Emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS

Martínez Samaniego, Edison Alberto

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación previo a la obtención del título de Ingeniero

en Electrónica, Automatización y Control

Ing. Vargas Vallejo, Vanessa Carolina, PhD.

30 de enero de 2023

Firmado electrónicamente por: VANESSA CAROLINA VARGAS VALLEJO

Informe de originalidad

NOMBRE DEL CURSO
Revisión proyecto tesis

NOMBRE DEL ALUMNO
EDISON ALBERTO MARTINEZ SAMANIEGO

NOMBRE DEL ARCHIVO
EDISON ALBERTO MARTINEZ SAMANIEGO - TitulacionEdisonMartinez

SE HA CREADO EL INFORME
31 ene 2023

Resumen

Fragmentos marcados	6	0,3 %
Fragmentos citados o entrecomillados	0	0 %

Coincidencias de la Web

bbc.com	2	0,1 %
cartagena99.com	2	0,1 %
aem.gob.mx	1	0 %
lisanews.org	1	0 %

1 de 6 fragmentos

Fragmento del alumno MARCADO

La función **de** estos incluye las **comunicaciones, observación de la Tierra, navegación y posicionamiento, y el estudio del espacio y** del **planeta por parte de la ciencia**.

Mejor coincidencia en la Web

"La órbita de los satélites alrededor de **la** Tierra ejecuta una multitud **de** funciones que incluyen **comunicaciones** (cobertura del teléfono celular y transferencia de datos), **observación de la Tierra,**...

¿Cuántos satélites hay orbitando la Tierra y cómo es posible que no ... https://www.bbc.com/mundo/noticias-46408633

2 de 6 fragmentos

Fragmento del alumno MARCADO

Se ha de notar que **los satélites pueden ser propiedad de organizaciones, empresas, gobiernos y personas particulares**. Por lo que, teóricamente, es posible que casi cualquier…

Mejor coincidencia en la Web

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS**" fue realizado por el/los señor/señores **Martínez Samaniego, Edison Alberto;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 30 de enero de 2023

Firma:

Firmado electrónicamente por:
VANESSA CAROLINA
VARGAS VALLEJO

.....

Ing. Vargas Vallejo, Vanessa Carolina, PhD.

C. C.1711309045

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Yo, **Martínez Samaniego, Edison Alberto**, con cédula de ciudadanía n°1726833419, declaro que el contenido, ideas y criterios del trabajo de titulación**: Emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 30 de enero de 2023

Firma

Martínez Samaniego, Edison Alberto

C.C.: 1726833419

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Yo **Martínez Samaniego, Edison Alberto**, con cédula de ciudadanía n°1726833419, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 30 de enero de 2023

Firma

Martínez Samaniego, Edison Alberto

C.C.:1726833419

Dedicatoria

Dedico con mucho cariño este trabajo de titulación de ingeniero a mis padres, que han estado incondicionalmente para mí en todo momento, sin importar las circunstancias. Gracias por todos los grandes valores que me han enseñado, especialmente a ser responsable, respetuoso y perseverante. Me han enseñado a ser la persona que soy hoy y lo han hecho todo con una enorme dosis de amor y sin pedir nada a cambio.

A mi hermana Cami por su cariño, apoyo y motivación durante los momentos más complejos. A toda mi familia porque me han apoyado incondicionalmente durante todos mis años de estudio, dándome consejos y augurándome siempre lo mejor.

Agradecimiento

Quiero dar gracias a Dios por haberme permitido llegar a una nueva meta en mi vida.

A mis padres y hermanos, las personas más importantes de mi vida. Gracias por su apoyo cuando las cosas se ponían difíciles, pero sobre todo gracias por enseñarme a ser perseverante y trabajar por mis sueños.

A toda mi familia: tíos/as, abuelitos/as, primos/as, pero sobre todo a mi abuelita Francia, a mi tía Esthelita, a mi tía Marthita, a mi tía Pía, a mi tío Santi, a mi tío Edguitar y a mi tío Italo por su infinito cariño y por siempre estar pendientes de mí.

A mi mejor amiga, Cami por estar en los momentos más difíciles, siempre dispuesta a escucharme y aconsejarme.

A mis amigos de la universidad, por su ayuda y constancia a lo largo de esta gran aventura.

A mi tutora, Dra. Vanessa Vargas, por la confianza que tuvo en mi para la realización de este proyecto, por su compromiso, dedicación y paciencia.

Por último, agradezco a la Universidad de las Fuerzas Armadas ESPE, a todos los docentes que tuve durante mi carrera universitaria, quienes con la enseñanza de sus valiosos conocimientos hicieron que pueda crecer día a día como un profesional.

Índice de contenidos

Índice de Tablas

Índice de Figuras

Abreviaciones

ADCS	Subsistema de Determinación y Control de Orientación
API	Interfaz de Programación de Aplicaciones
BSP	Paquete de Soporte de Placa
CCSDS	Comité Consultivo para Sistemas de Datos Espaciales
cFE	core Flight Executive
COM	Subsistema de Comunicaciones
COTS	Componentes comerciales
EPS	Subsistema de Potencia Eléctrica
ESA	Agencia Espacial Europea
EXA	Agencia Espacial Civil Ecuatoriana
GEO	Órbita geosíncrona
IMU	Unidad de Medición Inercial
LEO	Órbita terrestre baja
MEO	Órbita terrestre media
NASA	Administración Nacional de Aeronáutica y el Espacio
OBC	Computador de a bordo
OBSW	Software del computador de a bordo
OSAL	Capa de Abstracción del Sistema Operativo
OSR	Reflector Solar Óptico

PSP	Paquete de Soporte de Plataforma
RF	Radio Frecuencia
RSB	Constructor de Recursos de RTEMS
RTOS	Sistema Operativo en Tiempo Real
TCS	Subsistema de Control Térmico
TTC	Telemetría, Seguimiento y Telecomandos
UHF	Frecuencia Ultra Alta
VHF	Frecuencia Muy Alta

Resumen

El presente trabajo de titulación realiza la emulación de las funciones básicas de un computador de a bordo (OBC) de un nanosatélite utilizando un computador embebido COTS. El computador de a bordo es esencialmente el cerebro del nanosatélite, ya que es el responsable de controlar los diferentes subsistemas, ejecutar las acciones enviadas por la estación terrestre y procesar los datos de los diferentes subsistemas. Para cumplir con las funciones básicas, el OBC debe tener un sistema operativo de tiempo real, sobre el que se ejecuta el sistema de control de misión. Este último permite monitorear y controlar los dispositivos de hardware del satélite de manera coordinada, así como proporciona una interfaz de programación para las aplicaciones. Las aplicaciones seleccionadas para el presente proyecto incluyen el monitoreo de las variables de altitud, inercia y temperatura del satélite. Adicionalmente se requiere de una aplicación de telemetría para enviar la información procesada a la estación terrestre. La configuración del hardware y software del OBC obedece a una estructura por capas, correspondiéndole una metodología de desarrollo, implementación y validación del proyecto también por capas. En el proceso de selección de software se optó por sistemas de código abierto que pueden ejecutarse en diferentes plataformas de hardware. Así se tuvo mayor versatilidad en la selección del computador embebido, que debía tener bajo consumo de potencia y bajo costo. De esta manera, la implementación del sistema inició con la instalación del sistema operativo RTEMS sobre el computador embebido Beaglebone Black. Sobre este se configuró el sistema de control de misión cFS - NASA. Posteriormente se diseñaron e implementaron las aplicaciones del nanosatélite y la aplicación de usuario del sistema terrestre para emular la comunicación de los segmentos. Una vez terminada la implementación del sistema, se procedió a validar las funcionalidades del sistema y se midió su desempeño en cuanto a consumo de potencia y al alcance del enlace de comunicaciones.

Palabras clave: Computador de a bordo; OBC; nanosatélite; computador embebido

Abstract

The present work emulates the basic functions of a nanosatellite on-board computer (OBC) using a COTS embedded computer. The on-board computer is essentially the brain of the nanosatellite, since it is responsible for controlling the different subsystems, executing the actions sent by the ground station and processing the data from the different subsystems. To fulfill the basic functions, the OBC must have a real-time operating system, on which the mission control system runs. The latter makes it possible to monitor and control the satellite's hardware devices in a coordinated manner, as well as, to provide a programming interface for applications. The applications selected for this project include the monitoring of satellite altitude, inertia, and temperature variables. Additionally, a telemetry application is required to send the processed information to the ground station. The OBC hardware and software configuration obeys a layered structure, corresponding to a project development, implementation and validation methodology also layered. In the software selection process, were chosen open-source systems that can run on different hardware platforms. This increases the versatility to select the embedded computer, that also have to accomplish low power consumption and low cost requirements. Thus, the implementation of the system began with the installation of the RTEMS operating system on the Beaglebone Black embedded computer. Then, the cFS - NASA mission control system was configured on this platform. Subsequently, the nanosatellite applications and the terrestrial system user application were designed and implemented to emulate the communication of the segments. Once the system implementation was completed, the OBC system functionalities were validated. Also, its performance was measured in terms of power consumption and the range of nanosatellite-to-terrestrial station communication.

Keywords: On-board computer; OBC; nanosatellite; embedded computer

Capítulo 1. Introducción

Antecedentes

Desde la antigüedad, el ser humano se ha sentido fascinado y atraído por explorar el espacio exterior del planeta. La curiosidad y el deseo de descubrir lo desconocido le ha impulsado a inventar y a desarrollar tecnología que le permita cumplir con ese objetivo. Este ideal empezó con relatos ficticios de viajes espaciales, como es el caso de “Historia Verdadera” escrita por Luciano de Somata en el siglo II d.C. o las novelas “De la Tierra a la Luna” y “Alrededor de la Luna” escritas por Julio Verne en la década de los años 1860, entre muchos otros. En un inicio, debido a las limitaciones tecnológicas no fue posible explorar el espacio exterior, hasta que el 4 de octubre de 1957, la Unión Soviética lanzó el primer satélite artificial. Desde entonces, el ser humano ha enviado una gran cantidad de satélites al espacio, de los que, según los datos de la Union of Concerned Scientists, aproximadamente 5400 están en órbita y en funcionamiento (Union of Concerned Scientists, 2022). La función de estos incluye las comunicaciones, observación de la Tierra, navegación y posicionamiento, y el estudio del espacio y del planeta por parte de la ciencia.

Se ha de notar que los satélites pueden ser propiedad de organizaciones, empresas, gobiernos y personas particulares. Por lo que, teóricamente, es posible que casi cualquier persona pueda enviar satélites al espacio gracias al desarrollo de nanosatélites (Llorente, 2018). Según la empresa española Alén Space, se puede construir y poner en órbita un nanosatélite por 500 000 euros, en comparación de un satélite convencional que puede alcanzar los 500 millones de euros (Alén Space, s.f.). Por esta razón, tecnologías como los nanosatélites, han permitido a las empresas producir y lanzar satélites de forma económica y rápida, convirtiendo así la exploración del espacio en un nuevo modelo de negocio (Nature Astronomy, 2020).

Si bien los costos de las tecnologías espaciales están disminuyendo y cada vez existe un mayor número de aplicaciones de código abierto, uno de los obstáculos que dificultan su aplicación y utilización, es el desconocimiento de las ventajas que presentan las tecnologías espaciales. La Comisión de Ciencia y Tecnología para el Desarrollo de la ONU (2020) destaca que aportan al desarrollo sostenible al garantizar la seguridad alimentaria, al reducir el riesgo de desastres, prevenir las crisis humanitarias, entre otros.

De los 5400 satélites operativos, 74 satélites están registrados bajo la propiedad de países latinoamericanos (Union of Concerned Scientists, 2022). Esto se debe a que, como menciona Guzmán (2021) la mayoría de los países de América Latina han tenido una gran cantidad de desafíos al momento de desarrollar tecnología espacial debido a múltiples problemas económicos, sociales y políticos que han impedido que los gobiernos inviertan en el desarrollo de tecnología en esta área del conocimiento. Esto se refleja en la investigación deficiente y falta de mejora tecnológica de la región, lo que a su vez provoca que la mayoría de los países opten por comprar tecnología espacial de países desarrollados minimizando así el desarrollo nacional.

Según Alvarado (2019) Ecuador fue la sexta nación en crear su agencia espacial (Agencia Espacial Civil Ecuatoriana EXA) a nivel de Latinoamérica en el año 2007, existiendo una diferencia de 47 años con respecto a la primera nación latinoamericana. Cabe notar que la agencia EXA es un organismo civil independiente. Estos hechos evidencian que en nuestro país ha habido pocas iniciativas para el desarrollo en el área espacial lo que ha impedido beneficiarse de su utilidad en el sector de seguridad, defensa, agricultura, comunicaciones y otros. Por ello el gobierno ecuatoriano consciente de su importancia, en noviembre de 2020, suscribió la declaración para la creación de la Agencia Espacial Latinoamericana y del Caribe (ALCE) constituyéndose en un mecanismo para fortalecer la cooperación en el desarrollo de temas aeroespaciales en la región.

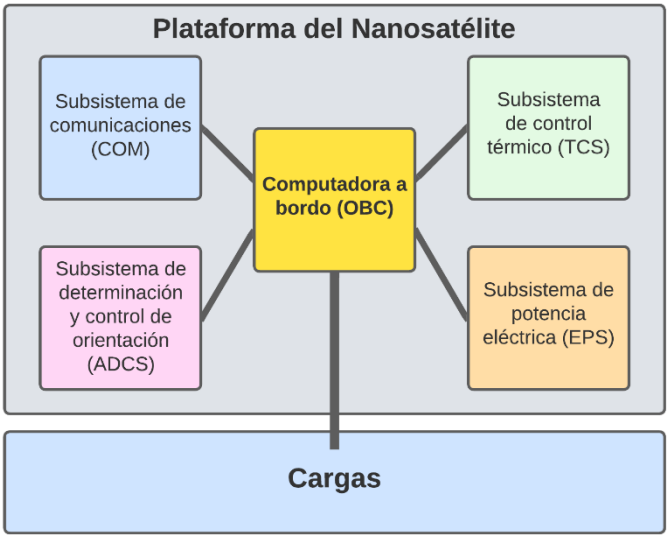
Hasta el momento sólo instituciones ecuatorianas privadas han lanzado satélites al espacio. EXA lanzó dos satélites, PEGASO y KRYSAOR, ambos en el 2013. Cuatro años más tarde, la Universidad Tecnológica Equinoccial UTE envió su primer nanosatélite UTE – UESOR en colaboración con la Universidad Suroeste de Rusia y en el año 2019 lanzo su segundo satélite, Ecuador – UTE. Por ello resulta importante que se trabaje en proyectos que apunten al lanzamiento de satélites por parte de instituciones gubernamentales. De hecho, la Fuerza Aérea Ecuatoriana a través de la Dirección de Investigación y Desarrollo Militar ha establecido lineamientos para el trabajo de proyectos orientados al desarrollo aeroespacial.

Por otra parte, la Universidad de las Fuerzas Armadas ESPE consciente de la importancia en este tema, creó en enero del 2012 el programa Misión Espacial para Observación de la Tierra que tenía por objetivo “fomentar las tecnologías espaciales en nuestro país mediante el desarrollo y la transferencia del conocimiento a partir de la ejecución de misiones espaciales con pico o nanosatélites, tanto para fines científicos como prácticos” (Tierra, 2011). Dentro de este programa se ejecutaron proyectos como el CUBESATESPE: FASE I cuyo objetivo fue diseñar y construir un nanosatélite bajo el estándar CubeSat. Entre los años 2011 a 2013 se desarrollaron proyectos relacionados con los sistemas estructurales y de alimentación de energía solar de un nanosatélite (Báez & Rodríguez, 2013) y otro relativo al sistema de transmisión y recepción de un CubeSat (Balarezo, 2012). Lamentablemente por factores económicos se detuvo el desarrollo en esta área.

Al momento, la Universidad de las Fuerzas ESPE busca retomar la temática satelital a través de proyectos como el “Propuesta de desarrollo y validación de un dispositivo educativo CANSAT para fomentar el acceso a las tecnologías aeroespaciales” que busca motivar el interés en temática satelital a todo nivel (López, 2021), y otros proyectos de mayor alcance científico que buscan desarrollar la construcción de un nanosatélite. Dentro del desarrollo, un aspecto clave y fundamental es el sistema de control, alimentación, monitoreo y

comunicaciones que conforman la arquitectura del nanosatélite. Addaim et al. (2010) detalla que la arquitectura consta de dos componentes principales, la plataforma del nanosatélite y la carga, misma que se puede observar en la Figura 1. Los principales módulos que componen a la plataforma son el computador de a bordo (OBC), el subsistema de comunicaciones (COM), el subsistema de determinación y control de orientación (ADCS), el subsistema de potencia eléctrica (EPS) y el subsistema de control térmico (TCS). Adicionalmente, al satélite se conectan cargas dependiendo de la aplicación, por ejemplo, un detector de tormentas ionosféricas, un reflector láser para mediciones precisas de órbita o una cámara HD para capturar imágenes del espacio o de la Tierra. Como departamento de Eléctrica, Electrónica y Telecomunicaciones, dentro del objetivo de construir el nanosatélite, existe el interés de contribuir en las áreas relacionadas al desarrollo e implementación del OBC.

Figura 1
Arquitectura general de un nanosatélite



Dentro de los componentes de los nanosatélites, el OBC cumple un papel fundamental al ser el elemento central que permite el control de los otros subsistemas, tanto los de la plataforma del nanosatélite como los instrumentos o cargas (Eickhoff, 2012). Además, el

computador a bordo es el encargado de implementar la gestión de misión. Durante la misión espacial se realizarán varias operaciones, tales como la recolección y procesamiento de datos obtenidos a partir de sensores. Otro ejemplo es la comunicación entre el nanosatélite y la estación terrestre. Si bien, la mayoría de las operaciones que se llevan a cabo durante la misión involucran a otros subsistemas, todas están relacionadas en algún momento con el OBC (Amadis, 2021). Así por ejemplo durante el monitoreo del voltaje de las baterías que realiza el módulo EPS los valores son almacenados en la memoria del OBC para una posterior transmisión de la información mediante el módulo de comunicaciones a la estación terrestre.

Un diseño e implementación deficiente de los módulos que componen el computador de a bordo del satélite puede ocasionar el fallo total de la misión, por esta razón es crucial conocer y entender en profundidad la arquitectura del sistema. De manera general, un OBC está constituido por un microprocesador, bancos de memoria e interfaces de conexión para conectar el computador a bordo a los otros subsistemas (Prasad, 2022). Con el propósito de disminuir el tiempo de desarrollo y los costos, existen empresas dedicadas a la tecnología espacial que utilizan circuitos integrados comerciales (COTS) para construir el OBC (Zeif, Kubicka, & Hörmer, 2022).

Dado que los nanosatélites son sistemas con altas restricciones temporales, se requiere un software que ejecute las acciones correctas en el intervalo de tiempo establecido. Por esta razón, se utiliza un sistema operativo en tiempo real (RTOS) implementado en la plataforma del satélite que garantice la confiabilidad, precisión y estabilidad del sistema. Además, su utilización permite llamar tareas en específico solo cuando es necesario, lo que facilita un mejor flujo de programa y respuesta a eventos, en comparación con un sistema operativo basado en bucles (Cooke, 2012).

Como menciona Camargo y Andrade (2013) debido a las condiciones del entorno de operación de un satélite, este no debe requerir ningún tipo de mantenimiento. Los RTOS

instalados sobre un sistema embebido proporcionan características de seguridad, robustez y fiabilidad que garantizan el correcto funcionamiento del subsistema de control de un nanosatélite, pero se ha de prestar atención al momento de seleccionar el sistema operativo ya que su rendimiento varía de acuerdo con el hardware que se utilice.

Justificación del proyecto

El desarrollo aeroespacial representa una oportunidad de innovación y avance, tecnológico y científico tanto a nivel institucional y nacional impulsando el desarrollo del país en este sector. De esta manera será posible aprovechar los innumerables beneficios que estas tecnologías permiten. Adicionalmente, considerando que el dominio de la Universidad de las Fuerzas Armadas ESPE es la seguridad y defensa y que este se alinea con la misión y visión de la Fuerza Aérea Ecuatoriana en el desarrollo de nuevas tecnologías espaciales resulta importante que se lleven a cabo proyectos de investigación, desarrollo e innovación en este ámbito.

Dentro de este contexto, el objetivo de implementar un nanosatélite a nivel institucional es factible considerando las diferentes disciplinas de la Universidad. Adicionalmente, el personal académico del departamento de Eléctrica, Electrónica y Telecomunicaciones está capacitado para investigar y desarrollar un sistema de gestión de misión tolerante a fallos en un OBC. Con el fin de alargar el tiempo de vida del nanosatélite, el OBC debe implementar un sistema fiable que garantice la ejecución de todas las acciones que requiere la misión espacial. En este contexto resulta fundamental realizar proyectos piloto en la selección e implementación del hardware y del sistema operativo que podría instalarse en un OBC.

El presente proyecto de titulación es un paso fundamental ya que a través de la emulación de las funciones básicas de un computador de a bordo de un nanosatélite se realizará el proceso de selección, implementación y validación tanto del sistema operativo como del software de control de misión sobre un OBC, verificando que el sistema sea confiable

y ligero. Adicionalmente dado que el sistema será instalado en un computador de bajo costo, esto permitirá realizar las primeras pruebas exploratorias a un costo accesible. Es decir que este trabajo es una de las primeras etapas que se llevan a cabo al desarrollar un proyecto con tecnologías espaciales. Por tanto, este proyecto aportará significativamente para el desarrollo de la línea de aprendizaje requerida en materia espacial, disminuyendo así la dependencia tecnológica del país.

Alcance

En el presente proyecto de titulación se resume la investigación del estado del arte realizada que permitió elegir un sistema operativo en tiempo real que cumpla con los altos requerimientos temporales de los nanosatélites. De esta manera se garantizarán los plazos de ejecución de las tareas de navegación. Para la selección del sistema operativo se consideró que sea de código abierto, que exista compatibilidad con diferentes plataformas de hardware, que sea de tiempo real, que sea versátil, que exista soporte técnico y que exista documentación técnica.

Posterior a ello se presenta una comparativa de frameworks de control de misión. Cabe señalar que el objetivo de este proyecto no es su diseño debido a su complejidad. Tal como menciona Eickhoff (2012) el desarrollo de este software es una tarea con un gran nivel de complejidad, por esta razón, autores como Bocchino et al. (2018) recomiendan usar frameworks diseñados para ser usados en múltiples misiones. Además, destacan que el desarrollar software de control de misión es sumamente costoso y/o implica software de baja calidad e ineficiente. Para la selección del software de control de misión se analizó que sea de código abierto, que exista documentación técnica, sea portable a varios sistemas operativos, en particular con el seleccionado previamente, que exista soporte técnico, que exista documentación técnica y que aplique los estándares del Consultative Committee for Space Data Systems (CCSDS; el nombre se puede traducir como “Comité Consultivo para Sistemas

de Datos Espaciales”). De esta manera se seleccionó el software más adecuado para el proyecto.

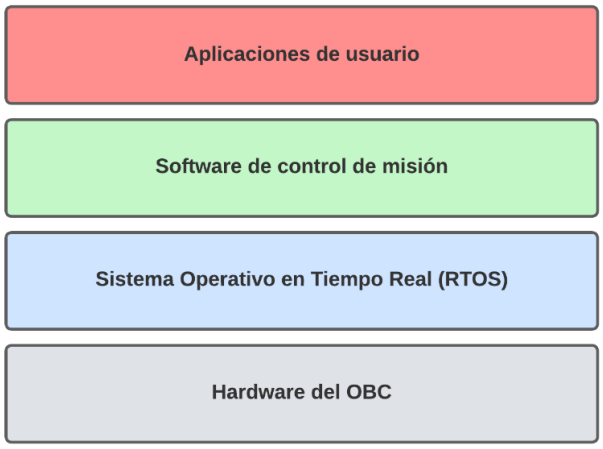
En lo que respecta a la plataforma de hardware, se seleccionó un computador embebido COTS para emular el OBC, ya que, una de las principales características de los sistemas embebidos es el bajo costo y un reducido consumo de potencia (DIEEC, 2011) (Lutkevich, 2020). Es importante notar que los nanosatélites son sistemas ciber físicos, es decir están caracterizados por su conexión al entorno físico mediante sensores y actuadores.

En este proyecto se configuró el sistema operativo y el software de control de misión sobre la plataforma de hardware seleccionada. Acto seguido se procedió a implementar las aplicaciones de usuario, las cuales permiten monitorear la temperatura ambiente, altitud e inercia del sistema. Además, se implementó una aplicación de telemetría para enviar los datos de las aplicaciones previamente mencionadas. De esta manera se obtuvo un sistema de software funcional de la gestión de misión del satélite instalado en un computador embebido, con una arquitectura como la que se observa en la Figura 2. Por otro lado, se desarrolló una aplicación de escritorio que permita recibir y visualizar la telemetría enviada por el OBC.

Finalmente, el sistema propuesto fue sido validado mediante pruebas de funcionamiento del sistema de control de misión, aplicaciones de usuario y de la aplicación de escritorio.

Figura 2

Arquitectura de software del sistema aplicativo a implementar



Objetivos

Objetivo general

Emular las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS.

Objetivos específicos

- Realizar el estudio del estado del arte de sistemas de gestión de misión para nanosatélites.
- Seleccionar un sistema operativo en tiempo real (RTOS) de código abierto.
- Seleccionar un sistema de control de misión que permita ejecutar todas las acciones que requiere la misión espacial.
- Seleccionar la plataforma de hardware compatible con el RTOS y el sistema de control de misión para realizar la emulación.
- Configurar el sistema operativo y el sistema de gestión de misión.
- Implementar una aplicación de telemetría.
- Implementar una aplicación de monitoreo de la altitud del nanosatélite.
- Implementar una aplicación de monitoreo de la inercia del nanosatélite.
- Implementar una aplicación de monitoreo de la temperatura del nanosatélite.
- Implementar una aplicación de escritorio que reciba la telemetría enviada por el nanosatélite.
- Validar el correcto comportamiento del sistema implementado en el computador embebido mediante pruebas de funcionalidad.

Oportunidades y retos de dispositivos electrónicos en el espacio

La evolución tecnológica en esta área se ha visto impulsada de gran manera por las disciplinas aeroespaciales. Esto se debe a que, desde que empezó la exploración espacial a finales de la década de los años 1950, el ser humano ha buscado soporte en la tecnología para

explorar lo desconocido en el espacio exterior. Los dispositivos electrónicos, como parte de las tecnologías usadas en el espacio, han contribuido en diferentes áreas tales como, por ejemplo, la comunicación y salud, entre otras. Pero es importante notar que, al encontrarse en el espacio, los dispositivos electrónicos se enfrentan a un ambiente extraño y hostil, diverso al entorno terrestre. A continuación, se explica la contribución de las tecnologías espaciales en diferentes áreas y los desafíos a los que se enfrentan en el espacio exterior.

Contribución de las tecnologías espaciales

La Organización Meteorológica Mundial (OMM, 2022) menciona que mediante el uso de tecnologías espaciales se miden a diario parámetros clave de la atmósfera, la tierra y la superficie de los océanos. De esta manera, proporcionan servicios de previsión meteorológica y de sequías a los agricultores, pastores y pescadores. Además, mediante los datos recolectados, se proporcionan servicios de monitoreo y predicción de ciclones tropicales y de inundaciones. Información que permite prevenir crisis humanitarias y alertar anticipadamente a los entes involucrados.

Por otra parte, las tecnologías espaciales contribuyen también a brindar servicios de conectividad al proporcionar acceso a infraestructura de prestación de servicios en zonas apartadas. Un ejemplo de esto se da en Bangladesh, donde se emiten programas de televisión y radio a personas que viven en zonas en las que el acceso a redes terrestres es limitado o inexistente (Comisión de Ciencia y Tecnología para el Desarrollo de las Naciones Unidas, 2020).

La contribución de las tecnologías espaciales se da en varios campos, como por ejemplo al permitir realizar reuniones en forma síncrona y en tiempo real a larga distancia, mediante las videollamadas. Este permite por ejemplo en el campo de la salud contar con servicios como la telemedicina y telesalud que sin las tecnologías espaciales serian impracticables. Como destaca Hay et al. (2006), otro aporte de las tecnologías espaciales a la

salud es la recolección de datos sobre las elevaciones y condiciones ambientales; comúnmente usadas en el seguimiento geográfico de enfermedades infecciosas.

Desafíos de la electrónica en el espacio

El reto más grande para la electrónica en el espacio es hacer frente a los distintos aspectos del entorno en el que operan los componentes. Como se mencionó previamente, los dispositivos electrónicos que abandonan la Tierra se enfrentan a condiciones inhóspitas. El primer desafío se presenta cuando la estructura en la que se encuentran los componentes electrónicos es eyectada del vehículo de transporte. Al realizarse la ignición de los propulsores, considerados dispositivos pirotécnicos, se induce lo que se conoce como choque pirotécnico. Este a su vez puede dañar las placas electrónicas o provocar cortocircuitos en los componentes (Arenas & Margasahayam, 2006). Una vez que los componentes electrónicos están fuera del vehículo de transporte, estos se enfrentan a condiciones físicas diversas que afectan el desempeño de los dispositivos, mismas que serán descritas a continuación.

El entorno en el espacio exterior. El factor más peligroso en el espacio interplanetario es la radiación. En particular, Petkov (2003) describe al Sol como una fuente activa de radiación y plasma, que emite protones, electrones, fotones e iones altamente cargados. El problema de la radiación radica en que, en caso de que una partícula ionizante atraviese a un componente eléctrico puede afectar su funcionamiento al dar lugar a transitorios y fallas permanentes que pongan en riesgo el funcionamiento del sistema, siendo los dispositivos de memoria, microprocesadores y FPGAs los más sensibles (Federal Aviation Administration, 2016).

Por otro lado, es necesario notar que en el espacio exterior las condiciones son de vacío, lo que ocasiona que materiales no metálicos como polímeros, adhesivos y gomas, desprendan gases. Los vapores que provienen de estos elementos eventualmente se

condensarán y se depositarán en otros materiales. Como consecuencia de esta contaminación se reduce el rendimiento de los componentes (Leonard, 2017).

Por último, se ha de considerar el entorno electroestático. Como describe Lu et al. (2019) el espacio cercano a la Tierra es un entorno de plasma de baja temperatura, cuya densidad y energía varía dependiendo de la altura. Una emisión intensa de radiación desde el Sol puede afectar al entorno de plasma, aumentando la densidad y temperatura, lo que a su vez puede ocasionar descargas electroestáticas.

Se ha de aclarar que, si bien las condiciones térmicas son un factor por considerar en la exploración espacial, los dispositivos electrónicos se encuentran encapsulados en un ambiente térmico controlado en el interior de las naves espaciales (Petkov, 2003). Por esta razón, este factor no será considerado como una condición del espacio exterior que afecte a los componentes electrónicos.

Organización del manuscrito

Como se indicó, el presente trabajo de titulación aportará en la emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido de bajo costo. Para ello, en el Capítulo 2 se realiza un estudio sobre las misiones satelitales, que es un nanosatélite y su arquitectura. Adicionalmente, se examinan las principales características del computador de a bordo y del software que se instala sobre este. En el Capítulo 3 se establece la plataforma de emulación, en donde se seleccionará tanto el software del OBC como la plataforma de hardware. En el Capítulo 4 se incluye los detalles más relevantes de la implementación del sistema. Posterior a ello, el Capítulo 5 describe la validación del sistema, se especifican las pruebas funcionales realizadas y los resultados obtenidos. Finalmente, el Capítulo 6 presenta las conclusiones, recomendaciones, además de los trabajos que se podrían realizar en el futuro.

Capítulo 2: Fundamentos teóricos y estado del arte

Como se mencionó en el capítulo anterior, las tecnologías espaciales traen consigo beneficios en diferentes áreas. Para que eso sea posible, se deben llevar a cabo dos procesos igual de importantes: el desarrollo del proyecto espacial y la misión propiamente dicha. El primero involucra el diseño, desarrollo y construcción de la tecnología espacial, para lo cual es necesario definir la aplicación y la arquitectura que se requiere de acuerdo a los objetivos planteados. El segundo proceso implica el planteamiento de objetivos, el traslado desde la superficie terrestre al espacio exterior (lanzamiento), y, por último, la comunicación con la tecnología espacial para su monitoreo y control. De esta manera, se puede asegurar el correcto funcionamiento de las tecnologías espaciales como consecuencia del cumplimiento de los objetivos establecidos.

En este capítulo se describen los fundamentos teóricos del presente trabajo de titulación, para lo cual se inicia con un estudio sobre las misiones satelitales, luego se describen a los nanosatélites y su arquitectura haciendo especial énfasis en el computador de a bordo y sus características.

Misiones satelitales

Introducción histórica a las misiones satelitales

Un satélite se define como un “objeto artificial que orbita el planeta Tierra u otro planeta” (Kongsberg NanoAvionics, 2022). Estos han sido utilizados desde que, el 4 de Octubre de 1957, la Unión Soviética lanzó el primer satélite artificial, el Sputnik 1. Este fue seguido un mes después por el Sputnik 2, el cual llevaba en su interior a Laika, una perra callejera que se convertiría en primer ser vivo terrestre en llegar al espacio. Posterior al éxito soviético, el 31 de Enero de 1958, Estados Unidos realizó el lanzamiento del Explorer 1. Este satélite junto al Explorer 3, lanzado en Marzo del mismo año, sirvieron para confirmar la existencia de una zona con partículas cargadas energéticamente, los cinturones de Van Allen (Loff, 2017). En otras

palabras, fueron estos eventos históricos los que dieron paso a la carrera espacial entre Estados Unidos y la Unión Soviética.

A finales de 1958, el satélite artificial SCORE fue lanzado al espacio por Estados Unidos, siendo considerado el primer satélite de comunicaciones. Esto se debe a que, como explica la National Aeronautics and Space Administration (NASA, 2022; el nombre se puede traducir como “Administración Nacional de Aeronáutica y el Espacio”) a diferencia de los satélites previamente lanzados, este satélite llevaba un sistema de comunicaciones que era capaz de grabar señales y retransmitirlas tanto a medida que las recibía o posteriormente bajo comando. Cuatro años más tarde, en 1962, se lanzaría el primer satélite de comunicaciones con propósito comercial, el Telstar 1. Este fue lanzado con el objetivo de realizar una transmisión de televisión transatlántica, probar la transmisión de llamadas telefónicas y demostrar que era posible transmitir datos tales como telefotos (Hoth, O'Neill, & Welber, 1963). Es decir que, si bien la carrera espacial fue iniciada como una competencia dominada por intereses políticos y por sus potenciales aplicaciones militares, luego dio paso a una explotación económica de los satélites para mediante ellos proveer servicios de conectividad y servicios de meteorología, entre otros. El éxito de una misión espacial depende de varios factores como son entre otros un lanzamiento adecuado, la ubicación correcta en la órbita de acuerdo al propósito del satélite, la utilización del mismo para brindar los servicios al mayor plazo posible y el control y monitoreo de la misión. Por ello el sistema de baterías, de comunicaciones y la estación terrestre cumplen con un papel fundamental.

Estación terrestre

El segmento o estación terrestre es el responsable de comunicarse con el satélite para poder controlarlo y recibir los datos que este envíe para posterior a ello procesarlos. De acuerdo con Planet Aerospace (2020) el segmento terrestre consiste en una estación o una red de estaciones, con sistemas de comunicación de radio frecuencia (RF) y sistemas computación

para el procesamiento de datos. Es importante notar que la estación terrestre y el satélite deben estar en constante comunicación para asegurar el éxito de la misión, por lo tanto, debe existir un enlace de comunicación permanente entre los dos, como se observa en la Figura 3. A continuación, se explican los detalles de dicha comunicación, la forma de enviar y recibir datos del satélite.

Figura 3
Enlace de comunicación entre satélite y estación terrestre

Comunicación con el satélite. La comunicación con los satélites debe ser del tipo full-dúplex, es decir que el flujo de datos es bidireccional. Esto permite que sea posible tanto enviar información de control como recibir la información que estos envían. La comunicación se realiza utilizando el espectro de radiofrecuencia, de 1GHz a 40GHz y se encuentra dividido en 6 bandas, las cuales según la Agencia Espacial Europea (ESA, s.f.) son:

- Banda L (1-2GHz): Usada para satélites empleados en sistemas GPS y satélites de telefonía móvil
- Banda S (2–4 GHz): Usada para satélites de comunicaciones, radares meteorológicos y radares de barcos

- Banda C (4–8 GHz): Usada para comunicaciones por satélites, redes de televisión satelital o transmisiones satelitales
- Banda X (8–12 GHz): Usada principalmente por los militares. Las subbandas son usadas para el monitoreo del clima, control de tráfico aéreo, control de tráfico marítimo y la detección de la velocidad de los vehículos
- Banda Ku (12–18 GHz): Usada para comunicaciones por satélite. La conexión de bajada se realiza de 10.7 GHz a 12.75 GHz.
- Banda Ka (26–40 GHz): Usada para satélites de comunicación. La conexión de subida se realiza de 27.5 GHz a 31 GHz.

Telemetría y telecomandos. La Telemetry, Tracking, and Command (TTC; el nombre se puede traducir como “Telemetría, Seguimiento y Telecomandos”) son funciones vitales que permiten la comunicación de datos entre la estación terrestre y el satélite. La telemetría de un satélite consiste en el envío de parámetros de un satélite a través del enlace de bajada (downlink) hacia la estación terrestre. Estos datos incluyen mediciones a través de sensores de: voltajes, corrientes, temperaturas y la orientación del satélite, entre otros. De esta manera es posible evaluar el rendimiento del satélite y tomar acciones correctivas de ser necesario. Para ser enviados a la estación terrestre, los datos son empaquetados a palabras de 8 bits y organizados en un formato preestablecido.

