la atmósfera, así como obtener soluciones con las que trabajar, se han desarrollado diversos modelos de atmósfera. Dichos modelos son una representación simplificada de las propiedades físicas de la atmósfera terrestre, como la temperatura, la presión y la densidad del aire, que se utilizan de forma extendida en cálculos de ingeniería aeronáutica y aeroespacial.

El modelo de atmósfera ISA (Atmósfera Estándar Internacional, por sus siglas en inglés) es un modelo atmosférico que establece condiciones promedio de presión, temperatura y densidad del aire a diferentes altitudes. Este modelo es utilizado para realizar cálculos y simulaciones.

El modelo ISA asume una atmósfera que es perfectamente gaseosa y en equilibrio termodinámico. Además, considera que la atmósfera es afectada solo por la gravedad (ignorando otros efectos como el viento o la humedad). También asume una temperatura de superficie de 15 grados Celsius, que disminuye a medida que aumenta la altitud hasta los 11 km (la tropopausa), donde la temperatura se mantiene constante hasta los 20 km.

El modelo ISA divide la atmósfera en varias capas. La relación entre la presión y la altitud en la atmósfera se puede describir mediante la ecuación barométrica, que se deriva de la ecuación de estado de los gases ideales y la ecuación de equilibrio hidrostático. La primera capa que modeliza la ISA es la troposfera, la cual abarca el rango de los 0 a 11 km de altura respecto el nivel del mar y en la cual la temperatura se modeliza con una variación lineal con respecto la altitud.

En la estratosfera, la cual abarca de los 11 km a los 20 km, la temperatura se modeliza constante. Le sigue la estratosfera superior, la cual abarca de los 20 km a los 32 km, donde la temperatura vuelve a variar linealmente con la altitud.

Finalmente se encuentra la mesosfera, la cual abarca de los 32 km a los 47km. En dicho entorno se sitúa la altitud máxima de funcionamiento de los componentes de lectura de presión más desarrollados del mercado (dentro de la gama de componentes cuyas dimensiones los permiten implentarse en una electrónica comedida en cuanto a tamaño). La ecuación 3.8 presenta la ecuación barométrica de la mesosfera en base al modelo ISA.

$$P = P_{32} \cdot \left(1 + \frac{L'' \cdot (h - h_{32})}{R \cdot T_{32}}\right)^{\frac{g \cdot M}{R \cdot L''}} \tag{3.8}$$

Donde P es la presión a una altitud h; h es la altitud en metros; L" es el gradiente de temperatura en esta capa; P_{32} es la presión a 32 km; h_{32} es la altitud de inicio de esta capa (32 km); T_{32} es la temperatura a 32 km; g es la aceleración debido a la gravedad (9.80665 m/s^2); M es la masa molar del aire seco (0.0289644 kg/mol); y R es la constante universal de los gases (8.3144598 J/mol·K).

La mesosfera es la última capa en la que se suelen tomar (de manera generalizada) los resultados obtenidos mediante el modelo ISA. En la termosfera y exosfera, se parte de modelos más complejos, y en aplicaciones estándar de ingeniería habitualmente se desprecia la atmósfera como simplificación.

Una vez conocido el método con el que obtener la altura máxima de operación a partir de un altímetro barométrico, se puede realizar la comparación de los modelos de altímetro. A partir del rango de medición y la ecuación barométrica, se obtendrá la altitud máxima de funcionamiento del componente, la cual deberá superar la altura máxima de vuelo modelizada por el vehículo lanzador.

En la tabla 3.4 se presenta la matriz de trade-off que compara los parámetros obtenidos para cada sensor.

Sensor	Rango de Medición	Frecuencia de Muestreo	Precisión (presión absoluta)	Consumo de Energía	Tamaño	Interfaz de Conexión	Voltaje de Entrada
Adafruit BMP388	30000 a 125000 Pa	200 Hz	50 Pa	0.65 mA	10.2 mm x 10.2 mm x 1 mm	I2C, SPI	1.65V - 5V
MPL3115A2	50000 a 110000 Pa	1 Hz	400 Pa	2 mA	3 mm x 5 mm x 1 mm	I2C	1.95V - 3.6V
DPS310	30000 a 120000 Pa	128 Hz	100 Pa	38 µA	2.0 mm x 2.5 mm x 1.0 mm	I2C, SPI	1.2V - 3.6V

Tabla 3.4: Matriz trade-off de altímetros

Con estos datos en mano, se pueden comparar las características y especificaciones de los distintos sensores barométricos en la matriz de trade-off. Al analizar las métricas clave, como el rango de medición de presión, la frecuencia de muestreo, la precisión, el consumo de energía,

el tamaño, el peso y las condiciones de operación, se puede determinar cuál es el sensor más adecuado para el proyecto.

En cuanto a la altitud máxima, el Adafruit BMP388 y DPS310 destacan como sensores, con una altitud máxima de aproximadamente 39,4 km (en base a la presión mínima de medición, y mediante el uso del modelo ISA de atmósfera a partir de la ecuación 3.8). Este es un parámetro muy importante, pues brinda la capacidad de realizar mediciones en un rango de altitudes más amplio.

En términos de frecuencia de muestreo, el BMP388 tiene el mayor valor, con 200 Hz. Una mayor frecuencia de muestreo permite obtener datos de altitud con mayor rapidez, lo que permitirá almacenar más datos de este tipo.

La precisión es otro factor crítico, en el cual vuelve a destacar el BMP388, el cual proporciona las mediciones de altitud más precisas.

En cuanto al consumo de energía, el BMP388 es el menos eficiente. Sin embargo, el consumo eléctrico de este componente sigue siendo muy reducido, por lo que, aun siendo el modelo con mayor consumo, sigue siendo competente en este ámbito.

Respecto al tamaño, el BMP388 tiene unas dimensiones de 10.2 mm x 10.2 mm x 1 mm mm. Aunque no es el sensor más pequeño de la lista, sus dimensiones son suficientes para adaptarse a la estructura del CubeSat de 3U sin problemas.

En relación a la interfaz de conexión, los 3 componentes muestran unas interfaces compaginables con el ordenador de a bordo seleccionado, al igual que con el caso de los voltajes de entrada.

En resumen, los 3 componentes comparados en este apartado son perfectamente compatibles con la misión que se plantea en este proyecto. La elección del Adafruit BMP388 como altímetro barométrico para la arquitectura de la aviónica se basará en su capacidad para proporcionar mediciones de mayor precisión, en un mayor rango de presiones, y a una mayor frecuencia de muestreo que el resto de opciones.

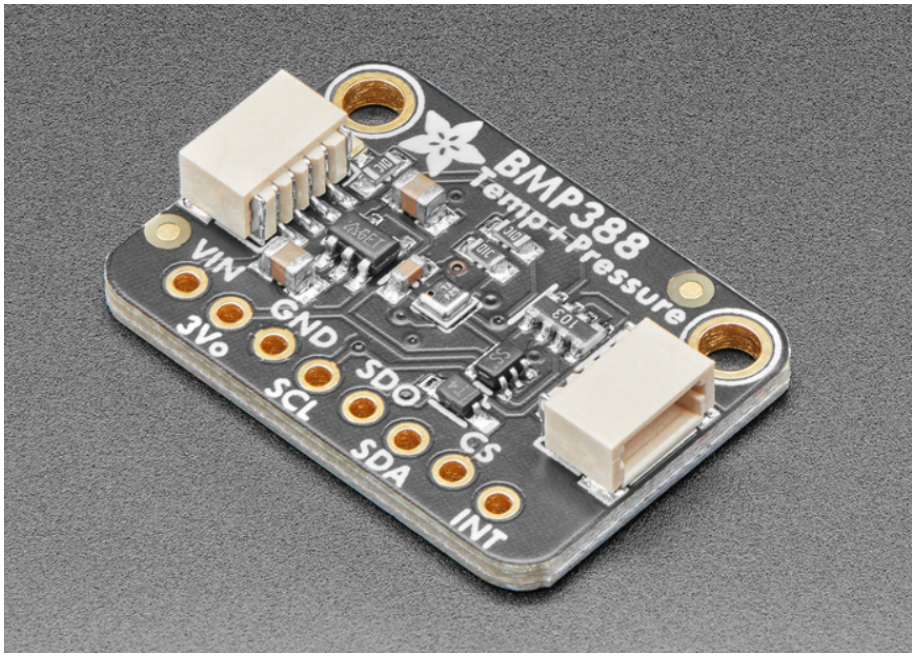

Figura 3.14: Imagen del Adafruit BMP388. Imagen de [15]

3.8. Elección de Cámara

En este apartado, se abordará la selección de una cámara para la aviónica del satélite. En conjunto, la selección de la cámara Raspberry Pi para la aviónica del CubeSat se llevará a cabo considerando una serie de factores clave, como la compatibilidad con el ordenador central, el control y la adquisición de imágenes y vídeos, el almacenamiento de datos y las restricciones de comunicación. Esta elección garantiza que el sistema de aviónica pueda cumplir con los requisitos de la misión y proporcionar información visual valiosa sobre la misión y el despliegue, al tiempo que mantiene una arquitectura sencilla y eficiente.

Para ello, se han tenido en cuenta los requisitos del subsistema de aviónica (CSM-CAM-REQ-1 y CSM-CAM-REQ-2), los cuales establecen que se deben tomar imágenes cada 20 minutos y un vídeo en el momento en que se active la separación de la arquitectura. Además, se debe considerar la ubicación de la cámara en la parte baja de la estructura del satélite principal, de modo que pueda grabar el despliegue apuntando hacia él.

Respecto a dicha posición de la cámara, el resultado esperado será el de imágenes de las

cuales se pueda extraer limitada información previo al despliegue de la estructura. Tras este evento, la distancia a la que se separarán los 2 cuerpos que conforman la estructura del CubeSat será suficiente como para que, tanto en las imágenes como en el vídeo, se distinga claramente el cuerpo de 1 U durante el vuelo.

Dado que se ha seleccionado una Raspberry Pi 4 como ordenador central para la aviónica, resulta lógico elegir una cámara desarrollada por el mismo fabricante en específico para dicho modelo, pues la interfaz se simplificará en gran medida y se facilitará la implementación. La cámara Raspberry Pi PiNoir Camera V2 es una opción adecuada, ya que ha sido diseñada específicamente para funcionar con la Raspberry Pi y puede proporcionar las funcionalidades necesarias para cumplir con los requisitos de la misión.

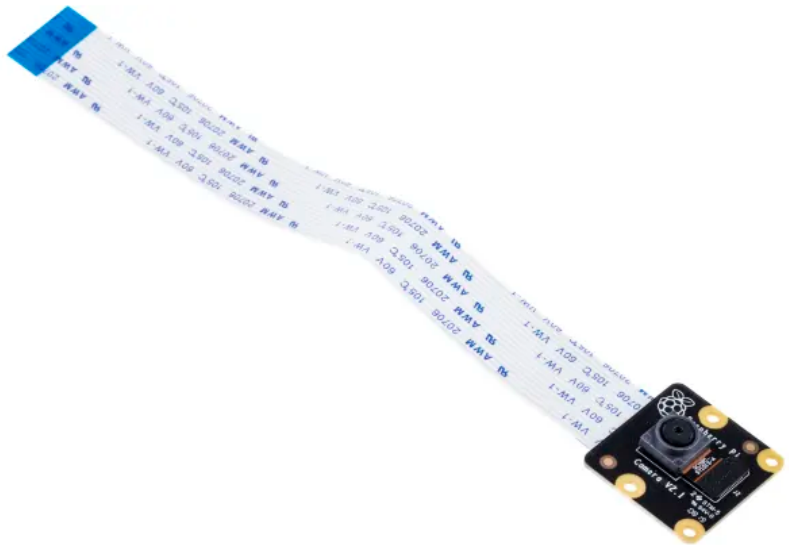

Figura 3.15: Módulo de cámara elegido. Imagen de [16]

La Raspberry Pi PiNoir Camera cuenta con un sensor de imagen Sony IMX477 de 12.3 megapíxeles y capacidad para capturar vídeo a 1080p a 30 fps. Además, es compatible con lentes intercambiables, lo que permite ajustar el campo de visión y la distancia focal según las necesidades de la misión.

La conexión entre la cámara Raspberry Pi PiNoirCamera y la Raspberry Pi 4 se realiza mediante un cable plano flexible (FFC) que se conecta al puerto CSI (Camera Serial Interface) en la Raspberry Pi. Este puerto es un conector especializado que permite la comunicación entre la cámara y el ordenador central utilizando el protocolo MIPI (Mobile Industry Processor Interface). Dicha interfaz es una especificación de interfaz de alta velocidad y bajo consumo de

energía diseñada para conectar componentes como cámaras y pantallas a sistemas embebidos y móviles.

Una vez que la cámara esté conectada a la Raspberry Pi 4, se podrá utilizar el software adecuado para controlar la adquisición de imágenes y vídeos. En este caso, se dispone de una biblioteca de software llamada "picamera"que facilita el control de la cámara Raspberry Pi a través de un conjunto de funciones y comandos en Python. Esto permite la configuración de parámetros como la resolución, la calidad de compresión, la exposición y el balance de blancos, entre otros.

En cuanto al almacenamiento de los datos obtenidos por la cámara, estos se guardarán en la tarjeta SD de la Raspberry Pi 4. La tarjeta SD actúa como unidad de almacenamiento principal para el sistema, lo que permite guardar imágenes y vídeos capturados por la cámara sin necesidad de utilizar componentes de almacenamiento adicionales. Es importante destacar que los datos de la cámara no se utilizarán para el data-handling ni afectarán el funcionamiento general de la misión, ya que su propósito principal es documentar el despliegue y proporcionar información visual sobre el estado del satélite.

Las imágenes y vídeos se almacenarán en la tarjeta SD y se recuperarán después de que el satélite haya completado su misión y haya sido recuperado.

En resumen, la elección de la cámara Raspberry Pi para la aviónica del CubeSat se basa en su compatibilidad con la Raspberry Pi 4, la facilidad de integración y control, y la capacidad de cumplir con los requisitos específicos de la misión.

Figura 3.16: Módulo de cámara conectado al OBC. Imagen de [16]

3.9. Elección de Relé

En este apartado, se explorará la elección de un Relé para la aviónica del CubeSat. La función principal del relé es la de abrir y cerrar partes del circuito, actuando como un interruptor controlado electrónicamente. En este contexto, es esencial comprender en detalle cómo funcionan los relés, cómo se conectan al ordenador de a bordo, y las funciones específicas que cumplen en el sistema de aviónica.

Un relé es un componente electromecánico que permite controlar un circuito de mayor potencia mediante otro circuito de menor potencia. Esto se logra utilizando una bobina de alambre enrollado y un conjunto de contactos que se abren o cierran en función del estado de la bobina. Cuando se aplica una corriente eléctrica a la bobina, esta genera un campo magnético que atrae a un núcleo de hierro, haciendo que los contactos cambien su estado y permitiendo o eliminando el paso de la corriente a través del circuito de mayor potencia. Cuando la corriente en la bobina se interrumpe, el campo magnético desaparece, y los contactos vuelven a su estado inicial.

Los relés tienen uno o más juegos de contactos, y estos contactos pueden estar en estado "normalmente abierto"(NO, por sus siglas en inglés) o "normalmente cerrado"(NC):

- **Normalmente abierto (NO):** Cuando el relé no está energizado (o en reposo), el circuito conectado a estos contactos está abierto, es decir, la corriente no puede fluir a través de él. Cuando se aplica energía al relé, el circuito se cierra y la corriente puede fluir.
- **Normalmente cerrado (NC):** A la inversa, cuando el relé no está energizado, el circuito conectado a estos contactos está cerrado, es decir, la corriente puede fluir a través de él. Cuando se aplica energía al relé, el circuito se abre y la corriente no puede fluir.

El uso de NO o NC depende de la aplicación específica y del comportamiento deseado en el caso de una falla de energía. Se utilizan conexiones NO cuando se desea que un dispositivo o circuito esté apagado por defecto y solo se encienda cuando se energiza el relé. Un ejemplo sería una luz que se enciende cuando se presiona un botón. Se utilizan conexiones NC cuando se desea que un dispositivo o circuito esté encendido por defecto y solo se apague cuando se energiza el relé. Un ejemplo sería un sistema de alarma que se desactiva cuando se introduce correctamente un código.

