UNIVERSIDAD LATINA DE PANAMÁ
SEDE PANAMÁ
FACULTAD DE INGENIERÍA
LICENCIATURA EN INGENIERÍA EN SISTEMAS INFORMÁTICOS

PROYECTO

"ARQUITECTURA MULTINÚCLEO PARA PROCESAMIENTO DE IMÁGENES EN SATÉLITES CANSAT DE OBSERVACIÓN TERRESTRE"

PREPARADO POR:
MARK DYLAN TREJOS CASTILLO
C.I.P. 8-999-582

PROFESOR: CARLOS FERNÁNDEZ

PROYECTO FINAL DE GRADUACIÓN PRESENTADO COMO REQUISITO PARA OPTAR POR EL TÍTULO DE LICENCIATURA EN INGENIERÍA EN SISTEMAS INFORMÁTICOS

PANAMÁ, REPÚBLICA DE PANAMÁ
2026

DEDICATORIA

Dedico este trabajo a mi padre, quien ha trabajado incansablemente durante su vida para dar las mejores oportunidades a sus hijos; y a mi madre, por todos sus consejos y su apoyo incondicional.

AGRADECIMIENTO

A mi familia, por el apoyo que me ha brindado durante todo el camino.

A mis profesores, que siempre estuvieron dispuestos a ayudarme a comprender lo que antes desconocía.

A todas las personas que han contribuido al avance de la ciencia y la computación

DECLARACIÓN JURADA

ÍNDICE GENERAL

Índice General

ÍNDICE DE GRÁFICOS

INTRODUCCIÓN

Los satélites en lata, también conocidos como ***cansats***, son herramientas clave en la educación y la investigación científica de bajo costo. Estos artefactos permiten recolectar una variedad de datos a través de sensores y capturar imágenes mediante módulos de cámara, que posteriormente pueden ser analizadas para obtener información relevante del clima y la superficie terrestre, entre otras posibles aplicaciones.

No obstante, estos dispositivos asequibles generalmente son ensamblados con microcontroladores y microprocesadores de bajo consumo eléctrico, como el *Raspberry Pi Zero 2 W.* Estas microcomputadoras presentan una capacidad de cómputo limitada. Dicha limitación impacta negativamente en el procesamiento de tareas intensivas, por ejemplo, la manipulación de imágenes.

Estos satélites suelen incorporar módulos de transmisión de datos mediante radiofrecuencias. Estos módulos permiten comunicaciones de hasta 10 kilómetros, a costa de un ancho de banda reducido, lo cual disminuye la velocidad de transferencia de datos.

Ante las restricciones expuestas, surge la necesidad de desarrollar una arquitectura efectiva que aproveche las ventajas de los microprocesadores multinúcleos modernos, los cuales pueden ser programados para el procesamiento paralelo de tareas, de modo que el sistema sea capaz de procesar y enviar más imágenes en el mismo lapso comparado con arquitecturas de un solo núcleo.

En el capítulo I se abordan los antecedentes, se realiza el planteamiento del problema, se justifica la investigación, se definen los objetivos generales y específicos, y se establece el alcance y los límites de la investigación.

En el capítulo II se revisan los antecedentes de investigaciones realizadas sobre *cansats* con sistemas de captura y transmisión de imágenes, se detallan las bases teóricas que sustentan la investigación, se definen las variables y se provee de un glosario técnico para asegurar el entendimiento del lector.

En el capítulo III se explica el tipo y diseño de la investigación, la población y muestra, el cálculo del muestreo, se define la hipótesis de la investigación, y se describe el instrumento y el procedimiento de la investigación.

En el capítulo IV se analizan e interpretan los resultados obtenidos en las pruebas realizadas con la propuesta sugerida.

En el capítulo V se hace una introducción de la propuesta, justificación de su diseño, definir su objetivo, enumerar los beneficios, escribir un apartado para el presupuesto y presentar el diseño de la propuesta.

CAPÍTULO 1.0

EL PROBLEMA

1.1 Antecedentes del problema de investigación

Los ***cubesats*** fueron ideados para formar parte de los planes de estudio de las carreras de aeronáutica y astronáutica, con el propósito de que alumnos experimentaran con las fases del ciclo completo del desarrollo de un satélite. (Chun et al., 2023)

Los ***cubesats*** son ideales para complementar el aprendizaje de los estudiantes en carreras *STEM* principalmente porque ellos constan de una gran accesibilidad que abren las puertas de entrada a este tipo de desarrollos, los factores que lo hacen accesible son el bajo coste de sus materiales y el tamaño reducido. Esta accesibilidad se manifiesta en el uso de componentes comerciales disponibles en el mercado, como la *Raspberry Pi* para la unidad de control a bordo, junto con otros módulos complementarios y software de código abierto. Estas características no solo permiten la construcción de satélites funcionales con recursos limitados, sino que también brindan a los estudiantes la oportunidad de experimentar con sistemas reales de control y comunicación espacial de forma práctica y eficiente.(Gadekar, 2025)

Los ***cubesats*** no solo han sido efectivos para contribuir con la educación de los estudiantes, sino que también han sido realmente útiles y rentables en misiones reales. Después de haberse lanzado el primer cubesat, en los 8 años siguientes se completaron 60 lanzamientos exitosos a la órbita utilizando este tipo de satélites. Entre esos lanzamientos, algunos fueron desplegados por universidades, escuelas secundarias y misiones de los Estados Unidos.(Deepak & Twiggs, s. f.)

En 1998, *DARPA* y *The Aerospace Corporation*, junto con la Universidad de Stanford, participaron en la construcción de picosatélites con masas que oscilaban entre los 0.1 y 1 kg. Ese año fue el inicio formal de los desarrollos de satélites miniaturizados con fines educativos. Estos picosatélites, debido a su reducido tamaño, requieren ser desplegados mediante un vehículo mayor. Ejemplo de este vehículo fue el *OPAL (Orbiting Picosat Automated Launcher),* un microsatélite construido por estudiantes de Stanford que logró poner en órbita seis picosatélites. Sin embargo, su desarrollo duró más de cuatro años, lo que impidió que los estudiantes conocieran el ciclo completo del desarrollo dentro de sus estudios universitarios.

En consecuencia, la Universidad de Stanford decidió reducir el tamaño de los satélites con el fin de completar todas las fases en un año académico.

Esta tendencia a la reducción de la escala contribuyó a la estandarización del formato cubesat, definido como un satélite cúbico de 10 cm por lado con una masa máxima de 1 kg. Se diseñó según la disponibilidad de celdas solares y los requisitos del empaquetamiento eficiente en el lanzador. En este proceso evolutivo de la tecnología y la educación, el profesor Bob Twiggs presentó en noviembre de 1998 el concepto de “cansat” durante el simposio University Space Systems Symposium (USSS), en Hawái. Mientras sostenía una lata de aluminio, propuso que un satélite pudiera ser construido dentro del contenedor cilíndrico.

En principio, se propuso el despliegue orbital, pero la falta de acceso a lanzamientos llevó a reducir el alcance de los proyectos, enfocándose en los lanzamientos suborbitales mediante cohetes aficionados. Este nuevo enfoque permitió simular condiciones clave del entorno espacial a menor costo y plazos de ejecución más cortos para programas universitarios. (Chun et al., 2023)

Los ***cansats*** tienen múltiples usos que abarcan en un amplio espectro de disciplinas y campos, desde la educación, vigilancia militar, estudios atmosféricos, aeronáutica, telecomunicaciones, pruebas de software y hardware. Algunos cansats integran módulos de cámara para poder tomar capturas cuando están en vuelo.

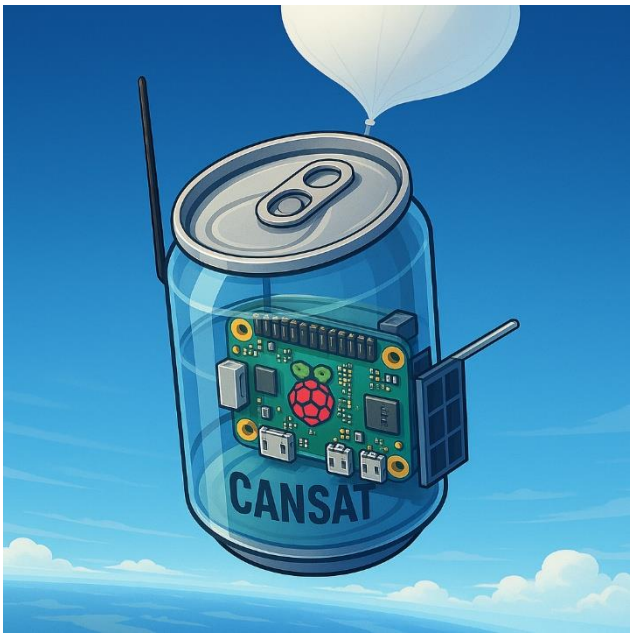


Ilustración 1. Ilustración de referencia de cansat en vuelo.

Actualmente, los fines educativos han sido los más recurrentes en el desarrollo de los cansats. Se han realizado múltiples competencias en diferentes países para envolver a los estudiantes en ciclos de desarrollo de cansats. Este tipo de torneos tienen un gran provecho para los estudiantes, debido a que los sumerge en una serie de problemas técnicos, contribuyendo a mejorar sus capacidades para resolver problemas y además acerca a los estudiantes a la realidad del campo. (Chun et al., 2023)

Procesamiento paralelo

El procesamiento paralelo es un recurso técnico que se suele implementar cuando se tiene como objetivo aprovechar los múltiples núcleos que puede poseer un CPU con el fin de incrementar el rendimiento en sistemas y aplicaciones que manejan grandes volúmenes de datos o que realizan tareas intensivas en términos de computación. Un enfoque común al momento de implementar el paralelismo es mediante el esquema Maestro-Esclavo, en este esquema, uno de los núcleos ejecuta un proceso que adquiere el rol de orquestador de los otros núcleos, el orquestador es el maestro y el resto de los núcleos orquestados son los esclavos. El maestro se encarga de distribuir las cargas entre los núcleos esclavos de manera equilibrada. Resulta crucial repartir de manera eficiente y balanceada las cargas, porque, en caso contrario, se pueden generar cuellos de botellas, anulando los beneficios que provee el paralelismo, repercutiendo en la eficiencia global del programa.(Murillo Armas, 2022)

El procesamiento paralelo se presenta como una estrategia clave en la solución de numerosos problemas científicos y de ingeniería, aunque no todos los problemas presentan el mismo grado de paralelismo ni responden de igual forma a su aplicación. En algunas tareas, debido a su naturaleza, sus ejecuciones son estrictamente seriales. Sin embargo, en un sinnúmero de campos existen problemas cuyas soluciones constan de tareas que son candidatas a ser realizadas de manera paralela, incluyendo el procesamiento de datos recolectados por satélites, el sensado remoto, la modelización

climática y oceánica, el procesamiento de imágenes, el reconocimiento de voz, la visión por computadora, el aprendizaje en redes neuronales y el análisis en bases de datos de gran volumen. Muchos de estos desafíos han sido catalogados como Grand Challenge Problems debido a su relevancia científica y económica, y a su potencial de ser resueltos mediante técnicas de computación paralela de alto rendimiento, lo cual refuerza la pertinencia de aplicar este enfoque al procesamiento de imágenes en sistemas de bajo consumo como los cansats. (Naiouf, 2004)

1.2 Planteamiento del problema

La Raspberry Pi Zero 2 es una microcomputadora con un chip de arquitectura ARM de 4 núcleos. Debido a su tamaño y arquitectura es un ordenador de bajo consumo energético. Estos factores han favorecido a su adopción como unidad computacional a bordo en muchos proyectos de cansats (ESA, 2018).

Ilustración 2. Raspberry Pi Zero 2 W utilizada durante el desarrollo de la propuesta

Raspberry Pi Zero 2W

El procesador de la *Raspberry Pi Zero 2*, lanzada en 2021, ofrece un 40% más rendimiento por hilo que su predecesor. En rendimiento multinúcleo es capaz de ofrecer

hasta 5 veces más que la *Pi Zero* original. Teóricamente esto permite la ejecución en paralelo de múltiples tareas empleando sus núcleos adicionales (Raspberry Pi, 2021).

Sin embargo, muchos programas para cansats no aprovechan completamente el paralelismo que ofrecen los procesadores multinúcleos. Tareas como el procesamiento de imágenes se siguen programando de manera secuencial. En el caso de la transmisión de datos, se diseñan los programas para esperar a que los paquetes estén completamente listos para ser enviados, lo que retrasa el tiempo de envío de datos. (González Torres et al., 2014)

Bajo el enfoque de ejecución secuencial, las imágenes son procesadas una tras otra, en lugar de distribuirse las tareas entre varios hilos. Por ejemplo, el programa toma una foto y luego procede a recortarla y agregar metadatos antes de pasar a la siguiente. Durante ese tiempo, los otros núcleos de la *CPU* permanecen inactivos. En algunos estudios se halló que la programación secuencial es menos eficiente cuando el hardware permite el procesamiento entre varios núcleos. En pruebas de laboratorio se observó que emplear múltiples hilos de ejecución puede duplicar la tasa de procesamiento de imágenes comparado con un solo hilo.