Por otro lado, los telecomandos son instrucciones codificadas que son transmitidas desde la estación terrestre hacia el satélite a través de la conexión de subida (uplink) con el fin de operar y controlar al satélite. Estos comandos sirven para controlar la orientación del satélite, el cambio de estado de relés conectados o para cambiar alguna configuración del satélite.

La tercera función del TTC es el seguimiento del satélite. Esto significa determinar la distancia que existe entre la estación terrestre y el satélite mediante la medición del tiempo de propagación (Winton et al., 1996).

Cargas y aplicaciones de usuario

Como se indicó anteriormente, todas las misiones satelitales tienen un propósito específico; pudiendo ser de investigación, de demostración del funcionamiento de tecnología desarrollada o la provisión servicios comerciales. Para cumplir con el objetivo de la misión, todo satélite tiene como componente principal a lo que se conoce como carga. La carga es el instrumento o sistema que permite, dependiendo de la misión, observar, medir, procesar o entregar el servicio.

Es importante notar que, de acuerdo con la ESA (s.f.), en el satélite la carga o los sistemas de cargas tienen sistemas de comunicaciones específicos al igual que sistemas de control que permiten cumplir con los objetivos de la misión. Como parte de estos últimos se incluyen las aplicaciones de usuario, programas informáticos diseñados para procesar los datos de las cargas de ser necesario. Por otra parte, en el segmento terrestre, al igual que en el satélite, existe un sistema de comunicaciones y equipamiento dedicados a la carga, para de esta manera controlar y obtener la información que esta envíe.

De manera general las cargas se pueden clasificar en:

- Cargas para observación de la Tierra
- Cargas para comunicación
- Cargas para aplicaciones científicas

Cargas para observación de la Tierra. Estas utilizan cámaras ópticas o infrarrojas que sirven para observar áreas particulares. Los satélites que tienen este tipo de cargas son puestos en órbita de tal manera que orbiten a la misma velocidad que el planeta Tierra orbita el

Sol. En otras palabras, se consigue que el satélite pase sobre el área de interés a la misma hora, garantizando que las condiciones de iluminación sean casi constantes, variando solo con la estación (Agencia Espacial Europea, 2020).

Cargas para comunicación. De acuerdo con Braun (2012) estas cargas son aquellas que reciben, filtran, procesan, amplifican y transmiten señales. Se utilizan para proveer servicios de comunicación como por ejemplo la transmisión de mensajes de texto o el servicio del Sistema de Identificación Automática que utilizan las embarcaciones, mediante el cual los barcos transmiten su posición y así evitan colisiones entre ellos.

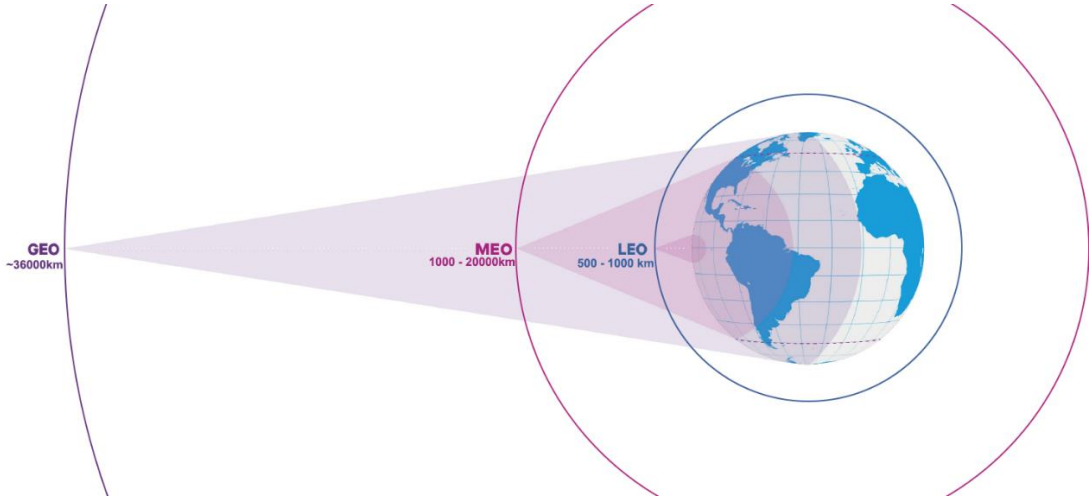
Cargas para aplicaciones científicas. Estas cargas varían de acuerdo con la aplicación científica ya que sirven para llevar a cabo experimentos que permitan demostrar hipótesis previamente planteadas. Un ejemplo es la misión del satélite SporeSat. Este satélite tenía por objetivo investigar como células vegetales en germinación perciben y responden a la gravedad. Para ello la carga del SporeSat tenía tres dispositivos “lab on a chip”, es decir dispositivos con funciones de un laboratorio, que integraban sensores en un único chip (NASA, 2017).

Tipos de Órbitas Terrestres

Dependiendo del objetivo de la misión, los satélites son ubicados en diferentes tipos de orbitas. De acuerdo a su altura se dividen en tres tipos de órbitas conocidas como baja, media y geosíncrona, mismas que se describen a continuación. En la Figura 4 se puede observar una imagen representativa de las alturas y áreas de cobertura de las diferentes órbitas terrestres.

Figura 4

Esquemático de la altitud y área de cobertura de las órbitas terrestres



Nota. Adaptado de *Schematic of orbital altitudes and coverage areas,* por Via Satellite, 2022, Via Satellite Archive (https://www.satellitetoday.com/content-collection/ses-hub-geo-meo-and-leo/)

Órbita terrestre baja (LEO). Los satélites en esta órbita se encuentran a alturas entre 500 y 1000 km sobre la superficie terrestre. El ángulo que se forma entre el plano ecuatorial y el plano de la órbita puede variar de 0 a 180 grados. Si este ángulo es de 90 grados, la órbita se llama órbita polar y eso permite que los satélites pasen sobre cada punto del planeta, teniendo una cobertura mundial. El periodo orbital depende de la altura a la que se encuentra el satélite, pero oscila entre los 90 y 120 min. Este tipo de orbita es muy usado por los satélites usados en aplicaciones científicas y por aquellos que son usados para mediciones meteorológicas.

Órbita terrestre media (MEO). Los satélites en esta órbita se encuentran a una altura de 1000 a 20000km sobre la superficie terrestre. El periodo orbital puede ser desde 6 hasta 12 horas. Este tipo de orbitas es usado por los satélites pertenecientes a los sistemas GPS y por algunos satélites usados para comunicaciones.

Órbita geosíncrona (GEO). Esta orbita tiene una altura de aproximadamente 36000km sobre la superficie terrestre. Esto permite la órbita de los satélites coincida con la rotación de la Tierra. En otras palabras, el periodo orbital es el mismo que el de la Tierra, 23 horas y 56 minutos. Si un satélite se ubica con una órbita GEO sobre el ecuador, conseguirá una órbita estacionaria que no se mueve con respecto a la superficie, es decir que el satélite estará siempre ubicado sobre la misma ubicación en la superficie terrestre. Este tipo de orbitas es muy utilizado para aplicaciones de monitoreo del clima y para aplicaciones de comunicaciones.

Nanosatélites

Desde que empezó la carrera espacial, el desarrollo, construcción y puesta en órbita de los satélites fue una tarea compleja y costosa. Sin embargo, gracias a la miniaturización de los dispositivos electrónicos, en 1980 apareció un nuevo paradigma, los satélites pequeños. Esto trajo consigo varias ventajas, entre las cuales se destacan una disminución en el tiempo y costo de desarrollo, lo que a su vez permitió que más naciones alrededor del mundo sean partícipes de la exploración espacial.

De acuerdo con la NASA (2017), los satélites pequeños se pueden clasificar conforme a su masa en:

- Minisatélite, de 100 a 180kg
- Microsatélite, de 10 a 100kg
- Nanosatélite, de 1 a 10kg
- Picosatélite, de 0.01 a 1kg
- Femtosatélite, de 0.001 a 0.01kg

De entre estos, los nanosatélites son actualmente los más utilizados debido a que, como destaca Planet Aerospace (2020) son los más adecuados para llevar cargas que pueden ser manejadas por medianas empresas, estudiantes e investigadores para realizar

experimentos científicos y tecnológicos. Además, los nanosatélites se pueden desarrollar adoptando un estándar preestablecido o utilizando un diseño propio. De los estándares existentes, uno de los más conocidos es el CubeSat, desarrollado a inicios de 1999 entre la Universidad Politécnica Estatal de California y la Universidad de Stanford. Estos son nanosatélites que utilizan un tamaño estándar de “una unidad” o “1U”, el cual mide 10x10x10 cm y puede ser agrandado a diferentes tamaños como 1.5U, 2U, 3U, 6U y hasta 12U. La ventaja de este estándar radica en que proporciona una plataforma rentable para realizar investigaciones científicas o demostraciones de nueva tecnología (NASA, 2017).

Ventajas y limitaciones

El paradigma de los satélites pequeños trajo consigo algunos beneficios sobre los satélites convencionales, es decir aquellos con un peso mayor a 180kg. El tamaño y la complejidad de los pequeños satélites se traducen en una reducción de costos, particularmente por la relación directamente proporcional que existe entre el peso y el costo de lanzamiento. Esto a su vez significa que la exploración espacial es más accesible para la mayoría de las naciones e incluso para diferentes empresas. Otro aspecto por destacar es que el tiempo de desarrollo de un pequeño satélite es más corto que el de un satélite convencional. Esto se debe a que, para subsistemas como el de comunicaciones, muchos nanosatélites utilizan dispositivos COTS lo que significa que los componentes y repuestos se pueden adquirir con gran facilidad. Además, en la actualidad existen diferentes compañías como Nanoavionics y Alén Space, entre otras, que ofrecen nanosatélites, ya fabricados, bajo el estándar CubeSat. De ser el caso, al comprar un nanosatélite ya fabricado, tanto el tiempo de desarrollo como el costo se ven reducidos considerablemente.

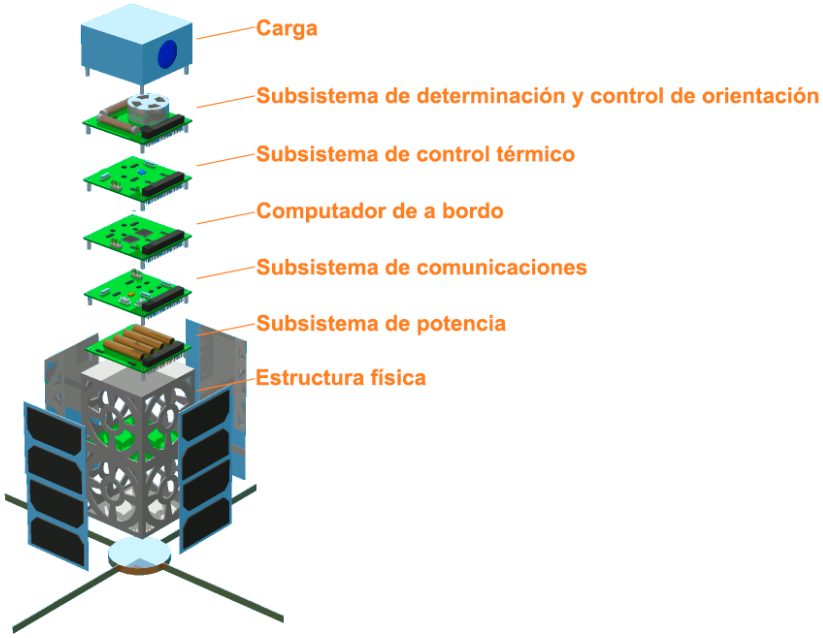
Sin embargo, el paradigma de los satélites pequeños también trae consigo algunas limitaciones. Como por ejemplo que las cargas de los nanosatélites se ven limitadas por el tamaño y la potencia disponibles en el sistema. Por esta razón, una misión que requiera una

carga muy grande como un telescopio no sería factible utilizando satélites pequeños. También se ha de tomar en consideración que la vida útil de los nanosatélites es menor que la de los satélites tradicionales debido a su limitado almacenamiento de energía y combustible.

Arquitectura de hardware básica de un nanosatélite

Como se mencionó anteriormente, la arquitectura de está compuesta por la plataforma del nanosatélite y la carga. Ambas montadas sobre una estructura física. La plataforma consta de diferentes subsistemas que sirven para garantizar que la misión satelital se lleve a cabo y sea exitosa. La Figura 5 muestra la representación gráfica de la arquitectura de un nanosatélite. De todos los componentes de hardware, el subsistema más importante es el computador de a bordo (OBC), el cual es esencialmente el cerebro del sistema al cual se conectan todos los subsistemas. De manera general, es el responsable de manejar toda la información en el nanosatélite, esto quiere decir que prepara datos para transmitirlos a la estación terrestre y ejecuta los telecomandos que recibe desde la Tierra. Además, el OBC procesa la información de los diferentes subsistemas y cargas que componen al nanosatélite y ejecuta los algoritmos y acciones de control correspondientes para mantener una orientación adecuada y temperatura dentro de los rangos establecidos. En resumen, el OBC es el sistema encargado de coordinar el funcionamiento de los otros subsistemas, ejecutar comandos e interpretar las medidas de los sensores. Posteriormente se explicará más sobre las características del computador de a bordo. A continuación, se detallan cada uno de los componentes de hardware del sistema.

Figura 5
Representación de la arquitectura de hardware básica de un nanosatélite



Nota. Adaptado de *Structure of a CubeSat,* por SpaceBillboard, 2013, SpaceBillboard (http://spacebillboard.com/mission/)

Estructura física. Es el cimiento sobre el que se colocan los diferentes subsistemas del nanosatélite, por lo que debe proporcionar facilidad de acceso para el montaje e integración de otros componentes. Además, la estructura debe ser capaz de soportar las condiciones del entorno espacial, brindando protección a los diferentes subsistemas.

Para elaborar una estructura liviana se utilizan en su mayoría aleaciones de aluminio, titanio, acero inoxidable, berilio y materiales compuestos. De estos, las aleaciones de aluminio son las más utilizadas por su relación fuerza – peso. La desventaja de usar este material es su alto coeficiente de expansión térmica, esto quiere decir que la dilatación y contracción del material es mayor cuando existen cambios de temperatura. Por otra parte, las aleaciones de titanio, a diferencia de las aleaciones de aluminio, tiene un coeficiente de expansión térmica bajo y debido a que tienen una relación fuerza – peso alto, son usadas en aplicaciones que

requieren una estructura muy rígida. El Berilio es un material que sirve para aplicaciones aeroespaciales en las que se necesita una gran rigidez y debido a su bajo coeficiente de expansión térmica es usado en vidrios y cristales ópticos. La desventaja de este material radica en su costo elevado y es difícil de tratar. En cuanto a las aleaciones de acero inoxidable, estas son usadas para proteger de la corrosión. Por último, los materiales compuestos son combinaciones de dos o más materiales que aprovechan las ventajas de cada uno. Por ejemplo, algunas láminas de material compuesto de epoxi con grafito pueden ser fabricadas para presentar un coeficiente de expansión térmico cercano a cero y otras para tener una rigidez extraordinaria (Planet Aerospace, 2020).

Subsistema de potencia (EPS). La energía eléctrica es indispensable para el funcionamiento de cualquier satélite y esta debe ser generada, regulada y distribuida a los diferentes subsistemas continuamente. La generación de la energía se basa en la utilización de paneles solares compuestos de células fotovoltaicas que convierten la luz solar en energía eléctrica. Es necesario notar que los satélites no siempre están expuestos a la luz solar, en particular durante la fase de eclipse, cuando la Tierra o la Luna previenen que la luz solar llegue al satélite. Durante estos periodos de tiempo el satélite funciona mediante baterías recargables. Esto significa que las baterías se descargan durante la fase de eclipse y se recargan durante el periodo en el que el satélite recibe luz solar.

Debido a que el periodo orbital es diferente para los nanosatélites ubicados en LEO y GEO, la frecuencia de la fase de eclipse no es la misma para ambos. En el caso de los nanosatélites que se encuentren en LEO, durante una órbita, el periodo de eclipse es de aproximadamente 36 min y el periodo de luz solar es de aproximadamente 65 min. Esto quiere decir que una batería completamente cargada, durante una órbita, atraviesa un ciclo de carga y descarga. Por lo tanto, a lo largo de un día, considerando que el satélite realiza 12 orbitas, atraviesa 14 ciclos aproximadamente y durante un año 5110 ciclos. Por otro lado, los satélites

de comunicación que se encuentran en GEO experimentan 90 ciclos al año. Estos factores son importantes a considerar ya que el rendimiento de las baterías se ve afectado cuando el proceso de carga y descarga se lleva a cabo múltiples veces.

Subsistema de comunicaciones (COM). Como se mencionó previamente, el sistema de comunicación de un satélite incluye un sistema destinado para el sistema TTC y otro para la transmisión y recepción de datos de la carga. En el caso de los nanosatélites, en el 2014, la Unión Internacional de Telecomunicaciones determinó que la mayoría de estos usan las bandas de frecuencia muy alta (VHF) y frecuencia ultra alta (UHF). Esto se debe a que muchos nanosatélites hacen uso de módulos de comunicación fabricados con componentes comerciales (Commercial off-the-shelf COTS), en particular para el receptor de telecomandos y transmisor de telemetría se utilizan módulos de una sola placa PCB. En el caso de las cargas, el sistema de comunicaciones es dependiente de la aplicación. Por lo general las cargas necesitan un ancho de banda y una tasa de transferencia mayor a las disponible en las bandas VHF y UHF, por lo que se adoptan sistemas de comunicación para las bandas L, S, X o Ku/Ka.

Según Elfvelin (2022) el uso de las bandas VHF y UHF permite cuatro modos de operación: semidúplex en VHF, semidúplex en UHF, full dúplex con VHF para la transmisión de la telemetría y en UHF para la recepción de telecomandos o viceversa. Sin embargo el autor menciona que el modo de operación más común es full dúplex con VHF para la recepción de telecomandos y en UHF para la transmisión de la telemetría.

Subsistema de determinación y control de orientación (ADCS). Este sistema es el responsable monitorear y controlar la orientación del satélite de acuerdo con los requerimientos de la carga. Por ejemplo, en el caso de los satélites de comunicación, la antena debe estar orientada hacia una ubicación en particular en la Tierra para de esta manera poder proveer los servicios de telecomunicaciones. El problema es que, a medida que el planeta continúa girando sobre su eje polar, el satélite debe rotar de manera que la antena este siempre apuntando a la

misma ubicación. Además, la órbita del satélite tiende a desviarse debido a la fuerza gravitacional ejercida por la Tierra o a causa de la presión producto de los vientos solares.

El subsistema está compuesto de sensores de orientación, actuadores y programas informáticos con software para de esta manera poder determinar y controlar la orientación del nanosatélite. De manera general, los sensores más usados para la medición de orientación son:

- Sensores solares: De manera general funcionan con una cámara rectangular que permite el paso de un rayo de luz hacia un panel con fotosensores y dependiendo de la posición del rayo se puede determinar la orientación del satélite.
- Seguidores de estrella: Son los sensores más precisos que sirven para determinar la orientación. Son dispositivos ópticos que, mediante una cámara capturan la posición de una estrella en particular. Esta imagen es procesada para obtener vectores que permitan conocer la posición del nanosatélite, los cuales serán comparados con vectores referenciales obtenidos a partir de un catálogo de estrellas almacenado a bordo del nanosatélite.
- Sensores de horizonte: Son sensores de infrarrojo que miden la radiación al borde de la atmósfera y la contrastan con el espacio exterior.
- Magnetómetros: Estos son sensores que miden la dirección y magnitud del campo magnético terrestre. Para determinar la orientación, primero se construye vectores de referencia a partir del modelo del campo de referencia geomagnético internacional y se los compara con los vectores medidos por el sensor.

- Giroscopios: Este tipo de sensores proporciona una medida de la velocidad angular del satélite en los tres ejes. Son utilizados generalmente junto a otro de los sensores de orientación para proporcionar datos más precisos.

Por otra parte, los actuadores del sistema se pueden clasificar en tres tipos:

- Dispositivos de cambio de momento: Mismos que no utilizan combustible para generar movimiento. El funcionamiento de estos actuadores se basa en el principio de la conservación del momento angular. Por ejemplo, la rueda de reacción se utiliza para girar a los satélites en cantidades muy pequeñas. Estos dispositivos funcionan con un motor eléctrico que hace girar a la rueda en un eje en particular, si la rueda gira en el sentido de las manecillas del reloj el satélite girara en sentido contrario para conservar el momento angular.
- Actuadores magnéticos: Estos dispositivos son energizados con el propósito de producir un campo magnético que produce torque al interactuar con el campo magnético de la Tierra.
- Dispositivos de expulsión: También conocidos como propulsores, estos actuadores expulsan masa en una dirección, produciendo el desplazamiento del satélite en la dirección contraria. Los propulsores pueden ser de gas caliente, es decir usan combustibles líquidos o gas frío, es decir la masa es gas comprimido.

Subsistema de control térmico (TCS). Los satélites en el espacio exterior se enfrentan a la temperatura del espacio que puede llegar a ser de 3 grados Kelvin o -270 ºC (NASA, 2014). Adicionalmente son expuestos a la radiación solar, a la radiación solar reflejada por la Tierra y a la radiación infrarroja emitida por la Tierra. Debido a que en el espacio exterior las condiciones son de vacío, la transferencia de calor por convección no es posible, por lo que la única manera de transferencia de energía es mediante radiación y a través de conducción mediante las superficies conductoras de la estructura física. Por esta razón, la función principal

del subsistema de control térmico es mantener la temperatura de todos los subsistemas dentro de un rango determinado durante toda la misión satelital. De no ser así el rendimiento de los diferentes elementos se vería afectado, poniendo en riesgo el éxito de la misión.

Durante la etapa de lanzamiento desde Tierra, debido a que todos los satélites se encuentran dentro del vehículo de lanzamiento estarán protegidos frente a los cambios de temperatura y presión mientras el vehículo se mueve a través de la atmosfera. Una vez que el nanosatélite sea liberado para ser puesto en órbita; este, estará en movimiento a una altura mayor a 200 km donde la presión es muy baja. Por lo tanto, la transferencia de calor se dará solo por conducción y radiación. Es en esta etapa donde el subsistema de control térmico empieza a funcionar. De manera general existen dos tipos de control térmico en un satélite, pasivo y activo.

El *control pasivo* involucra a cualquier método que no requiera una entrada de energía para funcionar. El método más usado se basa en las mantas de aislamiento multicapa, las cuales cubren a todas las superficies externas del satélite. Las mantas consisten en diferentes capas de escudos altamente reflectivos separados por una malla de nylon poco conductiva. Las capas externas son cubiertas en oro para así proteger de radiación UV y rayos X.

Otro método de control térmico pasivo es el uso del Optical Solar Reflector (OSR; el nombre se puede traducir como “Reflector Solar Óptico”) o de espejos de segunda superficie con baja tasa de absorción/emisión. La diferencia radica en que el OSR tiene vidrio de cuarzo como superficie protectora y el espejo de segunda superficie tiene láminas de plástico flexible. Mientras el OSR se utiliza para cubrir áreas amplias y planas, los espejos de segunda superficie son utilizados para cubrir superficies curvas o muy pequeñas.

También como método de control térmico pasivo se puede citar el uso de radiadores pasivos que, como dice su nombre, sirven para radiar calor excesivo en el nanosatélite.

Finalmente, otro método de control térmico pasivo es el revestimiento de superficies usando pintura o químicos más sofisticados para disminuir o aumentar la transferencia de calor de ser necesario. Por lo general los componentes internos son pintados de negro para maximizar el intercambio de calor con otros equipos y superficies, mientras que las superficies externas como los radiadores son pintadas de color blanco para permitir una emisión apropiada de calor al espacio exterior (Anvari et al., 2009). La única desventaja de este método es que el revestimiento se degrada rápidamente debido al entorno.

Por otra parte, el *control activo* se realiza mediante la implementación de elementos como calentadores y “louvres” (similar a persianas venecianas en forma). Los calentadores se utilizan para proteger a los sistemas cuando están en condiciones de baja temperatura. Por ejemplo, antes de que todos los sistemas enciendan los calentadores incrementan la temperatura de los componentes a la temperatura mínima de operación. Las “louvres” están hechas de láminas metálicas reflectantes y son colocadas encima de los radiadores. De tal manera que, bajo condiciones de alta temperatura se mantienen abiertas para permitir que el calor sea irradiado; en tanto que, cuando baja la temperatura, las “louvres” se cierran.

Cargas. Tal como se mencionó previamente, las cargas de los satélites son aquellos instrumentos o sistemas que se utilizan en las misiones satelitales para cumplir con un objetivo. En el caso de los nanosatélites se han de tener ciertas consideraciones con respecto a las cargas. Se ha de tener en cuenta que un nanosatélite como máximo pesa 10kg, por lo que el peso de la carga debe ser una fracción de esto, esto obliga a los responsables de la misión satelital o utilizar una carga que sea compatible con el nanosatélite. También se ha de analizar el consumo de energía utilizado por la carga, ya que los nanosatélites al ser físicamente pequeños no tienen la posibilidad de llevar una gran cantidad de baterías o baterías de gran capacidad por lo que su actividad seria limitada si la carga requiere una gran cantidad de energía eléctrica (Lucia et al., 2021). Por otra parte, se ha de tomar en consideración las

especificaciones del sistema de comunicaciones propio de la carga, ya que puede ser necesario cambiar de banda de frecuencias por requerimientos en las tasas de transferencia.

Considerando que el computador de a bordo es la parte esencial del presente proyecto de titulación se detalla en la siguiente sección de una manera más amplia.

Computador de a bordo

Si bien, se destacó que las cargas son los instrumentos o sistemas mediante los cuales se cumplen con el objetivo de la misión satelital, el computador de a bordo cumple con un rol primordial en el éxito de la misión a través de sus tres principales funciones en el manejo de información:

1. El OBC decodifica los telecomandos recibidos por el subsistema de comunicaciones para la posterior ejecución de la acción deseada.
2. El computador a bordo recolecta los datos de telemetría, los codifica en un formato establecido y envía a través del sistema de comunicaciones.
3. El OBC es el responsable de procesar las mediciones de los sensores del subsistema de determinación y control de orientación, y del subsistema de control térmico para tomar de ser necesario las acciones correctivas a través de los diferentes actuadores de los subsistemas.

Hardware del computador de a bordo

El hardware del OBC es dependiente del tipo de misión, órbita y carga. Por ejemplo, en una misión en la que se tenga una órbita GEO sobre el ecuador, el satélite estará siempre ubicado sobre la misma ubicación en la superficie terrestre, lo que le permitirá transmitir sin interrupciones a la estación terrestre todos los datos recolectados de la carga y los diferentes subsistemas. Por otra parte, si la órbita es de tipo LEO existirán periodos de tiempo en los que el satélite no tendrá contacto con la estación terrestre por lo que será necesario que el OBC

tenga la capacidad de almacenar los datos de forma segura hasta que la conexión con el segmento terrestre se reanude. En otras palabras, se necesitarán dispositivos de memoria. Estos pueden ser de tipo no volátil cuando los datos deben ser almacenados durante un gran periodo de tiempo, o de tipo volátil cuando la información va a ser procesada rápidamente.

Debido a las funciones que cumple el computador de a bordo es necesario que todos los subsistemas se encuentren conectados al OBC para que de esta manera sea posible el intercambio de información. Por lo general, para la comunicación con los subsistemas se utilizan protocolos como lo son el I^2C, SPI, CAN o USB (Velázquez, 2019).

En resumen, el computador de a bordo está compuesto por un microprocesador, memoria no volátil, memoria volátil e interfaces de comunicación que permitan la conexión con los diferentes subsistemas. Furano y Menicucci (2018) señalan que en la actualidad el desarrollo de las tecnologías espaciales se ha visto limitado al uso de microprocesadores o a máquinas de estado de gran complejidad implementadas en FPGAs.

Software del computador de a bordo

Para que el computador de a bordo sea capaz de cumplir las diferentes funciones mencionadas previamente se requiere de un conjunto de aplicaciones conocidas como el software del computador de a bordo (OBSW). De manera general, el OBSW se encuentra compuesto por un sistema operativo , el sistema de control de misión y las aplicaciones de usuario. El sistema operativo sirve de interfaz entre el procesador y el sistema de control de misión. El sistema de control de misión a su vez es una capa que proporciona servicios y hace posible la ejecución de operaciones necesarias para cumplir con el objetivo de la misión. Por último, las aplicaciones de usuario son aquellos programas que hacen uso de los servicios proporcionados por el sistema de control de misión para poder ejecutar las operaciones necesarias para cumplir con el objetivo de la misión.

Sistema operativo en tiempo real. El sistema operativo es el componente de software que administra el acceso a la memoria y otros periféricos del computador de a bordo. Además, este componente es el responsable de la administración y coordinación de actividades y recursos en el OBC.

Considerando que las aplicaciones satelitales requieren altas restricciones temporales y que el OBC ejecuta una gran cantidad de tareas, los sistemas operativos en tiempo real (RTOS) son los más utilizados en esta área (Cooke, 2012). Adicionalmente, la utilización de un RTOS ayuda a reducir la complejidad del código escrito, debido a que es posible descomponer un proyecto en diferentes tareas o procesos que se comuniquen y sincronicen mediante herramientas como mutex, semáforos y colas de mensajes.

Sistema de control de misión. También conocido como software de vuelo o framework de vuelo, es el sistema que permite al satélite realizar todas las operaciones necesarias para cumplir con el objetivo de la misión. Este software hace uso de la Application Programming Interface (API; el nombre se puede traducir como “Interfaz de Programación de Aplicaciones”)que proporciona el RTOS para brindar servicios de gestión del tiempo del OBC, interfaz de control de hardware, procesamiento de telemetría y telecomandos, entre otros.

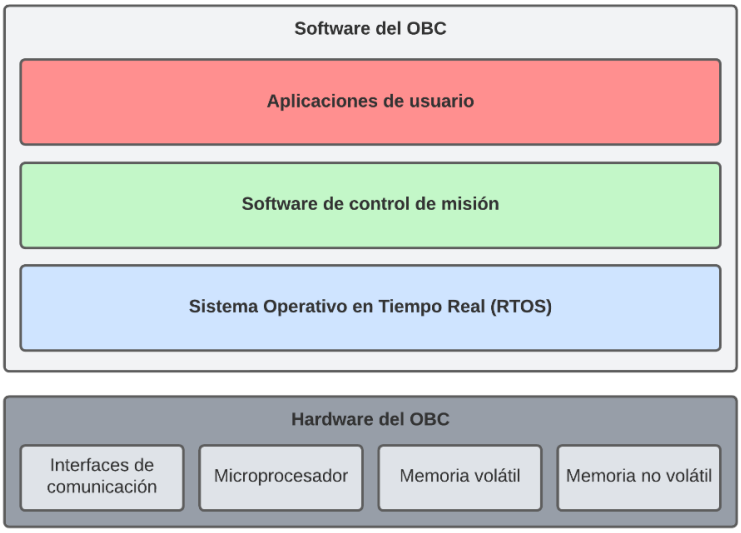
Dvorak (2009) menciona que existen cuatro características que distinguen al software de vuelo de otros programas utilizados comercialmente en ingeniería:

- No existen interfaces directas con el usuario, como monitor, teclado y mouse. Todas las interacciones realizadas a lo largo de una misión espacial entre los operadores de la estación terrestre y el satélite se realizan a través del enlace de comunicación RF.
- Una de las tareas principales del software de vuelo es monitorear y controlar los dispositivos de hardware sobre el satélite de manera coordinada.

- El sistema de control de misión realiza procesamiento en tiempo real, lo que quiere decir que debe satisfacer las restricciones temporales.
- Dado que se requiere programar el software de vuelo en un entorno de recursos limitados se requiere experiencia adicional.

Aplicaciones de usuario. Estos son programas están estrechamente ligados a la misión satelital y a la carga. Es a través de las aplicaciones de usuario que el OBC procesa los datos de los diferentes subsistemas y ejecutan las acciones de control cuando es necesario. Para poder ejecutarse, las aplicaciones de usuario se apoyan en los servicios proporcionados por el software de vuelo.

Figura 6
Arquitectura de Hardware y Software del computador de a bordo



Conforme a lo mencionado previamente, de manera general la arquitectura de hardware y software del OBC es como se ilustra en la Figura 6. Dado que el presente proyecto de titulación se centra en la emulación de las funciones básicas del computador de a bordo, en el Capítulo 3, se detallará la selección del OBSW, la selección de la plataforma de hardware embebido y se describirán los sensores, módulos y componentes externos utilizados para establecer la plataforma de emulación.

Capítulo 3: Plataforma de emulación del computador de a bordo

El presente proyecto de titulación tiene como objetivo la emulación de las funciones básicas de un computador de a bordo de un nanosatélite utilizando un computador embebido COTS. Para definir la plataforma del computador a utilizarse es necesario recordar que estas funciones se resumen en la decodificación de telecomandos para su ejecución, en la recolección de datos de telemetría para enviarlos a la estación terrestre y en el procesamiento de la información proveniente de los sensores del subsistema de determinación y control de orientación, y del subsistema de control térmico.

Por otra parte, considerando que tanto el hardware como el software del OBC deben ser compatibles entre ellos, se ha planteado seleccionar en primer lugar software de código abierto del OBC que sea compatible con varias tarjetas embebidas. Luego seleccionar la tarjeta embebida que sea compatible con el software y que permita la conexión con los sistemas de sensores necesarios para emular las aplicaciones deseadas.

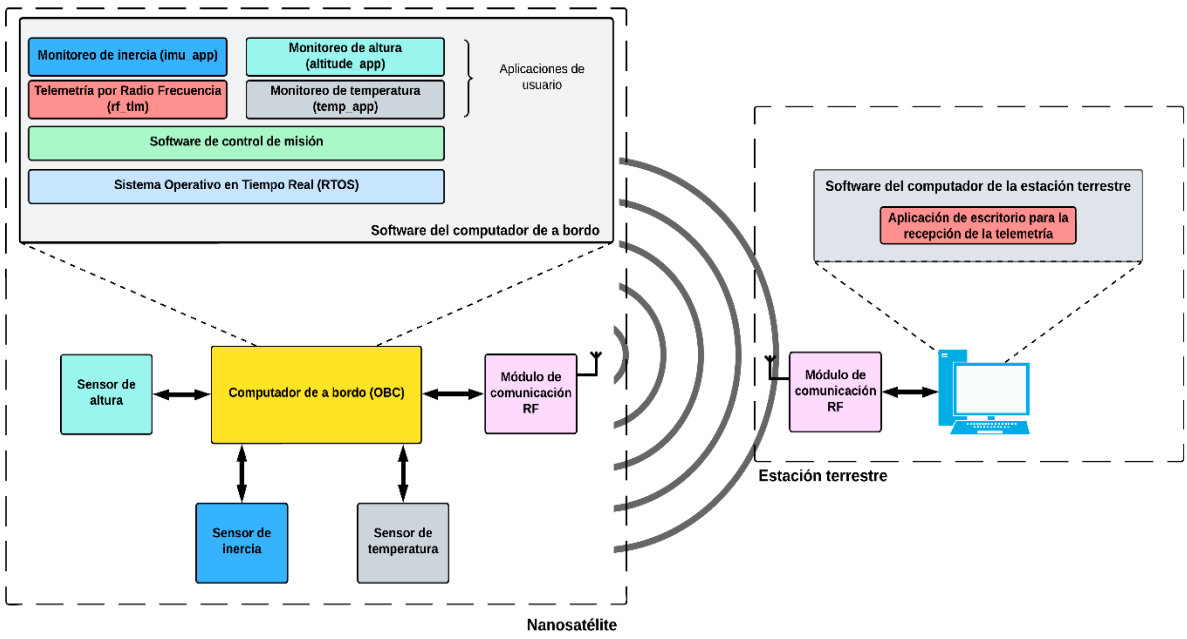
En cuanto al software, se requiere seleccionar el sistema operativo en tiempo real, el sistema de control de misión y finalmente las aplicaciones de usuario, que, para el presente trabajo, serán cuatro:

- Aplicación para monitorear la altitud del nanosatélite
- Aplicación para monitorear la inercia del nanosatélite
- Aplicación para monitorear la temperatura del nanosatélite
- Aplicación para enviar la telemetría por comunicación RF a la estación terrestre

Adicionalmente, para la emulación de la comunicación con la estación terrestre se desarrolló una aplicación de escritorio a través de la cual se puede monitorear la telemetría enviada por la plataforma de emulación mediante un sistema de comunicación RF. Por otra parte, para poder realizar las mediciones de inercia, temperatura, altura y para el envío de la

telemetría, se seleccionaron los sensores, módulos y componentes externos adecuados. Cabe señalar que para el presente proyecto no se va a realizar ninguna acción de control. Dentro de este marco, de manera general la arquitectura de hardware y de software del sistema es como se observa en la Figura 7.

Figura 7
Diagrama de bloques de la propuesta de diseño



De esta manera este capítulo se ha organizado de forma concordante con la metodología planteada. Es decir, en primer lugar, se presenta la selección del software del OBC, luego se realiza la selección del hardware del OBC y finalmente se detallan los dispositivos sensores utilizados.