Existen también relés con ambos tipos de contactos (NO y NC), que se pueden usar para controlar dos circuitos de manera opuesta. Cuando se energiza el relé, un circuito se cierra y el otro se abre.

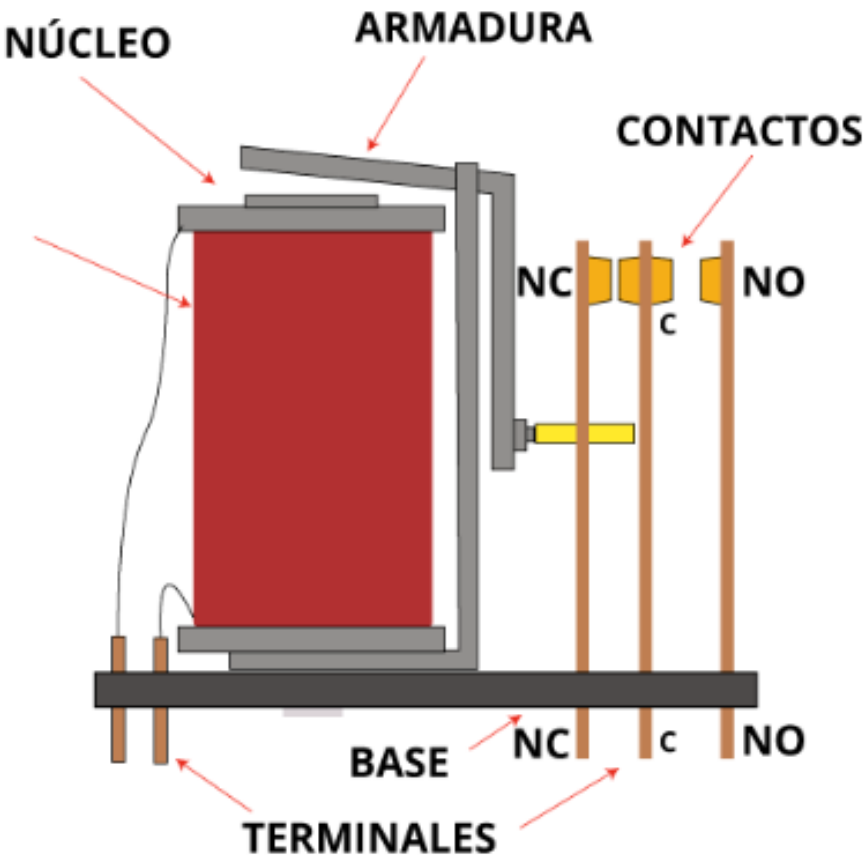

Figura 3.17: Imagen descriptiva de un relé con contactos NC y NO. Imagen de [17]

Una vez descrito el funcionamiento de un relé, se puede desarrollar su integración en la aviónica del satélite. Los relés se conectarán al ordenador de a bordo, en este caso una Raspberry Pi 4, a través de sus puertos GPIO (General Purpose Input/Output), que son una serie de pines de conexión en la placa del ordenador que permiten interactuar con dispositivos electrónicos externos. Estos pines GPIO pueden configurarse para proporcionar una tensión de salida (modo de salida) o para leer una tensión de entrada (modo de entrada), y pueden controlar dispositivos como relés, actuadores, sensores y otros componentes electrónicos. En el caso de los relés en la aviónica del CubeSat, los pines GPIO se utilizan para controlar la activación y desactivación de la bobina del relé, lo que a su vez permite controlar el estado de los contactos y, por tanto, el flujo de corriente en el circuito de mayor potencia.

En la aviónica del CubeSat, se utiliza un relé para controlar la parte de la aviónica asociada al solenoide, que es un dispositivo electromagnético utilizado para permitir el despliegue de la estructura del satélite. El relé en este caso permite controlar el flujo de corriente hacia el solenoide y, por tanto, activar o desactivar el dispositivo según sea necesario.

Uno de los aspectos clave en el uso de relés es la inclusión de un diodo en paralelo. Un diodo es un componente electrónico que permite el flujo de corriente en una sola dirección. Cuando la corriente en la bobina del relé se interrumpe, el campo magnético generado por la bobina colapsa rápidamente, lo que puede inducir una tensión de pico inversa, también conocida como voltaje de retroceso o FCEM (Fuerza Contra Elecro Motriz), en la bobina. Dicho voltaje puede ser lo suficientemente alto como para dañar los componentes electrónicos sensibles conectados al circuito.

Para proteger estos componentes, se coloca un diodo en paralelo con la bobina del relé, con la polaridad inversa a la dirección del flujo de corriente normal. Cuando se desconecta la alimentación de las cargas inductivas, el campo magnético almacenado en la bobina genera el voltaje de retroceso. El diodo se coloca con una polaridad opuesta a la de alimentación, de modo que cuando se genera el voltaje de retorno, el diodo lo elimina. Este diodo se conoce comúnmente como diodo de rueda libre o diodo de protección.

Una vez expuesto en detalle cómo funciona un relé, cómo se conecta al ordenador de a bordo y las funciones que cumple en la aviónica del CubeSat, se puede proceder a seleccionar un relé apropiado para el sistema. Al elegir un relé, es importante tener en cuenta factores como la capacidad de corriente y voltaje, la durabilidad y la fiabilidad del componente, así como la facilidad de integración con el resto del sistema de aviónica.

Un relé adecuado para esta aplicación sería el WINGONEER 5PCS KY-019 5V. Este relé presenta características apropiadas para el proyecto, como una capacidad de corriente de hasta 10 A, una tensión de conmutación máxima de 250 VAC/30 VDC, y una resistencia de contacto máxima de 1 MΩ.

Además, este relé es compatible y está diseñado y comúnmente extendido en aplicaciones dedicadas a la Raspberry Pi 4. Puede conectarse a los pines GPIO del ordenador de a bordo, lo que facilita su integración en el sistema de aviónica. El relé consta de la posibilidad de conectar el circuito en modo NC o NO. Por la naturaleza del uso que se la dará en la aviónica del satélite, las conexiones se realizarán mediante los terminales NO, pues el relé se encontrará una mayor parte de la misión sin necesidad de ser activado.

Figura 3.18: Imagen del KY-019 5V módulo relé. Imagen de [18]

Además de las características técnicas y la compatibilidad del relé WINGONEER 5PCS KY-019 5V con la Raspberry Pi 4, es importante tener en cuenta otros aspectos que pueden influir en la selección del relé adecuado para la aviónica del CubeSat, como su tamaño, peso y consumo de energía. El WINGONEER 5PCS KY-019 5V tiene unas dimensiones de 26 x 38 x 18 mm, lo que lo convierte en un componente compacto y fácil de integrar en el diseño del CubeSat.

La bobina del relé WINGONEER 5PCS KY-019 5V tiene una tensión nominal de 5 VDC, lo que implica un consumo de energía moderado. Sin embargo, cabe señalar que el relé solo consumirá energía cuando se active, es decir, cuando se envíe una señal desde el ordenador de a bordo para cambiar el estado del cicuito de alta potencia del relé.

Como se explicó anteriormente, el relé trabajará con un diodo de rueda libre colocado en paralelo que se encargará de proteger el circuito cuando se genere la Fuerza Contra Electromotriz al eliminar la señal entre el OBC y el relé. En este caso, el diodo elegido será el 1N4007. Este es un diodo de unión PN con una capacidad de corriente máxima de aproximadamente 1A. Se utiliza comúnmente en aplicaciones de rectificación de potencia debido a su alta capacidad de voltaje inverso, que es de aproximadamente 1000V. Este diodo es adecuado para la protección de relés, motores y otros inductores contra las corrientes inversas que pueden generarse cuando estos componentes se apagan, y permitirá mitigar los picos de voltaje inverso generados por los

componentes inductivos, aportando seguridad a la arquitectura de la aviónica.

Figura 3.19: Imagen del diodo 1N4007. Imagen de [19]

3.10. Elección de Solenoide

El presente apartado aborda la elección de un solenoide adecuado para la misión. Como se mencionó en apartados anteriores, el satélite consta de dos cuerpos independientes. Por un lado, se encuentra el cuerpo principal, que aloja el Ordenador de a Bordo (OBC), la mayor parte de la aviónica y la antena de emisión de microondas. Este cuerpo principal ocupa un volumen de 2U. Por otro lado, se encuentra el segundo cuerpo, que alberga el circuito de lectura de potencia transmitida y la rectena. Este segundo cuerpo ocupa un volumen de 1U.

Al inicio de la misión, ambos cuerpos se encuentran unidos, pero durante el transcurso de la misma, se despliega una estructura que permite separarlos hasta una distancia de 50 cm. Esta separación es posible gracias a un sistema de raíles retráctiles que, en posición vertical, permiten que el cuerpo más pequeño caiga a lo largo de la distancia mencionada. Es aquí donde entra en juego el solenoide, cuya función principal es la de activar el despliegue estructural.

El solenoide se conectará al OBC mediante un relé conectado a un puerto GPIO, permitiendo así que el ordenador controle su activación y desactivación. Dicho componente ha sido elegido

como solución al despliegue estructural solicitado por el requisto CSM-STR-REQ-1, pues supone un sistema de despliegue simple y con una interfaz sencilla con el resto de la electrónica. Además, los solenoides están ampliamente extendidos en la electrónica de este tipo de nanosatélites, lo que aporta un mayor grado de fiabilidad en dicha tecnología.

Para entender mejor la elección del solenoide, es importante explicar cómo funciona este componente. Un solenoide es un dispositivo electromecánico que convierte la energía eléctrica en movimiento lineal. Está compuesto por un carrete de alambre de cobre enrollado en forma de bobina y un núcleo de metal ferromagnético, como el hierro. Cuando se aplica una corriente eléctrica al alambre de la bobina, se crea un campo magnético alrededor de la misma. Este campo magnético atrae el núcleo de metal hacia el interior de la bobina, generando un movimiento lineal que puede utilizarse para accionar mecanismos, como el despliegue estructural en el satélite.

En el caso de la misión en cuestión, el solenoide debe cumplir con una serie de requisitos y especificaciones para asegurar el éxito del despliegue estructural:

- **Fuerza de actuación:** El solenoide debe ser capaz de generar suficiente fuerza para activar el despliegue de la estructura, teniendo en cuenta que su vástago, en caso de no contar con la fuerza suficiente, se puede quedar bloqueado a mitad de su carrera nominal.
- **Consumo de energía:** El solenoide debe tener un consumo de energía eficiente, ya que los recursos energéticos durante la misión se preveen limitados.
- **Tamaño y peso:** Dado que el espacio y el peso son factores críticos en el diseño de un satélite, el solenoide debe ser compacto y ligero para adaptarse a las restricciones del CubeSat.
- **Capacidad de activación múltiple:** Como medida de seguridad, el solenoide se activará de manera periódica a lo largo de la misión para asegurar el éxito del despliegue. Por lo tanto, debe ser capaz de soportar múltiples activaciones sin perder rendimiento o sufrir desgaste prematuro.

Para la selección del solenoide adecuado, se ha llevado a cabo una matriz de trade-off entre distintas opciones de solenoides del mercado, evaluando y comparando sus características y prestaciones con respecto a los requisitos y especificaciones de la misión. En la tabla 3.5 , se presenta la matriz de trade-off, donde se analizan varios parámetros, incluyendo la fuerza de actuación o la durabilidad entre otros.

Modelo	Fuerza de actuación (N)	Carrera máxima (mm)	Tensión nominal (V)	Consumo de corriente (A)	Durabilidad (Número de actuaciones)	Tamaño
K1037L-12V	7.84	10 mm	12	3.3	5×10^5	37 mm x 26 mm x 20 mm
Hechen Electro-imán HS-1264B	55	10 mm	12	2.5	1×10^6	64 mm x 32 mm x 38 mm
Johnson Electric SOB31-12VDC	3.9	12 mm	12	0.6	5×10^5	32 mm x 20 mm x 20 mm
Guardian Electric A420-067222	4.2	8 mm	12	0.55	1×10^6	35 mm x 21 mm x 21 mm
JF-0530B	5	10 mm	6	0.3	-	30 mm x 16 mm x 15 mm

Tabla 3.5: Matriz trade-off de solenoides

Tras el análisis y comparación de las distintas opciones en la matriz de trade-off, se ha seleccionado el Hechen Electroimán de solenoide HS-1264B como la opción más adecuada para la misión. Este modelo presenta varias ventajas en comparación con las otras opciones analizadas. En primer lugar, la fuerza de actuación del HS-1264B es de 55 N, lo que garantiza que será capaz de realizar el despliegue estructural de manera eficiente y efectiva. Este parámetro ha sido la clave de la elección de este modelo de solenoide, pues su respuesta respecto a la fuerza de actuación es muy superior a la del resto de opciones.

Por otro lado, la carrera máxima de 10 mm del HS-1264B es suficiente para permitir el despliegue estructural del satélite. Además, la tensión nominal y consumo de corriente lo hacen compatible con un sistema de energía del satélite, pues no requiere una potencia desmedida e inalcanzable para un módulo de batería limitado, como es el de un CubeSat.

En cuanto a durabilidad, el HS-1264B tiene una durabilidad de 1 millón de ciclos de activación, lo que garantiza un funcionamiento fiable y duradero a lo largo de toda la misión.

En cuanto al tamaño, con unas dimensiones de 64 mm x 32 mm x 38 mm, el HS-1264B es

el solenoide de mayor tamaño entre los comparados. No obstante, este sigue siendo compacto y ligero, lo que facilita su integración en la estructura del satélite. El incremento de tamaño de este modelo, queda completamente compensado con el aumento en fuerza de actuación que proporciona el solenoide, lo que permite que sea la opción más interesante.

En resumen, el Hechen Electroimán de solenoide HS-1264B ha sido seleccionado como el solenoide más adecuado para la arquitectura de la aviónica del satélite. Este solenoide cumple con los requisitos de despliegue estructural, eficiencia energética y durabilidad, lo que garantiza un funcionamiento fiable y eficiente durante toda la misión. Además, su fuerza de hace que sea una opción óptima para la integración en la estructura del satélite.

Figura 3.20: Imagen del solenoide seleccionado. Imagen de [20]

3.11. Elección de Batería

En la planificación y diseño de cualquier misión satelital, el cálculo del coste energético representa un componente crucial. A fin de garantizar el funcionamiento óptimo del satélite durante toda la misión, es imperativo determinar la demanda energética total de cada uno de los componentes y sistemas del satélite.

Para calcular el coste energético de la misión, primero se suman las demandas energéticas de cada componente individual. Estas demandas se expresan generalmente en vatios (W) o en miliwatios (mW). La suma total representa el consumo energético medio del satélite. Sin embargo, es importante considerar también los picos de demanda energética que pueden surgir en determinados momentos, como durante maniobras o comunicaciones intensivas. Por ello, se va a construir un perfil de consumo energético a lo largo de la misión, el cual modelizará la cantidad de energía eléctrica que deberán aportar las baterías en cada momento de la misión. Tras el conocimiento del perfil de consumo a lo largo de la misión, se puede integrar dicho valor en el tiempo para obtener el consumo energético total, el cual será otro requisito que deberán satisfacer las baterías.

Una vez determinado el consumo energético total, es esencial considerar el tipo y las características de la batería que mejor se adapten a las necesidades de la misión. En el caso de las misiones tipo cubesat, actualmente se prefieren las baterías de polímero de litio (LiPo) o las baterías de iones de litio (Li-ion), por encima incluso de las tradicionales de níquel cadmio (NiCd). Estas baterías han demostrado ser eficientes en términos de densidad energética, confiabilidad y longevidad en el hostil ambiente espacial.