Por lo tanto, un programa optimizado para procesar imágenes en paralelo es más eficiente que un código estrictamente secuencial. Además, se observó que, al aumentar de 1 a 4 hilos, la latencia de procesamiento por imagen se redujo de ~300 ms a ~120 ms en un caso de análisis de espectrogramas. Estos datos avalan la capacidad de las

arquitecturas multinúcleos para agilizar significativamente el procesamiento. (De Marco et al., 2025)

La relevancia de resolver esta problemática radica en su impacto en la arquitectura de sistemas computacionales y en las aplicaciones prácticas de los cansats. Un procesamiento más eficiente de imágenes no solo mejora su calidad, sino también la cantidad que se puede procesar en menores tiempos. Permite a instituciones educativas con recursos limitados realizar misiones científicas más ambiciosas, como el monitoreo climático en regiones remotas.

La presente investigación propone el desarrollo de una arquitectura de software que implementa procesamiento paralelo para distribuir tareas entre los hilos disponibles del procesador, optimizando el uso de recursos y reduciendo el tiempo de ejecución. Para guiar este estudio, se plantean las siguientes preguntas de investigación:

1. ¿Cómo puede el procesamiento paralelo reducir el tiempo de procesamiento de imágenes en cansats con procesadores de bajo consumo, como el *Raspberry Pi Zero 2 W*?

2. ¿Qué algoritmo de compresión con pérdida ofrece el mejor equilibrio entre reducción de tamaño y calidad de la imagen?

3. ¿Cuáles son las tareas por distribuir de manera paralela para garantizar la transmisión continua de datos sin tiempos de envío inactivo?

Hipótesis:

1. La implementación de una arquitectura de procesamiento paralelo reducirá el tiempo de procesamiento de imágenes en cansats en al menos un 30% en recorte y preparación de los metadatos con el procesamiento serial.
2. Un algoritmo de compresión optimizado para multinúcleos reducirá el tamaño en al menos un 50%, manteniendo un tiempo de ejecución bajo y una calidad aceptable.
3. La correcta distribución de tareas paralelas garantizará la transmisión constante e ininterrumpida de datos, asegurando el uso óptimo del ancho de banda disponible.

1.3 Justificación

El presente proyecto busca ofrecer una solución a un problema de rendimiento en procesamiento digital de imágenes utilizando hardware de bajo costo, contribuyendo así a la literatura en un área de creciente interés. El campo del procesamiento de imágenes en tiempo real ha experimentado un auge en los últimos años y las computadoras *Raspberry Pi* cada vez son más utilizadas para el procesamiento de volúmenes de datos en tiempo real (Khalid Hosny et al.). Sin embargo, se ha identificado una brecha en la investigación: la falta de soluciones de software capaces de ejecutar procesamiento paralelo de imágenes.

Desde el punto de vista técnico, El proyecto se diferencia de otros por la implementación de un sistema de procesamiento de imágenes paralelizado sobre la *Raspberry Pi Zero 2 W* para reducir los tiempos de ejecución. Tradicionalmente, las imágenes en los cansats se procesan de manera secuencial debido a las limitaciones del hardware y las limitaciones técnicas; ahora se propone una solución aprovechando la *CPU* multinúcleo de la *Pi Zero 2 W*. Este micro dispositivo integra un procesador *ARM* Cortex-A53 de 4 núcleos a 1 GHz (arquitectura comparable a la Raspberry Pi 3) que es aproximadamente cinco veces más potente que el modelo Zero W original. Gracias a esta mejora de hardware, la *Zero 2 W* es capaz de manejar tareas más exigentes de visión, incluyendo captura de movimiento y reconocimiento de objetos en el mismo equipo. (Smith, 2021)

En el ámbito práctico y social, los beneficios de esta investigación se extienden a diversos sectores que requieren soluciones de bajo costo y alta eficiencia. Al desarrollar

una solución de procesamiento de imágenes acelerado en *Raspberry Pi Zero 2 W*, se democratiza el acceso a tecnologías de visión artificial y computación en paralelo. Por su bajo precio (aproximadamente 15 USD por unidad, la computadora de cuatro núcleos más económica disponible y su diminuto tamaño, la *Raspberry Pi Zero 2 W* permite que incluso proyectos con recursos limitados implementen sistemas de procesamiento de imágenes avanzados.

Por ejemplo, en el sector educativo, este tipo de desarrollo beneficiará a estudiantes e instituciones académicas que pueden replicar experimentos de visión computacional sin requerir costosos laboratorios: se ha demostrado que, con componentes comercialmente asequibles, es posible montar prototipos funcionales (e.g., cámaras infrarrojas, sensores) para proyectos de investigación estudiantil, manteniendo el costo total por sistema por debajo de $400. (He & Damas, s. f.)

1.4 Objetivo general

Diseñar e implementar una arquitectura de procesamiento paralelo para optimizar el tiempo de procesamiento de imágenes capturadas por un *cansat* con una *Raspberry Pi Zero 2 W*, reduciendo el tiempo de ejecución en al menos un 30%

Objetivos específicos:

1. Diseñar una arquitectura modular en *Python* para separar las funciones de procesamiento y compresión de imágenes, asegurando escalabilidad y mantenibilidad.

2. Implementar un método de procesamiento paralelo que divida imágenes en partes parametrizadas y las gestione mediante una cola, reduciendo el tiempo de ejecución en al menos un 30%.
3. Determinar un algoritmo de compresión que equilibre una reducción de tamaño mínima del 50% y el tiempo de ejecución, optimizado para procesadores de bajo consumo.
4. Evaluar y optimizar el rendimiento del sistema en una *Raspberry Pi Zero 2 W* de 4 núcleos, ajustando parámetros y balanceando la carga para maximizar la eficiencia.
5. Establecer una nomenclatura estandarizada para los archivos procesados, facilitando la reconstrucción de imágenes con un error de ordenamiento menor al 1%.

1.5 Alcances y limites

Alcance: Este proyecto abarca el diseño, implementación y validación de un programa optimizado para el procesamiento paralelo de imágenes en un cansat con una *Raspberry Pi Zero 2 W*, centrándose en el recorte, compresión y adición de metadatos en imágenes JPEG de 640x480 px almacenadas en un directorio. Se evaluará el desempeño mediante el tiempo de procesamiento, uso de *CPU* y eficiencia de compresión, durante tres meses en un entorno de laboratorio.

Límites: No se abordarán el diseño estructural del cansat, la integración de sensores adicionales, ni la transmisión por radiofrecuencia. El enfoque se restringe a imágenes estáticas, excluyendo video o inteligencia artificial. El desarrollo es específico para la *Raspberry Pi Zero 2 W*, limitando su aplicabilidad a otras plataformas debido a

diferencias arquitectónicas. No se procesarán imágenes con datos sensibles, asegurando consideraciones éticas.

1.6 Línea de investigación a la que pertenece el estudio

La línea de investigación a la que pertenece este estudio es "Sistemas embebidos y procesamiento de imágenes en tiempo real", con un enfoque particular en la optimización del rendimiento computacional en plataformas de bajo consumo mediante técnicas de paralelismo. Esta línea se sitúa en la intersección entre la ingeniería de software, la visión por computadora y los sistemas de adquisición y procesamiento de datos en entornos con recursos limitados, abordando desafíos relacionados con la eficiencia energética, la computación distribuida y la miniaturización de soluciones tecnológicas aplicadas a la exploración aeroespacial, educativa y científica.

CAPÍTULO 2.0

MARCO TEÓRICO

2.1 Antecedentes de investigaciones realizadas en el tema

El desarrollo de los cansats (satélites del tamaño de una lata) ha pasado de ser una herramienta educativa básica para convertirse en una plataforma versátil para experimentación científica. En sus primeras etapas, a finales de los años noventa con iniciativas como *ARLISS* en Estados Unidos y Japón, la mayoría de los proyectos se centraban en medir parámetros atmosféricos elementales —como presión y temperatura— de manera similar a una radiosonda. Con el tiempo, las misiones fueron incorporando cargas útiles más complejas, entre ellas cámaras de bajo peso, con el fin de explorar capacidades de observación que superaban lo que podían ofrecer los sensores tradicionales.

En competencias posteriores, se mejoró el diseño de los módulos de imagen para enriquecer la telemetría y documentar visualmente el vuelo. En la actualidad, incluir cámaras en los prototipos suele ser frecuente e incluso requisito en las competencias internacionales, lo que demuestra la evolución del cansat desde un proyecto educativo introductorio hacia una plataforma con aplicaciones científicas y tecnológicas más amplias. (Chun et al., 2023) ***cansats* para monitoreo visual ambiental y detección de objetos.**

En algunos proyectos se ha incorporado cámaras en cansats para apoyar misiones de monitoreo ambiental y reconocimiento visual. Por ejemplo, (Hasan et al., 2021) propusieron un prototipo de cansat que combinó sensores meteorológicos para medir variables como la temperatura, presión, humedad o calidad de aire y una cámara con

visión en primera persona. En ese sistema, las imágenes transmitidas se procesan utilizando modelos *YOLO v3 / v4* para la detección de personas u otros objetos a partir del video recibido. Los experimentos presentados indican que es viable detectar blancos terrestres desde el descenso del cansat, aunque el procesamiento de las imágenes se realiza principalmente en la estación base, no dentro del módulo.

En otros trabajos, se abordó la inclusión de observaciones visuales de monitoreo climático en pequeña escala del entorno para complementar las mediciones tradicionales. (Colin, 2015) en su estudio, por ejemplo, discute un concepto de cansat para monitoreo climático en regiones pequeñas, bajo el cual podría evaluarse la cobertura nubosa o condiciones atmosféricas mediante imágenes del cielo (all-sky), aunque el documento se enfoca principalmente en sensores de temperatura y humedad y no entra en detalle sobre el procesamiento visual integrado.

En México, algunos proyectos universitarios han incorporado microcámaras en sus diseños de cansat. En la Universidad Autónoma de Baja California, por ejemplo, aparece documentación de uso de una microcámara para capturar fotografías del entorno antes del despliegue del cansat (Ortega et al., 2022). Sin embargo, en la mayoría de esos casos, el análisis de las imágenes ocurre después de la recuperación del dispositivo, no en tiempo real durante el vuelo.

Aunque la integración de visión por computadora en *cansats* representa un avance importante para misiones híbridas (telemetría y percepción visual), persisten desafíos

técnicos no resueltos: el diseño de hardware compacto y ligero con capacidad de procesamiento a bordo, limitaciones energéticas, estabilización (vibraciones y actitud durante el descenso) y transmisión eficiente de datos visuales. Un vacío identificable es el logro de procesamiento de imágenes en paralelo a bordo del cansat, lo cual es uno de los objetivos que se busca abordar en este proyecto de investigación.

Visión artificial y navegación autónoma basada en imágenes

Un campo de investigación derivado del uso de cámaras en *cansats* es la navegación autónoma asistida por visión. Los *cansats* de competiciones avanzadas (p. ej. *ARLISS*) a menudo deben desplazarse de forma autónoma tras aterrizar, guiándose hacia un objetivo predeterminado. Para conseguir este objetivo, los investigadores japoneses implementaron algoritmos de deep learning en un *cansat* (Akiyama et al., 2021). Akiyama y Saito presentaron un método que emplea clasificación de imágenes para reconocer una baliza objetivo en el terreno hasta a 10 metros de distancia.

Durante la competencia *ARLISS 2019,* este cansat identificó correctamente la dirección del objetivo mediante su cámara frontal y logró acercarse casi por completo al punto de aterrizaje designado, ganando el primer lugar. En iteraciones posteriores refinaron el algoritmo para mejorar la tasa de acierto a mayores distancias, combinando técnicas de subdivisión de imagen y detección de regiones de interés. El resultado fue un reconocimiento robusto hasta ~10 m, superando en rendimiento a enfoques basados en detección de objetos con modelos como SSD-MobileNet, especialmente en condiciones de objetivo pequeño o lejanos. Este avance permitió que en 2020 el cansat lograra un

aterrizaje de precisión “cero-distancia”, es decir, llegó exactamente al blanco en múltiples pruebas, ganando nuevamente su categoría.

Estas investigaciones demuestran que los algoritmos de visión artificial (redes neuronales entrenadas para clasificación/detección) pueden integrarse exitosamente en cansats para tareas autónomas. No obstante, implican un aumento en la complejidad computacional y consumo de energía a bordo. Un vacío por resolver es cómo implementar estos algoritmos de forma eficiente en hardware embebido muy limitado (microcontroladores o minicomputadoras) sin sacrificar la autonomía del vuelo; este desafío técnico es parte del panorama que motiva proyectos actuales en cansats con visión.

Uso de cámaras infrarrojas térmicas en cansats

Además de las cámaras convencionales, algunos equipos han añadido sensores infrarrojos de baja escala en sus cansats. Un caso de uso ocurrió en la competencia canadiense de cansat de 2021, donde un equipo utilizó un sensor *FLIR Lepton* como parte de su misión secundaria con el fin de poder capturar imágenes térmicas durante el descenso. Los estudiantes argumentaron que la termografía puede detectar fuentes de calor en el terreno.

Esto resulta conveniente para identificar incendios o anomalías térmicas, incluso bajo condiciones de cubierta ligera o humo. La elección del sensor obedece a su tamaño

reducido y en su capacidad de poder operar en un entorno con restricción de masa y costo.