Selección del software del computador de a bordo

Sistema operativo en tiempo real

Uno de los aspectos claves que se tomó en cuenta al seleccionar el sistema operativo en tiempo real es que este sea multiplataforma, esto quiere decir que el RTOS tenga la

capacidad de operar en diferentes plataformas de hardware. De esta manera, se garantiza que el software sea reutilizable en otras misiones espaciales en las que se empleen diferentes OBC. Otro aspecto que se tuvo presente es el propósito del RTOS, ya que existen algunos sistemas operativos que, están optimizados para funcionar en aplicaciones específicas y se busca un sistema versátil.

También se consideró como criterio de selección el uso mínimo de memoria del sistema operativo. Esto se debe a que, si el RTOS usa menos memoria, existe más espacio disponible para el sistema de control de misión y las aplicaciones de usuario. De igual manera, se verificó que el software tenga soporte técnico, es decir que la institución que lo desarrolla garantice el soporte y realice mantenimiento al sistema. Lo que significa que se realizan constantemente mejoras y corrección de errores del RTOS. Por último, al seleccionar el sistema operativo en tiempo real se consideró que este sea de código abierto y exista documentación técnica, para que, de ser necesario, se puedan realizar cambios al código fuente.

En resumen, los criterios de selección son:

- Código abierto: El código fuente y la documentación se encuentran distribuidos de manera gratuita en internet.
- Documentación técnica: La disponibilidad de guía de usuario y documentación técnica
- Multiplataforma: El diseño e implementación del sistema permiten su ejecución en diferentes plataformas de hardware.
- Propósito: El objetivo para el que fue diseñado el sistema operativo.
- Soporte técnico: La institución que desarrolló el sistema garantice el soporte técnico y mantenimiento al software.
- Memoria: El uso mínimo de memoria para el correcto funcionamiento

Tabla 1
Comparación de sistemas operativos en tiempo real

Criterio	FreeRTOS	Nut/OS	RTEMS	RT-Thread	ThreadX	VxWorks
Código abierto	Si	Si	Si	Si	Si	No
Documentación técnica	Si	Si	Si	Si	Si	No
Multiplataforma	Si[a]	Si[b]	Si[c]	Si[d]	Si[e]	Si[f]
Propósito	General[a]	General[b]	General[c]	Industria e IoT[d]	Industria e IoT[e]	General[f]
Soporte técnico	Si	Si	Si	Si	Si	Si
Memoria	Depende de la configuración y las aplicaciones[a]	-	Depende de la configuración y las aplicaciones[c]	3 KB Flash 1.2 KB RAM[d]	2 KB Flash 1 KB RAM[e]	Depende de la configuración y las aplicaciones[f]

Nota. [a]Recuperado de Amazon (2022) por Amazon Web Services, Inc. [b]Recuperado de egnite (s.f.) por egnite GmbH. [c]Recuperado de The RTEMS Project (s.f.) por On-Line Applications Research (OAR) Corporation. [d]Recuperado de RT-Thread (s.f.) por RT-Thread Development Team. [e]Recuperado de Meadows et al. (2022) por Microsoft. [f]Recuperado de Wind River Systems (s.f.) por Wind River Systems, Inc.

En la Tabla 1 se puede observar la comparación de los sistemas operativos en tiempo real. De los RTOS analizados, solo VxWorks necesita una licencia pagada para poder utilizarlo y acceder a su documentación técnica. En lo que respecta a la multiplataforma, se encontró el caso particular de que Nut/OS en su estado actual es compatible solo con procesadores AVR8, AVR32 y ARM. A pesar de ello, el sistema operativo tiene la capacidad de ser portado a otras plataformas de hardware, mientras que el resto de RTOS si cumple con el criterio de selección de multiplataforma.

Al comparar el propósito de cada sistema operativo se constató que tanto RT-Thread como ThreadX están diseñados específicamente para aplicaciones industriales y de IoT, mientras que el resto de RTOS se pueden utilizar en todo tipo de aplicaciones al ser de propósito general. Por otro lado, en cuanto al soporte técnico, se accedió a todos los

repositorios de los sistemas operativos de código abierto (FreeRTOS, Nut/OS, RTEMS, RT-Thread y ThreadX) y se pudo evidenciar que los diferentes RTOS se mantienen en constante mantenimiento.

Al buscar el uso mínimo de memoria de los diferentes sistemas operativos, en el caso de FreeRTOS, RTEMS y VxWorks es dependiente de la configuración y de las aplicaciones creadas, mientras que para Nut/OS no existe información relacionada al uso de memoria.

De acuerdo con lo antes expuesto, FreeRTOS, Nut/OS y RTEMS cumplen con los criterios de selección establecidos. Sin embargo, es importante notar que, en el caso de Nut/OS, al momento no ha sido portado a una gran cantidad de plataformas. Por esta razón, sería necesario realizar una serie de modificaciones al sistema operativo para portarlo a una nueva plataforma.

De acuerdo con la ESA (s.f.) el sistema operativo en tiempo real RTEMS es usado en varias misiones supervisadas por el departamento de Ingeniería en Software (TEC-SWE). Además, en el documento se señala que se encuentran en un proceso de certificación al sistema operativo, lo que evidencia la gran calidad técnica de RTEMS. Es por esta razón, y considerando los resultados de la comparación de los diferentes RTOS, que se decidió utilizar RTEMS como sistema operativo del computador de a bordo.

Sistema de control de misión

Como se mencionó previamente, el desarrollo de un sistema de control de misión desde cero es una tarea compleja y costosa. Por consiguiente, se analizaron cinco sistemas de control de misión con el fin de seleccionar uno para el presente proyecto. Es importante destacar que, uno de los criterios de selección más relevantes es la portabilidad del sistema de control de misión. Esto quiere decir que su diseño permita la ejecución en diferentes sistemas

operativos. De esta manera el sistema de control de misión seleccionado será capaz de funcionar correctamente en el RTOS seleccionado.

El Consultative Committee for Space Data Systems (CCSDS; el nombre se puede traducir como “Comité Consultivo para Sistemas de Datos Espaciales”) es un foro en el que participan múltiples agencias espaciales a nivel mundial con el fin de establecer estándares para los sistemas de comunicación y datos de los vuelos espaciales. Esto es fundamental ya que, en muchas misiones espaciales es necesario intercambiar datos con diferentes satélites o estaciones terrestres. Por esta razón, uno de los criterios de selección fue que el sistema de control de misión cumpla con los estándares establecidos por el CCSDS.

Con respecto a otros criterios de selección se estableció que el sistema sea de código abierto y exista documentación técnica. De esta manera es posible estudiar el funcionamiento del framework y de ser necesario, se pueden realizar cambios. Además, se verificó que el software tenga soporte técnico, es decir que la institución que lo desarrolla garantice el soporte y realice mantenimiento al sistema. Por último, se comparó el mínimo uso de memoria y recursos del CPU de todos los framework, ya que este puede ser un factor determinante a la hora de seleccionar la plataforma de hardware.

En resumen, los criterios de selección son:

- Código abierto: El código fuente y la documentación se encuentre distribuidos de manera gratuita en internet.
- Documentación técnica: La disponibilidad de guía de usuario y documentación técnica
- Portabilidad: El diseño e implementación del sistema permiten su ejecución en diferentes sistemas operativos y procesadores.

- Soporte técnico: La institución que desarrolló el sistema garantice el soporte técnico y mantenimiento al software.
- Estandarización CCSDS: El sistema de control de misión implemente estándares establecidos por el CCSDS
- Rendimiento: El uso mínimo de memoria y recursos del CPU

La Tabla 2 presenta la comparación de los diferentes sistemas de control de misión. De los framework analizados, solo GERICOS no es de código abierto. En otras palabras, solo está disponible para la organización que lo desarrolla (LEISA), por lo que no se tiene acceso a la documentación de este sistema. Por otro lado, todos los sistemas de código abierto (cFS, F prime, KubOS y SUCHAI) proporcionan documentación técnica necesaria para la implementación del framework.

Al comparar la portabilidad de los sistemas se puede observar que todos tienen la capacidad de ejecutarse en diferentes sistemas operativos en tiempo real, pero solo cFS y GERICOS han sido portados a RTEMS, el RTOS seleccionado.

En lo que respecta a soporte técnico, todos los sistemas lo poseen. Sin embargo, al acceder a los repositorios de los framework de código abierto (cFS, F prime, KubOS y SUCHAI) se observó que SUCHAI no ha recibido actualizaciones o modificaciones desde octubre del año 2021, lo que significa que el software del sistema no se mantiene en constante mantenimiento y evolución a diferencia de los otros frameworks.

Por otra parte, actualmente el grupo de trabajo de servicios de soporte de aplicaciones del CCSDS está trabajando en el proyecto “NASA core Flight System (cFS) as a CCSDS Onboard Reference Architecture” en el cual se evalúa la arquitectura de este framework y el uso de los estándares del CCSDS en este sistema. Lo que demuestra el interés por parte del

CCSDS en el cFS debido a su calidad técnica y capacidad de interoperabilidad con sistemas basados en los estándares establecidos por el foro.

Tabla 2
Comparación de sistemas de control de misión

Criterio	cFS	GERICOS	F prime	KubOS	SUCHAI
Código abierto	Si (GitHub)	No	Si (GitHub)	Si (GitHub)	Si (GitLab)
Documentación técnica	Si	No	Si	Si	Si
Portabilidad	Si es portable (ha sido portado a Linux, RTEMS y VxWorks) [a]	Si es portable (ha sido portado a RTEMS, ThreadX y FreeRTOS) [b]	Si es portable (ha sido portado a Linux, VxWorks y FreeRTOS) [c]	Si es portable (ha sido portado a FreeRTOS y Linux) [d]	Si es portable (ha sido portado a Linux y FreeRTOS) [e]
Soporte técnico	Si (por la NASA)	Si (por LESIA del Observatorio de Paris)	Si (por la NASA)	Si (por Xplore Inc.)	Si (SPEL de la Universidad de Chile)
Estandarización CCSDS	Si	No	No	No	No
Rendimiento Memoria	-	10 kB de RAM [b]	-	32MB - 64MB de RAM 100MB - 400MB para el almacenamiento del sistema operativo [d]	10 kB RAM y 64 kB de Flash [e]
Rendimiento CPU	-	Bajo consumo de recursos de CPU	-	-	Bajo consumo de recursos de CPU

Nota. [a]Recuperado de Jean Timmons (2020) por NASA. [b]Recuperado de Plasson et al.(2016) por Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique. [c]Recuperado de NASA (2018) por California Institute of Technology. [d]Recuperado de Kubos Corporation (2017) por Kubos Corporation. [e]Recuperado de Gonzalez et al. (2019) por Universidad de Chile.

Tomando en consideración lo mencionado anteriormente y conforme a los criterios definidos, se seleccionó el sistema de control de misión cFS. En cuanto al uso mínimo de memoria y consumo de recursos del CPU no es posible comprar todos los sistemas de control de misión debido a la ausencia de información.

Selección de plataforma de hardware embebido

Una vez seleccionado el software del computador de a bordo, es posible escoger una plataforma de hardware compatible tomando en cuenta ciertas consideraciones. Si bien el sistema operativo en tiempo real RTEMS es multiplataforma, se decidió seleccionar un computador embebido al que el RTOS ya haya sido portado. De esta manera es posible trabajar inmediatamente sin necesidad de modificar el código del sistema operativo.

Por otra parte, en cuanto a las especificaciones de hardware establecidas por el software, se tiene de la Tabla 2 que el sistema de control de misión necesita como mínimo 800kB de memoria Flash y 2MB de memoria RAM. Por esta razón, como primer criterio de selección, el computador embebido debe contar con más de 2MB de RAM. En cuanto al almacenamiento Flash se decidió que la plataforma de hardware debe contar con un conector microSD en el que se pueda poner un dispositivo de memoria desde el cual se pueda arrancar el sistema. Esto conlleva el beneficio de que se pueden utilizar tarjetas microSD de gran capacidad, cumpliendo así con el requisito de 800kB de memoria Flash.

Otro factor por considerar para la selección del computador embebido son sus interfaces de comunicación. Dado que los protocolos de comunicación más usados son UART, SPI e I^2C se analizará la cantidad de interfaces de hardware disponibles para estos protocolos. Por otra parte, es importante tener en cuenta el consumo de corriente del computador a bordo, ya que este es un factor determinante en la vida útil del nanosatélite.

Al igual que en el caso del RTOS y del sistema de control de misión, es primordial que exista documentación técnica referente a la plataforma de hardware. Y en este caso, también como criterio de selección se ha de considerar el precio del computador embebido.

En resumen, los criterios de selección son:

- Costo del computador embebido

- Documentación técnica: La disponibilidad de guía de usuario y documentación técnica
- Interfaces de comunicación: La cantidad de interfaces de hardware disponibles para los protocolos de comunicación UART, SPI e I^2C.
- El menor consumo de corriente por parte del computador embebido considerando que todo el consumo de potencia del sistema incluyendo sensores y dispositivos de comunicación y de control debe ser menor a 5W.
- Memoria RAM: La mayor capacidad de memoria RAM disponible en el computador embebido
- MicroSD: Si el computador embebido dispone de un conector microSD en el que se pueda poner un dispositivo de memoria desde el cual se pueda arrancar el sistema.

El sistema operativo en tiempo real RTEMS tiene una gran cantidad de computadores embebidos a los que ha sido portado, por lo que, para realizar la comparación de las plataformas de hardware se han seleccionado cinco en particular. En la Tabla 3 se presenta la comparación de los diferentes computadores embebidos. De las plataformas de hardware analizadas, se puede observar que todas poseen documentación técnica y un conector microSD desde el cual se puede arrancar el sistema. De la misma manera, todos los computadores cumplen con el requisito mínimo de 2MB de memoria RAM.

Tabla 3
Comparación de plataformas de hardware

Criterios	BeagleBone Black[a]	Cora Z7-07S[b]	LPC4088 Developer's Kit[c]	Raspberry Pi 4B[d]	Raspberry Pi 3A+[e]
Precio (USD)	$52.50	$149.00	$188.57	$45.00	$25.00
Documentación técnica	Si	Si	Si	Si	Si
Interfaces de comunicación	4x UART 2x SPI 2x I^2C	2x UART 2x SPI 2x I^2C	5 x UART 3 x SPI 3 x I^2C	6x UART 5x SPI 6x I^2C	2x UART 3x SPI 2x I^2C
Consumo de corriente	1A	2A	2.5A	3A	2.5A
Frecuencia del CPU	1GHz	667MHz	120 MHz	1.5GHz	1.4GHz
RAM	512MB DDR3	512 MB DDR3	32 MB SDRAM	2GB LPDDR4	512MB LPDDR2
microSD	Si	Si	Si	Si	Si

Nota. [a]Recuperado de Coley (2014) por BeagleBoard.org Foundation. [b]Recuperado de Digilent (2022) por Digilent. [c]Recuperado de Embedded Artists AB (s.f.) por Embedded Artists AB. [d]Recuperado de Raspberry Pi Foundation (s.f.) por Raspberry Pi Foundation. [e]Recuperado de Raspberry Pi Foundation (s.f.) por Raspberry Pi Foundation.

En lo que respecta a las interfaces de comunicación, todas las plataformas poseen al menos dos interfaces de hardware disponibles para cada protocolo de comunicación (UART, SPI e I^2C), siendo la Raspberry Pi 4B la que más interfaces tiene. Pero de la misma manera, esta plataforma es la que más corriente consume, mientras que la BeagleBone Black es la mejor en este aspecto.

Por otro lado, se puede considerar que los precios de la BeagleBone Black, Raspberry Pi 4B y Raspberry Pi 3A+ son similares y económicos a comparación del precio de la

plataforma Cora Z7-07S y LPC4088 Developer's Kit. En este proyecto se ha dado prioridad al consumo de corriente debido a su importancia en una aplicación espacial, por lo que se seleccionó al mejor candidato, el computador embebido BeagleBone Black.

Sensores, módulos y componentes externos

Unidad de medición inercial (IMU) MPU6050

En el caso de la aplicación para monitorear la inercia del nanosatélite se utilizó una unidad de medición inercial (IMU), es decir, un dispositivo electrónico que mide las velocidades angulares y aceleraciones en los tres ejes (X, Y, Z). En este caso en particular se seleccionó al IMU MPU6050 debido a que ya se tiene experiencia trabajando con el sensor.

El sensor funciona con un voltaje de alimentación de 2.375V-3.46V y se comunica mediante protocolo I^2C. Las características técnicas del sensor se pueden leer en la Tabla 4:

Tabla 4
Características técnicas de la unidad de medición inercial MPU6050

	Acelerómetro	**Giroscopio**	**Termómetro**
Rango	± 2g, ± 4g, ± 8g, ± 16g	± 250º/s, ± 500º/s, ± 1000º/s, ± 2000º/s	-40 ºC a +85 ºC
Sensibilidad	16,384 LSB/(g), 8,192LSB/(g), 4,096 LSB/(g), 2,048 LSB/(g)	131 LSB/(º/s), 65.5 LSB/(º/s), 32.8 LSB/(º/s), 16.4 LSB/(º/s)	340 LSB/ºC
Precisión	-	-	± 1 ºC

Sensor de temperatura AHT10

En el caso de la aplicación para monitorear la temperatura del satélite, se decidió utilizar un sensor que se comunique también mediante protocolo I^2C[1]. En este proyecto se seleccionó el sensor de humedad y temperatura AHT10. Sensor de bajo costo que funciona con un voltaje de alimentación de 1.8V-3.6V. Es necesario resaltar que, de acuerdo con la documentación

[1] Véase el Apéndice A para más información sobre el protocolo de comunicación I^2C

técnica de este sensor, el dispositivo necesita un bus I^2C independiente, es decir, no puede haber más dispositivos conectados. Las características técnicas del sensor se pueden observar en la Tabla 5.

Tabla 5
Características técnicas del sensor de temperatura y humedad AHT10

	Temperatura	**Humedad**
Rango	-40 ºC a 85 ºC	0 a 100 % de humedad relativa
Resolución	0.01 ºC	0.024 %RH
Precision	±0.3 ºC	±2 %RH

Sensor de presión con altímetro MPL3115A2

Para la aplicación de monitoreo de altura, se decidió utilizar un sensor de presión con altímetro que se comunique por I^2C. Debido a la disponibilidad y a sus características, se seleccionó el sensor MPL3115A2. Este es un sensor presión absoluta que incorpora capacidades de cálculo interno para convertir la presión medida en altitud a partir de una fórmula matemática minimizando la carga de cálculo al sistema principal. El voltaje de alimentación de este dispositivo es de 1.62V-3.6V. Sus características principales se resumen en la Tabla 6.

Tabla 6
Características técnicas del sensor de presión con altímetro MPL3115A2

	Altura	**Presión**	**Termómetro**
Rango	–698 to 11,775 m	20 – 110 kPA	-40 ºC a +85 ºC
Sensibilidad	0.3 m	1.5 Pa	-
Precision	-	± 0.4 kPa	± 1 ºC

Modulo Transceptor SX1278

Para establecer la comunicación RF con la estación terrestre y enviar la telemetría, se seleccionó el módulo transceptor SX1278. Este módulo sirve como equipo comunicador de los dos equipos terminales tanto del OBC como de la estación terrena. Este dispositivo utiliza el protocolo de comunicación inalámbrico LoRa RF[2] para enviar paquetes de hasta 256 bytes mediante radio frecuencia. En tanto que la comunicación con el equipo terminal la realiza mediante protocolo serial SPI[3]. El módulo funciona con un voltaje de alimentación de 1.8V-3.7V En la Tabla 7 se pueden observar las características técnicas del dispositivo.

Tabla 7
Características técnicas del módulo transceptor SX1278

Parámetro	Valor
Rango de frecuencia	137 - 525 MHz
Ancho de banda	7.8 - 500 kHz
Tasa de bits efectiva	.018 - 37.5 kbps
Sensibilidad	Hasta -148 dBm

Microcontrolador ATmega328P

Debido a que existieron problemas de conexión entre el computador embebido y el transceptor mediante el protocolo SPI, que no pudieron ser resueltas porque había que hacer cambios en el BSP del sistema operativo, se propuso utilizar un conversor de protocolos. Para ello se optó por un dispositivo microcontrolador que fue utilizado como un módulo conversor de protocolos I^2C a SPI, entre el equipo terminal (OBC) y el transceptor LoRa. De esta manera el el computador embebido envía la información de telemetría mediante protocolo I^2C al microcontrolador, quien convierte la información y la envía al transceptor mediante protocolo

[2] Véase el Apéndice B para más información sobre el protocolo de comunicación LoRa RF
[3] Véase el Apéndice A para más información sobre el protocolo de comunicación SPI

SPI. En este caso, se decidió trabajar con el microcontrolador ATmega328P de 8 bits de alto rendimiento cuyas características técnicas se presentan en la Tabla 8.

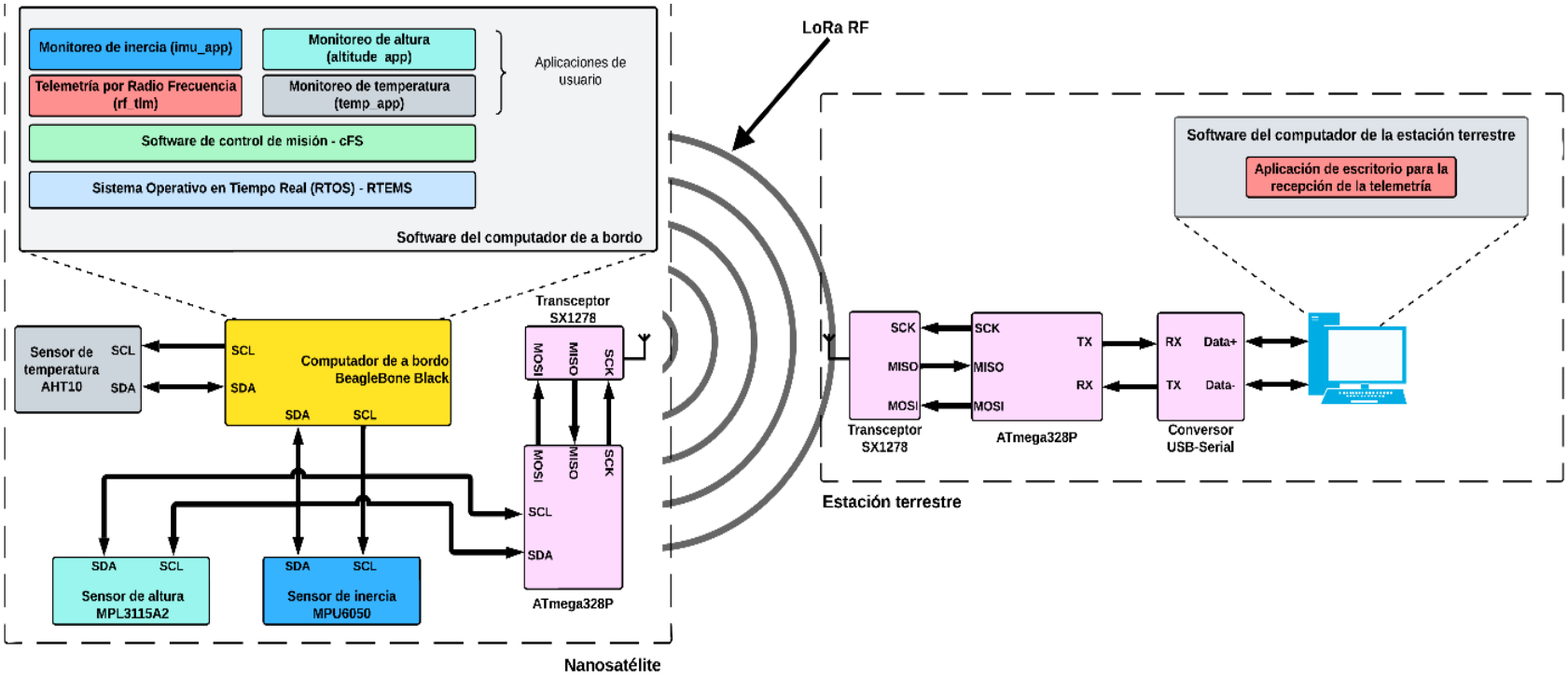
Tabla 8
Características técnicas del microcontrolador ATmega328P

Parámetro	Valor
Voltaje de alimentación	2.7V – 5.5V
Arquitectura	RISC
Memoria de programa	32 KB
Memoria EEPROM	1 KB
SPI	Si
I^2C	Si

Una vez seleccionados el sistema operativo en tiempo real, el sistema de control de misión, el computador embebido y los diferentes sensores, módulos y componentes que serán parte de la plataforma de emulación. Cuyos esquemas tanto de software como de hardware se ilustran en la Figura 8. En cuanto a la implementación del sistema utilizando los diferentes componentes seleccionados, la misma será detallada en el capítulo 4.

Figura 8

Diagrama de bloques de la propuesta de diseño considerando el software y hardware seleccionado



Capítulo 4: Implementación

Una vez establecida la plataforma de desarrollo seleccionada, se procedió a configurar el sistema embebido que emulará las funciones básicas de un computador de a bordo para un nanosatélite. Para ello se instaló y desarrolló el sistema operativo, el sistema control de emisión y las aplicaciones sobre el computador embebido BeagleBone Black utilizando las herramientas y frameworks correspondientes. Adicionalmente se diseñó e implementó la aplicación de escritorio para la estación terrestre. Finalmente, se configuró y conectó todos los sensores y transceptores necesarios para la emulación del sistema satélite-estación terrestre. En este capítulo se incluyen los detalles más destacados de la implementación, entre los que se incluye el proceso de instalación del sistema y diseño de las diferentes aplicaciones. El proceso de instalación del sistema operativo en tiempo real y el sistema de control de misión queda recogido en el manual de instalación, véase el Apéndice C. Adicionalmente, el mismo apéndice detalla la plataforma de desarrollo que se utilizó para la programación, compilación y construcción de las diferentes aplicaciones.

Sistema operativo en tiempo real en el computador embebido

Instalación

Para la instalación del sistema operativo en tiempo real se necesitan al menos dos recursos en particular:

- RTEMS Source Builder (RSB; el nombre se puede traducir como “Constructor de Recursos de RTEMS”)
- Kernel de RTEMS

Es importante considerar que sobre el sistema operativo se ejecutará el sistema de control de misión cFS y por tanto se deben considerar los requerimientos del cFS para interactuar con el hardware a través del sistema operativo. En este contexto, es necesario

resaltar que la distribución del sistema de control de misión cFS incluye aplicaciones ejemplo que permiten recibir comandos y enviar telemetría mediante protocolo UDP/IP para realizar las pruebas funcionales del sistema. Es por esta razón que, para la instalación de RTEMS fue necesario también las librerías de protocolo de red incluidas en un tercer recurso:

- LibBSD para RTEMS

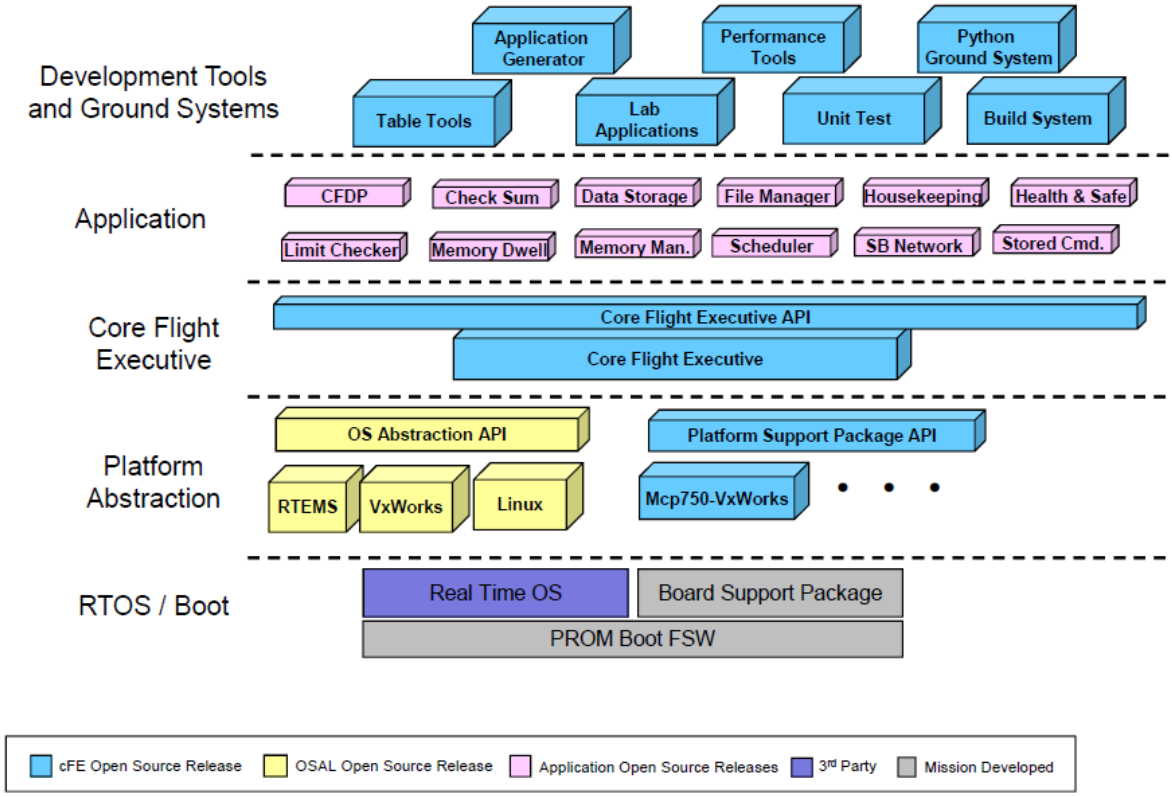
Una vez establecidos los recursos necesarios para la instalación de RTEMS se procedió a instalar el sistema operativo. Primero, mediante el uso del RSB, se instalaron las herramientas necesarias para construir el RTOS. Posterior a ello se utilizaron los recursos del kernel de RTEMS para construir la capa de software que permite al RTOS trabajar sobre la plataforma de hardware, lo que se conoce como Board Support Package (BSP; el nombre se puede traducir como “Paquete de Soporte de Placa”). Por último, se procedió a instalar el paquete LibBSD, la librería que proporciona las funciones necesarias para que la plataforma se comunique con el segmento terrestre mediante protocolos de red.

Sistema de control de misión

Una vez instalado el sistema operativo, se procedió con la instalación del cFS como se indica en el manual de instalación (Apéndice A). Previo el desarrollo de las diferentes aplicaciones de usuario, es primordial entender la arquitectura y funcionamiento del sistema de control de misión. De manera general, el cFS tiene una arquitectura compuesta por capas. En la Figura 9 se observa la interacción de las diversas capas con otros componentes de software dentro de un sistema integral que utiliza el cFS como sistema de control de misión.

Figura 9

Capas de la arquitectura de software utilizando el cFS como sistema de control de misión



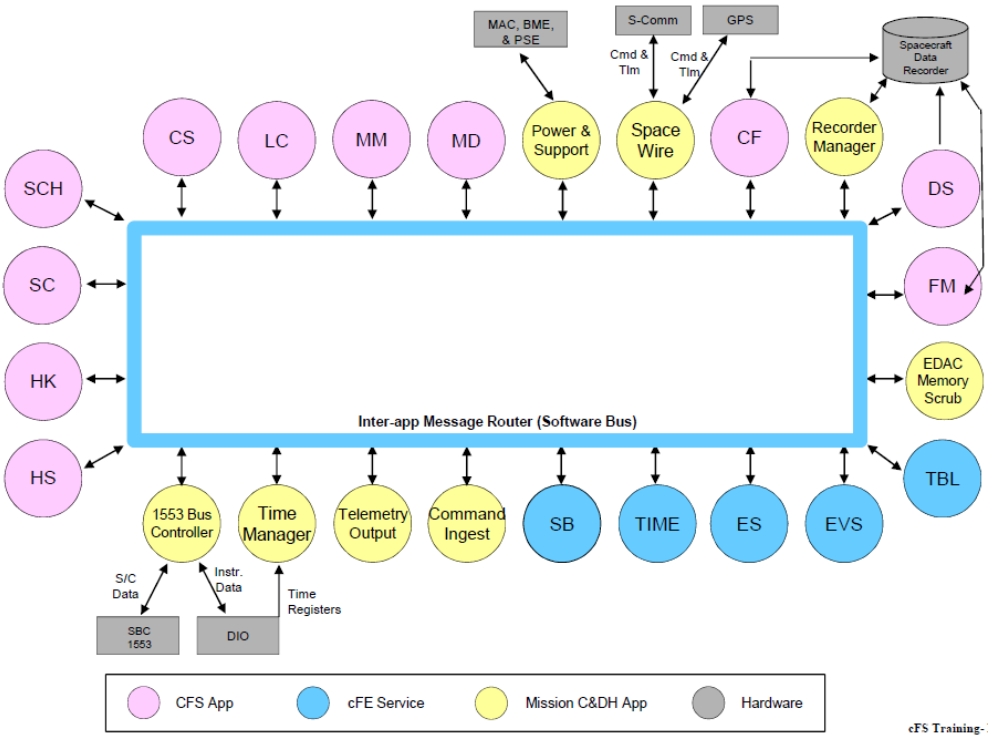
Nota. Adaptado de *cFS Architecture Layers,* por NASA, 2020, NASA Technical Reports Server (https://ntrs.nasa.gov/citations/20210022378)

De abajo hacia arriba, la primera capa está compuesta por el sistema operativo en tiempo real y con su respectivo BSP los cuales, como se mencionó previamente, son la interfaz entre la plataforma de hardware y el sistema de control de misión. La segunda capa corresponde a la plataforma de abstracción en donde se puede encontrar la Operative System Abstraction Layer (OSAL; el nombre se puede traducir como “Capa de Abstracción del Sistema Operativo”) y el Platform Support Package (PSP; el nombre se puede traducir como “Paquete de Soporte de Plataforma”). Estas son librerías de software que proporcionan a la siguiente capa, el cFE (Core Flight Executive), tanto de un API de soporte para el sistema operativo y como de un API para la plataforma de hardware, respectivamente.

La tercera capa es el cFE, el cual, es un framework que no depende de la plataforma de hardware y que proporciona servicios a las diferentes aplicaciones de usuario, por ejemplo: servicios de tiempo, bus de software (SB), entre otros. La cuarta capa corresponde a las aplicaciones de usuario mismas que son dependientes de la misión. La última capa son las herramientas de desarrollo y sistemas de software utilizados por el segmento terrestre.

Figura 10

Representación gráfica del bus de software y su conexión con aplicaciones y servicios del cFS



Nota. Adaptado de *Mission Application Example,* por NASA, 2020, NASA Technical Reports Server (https://ntrs.nasa.gov/citations/20210022378)

Al utilizar al cFS como sistema de control de misión, como se observa en la Figura 10, todos los servicios proporcionados por el cFE (en azul), las aplicaciones de usuario (en rosado) y las aplicaciones encargadas del manejo de comandos y datos (en amarillo) están conectadas entre sí a través del bus de software. El SB es uno de los servicios más útiles que proporciona el cFE, ya que permite la comunicación a través de mensajes entre aplicaciones basándose en

un modelo de publicación/suscripción. Es decir, todos los mensajes son identificados a través de un ID único, el cual es especificado por el emisor cuando envía el mensaje (publicación). Por otro lado, para que una aplicación pueda recibir los mensajes debe cumplir con dos requisitos, el primero, debe especificar los IDs de los mensajes que desea (suscripción) para que el SB le envíe la información que solicita, en otras palabras, la información a la que se ha suscrito. Y segundo, la aplicación suscriptora debe crear un canal al que lleguen los mensajes solicitados.

Es importante notar que el bus de software del cFE utiliza un mecanismo de direccionamiento de difusión, esto quiere decir que los mensajes publicados son enviados a todas las aplicaciones, pero solo aquellas que se suscriban recibirán el mensaje. Esto a su vez permite que la comunicación entre aplicaciones sea una a una, una a muchas, muchas a una o muchas a muchas.

El cFS ha definido que los mensajes pueden ser de dos tipos en particular, mensajes de comando y mensajes de telemetría. La diferencia radica en que, los primeros sirven para que las aplicaciones suscriptoras ejecuten ciertas acciones cuando la aplicación emisora del mensaje lo solicita, como por ejemplo leer los datos de un sensor. Por otra parte, los mensajes de telemetría son aquellos que transportan información de la aplicación como, por ejemplo: el valor del contador de errores, datos de los sensores, entre otros. En el caso del cFS, los mensajes tienen IDs conformados por 2 bytes, de los cuales, el primero permite identificar el tipo de mensaje y el segundo es definido por la aplicación. En el caso de los mensajes de comando, los IDs serán 0x18YY y en los mensajes de telemetría serán 0x08YY.

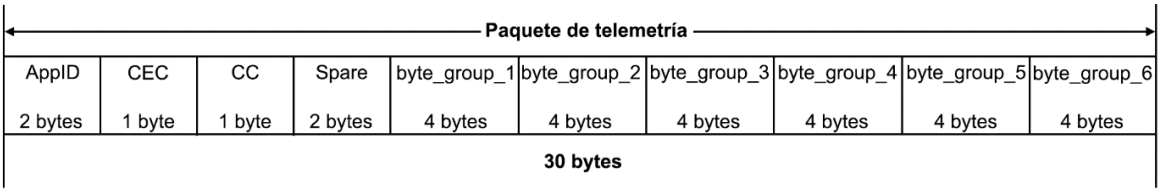
En la siguiente sección se exponen las diferentes aplicaciones de usuario elaboradas que son ejecutadas sobre el sistema de control de misión cFS.