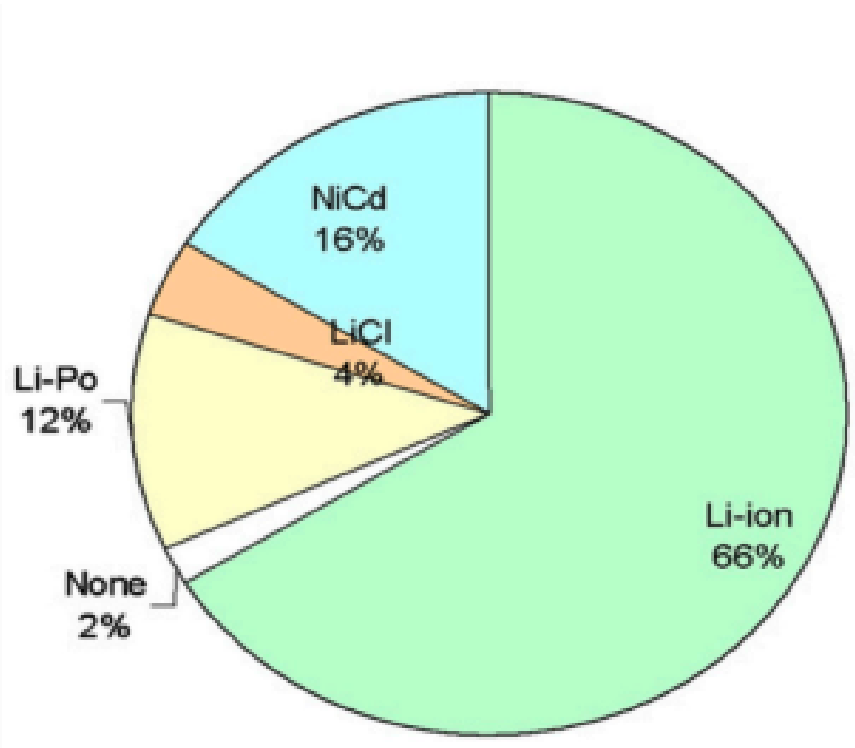

Figura 3.21: Tipos de baterías utilizadas en picosatélites y nanosatélites hasta 2010. Imagen de [21]

A nivel interno, las baterías de polímero de litio y las baterías de iones de litio operan mediante un proceso de intercambio de iones de litio entre el cátodo y el ánodo. Durante la descarga, los iones de litio se mueven del ánodo (hecho generalmente de grafito) hacia el cátodo (compuesto comúnmente de óxido de metal). Durante la carga, este proceso se invierte. El electrolito, que puede ser líquido en las baterías Li-ion o un polímero sólido en las baterías LiPo, facilita este movimiento de iones.

Es crucial mencionar que, en esta misión específica, las baterías se utilizarán únicamente en su modo de descarga. El satélite no cuenta con ningún medio para cargarlas nuevamente en la misión. Esto significa que la capacidad inicial de la batería debe ser suficiente para satisfacer todas las demandas energéticas de la misión hasta su conclusión. Esta característica única de la misión eleva la importancia de seleccionar baterías con una capacidad adecuada, una baja tasa de autodescarga y una confiabilidad probada en el tiempo.

En primer lugar, se va a obtener una aproximación del perfil de consumo en cada momento de la misión. Para ello, se partirá de una misión modelizada de 4 horas de vuelo (el tiempo de caída no es considerado). En dicha misión, a partir de la tercera hora, se comenzará con la trasmisión de microondas. El tiempo de uso del relé, así como el del solenoide, se estimarán inferiores a 1 minuto, pues estos se accionarán de manera puntual en el momento del despliegue. Por otro lado, la cámara se accionará cada 20 minutos, mientras que los lectores y sensores del sistema de aviónica se considerarán como en uso a lo largo de toda la misión.

En la tabla 3.6 se desglosa el consumo de cada componente:

Componente	Potencia [W]	Tiempo de uso [h]	Energía consumida [Wh]	Energía consumida [mAh]	Voltaje de suministro [V]	Amperaje de suministro [A]
OBC	5.5	4	22	4400	5	1.1
Generador de señal de microondas	0.5	1	0.5	100	5	0.1
Amplificador	4.8	1	4.8	960	5	0.96
Ina260	0.0063	4	0.025	5	5	0.0013
Sensor óptico	0.02	4	0.08	24.2424	3.3	0.0061
Altímetro barométrico	0.0026	4	0.01056	3.2	3.3	0.0008
Cámara	1.25	1	1.25	250	5	0.25
Relé	0.35	0.01	0.0035	0.7	5	0.07
Solenoide	30	0.01	0.3	25	12	2.5
Total	42.4289	-	28.9691	5768.1424	-	4.9881

Tabla 3.6: Tabla de consumo por componente para la misión modelizada

Por otro lado, se muestra en la figura 3.22 el perfil de consumo de potencia eléctrica a lo largo de la misión modelizada.

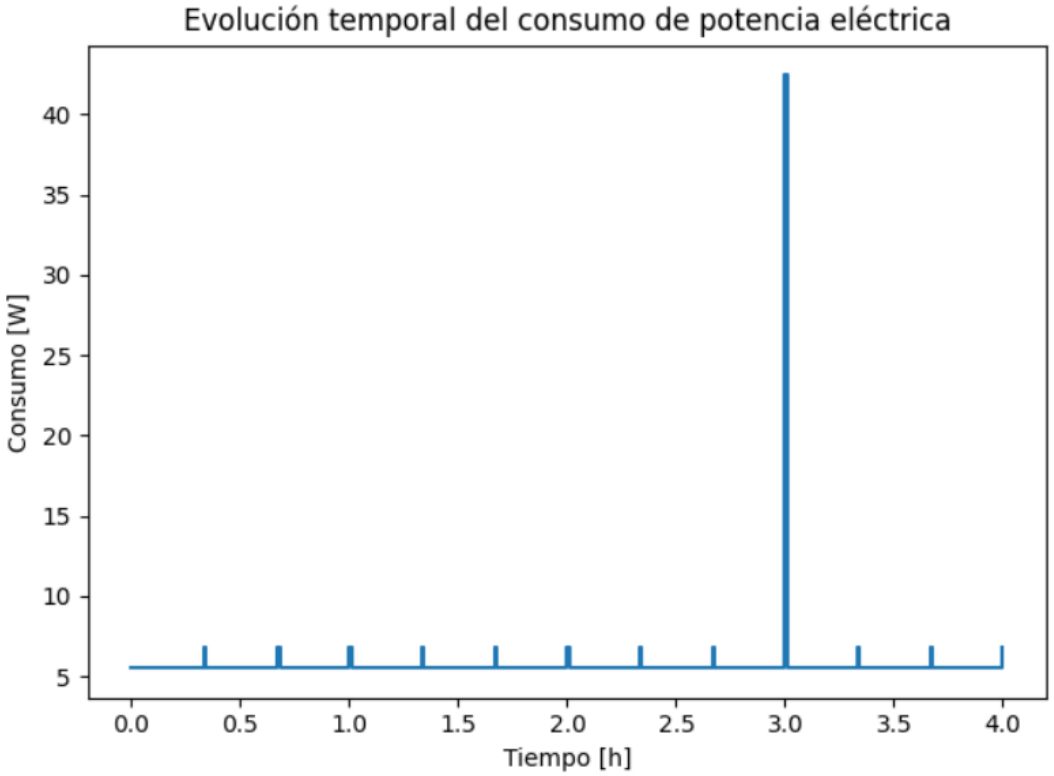

Figura 3.22: Evolución temporal del consumo de potencia eléctrica a lo largo de la misión (Elaboración propia)

A partir del análisis de los datos mostrados en la tabla, se pueden obtener varios datos de gran relevancia a la hora de definir los requisitos con los que tendrá que cumplir la batería seleccionada. En primer lugar, se muestra como, en momentos puntuales, la batería deberá ser capaz de suministrar 42,43 vatios de potencia, con un amperaje cercano a 5 amperios.

Además, la capacidad máxima de la batería seleccionada deberá superar los 5768,2 mAh.

Tras el análisis anterior y la obtención de los requisitos eléctricos de la batería, se propone el uso de una batería Li-Ion para dar respuesta al requisito CSM-PWR-REQ-1 del subsistema de la aviónica. En el mercado, se encuentran gran cantidad de módulos de baterías, los cuales a partir de situar un conjunto de baterías en serie o paralelo (en función de la necesidad de aumentar el voltaje de suministro de potencia del módulo, o de su capacidad máxima), son capaces de cumplir con los requisitos eléctricos de la misión. Dado que dichos modelos de baterías son prácticamente similares en cuanto a su comportamiento y dimensiones, se propone un modelo de módulo de batería similar al mostrado en la figura 3.23.

Figura 3.23: Imagen de batería de referencia. Imagen de [22]

Tras la elección de un módulo de batería capaz de cumplir con los requisitos de la misión, cabe recalcar que los componentes elegidos para la aviónica del satélite requieren distintos niveles de voltaje. La solución propuesta pasa por separar el suministro de potencia, de modo que en uno de los caminos se suministren 12 voltios (necesarios para el solenoide), y en el otro, 5 voltios, los cuales suministrarán al ordenador de a bordo. Dicho ordenador, a su vez, será el responsable de repartir la potencia eléctrica al resto de componentes.

3.12. Elección de Regulador de Tensión

Como se mencionó anteriormente, se debe integrar un componente que realice un control del voltaje comunicado al OBC, cumpliendo con el requisto CSM-PWR-REQ-2, el cual en este caso será de 5V, como requiere el ordenador central. A la hora de elegir un regulador de tensión a 5V, los parámetros que influyen en la decisión son muy reducidos. Los más críticos serán el voltaje de salida que sea capaz de proporcionar el regulador y la intensidad máxima que permitan que fluya a través de ellos. Un último parámetro podría ser el precio, pero para este componente (el cual es relativamente barato), la diferencia de precio entre las distintas opciones de mercado es muy reducida.

Por el lado de los requisitos de voltaje, el regulador de tensión debe estar preparado para dar un output de 5V, por lo que una opción es la de buscar reguladores cuya única tensión de salida sea la mencionada. No obstante, existe otra opción de alcanzar dicha tensión de salida, y es mediante un regulador de tensión con tensión de salida variable. Dentro de esta opción, lo habitual es encontrar reguladores con un controlador analógico (como una rueda pequeña) a partir del cual se establece el voltaje de salida de interés (dentro del rango de voltajes que permite el regulador).

Por parte del requisito de intensidad, el regulador debe permitir suministrar una intensidad suficiente para alimentar a todos los componentes que obtienen su potencia eléctrica a partir del ordenador de a bordo, así como al ordenador mismo. Para ello, se parte de la intensidad de trabajo de cada componente, buscando de ese modo el conocer la intensidad de trabajo a la que deba trabajar el regulador.

Componente	Amperaje de suministro [A]
OBC	1.1
Generador de señal de microondas	0.1
Amplificador	0.96
Ina260	0.0013
Sensor óptico	0.0061
Altímetro barométrico	0.0008
Cámara	0.25
Relé	0.07
Solenoide	2.5
Total	4.99

Tabla 3.7: Corriente de suministro a los componentes de la aviónica

En la tabla 3.7, se muestra la corriente de suministro de todos los componentes de la aviónica. No obstante, el regulador de voltaje, en la situación de máxima corriente, simplemente suministrará corriente al OBC, generador de señal de microondas, Ina260, sensor óptico, altímetro barométrico y cámara. Como resultado, la corriente máxima que fluirá a través del regulador de tensión será de 1,46 amperios.

El modelo de regulador de voltaje propuesto es el LM2596, el cual es capaz de regular el voltaje de salida entre 1,2 voltios y 37 voltios. Además, permite una corriente de salida máxima de 3 amperios, lo que supera con creces la cantidad necesaria. Por último, este modelo de regulador de voltaje presenta una pérdida de energía durante la conversión inferior al 8 %,

lo que permite que la capacidad del módulo de batería utilizado como referencia proporcione energía eléctrica al satélite manteniendo un margen de capacidad elevado.

Figura 3.24: Imagen del regulador de voltaje propuesto. Imagen de [23]

3.13. Arquitectura Final del Subsistema de Aviónica

Tras analizar y realizar la elección de todos los componentes de la aviónica, En este apartado se abordará la formación de la arquitectura final. En ella, se considerarán los requisitos a nivel de interfaz de cada componente, buscando una distribución que los satisfaga todos.

Otro aspecto a tener en cuenta es la distribución de potencia. Como se mencionó en apartados anteriores, la potencia suministrada por el módulo de batería se distribuirá por distintas vías, en función del voltaje que se requiera. En primer lugar, se encontrará un camino en el cual el voltaje será regulado a 5V, tensión de alimentación del ordenador de a bordo principalmente, pues después este suministrará potencia a casi todos los componentes electrónicos de la aviónica. Como excepción, se encuentra el amplificador de la señal de microondas de la payload, el cual requerirá una potencia tal que deberá conectarse directamente a la línea de 5V.

El camino restante es aquel que requiere una tensión de 12V, el cual suministra potencia al solenoide.

Es importante remarcar que, a lo largo de la misión, la batería suministra un voltaje menor a medida que se descarga. Esta bajada de voltaje, en sus términos máximos, no alcanza los 2V, de modo que el camino donde la tensión se regula a 5V no se ve afectado. Por otro lado, el solenoide sí que sufre esta bajada de tensión, pues su comunicación mediante la batería es directa. En el caso de un solenoide, si se ve suministrado con un voltaje menor del indicado por el fabricante, este puede perder fuerza de actuación. En el caso de la aviónica del cubesat, al ser una bajada inferior a 2V, la pérdida de fuerza de actuación no será significativa para viabilidad del despliegue estructural, de modo que la descarga del módulo de batería no tendrá una consecuencia directa en la actuación del CubeSat.

Una vez especificadas las conexiones y la distribución de potencia, se procede a mostrar una representación de la arquitectura final de la aviónica. Se utilizará para la representación el software Fritzing. Dicho software, presenta una herramienta de diseño de electrónica muy extendida en aplicaciones de ingeniería robótica, y, como tal, posee librerías y diseños sobre la mayoría de los componentes que se han mencionado en este trabajo. El resultado obtenido de la representación se muestra en la figura 3.25.

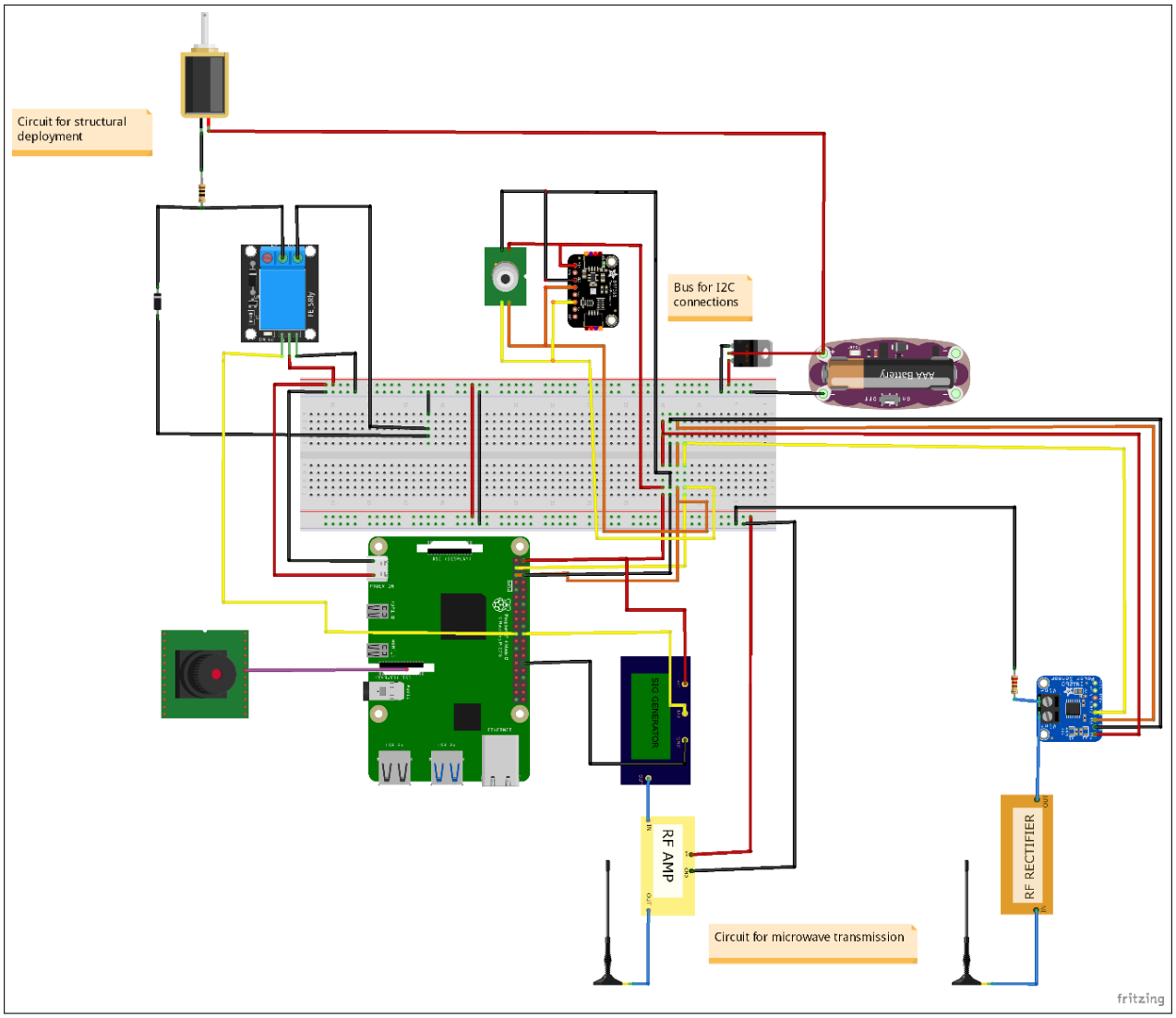

Figura 3.25: Representación de la arquitectura final de la aviónica del CubeSat (Elaboración propia)

En la figura 3.25, se muestran tanto los componentes elegidos para la aviónica, como los cables y conexiones que realizan las interfaces entre sus puertos. Se ha hecho uso de una breadbord para simplificar la circuitería lo máximo posible.