No obstante, los detalles públicos del proyecto no permiten confirmar con seguridad parámetros como la altitud exacta del lanzamiento, la transmisión en tiempo real o el grado de éxito en detección térmica bajo condiciones adversas. Tampoco se dispone de evidencia clara acerca de la calibración radiométrica en vuelo ni del límite de resolución práctica alcanzada por el sensor.

En consecuencia, aunque este experimento demuestra la viabilidad inicial de incorporar visión térmica en cansats, persisten desafíos abiertos: mejorar la resolución del sensor, combinar datos infrarrojos con visibles, calibrar dinámicamente en vuelo y optimizar el procesamiento embebido para satisfacer restricciones de tamaño, energía y tiempo de misión. (Students Win First Ever Canadian cansat Competition with Design Incorporating FLIR Lepton, 2021)

Síntesis de hallazgos y vacíos identificados

Al analizar la literatura existente, se puede apreciar un avance importante en la integración de diversos tipos de módulos para incrementar las funcionalidades al momento de trabajar con imágenes en *cansats*. Se han abordado casos de uso que van desde la recolección de datos ambientales visuales hasta la implementación de inteligencia artificial con el fin de permitir la detección de objetos o incluso la navegación autónoma. Estas investigaciones han demostrado la viabilidad de emplear cámaras para

ampliar las capacidades de un cansat más allá de la sensorística tradicional, convirtiendo a estas pequeñas sondas en plataformas de percepción remota multi-sensorial. No obstante, persisten vacíos y desafíos importantes que la presente propuesta pretende encarar. Por ejemplo, muchos de los proyectos revisados transmiten las imágenes al suelo para su procesamiento, debido a las limitaciones de cómputo a bordo. Esto implica rezagos en la toma de decisiones en tiempo real.

Además, la resolución y calidad de las imágenes obtenidas suelen ser bajas comparadas con satélites mayores, lo cual limita ciertas aplicaciones de análisis detallado. Aspectos como la estabilización de la cámara, la calibración de imágenes georreferenciadas y la fusión de datos de múltiple espectro (visible/infrarrojo) apenas comienzan a explorarse en el contexto de cansats. Asimismo, cada avance (por ejemplo, ejecutar redes neuronales a bordo) conlleva compromisos en consumo energético y diseño electrónico que requieren soluciones innovadoras.

2.2 Bases teóricas que sustentan la investigación

El desarrollo de satélites en general y *cansats* en particular conlleva la intersección entre diferentes disciplinas de estudio. Este proyecto de investigación se centra específicamente en los campos relacionados con el procesamiento digital de imágenes, sistemas embebidos y programación de procesos paralelos. A continuación, se presentarán las generalidades de las tres áreas antes mencionadas para brindar un panorama sobre el estado del arte.

Procesamiento digital de imágenes

Antes de abordar el procesamiento de imágenes, se requiere una definición formal del concepto de imagen. Las imágenes pueden definirse de diferentes maneras según el contexto. Sin embargo, para este documento se tomará la definición de "imagen digital" propuesta por los autores Rafael C. Gonzalez y Richard E. Woods en su obra "Digital Image Processing Fourth Edition":

> *"An image may be defined as a two-dimensional function, f(x,y), where x and y are spatial (plane) coordinates, and the amplitude of at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point. When x, y, and the intensity values of f are all finite, discrete quantities, we call the image a digital image"*. [Una imagen puede definirse como una función bidimensional, f(x, y), donde x e y son coordenadas espaciales (del plano), y la amplitud de f en cualquier par de coordenadas (x, y) se llama intensidad o nivel de gris de la imagen en ese punto. Cuando x, y y los valores de intensidad de f son todos cantidades finitas y discretas, llamamos a la imagen una imagen digital.] (Gonzalez & Woods, 2018, p. 18)

En síntesis, una imagen digital es una matriz de coordenadas cuyos elementos poseen un valor de tipo entero que representa el nivel de gris de la imagen –entendiendo gris como un rango de colores entre el blanco y el negro–. Esta explicación está restringida exclusivamente a las imágenes en escala de grises.

El procesamiento digital de imágenes es un campo cuyos límites no están explícitamente definidos, lo cual genera confusión al momento de identificar dónde comienzan y dónde terminan otras áreas relacionadas, como la visión por computadora y el análisis de imágenes, también llamado comprensión de imágenes (Gonzalez & Woods, 2018).

Los autores del libro antes referenciado mencionan una distinción común –a su juicio "limitante" y "artificial"– la cual define el procesamiento de imágenes como una "disciplina en la que tanto la entrada como la salida de un proceso son imágenes". Con el fin de exponer las barreras que artificialmente impone esta definición, los autores utilizan como ejemplo la tarea de calcular la intensidad promedio de una imagen, operación que toma una imagen como entrada y da como resultado un único número.

Otra definición de procesamiento de imágenes, propuesta por el autor Scott Umbaugh en su obra *"Digital Image Processing and Analysis: Digital Image Enhancement, Restoration and Compression*" es la siguiente:

> *Digital image processing can be defined as the acquisition and processing of visual information by computer.* [El procesamiento digital de imágenes puede definirse como la adquisición y el procesamiento de información visual mediante una computadora](Umbaugh, 2023, p. 20).

Esta definición limita en menor medida los alcances del procesamiento de imágenes comparada con la definición cubierta –y criticada– por los primeros autores. Sin embargo, puede resultar ambiguo trabajar con esta definición, debido a que no se

establecen límites claros con respecto a los otros campos que también operan con imágenes digitales.

Dada la dificultad de establecer definiciones precisas para ubicar los límites entre los diferentes campos sin ser extremadamente rígidos, como para excluir tareas propias del procesamiento de imágenes, los autores Gonzalez y Woods ofrecen el siguiente paradigma para una mejor clasificación.

Un paradigma conceptual conveniente consiste en clasificar los procesos en un continuo de tres niveles: bajo, medio y alto. Siendo los procesos de bajo nivel aquellos que toman una imagen como entrada, realizan ciertas operaciones primitivas como reducir el ruido o incrementar el contraste y devuelven una imagen como resultado. Los procesos de nivel medio toman una imagen como entrada y arrojan como resultado atributos como bordes, contornos y la identidad individual de los objetos.

Finalmente, los procesos de nivel alto son aquellos que a partir de una imagen intentan interpretar o “dar sentido” a los objetos reconocidos dentro de la imagen y en el extremo más alto de este continuo, lograr emular funciones cognitivas típicas de los humanos (Gonzalez & Woods, 2018).

Dentro del procesamiento digital de imágenes, se distinguen dos grandes áreas delimitadas según el “receptor final” de la información visual. Estas áreas son: aplicaciones de visión por computadora y aplicaciones de visión humana.

Las aplicaciones de visión por computadora son aquellas cuyas salidas son orientadas a computadoras, mientras que las aplicaciones de visión humana producen resultados destinados para el consumo humano (Umbaugh, 2023).

Existe una amplia gama de campos que utilizan el procesamiento de imágenes para mejorar el contraste, codificar la intensidad de los colores, destacar zonas, ayudar a la interpretación de las radiografías, entre otros fines. Todos estos objetivos pertenecen a aplicaciones destinadas a la interpretación humana. En cambio, las tareas computacionales como la identificación de caracteres, procesamiento de huellas digitales, procesado automatizado de imágenes satelitales pertenecen a la resolución de problemas relacionados con la percepción por máquina, es decir, extraer información en un formato adecuado para el procesamiento computacional (Gonzalez & Woods, 2018).

En ambas obras se reconoce la distinción según el receptor o interpretador final del procesado de imágenes. Sin embargo, *Umbaugh* usa las categorías de “aplicaciones de visión por computadora” y “aplicaciones de visión humana”. Por su parte, Gonzalez y Woods optan por describir dos grandes áreas de aplicaciones del procesamiento digital de imágenes, sin atribuirles un nombre fijo. Para evitar confusiones –y la ambigüedad de “visión por computadora” que antes se diferenció como otro campo–, en este documento utilizaremos los términos: aplicaciones orientadas a la interpretación humana de imágenes y aplicaciones orientadas al procesamiento computacional de imágenes.

La compresión de imágenes se encarga de recopilar técnicas cuyo fin es reducir el almacenamiento requerido para guardar una imagen o el ancho de banda necesario para poder transmitirla. A pesar de los avances significativos en tecnologías de almacenamiento digital, la transmisión de imágenes no se ha desarrollado al mismo ritmo. Este escenario otorga una gran relevancia a la compresión de imágenes, particularmente en el internet, red donde se transmite una gran cantidad de material visual (Gonzalez & Woods, 2018).

Las aplicaciones multimedia y la distribución de contenido visual han contribuido al crecimiento del campo de la compresión de imágenes. Pese a las mejoras en los sistemas de almacenamiento y en el ancho de banda, el volumen de generación de datos supera sus capacidades. Esta diferencia repercute significativamente en tecnologías cada vez más demandadas como la televisión de alta definición, el video en streaming, la telemedicina y el aprendizaje remoto. En estos escenarios, la compresión de imágenes asume un papel crítico en los entornos que requieren explotar al máximo las virtudes de las tecnologías actuales de almacenamiento y transmisión de información visual.

La compresión de imágenes se logra mediante la identificación y eliminación de redundancias en los datos. Este proceso se aprovecha de las limitaciones del sistema visual humano para eliminar aquellos datos que el ojo humano no puede percibir. Dependiendo del propósito de la imagen, es posible reducir datos considerados innecesarios. Por ejemplo, es aceptable una mayor pérdida de detalle en una recopilación de imágenes para visualización rápida, que en un sistema de diagnóstico

médico. El resultado final de una compresión de imagen se denomina “imagen comprimida”. La razón de compresión se obtiene al comparar el tamaño del archivo original con el del archivo comprimido (Umbaugh, 2023).

Programación paralela

Durante un largo periodo de tiempo, la programación siguió un paradigma estrictamente secuencial, es decir, su ejecución se basaba en seguir una serie de pasos ordenados de principio a fin. Este enfoque de programación es heredado de las limitaciones arquitectónicas de los primeros ordenadores, donde una única unidad lógica-aritmética (*ALU*) ejecutaba instrucciones una tras otra.

En este contexto, la frecuencia de reloj es la unidad de medida del rendimiento de una computadora, la cual se expresa en hercios y refleja la cantidad de instrucciones ejecutables por segundo. No obstante, el surgimiento de nuevos procesadores como el Intel 80386 y 80486, trajo consigo los conceptos de multitarea, programación con interrupciones y la programación segmentada mediante pipeline. Esta evolución en la arquitectura de los procesadores permitió que múltiples instrucciones se ejecutaran de manera simultánea.

En la actualidad, hasta los procesadores más limitados en capacidad de procesamiento incorporan arquitecturas multinúcleo, lo que permite el surgimiento de oportunidades de mejora en el diseño de programas más eficientes mediante el paralelismo. Si bien es cierto que la programación paralela trae muchos beneficios, muchos desarrolladores no

explotan su potencial debido a la dificultad técnica en su implementación y la relativa novedad de esta práctica. En campos como el procesamiento científico o el manejo de grandes volúmenes de datos, la programación secuencial resulta ineficiente, dado que no aprovecha todos los recursos disponibles del procesador. Esto provoca tiempos de ejecución innecesariamente más extensos y menor escalabilidad en las soluciones de software. La programación paralela consiste en dividir una gran tarea en subtareas similares que se ejecutan simultáneamente, aprovechando en mayor medida los recursos disponibles y habilitando la posibilidad de resolver tareas computacionalmente intensas que antes resultaban inviables. (Nelli, 2023)

El vertiginoso crecimiento del volumen de datos generados ha otorgado a la programación concurrente un mayor grado de importancia. Las arquitecturas de programas que implementan la concurrencia, es decir, la capacidad de gestionar múltiples tareas parcial o totalmente independientes, obtienen mejores resultados en términos de tiempos de ejecución y eficiencia global del sistema.

Esta característica permite desarrollar flujos de tareas capaces de operar independientemente de sus diferentes estados. Estos programas son más complejos a nivel técnico y, por ende, más difíciles de desarrollar. Sin embargo, el crecimiento de la información digital, la cual se duplica cada dos años según estimaciones de la International Data Corporation, exige aprovechar al máximo las capacidades computacionales para satisfacer la demanda de procesamiento eficiente.

Resulta esencial entender la diferencia entre paralelismo y concurrencia. A pesar de que ambos son conceptos que se refieren a los flujos con ejecución simultánea de tareas, la concurrencia se refiere a la ejecución de múltiples hilos y procesos con recursos compartidos. Esto crea la necesidad de incorporar mecanismos de sincronización para evitar condiciones de carrera. Una condición de carrera se produce cuando un proceso altera un recurso o valor compartido con otro, lo cual puede provocar fallos en la ejecución del programa.

Por otro lado, el paralelismo consiste en ejecutar múltiples tareas en núcleos diferentes. Existen diversos tipos de tareas, algunas son estrictamente secuenciales mientras que otras son especialmente adecuadas para ser ejecutadas de manera paralela. Las tareas pertenecientes al procesamiento de imágenes, el renderizado gráfico o la minería de datos suelen ser candidatas idóneas para aprovechar arquitecturas multinúcleo debido a que son altamente divisibles sin necesidad de comunicarse entre ellas. (Nguyen, 2018)

Sistemas Embebidos

Las computadoras de propósito general permiten al usuario ejecutar diferentes tipos de tareas según su necesidad. Sin embargo, para sistemas completos que requieren de procesos específicos, los cuales deben ser ejecutados de manera eficiente, confiable y autónoma, resulta conveniente utilizar sistemas embebidos. Algunos ejemplos de estos son los microondas, reproductores MP3 y relojes despertadores. Pese a su presencia poco visible, estos equipos también cuentan con procesadores y software que definen

su funcionamiento, lo que evidencia el alcance e importancia de la computación embebida en la vida cotidiana.