Aplicaciones de usuario

Para el diseño de las aplicaciones de usuario, se consideró algunas características específicas de los módulos sensores seleccionados. Así, por ejemplo, tomando en cuenta que la unidad de medición inercial IMU MPU6050 y el sensor de altura MPL3115A2 tienen la capacidad de medir la temperatura del dispositivo, se diseñó que las respectivas aplicaciones de usuario para el monitoreo de inercia y el monitoreo de altitud monitoricen también las temperaturas de sus dispositivos. De esta manera esta información complementará la medición de temperatura ambiental realizada mediante el sensor AHT10.

Otra consideración previa al diseño hace referencia a la estandarización del mensaje de telemetría RF enviado por parte de las diferentes aplicaciones de monitoreo: inercia, altitud y temperatura. Para ello en este proyecto se estableció paquetes de telemetría con un tamaño de 30 bytes. Cada paquete tiene una estructura como la de la Figura 11.

Figura 11
Estructura del paquete de telemetría



La estructura del paquete de telemetría consiste en 10 campos en la siguiente secuencia:

- AppID: 2 bytes que sirven para identificar a la aplicación que envía el paquete.
- CEC: 1 byte en el que se registra el contador de errores registrados por la aplicación.
- CC: 1 byte en el que se registra el contador de comandos recibidos por la aplicación.

- Spare: 2 bytes vacíos
- byte_group_1: 4 bytes en los que la aplicación puede registrar información.
- byte_group_2: 4 bytes en los que la aplicación puede registrar información.
- byte_group_3: 4 bytes en los que la aplicación puede registrar información.
- byte_group_4: 4 bytes en los que la aplicación puede registrar información.
- byte_group_5: 4 bytes en los que la aplicación puede registrar información.
- byte_group_6: 4 bytes en los que la aplicación puede registrar información.

Scheduler (SCH; el nombre se puede traducir como "Planificador"). Esta aplicación viene como parte de la distribución del cFS. Es la encargada de enviar mensajes a través del bus de software periódicamente, con un intervalo de tiempo prestablecido. En este proyecto el intervalo es de 4 segundos. Los mensajes llevan comandos hacia las aplicaciones suscritas para que ejecuten alguna acción en concreto. Por ejemplo, enviar la telemetría o leer los datos de un sensor.

Aplicación para el monitoreo de inercia (IMU_APP). Esta aplicación fue desarrollada bajo la estructura de las aplicaciones de usuario del cFS. El objetivo principal de esta aplicación es adquirir la aceleración y velocidad angular en los tres ejes, datos que son proporcionados por el sensor de inercia. La información es enviada por paquetes de telemetría de cada que el SCH lo solicite.

Con respecto a los paquetes de telemetría de esta aplicación, estos están codificados como se indica en la Tabla 9.

Tabla 9
Codificación del paquete de telemetría para la aplicación de monitoreo de inercia

Estructura del paquete	Variable	Tipo de variable
AppID	0x08E1	palabra
CEC	Contador de errores registrados por la aplicación	byte
CC	Contador de comandos recibidos por la aplicación	byte
Spare	Vacío	-
byte_group_1	Aceleración en X	float
byte_group_2	Aceleración en Y	float
byte_group_3	Aceleración en Z	float
byte_group_4	Velocidad angular en X	float
byte_group_5	Velocidad angular en Y	float
byte_group_6	Velocidad angular en Z	float

La implementación de la aplicación se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 12 y Figura 13. La aplicación consta de dos partes principales, la primera que es la etapa de configuración e inicialización de la aplicación y la segunda que constituye el lazo principal. La primera parte se encuentra descrita a continuación:

1. Se inicializan las variables globales de la aplicación y la estructura de datos IMU_APP_Data.
2. Se inicializa el mensaje (HkTlm, ID = 0x08E1) mediante el cual se va a enviar los datos de inercia a la aplicación de telemetría por UDP/IP.
3. Se inicializa el mensaje (OutData, ID = 0x08E2) mediante el cual se va a enviar los datos de inercia a la aplicación de telemetría por RF.
4. Se inicializa el mensaje (TempData, ID = 0x08E3) mediante el cual se va a enviar la temperatura registrada por el sensor a la aplicación de temperatura.
5. Se crea el canal para recibir comandos enviados por el SCH a través del SB y se suscribe a los mensajes de la Tabla 10.

Figura 12
Diagrama de flujo de la aplicación imu_app, parte 1

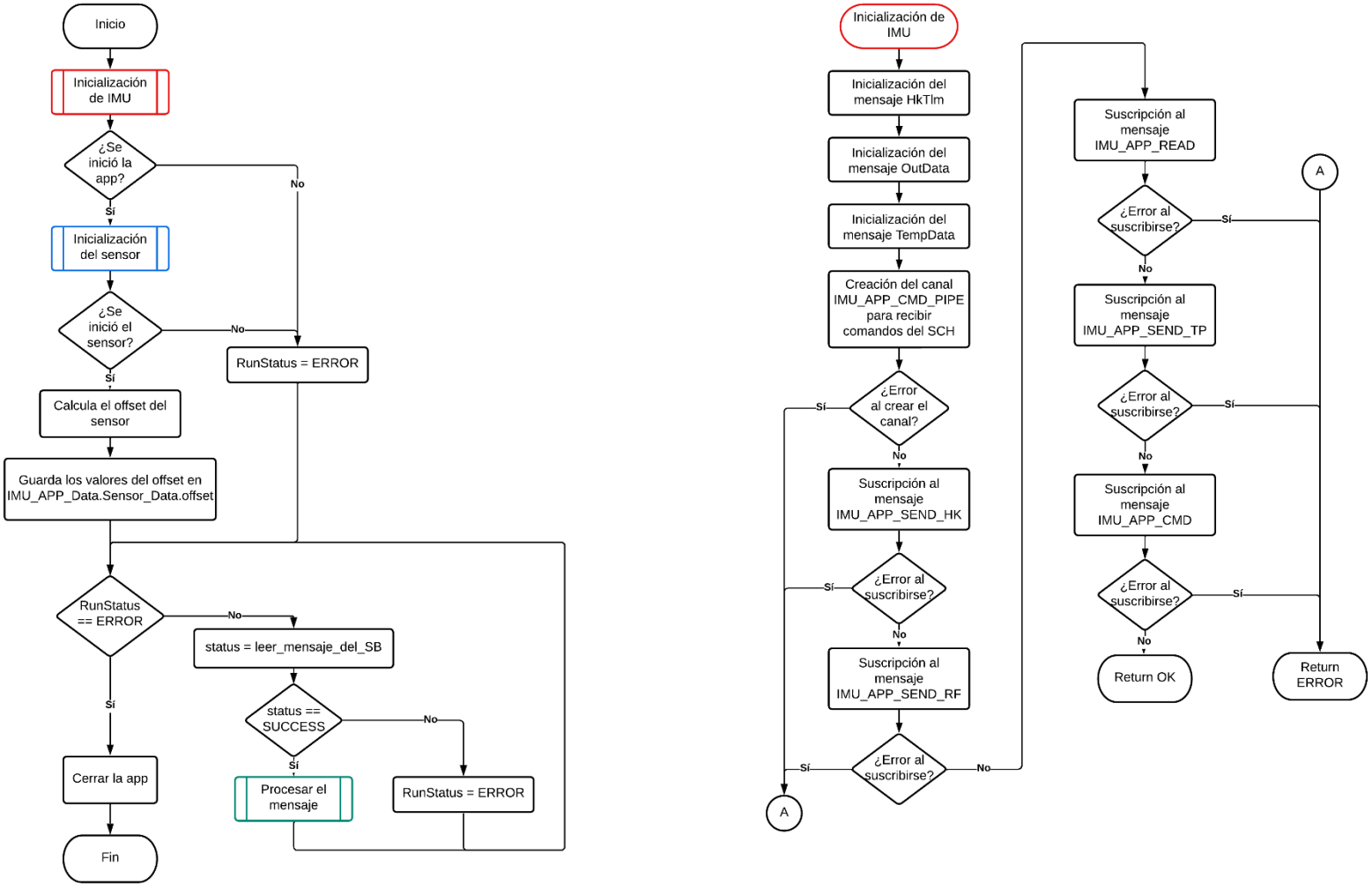
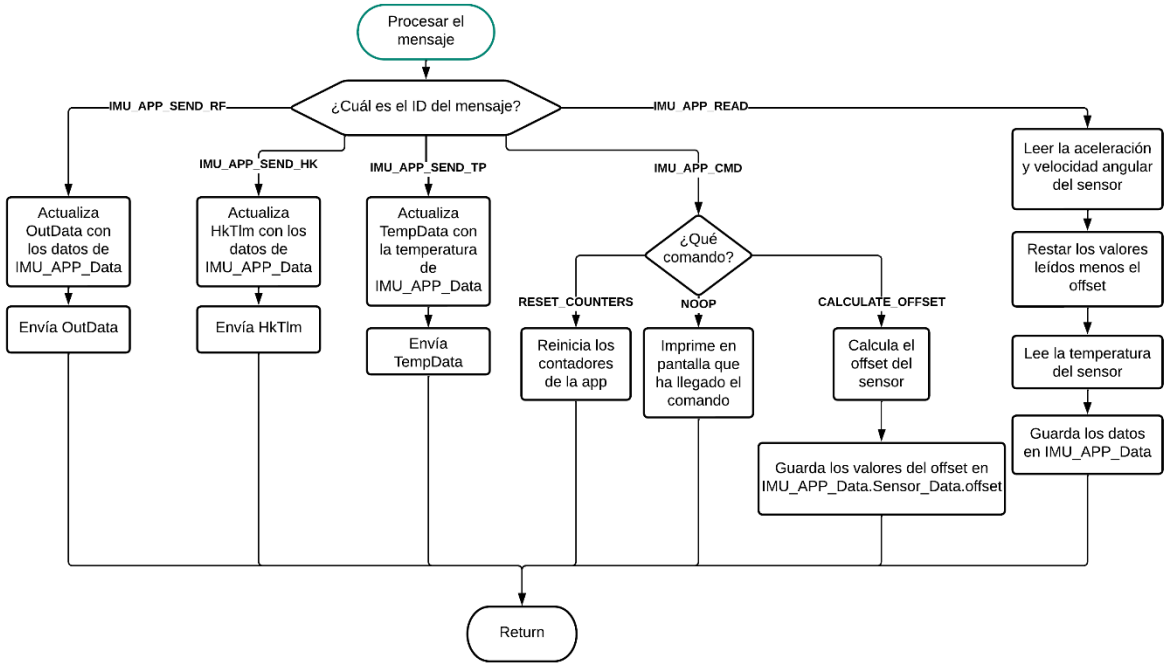


Figura 13
Diagrama de flujo de la aplicación imu_app, parte 2



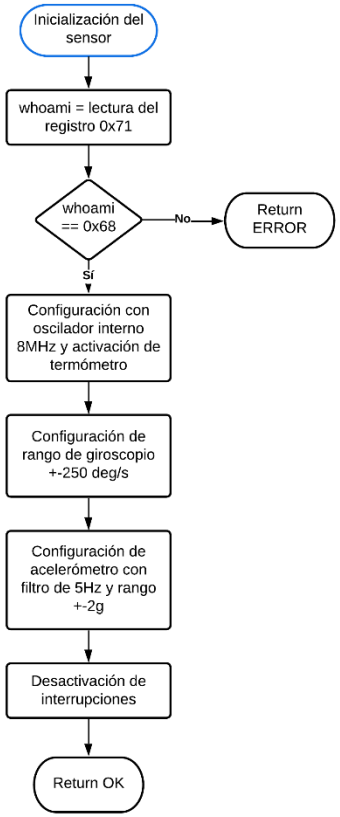


Tabla 10
Mensajes a los que se suscribe la app de monitoreo de inercia

Mensaje	ID	Función
IMU_APP_CMD_MID	0x18E0	Mensaje a través del cual llegan los comandos de la estación terrestre
IMU_APP_SEND_HK_MID	0x18E1	Enviar telemetría a la app de telemetría por UDP/IP
IMU_APP_SEND_RF_MID	0x18E2	Enviar telemetría a la app de telemetría por RF
IMU_APP_SEND_TP_MID	0x18E3	Enviar la temperatura a la app de monitoreo de temperatura
IMU_APP_READ_MID	0x18E4	Leer los datos del sensor de inercia

6. Si no hubo errores al suscribirse a los mensajes y al crear el canal del SB, la aplicación inicializa el sensor MPU6050. Caso contrario escribe en consola que ha habido un error y cierra la aplicación. En caso de que exista un error al inicializar el sensor, de igual manera, se escribirá en consola un mensaje indicando el error y posterior a ello cierra la aplicación.
7. A través de la obtención de un promedio de mediciones de los valores de las aceleraciones y velocidades angulares del sensor se calcula el sistema de referencia inercial que servirá para la determinación de reposo o movimiento del sistema. Estos valores serán almacenados en la estructura de datos IMU_APP_Data.Sensor_Data.offset.

El lazo principal espera indefinidamente hasta que llegue un mensaje. Dependiendo del mensaje recibido se realiza las siguientes acciones:

1. Leer (mensaje IMU_APP_READ_MID) :
   i. La aplicación lee los datos del acelerómetro y del giroscopio.
   ii. Calcula los valores reales de acuerdo con la Ecuación 1.

$$Dato\ real = Dato\ leído - Offset \qquad (1)$$

iii. Lee la temperatura del sensor.

iv. Actualiza la información en IMU_APP_Data.

2. Enviar Hk (mensaje IMU_APP_SEND_HK_MID):

   i. Actualiza el mensaje HkTlm con los datos de IMU_APP_Data.

   ii. Construye y envía el mensaje HkTlm a través del SB.

3. Enviar RF (mensaje IMU_APP_SEND_RF_MID):

   i. Actualiza el mensaje OutData con los datos de IMU_APP_Data.

   ii. Construye y envía el mensaje OutData a través del SB.

4. Enviar Temperatura (mensaje IMU_APP_SEND_TP_MID):

   i. Actualiza el mensaje TempData con la temperatura guardada en IMU_APP_Data.

   ii. Construye y envía el mensaje TempData a través del SB.

5. Comando (mensaje IMU_APP_CMD_MID), identifica si es uno de los siguientes comandos:

   i. Comando NoOp. No ejecuta una operación, solo escribe en consola que ha recibido el mensaje.

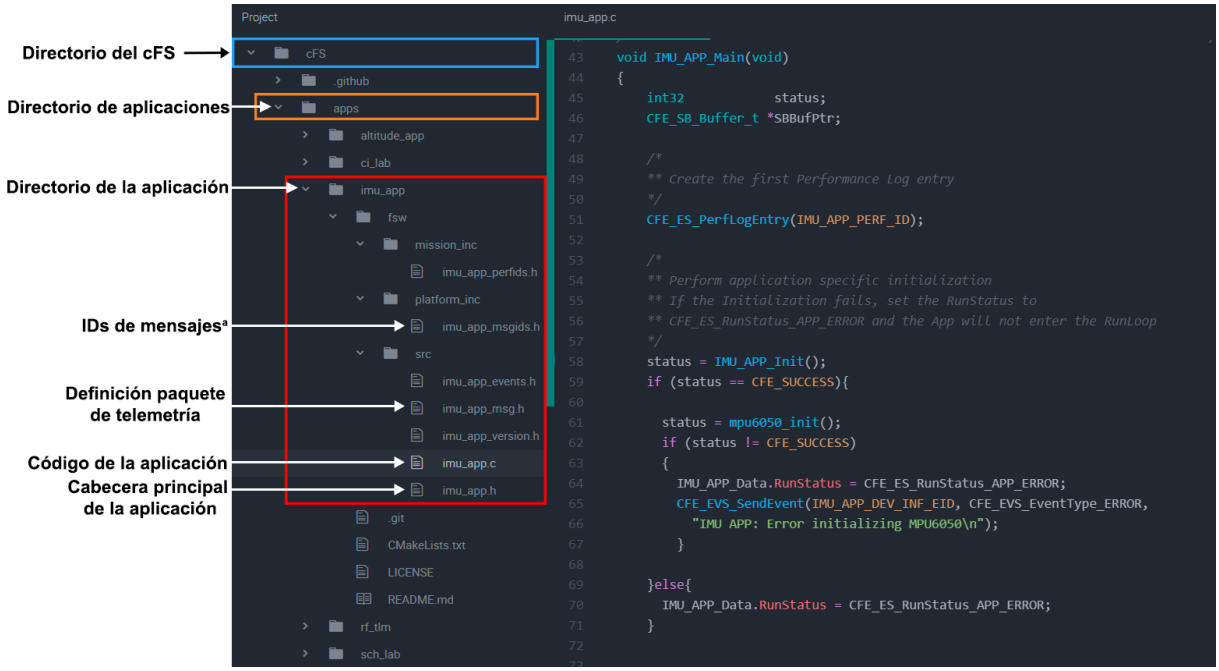
   ii. Comando Reiniciar contadores. Reinicia los contadores de la aplicación.

   iii. Comando Calcular offset. Calcula el offset del sensor al realizar el promedio de 500 medidas consecutivas.

La implementación de la aplicación para el monitoreo de inercia se puede observar en la Figura 14.

Figura 14

Implementación de la aplicación para el monitoreo de inercia



Nota. [a]En el archivo terminado en msgids.h se encuentran definidos los IDs de los mensajes que la aplicación envía y de los mensajes a los que se suscribe.

Aplicación para el monitoreo de altitud (ALTITUDE_APP). Esta aplicación de igual forma fue desarrollada bajo la estructura de las aplicaciones de usuario del cFS. El objetivo principal de esta aplicación es adquirir datos del sensor de altitud, en este caso el MPL3115A2 y enviar paquetes de telemetría cada que el SCH lo solicite.

Con respecto a los paquetes de telemetría de esta aplicación, estos están codificados como se indica en la Tabla 11.

Tabla 11
Codificación del paquete de telemetría para la aplicación de monitoreo de altitud

Estructura del paquete	Variable	Tipo de variable
AppID	0x08A1	palabra
CEC	Contador de errores registrados por la aplicación	byte
CC	Contador de comandos recibidos por la aplicación	byte
Spare	Vacío	-
byte_group_1	Altura	float
byte_group_2	Presión a nivel del mar configurada en la app	float
byte_group_3	Offset de altitud configurada en la aplicación	float
byte_group_4	Vacío	-
byte_group_5	Vacío	-
byte_group_6	Vacío	-

La implementación de la aplicación se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 15 y Figura 16. La aplicación consta de dos partes principales, la primera que es la etapa de configuración e inicialización de la aplicación y la segunda que constituye el lazo principal. La primera parte se encuentra descrita a continuación:

1. Se inicializan las variables globales de la aplicación, la estructura de datos ALTITUDE_APP_Data, se establece la presión del nivel del mar en 1013.26 hPa y el offset de altura en 0m.
2. Se inicializa el mensaje (HkTlm, ID = 0x08A1) mediante el cual se va a enviar los datos de altitud a la aplicación de telemetría por UDP/IP.
3. Se inicializa el mensaje (OutData, ID = 0x08A2) mediante el cual se va a enviar los datos de altitud a la aplicación de telemetría por RF.
4. Se inicializa el mensaje (TempData, ID = 0x08A3) mediante el cual se va a enviar la temperatura registrada por el sensor a la aplicación de temperatura.
5. Se crea el canal (ALTITUDE_APP_CMD_PIPE) para recibir comandos enviados por el SCH a través del SB y se suscribe a los mensajes de la Tabla 12.

Figura 15

Diagrama de flujo de la aplicación altitude_app, parte 1

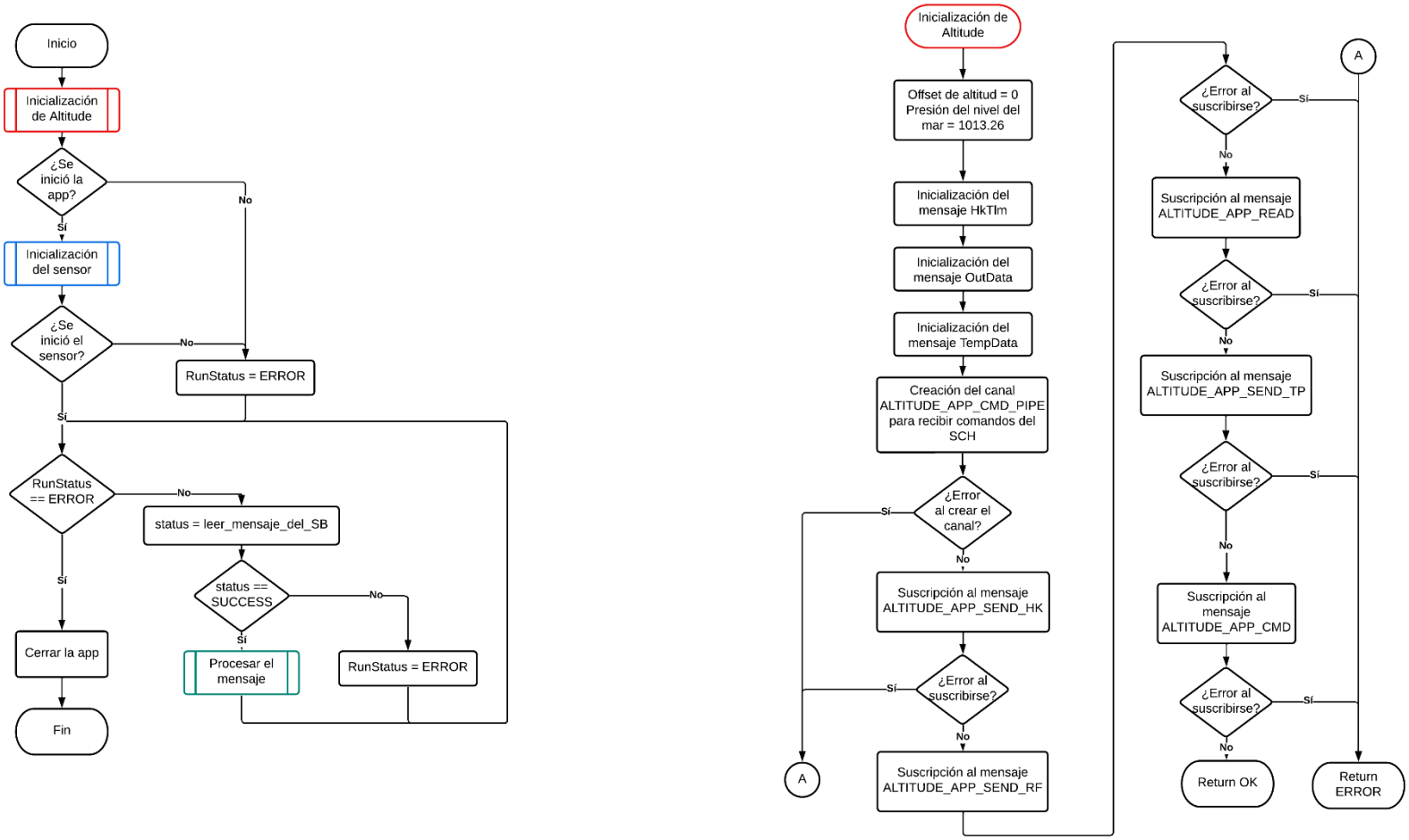
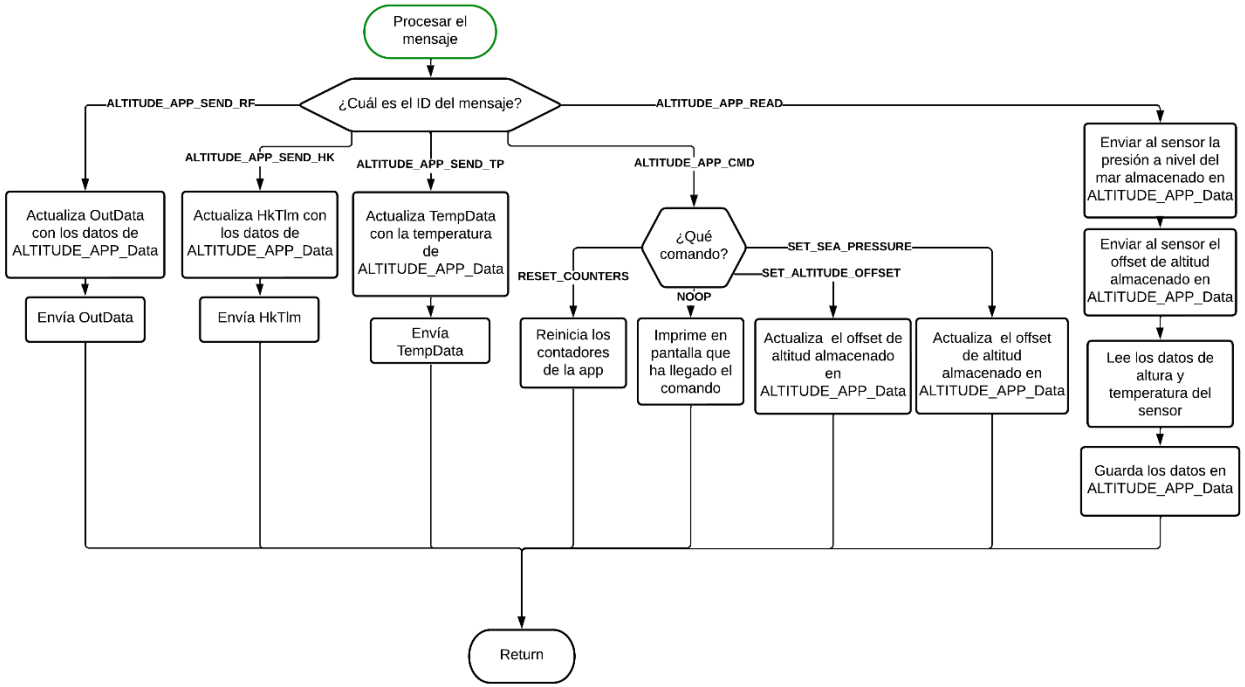


Figura 16
Diagrama de flujo de la aplicación altitude_app, parte 2



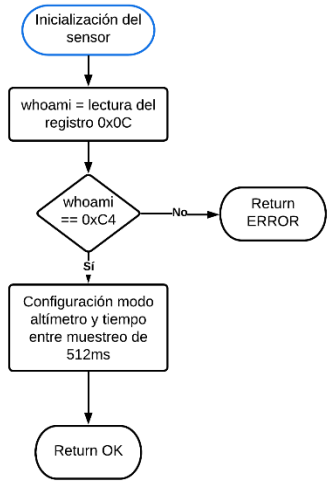


Tabla 12
Mensajes a los que se suscribe la app de monitoreo de altitud

Mensaje	ID	Función
ALTITUDE_APP_CMD_MID	0x18A0	Mensaje a través del cual llegan los comandos de la estación terrestre
ALTITUDE_APP_SEND_HK_MID	0x18A1	Enviar telemetría a la app de telemetría por UDP/IP
ALTITUDE_APP_SEND_RF_MID	0x18A2	Enviar telemetría a la app de telemetría por RF
ALTITUDE_APP_SEND_TP_MID	0x18A3	Enviar la temperatura a la app de monitoreo de temperatura
ALTITUDE_APP_READ_MID	0x18A4	Leer los datos del sensor de altitud

6. Si no hubo errores al suscribirse a los mensajes y al crear el canal del SB, la aplicación inicializa el sensor MPL3115A2. En caso de que exista un error, ya sea al suscribirse a los mensajes, al crear el canal del SB o al inicializar el sensor, se escribirá en consola que ha habido un error y se cerrará la aplicación.

El lazo principal espera indefinidamente hasta que llegue un mensaje. Dependiendo del mensaje recibido se realiza las siguientes acciones:

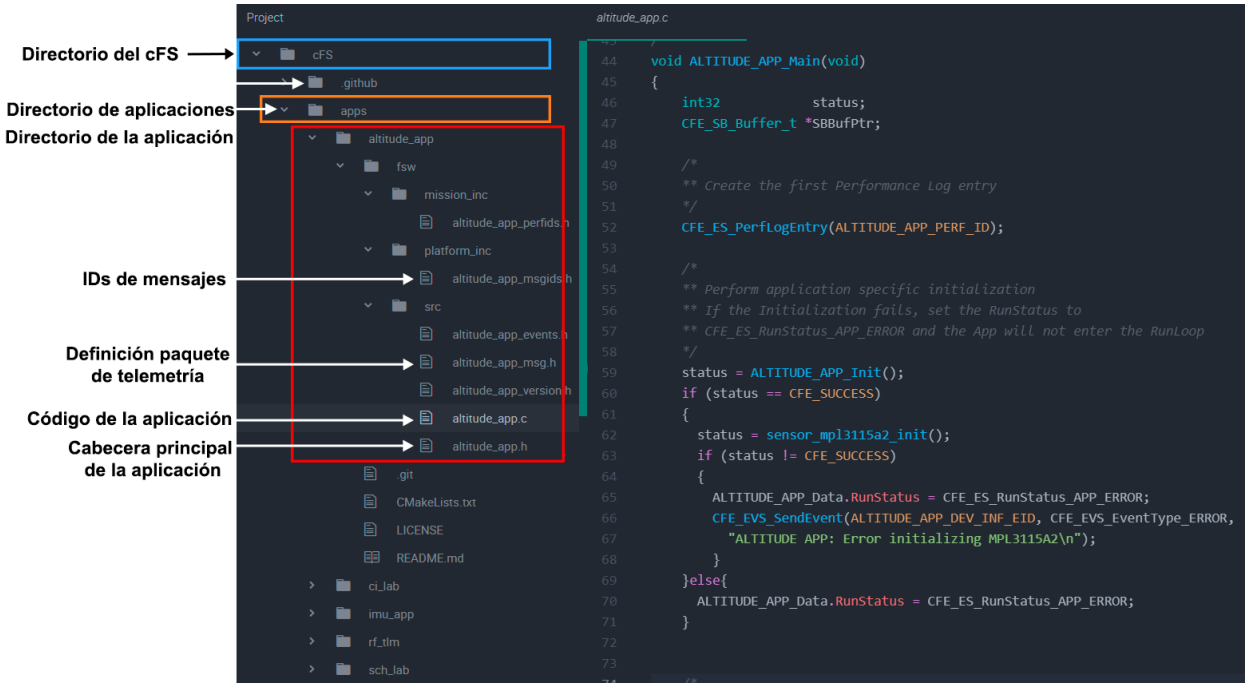
- Leer (mensaje ALTITUDE_APP_READ_MID):
  i. La aplicación envía al sensor la presión a nivel del mar almacenado en ALTITUDE_APP_Data.
  ii. La aplicación envía al sensor el offset de altitud almacenado en ALTITUDE_APP_Data.
  iii. La aplicación lee los datos de altura y temperatura del sensor y los copia a ALTITUDE_APP_Data.
- Enviar Hk (mensaje ALTITUDE_APP_SEND_HK_MID):
  i. Actualiza el mensaje HkTlm con los datos de ALTITUDE_APP_Data.

ii. Construye y envía el mensaje HkTlm a través del SB.

- Enviar RF (mensaje ALTITUDE_APP_SEND_RF_MID):
  i. Actualiza el mensaje OutData con los datos de ALTITUDE_APP_Data.
  ii. Construye y envía el mensaje OutData a través del SB.
- Enviar Temperatura (mensaje ALTITUDE_APP_SEND_TP_MID):
  i. Actualiza el mensaje TempData con la temperatura guardada en ALTITUDE_APP_Data.
  ii. Construye y envía el mensaje TempData a través del SB.
- Comando (mensaje ALTITUDE_APP_CMD_MID), identifica si es uno de los siguientes comandos:
  i. Comando NoOp. No ejecuta una operación, solo escribe en consola que ha recibido el mensaje.
  ii. Comando Reiniciar contadores. Reinicia los contadores de la aplicación.
  iii. Comando Configurar offset de altura. Cambia el valor del offset de altura al valor ingresado por el usuario al enviar el comando, lo almacena en ALTITUDE_APP_Data.
  iv. Comando Configurar presión a nivel del mar. Cambia el valor de la presión a nivel del mar al valor ingresado por el usuario al enviar el comando, lo almacena en ALTITUDE_APP_Data.
  v. Comando Reiniciar offset de altura y presión a nivel del mar. Reinicia el valor del offset de altura y de la presión a nivel del mar a sus valores por defecto, 0m y 1013.26 hPa respectivamente.

La implementación de la aplicación para el monitoreo de altitud se puede observar en la Figura 17.

Figura 17
Implementación de la aplicación para el monitoreo de altitud



Aplicación para el monitoreo de temperatura (TEMP_APP). Esta aplicación, al igual que las descritas previamente fue desarrollada bajo la estructura de las aplicaciones de usuario del cFS. El objetivo principal de esta aplicación es adquirir datos del sensor AHT10, recibir los mensajes con los valores de temperatura enviados por la aplicación de monitoreo de inercia y de altitud, y enviar paquetes de telemetría, cada que el SCH lo solicite.

Es necesario destacar que, con el sensor AHT10 se recomienda que la frecuencia a la que se realizan las mediciones sea mayor a dos segundos, ya que con una frecuencia alta el sensor tiende a calentarse (ASAIR, 2019). Es por esta razón que se implementó un contador que aumenta cada que el SCH solicita la lectura del sensor. Una vez que el contador llegue a 7 se realiza la lectura de temperatura y se reinicia el contador, caso contrario no se lee los datos del sensor. Tomando en consideración que el SCH envía mensajes cada 4 segundos, la lectura de temperatura se realizará aproximadamente cada 30 segundos, previniendo así el calentamiento del sensor.

Con respecto a los paquetes de telemetría de esta aplicación, estos están codificados como se indica en la Tabla 13.

Tabla 13
Codificación del paquete de telemetría para la aplicación de monitoreo de temperatura

Estructura del paquete	Variable	Tipo de variable
AppID	0x08D1	palabra
CEC	Contador de errores registrados por la aplicación	byte
CC	Contador de comandos recibidos por la aplicación	byte
Spare	Vacío	-
byte_group_1	Temperatura registrada por el sensor AHT10	float
byte_group_2	Humedad registrada por el sensor AHT10	float
byte_group_3	Temperatura registrada por el sensor MPU6050	float
byte_group_4	Temperatura registrada por el sensor MPL3115A2	float
byte_group_5	Vacío	-
byte_group_6	Vacío	-

La implementación de la aplicación se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 18 y la Figura 19. La aplicación consta de dos partes principales, la primera que es la etapa de configuración e inicialización de la aplicación y la segunda que constituye el lazo principal. La primera parte se encuentra descrita a continuación:

1. Se inicializan las variables globales de la aplicación, la estructura de datos TEMP_APP_Data y se establece el contador Timecounter en 0.
2. Se inicializa el mensaje (HkTlm, ID = 0x08D1) mediante el cual se va a enviar los datos de temperatura a la aplicación de telemetría por UDP/IP.
3. Se inicializa el mensaje (OutData, ID = 0x08D2) mediante el cual se va a enviar los datos de temperatura a la aplicación de telemetría por RF.
4. Se crea un canal (TEMP_APP_CMD_PIPE) para recibir comandos enviados por el SCH a través del SB.

La aplicación se suscribe a través del SB a los mensajes enviados por el SCH de la Tabla 14.