En la representación, los cables rojos y negros son dedicados a la trasmisión de potencia. Por otro lado, los cables naranjas se utilizan en la señal de reloj SCL (del inglés Serial Clock), pertenecientes a las interfaces I2C. En cuanto a las conexiones mostradas en amarillo, estas representan las conexiones de datos, como las que se pueden dar entre un puerto SDA (del inglés Serial Data) en las interfaces I2C, o las conexiones con puertos GPIO del ordenador central.

Tanto el lector de potencia, como el sensor de distancia y el altímetro barométrico, comparten el mismo tipo de interfaz con el ordenador de a bordo (interfaz I2C). Por esta razón, se ha utilizado un bus I2C en el que se sitúan dichos componentes, ubicados con distintas localizaciones o "addresses", las cuales serán utilizadas en el desarrollo del código que rija la misión.

Finalmente se ha obtenido una configuración de todos los componentes involucrados en el hardware de la aviónica del CubeSat. El siguiente paso, será el de diseñar un código a implementar en el OBC, el cual se ejecute durante toda la misión y permita dirigir cada una de las acciones que debe tomar la electrónica del satélite.

Capítulo 4
Diseño e Implementación del Software de la Aviónica

El software de aviónica es el núcleo que proporciona control, monitoreo y funcionalidad a los sistemas embarcados en un satélite. Este software es crucial para garantizar que el satélite funcione de manera eficiente y realice sus misiones con precisión. Su diseño y desarrollo son esenciales para determinar la eficiencia, fiabilidad y seguridad de las operaciones del satélite.

En este apartado, se tomará la configuración de la arquitectura diseñada en el apartado anterior, y se construirá para ella un software que se integrará en el ordenador central. Al igual que en el apartado anterior, el diseño de la estructura del código se definirá para responder ante unos requisitos de diseño, los cuales serán requeridos por la misión. En este caso, cabe recalcar que se deben definir los estados en los que se encontrará el satélite a lo largo de la misión, así como las principales tareas que deberá ejecutar en cada una de las etapas del vuelo.

Mediante la definición de los requisitos del software y del análisis de riesgos, se podrá definir la estructura del código, de modo que el siguiente paso sea su desarrollo e implementación. Una vez obtenido un código para la realización de la misión, se abordará la validación del código mediante un test de simulación del vuelo, donde se replicarán situaciones similares a los de la misión de diseño y se pondrá a prueba el data-handling de la aviónica.

4.1. Modos de la Misión

En el contexto de los satélites, un "modo"se refiere a un estado operativo específico del satélite que define un conjunto particular de funciones o comportamientos. Estos modos son esenciales para la operación del satélite ya que determinan cómo responderá ante ciertos eventos

o condiciones. Ejemplos comunes de modos incluyen el modo de inicio (después del lanzamiento), el modo de operación normal, el modo de ahorro de energía y el modo de recuperación de fallos.

La importancia de los modos radica en que permiten una operación flexible y adaptable del satélite. Por ejemplo, si un satélite experimenta un problema, puede cambiar automáticamente a un modo seguro que conserve energía y proteja sus sistemas críticos mientras los operadores en tierra diagnostican y abordan el problema.

En el caso del satélite que se está diseñando, no mostrará una gran cantidad de modos, pues la misión será relativamente corta. De este modo, los modos estarán principalmente caracterizados por la cantidad de componentes que se encuentran en uso en cada momento.

El software de aviónica debe ser diseñado teniendo en cuenta estos modos. Cada modo tiene requisitos de software específicos asociados que definen cómo debe comportarse el satélite en ese modo. Esto implica que el software debe ser lo suficientemente versátil como para manejar transiciones entre modos y ejecutar funciones específicas asociadas con cada modo. Además, la precisión es esencial, ya que un fallo en el reconocimiento o ejecución de un modo podría tener consecuencias graves para la misión.

Los modos a considerar para la misión serán:

- **Modo de travesía:** En este modo, el satélite se encontrará con la estructura contraída, y realizará, de forma periódica, la lectura y almacenamiento de los datos leídos por sus sensores. Además, realizará una imagen y un video de forma periódica.

  El modo de travesía se caracterizará por ser el modo en el cual se dará gran parte del ascenso, cuando el satélite se encuentre contraído y sin emitir energía por su payload.

- **Modo de despliegue:** Pasada la travesía, se accionará el solenoide tras un tiempo definido, con el fin de desplegar la estructura del satélite y prepararlo para la transmisión. Dicha acción requerirá un alto coste energético, por lo que apenas durará un par de segundos.

  Durante el modo de despliegue, se respetarán los periodos establecidos para cada componente de lectura en el modo de travesía. Su principal característica, será la periodicidad de este modo, una vez sea activado por primera vez. Esto implica que, pasada la travesía inicial, el solenoide será activado de manera periódica durante el resto

de la misión. Esta decisión tiene una justificación a nivel de fallo de despliegue estructural, la cual será explicada en el apartado 4.3.

- **Modo de emisión:** dicho modo comienza con la trasmisión de energía a partir de la activación del sistema de transmisión de microondas. Además, se seguirán realizando la lectura y almacenamiento de los datos obtenidos por el resto de componentes.

  El modo de emisión se activará durante una hora, en la cual se tratará de realizar la lectura de la potencia transmitida por la payload.

- **Modo de reposo:** Una vez finalizada la trasmisión de microondas, el satélite permanece en su modo final, en el cual realiza y guarda la lectura de sus sensores hasta que termine la misión.

Si bien el satélite cuenta con unos modos marcados a lo largo de la misión, el cambio de un modo a otro se llevará a cabo mediante la activación o desactivación de los componentes pertinentes en cada momento. La tabla 4.1, muestra de manera visual los componentes que se encuentran activados en cada uno de los modos definidos.

Componente	Modo de travesía	Modo de despliegue	Modo de emisión	Modo de reposo
Cámara	Activado	Activado	Activado	Activado
Sensor láser	Activado	Activado	Activado	Activado
Altímetro	Activado	Activado	Activado	Activado
Solenoide	No Activado	Activado	No Activado	No Activado
Lector de potencia	Activado	Activado	Activado	Activado
Sistema de transmisión	No Activado	No Activado	Activado	No Activado
Estructura del satélite	No desplegada	Desplegada	Desplegada	Desplegada

Tabla 4.1: Modos de operación del satélite durante la misión

De este modo, el control del momento en el que comienza a trabajar un componente, o la periodicidad con la que se activa, serán los parámetros que regirán el cambio de un modo a otro, y, por tanto, los que regirán en que momento el satélite debe llevar a cabo cada una de sus tareas.

4.2. Requisitos del Software

El proceso de desarrollo de software comienza con la definición de requisitos claros y concisos. Estos requisitos describen en detalle lo que se espera del software, incluyendo cómo debe interactuar con el hardware, cómo debe responder a diferentes entradas y cuáles son sus objetivos de rendimiento.

En el caso de los requisitos del software, estos indicarán como se regirá la programación detrás de cada componente, así como su relación con el data-handling del ordenador central. Además, los requisitos definirán con exactitud cómo se debe utilizar cada componente y en qué momento hacerlo.

A partir de la misión definida, así como los modos del satélite a lo largo de esta, se presentan los requisitos que especifican el diseño y la estructura del software del CubeSat en la tabla 4.2.

No.	Título	Requisito	Explicación	Método de verificación
CSM-MSN-REQ-1	Código modular	La estructura del código será modular, con un bucle principal y un módulo por cada componente conectado al OBC	Para asegurar la legibilidad y organización en el código, este debe ser modular	Inspección del código
CSM-MSN-REQ-2	Bucle principal en continua ejecución	El bucle principal se debe ejecutar a lo largo de toda la misión	El software debe mantenerse activo a lo largo de toda la misión	Inspección del código
CSM-MSN-REQ-3	Data handling desde el bucle principal	Cada módulo tendrá por lo menos una función de data handling que será llamada desde el bucle principal	Se deben gestionar todas las tareas a lo largo de la misión	Inspección del código
CSM-MSN-REQ-4	Información de vuelo almacenada	Los datos obtenidos por los lectores serán almacenados en un documento de información de vuelo	El análisis de la misión se realizará a partir de un documento con la información del vuelo	Inspección del documento de información del vuelo

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-INA-REQ-1	Lectura y almacenamiento de datos del monitor de potencia	Se deben extraer datos de voltaje, intensidad y potencia	El análisis de la trasmisión de energía se realizará a partir de dichos datos	Inspección del documento de información del vuelo
CSM-INA-REQ-2	Periodicidad de la lectura	Se realizará una lectura en cada ejecución del bucle principal	Al ser un componente con un bajo consumo, se puede realizar una lectura en todo momento de la misión	Inspección del código
CSM-ALT-REQ-1	Lectura y almacenamiento de datos de lectura de presión	Se debe leer y almacenar el valor de presión	Los valores de presión serán utilizados en el análisis de la misión	Inspección del documento de información del vuelo
CSM-ALT-REQ-2	Obtención de altura a partir de la presión	La altura será obtenida a partir de aplicar el modelo ISA con la presión	La obtención de la altura a partir de la presión se realizará directamente en el código	Inspección del código
CSM-ALT-REQ-3	Periodicidad de la lectura	Se debe realizar una lectura de presión cada 60 segundos	Debido al margen de error que hay en la lectura de presión y la velocidad ascensional del vehículo lanzador, un dato por minuto es suficiente para el análisis de la misión	Inspección del código
CSM-CAM-REQ-1	Toma de imágenes y vídeos	Se deben tomar y almacenar las imágenes y vídeos	El contenido visual obtenido en la misión será utilizado para su análisis	Inspección de imágenes y vídeos almacenados
CSM-CAM-REQ-2	Periodicidad de imágenes y vídeos	Se debe tomar una imagen y un vídeo cada 20 minutos	Al tomar una imagen y vídeo cada 20 minutos se registrará la misión en todas sus etapas	Inspección del código
CSM-CAM-REQ-3	Duración de vídeo	Los vídeos deberán tener una duración de 10 segundos	Una duración de 10 segundos será suficiente para el análisis	Inspección del código
CSM-CAM-REQ-4	Indicación de toma de imagen y vídeo	Se debe reflejar en la información del vuelo que se han tomado una imagen y un video	El momento en el que se han tomado las imágenes y los vídeos será utilizado en el análisis de misión	Inspección del documento de información del vuelo

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-LSR-REQ-1	Toma de lecturas del láser	Se deben tomar y almacenar las lecturas del sensor láser de distancia	La lectura de la distancia entre los cuerpos de la estructura se requerirá en el análisis de la misión	Inspección del documento de información del vuelo
CSM-LSR-REQ-2	Periodicidad de la lectura	Se debe ejecutar la lectura cada 5 segundos	Una lectura cada 5 segundos permite obtener información del estado estructural del satélite en todo momento	Inspección del código
CSM-LSR-REQ-3	Interpretación de distancia	En caso de que la distancia registrada por el láser sea igual o mayor a 50 cm, se debe indicar que la estructura está desplegada	El momento en el que la lectura alcanza los 50 cm, se considera la estructura como desplegada	Inspección del código
CSM-GEN-REQ-1	Inicio de la transmisión inalámbrica	Se debe iniciar la trasmisión de microondas pasados 180 minutos de ejecución del código	Pasadas 3 horas de vuelo, la altura será suficiente como para probar la tecnología a demostrar	Inspección del código
CSM-GEN-REQ-2	Duración de la transmisión	La trasmisión debe durar 1 hora	1 hora será margen suficiente para probar la tecnología	Inspección del código
CSM-GEN-REQ-3	Indicación de transmisión	Se debe almacenar una indicación que señale si la transmisión está activada o no	El uso del sistema de transmisión se requerirá en el análisis de la misión	Inspección del documento de información del vuelo
CSM-SOL-REQ-1	Inicio de la activación	Se debe accionar por primera vez el solenoide pasados 170 minutos de ejecución del código	El accionamiento del solenoide marcará el despliegue estructural	Inspección del código
CSM-SOL-REQ-2	Duración de la activación	El solenoide debe mantenerse activo 2 segundos cada vez que se accione	Se activará el solenoide el tiempo suficiente para que realice su carrera máxima	Inspección del código

Sigue en la página siguiente.

No.	Título	Requisito	Explicación	Método de verificación
CSM-SOL-REQ-3	Periodicidad en la activación	Una vez activado por primera vez, cada 5 minutos el solenoide debe volver a activarse	Para asegurar el despliegue estructural, se realizará una activación periódica del solenoide	Inspección del código
CSM-SOL-REQ-4	Indicación de la activación	Se debe almacenar información que indique si el solenoide está activado o no	La activación del solenoide se requerirá en el análisis de la misión	Inspección del documento de información del vuelo

Tabla 4.2: Requisitos del subsistema de aviónica

4.3. Análisis de Riesgos y Fallos

El análisis de riesgos y fallos es una fase crucial en el desarrollo del software de aviónica, especialmente dado el entorno hostil del espacio y las consecuencias potencialmente desastrosas de un fallo en el sistema. Esta fase se enfoca en identificar, evaluar y mitigar los posibles riesgos asociados con el software, así como anticipar y planificar posibles fallos.

La primera etapa del análisis implica identificar todos los posibles riesgos que podrían afectar el correcto funcionamiento del software de aviónica. Estos pueden surgir debido a factores externos, como la radiación espacial, o internos, como errores de código o fallos de hardware. Los riesgos suelen categorizarse en base a 2 parámetros: probabilidad e impacto. En base a estos dos parámetros, los fallos pueden categorizarse desde insignificantes, hasta catastróficos.

Una vez identificados los riesgos, se evalúan en función de su probabilidad de ocurrencia y la gravedad de su impacto.

Además del análisis de riesgos, es fundamental anticipar posibles fallos en el software y planificar cómo el sistema responderá en caso de que ocurran. Esto implica diseñar el software de manera que pueda detectar y corregir fallos por sí mismo o cambiar a un modo seguro hasta que el fallo pueda ser abordado por operadores en tierra.

En cuanto a los posibles fallos a los que se puede enfrentar el satélite en la misión, la

posibilidad de actuación respecto a ellos es limitada una vez haya comenzado la misión. Por esa razón, en este capítulo se hará énfasis en la mitigación de los principales riesgos a los que se enfrenta el satélite.

Se presentan a continuación los principales riesgos a los que se enfrenta el satélite, así como las medidas que serán tomadas para reducirlos o, en el mejor de los casos, mitigarlos:

- **Riesgo 1:** Fallo en la lectura de un componente. Es altamente probable que, una vez comience la misión, varias medidas tomadas por los sensores del satélite fallen. Como solución, en el código se interpretarán todos los posibles escenarios en los que una medida falle, asignando a la variable que se trate de medir, con el símbolo "$".

  De este modo, cuando la variable de una lectura sea guardada en el documento de información del vuelo, este mostrará el símbolo "$", por lo que dicho fallo quedará ubicado y localizado temporalmente.