En muchas ocasiones, los sistemas embebidos funcionan como subsistemas dentro de una estructura más compleja. Un claro ejemplo se encuentra en la industria automotriz, donde múltiples sistemas embebidos controlan desde los frenos antibloqueo hasta la gestión de emisiones y la interfaz del tablero digital. Incluso periféricos de uso común, como teclados, ratones y módems, constituyen sistemas embebidos con microprocesadores internos que ejecutan software dedicado.

Aunque sería posible implementar sus funciones con circuitos integrados de propósito fijo, el uso de microprocesadores y software en sistemas embebidos ofrece una mayor flexibilidad, reducción de costos y menor consumo energético, características fundamentales en el diseño moderno de dispositivos electrónicos compactos y funcionales. (Barr & Massa, 2009)

2.3. Variables de la investigación

2.3.1. Definición conceptual de la variable

La investigación identifica tres variables principales que estructuran el análisis del proyecto:

Variable independiente: arquitectura de procesamiento paralelo

Desde una perspectiva teórica, la arquitectura de procesamiento paralelo se define como un modelo computacional que distribuye tareas intensivas de cómputo entre múltiples núcleos de procesamiento disponibles en un CPU, permitiendo su ejecución simultánea.

Esta variable se fundamenta en los principios del paralelismo descrito por Nelli (2023), quien señala que la programación paralela consiste en dividir una gran tarea en subtareas similares que se ejecutan simultáneamente. En el contexto específico de este proyecto, la arquitectura paralela implementa el esquema Maestro-Esclavo, donde un núcleo orquesta la distribución equilibrada de cargas entre los núcleos restantes.

Variable dependiente 1: Tiempo de procesamiento de imágenes

El tiempo de procesamiento de imágenes se define conceptualmente como la duración total requerida para completar las operaciones secuenciales de adquisición, recorte, compresión y generación de metadatos sobre una imagen digital. Según Gonzalez y Woods (2018), las imágenes digitales son matrices de coordenadas cuyos elementos representan niveles de intensidad. El tiempo de procesamiento, medido en milisegundos, representa el período desde que una imagen es leída del almacenamiento hasta que se genera el archivo final comprimido con sus metadatos asociados.

Variable dependiente 2: Eficiencia de compresión de imágenes

La eficiencia de compresión se define conceptualmente como la capacidad de reducir el tamaño de archivo de una imagen manteniendo una calidad visual aceptable. Umbaugh (2023) define la compresión digital de imágenes como el proceso de identificación y eliminación de redundancias en datos visuales, aprovechando las limitaciones del sistema visual humano. La razón de compresión se obtiene comparando el tamaño del archivo original con el del archivo comprimido, expresada típicamente como un porcentaje.

Variable dependiente 3: Utilización de Recursos del CPU

La utilización de recursos del *CPU* se define conceptualmente como el porcentaje de capacidad de procesamiento efectivamente empleado durante la ejecución del programa. Esta variable refleja la eficiencia general del sistema en cuanto a aprovechamiento de la capacidad disponible del *Raspberry Pi Zero 2 W*, medida en términos de porcentaje de uso de *CPU* y distribución de carga entre los cuatro núcleos disponibles.

2.3.2. Definición operacional de la variable

Variable Independiente: arquitectura de procesamiento paralelo

Indicadores de operación:

- Número de hilos de ejecución: 1 hilo (modo serial) vs. 4 hilos (modo paralelo)
- Distribución de particiones: División de la imagen en mosaicos parametrizables
- Algoritmo de balanceo de carga: Esquema Maestro-Esclavo con cola de tareas

Escala de medición: Nominal (presencia/ausencia de paralelismo)

Técnica de medición: Parámetro de configuración del programa

Operacionalización en el instrumento: El programa Python desarrollado incluye un módulo de orquestación central que controla el modo de ejecución. En modo paralelo,

las imágenes se dividen en 16 particiones (4x4) y se distribuyen mediante una cola de tareas) entre 4 trabajadores (threads). En modo serial, las mismas operaciones se ejecutan secuencialmente en un único hilo.

Variable dependiente 1: tiempo de procesamiento de imágenes

Indicadores de medición:

- Tiempo total de procesamiento: suma de tiempos de recorte, compresión y generación de metadatos
- Tiempo promedio por imagen
- Reducción porcentual de tiempo (paralelo vs. serial)

Escala de medición: Razón (milisegundos)

Técnica de medición: Captura programática de timestamps usando la biblioteca time de *Python*

Fórmula de cálculo:

$$T_{\text{total}} = T_{\text{recorte}} + T_{\text{compresión}} + T_{\text{metadatos}}$$

Fórmula de reducción:

$$\text{Reducción } (\%) = \frac{T_{\text{serial}} - T_{\text{paralelo}}}{T_{\text{serial}}} \times 100$$

Operacionalización en el instrumento: El módulo de captura de métricas registra el timestamp al inicio y final de cada operación. Los datos se almacenan automáticamente en un archivo CSV con precisión de milisegundos. Meta esperada: reducción mínima del 30%.

Variable dependiente 2: eficiencia de compresión de imágenes

Indicadores de medición:

- Razón de compresión: porcentaje de reducción de tamaño
- Tiempo de compresión por imagen
- Relación calidad-compresión (escala subjetiva 1-5)

Escala de medición: Razón (porcentaje)

Técnica de medición: Comparación de tamaños de archivo original vs. comprimido; inspección visual de artefactos.

Fórmula de cálculo:

$$\text{Razón de compresión } (\%) = \frac{\text{Tamaño original} - \text{Tamaño comprimido}}{\text{Tamaño original}} \times 100$$

El módulo de evaluación calcula la razón de compresión comparando tamaños de archivo. Meta esperada: mínimo 50% de reducción manteniendo calidad visual.

Variable Dependiente 3: Utilización de Recursos del CPU

Indicadores de medición:

- Porcentaje promedio de uso de CPU durante ejecución
- Distribución de carga entre los 4 núcleos
- Uso de memoria RAM

Escala de medición: Razón (porcentaje)

Técnica de medición: Monitoreo mediante la biblioteca psutil de Python durante la ejecución

$$\text{Promedio CPU (\%)} = \frac{\sum_{i=1}^{n} \text{CPU}_i}{n}$$

Donde (n) es el número de muestras tomadas.

Operacionalización en el instrumento: El programa captura la utilización de CPU cada 100 milisegundos durante la ejecución, registrando el uso por núcleo y total. Los datos se consolidan en el reporte final en formato CSV. Meta esperada: distribución equilibrada entre núcleos en modo paralelo (idealmente ≥75% de utilización por núcleo).

2.3.3 Mapa de Variable

Variable	Tipo	Dimensión	Indicador	Escala	Instrumento	Objetivo
Arquitectura de procesamiento paralelo	Independiente	Modo de ejecución	Número de hilos (1 vs. 4)	Nominal	Parámetro del programa	Medir impacto del paralelismo
Arquitectura de procesamiento paralelo	Independiente	Distribución de carga	División en mosaicos	Ordinal	Cola de tareas	Evaluar balanceo
Tiempo de procesamiento	Dependiente	Tiempo total	Duración completa (ms)	Razón	Captura de timestamps	Cumplir Objetivo Esp. 2
Tiempo de procesamiento	Dependiente	Reducción porcentual	% mejora (paralelo vs. Serial)	Razón	Fórmula de cálculo	Validar Hipótesis 1
Eficiencia de compresión	Dependiente	Razón de compresión	% reducción de tamaño	Razón	Comparación de archivos	Cumplir Objetivo Esp. 3

Eficiencia de compresión	Dependiente	Calidad de imagen	Escala subjetiva (1-5)	Ordinal	Inspección visual	Validar Hipótesis 2
Utilización de CPU	Dependiente	Uso total	% promedio de CPU	Razón	Psutil library	Medir eficiencia de recursos
Utilización de CPU	Dependiente	Distribución por núcleo	% por cada núcleo	Razón	Monitoreo en tiempo real	Validar Objetivo Esp. 4

2.4 Glosario

Algoritmo de compresión: Procedimiento matemático que reduce el tamaño de un archivo identificando y eliminando redundancias en los datos.

Ancho de banda: Capacidad máxima de transferencia de datos en un canal de comunicación, medida en kilobits por segundo (Kbps).

Arquitectura multinúcleo: Diseño de procesador que integra múltiples núcleos de procesamiento independientes capaces de ejecutar instrucciones simultáneamente.

ARM (*Advanced RISC Machine):* Arquitectura de procesador de conjunto reducido de instrucciones, ampliamente utilizada en dispositivos embebidos y computadoras de bajo consumo.

Balanceo de carga: Distribución equilibrada de tareas entre múltiples procesos o hilos para optimizar el rendimiento global del sistema.

cansat: Satélite del tamaño aproximado de una lata de aluminio estándar, utilizado en aplicaciones educativas y científicas suborbitales.

Compresión con pérdida: Técnica de compresión que elimina datos considerados innecesarios para el sistema visual humano, resultando en cierta degradación de la calidad, pero mayor reducción de tamaño.

Compresión sin pérdida: Técnica de compresión que permite recuperar exactamente los datos originales sin ninguna pérdida de información.

CPU (Unidad Central de Procesamiento): Componente principal de una computadora que ejecuta instrucciones del programa.

Esquema Maestro-Esclavo: Modelo de procesamiento paralelo en el que un proceso maestro coordina la distribución de tareas entre procesos esclavos.

Hilo de ejecución (thread): Unidad básica de procesamiento que puede ser ejecutada independientemente dentro de un proceso.

Imagen digital: Matriz bidimensional de píxeles donde cada elemento posee un valor que representa la intensidad o color en esa posición.

JPEG (Joint Photographic Experts Group): Formato estándar de compresión de imágenes con pérdida ampliamente utilizado en fotografía digital.

Latencia: Tiempo de retraso entre la solicitud de una operación y la entrega del resultado.

Metadatos: Información descriptiva sobre un archivo, como fecha de creación, resolución, algoritmo de compresión utilizado, entre otros.

Mosaico de imagen: División de una imagen en fragmentos o particiones de menor tamaño para procesamiento paralelo.

Procesamiento paralelo: Ejecución simultánea de múltiples tareas utilizando recursos computacionales independientes.

Procesamiento serial: Ejecución secuencial de tareas, una después de la otra, utilizando un único recurso de procesamiento.

***Raspberry Pi Zero 2 W*:** Microcomputadora de bajo costo equipada con procesador ARM Cortex-A53 de 4 núcleos, 1 GHz de frecuencia y 512 MB de RAM.

Rendimiento: Medida de la velocidad y eficiencia de un sistema para completar operaciones, típicamente expresada en operaciones por segundo o tiempo de ejecución.

Sensado remoto: Tecnología de adquisición de información sobre objetos o áreas sin contacto físico directo, típicamente desde satélites o plataformas aéreas.

Sistemas embebidos: Sistemas computacionales especializados diseñados para realizar funciones específicas dentro de dispositivos más grandes, con restricciones de tamaño, energía y costo.

Telemetría: Transmisión de datos científicos o de medición recolectados remotamente desde dispositivos a una estación receptora.

Tiempo de ejecución: Duración total requerida para completar un programa o proceso desde su inicio hasta su finalización.

Visión por computadora: Campo de la inteligencia artificial que desarrolla métodos y algoritmos para que las computadoras interpreten y analicen información visual.

CAPÍTULO 3.0
MARCO METODOLÓGICO

3.1 Tipo y diseño de la investigación

La investigación se enmarca en un enfoque cuantitativo de carácter experimental. Dicha elección corresponde a la necesidad de medir, analizar y comparar el rendimiento de un sistema de procesamiento de imágenes bajo diferentes condiciones de ejecución. La condición en concreto con la cual se busca experimentar es el paralelismo, con el fin de establecer mediante evidencia empírica si el procesamiento en paralelo incrementa o no, la eficiencia en términos de tiempo de ejecución las tareas comparado con un enfoque en serie.

El diseño experimental fue realizado mediante el control de una variable independiente, la cual fue el modo de ejecución del sistema, mediante esta manipulación se originaron dos escenarios posibles: ejecución en paralelo y ejecución serial. En el primer escenario, la división, compresión y conversión a audio por cada uno de los fragmentos de imagen se distribuyó entre los cuatro hilos del procesador.

En el siguiente escenario, el sistema fue establecido en modo secuencial para que realice todas las tareas en un único hilo de ejecución. Las variables dependientes consistieron en las métricas de rendimiento registradas por el programa. El tiempo de procesamiento por tapa, el uso promedio de *CPU* y el tiempo final que tomó procesar la imagen a lo largo de todas las fases del programa conforman el conjunto de las variables dependientes.