Figura 18

Diagrama de flujo de la aplicación temp_app, parte 1

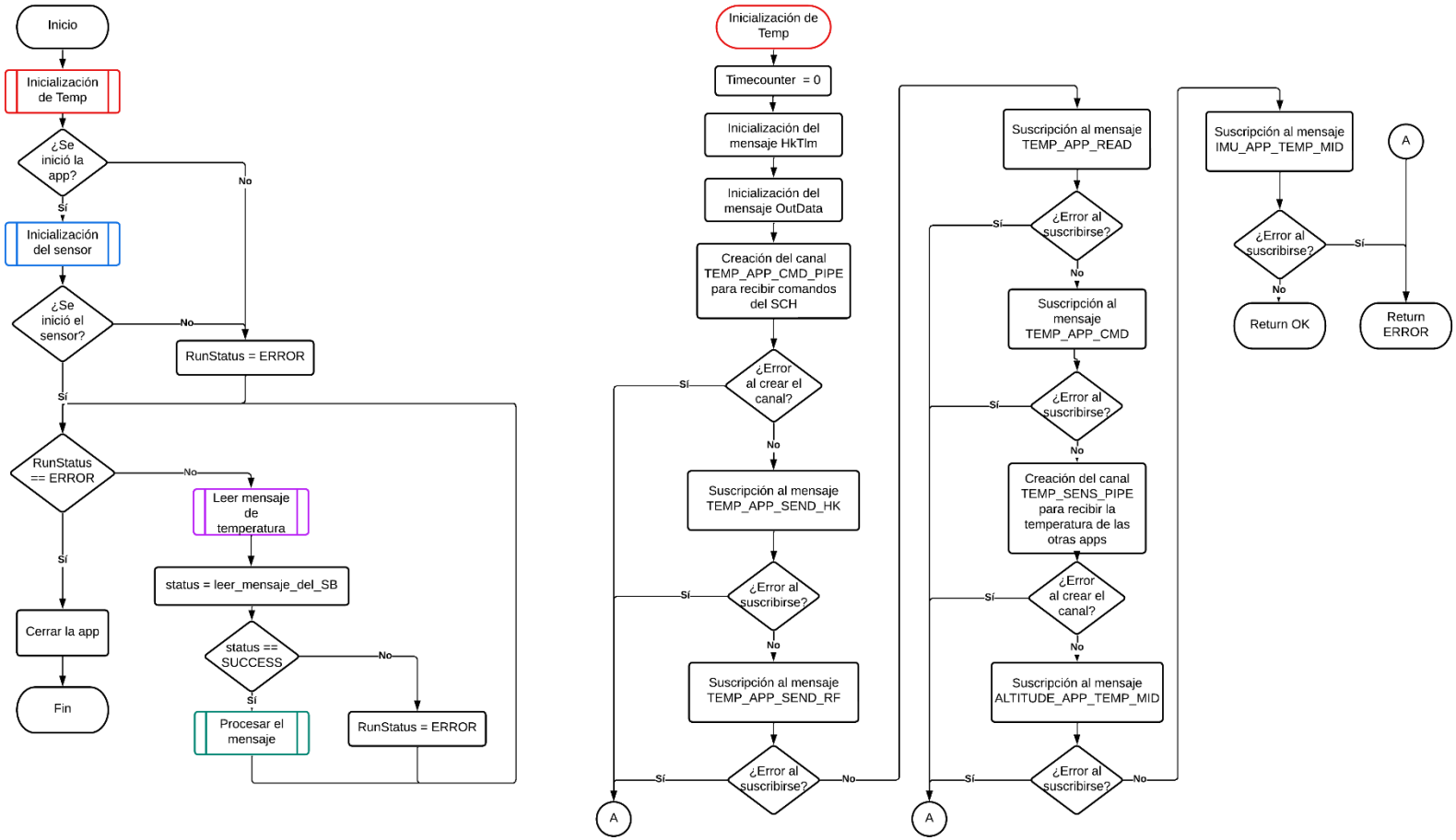


Figura 19
Diagrama de flujo de la aplicación temp_app, parte 2

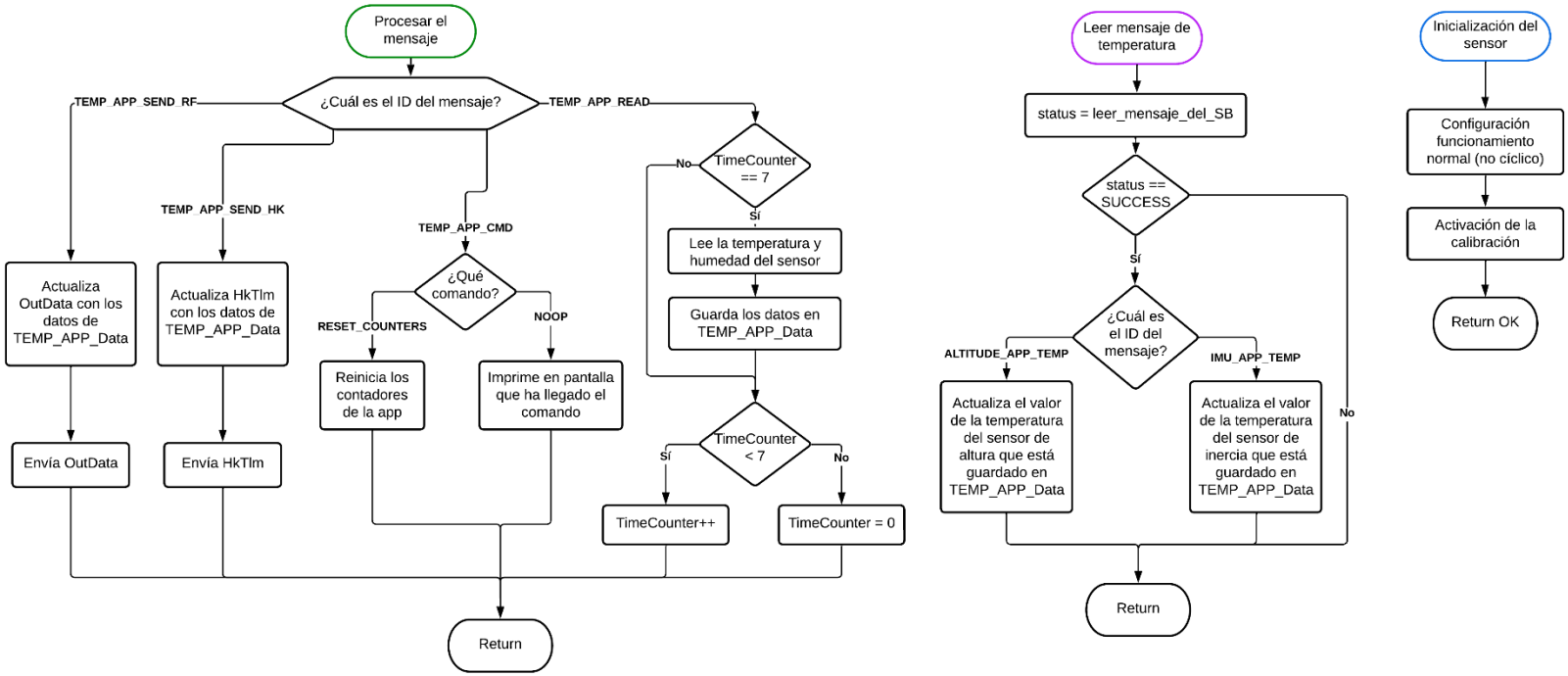


Tabla 14
Mensajes enviados por el SCH a los que se suscribe la app de monitoreo de temperatura

Mensaje	ID	Función
TEMP_APP_CMD_MID	0x18D0	Mensaje a través del cual llegan los comandos de la estación terrestre
TEMP _APP_SEND_HK_MID	0x18D1	Enviar telemetría a la app de telemetría por UDP/IP
TEMP _APP_SEND_RF_MID	0x18D2	Enviar telemetría a la app de telemetría por RF
TEMP _APP_READ_MID	0x18D3	Leer los datos del sensor de temperatura

5. Se crea un canal (TEMP_SENS_PIPE) para recibir los mensajes de temperatura enviados a través del SB por la aplicación de monitoreo de inercia y la aplicación de monitoreo de altitud.
6. La aplicación se suscribe a los mensajes de las aplicaciones de monitoreo de inercia y de altitud como se observa en la Tabla 15.

Tabla 15
Mensajes enviados por otras aplicaciones a los que se suscribe la app de monitoreo de temperatura

Mensaje	ID	Función
ALTITUDE_APP_TEMP_MID	0x08A3	Mensaje a través del cual llega la temperatura registrada por el sensor de altitud
IMU_APP_TEMP_MID	0x08E3	Mensaje a través del cual llega la temperatura registrada por el sensor de inercia

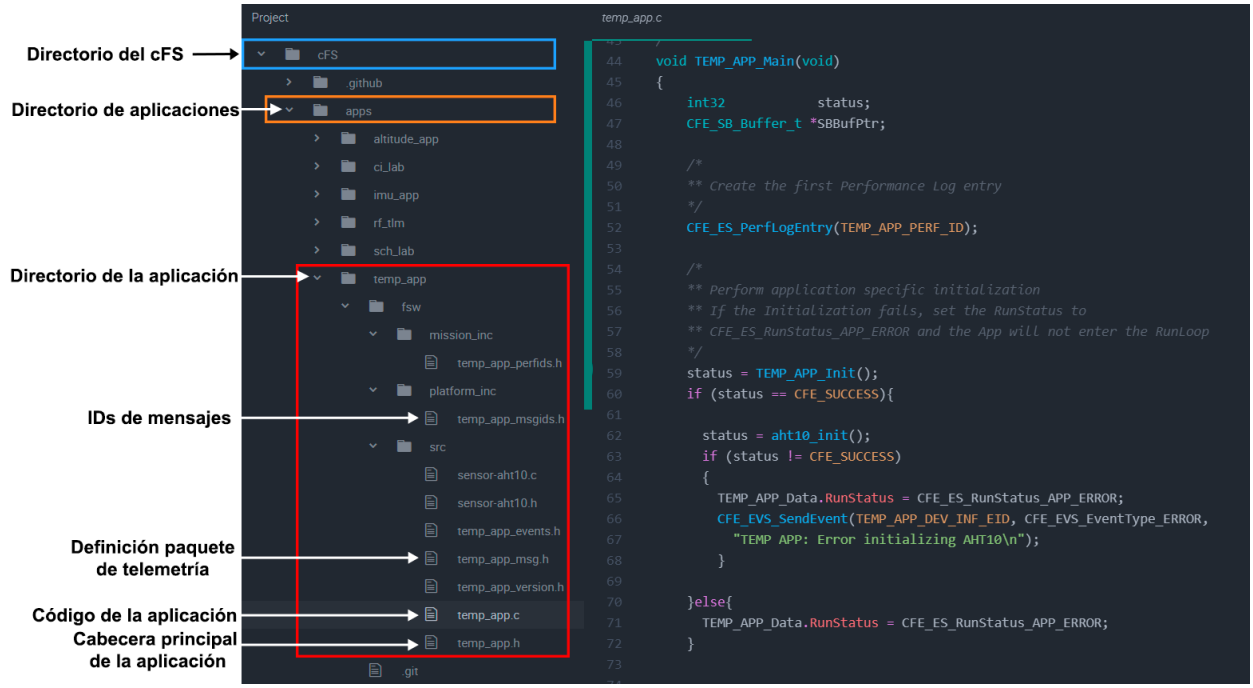
7. Si no hubo errores al suscribirse a los mensajes y al crear los canales del SB, la aplicación inicializa el sensor AHT10. En caso de que exista un error, ya sea al suscribirse a los mensajes, al crear los canales del SB o al inicializar el sensor, se escribirá en consola que ha habido un error y se cerrará la aplicación.

En el lazo principal se realizan las siguientes acciones:

1. La aplicación lee el canal mediante el cual recibe los mensajes de temperatura enviados por las otras aplicaciones (TEMP_SENS_PIPE). Si existe un mensaje, determina que aplicación lo envió y almacena el valor de temperatura en TEMP_APP_Data. Este proceso se repite hasta que ya no haya mensajes en el canal. En el caso de que no lleguen mensajes al canal o se hayan leído todos, se pasa al siguiente punto (punto 2).
2. La aplicación espera indefinidamente hasta que llegue un mensaje enviado por el SCH. Dependiendo del mensaje recibido:
    - Leer (mensaje TEMP_APP_READ_MID):
        i. La aplicación verifica que Timecounter sea igual a 7.
        ii. En caso de que Timecounter sea igual a 7 la aplicación lee el valor de temperatura del sensor. Caso contrario no realiza la lectura.
        iii. La aplicación verifica que Timecounter sea menor a 7, de ser así aumenta el valor de Timecounter en 1. Caso contrario reinicia el valor de Timecounter a 0.
    - Enviar Hk (mensaje TEMP_APP_SEND_HK_MID):
        i. Actualiza el mensaje HkTlm con los datos de TEMP_APP_Data.
        ii. Construye y envía el mensaje HkTlm a través del SB.
    - Enviar RF (mensaje TEMP_APP_SEND_RF_MID):
        i. Actualiza el mensaje OutData con los datos de TEMP_APP_Data.
        ii. Construye y envía el mensaje OutData a través del SB.
    - Comando (mensaje TEMP_APP_CMD_MID), identifica si es uno de los siguientes comandos:
        i. Comando NoOp. No ejecuta una operación, solo escribe en consola que ha recibido el mensaje.
        ii. Comando Reiniciar contadores. Reinicia los contadores de la aplicación.

La implementación de la aplicación para el monitoreo de temperatura se puede observar en la Figura 20.

Figura 20
Implementación de la aplicación para el monitoreo de temperatura



Aplicación de telemetría por RF (rf_tlm). Esta aplicación, al igual que las descritas previamente fue desarrollada bajo la estructura de las aplicaciones de usuario del cFS. El objetivo principal de esta aplicación es recibir los mensajes de telemetría OutData enviados por las aplicaciones de monitoreo de inercia, de altitud y de temperatura. Seguidamente, la aplicación envía los paquetes de telemetría recibidos al sistema de comunicación RF de manera periódica. Es decir, al microcontrolador ATmega328P conectado al transceptor SX1278.

Figura 21
Diagrama de flujo de la aplicación rf_tlm, parte 1

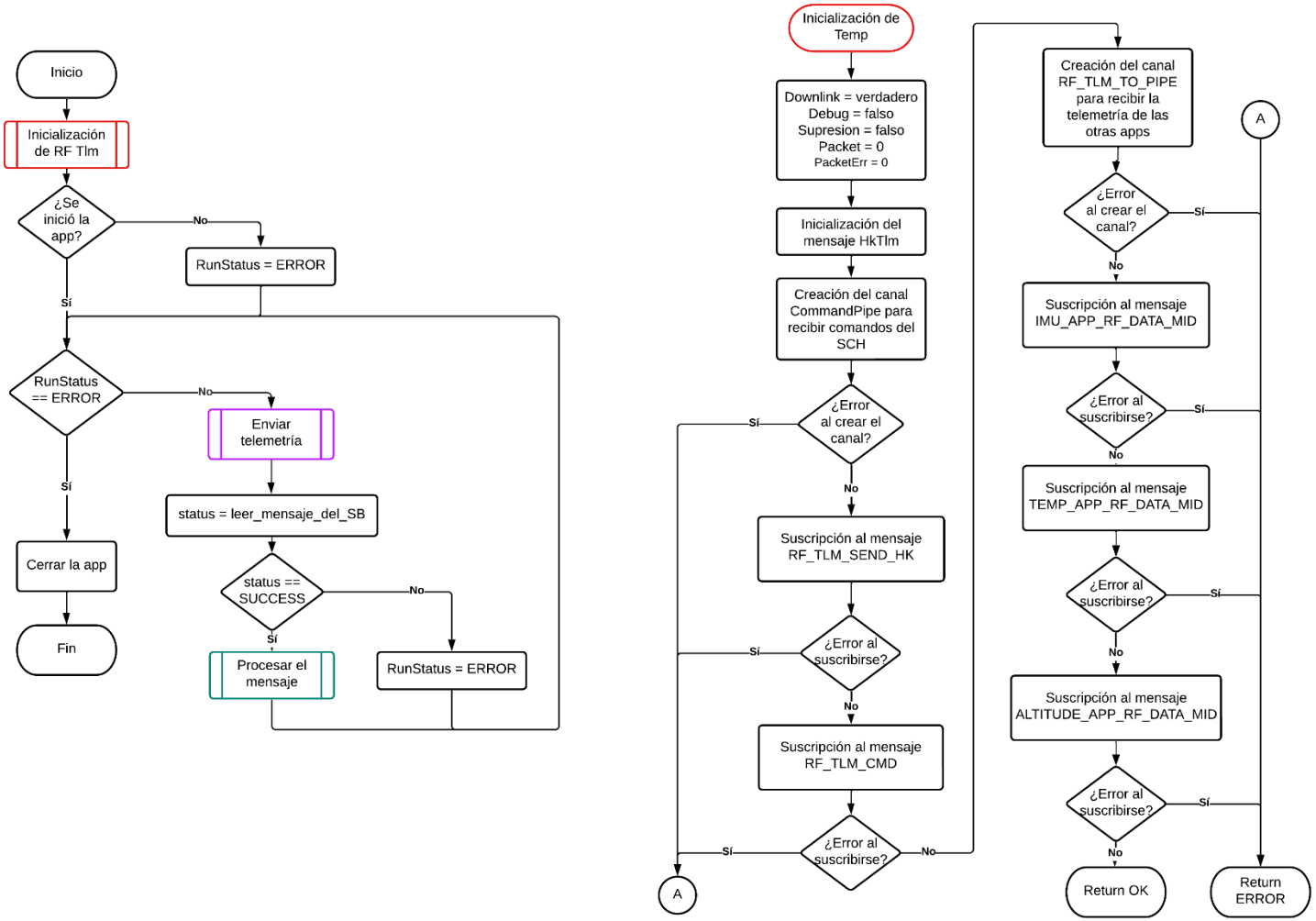
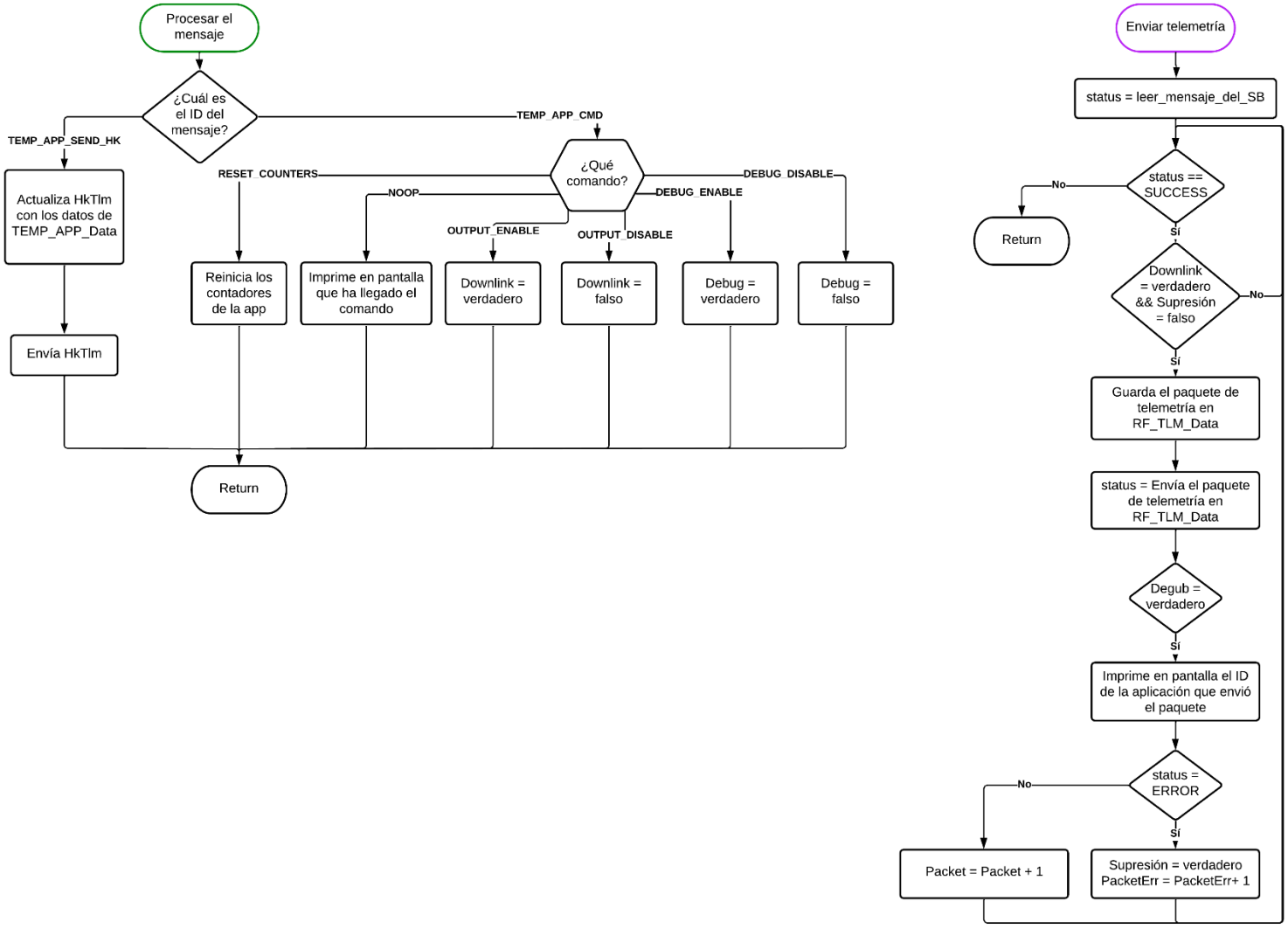


Figura 22
Diagrama de flujo de la aplicación rf_tlm, parte 2



La implementación de la aplicación se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 21 y la Figura 22. La aplicación consta de dos partes principales, la primera que es la etapa de configuración e inicialización de la aplicación y la segunda que constituye el lazo principal. La primera parte se encuentra descrita a continuación:

1. Se inicializan las variables globales de la aplicación, la estructura de datos RF_TLM_Data, se establece la variable Debug en falso, Downlink en verdadero, Supresión en falso y los contadores Packet y PacketErr en 0.
2. Se inicializa el mensaje (HkTlm, ID = 0x08F1) mediante el cual se va a enviar a la aplicación de telemetría por UDP/IP los contadores Packet y PacketErr.
3. Se crea un canal (RF_TLM_CMD_PIPE) para recibir comandos enviados por el SCH a través del SB.
4. La aplicación se suscribe a través del SB a los mensajes enviados por el SCH de la Tabla 16.

Tabla 16
Mensajes enviados por el SCH a los que se suscribe la app de telemetría

Mensaje	ID	Función
RF_TLM_CMD_MID	0x18F0	Mensaje a través del cual llegan los comandos de la estación terrestre
RF_TLM_SEND_HK_MID	0x18F1	Enviar telemetría a la app de telemetría por UDP/IP

5. Se crea un canal (RF_TLM_TO_PIPE) para recibir los paquetes de telemetría enviados a través del SB por la aplicación de monitoreo de inercia, de monitoreo de altitud y monitoreo de temperatura.
6. La aplicación se suscribe a los mensajes de las aplicaciones de monitoreo de inercia temperatura y de altitud como se observa en la Tabla 17.

Tabla 17
Mensajes enviados por otras aplicaciones a los que se suscribe la app de telemetría

Mensaje	ID	Función
ALTITUDE_APP_RF_DATA_MID	0x08A2	Mensaje a través del cual llega la telemetría de la aplicación de monitoreo de altura
IMU_APP_RF_DATA_MID	0x08E2	Mensaje a través del cual llega la telemetría de la aplicación de monitoreo de inercia
TEMP_APP_RF_DATA_MID	0x08D2	Mensaje a través del cual llega la telemetría de la aplicación de monitoreo de temperatura

7. Si hubo errores al suscribirse a los mensajes o al crear los canales del SB se escribirá en consola que ha habido un error y se cerrará la aplicación, caso contrario la aplicación entrará al lazo principal.

En el lazo principal se realizan las siguientes acciones:

1. La aplicación lee el canal RF_TLM_TO_PIPE mediante el cual recibe los mensajes de telemetría enviados por las otras aplicaciones.
2. Una vez que se ha leído el canal existen tres posibles situaciones:
   - Si existe un mensaje, Downlink es verdadero y Supresión es falso:
     i. Almacena el paquete de telemetría en RF_TLM_Data.
     ii. Envía el paquete de telemetría almacenado en RF_TLM_Data al microcontrolador.
     iii. Si Debug es verdadero identifica la aplicación que envió el mensaje e imprime en pantalla el nombre de la aplicación. Caso contrario pasa al siguiente punto.
     iv. Si existió un error al enviar el paquete al microcontrolador desactiva la telemetría al configurar Supresión como verdadero e incrementa el

contador PacketErr en uno. Caso contrario, si no existieron errores el contador Packet incrementa en uno.

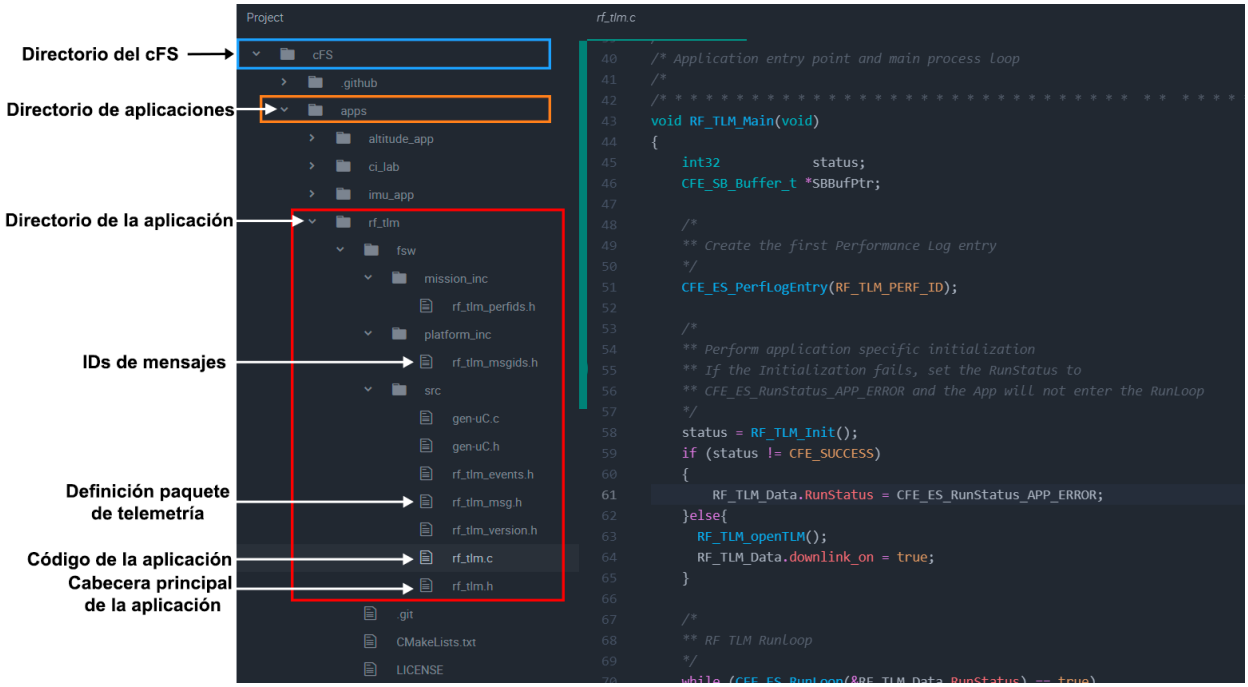
v. Regresa al punto 1.

- Si existe un mensaje y, Downlink es falso o Supresión es verdadero:
  i. Regresa al punto 1.
- Si no existe un mensaje porque no ha llegado o ya se han leído todos, pasa al punto 3.

3. La aplicación espera indefinidamente hasta que llegue un mensaje enviado por el SCH. Dependiendo del mensaje recibido:
   - Enviar Hk (mensaje RF_TLM_SEND_HK_MID):
     i. Actualiza el mensaje HkTlm con los contadores Packet y PacketErr.
     ii. Construye y envía el mensaje HkTlm a través del SB.
     iii. Regresa al punto 1.
   - Comando (mensaje RF_TLM_CMD_MID), identifica si es uno de los siguientes comandos, realizando la acción correspondiente y retornando al punto 1:
     i. Comando NoOp. No ejecuta una operación, solo escribe en consola que ha recibido el mensaje.
     ii. Comando Reiniciar contadores. Reinicia los contadores de la aplicación.
     iii. Comando habilitar telemetría. Configura Downlink como verdadero.
     iv. Comando deshabilitar telemetría. Configura Downlink como falso.
     v. Comando activar debug. Configura Debug como verdadero.
     vi. Comando desactivar debug. Configura Debug como falso.

La implementación de la aplicación para el monitoreo de inercia se puede observar en la Figura 23.

Figura 23
Implementación de la aplicación para el envío de telemetría



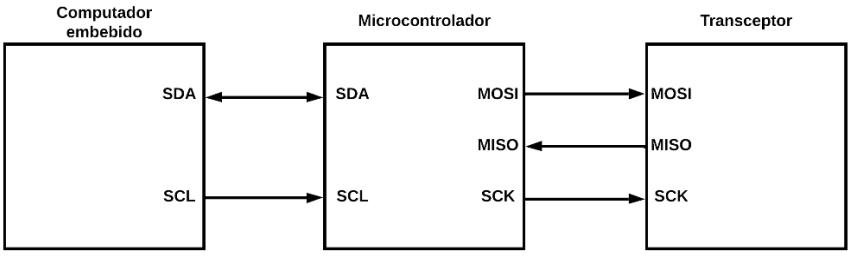
Conexiones del computador embebido con los diferentes módulos hardware

Considerando que la mayoría de los dispositivos sensores utilizan protocolo I^2C, para establecer la comunicación del computador embebido con los dispositivos sensores, se desarrollaron una serie de funciones basadas en un driver para I^2C del sistema operativo RTEMS. De esta manera se logró comunicar exitosamente al computador embebido con los diferentes sensores conectados al bus I^2C.

De manera similar se planificó una conexión directa entre el computador embebido con el transceptor mediante protocolo SPI. Así se desarrollaron una serie de funciones en un driver SPI del RTEMS, que sin embargo no tuvieron éxito y había que trabajar en más profundos cambios del sistema operativo. Por lo tanto, con la finalidad de establecer la conexión entre el computador embebido y el transceptor que utiliza protocolo SPI se requirió un conversor I^2C /SPI. Para ello se seleccionó al microcontrolador ATmega328P, que cumpliría con la función de interfaz o medio de comunicación entre el computador embebido y el transceptor. De esta

manera, la conexión entre el computador embebido y el microcontrolador se realizaría a través del protocolo I^2C y la conexión entre el microcontrolador y el transceptor se realizaría mediante el protocolo SPI, como se observa en el diagrama de bloques de la Figura 24.

Figura 24
Conexión entre el computador embebido, microcontrolador y transceptor



El hardware implementado se puede observar en la Figura 25. En el recuadro azul se encuentra el sensor de inercia MPU6050, en el recuadro morado el sensor de altitud MPL3115A2, en el recuadro verde el sensor de temperatura AHT10 y por último, en el recuadro rojo el microcontrolador ATmega328P con el transceptor SX1278.

Figura 25
Implementación del hardware del sistema

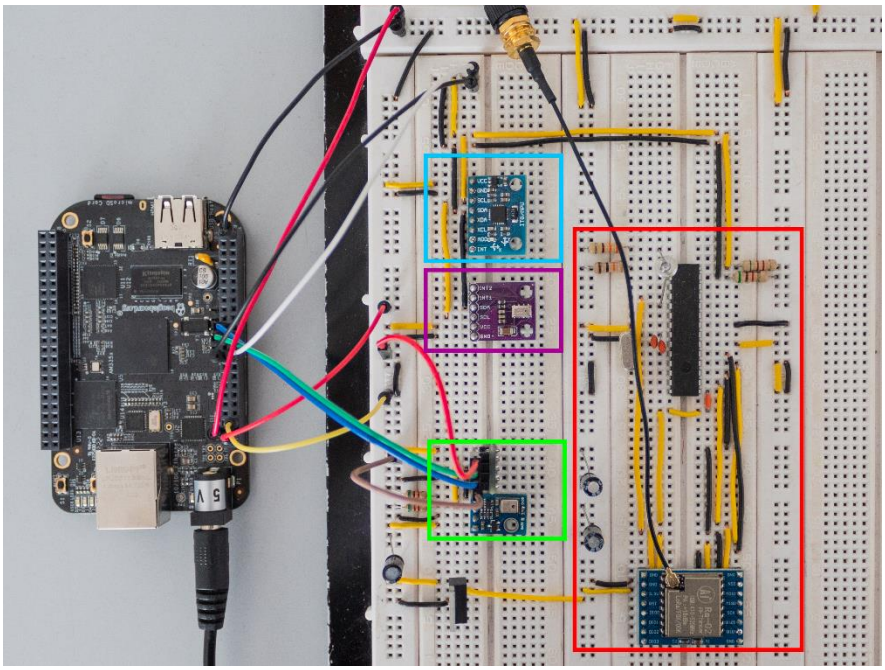
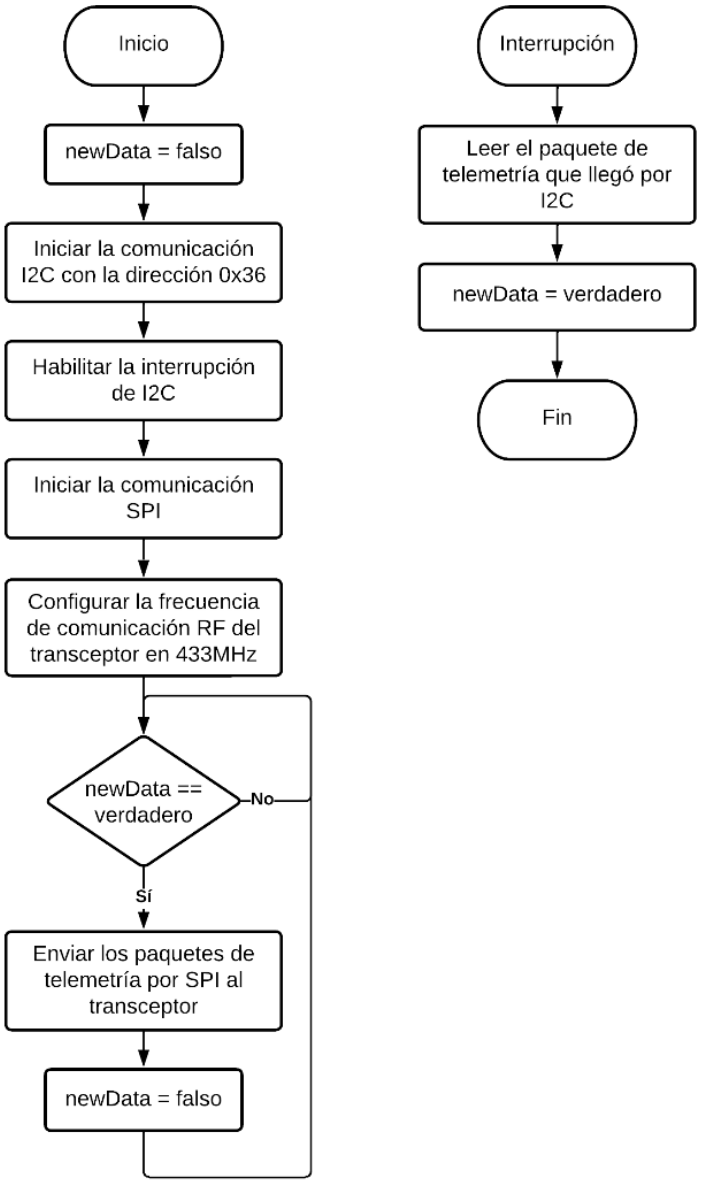


Figura 26
Diagrama de flujo del programa de conversión de protocolos



La implementación del programa de conversión de protocolos que realiza el microcontrolador se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 26. De manera general, el funcionamiento del programa se describe a continuación:

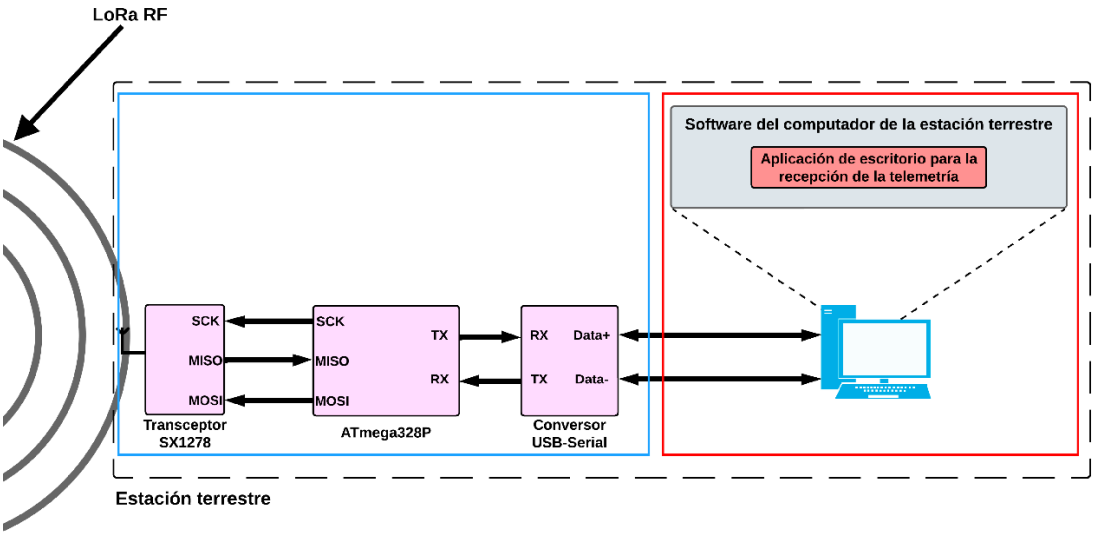
1. Se configuran las variables globales del programa y se establece newData como falso.

2. Se inicia la comunicación I^2C y en seguida se configura al microcontrolador como un dispositivo esclavo en el bus I^2C con la dirección 0x36.
3. Se habilita la interrupción correspondiente al protocolo I^2C
4. Se inicia la comunicación SPI y acto seguido inicializa el transceptor y se configura la frecuencia de comunicación RF en 433MHz.
5. Comprueba si newData es verdadero, de ser así:
    - Envía los paquetes de telemetría recibidos al transceptor a través del puerto SPI.
    - Configura newData como falso.
    - Regresa al punto 5.
6. En caso de generarse una interrupción por la llegada de datos a través del protocolo I^2C, el microcontrolador lee el paquete enviado por el computador embebido y configura newData como verdadero.

Sistema terrestre

Una vez implementado el RTOS, el sistema de control de misión y las aplicaciones de usuario se procedió a desarrollar el sistema terrestre a través del cual se recibe y presenta la telemetría enviada mediante comunicación RF. Como se observa en la Figura 27 de manera general, el sistema terrestre está dividido en dos partes, la primera, en el recuadro azul, es el sistema de comunicaciones mediante el que se recibe los paquetes de telemetría y la segunda parte, en el recuadro rojo, está conformada por un computador en el que se ejecuta la aplicación de escritorio mediante la cual se muestra la información recibida al operador.

Figura 27
Estructura general de la estación terrestre



Sistema de comunicaciones

Para la recepción de los paquetes enviados por RF se decidió utilizar un transceptor de las mismas características que utiliza el computador de a bordo. Dado que el transceptor utiliza el protocolo SPI se decidió, al igual que en el computador de a bordo, utilizar un microcontrolador que sirva como interfaz o medio de comunicación entre el computador del segmento terrestre y el transceptor. En este caso, el microcontrolador utiliza el puerto serie USART conectado a un conversor USB-serial para comunicarse con el computador del segmento terrestre. Por otra parte, tomando en consideración que el transceptor del segmento terrestre solo se utilizará para recibir los paquetes de telemetría, se configurará al pin DIO0 del transceptor para que este genere una interrupción externa por cambio de nivel cuando se haya recibido un paquete de telemetría. De esta manera se le indicará al microcontrolador que puede leer el paquete del transceptor, por lo que la conexión entre los diferentes dispositivos, de manera general, es como se ilustra en la Figura 28. En la Figura 29 se puede observar el sistema de comunicaciones ya implementado.