- **Riesgo 2:** Fallo en la electrotecnia del satélite. Elevando el riesgo 1 al extremo, puede ocurrir que, uno o varios componentes fallen en su comunicación en físico a nivel de interfaz, así como en la propia electrónica interna del componente. Este caso, si bien es más improbable que el anterior, presenta un impacto mucho mayor, pues incapacita al componente o conexión dañada de realizar su trabajo.

  Como medida de mitigación del riesgo, todas las decisiones que involucren la actuación de una parte del satélite o el cambio de uno de sus modos, no dependerán de la lectura de ningún componente. De este modo, decisiones como el momento en el cual se despliega la estructura, o el momento en el cual se comienza la transmisión inalámbrica, serán determinadas por una medida temporal. En muchas misiones, el precursor a que se cambe de modo es una medida, por ejemplo, de la altura de vuelo. Si se implementase la misma metodología en el CubeSat que se está diseñando, se expondría al mismo a no desplegar su estructura o comenzar la transmisión por un fallo en el altímetro barométrico o en el sensor óptico de distancia.

  Por otro lado, se añadirá un fichero de fallos y avisos, en el cual se indicará el momento en el que se haya perdido la comunicación con un componente.

- **Riesgo 3:** Fallo en el despliegue de la estructura. Existe la posibilidad de que, al accionar el solenoide, la estructura de los raíles que permite el despliegue falle. Dicho fallo sería catastrófico, pues sin el despliegue estructural, las antenas de microondas no se

encontrarían a la distancia de diseño para la transmisión, por lo que no se podría probar correctamente la tecnología.

Para evitar dicho fallo, se accionará periódicamente el solenoide una vez se haya mandado accionar por primera vez, de modo que se reduzca al mínimo la probabilidad de que se quede alguna pieza atascada al accionarlo una vez.

Si bien no han sido considerados para el diseño de la aviónica del satélite, existen otro tipo de riesgos, asociados al vehículo lanzador. El vehículo lanzador se ha diseñado y probado de manera ajena al satélite, al igual que será controlado de forma remota. Esto implica, que el satélite no contará en ningún momento con la capacidad de actuar de ningún modo sobre el vehículo.

Como consecuencia, el riesgo ante un fallo por parte del vehículo lanzador que comprometa la trayectoria de diseño para el satélite, no es mitigable. Si bien el vehículo lanzador cuenta con medidas de seguridad en su diseño (con el fin de evitar cualquier potencial fallo durante la misión), en este trabajo no se abordarán dichas casuísticas.

4.4. Definición y Desarrollo del Código

Una vez definidos los modos en los que se encontrará el satélite a lo largo de la misión, así como los requisitos que debe cumplir el software y las medidas de mitigación de riesgos, se puede proceder a desarrollar el código que ejecutará dicho software.

En primer lugar, se va a presentar una estructura organizada de cómo debe funcionar el código, en base a las directrices marcadas en los apartados anteriores. A partir de dicha estructura, se desarrollará el código final.

La estructura del código será modular. Esto significa, que cada componente tendrá asociadas un paquete de funciones y variables, que se comunicarán con un código "main" durante la ejecución del código.

Se presenta en la figura 4.1 , un diagrama de bloques que representa las funciones del código del software.

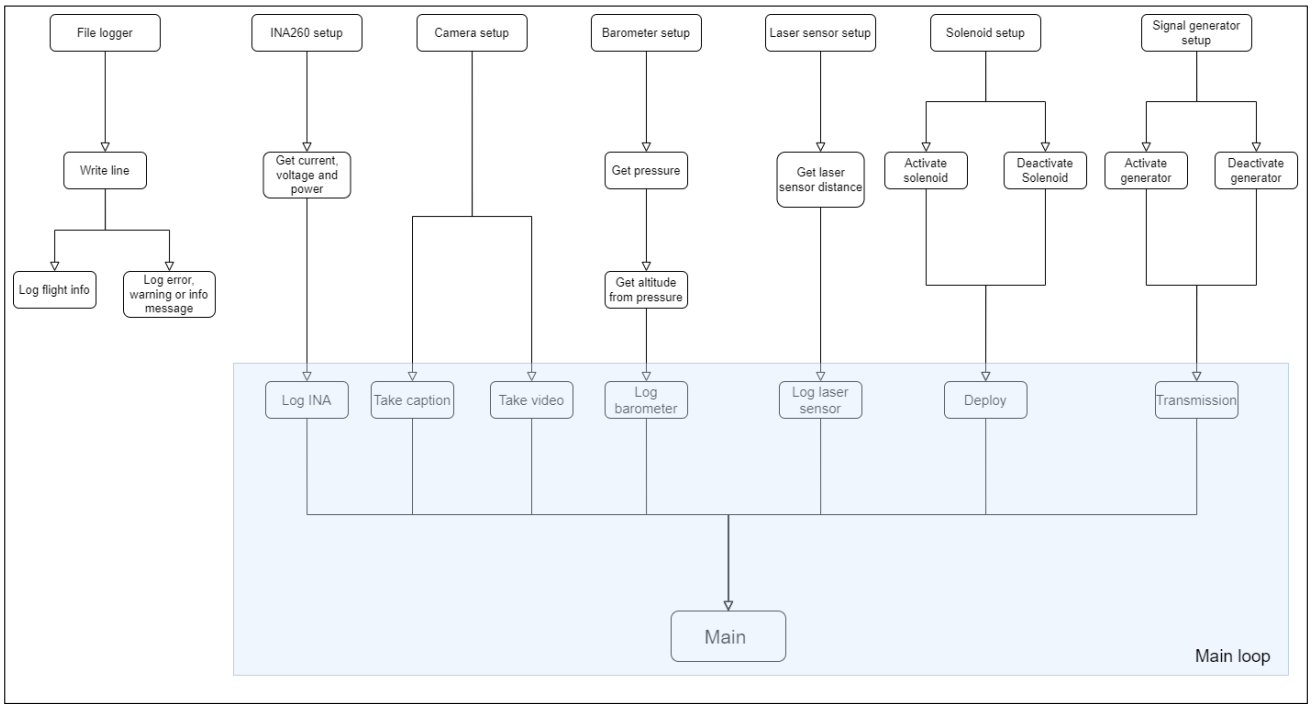

Figura 4.1: Diagrama de flujo del software (Elaboración propia)

En la figura 4.1, se representa el código embebido en el OBC del CubeSat. Cada una de las ramas del gráfico representa un módulo, desembocando en el Main, el cual será el que se ejecutará durante la misión. El Main, contará con un bucle principal "main loop", el cual llamará a todas las funciones que se muestran contenidas en el recuadro azul.

Las funciones llamadas en el bucle principal representarán el data-handling del código, pues estas ejecutarán las funciones que se encuentran por encima de ellas en base a los criterios definidos en los apartados anteriores.

Se muestra, además, un módulo de grabación de datos, "File logger", el cual contendrá las funciones de almacenamiento de datos y fallos o avisos, las cuales serán utilizadas en las funciones de data handling.

Como resultado, se obtienen unos módulos dedicados a cada componente, los cuales siguen la misma estructura:

- **Importación de objetos y librerías:** Al importar objetos y librerías, se establecen las dependencias del módulo y se garantiza que tiene acceso a las herramientas necesarias

para su funcionamiento.

- **Declaración de constantes y variables:** Se establece un espacio en memoria para almacenar datos y parámetros que serán utilizados a lo largo del módulo.
- **Setup del componente:** Se prepara el componente para su operación normal, garantizando que todas las configuraciones necesarias estén en su lugar. Es la interfaz entre hardware y software.
- **Funciones de lectura del componente o de accionamiento:** Son las encargadas de realizar y ejecutar las tareas del componente.
- **Funciones de data handling del componente:** Llama a las funciones de lectura o de accionamiento. Gestiona los datos obtenidos, lo que puede incluir su procesamiento, almacenamiento, transmisión o cualquier otra operación necesaria para el correcto funcionamiento del sistema.

El código será desarrollado en el lenguaje Python. Python es conocido por su sintaxis clara y legible, lo que facilita la escritura y comprensión del código. Además, cuenta con muchas bibliotecas que facilitan el setup de los componentes en el software.

En el anexo 6.1 se pueden encontrar todos los módulos que conforman el software de la misión. Además, se ha creado un repositorio en GitHub con el código desarrollado para el software, ubicado en la carpeta software_final:

TFGAMIN. (2023). CubeSat_SW [Software]. GitHub.
`https://github.com/TFGAMIN/CubeSat_SW/tree/main`

4.5. Test y Verificación del Código

El proceso de test y verificación es esencial para garantizar que el software desarrollado funcione correctamente y cumpla con los requisitos establecidos. En los anexos de la memoria, se han desarrollado dos versiones de código distintas.

Aunque a primera vista, el código en el anexo 6.2 parece similar al código mostrado en el anexo 6.1 , hay diferencias clave en su implementación. En lugar de interactuar directamente con los componentes del satélite, este código ha sido modificado para simular las lecturas y

comportamientos de los componentes. Esto se hace para permitir la prueba y verificación del software sin la necesidad de tener los componentes físicos presentes o en funcionamiento.

Al reemplazar las funciones de configuración (setup) de los componentes por simulaciones numéricas, se puede emular el comportamiento de los componentes sin necesidad de hardware real. La simulación se ha llevado a cabo a partir del uso de una función que devuelve 4 posibles factores. Dichos factores, se determinan por constantes temporales, de modo que, a lo largo de la misión, el valor simulado pasará por cada uno de ellos.

Como resultado, el código desarrollado para la validación del software es en esencia muy parecido a la versión expuesta en el apartado 4.4, pero sencillamente el setup de cada componente se ha sustutuido por la simulación numérica de sus lecturas, de modo que el código pueda ejecutarse y probarse en cualquier ordenador, sin necesidad de contar con la arquitectura de la aviónica desarrollada para el CubeSat.

La figua 4.2 muestra un diagrama de bloques a modo de explicación de la estructura del código desarrollado para la validación, de manera análoga a como se explicó la estructura del código principal en el apartado 4.4.

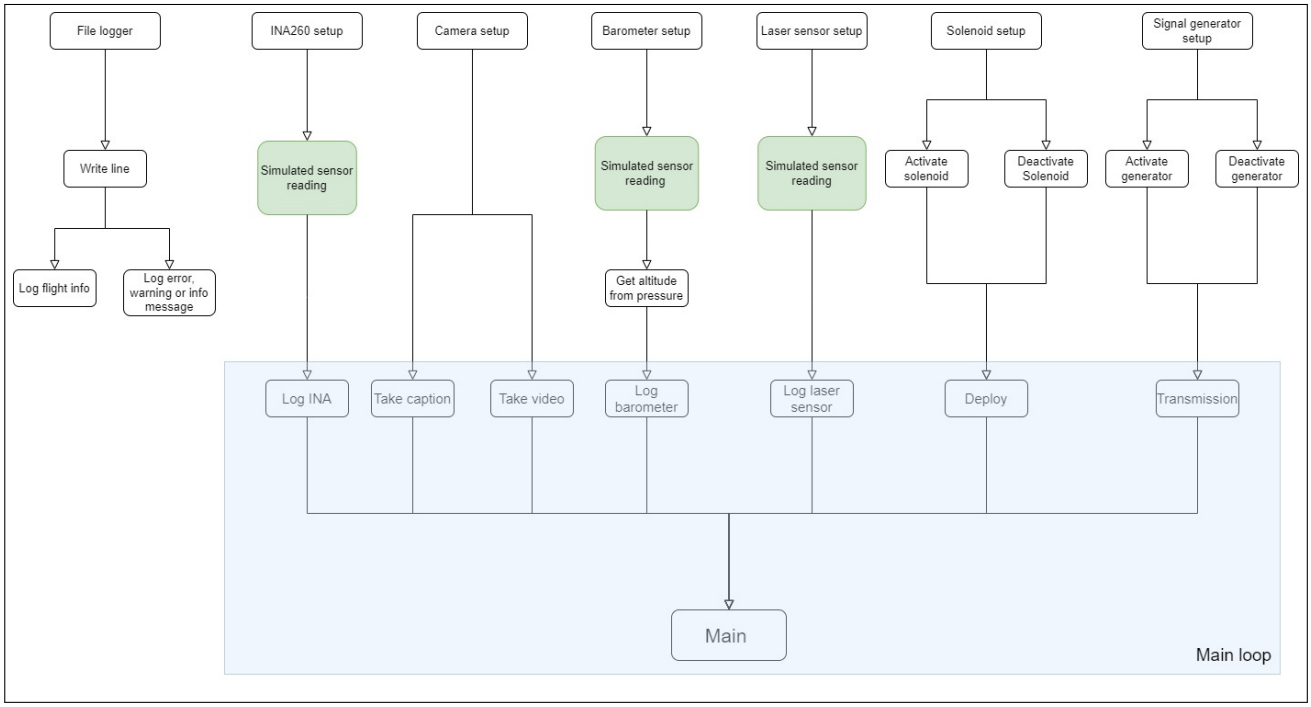

Figura 4.2: Diagrama de flujo del software de validación (Elaboración propia)

En la figura 4.2 , se muestran en color verde las funciones de setup que han sido cambiadas por funciones de simulación numérica. Además, las funciones que ejecutan una acción por parte de un componente tras una orden del OBC, han sido suprimidas para permitir la ejecución del código. Esto significa que, por ejemplo, si la función "deploy"del solenoide ejecutaba una función dentro de la librería del componente que lo permite desplegarse, y a su vez almacenaba la información de que se había desplegado, en esta versión del código preparada para la validación tan solo se almacena la información de que se ha desplegado (pues no hay un componente que ejecutar).

Una vez explicado cómo se va a realizar la lectura de los componentes ficticios, se puede proceder a llevar a cabo los test de verificación. En el caso del software desarrollado, se va a realizar una simulación de la misión de 4 horas, en las cuales se expondrá al código a distintas situaciones y se analizarán los resultados. La información de la misión almacenada a lo largo del test se genera al ejecutar el código en una carpeta log.

Para estudiar los resultados obtenidos, se elegirán datos almacenados en el documento de información de la misión en función del momento de la misión de interés. Se considerarán como resultados válidos, aquellos en los que se haya almacenado el resultado esperado.

En resumen, el test se realizará mediante la ejecución del código de validación durante 4 horas, donde los valores de la simulación numérica de lectura de los sensores, así como periodos de ejecución de las funciones, han sido elegidos de forma arbitraria para poder simular una gran variedad de situaciones en cada componente. El método de análisis del funcionamiento del código se realiza mediante la lectura posterior del documento de información de la misión. Además, se podrá comprobar el correcto paso por los distintos modos del satélite mediante la revisión de los componentes activados en cada momento.

A continuación, se presentan los resultados obtenidos en el test, así como las condiciónes en las que se ha elegido cada uno de los valores arbitrarios en cada módulo.

- **INA260:** Se han simulado 4 escenarios de lectura, en función del tiempo transcurrido desde el inicio de la misión. Los resultados son:
    - De 0 a 60 segundos. Se ha simulado una lectura incorrecta del componente, por lo que los valores de intensidad, voltaje y potencia se muestran bajo el carácter "$". El resultado almacenado a los 5 segundos de misión es: intensidad leída = $ V; voltaje leído = $ A; potencia leída = $ W.

- De 60 a 10800 segundos. Se ha simulado una lectura nula del componente, por lo que los valores de intensidad, voltaje y potencia son iguales a 0. El resultado almacenado a los 846 segundos de misión es: intensidad leída = 0 A; voltaje leído = 0 V; potencia leída = 0 W.
- De 10800 a 14400 segundos. Se ha simulado la trasmisión de energía, por lo que los valores de intensidad, voltaje y potencia no son nulos, sino de 2, 5 y 10 respectivamente. El resultado almacenado a los 14167 segundos de misión es: intensidad leída = 2 A; voltaje leído = 5 V; potencia leída = 10 W.
- De 14400 en adelante. La misión se había planificado de 4 horas, por lo que se asignó un valor de intensidad, voltaje y potencia nulos. El resultado almacenado a los 14433 segundos de misión es: intensidad leída = 0 A; voltaje leído = 0 V; potencia leída = 0 W.