Para garantizar la validez interna del experimento se empleó un mismo conjunto de imágenes en ambos escenarios, compuesto por un lote de cinco imágenes seleccionadas con características homogéneas en cuanto a resolución y formato. Cada configuración fue ejecutada cinco veces de manera independiente, generando un total de diez corridas experimentales. Este esquema permitió obtener un conjunto de medidas repetidas donde se mantuvo las condiciones constantes, lo cual reduce la influencia de factores externos y posibilita la aplicación de un análisis comparativo robusto.

El diseño corresponde a un experimento de medidas repetidas con un solo factor, donde el factor corresponde al tipo de ejecución (paralela o secuencial). Este enfoque facilita la identificación de diferencias significativas en las métricas de rendimiento atribuibles únicamente a la condición de procesamiento, descartando variaciones asociadas al conjunto de datos o a fluctuaciones aisladas del sistema operativo. En suma, el diseño metodológico aplicado busca no solo describir el comportamiento del sistema, sino también establecer relaciones de causalidad entre el uso de paralelismo y la eficiencia lograda en el procesamiento de imágenes a bordo de un cansat simulado.

3.2 Población y muestra

El en el marco de la presente investigación, se definió la población como el conjunto de todas las posibles ejecuciones en sus distintos modos y configuraciones, tales como: porcentaje de calidad de la imagen, modo de ejecución (paralela o serial), cantidad de filas y columnas en las cuales se divide cada imagen, entre otras posibles configuraciones, las cuales son ejecutadas en el mismo entorno computacional, la

Raspberry Pi Zero 2 W. Esta población abarca un amplio nivel conceptual, ya que incluye toda combinación de imágenes de entrada, ajustes de calidad, resoluciones y modos de ejecución que el sistema permite. Sin embargo, este estudio está delimitado a un subconjunto controlado de ejecuciones para garantizar la posibilidad de ser replicado y facilitar el análisis comparativo.

La muestra seleccionada está compuesta de un lote de cinco imágenes capturadas en la plataforma de *Google Earth* a una altura de 1000 metros con la intención de emular el tipo de fotografía que puede tomar un cansat sobre la superficie. Las imágenes fueron tomadas en formato JPG con la calidad ajustada al máximo y una resolución homogénea de 640x480 pixeles. Esta elección corresponde al formato y calidad con la cual un módulo de cámara podría capturar las imágenes durante el vuelo del *cansat.*

El lote de imágenes fue procesado múltiples veces en dos condiciones experimentales: ejecución en modo paralelo, aprovechando los 4 hilos con los que cuenta la *Raspberry Pi Zero 2 W* y ejecución en modo secuencial, que fuerza al programa a realizar todas las tareas sobre un solo hilo del procesador. Ambas configuraciones fueron ejecutadas 5 veces, lo cual generó un total de diez ejecuciones experimentales como muestra final sobre la cual se realizó el análisis.

La elección de esta muestra obedece a criterios de factibilidad y control experimental. En primer lugar, el equipo cuenta con recursos limitados, lo cual impide procesar una cantidad masiva de imágenes en un lapso razonable, sobre todo en el modo serial; por

otro, se busca reducir variabilidades externas mediante la elección de una cantidad fija de imágenes en un solo lote.
Si bien es cierto que el tamaño de la muestra es reducido en términos absolutos, su diseño de medidas repetidas permite extraer conclusiones válidas sobre las diferencias de desempeño entre los dos modos de ejecución. En consecuencia, la muestra se considera representativa de las condiciones bajo estudio y suficiente para sustentar el análisis comparativo de eficiencia en el procesamiento de imágenes.

3.2.1 Cálculo del muestreo

La investigación se planteó desde un enfoque experimental con medidas repetidas, el tamaño y selección de la muestra se debe a las limitaciones técnicas que presenta el entorno en el cual se realizaron las pruebas. No obstante, la cantidad de imágenes y repeticiones permiten observar tendencias consistentes, manteniendo la factibilidad del experimento.

El lote de cinco imágenes como unidad experimental, sobre el cual se ensayó cinco veces cada modo de ejecución (serial y paralelo) generó diez lotes de ejecuciones. Estadísticamente es una cantidad suficiente para identificar diferencias notables en las métricas de rendimiento de ambos modos, aun cuando no se persiga un análisis inferencial con altos niveles de significancia.

El procedimiento de muestreo seguido puede clasificarse como no probabilístico por conveniencia, en tanto la selección del conjunto de imágenes y el número de repeticiones obedecieron a criterios prácticos y técnicos más que a un muestreo aleatorio de una

población amplia. No obstante, la estrategia adoptada asegura un control suficiente sobre las variables externas, reduce la variabilidad experimental y permite alcanzar el objetivo central: comparar de manera objetiva la eficiencia del sistema bajo condiciones de paralelismo y ejecución secuencial.

3.3 Hipótesis

La hipótesis central que orienta esta investigación se formula de la siguiente manera: Hipótesis de investigación (H1):

Procesar lotes de imágenes para convertirlas a un formato adecuado para la transmisión mediante radiofrecuencia requiere de una serie de tareas las cuales se pueden realizar de manera paralela. Por lo tanto, procesar paralelamente varias imágenes reduce significativamente el tiempo que toma la conversión del lote completo.

De esta hipótesis principal se derivan los siguientes supuestos específicos:

I. El tiempo total de procesamiento por lote de imágenes será menor mediante una ejecución paralela de las tareas comparado con la ejecución serial de estas.
II. El paralelismo reducirá los tiempos de espera que se presentan en las ejecuciones seriales debido a la cola que genera la división de imágenes.
III. El sistema será reproducible y consistente en la duración que tome procesar el mismo lote de imágenes, evidenciando diferencias mínimas en las métricas y estabilidad en los tiempos de ejecución bajo configuraciones idénticas.
IV. No se presentarán diferencias en la calidad de las imágenes independientemente de cuál sea el modo de procesamiento (serial o paralelo), ya que la compresión se realiza utilizando el mismo modo con los mismos parámetros.

Hipótesis nula (H0):

El procesamiento en paralelo de las imágenes no reduce los tiempos de procesamiento de los lotes de imágenes en la Raspberry Pi Zero 2 W.

El contraste de ambas hipótesis se realizará comparando los resultados obtenidos de las ejecuciones en el modo paralelo versus serial, para identificar una mejoría significativa en el tiempo que toma procesar el mismo lote de imágenes.

3.4 Descripción del instrumento

El instrumento empleado en la investigación fue un programa desarrollado en *Python* y diseñado para ejecutarse específicamente en un entorno Raspbian el cual corre sobre una minicomputadora *Raspberry Pi Zero 2 W.* El programa cuenta con un archivo de tipo *YAML* donde se establecen los parámetros, los cuales pueden ser actualizados según la necesidad.

Parámetros como la cantidad de filas y columnas en las cuales se desea dividir cada imagen, el porcentaje de calidad de la imagen que se necesite retener durante la compresión o si se desea generar métricas de rendimiento de cada ejecución se pueden controlar desde el archivo de configuraciones.

El programa fue diseñado desde una lógica modular con el fin de favorecer la legibilidad y facilidad de mantenimiento del programa, las responsabilidades fueron separadas en los distintos módulos y cada uno de ellos se invoca cuando el módulo central lo requiere. Los componentes principales incluyen el módulo de orquestación principal, responsable de coordinar toda la secuencia, garantizar la trazabilidad de cada ejecución y consolidar los resultados en formatos estandarizados *(JSON, CSV).* El módulo central se apoya con otros que proporcionan utilidades como: convertir las imágenes en pistas de audio,

instalar dependencias requeridas u obtener la cantidad de hilos de procesamiento disponibles con los que cuenta el sistema, entre otras funcionalidades para lograr fin del programa.

3.5 Procedimiento de la investigación

El procedimiento de la investigación se estructuró en una secuencia de etapas claramente definidas que aseguraron la coherencia del diseño experimental y la validez de los resultados obtenidos. Dichas etapas se describen a continuación:

1. Preparación del entorno

Esta fase consistió en instalar el sistema operativo, ya que los *Raspberry Pi* por defecto no lo incluyen. Se instalaron las dependencias necesarias de *Python* y se levantó un entorno virtual para garantizar el aislamiento de las versiones de las bibliotecas utilizadas y evitar incompatibilidades no deseadas.

2. Desarrollo del prototipo

El foco de la investigación radica en demostrar que la ejecución de tareas de procesamiento de imágenes mejora sustancialmente en comparación con una ejecución en serie. Para demostrarlo se desarrolló un programa que ejecuta una serie de tareas que pueden ejecutarse en simultáneo o serialmente, el modo se controla mediante un parámetro, de manera que se garantiza la consistencia entre los pasos tomados en cada modo de ejecución.

3. Selección de la muestra: Se capturó una serie de cinco imágenes desde Google Earth con la finalidad de emular el tipo de imágenes que se procesarán en un caso de uso real. Estas cinco imágenes se tomaron con el mismo formato y resolución para reducir factores externos que pudiesen influir en las pruebas de ambos modos de ejecución

Ilustración 3.Captura satelital obtenida desde Google Earth (2025).

4. Ejecución de la prueba

Se ejecutó cinco veces por cada modo de ejecución el programa, utilizando el mismo lote de imágenes. Durante cada ejecución, el programa operó automáticamente, leyendo las imágenes, particionándolas en mosaicos, comprimiendo cada fragmento y generando los metadatos asociados.

5. Registro de resultados

Los resultados se generaron de manera programática, es decir, el propio programa se encarga de registrar los tiempos que toma realizar cada fase, el consumo de *CPU*, entre otras métricas. Lo cual al final de la prueba generó un conjunto de resultados por cada imagen procesada.

CAPÍTULO 4.0
ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

4.1 Análisis e interpretación de los resultados

El programa incorpora una funcionalidad que captura las métricas de consumo de recursos y de tiempo en milisegundos que toma ejecutar cada una de las partes del programa y las registra en un archivo de tipo *CSV.* Esta función facilita la recopilación y comparación de estadísticas entre ambos modos, serial o paralelo.

A continuación, ofreceré el análisis e interpretación de los datos obtenidos después de realizar diez ejecuciones del programa, cinco en modo paralelo y otras cinco en modo serial.

4.1.1 Panorama del registro

El archivo con los registros de rendimiento consta de diez ejecuciones del programa sobre un lote de cinco imágenes, la primera mitad corresponde a las cinco ejecuciones en modo paralelo y la segunda mitad en modo serial, dando como resultado 200 sub-imágenes procesadas.

Los resultados evidencian que las ejecuciones del programa en modo paralelo presentan un tiempo de ejecución entre 113 y 127 segundos, por otro lado, las ejecuciones en modo serial se extienden a un rango de 455 a 458 segundos, aproximadamente cuatro veces más. Esta diferencia entre los tiempos de ejecución expone el impacto directo del paralelismo para recortar el tiempo total que toma procesar cada imagen. En promedio, cada imagen procesada en modo paralelo toma 22.2 segundos frente a los 91.2 segundos en el modo secuencial. Esta diferencia confirma un mejor aprovechamiento de los núcleos de la *CPU* en el modo paralelo.

Además, la métrica total_image_ms, que representa el tiempo total requerido para procesar cada sub-imagen hasta su conversión final en una pista de audio .WAV, coincide casi exactamente con la métrica total_run_ms al sumar los valores correspondientes a todas las sub-imágenes de una ejecución. Esta correlación demuestra que la canalización de procesamiento permaneció activa durante todo el flujo de trabajo, sin periodos significativos de espera o inactividad entre tareas.

4.1.2 Diferencias entre los regímenes de ejecución

Los resultados recopilan claramente diferencias entre ambos regímenes de ejecución, en el modo paralelo, el sistema presenta valores de “queu_share” cercanos al 0.04%, “wav_share” superiores al 91% y una tasa de procesamiento aproximada de 0.17 sub-imagen por segundo. Estos indicadores comprueban que las cuatro divisiones de cada imagen se procesan de manera paralela.

En su contraparte, las ejecuciones en modo serial exhiben un queue_share promedio del 37.5 %, un wav_share reducido a 24.9 % y una tasa de procesamiento de solo 0.04 divisiones por segundo. Bajo este contexto, cada fragmento de imagen tomo 22.9 segundos en ser procesados, pero debido a su ejecución en serie, se genera una cola que incrementa el tiempo de espera de cada uno de los fragmentos en un patrón secuencial (0 / 22.9 / 45.9 / 68.9 s). Esta diferencia de 22 segundos en el modo paralelo versus los 91 segundos en el modo secuencial es producto de la ejecución en serie, que multiplica por cuatro el tiempo que toma procesar cada imagen en modo secuencial.

Además, se observa que en el modo paralelo las métricas *stage_wav_ms* y *total_image_ms* es prácticamente perfecta (≈1.0), esto indica que la fase del programa que toma la mayor parte del tiempo de ejecución es la conversión de imagen a pista de audio .WAV, mientras que en el modo serial, son los tiempos de cola los que adquieren un peso predominante en el tiempo y se convierten en el mayor factor limitante del programa.

4.2 Prueba de hipótesis

La hipótesis de la investigación planteaba que el tiempo de procesamiento de los lotes de imágenes serían reducidos mediante la implementación del paralelismo. esta afirmación se demostró en los resultados finales de la prueba que indican que las ejecuciones seriales toman en promedio 456,214 ms, mientras que las ejecuciones en paralelo toma 119,189 ms. Lo que refleja una reducción del 74% en la duración total del programa.