Figura 28

Conexión entre el transceptor, microcontrolador y computador del segmento terrestre

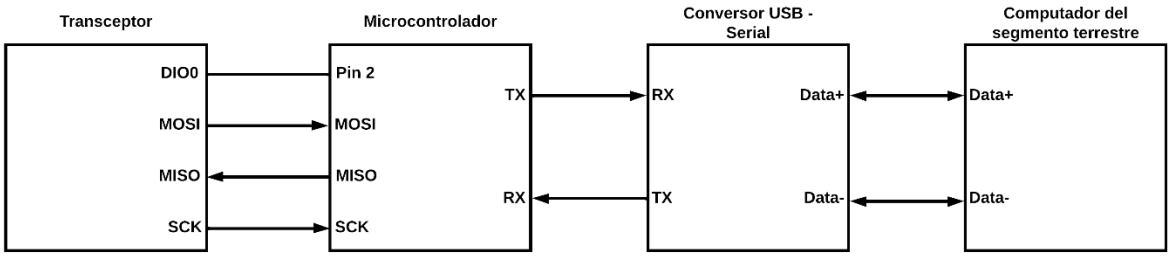
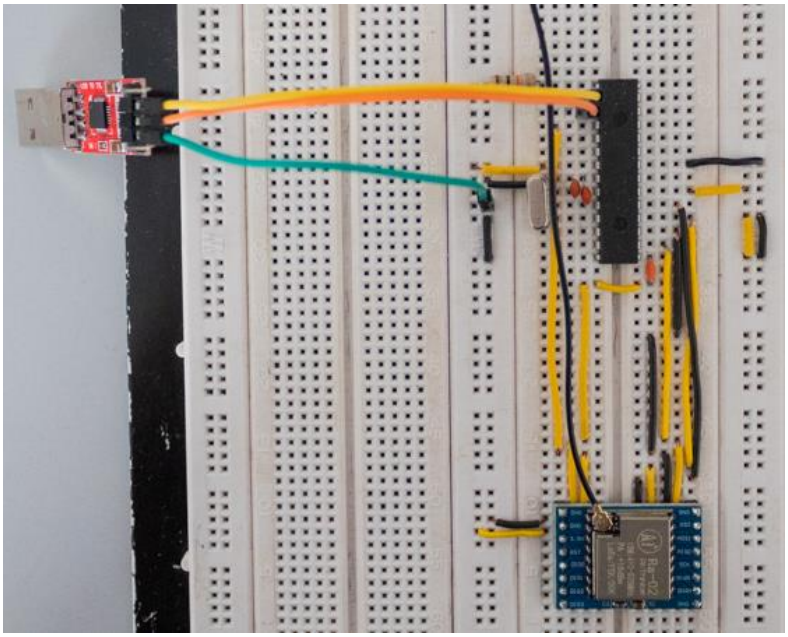


Figura 29

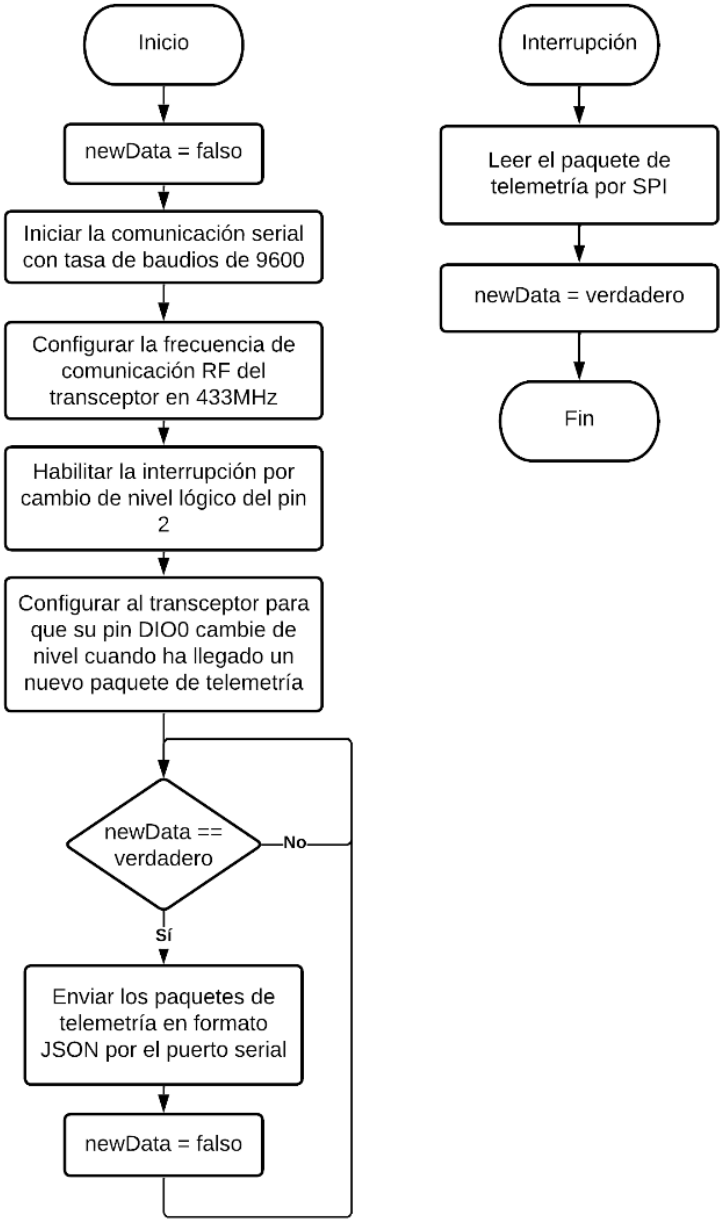
Implementación del hardware del sistema de comunicaciones de la estación terrestre



La implementación del programa del microcontrolador que sirve de interfaz entre el transceptor y el computador de la estación terrestre se realizó de acuerdo al diagrama de flujo que se ilustra en la Figura 30.

Figura 30

Diagrama de flujo del programa del microcontrolador de la estación terrestre



De manera general, el funcionamiento del programa se describe a continuación:

1. Se configuran las variables globales del programa y se establece newData como falso.

2. Se inicia la comunicación por el puerto serial y se configura la tasa de baudios a 9600.
3. Se inicia la comunicación SPI y acto seguido inicializa el transceptor y se configura la frecuencia de comunicación RF en 433MHz.
4. Se habilita la interrupción por cambio de nivel lógico del pin 2 del microcontrolador
5. Se configura al transceptor para que su pin DIO0 cambie de nivel cuando ha llegado un nuevo paquete de telemetría.
6. Comprueba si newData es verdadero, de ser así:
    - Envía el paquete de telemetría en formato JSON al computador del segmento terrestre.
    - Configura newData como falso.
    - Regresa al punto 6.
7. En caso de generarse una interrupción el cambio de nivel del pin DIO0, el microcontrolador lee el paquete recibido por el transceptor y configura newData como verdadero.

Aplicación de escritorio

Para el diseño de la aplicación de escritorio que se ejecuta en computador del segmento terrestre y que realiza la lectura de paquetes de telemetría enviados por el computador de a bordo se tomaron en cuenta las siguientes consideraciones:

- La información presentada al usuario debe tener un formato con el que se encuentre familiarizado, en este caso se ha considerado que el formato de la información debe ser similar al que se usa en la aplicación de escritorio de la distribución del cFS.
- La manera en la que la interfaz funciona debe ser intuitiva para el usuario.
- La interfaz no debe contener una cantidad abrumadora de elementos visuales.

- La aplicación de escritorio debe permitir visualizar la telemetría enviada por las diferentes aplicaciones de usuario.

De esta manera, la aplicación de escritorio es de fácil uso, con un diseño eficaz y versátil, ya que permite visualizar los diferentes paquetes de telemetría enviados por las aplicaciones de usuario del computador de a bordo. Se diseñó a la aplicación de tal forma que esta cuente con tres ventanas:

1. Ventana principal, la cual se usa para seleccionar el puerto al que está conectado el microcontrolador y para seleccionar la tasa de baudios, como se observa en la Figura 31.
2. Ventana de telemetría general, en la cual se encuentran a manera de lista las distintas aplicaciones de usuario del OBC que envían telemetría. En este caso, la aplicación para monitoreo de altitud, la aplicación para monitoreo de inercia y la aplicación para monitoreo de temperatura. En esta ventana, solo se muestran las distintas aplicaciones y a cada una le corresponde un botón que permite abrir la ventana de telemetría particular, como se visualiza en la Figura 32.
3. Ventana de telemetría particular, la cual muestra la telemetría de una sola aplicación, en esta se podrá observar los distintos datos enviados por cada aplicación a manera de tablas, como se ilustra en la Figura 33.

Figura 31
Ventana principal de la aplicación de escritorio de la estación terrestre

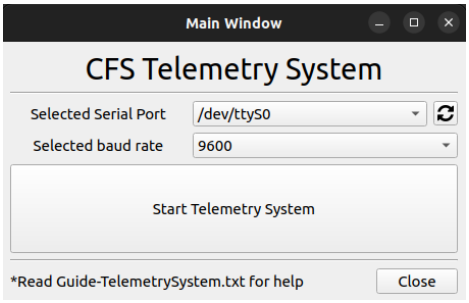


Figura 32

Ventana de telemetría general de la aplicación de escritorio de la estación terrestre

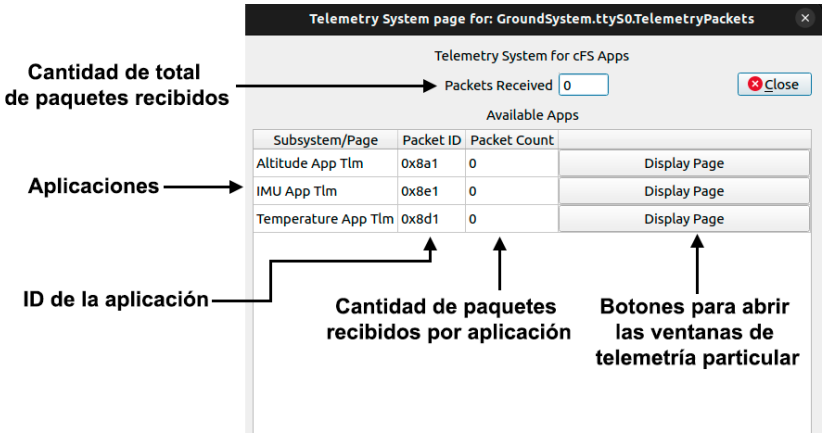
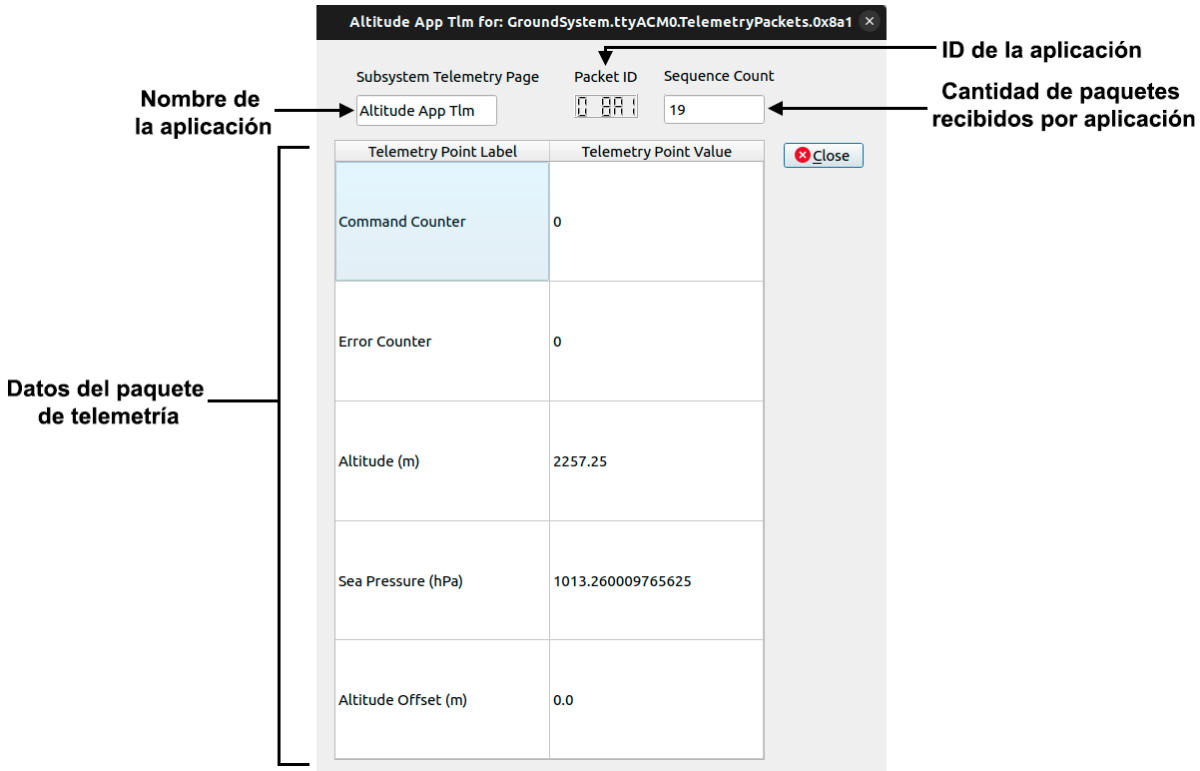


Figura 33

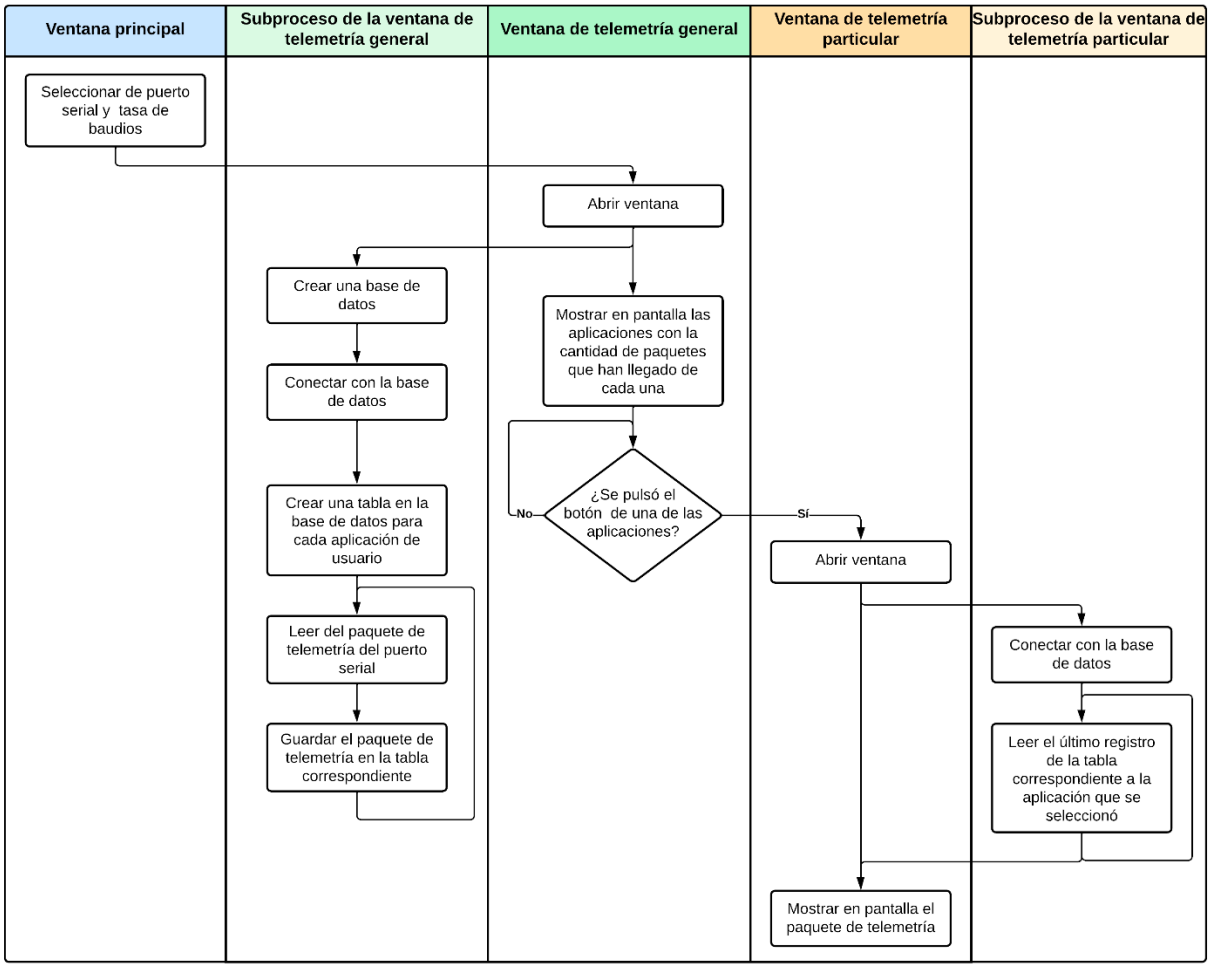
Ventana de telemetría específica de la aplicación de escritorio de la estación terrestre



Para el desarrollo de la aplicación de escritorio utilizó el lenguaje de programación Python y para la elaboración de la interfaz gráfica la biblioteca PyQt, una biblioteca gráfica del

lenguaje de programación Python. La implementación de la aplicación de escritorio se realizó de acuerdo al diagrama de flujo que se observa en la Figura 34.

Figura 34
Diagrama de flujo de la aplicación de escritorio de la estación terrestre



De manera general el funcionamiento de la aplicación de escritorio es el siguiente:

1. A través de la ventana principal se selecciona el puerto al que está conectado el microcontrolador y la tasa de baudios. Una vez seleccionados estos valores, se procede a abrir la ventana de telemetría general mediante un botón.
2. Al iniciar la ventana de telemetría general, se inicia un subproceso el cual:
   i. Crea una base de datos

ii. Inicia la conexión con una base de datos y crea una tabla por cada aplicación de usuario. En este caso, son tres tablas.

iii. Lee el paquete de telemetría enviado por el microcontrolador.

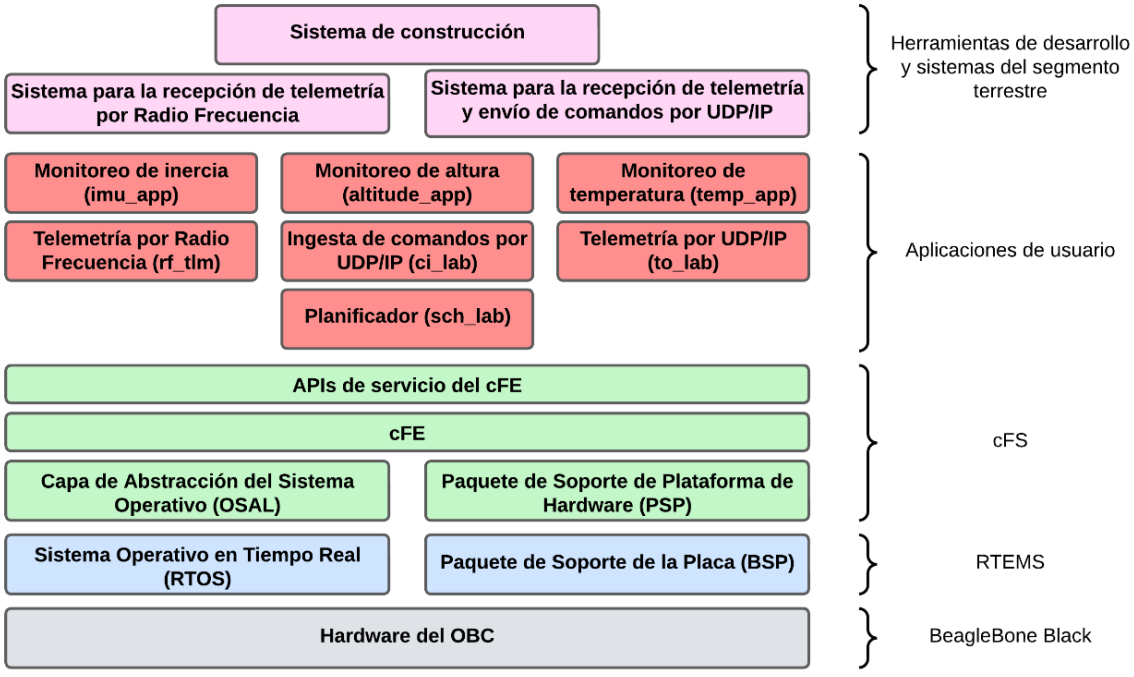
iv. A partir del App ID del paquete, determina que aplicación envió el paquete para así almacenar los datos la tabla correspondiente.

v. Regresa al punto iii.

3. Al abrir una ventana de telemetría particular, se inicia un subproceso el cual establece una conexión con la base de datos y constantemente lee el último registro de la tabla correspondiente a la aplicación de la cual se abrió la ventana. Posterior a ello se presentan los datos del registro en pantalla.

Figura 35

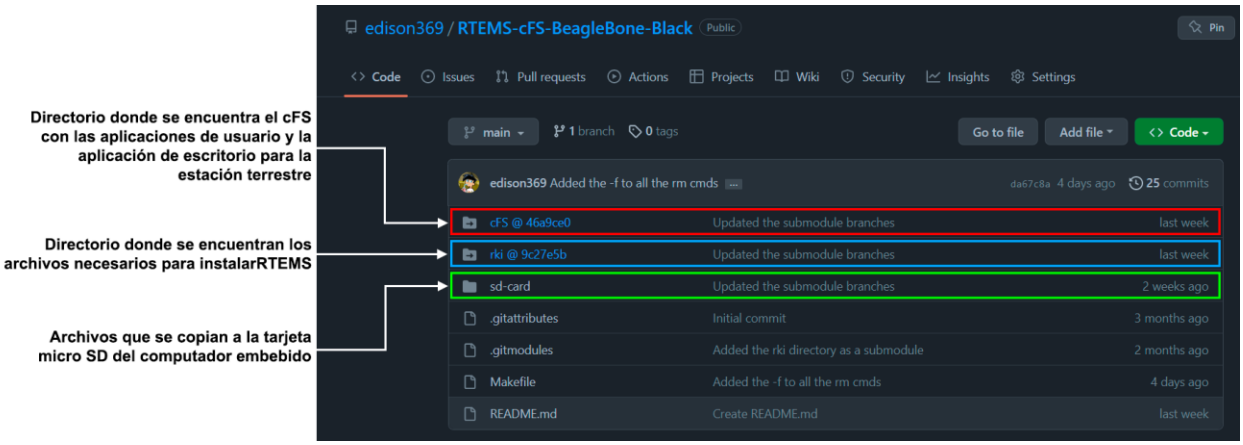
Arquitectura de Software del sistema del proyecto de titulación



Una vez implementado el sistema propuesto del computador de a bordo y de los componentes de hardware necesarios para las aplicaciones de monitoreo de la inercia, la

altitud y temperatura del nanosatélite y telemetría, se obtuvo la arquitectura de software del sistema como se ilustra en la Figura 35. Cabe indicar, que todos los recursos de software de este proyecto de titulación se encuentran disponibles en el GitHub del autor, cuya dirección web es https://github.com/edison369/RTEMS-cFS-BeagleBone-Black. Como se observa en la Figura 36, en el repositorio se encuentra el directorio del cFS, con las aplicaciones de usuario y la aplicación de escritorio para el sistema terrestre, además se encuentra un directorio en el que se encuentran todos los archivos necesarios para instalar RTEMS en el computador embebido.

Figura 36
Repositorio del proyecto de titulación



En caso de requerir probar el sistema, el repositorio debe ser clonado y los submódulos fueron actualizados mediante de la siguiente forma:

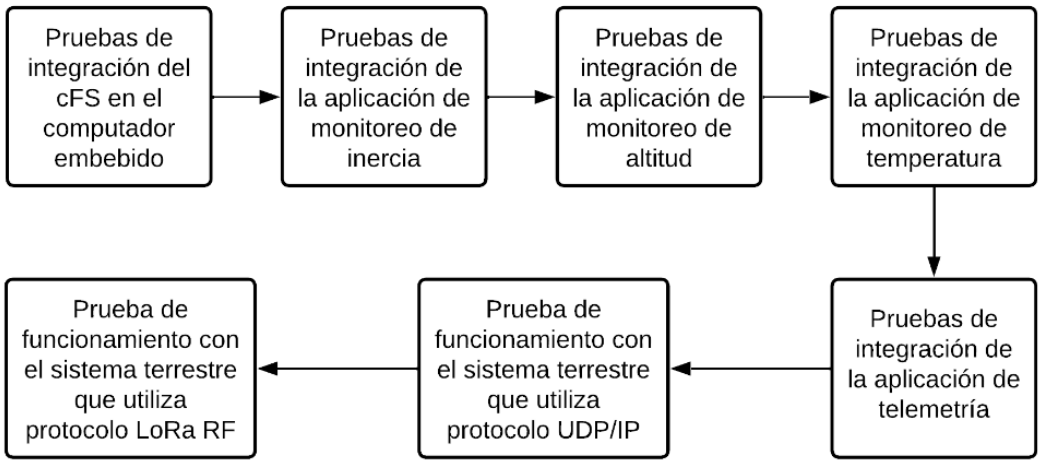
- $ git clone https://github.com/edison369/RTEMS-cFS-BeagleBone-Black.git
- $ cd RTEMS-cFS-BeagleBone-Black
- $ git submodule update --init –recursive

La validación del sistema implementado mediante las pruebas planteadas se detalla en el Capítulo 5.

Capítulo 5: Validación

El presente capítulo resume las diferentes pruebas funcionales de la implementación del sistema. La implementación de la propuesta se realizó por capas tal como se indicó en el capítulo anterior, en la Figura 35. Para ello se inició con la instalación del sistema operativo sobre el computador embebido, luego se configuró la capa del cFS, en un tercer lugar se diseñaron e implementaron las aplicaciones hasta integrar las funcionalidades básicas para emular el computador de a bordo del nanosatélite incluyendo los sensores. Finalmente se probó el sistema mediante la conexión con una estación terrena. Esta conexión fue realizada inicialmente mediante protocolo ethernet y posteriormente mediante protocolo LoRa RF. Esta metodología de desarrollo y pruebas se resume en la Figura 37. Adicionalmente a las pruebas funcionales del sistema, se realizaron pruebas complementarias de medición del desempeño del sistema en cuanto a consumo de potencia y al alcance de las comunicaciones con el sistema terrestre.

Figura 37
Metodología de desarrollo de pruebas funcionales



Integración del sistema de control de misión en el computador embebido

Ejecución del cFS en el computador embebido

Procedimiento. Esta prueba se realiza para verificar que el sistema de control de misión cFS fue implementado correctamente sobre el computador embebido. Para ello, primero se accede al directorio del cFS en la plataforma de desarrollo y se construye el ejecutable del sistema de control de misión:

- $ cd cFS
- $ make SIMULATION=arm-bbb-rtems6 prep
- $ make
- $ make install

Una vez construido el ejecutable del cFS, se procede a construir la imagen del kernel RTEMS usando el ejecutable del sistema de control de misión:

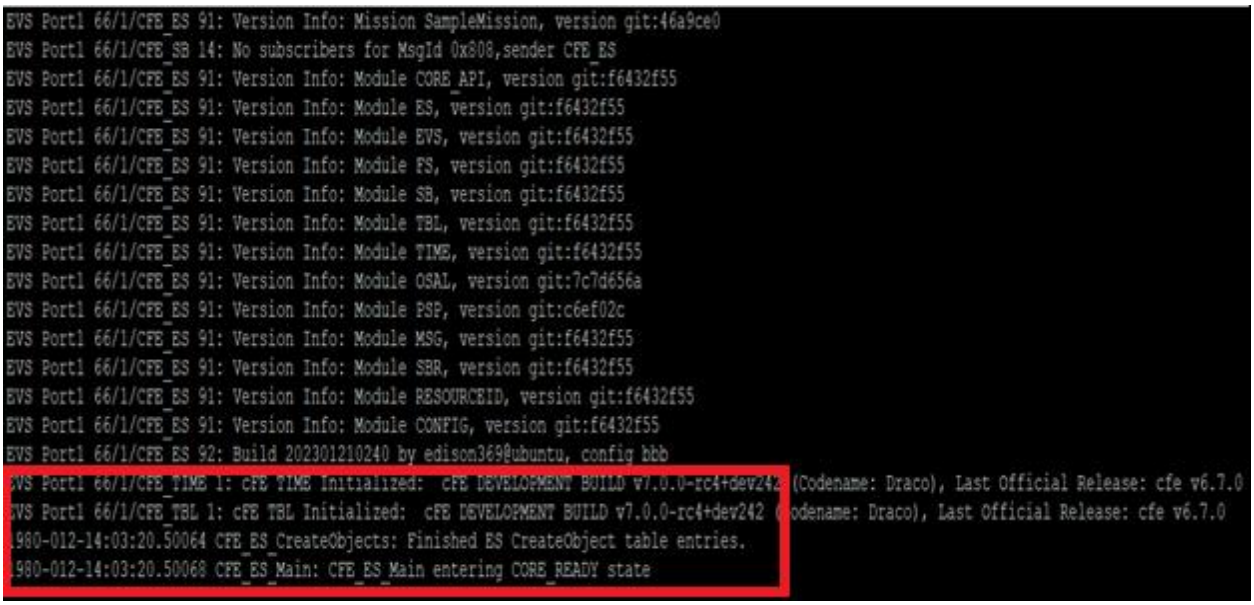
- $ cd ../rki/build/bbb-libbsd-cfs
- $ make

Al usar el comando make, se crea el archivo “rtems-rki.img” en el que se encuentra la imagen del kernel de RTEMS, el ejecutable del cFS compilado y los diferentes objetos correspondientes a las aplicaciones del cFS empaquetados en un sistema de archivos propio de RTEMS. Este archivo se debe copiar a la tarjeta microSD del computador embebido y una vez que se encienda el computador embebido, dos segundos después de que este haya iniciado, se ejecutará el cFS automáticamente.

Resultado esperado. En caso de haber seguido el procedimiento de manera correcta y que no se den fallas de hardware, el cFS debería ejecutarse sin problemas. Una vez que inicie el cFS se debe visualizar en el terminal del computador embebido que este se está ejecutando.

Resultados obtenidos. Como se esperaba, la prueba fue exitosa y el cFS se ejecutó de manera correcta. Como se puede observar en la Figura 38, en las últimas líneas impresas en pantalla se indica que el cFE y algunos de los servicios que este provee, están siendo inicializados.

Figura 38
Resultados de la prueba de ejecución del sistema de control de misión en el computador embebido



Integración de las aplicaciones de usuario al sistema de control de misión

Una vez que se comprobó que el cFS se ejecuta correctamente en el computador embebido, se procedió a validar que las aplicaciones de usuario desarrolladas se integren correctamente al sistema de control de misión. Para tal efecto, se comprobó con cada aplicación que:

- Sea cargada por el cFE
- Se inicialice

Acto seguido, en el caso de las aplicaciones de monitoreo, se validó que la aplicación realice la lectura del sensor correspondiente y en el caso de la aplicación de telemetría se

comprobó que esta reciba los datos de las otras aplicaciones y envíe los paquetes al microcontrolador que sirve de interfaz entre el OBC y el transceptor.

Aplicación de monitoreo de inercia

Procedimiento. Mediante esta prueba se busca verificar el correcto funcionamiento de la aplicación de monitoreo de inercia "*IMU_App*". Para ello se validará que la aplicación se cargue, se inicialice, inicialice el sensor MPU6050 y lea los datos correspondientes a la aceleración y velocidad angular cada que el SCH lo solicite.

A fin de realizar esta prueba, se debe agregar la aplicación IMU_App al cFS. Por lo tanto, se modificó la lista de aplicaciones en el archivo "targets.cmake" de la siguiente forma:

```
SET(cpu1_APPLIST ci_lab to_lab sch_lab imu_app)
```

De la misma manera, se modificó el archivo "cpu1_cfe_es_startup.scr". En este se incluyen las aplicaciones para que estas sean cargadas una vez que inicia el cFS y se deben incluir algunos parámetros de la aplicación como su directorio, función principal, prioridad, entre otros, en este caso se agregó la línea:

```
CFE_APP, imu_app,    IMU_APP_Main,      IMU_APP,     90,   16384, 0x0, 0;
```

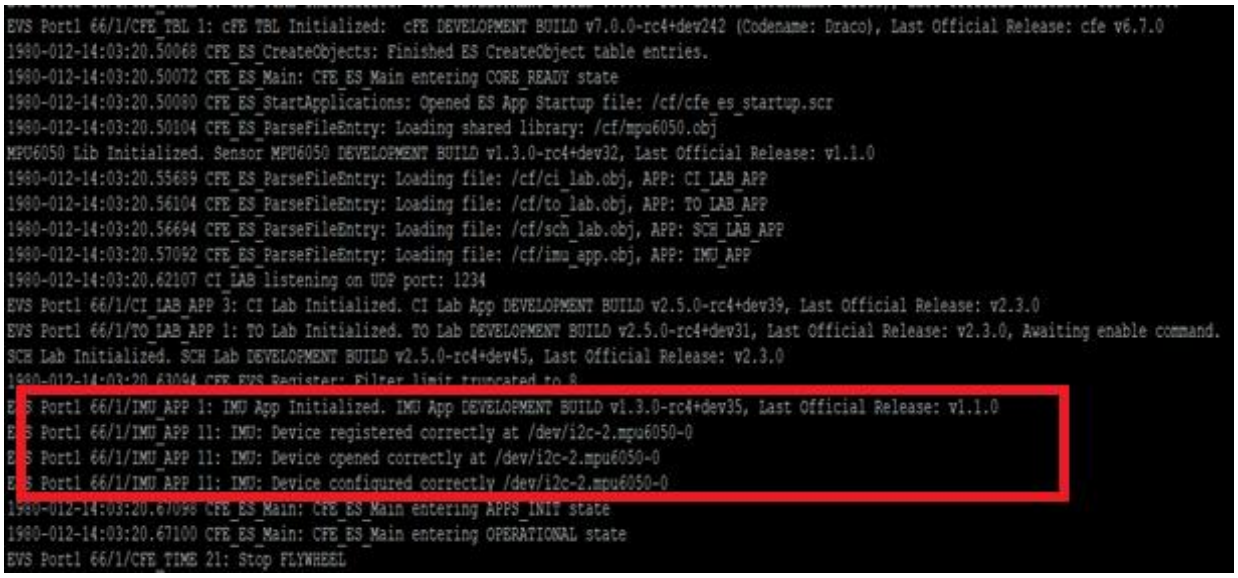
Por otro lado, la conexión entre el sensor MPU6050 y el OBC se debe realizar correctamente. De esta manera, será posible ejecutar el cFS y validar el funcionamiento.

Resultados esperados. Si el software se ha desarrollado correctamente, los dispositivos de hardware funcionan sin fallas y las conexiones se realizan de la manera correcta, se espera que el cFS se ejecute y cargue la aplicación imu_app. Si la aplicación inicia correctamente, el sensor debe ser inicializado y posterior a ello se deben imprimir en pantalla las lecturas de la aceleración y velocidad angular registradas por el sensor.

Resultados obtenidos. Como se puede observar en la Figura 39 una vez que el cFE inicia, las aplicaciones se cargan e inician. En el caso de la aplicación de monitoreo de inercia, una vez que inicia se realiza la configuración del sensor. Una vez inicializado el sistema, se procedió a realizar las pruebas con el sensor de inercia MPU6050. Para ello se realizaron dos tipos de pruebas, la primera con el sensor inmóvil y la segunda con el sensor en movimiento.