- **Cámara:** Se ha configurado el módulo de cámara para tomar una imagen cada 10 segundos y un vídeo cada minuto, el cual mantiene una duración de 10 segundos. Se ha controlado la correcta ejecución del test a partir de la función de almacenamiento de información de toma de imagen y vídeo. Los resultados obtenidos en distintos momentos son:
  - Segundo 10. Imagen tomada.
  - Segundo 35. Imagen y vídeo no tomados.
  - Segundo 60. Imagen y vídeos tomados.
- **Altímetro:** Se han simulado 4 escenarios de lectura, en función del tiempo transcurrido desde el inicio de la misión. Además, el altímetro se ha configurado para realizar una lectura cada 60 segundos. Los resultados son:
  - De 0 a 60 segundos. Se ha simulado una lectura de presión de 1013 milibares. El resultado almacenado a los 0 segundos de misión es: presión leída = 1013 mbar; altura obtenida= 8,41 m.
  - De 60 a 8000 segundos. Se ha simulado una lectura de presión de 900 milibares. El resultado almacenado a los 120 segundos de misión es: presión leída = 900 mbar; altura obtenida= 996,16 m.
  - De 8000 a 13000 segundos. Se ha simulado una lectura de presión de 20 milibares. El resultado almacenado a los 12780 segundos de misión es: presión leída = 20 mbar; altura obtenida= 26388 m.

  - De 13000 segundos en adelante. Se ha simulado una lectura de presión fallida, mediante el símbolo "$". El resultado almacenado a los 14282 segundos de misión es: presión leída = $ mbar; altura obtenida= $ m.

- **Sensor laser:** Se han simulado 4 escenarios de lectura, en función del tiempo transcurrido desde el inicio de la misión. Además, el sensor de distancia se ha configurado para realizar una lectura cada 5 segundos. Los resultados son:

  - De 0 a 10 segundos. Se ha simulado una lectura de distancia fallida, mediante el símbolo "$". El resultado almacenado a los 5 segundos de misión es: distancia leída = $ mm.
  - De 10 a 10200 segundos. Se ha simulado una lectura de distancia de 0 milimetros. El resultado almacenado a los 3831 segundos de misión es: distancia leída = 0 mm.
  - De 10200 a 14400 segundos. Se ha simulado una lectura de distancia tras el despliegue estructural de 500 milimetros. El resultado almacenado a los 13642 segundos de misión es: distancia leída = 500 mm.
  - De 14400 segundos en adelante. Se ha simulado una lectura de presión nula. El resultado almacenado a los 14428 segundos de misión es: distancia leída = 0 mm.

- **Solenoide:** Se ha configurado el solenoide para desplegarse por primera vez a los 10200 segundos. Después, se vuelve a activar cada 5 minutos. Se ha controlado la correcta ejecución del test a partir de la variable "is_solenoid_activated". Los resultados obtenidos en distintos momentos son:

  - Segundo 10. Solenoide no activado.
  - Segundo 11844. Solenoide activado.
  - Segundo 11849. Solenoide no activado.

- **Generador de señal:** Se ha configurado el generador de señal para accionarse por primera vez a los 10800 segundos. Se mantendrá activado durante una hora, para después finalizar la transmisión. Se ha controlado la correcta ejecución del test a partir de la variable "is_generator_activated". Los resultados obtenidos en distintos momentos son:

  - Segundo 10. Generador no activado.
  - Segundo 10943. Generador activado.
  - Segundo 14428. Generador no activado.

Tras el análisis de los resultados obtenidos en el documento de información grabada de la misión en el test realizado, se consideran como aptas todas las consignas y requisitos determinados para el software desarrollado a nivel de data-handling, así como instrucciones a nivel de funcionalidad del código, como puede ser la periodicidad de activación de cada componente.

Como se indica en los requisitos del software, los siguientes requisitos han sido validados a partir de la inspección del documento de información de vuelo obtenido en el test: CSM-MSN-REQ-4; CSM-INA-REQ-1; CSM-ALT-REQ-1; CSM-CAM-REQ-4; CSM-LSR-REQ-1; CSM-GEN-REQ-3; CSM-SOL-REQ-4.

También, se ha podido revisar como se han dado los cambios de modo en el código a lo largo de la ejecución. A partir de los periodos de activación designados para cada módulo, los componentes han formado distintas secuencias de activación que han permitido el cambio de modo. En la tabla 4.3, se muestra en que momento exacto ha tenido lugar cada uno de los modos de la misión a lo largo del test.

Modo del satélite	Momento de ejecución en el test	Periodicidad
Modo de travesía	De 0 a 10200 segundos	-
Modo de despliegue	A partir de 10200 segundos	Cada 300 segundos
Modo de emisión	De 10800 a 14400 segundos	-
Modo de reposo	De 14400 segundos a fin de la ejecución	-

Tabla 4.3: Programación de modos del satélite durante el test

Cabe recalcar que, como se muestra en la tabla 4.3, el cambio de modo del satélite a lo largo del test se ha dado tras la ejecución o desactivación del solenoide y sistema de trasmisión. Este se debe a que, en esencia, las diferencias sustanciales entre los modos se dan cuando dichos componentes se activan, pues además de ser aquellos que acarrean un mayor consumo eléctrico, son los principales en la ejecución de la misión principal del Cubesat.

Se remarca que, en este test, no se han probado las funciones de error, pues la simulación

numérica simula la lectura, correcta o incorrecta, del componente en cuestión, pero no es capaz de simular el fallo en la ejecución de la función de la propia lectura (caso para el cual están diseñadas las funciones de guardado de avisos y fallos, pues se utilizarán cuando falle la conexión con un componente, no cuando este realice una lectura incorrecta).

Capítulo 5
Conclusiones

En todos los capítulos anteriores, se ha detallado el proceso de diseño de la aviónica del CubeSat para la misión de transmisión de microondas desde la definición del entorno y objetivos de la misión, hasta el desarrollo fonal del propio subsistema de aviónica.

A lo largo de la memoria, se han tomado distintas decisiones y enfoques ante los retos de ingeniería que se han presentado ante la realización de las distintas tareas del trabajo. En este capítulo, se explorarán las lecciones aprendidas a partir del desarrollo deel proyecto. Estas permitirán entender las dificultades inherentes al propio trabajo, así como las trabas que se han presentado previo a la obtención del resultado final.

Por otro lado, este capítulo culmina con la muestra de una serie de trabajos futuros hacia los que se podría enfocar una futura iteración de este proyecto. Esto implica aplicar modificaciones y mejoras en el diseño del subsistema de la aviónica, permitiendola dar el siguiente paso hacia el desarrollo de un satélite plenamente funcional que se enfrente al entorno espacial, con todos los retos a nivel de ingeniería que ello conlleva.

5.1. Lecciones Aprendidas

A lo largo del desarrollo del presente proyecto de fin de grado, se han adquirido diversas lecciones que pueden servir como guía para futuras investigaciones y desarrollos en el ámbito de la aviónica para CubeSats. A continuación se presenta como ha sido el desarroyo del trabajo, de modo que después se puedan especificar en concreto las principales lecciones aprendidas en cada apartado.

Por el lado del desarrollo del trabajo, se mencionó anteriormente que este partió de un proyecto universitario. En dicho proyecto universitario, un grupo de estudiantes se repartieron las tareas asignadas al diseño de un CubeSat a través de distintos subsistemas, tales como:

estructura, payload, aviónica o telecomunicaciones. El resultado final de dicho proyecto fue el correcto diseño y construcción de un CubeSat que fue lanzado mediante un globo para realizar una misión en unas condiciones muy similares a las panteadas en esta memoria.

En esencia, el Cubesat a desarrollar en el proyecto universitario tenía un fin parecido al propuesto en este Trabajo Fin de Grado, pero las complicaciones en el diseño y construcción imposibilitaron que finalmente dicho satélite fuese capaz de comprobar la tecnología de emisión de microondas. Como consecuencia, este trabajo investiga un diseño del subsistema de aviónica evolucionado, más complejo y capaz de proveer suminstro a un sistema de emisión de microondas modelizado.

Para el desarrollo de este trabajo, se ha partido de muchos requisitos heredados por la misión inicial del proyecto universitario, así como por todos los avances en cuanto a diseño estructral que tuvo el proyecto inicial, pues en este trabajo se han mantenido dichos diseños como premisa sobre la que se ha diseñado el subsistema de aviónica. En cuanto a la propia aviónica, se ha partido de los requisitos definidos en primera instancia para realzar una nueva elección de componentes con respecto al proyecto universitario, así como se han añadido nuevos componentes que brindaban la posibilidad de realizar nuevas funcionalidades por parte de la electrónica del satélite.

El mayor reto en cuanto al diseño del hardware del subsistema de aviónica se ha encontrado en la modelización del sistema de trasmisión de microondas. No se partía de ningún diseño ni estudio en dicho área por el proyecto universitario, pues la misión principal de la payload no se pudo cumplir. Los componentes elegidos para el sistema de trasmisión han servido como guía de referencia a un diseño próximo de dicho sistema, así como han permitido llevar a cabo el balance de potencias y estudio del coste energético del satélite, indispensables para poder finalizar el diseño de la aviónica.

Una vez obtenido el hardware del subsistema, el desarrollo del software se ha realizado a partir de los requisitos de software y con la intención de desarrollar un modelo capaz de ser sometido a pruebas de verificación, como se ha realizado con exito en el apartado 4.

Conocido el proceso de desarrollo del trabajo y el contexto en el que este se ha lllevado a cabo, se han extraído diversas lecciones relacionadas a cada apartado de este Trabajo Fin de Grado.

En primer lugar, se ha constatado la relevancia creciente de los CubeSats en la industria espacial. Estos pequeños satélites modulares y de bajo costo han permitido a diversas entidades, desde universidades hasta empresas, acceder a investigaciones desarrolladas en entornos espaciales. En relación con la transmisión de energía mediante microondas, se ha identificado como una de las tecnologías más prometedoras para la gestión energética en el espacio. Su capacidad para transferir energía de forma inalámbrica y eficiente entre sistemas espaciales puede revolucionar las operaciones y misiones futuras, ofreciendo soluciones más flexibles y eficientes.

El dimensionamiento del balance de potencias de la payload ha sido otro punto crucial. Se ha constatado la importancia de realizar un balance adecuado para garantizar que el sistema tenga suficiente energía para funcionar de manera óptima en todas las fases de la misión.

En relación con la elección de componentes, se han basado las decisiones tomadas en los requisitos y metas de la misión. Cada componente, desde el monitor de corriente hasta el solenoide que posibilita el despliegue estructural, ha sido seleccionado tras un análisis detallado de sus características y de cómo se alinean con los objetivos del proyecto. Se ha aprendido que una elección adecuada no solo garantiza la funcionalidad del sistema, sino que también puede optimizar su rendimiento y eficiencia.

Respecto al desarrollo de un software para este proyecto, la definición precisa de los modos de la misión es esencial para garantizar que el CubeSat pueda operar en diferentes escenarios y condiciones. Cada modo tiene sus propias necesidades y requisitos, y es imperativo que el software pueda adaptarse y responder adecuadamente a cada uno de ellos.

El análisis de riesgos y fallos ha permitido identificar elecciones de diseño del software que han permitido que se proteja ante potenciales amenazas para la misión. Es vital anticipar posibles fallos y, en este caso, la seguridad del sistema se ha antepuesto a un desempeño más complejo de la misión. En cuanto al desarrollo del código, este se ha presentado mediante un diseño robusto y modular. El código debe ser lo suficientemente flexible como para adaptarse a cambios y actualizaciones, pero también lo suficientemente sólido como para resistir fallos en lecturas, conexiones o incluso componentes de la aviónica.

Por último, el proceso de prueba y verificación del código ha resaltado la necesidad de asegurar que el software no solo cumpla con sus funciones designadas, sino que también sea seguro y confiable. Cada prueba ha proporcionado información valiosa sobre el comportamiento

del sistema y, dado que el resultado del test realizado ha sido exitoso, ha demostrado la viabilidad para su uso en la misión designada para el satélite.

En resumen, en este trabajo se ha abordado el proceso completo del diseño de la aviónica de un CubeSat, realizando un estudio transversal, que ha integrado todos los aspectos que giran en torno a la electrónica del satélite, como puede ser la misión, el vehículo lanzador o la propia carga de pago. Una vez definidos todos los aspectos que afectan al diseño del satélite, se ha procedido a dar forma a la estructura del hardware, para finalmente realizar la elección de cada uno de los componentes y desarrollar un código embebido al OBC para controlar toda la electrónica a lo largo de la misión.

5.2. Trabajos Futuros

El presente proyecto de fin de grado ha sentado las bases para el diseño de la aviónica de un CubeSat de 3U, centrado en un sistema de transmisión de energía. Sin embargo, como todo proyecto de investigación y desarrollo, abre diversas vías para futuras investigaciones y mejoras. A continuación, se presentan diversas tecnologías cuya implantación podría haría más viable el despliegue de la misión en un entorno espacial.

5.2.1. Sistema de Control de Actitud

Uno de los aspectos más notables a considerar en futuras iteraciones es la integración de un sistema de control de actitud. Aunque en esta versión del CubeSat no se ha puesto en órbita y, por lo tanto, no se ha implementado un control de actitud, es esencial para futuras versiones que busquen operar en el espacio.

Un sistema de control de actitud es esencial para la operación efectiva de un CubeSat en el espacio. Este sistema permite al satélite mantener y cambiar su orientación en el espacio. Se compone de sensores para determinar la orientación actual (como giróscopos y magnetómetros) y actuadores (como ruedas de reacción y magnetorquers) para realizar ajustes. El control de actitud es crucial para diversas funciones, como la orientación de antenas, paneles solares y la realización de maniobras espaciales.

Para integrar un sistema de control de actitud en el CubeSat, primero se deberían seleccionar sensores y actuadores adecuados que cumplan con las restricciones de tamaño, peso y consumo energético. Luego, se desarrollaría un software de control que procese la información de los sensores y emita comandos a los actuadores. Este sistema mejoraría la eficiencia del sistema de transmisión de energía al permitir una orientación óptima del satélite.

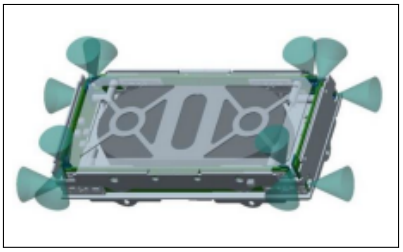

Figura 5.1: Sistema de control de actitud mediante propilsión de gas frío. Imagen de [24]

5.2.2. Integración de Placas Solares

En relación con la gestión energética, se ha identificado una oportunidad significativa en la utilización de placas solares. En este trabajo, las baterías se han utilizado principalmente en su modo de descarga.

Las placas solares son dispositivos que convierten la luz solar en electricidad. Su integración en el satélite sería fundamental para la generación de energía en el espacio. Estas placas capturan la radiación solar y la convierten en energía eléctrica, que luego se almacena en baterías para su uso continuo.

La instalación de placas solares requeriría un cuidadoso cálculo para maximizar la exposición solar sin afectar el balance térmico del satélite. La energía generada proporcionaría una fuente renovable y aumentaría significativamente la autonomía y eficiencia del CubeSat.

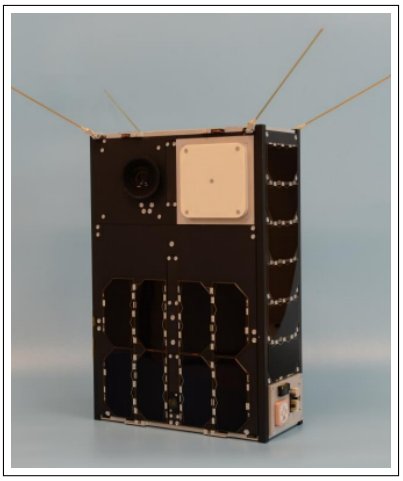

Figura 5.2: Cubesat con sistema de paneles solares integrado en la estructura. Imagen de [24]

5.2.3. Nuevos Componentes y Software

Como vía de mejora, se sugiere explorar otros métodos y tecnologías que puedan mejorar la eficiencia y funcionalidad del CubeSat. Esto incluye la investigación de nuevos componentes, software más avanzado y técnicas de optimización que puedan adaptarse a las cambiantes condiciones del espacio.

La selección de componentes avanzados y el desarrollo de software robusto deben considerar las limitaciones de espacio y energía del CubeSat. Los nuevos componentes podrían ofrecer mejoras en términos de precisión, durabilidad y consumo de energía, mientras que un software avanzado podría mejorar la gestión de recursos y la eficiencia operativa del satélite.