El segundo supuesto indicaba que la implementación del paralelismo reduciría el tiempo que agregan las colas internas de las ejecuciones seriales. Estas colas se generan debido a la incapacidad del modo secuencial de procesar todos las sub-imágenes al mismo tiempo. Los registros experimentales corroboran este punto: en el modo paralelo, el parámetro *queue_share* se mantuvo en valores cercanos al **0.04 %**, con tiempos de espera promedio de tan solo **10.37 ms**, mientras que en el modo serial se elevó al **37.5 %**, alcanzando colas de hasta **68.9 s** para los últimos fragmentos. Estos resultados evidencian la efectividad del paralelismo, al poder manejar las mismas tareas con todos

los fragmentos de imagen en simultáneo, la cola de procesamiento se reduce de 68 segundos a solo 10.37 ms, recortando significativamente los tiempos de ejecución.

El tercer planteamiento sugería que el sistema debía ser estable y consistente tanto en el rendimiento como en los tiempos de ejecución, este punto se demostró mediante los resultados. La desviación estándar de los tiempos de ejecución total de cada lote se mantuvo por debajo del 1% entre las ejecuciones de ambos grupos (serial y paralelo) y las métricas de la fase más compleja a nivel computacional: la conversión de imágenes a pistas de audio mostró una variación inferior a los 0.4 segundos entre sub-imágenes de una misma ejecución. Estos resultados demuestran un alto grado de consistencia en el rendimiento del sistema.

El cuarto supuesto señalaba que no deberían existir diferencias entre la compresión de imágenes en los modos seriales y paralelo, esto se cumplió con el propio diseño del programa, el método utilizado para comprimir las imágenes es el mismo para ambos modos, por ende, no existen posibles diferencias siempre que los parámetros de calidad de la imagen sean los mismos.

CAPÍTULO 5.0
PROPUESTA DE LA INVESTIGACIÓN

5.1 Introducción de la propuesta

La propuesta de esta investigación nace de la necesidad de mejorar el rendimiento de los sistemas de procesamiento de imágenes de los cansats. Estos equipos generalmente suelen incorporar unidades computacionales de bajo consumo energéticos, compactos en tamaño y limitados en recursos. La *Raspberry Pi Zero 2 W* es un buen ejemplo del tipo de microcomputadora. No obstante, a pesar de ser una computadora a una escala reducida, el procesador de esta permite la programación en multinúcleo, esta virtud posee un potencial que tiende a ser desaprovechado en el procesamiento de tareas intensivas.

La propuesta de esta investigación consiste en una arquitectura modular de un sistema que es capaz de procesar lotes de imágenes sobre las cuales se realizan las tareas de segmentación, compresión y transformación a pistas de audio de manera concurrente entre cada segmento de imagen. Este programa aprovecha el paralelismo para mejorar la eficiencia del sistema, recortar tiempos de ejecución y optimizar la utilización de los núcleos del procesador.

Este enfoque permite al cansat poder capturar y enviar una mayor cantidad de imágenes en vuelo, lo cual incrementa la información visual del terreno obtenida durante su misión. Además, el sistema propuesto presenta un alto grado de desacoplamiento lo cual facilita la toma de métricas entre procesos, la mejora continua del sistema y facilita la implementación de nuevas funcionalidades de software.

La propuesta en sí demuestra la viabilidad técnica de implementar técnicas de procesamiento en paralelo en los sistemas de procesamiento de imágenes desplegados en unidades computacionales limitadas, mejorando el rendimiento de los cansats en misiones educativas y experimentales.

5.2 Justificación de la propuesta

La propuesta se justifica por su misión, la cual es aprovechar los recursos computacionales disponibles en microcomputadoras de bajo consumo destinadas a ser parte de misiones educativas y experimentales como en los cansats. La transmisión de imágenes es una tarea que demanda tiempo y ancho de banda, por lo que resulta imprescindible reducir el tamaño de las imágenes para disminuir los tiempos de envío y dependiendo del protocolo de radiofrecuencia, puede ser necesario convertir las imágenes a pistas de audio para poder transmitirlas.

Los resultados experimentales muestran mejoras significativas en los tiempos de procesamiento al ejecutar los procesos en paralelo comparado con el modo secuencial. Los resultados de las pruebas confirman que los diseños de sistemas capaces de aprovechar las arquitecturas de los procesadores multinúcleos mejoran el desempeño de los programas, permitiendo obtener resultados provechosos en microcomputadores de bajo costo. Así, la propuesta se basa en datos empíricos que validan la viabilidad y pertinencia de la investigación.

Desde el plano académico, la propuesta ofrece un caso de estudio práctico que demuestra los resultados de implementar el paralelismo en procesos intensivos en equipos limitados pero asequibles. Esto expande el horizonte de posibilidades a investigadores y estudiantes para experimentar con nuevas aplicaciones implementando procesamiento en paralelo en equipos limitados. A demás democratiza las posibilidades de poder poner en práctica este tipo de técnicas de programación en casos reales de uso, como en los cansats.

En el aspecto práctico, la propuesta ofrece una arquitectura escalable y replicable. Estos atributos del diseño brindan la posibilidad de implementar nuevos módulos y funciones que expandan el alcance del programa. Esto permitiría desarrollar nuevos flujos de procesamiento más ambiciosos que puedan brindar una mayor cantidad de información relevante para los usuarios. A demás, con el flujo actual el sistema es capaz de transmitir una mayor cantidad de imágenes a los observadores, esto mejora la experiencia general y habilita a los investigadores de poder obtener más información del terreno sobre el cual se eleva el cansat.

5.3 Objetivos de la propuesta

Los objetivos de la propuesta giran en torno a los beneficios del diseño y desarrollo de un sistema capaz de explotar las ventajas técnicas de los microprocesadores con arquitecturas multinúcleos para misiones de equipos cansats. Estos objetivos se formulan de la siguiente manera:

Objetivo general:

Diseñar la arquitectura y desarrollar el programa correspondiente a un sistema de procesamiento de imágenes capaz de implementar procesos en paralelo aprovechando los cuatro núcleos de la *Raspberry Pi Zero 2 W* para ser utilizado en misiones de *cansats.*

Objetivos específicos

- Desarrollar un programa que orqueste la ejecución de las tareas de segmentación, compresión y conversión de imágenes a formato WAV sobre lotes de imágenes de manera paralela, reduciendo los tiempos de ejecución comparado con un procesamiento serial.
- Implementar un mecanismo que registre las métricas de tiempos y consumo de recursos con el fin de permitir la trazabilidad y el estudio comparativo del rendimiento del programa.
- Validar la eficiencia del programa en los modos paralelos paralelo y serial ante escenarios homogéneos con entradas iguales.
- Evaluar la interoperabilidad del sistema para trabajar en simultaneo con otros módulos del cansat como la transmisión de imágenes o la medición de datos atmosféricos.
- Diseñar una arquitectura que pueda servir de referencia para futuras investigaciones y desarrollos de sistemas para proyectos de ingeniería aeroespacial educativa.

5.4 Metas por alcanzar

La propuesta se desarrolla con base a unas metas específicas predefinidas con el fin de garantizar logros tangibles y medibles, los cuales guían el desarrollo para obtener resultados que mejoren a los sistemas existentes. Estas metas derivan de los objetivos planteados.

A continuación, se presentan las metas a alcanzar:

Reducción de tiempos de procesamiento: Alcanzar una reducción del 70% en los tiempos de procesamiento de cada imagen, con el objetivo de aumentar la ratio de salida de imágenes con respecto al tiempo en el modo paralelo comparado con el enfoque serial.

Equilibrio entre los núcleos de la *CPU*: Demostrar una distribución eficiente entre los núcleos de la *CPU* con el fin de evitar desbalances que generen sobrecargas y tiempos de ocio entre los hilos.

Generación de metadatos completos: Generar manifiestos que contengan información relevante sobre las métricas de ejecución e información sobre las imágenes generadas, lo cual permita indagar en el rendimiento y en los resultados de cada corrida del programa.

Replicabilidad académica: Desarrollar un prototipo funcional, escalable y bien documentado lo cual permita servir de punto de partida para futuras investigaciones y experimentos educativos.

Escalabilidad del sistema: Sentar las bases de un programa que pueda extenderse para implementar nuevas funcionalidades que enriquezcan los resultados de los

sistemas embebidos de los cansats, derivando en dispositivos capaces de ejecutar misiones más complejas a nivel técnico.

5.5 Beneficios de la propuesta

La propuesta aporta beneficios de carácter técnico, académico y práctico, los cuales convierten esta solución en un referente en la optimización de sistemas embebidos en entornos con recursos limitados. En el caso específico de esta investigación, en dispositivos *cansats.*

Beneficios técnicos:

La propuesta se centra en explotar las virtudes del *Raspberry Pi Zero 2 W* y utilizar mecanismos y técnicas de programación para compensar las limitaciones intrínsecas de los CPUs de bajos consumo, por ejemplo, se implementó la carga de las imágenes en la memoria RAM para reducir las posibles latencias de entrada/salida que supondría operar las imágenes desde la memoria *SD*.

El programa también es capaz de generar métricas y archivos de metadatos los cuales resultan convenientes al momento de realizar experimentos y pruebas de rendimiento.

Sin embargo, la mejora técnica central de la propuesta es la implementación del paralelismo, el cual permite operar cuatro sub-imágenes al mismo tiempo; esto incrementa matemáticamente la salida del programa hasta cuatro veces. En otras palabras, en el tiempo que toma procesar una imagen completa en un procesamiento serial, la propuesta produce cuatro veces más imágenes en el mismo lapso.

Además, el diseño está parametrizado desde un archivo de configuración externo y controlable por el usuario. Uno de los parámetros más relevantes es el que permite definir el número de trabajadores. En un hipotético escenario en el que se produjeran minicomputadores con un mayor número de hilos, el programa podría adaptarse a ellos e incrementar su tasa de procesamiento de imágenes en simultáneo.

Beneficios académicos:

Los beneficios académicos de esta propuesta están estrechamente relacionados con la misión original de los desarrollos de nanosatélites en lata: brindar a los estudiantes de escuelas y universidades la oportunidad de experimentar el ciclo completo del desarrollo de satélites y la ingeniería aeroespacial. En el caso particular de esta propuesta, se tiene como meta acercar a los estudiantes al desarrollo de aplicaciones capaces de operar varios núcleos en simultáneo sin la necesidad de adquirir computadoras de torre o laptops que tienden a ser más costosas que los *Raspberry Pi*.

Además, conectar una problemática real como la falta de software optimizado para computadoras de bajos recursos, con el paralelismo como técnica para mejorar el desempeño de los sistemas puede incentivar a los alumnos a experimentar y optimizar cualquier sistema en el que reconozcan posibles cuellos de botella y oportunidades de mejora. Todo lo anterior es posible desde plataformas asequibles como los dispositivos *Raspberry Pi* que brindan un entorno seguro para ensayos sin tener que asumir pérdidas económicas cuantiosas en caso de fallas técnicas que inutilicen al equipo.

Beneficios prácticos

Operacionalmente, los beneficios de esta propuesta son varios. En primer lugar, la reducción del tamaño de los paquetes a enviar entendiendo “paquete” como cada fragmento de imagen individual, lo que permite aprovechar el ancho de banda disponible y garantizar la transmisión continua desde el momento en el que se termina de procesar la primera imagen hasta la última. Esta mejora mitiga la inactividad en el ancho de banda por la carencia de paquetes listos para la transmisión.

Sumado a lo anterior, el diseño está concebido para permitir la expansión de funcionalidades de los cansats, tanto para características nuevas dentro del procesado de imágenes como para poder agregar nuevas capacidades desacopladas del programa. Esto es posible debido a la arquitectura modular del software, la cual contribuye a la interoperabilidad del programa. Aunado a lo anterior, la propuesta sirve de ejemplo para demostrar cómo la optimización y un diseño pensado para correr en entornos limitados pueden abaratar los costes de producción y operación de sistemas informáticos reales.

Esto resulta cada día más crucial, ya que los costos operativos en los departamentos de Tecnologías de la Información suelen ser elevados y en ocasiones, un mejor diseño puede favorecer a la reducción de gastos e incrementar los beneficios utilitarios en el sector privado. Asimismo, favorece una mejor distribución de los recursos en el sector público.

En conjunto, los beneficios técnicos, académicos y prácticos de la propuesta evidencian su potencial para servir como plataforma de desarrollo, aprendizaje y optimización en sistemas embebidos de bajo costo.