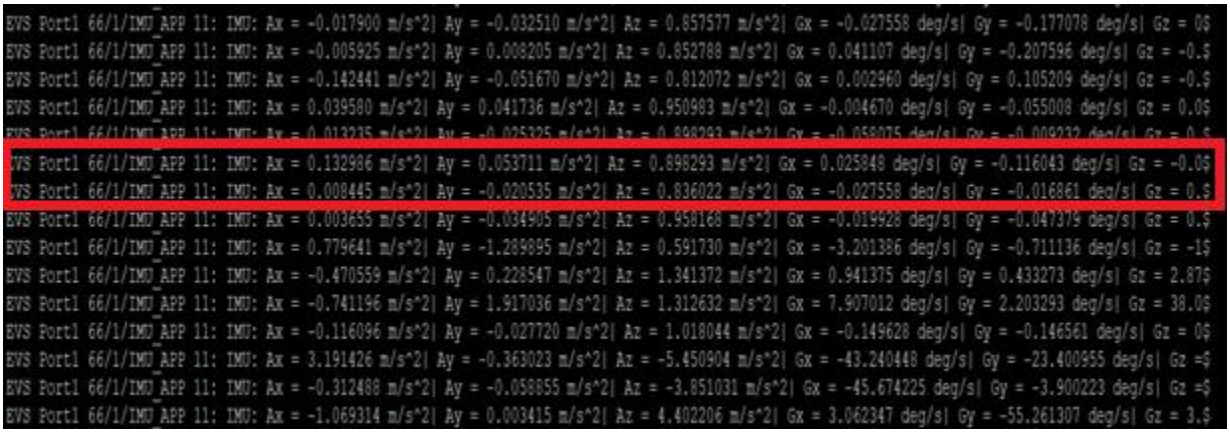
Figura 39

Ejecución de la aplicación para monitoreo de inercia e inicialización del sensor MPU6050



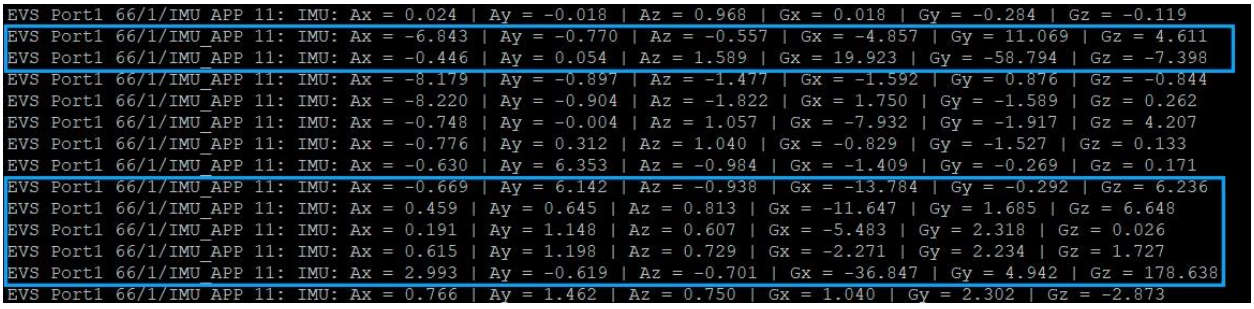
En la Figura 40 se puede observar las medidas de aceleración y velocidad angular monitoreadas cada 4 segundos según la configuración del Scheduler cuando el sensor estaba inmóvil. Como se puede observar dada la condición de inmovilidad, las medidas obtenidas por la aplicación son cercanas a cero.

Figura 40
Lectura de aceleración y velocidad angular en condición de inmovilidad



En tanto que la Figura 41 muestra el resultado obtenido cuando se movía el sensor En virtud de los resultados mostrados tanto en condición de inmovilidad como de movimiento, se puede concluir que las pruebas fueron exitosas.

Figura 41
Lectura de aceleración y velocidad angular en condición de movimiento



Aplicación de monitoreo de altitud

Procedimiento. A través de esta prueba se busca validar el funcionamiento de la aplicación de monitoreo de altitud “ALTITUDE_APP”. Para ello, al igual que en la prueba anterior, se validará que la aplicación se cargue, se inicialice, inicialice el sensor MPL3115A2 y lea los datos correspondientes a la altitud cada que el SCH lo solicite. En este caso, para probar la validez de los datos se utilizará como patrón de medida, las lecturas registradas por

los sensores incluido en el GPS de diferentes teléfonos móviles con la finalidad de realizar la calibración del sensor utilizado por el sistema embebido.

Para realizar la prueba, se debe agregar la aplicación altitude_app al cFS por lo que se modificó la lista de aplicaciones en el archivo “targets.cmake” de la siguiente forma:

SET(cpu1_APPLIST ci_lab to_lab sch_lab altitude_app)

Además, se modificó el archivo “cpu1_cfe_es_startup.scr”, en el que se agregó la siguiente línea:

CFE_APP, altitude_app,ALTITUDE_APP_Main, ALTITUDE_APP, 90,   16384, 0x0, 0;

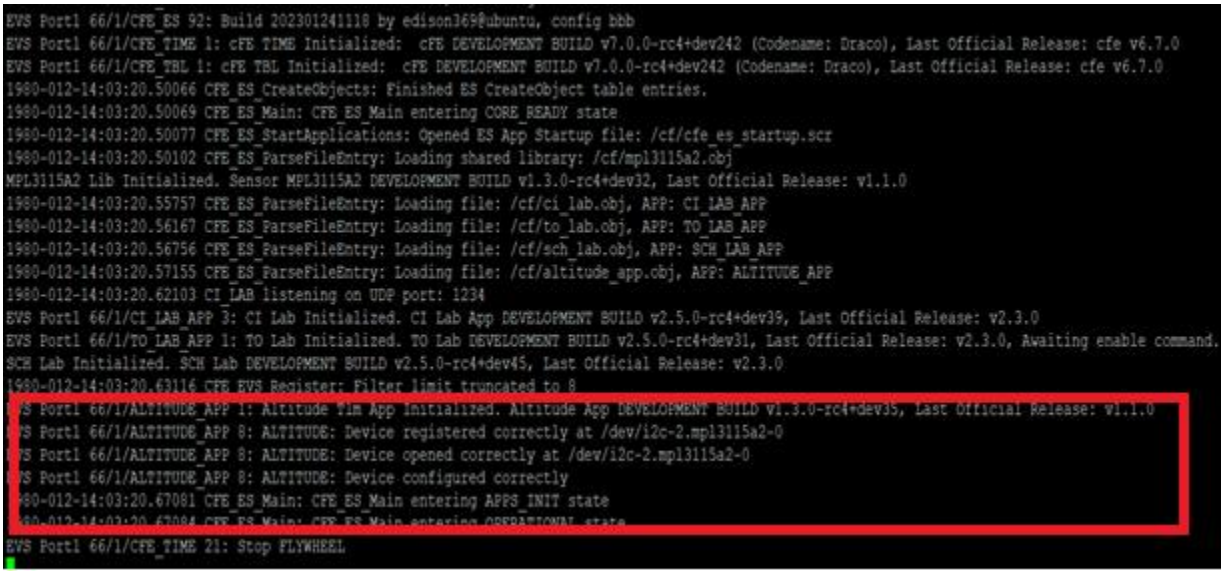
Es importante notar que la conexión de hardware entre el sensor MPL3115A2 y el computador embebido se debe realizar correctamente. Una vez hecho esto se procede a ejecutar el cFS y validar el funcionamiento.

Resultados esperados. Si el software se ha desarrollado correctamente, los dispositivos de hardware funcionan sin fallas y las conexiones se realizan de la manera correcta, se espera que el cFS se ejecute y cargue la aplicación ALTITUDE_APP. Si la aplicación inicia correctamente, el sensor debe ser inicializado y posterior a ello se deben imprimir en pantalla las lecturas de la altitud registradas por el sensor.

Resultados obtenidos. Como se muestra en la Figura 42 una vez que el cFE inicia, la aplicación de monitoreo de altitud se carga, posterior a ello inicia y realiza la configuración del sensor.

Figura 42

Ejecución de la aplicación para monitoreo de altitud e inicialización del sensor MPL3115A2



En lo que respecta a las mediciones, se realizaron mediciones en diferentes lugares. Las primeras pruebas se obtuvieron en Tumbaco. Como se puede observar en la Figura 43, aplicación ALTITUDE_APP obtuvo los datos a partir del sensor MPL3115A2, siendo la altitud promedio de 2243.78 m. Por otra parte, el GPS del teléfono móvil modelo Motorola Moto G Power utilizando la aplicación Device Help en su funcionalidad “Pruebas de Hardware” entregó un valor de altura de 2359 m como se ilustra en la captura de pantalla del teléfono mostrado en la Figura 44. Cabe señalar que, al momento de realizar las pruebas, el teléfono estaba conectado a un total de 27 satélites. Al comparar esta información con los datos obtenidos por el sensor, se evidencia una diferencia de aproximadamente 115 m. La diferencia de la medida puede deberse a varios factores, primero como indica la hoja de datos técnicos del sensor, se pueden generar desajustes en la calibración del sensor debido a estrés durante la fabricación y montaje del dispositivo, por lo que se recomienda establecer el offset de altitud del sensor. Segundo, el patrón seleccionado puede proporcionar una medida diferente dependiendo del tipo de red utilizada, ya sea red satelital o red de radio bases.

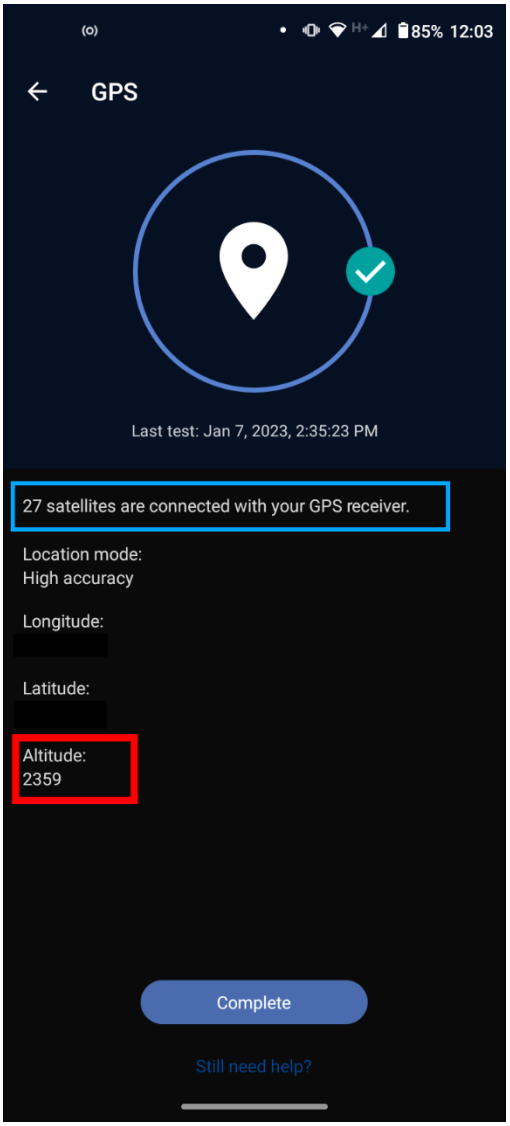
Figura 43

Lectura de altitud sin offset en Tumbaco

```
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.687500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.875000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2244.000000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.937500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2244.125000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.687500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.812500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.625000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.625000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.937500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.625000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.812500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2243.437500 m
```

Figura 44

Altitud de referencia obtenida por la aplicación móvil en Motorola Moto G Power



Para encontrar la causa se realizó un set de pruebas en el campus Sangolquí de la Universidad de las Fuerzas Armadas, obteniéndose por parte de la aplicación ALTITUDE_APP un valor de 2385 m como se puede observar en la Figura 45. En tanto que por la aplicación “ALTÍMETRO PRECISO” del teléfono Samsung A73 se obtuvo una medida de 2495 m tanto para los datos obtenidos por conexión satelital y por radio base. Las mediciones obtenidas de la aplicación del celular se muestran en la Figura 46. Es decir, también la aplicación midió 111 m menos, de manera similar a lo que sucedió en las pruebas de Tumbaco. De esta manera se puede concluir que la medida del sensor MPL3115A2 requiere un offset positivo de 113 m si se trabaja con el promedio de las medidas.

Figura 45
Lectura de altitud sin offset en Sangolquí

```
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.312500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.562500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.125000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.875000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.500000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.312500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.375000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.312500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.250000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2385.187500 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2384.875000 m
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2384.687500 m
```

Figura 46

Altitud de referencia obtenida por la aplicación "Altímetro Preciso" en Samsung A73

En estas pruebas iniciales se buscó verificar el funcionamiento de la aplicación. Pruebas que fueron exitosas a pesar de la diferencia del error que tiene la medida registrada por el sensor en comparación a la medida de los sistemas de referencia. Luego con la información obtenida se trabajó con la calibración del sensor manejando un offset en la medida. Una vez considerado el offset, se realizaron otras pruebas obteniéndose medidas con un error de ± 3 m respecto al patrón. La Figura 47 y Figura 48 ilustran los nuevos resultados.

Figura 47
Lectura de altitud con offset en Sangolquí

```
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2510.937500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2510.562500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2510.687500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2511.437500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2511.375000
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2510.562500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2510.187500
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2509.375000
EVS Port1 66/1/ALTITUDE_APP 8: ALTITUDE: Altitude = 2509.500000
```

Figura 48
Altitud de referencia obtenida por la aplicación "Altímetro Preciso" en Samsung A73

Aplicación de monitoreo de temperatura

Procedimiento. El objetivo de esta prueba es validar el funcionamiento de la aplicación de monitoreo de temperatura TEMP_APP. Por consiguiente, la aplicación debe ser cargada por el cFS y una vez que esta inicie, el sensor AHT10 debe ser inicializado. La aplicación debe registrar la temperatura y humedad medidas por el sensor y, además, debe recibir el valor de temperatura registrado por los sensores de inercia MPU6050 y de altitud MPL3115A2.

Tomando en consideración el diseño del sistema, es necesario que las aplicaciones para monitoreo de inercia y altitud también sean cargadas, ya que mediante estas aplicaciones se inicializan y registran los valores de temperatura de los sensores antes mencionados. Por otra parte, se realizará una comparación de la temperatura registrada por el sensor AHT10 con la temperatura registrada por una referencia. En este caso, se utilizó el multímetro Pro'skit MT-1860, el cual permite medir temperatura en un rango de -20 a 400 ºC con una resolución de 0.1 ºC y una precisión de ±(1.0 +50d).

A fin de realizar esta prueba, se debe agregar la aplicación TEMP_APP al cFS y para que esta reciba los valores de temperatura medida por los sensores MPU6050 y MPL3115A2 se deben agregar las aplicaciones de monitoreo de inercia y monitoreo de altitud. Por lo tanto, se modificó la lista de aplicaciones en el archivo "targets.cmake" de la siguiente forma:

```
SET(cpu1_APPLIST ci_lab to_lab sch_lab temp_app imu_app altitude_app)
```

De la misma manera, se modificó el archivo "cpu1_cfe_es_startup.scr" incluyendo las 3 aplicaciones de monitoreo de la siguiente manera:

```
CFE_APP, temp_app,    TEMP_APP_Main,      TEMP_APP,     90,   16384, 0x0, 0;

CFE_APP, altitude_app,ALTITUDE_APP_Main,  ALTITUDE_APP, 90,   16384, 0x0, 0;

CFE_APP, imu_app,     IMU_APP_Main,       IMU_APP,      90,   16384, 0x0, 0;
```

Al igual que en las pruebas anteriores es importante resaltar que la conexión entre el OBC y los diferentes sensores se debe realizar de manera correcta.

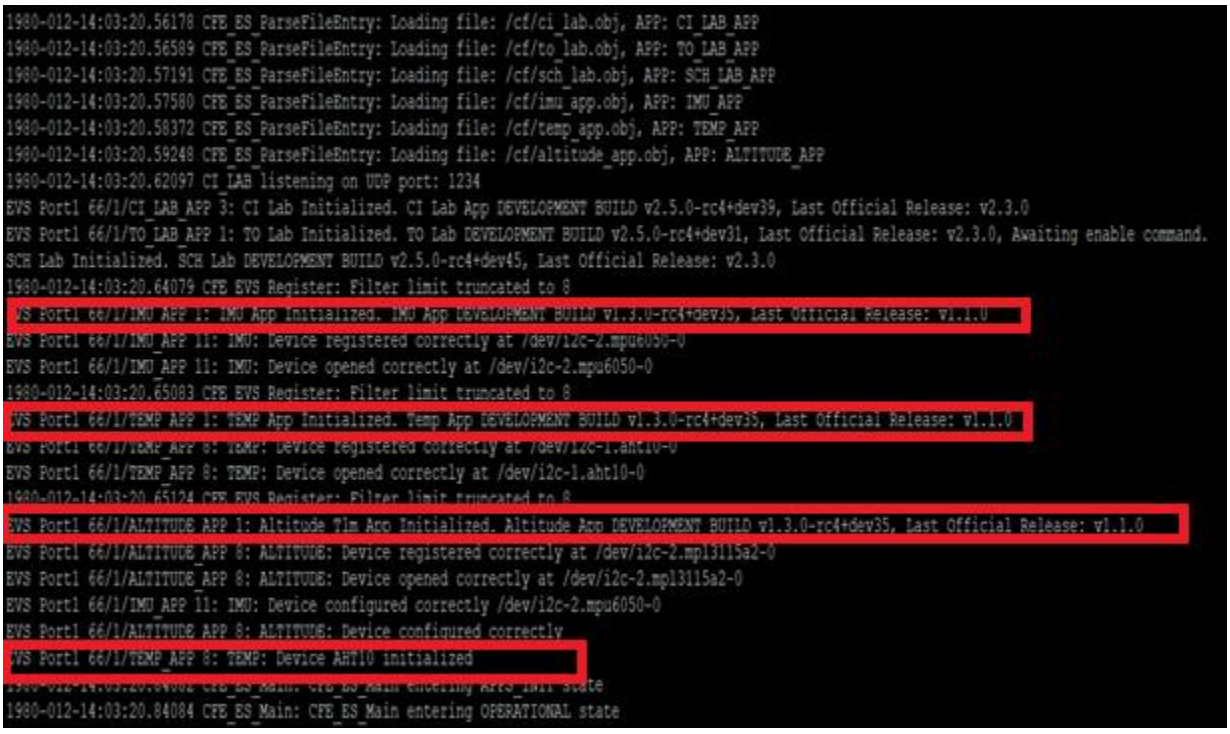
Resultados esperados. Se espera que el cFS se ejecute y cargue las tres aplicaciones de monitoreo sin ningún problema. Todas las aplicaciones deben ser iniciadas y cada una debe inicializar correctamente al sensor correspondiente. Posterior a ello se deben imprimir en

pantalla las lecturas de temperatura y humedad registradas por el sensor y las lecturas de temperatura enviadas por las otras aplicaciones.

Resultados obtenidos. Como se observa en la Figura 49 todas las aplicaciones son cargadas e inician correctamente. Una vez que todos los sensores fueron inicializados, las aplicaciones para monitoreo de inercia y altitud empezaron a registrar y enviar las medidas de temperatura de sus respectivos sensores.

Figura 49

Ejecución de las aplicaciones para monitoreo e inicialización de los sensores



Como se aprecia en la Figura 50 estas fueron recibidas correctamente. Por último, al comparar la lectura de temperatura del sensor AHT10 con respecto a la del termómetro digital, misma que se observa en la Figura 51, se puede notar que existe un error de aproximadamente 0.19 ºC. Por consiguiente, se puede considerar que la prueba realizada ha sido exitosa.

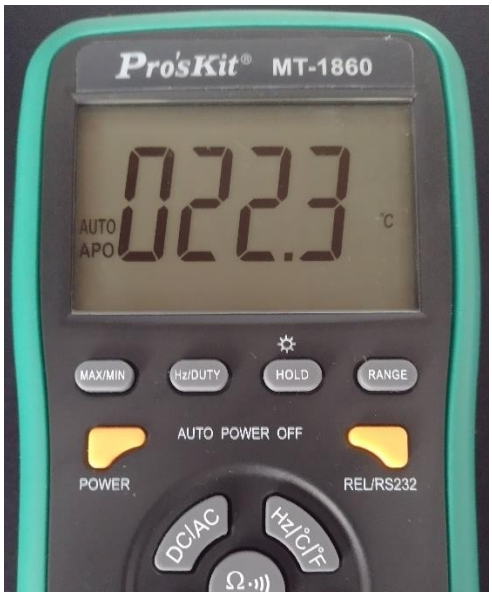
Figura 50

Lectura de la temperatura y humedad mediante el sensor AHT10 y lectura de temperatura mediante los sensores MPU6050 y MPL3115A2

```
EVS Port1 66/1/TEMP_APP 8: TEMP: Temp = 22.494316 deg C | Hum = 54.018688 % | Temp MPU6050 = 23.777060 deg C | Temp MPL3115A2 = 22.312500 deg C
EVS Port1 66/1/TEMP_APP 8: TEMP: Temp = 22.494316 deg C | Hum = 54.018688 % | Temp MPU6050 = 23.824118 deg C | Temp MPL3115A2 = 22.312500 deg C
EVS Port1 66/1/TEMP_APP 8: TEMP: Temp = 22.494316 deg C | Hum = 54.018688 % | Temp MPU6050 = 23.918236 deg C | Temp MPL3115A2 = 22.312500 deg C
EVS Port1 66/1/TEMP_APP 8: TEMP: Temp = 22.494316 deg C | Hum = 54.018688 % | Temp MPU6050 = 23.824118 deg C | Temp MPL3115A2 = 22.312500 deg C
```

Figura 51

Temperatura de referencia para la prueba realizada



Aplicación de telemetría

Procedimiento. Por medio de esta prueba se busca verificar el correcto funcionamiento de la aplicación de telemetría RF_TLM_App. Al igual que con las otras aplicaciones, es necesario que esta aplicación se cargue, se inicialice, e inicie correctamente. Esta prueba, al igual que en el caso anterior, requiere que otras aplicaciones sean cargadas, en este caso todas las aplicaciones de monitoreo (de inercia, altitud y temperatura). Esto se debe a que la aplicación de telemetría recibe los paquetes enviados por otras aplicaciones para posterior a ello enviarlos al microcontrolador conectado al transceptor.

A fin de realizar esta prueba, se deben agregar las aplicaciones IMU_APP, TEMP_APP, ALTITUDE_APP Y RF_TLM al cFS. Por lo tanto, se modificó la lista de aplicaciones en el archivo “targets.cmake” de la siguiente forma:

SET(cpu1_APPLIST ci_lab to_lab sch_lab imu_app rf_tlm temp_app altitude_app)

De la misma manera, se modificó el archivo “cpu1_cfe_es_startup.scr” en donde se agregó las líneas:

CFE_APP, imu_app, IMU_APP_Main, IMU_APP, 90, 16384, 0x0, 0;

CFE_APP, temp_app, TEMP_APP_Main, TEMP_APP, 90, 16384, 0x0, 0;

CFE_APP, altitude_app,ALTITUDE_APP_Main, ALTITUDE_APP, 90, 16384, 0x0, 0;

CFE_APP, rf_tlm, RF_TLM_Main, RF_TLM, 90, 16384, 0x0, 0;

Dado que están conectadas las aplicaciones de monitoreo, es necesario que los sensores estén conectados correctamente. En cuanto a la aplicación de telemetría, dado que esta envía los datos al microcontrolador, es primordial que este se encuentre adecuadamente conectado al OBC.

Resultados esperados. Igual que en las pruebas anteriores se espera que las aplicaciones sean cargadas e iniciadas correctamente. En el caso de las aplicaciones de monitoreo es necesario que se inicialicen adecuadamente los sensores y posterior a ello, la aplicación de telemetría debe imprimir en pantalla que aplicaciones han enviado el paquete de telemetría.

Resultados obtenidos. Como se puede observar en la Figura 52 todas las aplicaciones se han cargado e iniciado de manera adecuada. Una vez que todos los dispositivos se han inicializado, la aplicación de telemetría empieza a recibir los paquetes enviados por otras aplicaciones, como se puede apreciar en la Figura 53. Además, en esta imagen se puede

observar que los paquetes han sido enviados exitosamente, por lo que se puede concluir que la prueba de funcionamiento fue sido exitosa.

Figura 52
Ejecución de la aplicación para envío de telemetría por comunicación RF

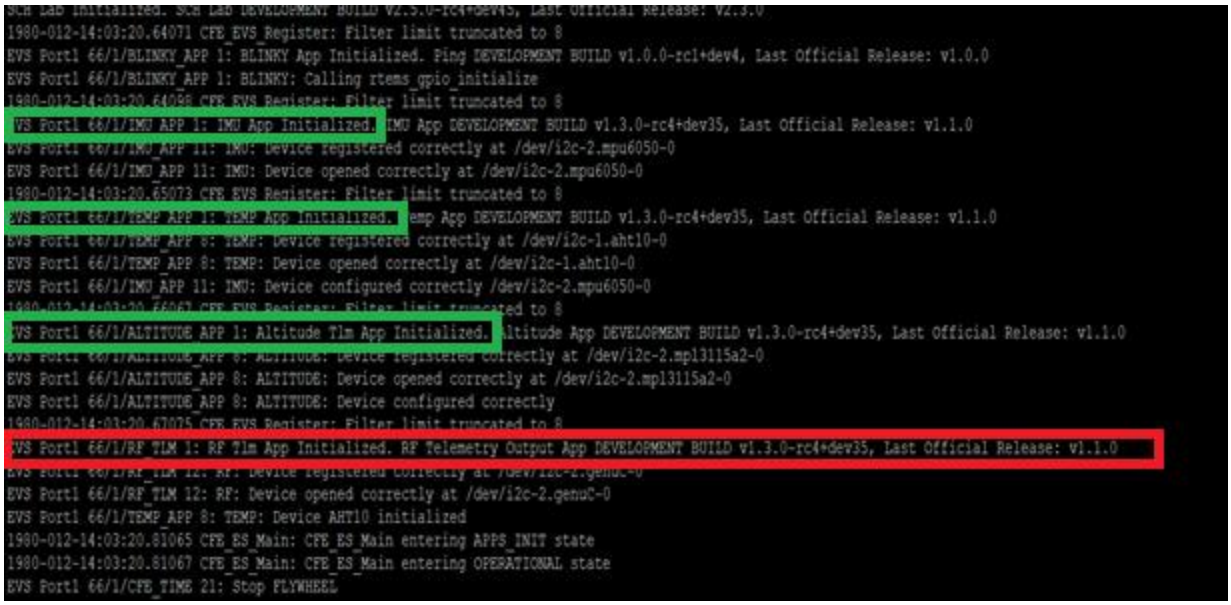
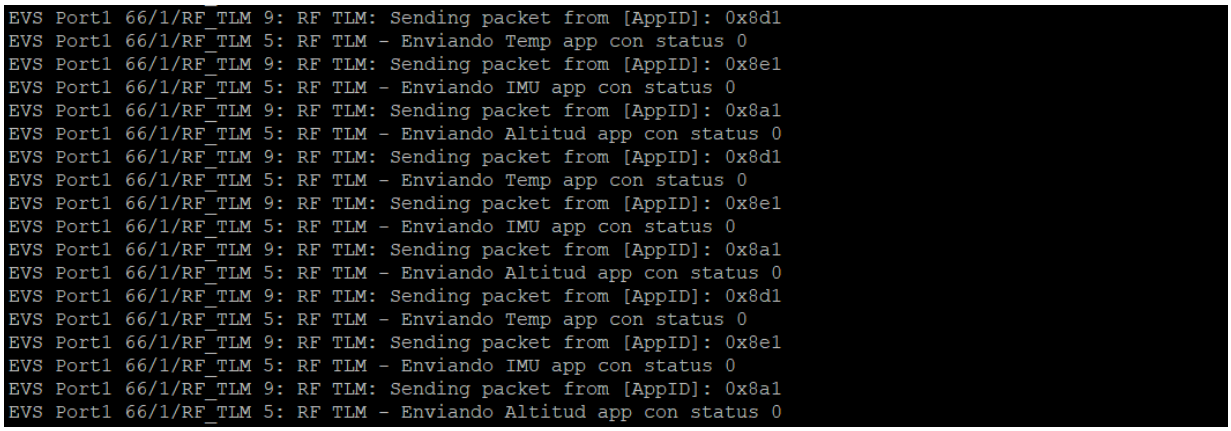


Figura 53
Recepción y envío de los paquetes de telemetría de las aplicaciones de usuario



Funcionamiento del sistema total con el sistema terrestre

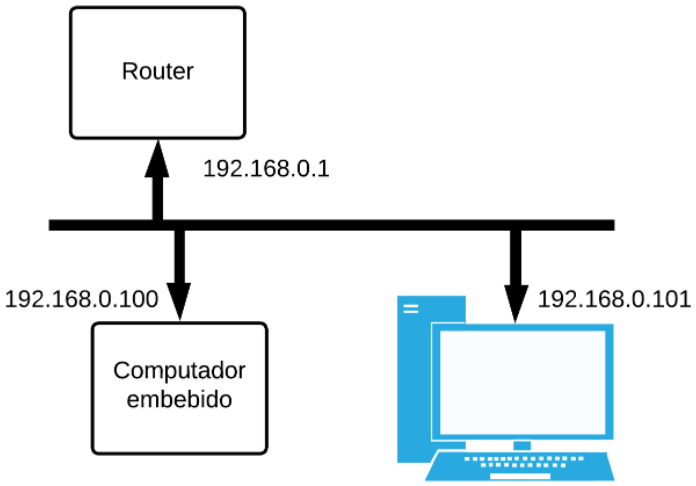
Para validar el funcionamiento del sistema completo se realizaron pruebas de funcionalidad con el sistema terrestre. Al principio se comprobó la correcta ejecución del

sistema utilizando la aplicación de escritorio para el sistema terrestre proporcionada en la distribución del cFS y después se realizaron las pruebas con el sistema terrestre implementado.

Prueba de funcionamiento con el sistema terrestre que utiliza protocolo UDP/IP

Procedimiento. El objetivo de esta prueba es comprobar el OBC es capaz de recibir telecomandos y enviar telemetría. Para ello se utilizó la aplicación de escritorio del sistema terrestre que se encuentra en la distribución del cFS. Dado que el sistema terrestre que se encuentra en la distribución del cFS utiliza protocolo UDP/IP para el envío de telecomandos y recepción de telemetría se procedió a conectar a través de un router al computador embebido y al computador de la estación terrestre, como se observa en la Figura 54.

Figura 54
Conexión entre el computador embebido y el computador de la estación terrestre



Para ejecutar la aplicación de escritorio es necesario acceder al directorio de herramientas del cFS. Acto seguido se debe ejecutar la aplicación a través de los siguientes comandos:

- $ cd cFS/tools/ cFS-GroundSystem
- $ python GroundSystem.py

Es necesario notar que las conexiones de los diferentes sensores y dispositivos al OBC, se deben haber realizado correctamente.

Resultados esperados. Se espera que el computador de a bordo ejecute las acciones correspondientes cuando se envía un comando desde la estación terrestre. Agregando a lo anterior, se espera recibir la telemetría de las distintas aplicaciones de usuario.

Resultados obtenidos. Como se puede observar en la Figura 55 al abrir la ventana principal (ventana a la izquierda) de la aplicación se seleccionó la dirección IP del OBC para poder enviar telecomandos y recibir telemetría. A través de la ventana de telecomandos (ventana de la derecha) se seleccionaron las diferentes aplicaciones para enviar comandos NOOP a cada aplicación de usuario, los cuales fueron receptados de manera correcta por el OBC como se observa en la Figura 56. Por último, se comprobó que el computador de a bordo envíe la telemetría para lo cual se abrieron las ventanas de telemetría particular de cada aplicación como se observa en la Figura 57. En evidencia de los resultados obtenidos se puede concluir que esta prueba de funcionalidad fue exitosa.

Figura 55
Ventana principal y ventana de comandos de la aplicación de escritorio de la distribución del cFS

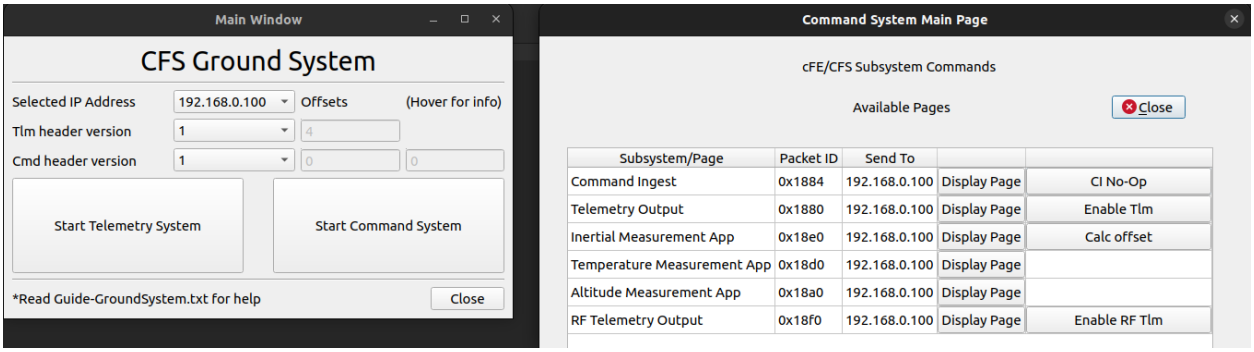


Figura 56
Recepción de comandos NOOP en el computador de a bordo

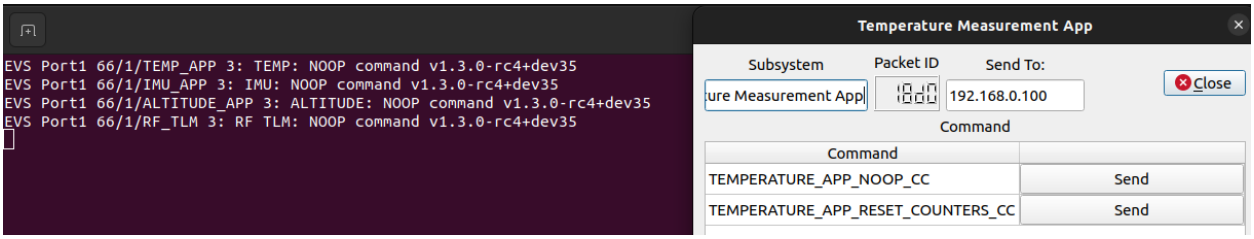
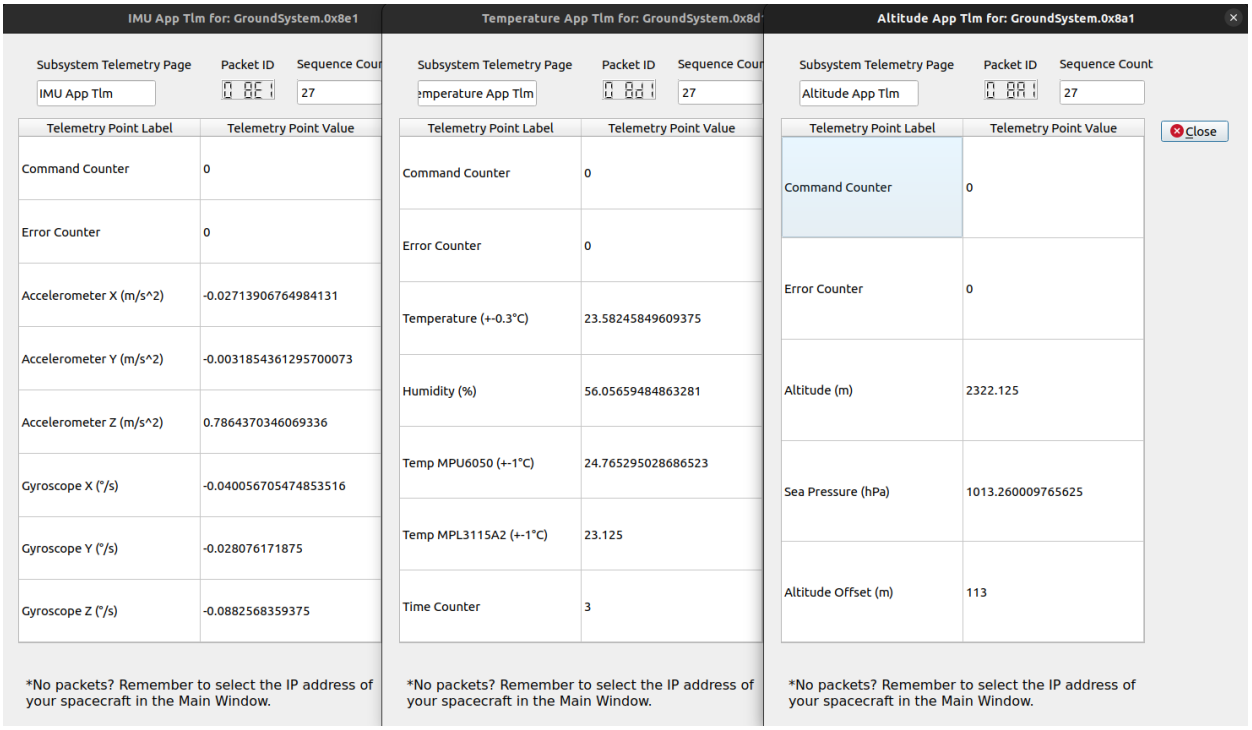


Figura 57
Ventanas de telemetría de las aplicaciones de usuario



Prueba de funcionamiento con el sistema terrestre que utiliza protocolo LoRa RF

Procedimiento. El objetivo de esta prueba es comprobar que el computador embebido que emula el nanosatélite se comunica con el sistema terrestre vía radio frecuencia. En otras palabras, se prueba el total funcionamiento del sistema, siendo el transceptor del sistema terrestre, el que recibe los paquetes de telemetría enviados por el OBC, información que es procesada y desplegada por la aplicación de escritorio del sistema. Para ejecutar la aplicación

de escritorio es necesario acceder al directorio de herramientas del cFS . Acto seguido se debe ejecutar la aplicación a través de los siguientes comandos:

- $ cd cFS/tools/RF-TelemetrySystem/Desktop
- $ python GroundSystem.py

Cabe señalar que previo la ejecución de esta experiencia, se hicieron las pruebas necesarias para la comprobación de las conexiones entre los distintos dispositivos hardware del sistema tanto del nanosatélite como del segmento terrestre y el enlace entre ambos segmentos.

Resultados esperados. Se espera que se reciban los paquetes de telemetría en el segmento terrestre para que estos puedan ser visualizados mediante la ventana de telemetría particular de cada aplicación. Esto quiere decir que el funcionamiento del sistema es el correcto y las conexiones realizadas son las adecuadas.

Resultados obtenidos. Como se puede observar en la Figura 58, se ha utilizado la ventana principal (izquierda) para seleccionar el puerto serial del computador de la estación terrestre. Este puerto está conectado al microcontrolador que realiza las funciones de conversor de protocolos entre el transceptor y el computador. Una vez abierta la ventana de telemetría general empiezan a llegar paquetes de telemetría.