5.2.4. Avances en el Sistema de Transmisión de Energía

Por el lado del sistema de transmisión de energía, se sugiere la elección de componentes que se encuentren en el mercado o que, en su defecto, sean desarrollados en proyectos de investigación científica. Dichos componentes no están muy extendidos actualmente, pero cada año aparecen nuevos estudios y diseños que permiten que la tecnología siga avanzando.

En cuanto a la tecnología de transmisión, es esencial considerar la retroalimentación y los datos obtenidos de esta versión del CubeSat para informar y guiar futuros desarrollos. Cada lección aprendida, cada desafío enfrentado y cada éxito logrado proporciona información valiosa que puede ser utilizada para mejorar y perfeccionar futuras iteraciones del proyecto, con el fin

de alcanzar una transmisión de energía eficiente y viable a nivel energético.

Bibliografía

[1] nanosats.eu, "Base de datos de nanosats.eu," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.nanosats.eu/

[2] European Space Agency (ESA), "Race double cubesat mission," 2019, Último acceso: 2023-11-02. [Online]. Available: https://www.esa.int/ESA_Multimedia/Images/2019/06/RACE_double_CubeSat_mission

[3] Tiempo.com, "El modelo atmósfera teórica isa para estudiar el comportamiento de la atmósfera real," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.tiempo.com/noticias/ciencia/el-modelo-atmosfera-teorica-isa-para-estudiar-el-comportamiento-de-la-atmosfera-real.html

[4] NanoAvionics, "Cubesat 101: The comprehensive guide to understanding satellite technology," 2023, Último acceso: 2023-11-02. [Online]. Available: https://nanoavionics.com/blog/cubesat-101-the-comprehensive-guide-to-understanding-satellite-technology/

[5] B. Perdiguero García, "Diseño y simulación de un modelo físico-numérico para un globo estratosférico," 2023.

[6] Dykbcells, "Generador de señal de frecuencia de puntos para microondas," 2023, extraído del catálogo de componentes de Dykbcells.

[7] TZT, "Amplificador de potencia de microondas modelo sbb5089+sza2044," 2023, extraído del catálogo de productos de TZT.

[8] FARAJIAJ, "Antena wifi de doble frecuencia 2,4g/5,8g," 2023, extraído del catálogo de productos de FARAJIAJ.

[9] D. Wang and R. Negra, "Design of a rectifier for 2.45 ghz wireless power transmission," 2012.

[10] Electrokit, "Sensor de corriente ina260 36v 15a i2c," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.electrokit.com/en/product/stromsensor-ina260-36v-15a-i2c/

[11] Raspberry Pi, "Raspberry pi 4 model b," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[12] ——, "Tarjeta sd de memoria 64 gb," 2023, extraído del catálogo de productos de Raspberry Pi.

[13] STMicroelectronics, "Vl53l0x datasheet en alldatasheet," 2023, Último acceso: 2023-11-02. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/948120/STMICROELECTRONICS/VL53L0X.html

[14] ——, "Sensor vl53l0x de tof," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html

[15] Adafruit, "Barómetro del catálogo de adafruit," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.adafruit.com/product/3966

[16] Raspberry Pi, "Cámara pi noir v2 - producto en raspberry pi," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.raspberrypi.com/products/pi-noir-camera-v2/

[17] Electrotec, "Esquema de funcionamiento de un relé," 2023, Último acceso: 2023-11-02. [Online]. Available: https://electrotec.pe/blog/FuncionamientoReleNEW

[18] Joy-IT. (n.d.) Ky-019. [Online]. Available: https://datasheetspdf.com/pdf-file/1402030/Joy-IT/KY-019/1

[19] Farnell, "Imagen del diodo rectificador 1n4007," 2023, Último acceso: 2023-11-02. [Online]. Available: https://es.farnell.com/diodes-inc/1n4007-t/diodo-rectificador-1000v-1a-do/dp/1843698

[20] Heschen, "Heschen solenoide electromagnet hs-1264b," 2023, Último acceso: 2023-11-02. [Online]. Available: https://heschen.com/products/hechen-solenoid-electromagnet-hs-1264b

[21] V. Knap, L. K. Vestergaard, and D.-I. Stroe, "A review of battery technology in cubesats and small satellite solutions," GomSpace A/S, Tech. Rep., 2020.

[22] ALEAIVY, "Batería de iones de litio 12v 30ah," 2023, extraído del catálogo de productos de ALEAIVY.

[23] AZ Delivery, "Regulador de voltaje lm2596s," 2023, Último acceso: 2023-11-02. [Online]. Available: https://www.az-delivery.de/es/products/lm2596s-dc-dc-step-down-modul-1

[24] R. Walker, "Cubesats: State-of-the-art and future potential for small low-cost science missions," 2018, presentado en ESA SCI Science Workshop Akersloot, 6 de noviembre de 2018.

[25] NASA, "8.0 small spacecraft avionics," 2023, Último acceso: 2023-11-27. [Online]. Available: https://www.nasa.gov/smallsat-institute/sst-soa/small-spacecraft-avionics/

[26] W. C. Brown, "The history of power transmission by radio waves," *IEEE Transactions on Microwave Theory and Techniques*, 1984.

[27] J. O. McSpadden and J. C. Mankins, "Space solar power programs and microwave wireless power transmission technology," *IEEE Microwave Magazine*, 1998.

[28] N. Shinohara, "Power without wires," *IEEE Microwave Magazine*, 2011.

[29] P. Koert, J. Cha, and Y. H. Suh, "A compact rectenna for wireless power transmission at microwave frequencies," in *IEEE MTT-S International Microwave Symposium Digest*, 1995.

[30] M. H. Rashid, *Power electronics handbook*. Elsevier, 2016.

[31] R. W. Erickson and D. Maksimovic, *Fundamentals of power electronics*. Springer Science & Business Media, 2001.

[32] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power electronics: converters, applications, and design*. John Wiley & Sons, 2003.

[33] B. K. Bose, *Modern power electronics and AC drives*. Prentice Hall, 2018.

[34] J. H. Williams, *The electronics handbook*. CRC press, 1998.

[35] P. Horowitz and W. Hill, *The art of electronics*. Cambridge university press, 2015.

[36] J. Millman and C. C. Halkias, *Integrated electronics: analog and digital circuits and systems*. McGraw-Hill, 1987.

[37] A. S. Sedra and K. C. Smith, *Microelectronic circuits*. Oxford University Press, 1997.

[38] E. M. Purcell and D. J. Morin, *Electricity and magnetism*. Cambridge University Press, 2013.

[39] D. J. Griffiths, *Introduction to electrodynamics*. Pearson, 2013.

[40] P. A. Tipler and G. Mosca, *Physics for scientists and engineers*. Macmillan, 2008.

[41] J. D. Jackson, *Classical electrodynamics*. John Wiley & Sons, 1999.

[42] J. Gomis-Tena Dolz, E. Figueres Amorós, and G. Garcerá Sanfeliú, *Electrónica industrial. Problemas resueltos*. Editorial Universitat Politècnica de València, 2019.

[43] K. Woellert, P. Ehrenfreund, A. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations," *Adv. Space Res.*, 2011.

[44] T. Villela, C. Costa, A. Brandão, F. Bueno, and R. Leonardi, "Towards the thousandth cubesat: A statistical overview," *Int. J. Aerosp. Eng.*, 2019.

[45] A. Camps, *Nanosatellites and Applications to Commercial and Scientific Missions*. IntechOpen, 2019.

[46] N. Saeed, A. Elzanaty, H. Almorad, H. Dahrouj, T. Al-Naffouri, and M.-S. Alouini, "Cubesat communications: Recent advances and future challenges," *IEEE Commun. Surv.*, 2020.

[47] California Polytechnic State University, *CubeSat Design Specification*, 2015. [Online]. Available: https://www.cubesat.org/s/cds_rev13_final2.pdf

[48] Electrotec. (n.d.) Funcionamiento del relé electromecánico. [Online]. Available: https://electrotec.pe/blog/FuncionamientoReleNEW

[49] Pan Jit International Inc. (n.d.) 1n4007 datasheet. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/14624/PANJIT/1N4007.html

[50] Transmotec. (n.d.) Datasheet k1037l. [Online]. Available: https://www.transmotec.es/Download/Datasheets/Transmotec-Datasheet-K1037L.pdf

[51] Johnson Electric. (n.d.) Johnson dc motors. [Online]. Available: https://pdf.directindustry.es/pdf/johnson-electric/johnson-dc-motors/665-6344.html

[52] Guardian Electric. (n.d.) Guardian electric a420-066092-00. [Online]. Available: https://www.guardiancatalog.com/Guardian-Electric-TP6X12-I-12D-p/a420-066092-00.htm

[53] Hobbytronics. (n.d.) Jf-0530b. [Online]. Available: https://www.hobbytronics.co.za/Content/external/1274/D2512640695.pdf

[54] Mini-Circuits. (n.d.) Zx60-p103ln datasheet. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1029136/MINI/ZX60-P103LN.html

Capítulo 6
Anexos

Se presentan a continuación los códigos desarrollados para el proyecto, explicados a lo largo del capítulo 4. Dichos códigos, a su vez, han sido implementados en el repositorio público de GitHub:

TFGAMIN. (2023). CubeSat_SW [Software]. GitHub.
`https://github.com/TFGAMIN/CubeSat_SW/tree/main`

Como anexos al proyecto, se encuentran las dos versiones del código. En primer lugar, el anexo 6.1 mostrará los códigos a implementar en ordenador de abordo del satélite, mientras que el anexo 6.2 presenta la variación del código realizada para realizar los test y validación del código.

6.1. Códigos del Software

A continuación, se muestran los distintos documentos con extensión .py que conforman todos los módulos del software de la misión.

6.1.1. bmp388.py

```
from time import time
import numpy as np
from file_logger import log_flight_info , log_error_msg
from bmp388 import BMP388

try:
    from smbus2 import SMBus
except ImportError:
    from smbus import SMBus
```

```
10
11 # Initialise the BMP388
12 bus = SMBus(1)
13 bmp388 = BMP388(i2c_dev=bus)
14
15
16 ## time constants for each element
17 time_counter_barometer = 0
18 TIME_STEP_BAROMETER = 1*60
19
20 ## log key
21 BAROMETER_LK = 5
22
23 ## function to get the data from the barometer
24 def get_pressure():
25     try:
26         pressure = bmp388.get_pressure()
27         return pressure
28     except:
29         return "$"
30
31
32 PRESSURE_11 = 22.65
33 PRESSURE_25 = 2.488
34
35
36
37
38 def get_altitude_from_pressure():
39
40     try:
41         pressure = get_pressure()
42         #we divide by 10 to go from millibar to kilopascal
43         pressure /= 10
44
45
46         if (pressure > PRESSURE_11):
47             T = (pressure/101.29)**(0.19026)*288.08 - 273.1
48             h = (T - 15.04)/-6.49E-3
49
50         elif ((pressure < PRESSURE_11) and (pressure > PRESSURE_25)):
51             T = -56.46
52             h = (np.log(pressure/22.65) - 1.73)/-1.57E-4
53
54         else:
55             T = (pressure/2.488)**(-0.087812)*216.6 - 273.1
56             h = (T + 131.21)/2.99E-3
57
58         #we get the height in meters
59         return h
60
61     except:
62         pressure= "$"
63         T= "$"
64         h= "$"
65         return "$"
66
67 def log_barometer(t0: float):
```

```
68
69      global time_counter_barometer, TIME_STEP_BAROMETER, BAROMETER_LK
70
71      t = time() - t0
72
73      ## barometer data
74      if t > time_counter_barometer:
75
76          try:
77              altitude = get_altitude_from_pressure()
78              pressure = get_pressure()
79          except:
80              altitude = "$"
81              pressure = "$"
82              log_error_msg("failed_barometer_reading")
83
84          info_arr = [BAROMETER_LK, altitude, pressure, t]
85
86          log_flight_info (info_arr)
87
88          time_counter_barometer += TIME_STEP_BAROMETER
```

6.1.2. camera.py

```
1  from time import time
2  from picamera import PiCamera
3  from file_logger import log_flight_info , log_error_msg
4
5
6  ## time constants
7  time_counter_image = 0
8  TIME_STEP_IMAGE= 20*60
9
10 time_counter_video = 0
11 TIME_STEP_VIDEO = 21*60
12 VIDEO_DURATION = 10
13
14 ## other constants
15 image_counter = 0
16 video_counter = 0
17
18 IMAGE_RESOLUTION_H = 1280
19 IMAGE_RESOLUTION_V = 720
20
21 IMAGE_LK = 2
22 VIDEO_LK = 3
23
24
25 ## camera setup
26 camera = PiCamera()
27 camera.resolution = (IMAGE_RESOLUTION_H, IMAGE_RESOLUTION_V)
28
29
30 def take_caption(t0: float):
31
32
```

```
33      global time_counter_image, TIME_STEP_IMAGE, image_counter
34
35      file_name = "images/img_" + str(image_counter) + ".jpg"
36
37      current_time = time() - t0
38
39      if current_time < time_counter_image:
40          return
41
42      try:
43          camera.capture(file_name)
44
45          log_image(t0)
46
47          time_counter_image += TIME_STEP_IMAGE
48
49          image_counter += 1
50
51      except:
52          log_error_msg("failed_camera_caption")
53          return
54
55
56  def take_video(t0: float):
57
58
59      global time_counter_video, TIME_STEP_VIDEO, video_counter
60
61      file_name = "videos/video_" + str(video_counter) + ".h264"
62
63      current_time = time() - t0
64
65      if current_time < time_counter_video:
66          return
67
68      try:
69          camera.start_recording(file_name)
70          camera.wait_recording(VIDEO_DURATION)
71          camera.stop_recording()
72
73          log_video(t0)
74
75          time_counter_video+= TIME_STEP_VIDEO
76
77          video_counter += 1
78
79      except:
80          log_error_msg("failed_camera_video")
81          return
82
83  def log_image(t0: float):
84
85      t = time() - t0
86
87      info_arr = [IMAGE_LK, "caption", t]
88
89      log_flight_info (info_arr)
90
```

```
91
92
93 def log_video(t0: float):
94
95     t = time() - t0
96
97     info_arr = [VIDEO_LK, "video", t]
98
99     log_flight_info (info_arr)
```

6.1.3. file_logger.py

```
1  from time import time
2  from os import makedirs,path
3
4  #log errors, warnings or debug msgs here, useful during testing and mission analisys
5  ERROR_FILE="log/err.txt"
6  #csv file for flight data, useful for analysis afterwards
7  FLIGHT_INFO_FILE="log/flight_info.csv"
8
9  #writes a line to a file with a timestamp to use only in this module
10 def write_line(file,line):
11     makedirs(path.dirname(file), exist_ok=True)
12     with open(file,'a') as f:
13         print("{t:.6f},{data}".format(t=time(),data=line),file=f)
14
15 #input a list to this function, it will write it to the TELEMETRY_FILE in csv format
16 def log_flight_info (list):
17     write_line(FLIGHT_INFO_FILE,",".join(map(str,list)))
18
19 #log error msg to ERROR_FILE
20 def log_error_msg(error_str):
21     write_line(ERROR_FILE,"ERROR:   {s}".format(s=error_str))
22
23 #log warning msg to ERROR_FILE
24 def log_warning_msg(warning_str):
25     write_line(ERROR_FILE,"WARNING: {s}".format(s=warning_str))
26
27 #log info msg to ERROR_FILE
28 def log_info_msg(warning_str):
29     write_line(ERROR_FILE,"INFO:    {s}".format(s=warning_str))
```

6.1.4. ina260.py

```
1  from time import time
2
3  from file_logger import log_flight_info, log_error_msg
4
5  import board
6  import busio
7  import adafruit_ina260
8
9
10 ## time constants
```

```
time_counter_ina = 0


## log keys
INA_LK = 1

## initialize i2c
i2c = busio.I2C(board.SCL, board.SDA)

## ina (PAYLOAD) setup
ina1_adress = 0x40
ina1 = adafruit_ina260.INA260(i2c, i2c_address=ina1_adress)



def get_current_voltage_power_1():
    try:
        return ina1.current, ina1.voltage, ina1.power
    except:
        return "$", "$", "$"




def log_ina(t0: float):

    global time_counter_ina,  INA_LK

    t = time() - t0

    ## ina_data
    if t > time_counter_ina:

        t = time() - t0

        try:
            current, voltage, power = get_current_voltage_power_ina_1()
        except:
            current, voltage, power = "$", "$", "$"
            log_error_msg("failed_INA260_reading")


        info_arr = [INA_LK, current, voltage, power, t]

        log_flight_info (info_arr)
```