5.6 Cronograma de actividades

Sec.	Actividad	2025						
		May	Jun	Jul	Ago	Sep	Oct	Nov
1	Revisión bibliográfica y estado del arte	□□□	□					
2	Diseño del sistema y definición de arquitectura		□□	□				
3	Implementación del prototipo en Raspberry Pi			□□□	□			
4	Ejecución de pruebas experimentales (modo paralelo y secuencial)				□□			
5	Registro y análisis de resultados					□□		
6	Contrastación de hipótesis y validación del sistema					□	□	
7	Redacción de capítulos finales y elaboración de propuesta						□□	□
8	Revisión, discusión y correcciones						□□	□
9	Entrega y presentación definitiva							□□

Leyenda de colores

□ Revisión teórica y documental

□ Diseño y estructuración

□ Programación e implementación

 Experimentación y pruebas

 Análisis e interpretación

 Entrega final

5.7 Presupuesto

Sec.	Categoría	Valor ($USD)
1	Raspberry Pi Zero 2 W con accesorios (cables, tarjeta microSD, disipador, carcasa, Fuente de alimentación)	$ 60.00
2	Módulo de cámara compatible (Camera Module v2 o similar)	$ 35.00
3	Material de papelería, impresión y encuadernación del documento final	$ 40.00
4	Transporte y logística de pruebas	$ 30.00
5	**Subtotal de hardware y materiales**	**$ 165.00**
6	Bibliotecas Python y sistema operativo (software libre)	**$ 0.00**
7	**Total general estimado**	**$ 165.00**

5.8 Diseño de la propuesta

El diseño de la propuesta se basa en los resultados de las pruebas experimentales los cuales demuestran la superioridad de la implementación del procesamiento paralelo frente al secuencial en la *Raspberry Pi Zero 2 W*. Además, el diseño se sustenta teóricamente sobre los conceptos y bases que plantean el paralelismo, los sistemas

embebidos y los cansats para obtener como resultado un programa consistente, robusto y funcional.

5.8.1. Enfoque arquitectónico general

A nivel general, la arquitectura del programa sigue un enfoque modular y un alto nivel de desacoplamiento entre los componentes. Esta línea de diseño favorece la mantenibilidad y reusabilidad y facilita la actualización continua del software. Cada funcionalidad de la aplicación se encuentra encapsulada y se invocan desde una función principal que coordina toda la ejecución del programa.

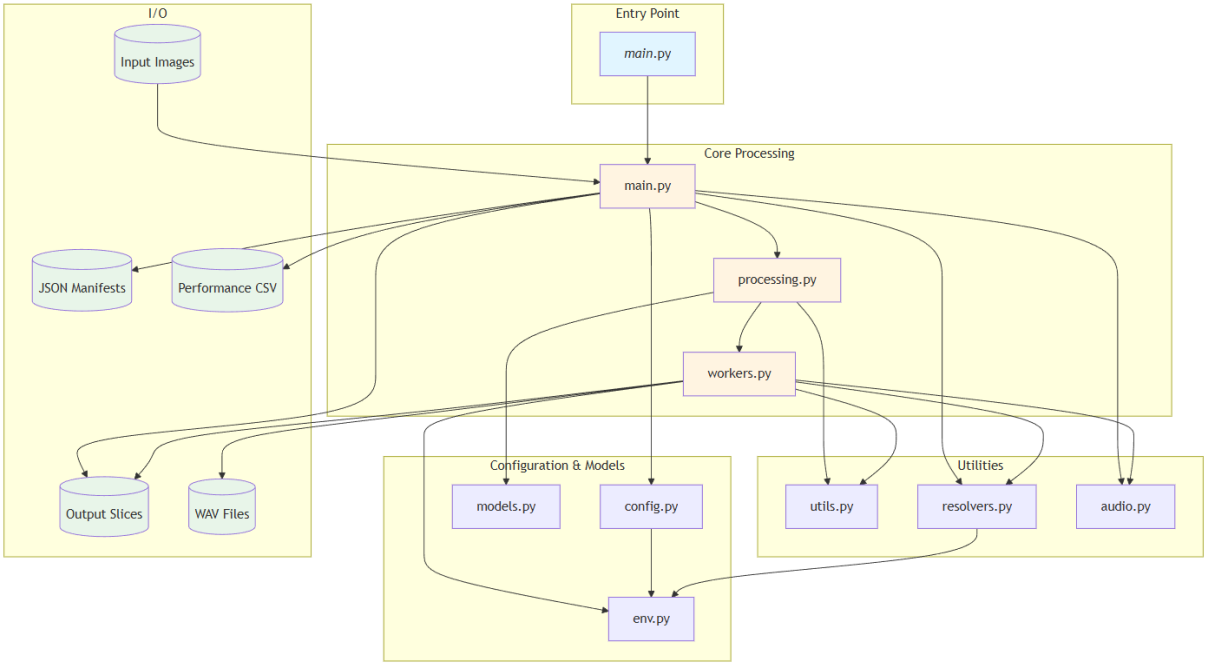


Ilustración 4. Diagrama de flujo de datos entre módulos

5.8.2. Módulos principales del sistema

Módulo de configuración (config)

El módulo de configuración se encarga de leer las configuraciones definidas en el archivo properties.yml y las carga como atributos definidos en un un objeto de la clase “Config”. Este objeto permite el acceso a los valores de configuración de cada parte del programa. A continuación, se detalla en el siguiente cuadro cada parámetro de configuración y sus valores posibles.

Parámetro	Tipo de dato / Valores posibles	Función
input_dir	str (ruta)	Directorio donde se ubican las imágenes de entrada.
output_dir	str (ruta)	Directorio donde se almacenan las imágenes procesadas y comprimidas.
report_csv_path	str (ruta)	Ruta al archivo CSV donde se registran las métricas de rendimiento.
image_output	bool (True/False)	Activa o desactiva la generación de imágenes comprimidas.
wav_output	bool (True/False)	Habilita la generación opcional de señales SSTV en formato WAV.
wav_output_dir	str (ruta)	Carpeta donde se guardan los archivos WAV generados.
wav_sample_rate	int (Hz)	Frecuencia de muestreo para los archivos WAV.
wav_reproduction_speed	float (≥ 0.01)	Factor de aceleración o desaceleración en la reproducción del audio.
rows	int	Número de divisiones verticales (filas) en que se segmenta cada imagen.
cols	int	Número de divisiones horizontales (columnas) para el slicing de la imagen.

max_workers	int	Cantidad máxima de procesos o hilos concurrentes usados para compresión.
format	str ("jpeg", "webp", etc.)	Formato de compresión a aplicar sobre las subimágenes.
quality	int (1–100)	Nivel de calidad de compresión; mayor valor implica menos pérdida.
rf_packet_size	int (bytes)	Tamaño del paquete simulado para transmisión por radiofrecuencia.
rf_preamble	str	Cadena que se antepone a cada paquete RF como preámbulo identificador.
engine	str ("auto", "pillow", "pyvips")	Motor de procesamiento de imágenes a utilizar.
executor	str ("auto", "process", "thread")	Tipo de ejecutor concurrente empleado.
compute_sha256	bool	Determina si se calcula el hash SHA-256 de cada archivo generado.
jpeg_subsampling	str ("420", "444")	Tipo de submuestreo de color en compresión JPEG.
jpeg_optimize	bool	Activa la optimización interna del codificador JPEG.
jpeg_progressive	bool	Genera imágenes JPEG progresivas (carga por capas).
webp_method	int (0–6)	Método de compresión de WebP; valores mayores implican más esfuerzo y calidad.
profile	str ("auto", personalizado)	Perfil general de configuración o entorno.
performance_profile	str ("speed", "ratio", "custom")	Perfil de rendimiento: prioriza velocidad, compresión o configuración personalizada.
use_tmpfs_cache	bool	Indica si se usa caché temporal en memoria (/dev/shm) para acelerar el acceso.
pin_affinity	bool	Fija la afinidad de los procesos a núcleos específicos del CPU.
tmpfs_dir	str (ruta)	Directorio temporal en memoria para caché (/dev/shm/cansat_cache).

desired_resolution	str (e.g. "1280x720")	Resolución objetivo a la que se remuestrean las imágenes antes del slicing.

Módulo de detección de entorno (env)

Se encarga de detectar si el entorno cuenta con las librerías necesarias para poder ejecutar ciertas características, en caso de que no estén instaladas, se desactiva la ejecución de estas.

Módulo de resolución de recursos (resolvers)

Este módulo determina en tiempo de ejecución el número de núcleos, el tipo de paralelismo y el motor de imagen que se empleará en la ejecución del programa.

Módulo de procesamiento (*processing*)

Este módulo se encarga de procesar las imágenes de principio a fin. Realiza las funciones de preparar el directorio de salida y activar la utilización de memoria cache, aplica el redimensionado si el usuario lo habilita, abre las imágenes y calcula los mosaicos a dividir según las filas y columnas. A demás, el módulo se encarga de enviar todas las ejecuciones al *"pool"* de trabajadores para el procesamiento en paralelo. También, se encarga de capturar los tiempos de salida para generar las métricas de procesamiento.

Módulo de trabajadores (*workers*)

El módulo de los trabajadores se encarga de recibir cada segmento de imagen, y ejecutar sobre él todo el proceso de recorte, compresión y registro de métricas de rendimiento. Además, opera bajo los parámetros definidos por el usuario, aplica técnicas de optimización como la afinidad de *CPU* en sistemas *Linux*, asignando cada proceso a un núcleo específico para mejorar la eficiencia. Dependiendo del motor configurado, utiliza VIPS o Pillow para realizar la apertura, el recorte y la compresión del fragmento en formatos como JPEG o WebP, ajustando dinámicamente opciones de calidad, progresividad y muestreo cromático.

Durante toda la ejecución mide con precisión los tiempos de carga, recorte, compresión y guardado, además del porcentaje de uso de *CPU* por etapa, lo que permite analizar el comportamiento del paralelismo y la eficiencia del sistema. Adicionalmente, calcula el tamaño final del archivo, su *hash SHA-256* para asegurar integridad, y si la opción está habilitada, genera un archivo de audio ***SSTV* en formato *WAV*** a partir del segmento comprimido para que un hipotético módulo de transmisión por radiofrecuencia pueda enviar la imagen como pista de audio. La generación de la pista de audio se logra mediante el módulo de audio.

Módulo de audio (audio)

Permite la conversión opcional de imágenes a señales *SSTV,* generando archivos *WAV* que simulan la transmisión por radiofrecuencia. Esto añade un componente experimental relevante para las pruebas de comunicación en misiones educativas.

Módulo de utilidades (utils): Reúne funciones auxiliares esenciales, como la creación de directorios, el listado de imágenes, el cálculo de hashes criptográficos y la gestión de memoria temporal en cachés.

Módulo principal (main): Actúa como controlador maestro del sistema. Inicializa la configuración, aplica los perfiles de rendimiento, determina el motor de procesamiento y ejecuta el pipeline completo. Finalmente, genera manifiestos de resultados en formatos JSON, JSONL y CSV.

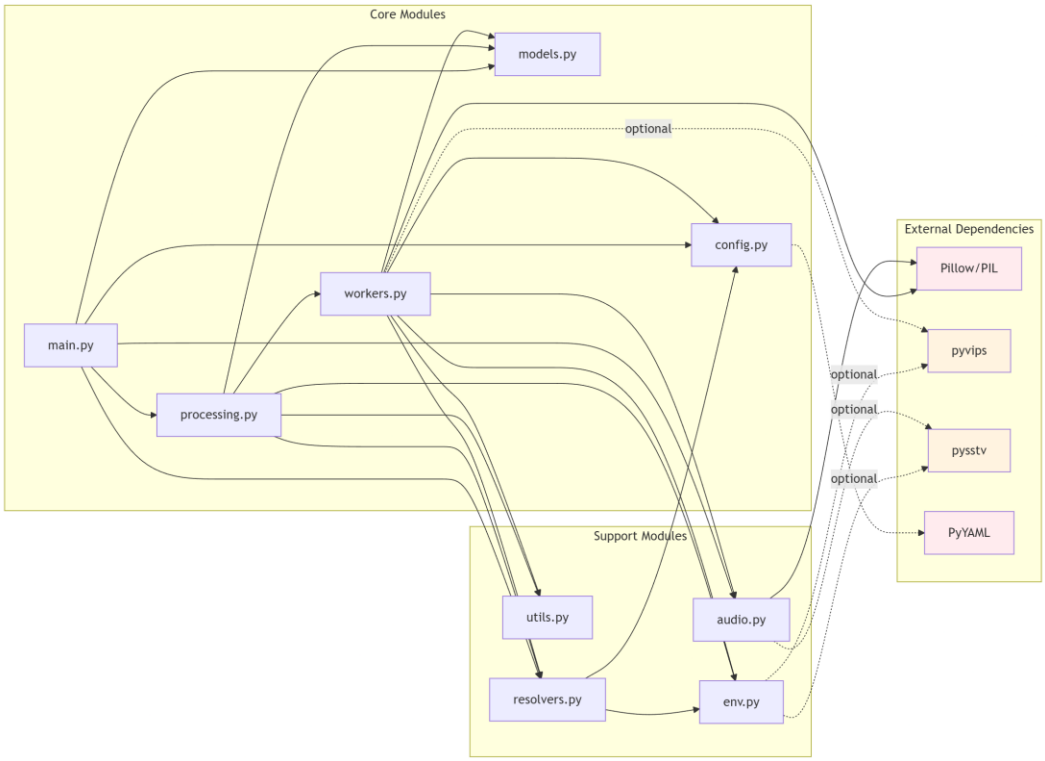


Ilustración 5. Diagrama de módulos y dependencias

5.8.3 Flujo de datos y procesamiento

El flujo inicia con la captura o selección de imágenes almacenadas en un directorio de entrada. Una vez cargadas, se aplican las configuraciones definidas por el usuario, incluyendo ajustes de resolución. Posteriormente, la imagen es dividida en mosaicos, y

cada fragmento se asigna a un trabajador. Los trabajadores recortan, comprimen y registran la información del fragmento, generando metadatos detallados. Estos resultados son devueltos al módulo coordinador, que consolida un manifiesto global por imagen y, finalmente, un resumen agregado por corrida experimental.