En el caso de la Figura 58 se puede observar que ha llegado un total de 60 paquetes de los cuales 16 son de la aplicación de monitoreo de altitud, 17 de la aplicación de monitoreo de inercia y 27 de la aplicación de monitoreo de temperatura. Una vez que se abre la ventana de telemetría particular de cada aplicación se puede observar la decodificación del contenido de los paquetes. Así se puede observar la información del monitoreo de la altitud, inercia y temperatura en la Figura 59, Figura 60 y Figura 61 respectivamente.

Figura 58

Ventana principal y ventana de telemetría general de la aplicación de escritorio del segmento terrestre

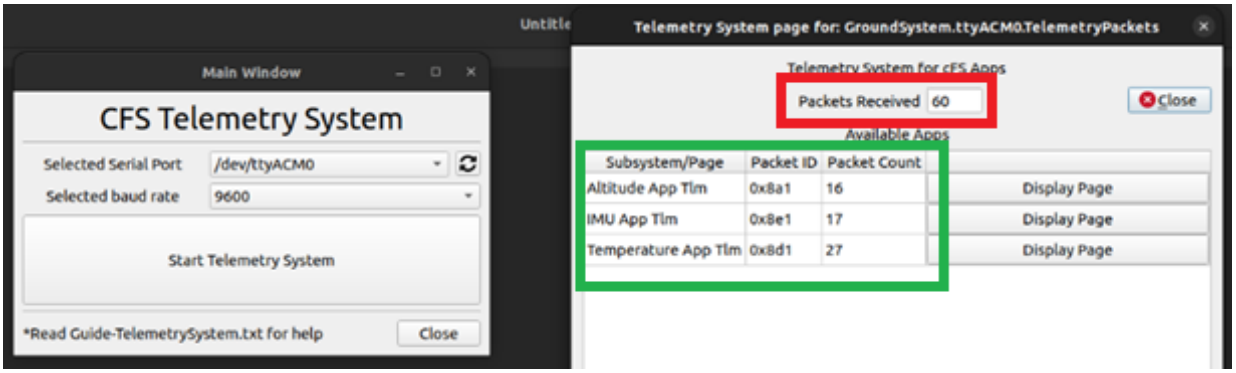


Figura 59

Ventana de telemetría particular de la aplicación de monitoreo de altitud

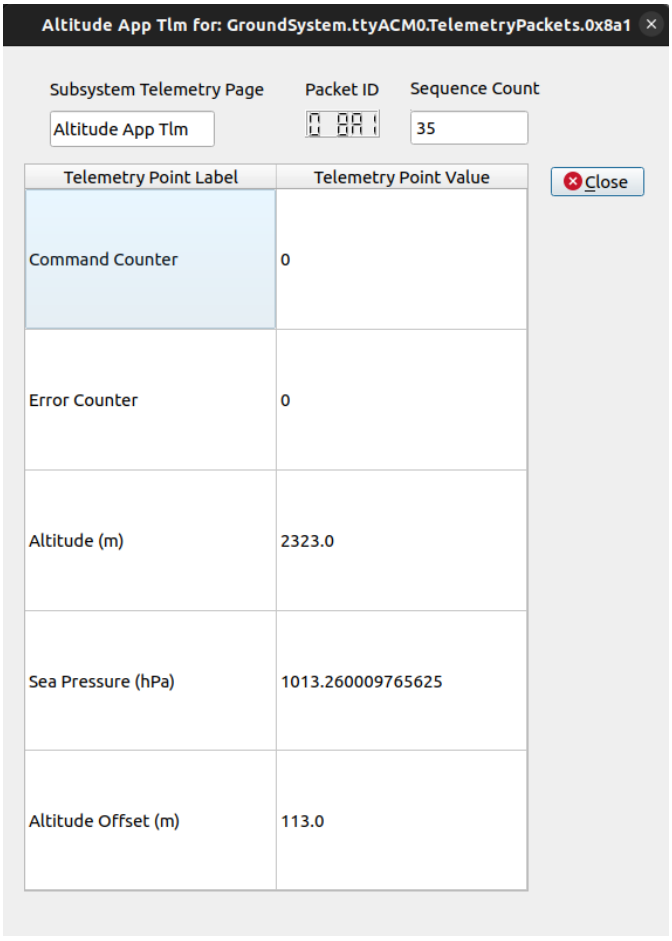


Figura 60

Ventana de telemetría particular de la aplicación de monitoreo de inercia

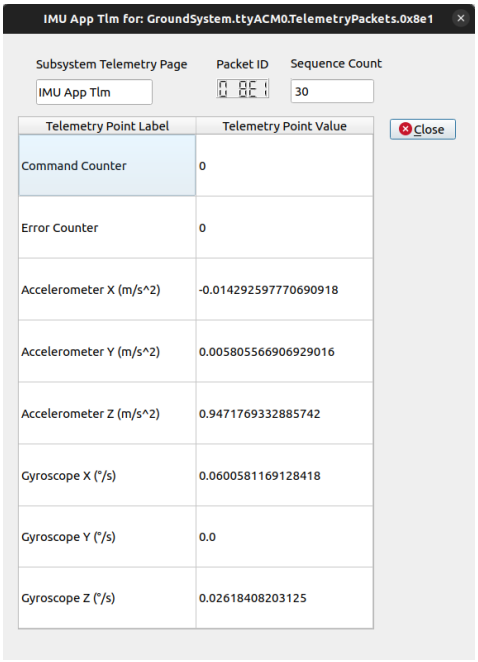
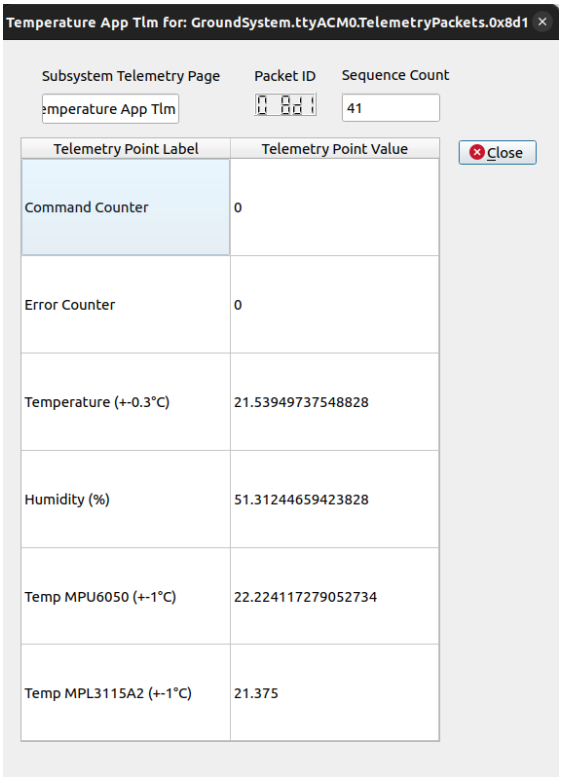


Figura 61

Ventana de telemetría particular de la aplicación de monitoreo de temperatura



Pruebas de desempeño

Consumo de energía

Procedimiento. Como se mencionó previamente, los nanosatélites, no siempre están expuestos a la luz solar, por lo que requieren de baterías recargables para su funcionamiento. Sin embargo, es importante notar que, debido a las limitaciones de tamaño, los nanosatélites no pueden llevar en su interior una gran cantidad de baterías. Por lo tanto, minimizar el consumo de energía de los componentes de los nanosatélites es uno de los factores más importantes a considerar. Por esta razón, se realizó esta prueba de consumo de energía para evaluar esta característica en el sistema implementado. Para ello, se conectó al sistema a una fuente de 5V y se midió la corriente de entrada al sistema mediante un multímetro Pro'skit MT-1860.

Dado que el multímetro tiene la capacidad de enviar las medidas a través de una conexión serial, se lo conectó a un computador para registrar las mediciones de corriente a través de la aplicación de escritorio HandCOM, distribuida por el fabricante del multímetro. Durante la prueba se realizaron 322 medidas a lo largo de un periodo de tiempo de 3 minutos y 7 segundos. En el transcurso de la prueba el sistema funcionó con las tres aplicaciones de medición (inercia, altitud y temperatura) y estuvo enviando telemetría al sistema terrestre implementado, es decir a través del protocolo LoRa RF.

Resultados obtenidos. Una vez se exportaron los datos de la aplicación de escritorio se pudieron obtener la corriente máxima, mínima y promedio consumida por el sistema en funcionamiento, las cuales se detallan en la Tabla 18. A partir de las mediciones se pudo realizar la gráfica de la Figura 62. Como se puede observar en la gráfica de la corriente, el valor medido de la corriente máxima se da cuando el sistema se enciende. A partir de los valores obtenidos se puede determinar la potencia mediante la ecuación:

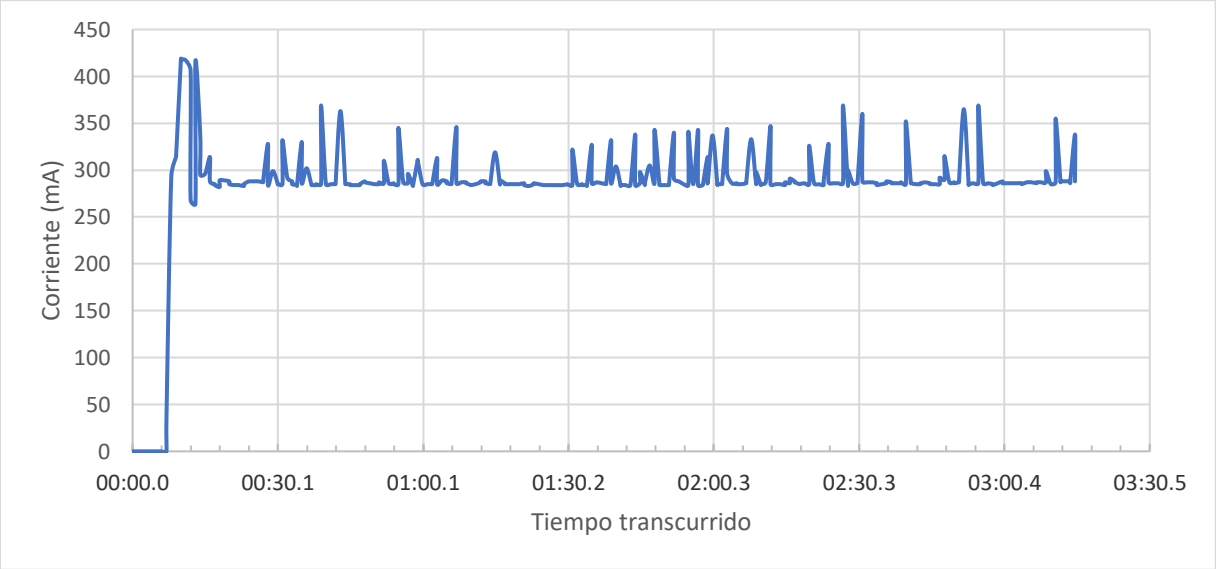
$$P = V \cdot I \tag{2}$$

Donde V es el voltaje de la fuente e I es la intensidad de corriente que atraviesa el sistema. En este caso, se calculó la potencia a partir de la corriente promedio por lo que se obtuvo que, en funcionamiento el sistema consume 1.47 W.

Tabla 18
Corriente máxima, mínima y promedio a partir de las mediciones realizadas

Medición	Valor (mA)
Corriente máxima	419
Corriente mínima	264
Corriente promedio	294.180

Figura 62
Medición de la corriente del sistema implementado



Alcance de la comunicación RF

Procedimiento. Mediante esta prueba se busca determinar el alcance de la señal de radio emitida por el computador de a bordo. Se realizaron varias pruebas tanto en un parque de la ciudad de Quito como en el campus Matriz de la Universidad de las Fuerzas Armadas ESPE. Como previsto la línea de vista del enlace juega un papel fundamental en la medición del alcance. Para ello se configuró los equipos de transmisión con la opción de LoRa a larga

distancia. Para las pruebas se dejó en un lugar fijo el computador embebido porque el sistema requiere alimentación y el presente proyecto no considera una alimentación por baterías. Para la emulación de la estación terrestre se utilizó un computador portátil en el que estaba instalada la aplicación de escritorio para recibir la telemetría. En este caso, para determinar el alcance de la señal se decidió verificar que lleguen paquetes de telemetría a la estación terrestre. Con este fin se conectó al computador embebido a la red eléctrica y el sistema terrestre se movilizó, alejándose del computador embebido, hasta que se identificó que ya no llegaban más paquetes. Una vez que ceso la recepción de telemetría se midió la distancia entre el sistema implementado y la estación terrestre. Para medir la ubicación de los puntos se utilizó google maps y luego se calcularon las distancias. La Figura 63 y Figura 64 muestran la ubicación del nanosatélite (posición A), de la estación terrestre en dirección sureste (posición B), y de la estación terrestre en dirección suroeste (posición C) respectivamente

Resultados obtenidos. Una vez que ya no se recibieron paquetes de telemetría en la estación terrestre se midieron 540 m como máximo en la dirección sureste en tanto que en la dirección suroeste se midió como máximo 694 m, como se observa en la Figura 63 y Figura 64 respectivamente. Se estima que el alcance de la señal de radio sería mayor que las medidas, considerando que al realizar las pruebas, no se pudo tener una línea de vista directa entre la estación terrestre y el computador embebido.

Figura 63

Medición del alcance de la comunicación RF entre puntos A y B

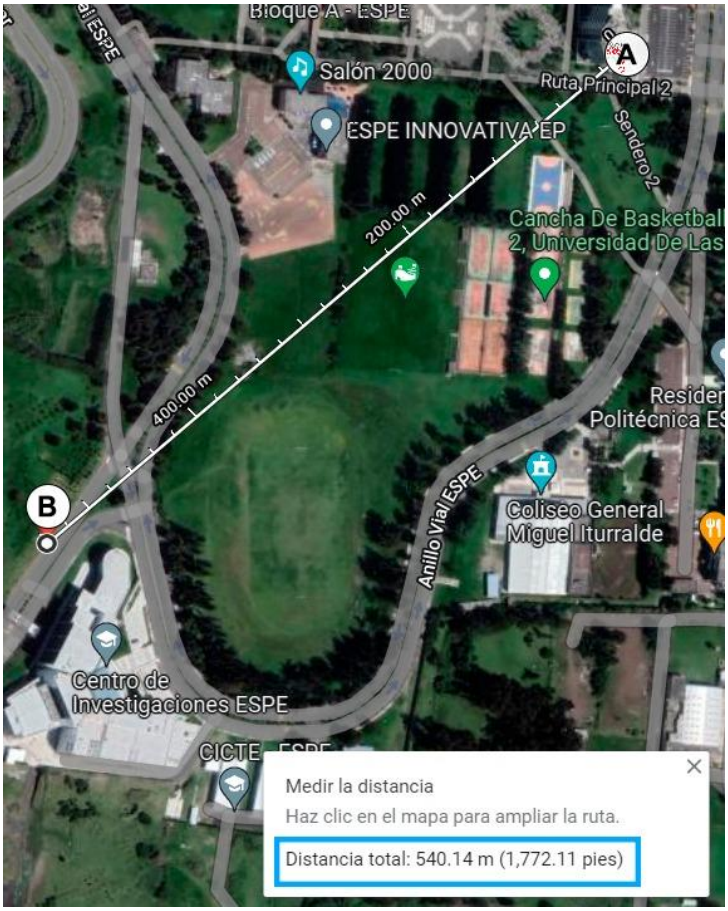
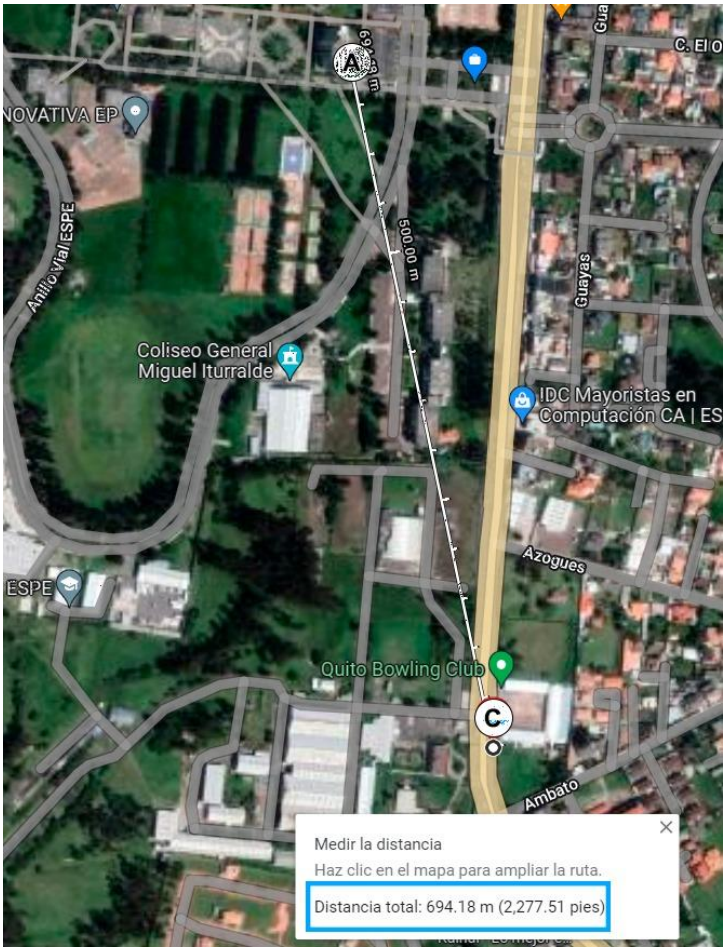


Figura 64

Medición del alcance de la comunicación RF entre puntos A y C



Conclusiones

- El objetivo principal del presente proyecto de titulación se cumplió al lograr emular las funciones de manejo de datos y envió de telemetría, propias de un computador de a bordo de un nanosatélite utilizando para ellos un computador embebido tipo COTS.
- La selección del sistema operativo en tiempo real multiplataforma RTEMS y del sistema de control de misión portable cFS, ambos de código abierto, aseguran que el software del computador de a bordo pueda ser reutilizado en diferentes plataformas de hardware y por lo tanto en diferentes misiones espaciales. Factor que maximiza su utilidad, considerando que para la selección de la placa del OBC se deben considerar además otros aspectos claves de la misión, como el tipo de órbita, propósito, consumo, entre otros.
- La selección de la plataforma de hardware BeagleBone Black, además de ser compatible con RTEMS y cFS, permitió cumplir con los requisitos de minimizar consumo de potencia y precio accesible. Características que cumple la placa seleccionada ya que tiene un consumo de corriente bajo y el precio bordea los USD $73 en el mercado local.
- La metodología de diseño, desarrollo e implementación por capas de software sobre el computador embebido permitió realizar el monitoreo vía telemetría de las variables de inercia, altitud y temperatura de los distintos sensores del nanosatélite. De esta manera se obtuvo una plataforma de codiseño hardware/software capaz de emular la función básica de manejo de datos de un computador de a bordo de un nanosatélite.
- El trabajar con lenguajes abiertos favoreció el desarrollo de los aplicativos. Así, por ejemplo, el contar con un sistema operativo y sistema de control de misión de código abierto en lenguaje C facilitó la implementación de las diferentes aplicaciones. Por

otra parte, la implementación de la aplicación de escritorio, que sirve para visualizar los datos enviados por el OBC mediante el protocolo de comunicación LoRa RF, se realizó utilizando lenguaje Phyton, minimizando el costo de la plataforma de desarrollo.

- Mediante las diferentes pruebas de funcionalidad se validó que el sistema implementado en el computador embebido funcione de manera adecuada. Además, se elaboraron pruebas de desempeño mediante las cuales se estableció el consumo de energía del sistema en funcionamiento, el cual es 1.47 W y se buscó determinar el alcance de la señal de radio emitida por el OBC, el cual se estableció que es al menos 694 m. Sin embargo, se presume que debido a la geografía del lugar en el que se realizaron las pruebas no fue posible obtener el alcance máximo en condiciones de línea de vista.

Recomendaciones

- En caso de necesidad de réplica o ampliación del presente proyecto de titulación se recomienda seguir los pasos detallados en el manual de instalación, mismo que se encuentra en el Apéndice C o a su vez, en el repositorio del autor, en el sitio web https://github.com/edison369/RTEMS-cFS-BeagleBone-Black.
- Para el diseño de nuevas aplicaciones, para la conexión con el sistema de control de misión se recomienda utilizar la herramienta de entrenamiento cFS-101, la cual proporciona los recursos necesarios para aprender a desarrollar aplicaciones con el sistema de control de misión cFS. Se puede acceder a esta herramienta a través del sitio web https://github.com/nasa/CFS-101.
- Si se requiere probar otro computador embebido se recomienda considerar al Raspberry Pi Zero, Raspberry Pi 3 A+ y Raspberry Pi 4B. Estos tres computadores embebidos son de precio similar al que se usó en el presente proyecto de titulación, además, son compatibles con el sistema operativo en tiempo real RTEMS.
- En caso de optar por otro computador embebido, si se desea mantener el sistema operativo RTEMS, se recomienda seguir el manual de instalación del RTOS que se encuentra en la página web https://docs.rtems.org/branches/master/user/index.html.
- Si se desea probar otro sistema operativo se recomienda utilizar VxWorks ya que es uno de los RTOS compatibles con el sistema de control de misión cFS. Otra alternativa es FreeRTOS, sin embargo, para poder utilizar el mismo sistema de control de misión es necesario portarlo al sistema operativo, para ello se deben realizar cambios a la capa de la plataforma de abstracción (OSAL y PSP) del cFS.
- Realizar las mediciones del alcance de la señal de radio emitida por el OBC en un lugar amplio y en el que, durante toda la prueba, se mantenga una línea de visión directa con el objeto de estudio.

Trabajos futuros

Dado que este proyecto representa una de las etapas iniciales en el desarrollo de un nanosatélite, es posible seguir ampliando la funcionalidad del sistema. Por esta razón, a través de este documento se ha detallado el desarrollo de este proyecto, con el propósito de que futuros estudiantes puedan iniciar con conocimientos previos sobre el tema. Además, es por este motivo que se han publicado todos los recursos de este proyecto en un repositorio público en GitHub, de tal manera que cualquier persona pueda acceder libremente y trabajar sobre el proyecto desarrollado. De esta manera será posible continuar con el desarrollo de otras aplicaciones tanto del sistema de control de misión como por ejemplo control de energía, de alimentación por paneles solares, etc.

Tomando en consideración que ya se ha implementado la comunicación del OBC con diferentes sensores, en el futuro sería interesante incluir actuadores en el diseño e implementar técnicas de control para que sea posible desarrollar el subsistema de determinación y control de orientación y el subsistema de control térmico.

Adicionalmente, se deben probar extensamente la comunicación por telemetría, con el objeto de probar el alcance deseado, utilizando de ser necesario, diferentes protocolos y tecnología de comunicaciones.

Finalmente, es importante notar, que el desarrollo de los nanosatélites es una tarea en la que pueden participar múltiples disciplinas como ingenieros electrónicos, eléctricos, mecánicos, geógrafos, de sistemas, físicos, entre otros por lo que se podrían elaborar varios proyectos en los que se requiera la participación de estudiantes de las diferentes carreras de la Universidad de las Fuerzas Armadas ESPE.

Bibliografía

Addaim, A., Kherras, A., & Zantou, E. B. (2010). Design of Low-cost Telecommunications CubeSat-class Spacecraft. En *Aerospace Technologies Advancements.* IntechOpen. doi:https://doi.org/10.5772/6925

Agencia Espacial Europea. (2 de Marzo de 2020). *Polar and Sun-synchronous orbit*. Enabling & Support | ESA: https://www.esa.int/ESA_Multimedia/Images/2020/03/Polar_and_Sun-synchronous_orbit

Agencia Espacial Europea. (s.f.). *About Payload Systems*. Space Engineering & Technology | ESA: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/About_Payload_Systems

Agencia Espacial Europea. (s.f.). *RTEMS.* Enabling & Support | ESA: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Software_Systems_Engineering/RTEMS

Agencia Espacial Europea. (s.f.). *Satellite frequency bands*. Applications | ESA: https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands

Alén Space. (s.f.). *Guía básica de nanosatélites*. Recuperado el 21 de Mayo de 2022, de Nanosatélites - Alén Space: https://alen.space/es/guia-basica-nanosatelites/#0

Alvarado, O. S. (2019). La importancia de un sistema aeroespacial en Latinoamérica: Avances en el caso peruano. *Pensamiento Conjunto*(1), 73-77. http://www.pensamientoconjunto.com.pe/index.php/PC/article/download/47/46

Amadis, H. (2021). *Development of an on-board computer for a nanosatellite.* [Tesis de maestría, Université de Liège]. https://matheo.uliege.be/handle/2268.2/11659

Amazon Web Services. (26 de Enero de 2022). *The FreeRTOS™ Kernel*. FreeRTOS: https://www.freertos.org/RTOS.html

Anvari, A., Farhani, F., & Niaki, K. (2009). Comparative Study on Space Qualified Paints Used for Thermal Control of a Small Satellite. *Iranian Journal of Chemical Engineering(IJChE), 6*(2), 50-62. http://www.ijche.com/article_10361_3e7db599f47c97944b269467141aeb1a.pdf

Arenas, J. P., & Margasahayam, R. N. (2006). Noise and vibration of spacecraft structures. *Ingeniare. Revista chilena de ingeniería, 14*(3), 251-264. doi:https://dx.doi.org/10.4067/S0718-33052006000200009

ASAIR. (16 de Julio de 2019). *AHT10 Technical Manual.* ECA e-shop: https://server4.eca.ir/eshop/AHT10/Aosong_AHT10_en_draft_0c.pdf

Báez, A., & Rodríguez, O. (2013). *Diseño de los sistemas estructural, de alimentación de energía solar y construcción de prototipo estructural de un picosatélite para el C.I.E. de la ESPE.* [Tesis de pregrado, Universidad de las Fuerzas Armadas ESPE]. http://repositorio.espe.edu.ec/handle/21000/6830

Balarezo, J. (2012). *Configuración y análisis del sistema de transmisión y recepción del Picosatélite Cubesat kit.* [Tesis de pregrado, Universidad de las Fuerzas Armadas ESPE]. http://repositorio.espe.edu.ec/handle/21000/6239

Bocchino, R., Canham, T., Watney, G., Reder, L., & Levison, J. (2018). F Prime: An Open-Source Framework for Small-Scale Flight Software Systems. *Small Satellite Conference.*

Pasadena: California Institute of Technology. https://digitalcommons.usu.edu/smallsat/2018/all2018/328/

Braun, T. M. (2012). *Satellite Communications Payload and System.* Hoboken: John Wiley & Sons.

Coley, G. (22 de Mayo de 2014). *BeagleBone Black System Reference Manual*. BeagleBoard.org: https://cdn.sparkfun.com/datasheets/Dev/Beagle/BBB_SRM_C.pdf

Comisión de Ciencia y Tecnología para el Desarrollo de las Naciones Unidas. (2020). *Contribución de las tecnologías espaciales al desarrollo sostenible y ventajas de la colaboración internacional en la investigación sobre este ámbito.* Ginebra. https://unctad.org/system/files/official-document/ecn162020d3_es.pdf

Cooke, C. (2012). Implementation of a Real-Time Operating System on a Small Satellite Platform. *Space Grant Undergraduate Research Symposium*. https://spacegrant.colorado.edu/COSGC_Projects/symposium/2012/08_Implementation%20of%20a%20Real-Time%20Operating%20System%20on%20a%20Small%20Satellite%20Platform.pdf

DIEEC. (2011). *Sistemas Embebidos.* Recuperado el 22 de Mayo de 2022, de Departamento de Ingeniería Eléctrica, Elctrónica y Control - UNED: http://www.ieec.uned.es/investigacion/dipseil/pac/archivos/informacion_de_referencia_ise5_3_1.pdf

Digilent. (12 de Septiembre de 2022). *Cora Z7*. Digilent Reference: https://digilent.com/reference/programmable-logic/cora-z7/start

Dvorak, D. L. (2009). *NASA Study on Flight Software Complexity.* California: Jet Propulsion Laboratory | NASA. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf

egnite GmbH. (s.f.). *Nut/OS*. Nut/OS | Modular RTOS: http://www.ethernut.de/en/firmware/nutos.html

egnite. (s.f.). *Nut/OS*. Embedded Systems | egnite: https://www.egnite.de/products/embedded-systems/nut-os/

Eickhoff, J. (2012). *Onboard Computers, Onboard Software and Satellite Operations.* Immenstaad: Springer-Verlag Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-642-25170-2

Elfvelin, M. (2022). *Design and Development of the Space Campus Ground Station for Small Satellites.* [Tesis de maestría, Luleå University of Technology]. https://www.diva-portal.org/smash/get/diva2:1646645/FULLTEXT01.pdf

Embedded Artists AB. (s.f.). *LPC4088 Developer's Kit*. Products | Embedded Artists: https://www.embeddedartists.com/products/lpc4088-developers-kit/

Federal Aviation Administration. (Febrero de 2016). *Single Event Effects Mitigation Techniques Report.* Federal Aviation Administration - U.S. Department of Transportation: https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/TC-15-62.pdf

Furano, G., & Menicucci, A. (2018). Roadmap for On-Board Processing and Data Handling Systems in Space. En *Dependable Multicore Architectures at Nanoscale* (págs. 253–281). doi:https://doi.org/10.1007/978-3-319-54422-9_10

Gonzalez, C. E., Rojas, C. J., Bergel, A., & Diaz, M. A. (2019). An Architecture-Tracking Approach to Evaluate a Modular and Extensible Flight Software for CubeSat Nanosatellites. *IEEE Access, 7*, 126409-126429. doi:10.1109/ACCESS.2019.2927931

Grusin, M. (s.f.). *Serial Peripheral Interface (SPI)*. SparkFun Electronics: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

Hay, S. I., Tatem, A. J., Graham, A. J., & Rogers, D. J. (2006). Global Environmental Data for Mapping Infectious Disease Distribution. *Advances in Parasitology, 62*, 37-77. doi:https://doi.org/10.1016/S0065-308X(05)62002-7

Hoth, D. F., O'Neill, E. F., & Welber, I. (1963). The Telstar satellite system. *The Bell System Technical Journal, 42*(4), 765-799. doi:10.1002/j.1538-7305.1963.tb04019.x

Jean Timmons, E. (9 de Abril de 2020). *Core Flight System (cFS) Training.* NTRS - NASA Technical Reports Server: https://cfs.gsfc.nasa.gov/cFS-OviewBGSlideDeck-ExportControl-Final.pdf

Kongsberg NanoAvionics. (1 de Septiembre de 2022). *What is a Satellite?* Kongsberg NanoAvionics Web site: https://nanoavionics.com/blog/what-is-a-satellite/

Kubos Corporation. (2017). *Working with KubOS and an OBC*. Recuperado el 19 de Enero de 2023, de KubOS Docs: https://docs.kubos.com/1.21.0/index.html

Leonard, C. (2017). *Challenges for Electronic Circuits in Space Applications.* Recuperado el 13 de Enero de 2023, de Analog Devices Web Site: https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/Challenges-for-Electronic-Circuits-in-Space-Applications.pdf

Llorente, A. (2 de Diciembre de 2018). ¿Cuántos satélites hay orbitando la Tierra y cómo es posible que no choquen? *BBC News Mundo*. https://www.bbc.com/mundo/noticias-46408633

Loff, S. (3 de Agosto de 2017). *Explorer 1 Overview*. Explorer and Early Satellites | NASA: https://www.nasa.gov/mission_pages/explorer/explorer-overview.html

López, A. (2021). *Propuesta de desarrollo y validación de un dispositivo educativo CANSAT para fomentar el acceso a las tecnologías aeroespaciales.* Sangolquí: Capítulo Técnico RAS de la Rama Estudiantil ESPE.

Lu, Y., Shao, Q., Yue, H., & Yang, F. (2019). A Review of the Space Environment Effects on Spacecraft in Different Orbits. *IEEE Access, 7*, 93473-93488. doi:10.1109/ACCESS.2019.2927811

Lucia, B., Denby, B., Manchester, Z., Desai, H., Ruppel , E., & Colin, A. (2021). Computational nanosatellite constellations: Opportunities and challenges. *GetMobile: Mobile Computing and Communications, 25*(1), 16-23.

Lutkevich, B. (Diciembre de 2020). *Embedded System*. Recuperado el 22 de Mayo de 2022, de TechTarget Technical Support: https://www.techtarget.com/iotagenda/definition/embedded-system

Meadows, P., Du, L., Conger, D., & Teebken, T. (25 de Mayo de 2022). *Overview of Azure RTOS ThreadX*. Microsoft Azure RTOS: https://learn.microsoft.com/en-us/azure/rtos/threadx/overview-threadx

NASA. (30 de Enero de 2014). *The Coldest Spot in the Known Universe*. NASA Science: https://science.nasa.gov/science-news/science-at-nasa/2014/30jan_coldspot

NASA. (7 de Agosto de 2017). *SporeSat*. Living in Space | NASA: https://www.nasa.gov/centers/ames/engineering/projects/sporesat.html

NASA. (7 de Agosto de 2017). *What are SmallSats and CubeSats?* Science Instruments | NASA: https://www.nasa.gov/content/what-are-smallsats-and-cubesats

NASA. (18 de Enero de 2018). *F' Software Framework.* Recuperado el 19 de Enero de 2023, de F Prime Architecture: https://github.com/nasa/fprime/blob/master/docs/Architecture/FPrimeArchitectureShort.pdf

NASA. (28 de Octubre de 2022). *SCORE - Spacecraft - Details*. NASA Space Science Data Coordinated Archive: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1958-006A

Nature Astronomy. (2020). A space science boom or death by a thousand small satellites. *Nat Astron, 4*(1011). doi:https://doi.org/10.1038/s41550-020-01257-0

Organización Meteorológica Mundial. (2022). *Programas*. Organización Meteorológica Mundial: https://public.wmo.int/en/programmes

Petkov, M. P. (Enero de 2003). *The Effects of Space Environments on Electronic Components.* NASA Jet Propulsion Laboratory: https://trs.jpl.nasa.gov/bitstream/handle/2014/7193/03-0863.pdf?

Pini, A. (14 de 02 de 2019). *Por qué y cómo usar la interfaz periférica serial para simplificar las conexiones entre distintos dispositivos*. Digi-Key Electronics: https://www.digikey.com/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices

Planet Aerospace. (2020). *Quintessence of Nano-Satellite Technology: Small is Big.* Chennai: Notion Press.

Plasson, P., Cuomo, C., Gabriel, G., Gauthier, N., Gueguen, L., & Malac-Allain, L. (2016). GERICOS: A Generic Framework for the Development of On-Board Software. *DASIA 2016 - Data Systems In Aerospace, 736*. https://ui.adsabs.harvard.edu/abs/2016ESASP.736E..39P

Prasad, N. (5 de Mayo de 2022). *An overview of on-board computer (OBC) systems available on the global space marketplace*. Recuperado el 28 de Junio de 2022, de Satsearch:

https://blog.satsearch.co/2020-03-11-an-overview-of-on-board-computer-obc-systems-available-on-the-global-space-marketplace

Raspberry Pi Foundation. (s.f.). *Raspberry Pi 3 Model A+*. Products | Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-3-model-a-plus/

Raspberry Pi Foundation. (s.f.). *Raspberry Pi 4*. Products | Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

RT-Thread. (s.f.). *About RT-Thread*. RT-Thread: https://www.rt-thread.io/about.html#advantage

Semtech. (s.f.). *What are LoRa® and LoRaWAN®?* LoRa Developer Portal : https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/

The RTEMS Project. (2021). *What is RTEMS?* RTEMS Real Time Operating System (RTOS): https://www.rtems.org/

Thompson, E. (s.f.). *I2C Bus Technical Overview.* Micro Computer Control Corporation: http://www.mcc-us.com/I2CBusTechnicalOverview.pdf

Tierra, A. (2011). *Perfil del programa "Misión Espacial para Observación de la Tierra".* Sangolquí: Departamento de Ciencias de la Tierra y Construcción, ESPE.

Unión Internacional de Telecomunicaciones. (2014). *Characteristics, definitions and spectrum requirements of nanosatellites and picosatellites, as well as systems composed of such satellites.* Geneva: Unión Internacional de Telecomunicaciones. https://www.itu.int/en/ITU-R/space/Documents/R-REP-SA.2312-2014-PDF-E.pdf

Union of Concerned Scientists. (1 de Mayo de 2022). *UCS Satellite Database.* REPORTS & MULTIMEDIA - UCSUSA: https://www.ucsusa.org/resources/satellite-database

Velázquez, P. A. (2019). *Sistema de potencia de a bordo de un nanosatélite CubeSat.* [Tesis de pregrado, Universidad Nacional Autónoma de México]. http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/17140/Tesis.pdf?

Wind River Systems. (s.f.). *VxWorks*. Productos | Wind River: https://www.windriver.com/products/vxworks

Winton, A., Gerner, J. L., Michel, P., & Morgan-Owen, R. (Mayo de 1996). *The Transponder - A Key Element in ESA Spacecraft TTC Systems*. Agencia Espacial Europea: https://www.esa.int/esapub/bulletin/bullet86/wint86.htm

Zeif, R., Kubicka, M., & Hörmer, A. J. (2022). Development and application of an embedded computer system for CubeSats exemplified by the OPS-SAT space mission. *Elektrotech. Inftech, 139*, 8-15. doi:https://doi.org/10.1007/s00502-022-00991-9

Apéndices