6.1.5. laser_sensor.py

```
from time import time

from file_logger import log_flight_info , log_error_msg

import VL53L0X


## time constants
```

```
time_counter_laser = 0
TIME_STEP_LASER = 5

## log key
LASER_LK = 4

## other constants
DISTANCE_SEPARATION = 500 # mm



## initialize sensor obj
SENSOR_I2C_ADDRESS = 0x29
sensor1=VL53L0X.VL53L0X(i2c_bus=1, i2c_address=SENSOR_I2C_ADDRESS)


def get_laser_sensor_distance():

    ## obtain distance
    try:
        distance = sensor1.get_distance()
    except:
        distance = "$"

    return distance


def is_separated():
    try:
        distance = get_laser_sensor_distance()

        if distance >= DISTANCE_SEPARATION:
            return True
        else:
            return False

    except:
        return "$"


def log_laser_sensor(t0: float):

    global time_counter_laser, TIME_STEP_LASER, LASER_LK

    t = time() - t0

    ## laser sensors data
    if t > time_counter_laser:
        try:
            distance = get_laser_sensor_distance()
        except:
            distance= "$"
            log_error_msg("failed_laser_sensor_reading")

        time_counter_laser += TIME_STEP_LASER

        if is_separated()== True:
            info_arr = [LASER_LK, distance,"separated", t]
```

```
67                 log_flight_info (info_arr)
68             else:
69                 info_arr = [LASER_LK, distance,"separation_not_confirmed", t]
70                 log_flight_info (info_arr)
```

6.1.6. main.py

```
1  from time import time, sleep
2  from ina260 import log_ina
3  from bmp388 import log_barometer
4  from laser_sensor import log_laser_sensor
5  from camera import take_caption, take_video
6  from solenoid import deploy
7  from signal_generator import transmission
8
9
10 ## get initial time
11 t0 = time()
12 tstep = 5
13
14 while True:
15
16     log_ina(t0)
17
18     take_caption(t0)
19
20     take_video(t0)
21
22     log_laser_sensor (t0)
23
24     log_barometer(t0)
25
26     deploy(t0)
27
28     transmission(t0)
29
30     sleep(tstep)
```

6.1.7. signal_generator.py

```
1  import RPi.GPIO as GPIO
2  from time import time
3  from file_logger import log_flight_info
4
5
6  # constants
7  time_to_transmit = 180*60
8  TRANSMISSION_DURATION = 60*60
9  is_generator_activated = False
10 transmission_done= False
11
12 # log key
13 GENERATOR_LK = 7
14
```

```
15
16 # setup
17 PIN_GENERATOR = 25
18 GPIO.setmode(GPIO.BOARD)
19 GPIO.setup(PIN_GENERATOR, GPIO.OUT)
20 GPIO.output(PIN_GENERATOR, GPIO.LOW)
21
22
23 def activate_generator():
24     GPIO.output(PIN_GENERATOR, True)
25
26
27 def deactivate_generator():
28     GPIO.output(PIN_GENERATOR, False)
29
30
31 def transmission(t0: float):
32
33     t = time() - t0
34
35     if ((t > time_to_transmit) and (is_generator_activated == False)):
36
37         activate_generator()
38         is_generator_activated = True
39
40     elif ((t > (time_to_transmit + TRANSMISSION_DURATION)) and (is_generator_activated == True)):
41
42         deactivate_generator()
43         is_generator_activated = False
44
45
46     if (is_generator_activated == True):
47
48         info_arr = [GENERATOR_LK, "transmission_activated", t]
49
50         log_flight_info(info_arr)
51
52     else:
53
54         info_arr = [GENERATOR_LK, "transmission_not_activated", t]
55
56         log_flight_info(info_arr)
```

6.1.8. solenoid.py

```
1 import RPi.GPIO as GPIO
2 from time import time
3
4
5
6 # constants and variables
7 time_to_deploy = 170*60
8 DEPLOY_PULSE_DURATION = 2
9 is_solenoid_activated = False
10 TIME_TO_CHECK_SEPARATION = 5*60
11
```

```
12 # log key
13 MECHANISM_LK = 6
14
15
16 # setup
17 PIN_SOLENOID = 9
18 GPIO.setmode(GPIO.BOARD)
19 GPIO.setup(PIN_SOLENOID, GPIO.OUT)
20 GPIO.output(PIN_SOLENOID, GPIO.LOW)
21
22 def activate_solenoid():
23     GPIO.output(PIN_SOLENOID, True)
24
25
26 def deactivate_solenoid():
27     GPIO.output(PIN_SOLENOID, False)
28
29
30 def deploy(t0: float):
31
32     global time_to_deploy, DEPLOY_PULSE_DURATION, is_solenoid_activated,
       TIME_TO_CHECK_SEPARATION
33
34     t = time() - t0
35
36     if ((t > time_to_deploy) and (is_solenoid_activated == False)):
37
38         activate_solenoid()
39         is_solenoid_activated = True
40
41     elif ((t > (time_to_deploy + DEPLOY_PULSE_DURATION)) and (is_solenoid_activated == True)):
42
43         deactivate_solenoid()
44         is_solenoid_activated = False
45         time_to_deploy = t + TIME_TO_CHECK_SEPARATION
46
47     if (is_solenoid_activated == True):
48
49         info_arr = [MECHANISM_LK, "solenoid_activated", t]
50
51         log_flight_info(info_arr)
52
53     else:
54
55         info_arr = [MECHANISM_LK, "solenoid_not_activated", t]
56
57         log_flight_info(info_arr)
```

6.2. Códigos de Validación

A continuación, se muestran los distintos documentos con extensión .py que conforman todos los módulos del software de validación y test de la misión.

6.2.1. bmp388.py

```
from time import time
import numpy as np
from file_logger import log_flight_info , log_error_msg




## time constants for each element
time_counter_barometer = 0
TIME_STEP_BAROMETER = 1*60
T1 = 60
T2 = 8000
T3 = 13000

## log key
BAROMETER_LK = 5

def simulated_sensor_reading(t0: float):

    simulation = time() - t0

    if 0 <= simulation < T1:
        return 1013

    elif T1<= simulation < T2:
        return 900

    elif T2 <= simulation < T3:
        return 20

    else:
        return "$"


PRESSURE_11 = 22.65
PRESSURE_25 = 2.488




def get_altitude_from_pressure(t0: float):

    t = time() - t0

    try:
        pressure = simulated_sensor_reading(t0)
        #we divide by 10 to go from millibar to kilopascal
        pressure /= 10


        if (pressure > PRESSURE_11):
            T = (pressure/101.29)**(0.19026)*288.08 - 273.1
            h = (T - 15.04)/-6.49E-3

        elif ((pressure < PRESSURE_11) and (pressure > PRESSURE_25)):
```

```
            T = -56.46
            h = (np.log(pressure/22.65) - 1.73)/-1.57E-4

        else:
            T = (pressure/2.488)**(-0.087812)*216.6 - 273.1
            h = (T + 131.21)/2.99E-3

        #we get the height in meters
        return h

    except:
        pressure= "$"
        T= "$"
        h= "$"
        return "$"

def log_barometer(t0: float):

    global time_counter_barometer, TIME_STEP_BAROMETER, BAROMETER_LK

    t = time() - t0

    ## barometer data
    if t > time_counter_barometer:

        try:
            altitude = get_altitude_from_pressure(t0)
            pressure = simulated_sensor_reading(t0)
        except:
            altitude = "$"
            pressure = "$"
            log_error_msg("failed_barometer_reading")

        info_arr = [BAROMETER_LK, altitude, pressure, t]

        log_flight_info (info_arr)

        time_counter_barometer += TIME_STEP_BAROMETER
```

6.2.2. camera.py

```
from time import time
from file_logger import log_flight_info , log_error_msg


## time constants
time_counter_image = 0
TIME_STEP_IMAGE= 10

time_counter_video = 0
TIME_STEP_VIDEO = 1*60
VIDEO_DURATION = 10

## other constants
image_counter = 0
video_counter = 0
```

```
16
17
18 IMAGE_LK = 2
19 VIDEO_LK = 3
20
21
22 def take_caption(t0: float):
23
24
25     global time_counter_image, TIME_STEP_IMAGE, image_counter
26
27     current_time = time() - t0
28
29     if current_time < time_counter_image:
30         return
31
32     try:
33
34         log_image(t0)
35
36         time_counter_image += TIME_STEP_IMAGE
37
38         image_counter += 1
39
40     except:
41         log_error_msg("failed_camera_caption")
42         return
43
44
45 def take_video(t0: float):
46
47
48     global time_counter_video, TIME_STEP_VIDEO, video_counter
49
50     current_time = time() - t0
51
52     if current_time < time_counter_video:
53         return
54
55     try:
56         log_video(t0)
57
58         time_counter_video+= TIME_STEP_VIDEO
59
60         video_counter += 1
61
62     except:
63         log_error_msg("failed_camera_video")
64         return
65
66 def log_image(t0: float):
67
68     t = time() - t0
69
70     info_arr = [IMAGE_LK, "caption", t]
71
72     log_flight_info (info_arr)
73
```

```
74
75
76 def log_video(t0: float):
77
78     t = time() - t0
79
80     info_arr = [VIDEO_LK, "video", t]
81
82     log_flight_info (info_arr)
```

6.2.3. file_logger.py

```
1  from time import time
2  from os import makedirs,path
3
4  #log errors, warnings or debug msgs here, useful during testing and mission analisys
5  ERROR_FILE="log/err.txt"
6  #csv file for flight data, useful for analysis afterwards
7  FLIGHT_INFO_FILE="log/flight_info.csv"
8
9  #writes a line to a file with a timestamp to use only in this module
10 def write_line(file,line):
11     makedirs(path.dirname(file), exist_ok=True)
12     with open(file,'a') as f:
13         print("{t:.6f},{data}".format(t=time(),data=line),file=f)
14
15 #input a list to this function, it will write it to the TELEMETRY_FILE in csv format
16 def log_flight_info (list):
17     write_line(FLIGHT_INFO_FILE,",".join(map(str,list)))
18
19 #log error msg to ERROR_FILE
20 def log_error_msg(error_str):
21     write_line(ERROR_FILE,"ERROR:   {s}".format(s=error_str))
22
23 #log warning msg to ERROR_FILE
24 def log_warning_msg(warning_str):
25     write_line(ERROR_FILE,"WARNING: {s}".format(s=warning_str))
26
27 #log info msg to ERROR_FILE
28 def log_info_msg(warning_str):
29     write_line(ERROR_FILE,"INFO:    {s}".format(s=warning_str))
```

6.2.4. ina260.py

```
1  from time import time
2
3  from file_logger import log_flight_info, log_error_msg
4
5
6
7
8  ## time constants
9  time_counter_ina = 0
10 TIME_STEP_INA = 1 #For the moment, don't needed
```

```
T1 = 60
T2 = 180*60
T3 = 240*60

## log keys
INA_LK = 1



def simulated_sensor_reading(t0: float):

    simulation = time() - t0

    if 0 <= simulation < T1:
        return "$", "$", "$"

    elif T1 <= simulation < T2:
        return 0,0,0

    elif T2 <= simulation < T3:
        return 2,5,10

    else:
        return 0,0,0




def log_ina(t0: float):

    global time_counter_ina, TIME_STEP_INA, INA_LK

    t = time() - t0

    ## ina_data
    if t > time_counter_ina:

        t = time() - t0

        try:
            current, voltage, power = simulated_sensor_reading(t0)
        except:
            current, voltage, power = "$", "$", "$"
            log_error_msg("failed_INA260_reading")


        info_arr = [INA_LK, current, voltage, power, t]

        log_flight_info (info_arr)
```

6.2.5. laser_sensor.py

```
from time import time

from file_logger import log_flight_info , log_error_msg

```

```
5
6
7
8  ## time constants
9  time_counter_laser = 0
10 TIME_STEP_LASER = 5
11 T1 = 10
12 T2 = 170*60
13 T3 = 240*60
14
15 ## log key
16 LASER_LK = 4
17
18 ## other constants
19 DISTANCE_SEPARATION = 500 # mm
20
21 def simulated_sensor_reading(t0: float):
22
23     simulation = time() - t0
24
25     if 0 <= simulation < T1:
26         return "$"
27
28     elif T1<= simulation < T2:
29         return 0
30
31     elif T2 <= simulation < T3:
32         return 500
33
34     else:
35         return 0
36
37
38
39
40
41 def is_separated(t0):
42     try:
43         distance = simulated_sensor_reading(t0)
44
45         if distance >= DISTANCE_SEPARATION:
46             return True
47         else:
48             return False
49
50     except:
51         return "$"
52
53
54 def log_laser_sensor (t0: float):
55
56     global time_counter_laser, TIME_STEP_LASER, LASER_LK
57
58     t = time() - t0
59
60     ## laser sensors data
61     if t > time_counter_laser:
62         try:
```

```
63              distance = simulated_sensor_reading(t0)
64          except:
65              distance= "$"
66              log_error_msg(" failed_laser_sensor_reading ")
67
68          time_counter_laser += TIME_STEP_LASER
69
70          if is_separated(t0)== True:
71              info_arr = [LASER_LK, distance,"separated", t]
72              log_flight_info (info_arr)
73          else:
74              info_arr = [LASER_LK, distance,"separation_not_confirmed", t]
75              log_flight_info (info_arr)
```

6.2.6. main.py

```
1  from time import time, sleep
2  from ina260 import log_ina
3  from bmp388 import log_barometer
4  from laser_sensor import log_laser_sensor
5  from camera import take_caption, take_video
6  from solenoid import deploy
7  from signal_generator import transmission
8
9
10 ## get initial time
11 t0 = time()
12 tstep = 5
13
14 while True:
15
16     log_ina(t0)
17
18     take_caption(t0)
19
20     take_video(t0)
21
22     log_laser_sensor (t0)
23
24     log_barometer(t0)
25
26     deploy(t0)
27
28     transmission(t0)
29
30     sleep(tstep)
```

6.2.7. signal_generator.py

```
1  from time import time
2  from file_logger import log_flight_info
3
4  # constants
5  time_to_transmit = 180*60
```

```
TRANSMISSION_DURATION = 60*60
is_generator_activated = False
transmission_done= False


# log key
GENERATOR_LK = 7



def transmission(t0: float):
    global is_generator_activated, transmission_done

    t = time() - t0

    if not transmission_done:
        if ((t > time_to_transmit) and (is_generator_activated == False)):
            is_generator_activated = True

        elif ((t > (time_to_transmit + TRANSMISSION_DURATION)) and (is_generator_activated ==
    True)):
            is_generator_activated = False
            transmission_done = True


    if (is_generator_activated == True):

        info_arr = [GENERATOR_LK, "transmission_activated", t]

        log_flight_info(info_arr)

    else:

        info_arr = [GENERATOR_LK, "transmission_not_activated", t]

        log_flight_info(info_arr)
```

6.2.8. solenoid.py

```
from time import time
from file_logger import log_flight_info


# constants and variables
time_to_deploy = 170*60
DEPLOY_PULSE_DURATION = 2
is_solenoid_activated = False
TIME_TO_CHECK_SEPARATION = 5*60

# log key
MECHANISM_LK = 6



def deploy(t0: float):
```

```
17    global time_to_deploy, DEPLOY_PULSE_DURATION, is_solenoid_activated,
      TIME_TO_CHECK_SEPARATION
18
19    t = time() - t0
20
21    if ((t > time_to_deploy) and (is_solenoid_activated == False)):
22
23        is_solenoid_activated = True
24
25    elif ((t > (time_to_deploy + DEPLOY_PULSE_DURATION)) and (is_solenoid_activated == True)):
26
27        is_solenoid_activated = False
28        time_to_deploy = t + TIME_TO_CHECK_SEPARATION
29
30    if (is_solenoid_activated == True):
31
32        info_arr = [MECHANISM_LK, "solenoid_activated", t]
33
34        log_flight_info (info_arr)
35
36    else:
37
38        info_arr = [MECHANISM_LK, "solenoid_not_activated", t]
39
40        log_flight_info (info_arr)
```