5.8.4 Perspectiva de transmisión y almacenamiento

La propuesta no incorpora un módulo para la transmisión de las imágenes. Sin embargo, sí prepara las imágenes para que puedan ser enviadas mediante radiofrecuencia en archivos de formato .WAV. El programa guarda las pistas de audio en una ruta especificada por el usuario, la cual por defecto es en: "./output_wav", relativa al programa. Esto facilita la lectura de estas pistas de audio por parte de otros programas o módulos que permitan agregar la capacidad de transmisión al cansat.

5.8.5 Escalabilidad y replicabilidad

La propuesta sigue un diseño modular, lo cual permite la experimentación y replicabilidad de cada una de las partes del programa, esto facilita las pruebas y replicabilidad del sistema. También permite mejoras técnicas, ya que, al aislar los componentes, se puede mejorar partes individuales sin afectar al resto de componentes. Además, el programa emplea bibliotecas abiertas en Python, lo cual contribuye a la mantenibilidad del software.

5.8.6 Valor agregado de la propuesta

El valor de la propuesta no solo está en el aspecto técnico, que mejora significativamente los tiempos de ejecución de procesamiento de imágenes, sino que también ofrece a las instituciones académicas y a los estudiantes un programa que puede servir como base

para futuras investigaciones sobre los conceptos de sistemas embebidos, optimización de procesos y paralelismo. Aunado a lo anterior, el precio relativamente accesible de desarrollo y ejecución del proyecto permite a los estudiantes experimentar y poner en práctica los principios de los campos antes mencionados.

CONCLUSIONES

La investigación desarrollada permitió demostrar que la aplicación de técnicas de procesamiento paralelo en la *Raspberry Pi Zero 2 W* mejora de manera significativa la eficiencia del pipeline de imágenes en comparación con un enfoque secuencial. Los resultados experimentales, obtenidos a partir de diez corridas controladas, evidenciaron una reducción de los tiempos de ejecución cercana al 74% y un mejor aprovechamiento de los recursos de *CPU*, confirmando la validez de la hipótesis planteada.

El diseño arquitectónico propuesto, basado en módulos independientes y de bajo acoplamiento, mostró ser una estrategia viable para organizar tareas de alta demanda computacional en un entorno embebido. La separación de responsabilidades entre los módulos de configuración, procesamiento, trabajadores y utilidades facilitó la trazabilidad de cada fragmento de imagen y generó un sistema robusto y replicable.

En el plano académico, la investigación aporta un caso práctico que evidencia cómo conceptos avanzados de computación paralela pueden implementarse en dispositivos de bajo costo, generando un valor formativo significativo para proyectos educativos tipo cansat. En el plano práctico, se validó una solución que optimiza la transmisión y almacenamiento de imágenes, mejorando la eficiencia de las misiones y sentando bases para futuras integraciones con subsistemas de comunicación y control.

Asimismo, el carácter modular y flexible de la propuesta garantiza su escalabilidad, posibilitando la incorporación de nuevas funciones como cifrado, priorización de datos o

compresión avanzada, sin alterar la estructura central. Esta adaptabilidad refuerza su pertinencia en escenarios de investigación y experimentación de bajo presupuesto.

En conclusión, el trabajo no solo alcanzó sus objetivos generales y específicos, sino que también dejó sentadas las bases para investigaciones posteriores en el campo de los nanosatélites y la computación paralela aplicada a sistemas embebidos. La combinación de evidencia empírica, diseño modular y aplicabilidad real convierte a la propuesta en un aporte concreto al desarrollo de soluciones tecnológicas en el ámbito aeroespacial educativo.

RECOMENDACIONES

Ampliar el tamaño de la muestra: En futuras investigaciones sería conveniente incluir un mayor número de imágenes y diferentes resoluciones, con el fin de robustecer el análisis estadístico y validar la propuesta en escenarios más variados.

Explorar nuevas técnicas de compresión: Implementar y comparar algoritmos alternativos (como HEIF/HEVC o JPEG XL) que podrían ofrecer un mayor equilibrio entre calidad y tamaño de archivo en el contexto de transmisión limitada.

Incorporar pruebas en condiciones reales de transmisión: Integrar el sistema con módulos de radiofrecuencia y evaluar el desempeño en escenarios de campo, simulando interferencias y pérdidas de señal propias de misiones espaciales.

Optimizar la gestión energética: Analizar el impacto del procesamiento paralelo en el consumo de energía del dispositivo y proponer mecanismos de ahorro energético que prolonguen la autonomía en misiones prolongadas.

Agregar métricas de temperatura: Añadir métricas de temperatura del hardware puede ser útil en la detección y explicación de posibles límites técnicos o degradado en el rendimiento del programa cuando este se ejecute bajo extensos periodos de funcionamiento, lo cual puede disminuir las salidas del aplicativo.

Promover la replicabilidad educativa: Documentar y difundir la arquitectura y el código como recurso académico, fomentando su uso en cursos y proyectos de ingeniería que

busquen introducir a los estudiantes en la práctica de la computación paralela en sistemas embebidos.

Estas recomendaciones ofrecen un marco de continuidad y expansión para futuros trabajos, asegurando que la propuesta no solo permanezca vigente, sino que también evolucione para responder a desafíos más complejos en el campo de los nanosatélites y la exploración aeroespacial educativa

BIBLIOGRAFÍA

Fernández, C. (2023). *Diseño De Un Cansat a Bajo Costo*. https://revistas.ulatina.edu.pa/index.php/genteclave/article/view/284

Akiyama, M., Saito, T., Graduate School of Electrical and Information Engineering, Shonan Institute of Technology 1-1-25 Tsujido-nishikaigan, Fujisawa, Kanagawa 251-8511, Japan, & Department of Information Science, Faculty of Engineering, Shonan Institute of Technology 1-1-25 Tsujido-nishikaigan, Fujisawa, Kanagawa 251-8511, Japan. (2021). A Novel Method for Goal Recognition from 10 m Distance Using Deep Learning in CanSat. *Journal of Robotics and Mechatronics, 33*(6), 1359-1372. https://doi.org/10.20965/jrm.2021.p1359

Barr, M., & Massa, A. (2009). *Programming Embedded Systems: With C and GNU Development Tools*.

Chun, C., Tanveer, M. H., & Chakravarty, S. (2023). The CanSat Compendium: A Review of Scientific CanSats. *Machines, 11*(7), Article 7. https://doi.org/10.3390/machines11070675

Colin, A. (2015). The cansat technology for climate monitoring in small regions at altitudes below 1km. *ResearchGate*. https://www.researchgate.net/publication/317840722_The_cansat_technology_for_climate_monitoring_in_small_regions_at_altitudes_below_1km

De Marco, R., Di Nardo, F., Rongoni, A., Screpanti, L., & Scaradozzi, D. (2025). Real-Time Dolphin Whistle Detection on Raspberry Pi Zero 2 W with a TFLite Convolutional Neural Network. *Robotics, 14*(5), Article 5. https://doi.org/10.3390/robotics14050067

Deepak, R. A., & Twiggs, R. J. (s. f.). *Thinking Out of the Box: Space Science Beyond the CubeSat*.

ESA. (2018). *Getting Started with CanSat – A Guide to the Primary Mission | Teach with Space T08*. https://www.esa.int/Education/CanSat/Getting_Started_with_CanSat_A_Guide_to_the_Primary_Mission_Teach_with_Space_T08

Gadekar, A. (2025). *Design and Implementation of an Efficient Onboard Computer System for CanSat Atmosphere Monitoring* (No. arXiv:2308.03496). arXiv. https://doi.org/10.48550/arXiv.2308.03496

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing, Global Edition*. Pearson Education.

González Torres, A. S., Barrera Ortiz, D. R., Andreu Rivas, M. F., Rodríguez Torres, R. J., Bagur Nájera, J. A., Luttmann Von Ahn, J. C., & Medrano Yax, J. F. (2014). *CanSat Fase 2*. [Thesis, Universidad del Valle de Guatemala]. https://repositorio.uvg.edu.gt/xmlui/handle/123456789/723

Hasan, M., Rahman, I., Hossam, M., & Sadman, S. (2021). Design of CanSat for Environmental Monitoring and Object Detection. *ResearchGate*. https://doi.org/10.1109/ICEEICT53905.2021.9667830

He, Y., & Damas, M. C. (s. f.). *A Raspberry Pi Powered 1U CubeSat*. Recuperado 6 de agosto de 2025, de https://digitalcommons.usu.edu/smallsat/2021/all2021/1

Khalid Hosny, Ahmad Salah, & Amal Magdi. (s. f.). Parallel Image Processing Applications Using Raspberry Pi. *ResearchGate*. https://doi.org/10.1007/978-3-031-18735-3_6

Murillo Armas, A. (2022, julio). *Estimación Eficiente de Parámetros Ambientales mediante Procesamiento Paralelo de Algoritmos Básicos de Machine Learning sobre una*

Arquitectura Edge-IoT Distribuida [Info:eu-repo/semantics/bachelorThesis]. E.T.S.I de Sistemas Informáticos (UPM). https://oa.upm.es/71619/

Naiouf, M. (2004). *Procesamiento paralelo* [Tesis, Universidad Nacional de La Plata]. https://doi.org/10.35537/10915/2264

Nelli, F. (2023). *Parallel and High Performance Programming with Python: Unlock parallel and concurrent programming in Python using multithreading, CUDA, Pytorch and Dask.*

Nguyen, Q. (2018). *Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming*.

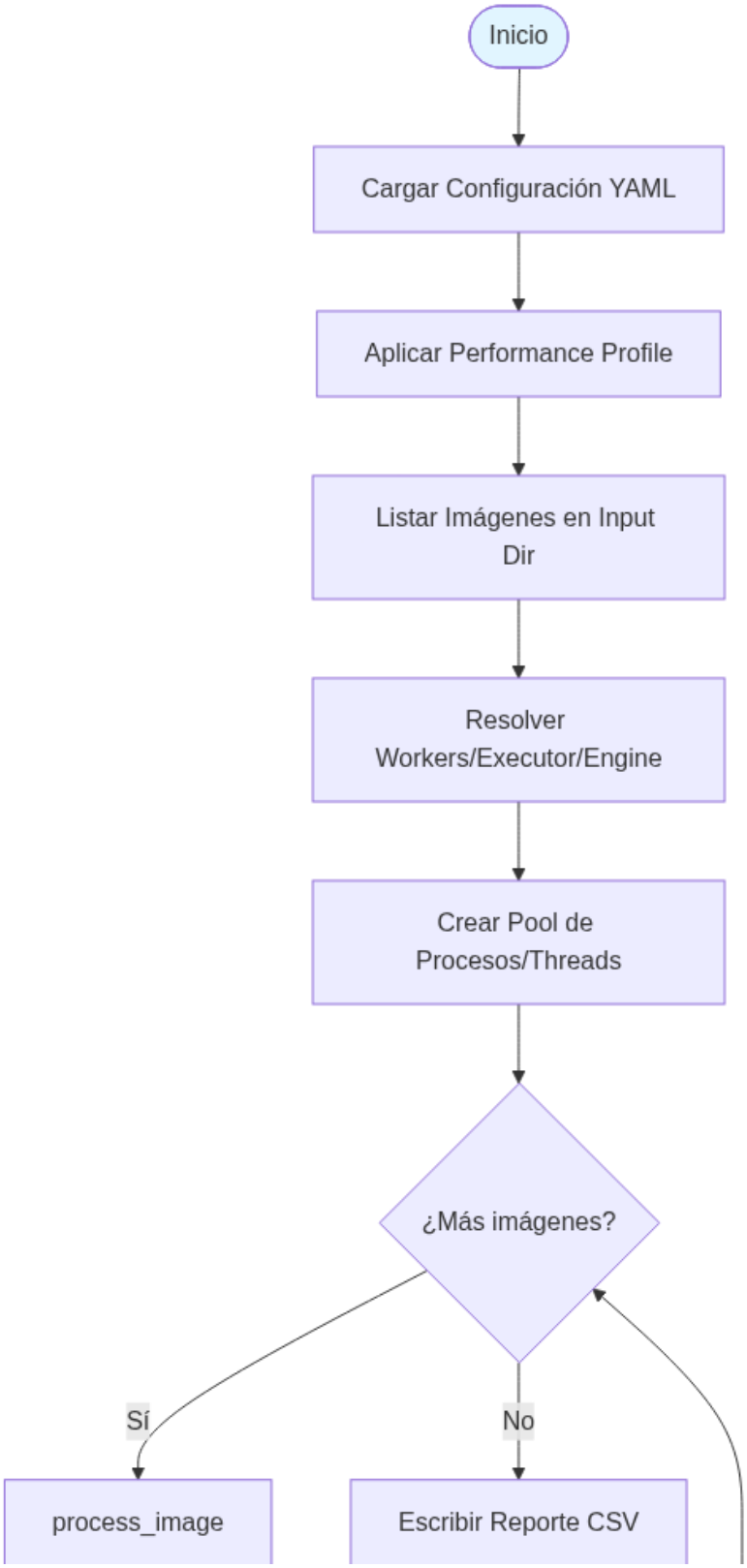
Raspberry Pi. (2021). *Buy a Raspberry Pi Zero 2 W*. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/

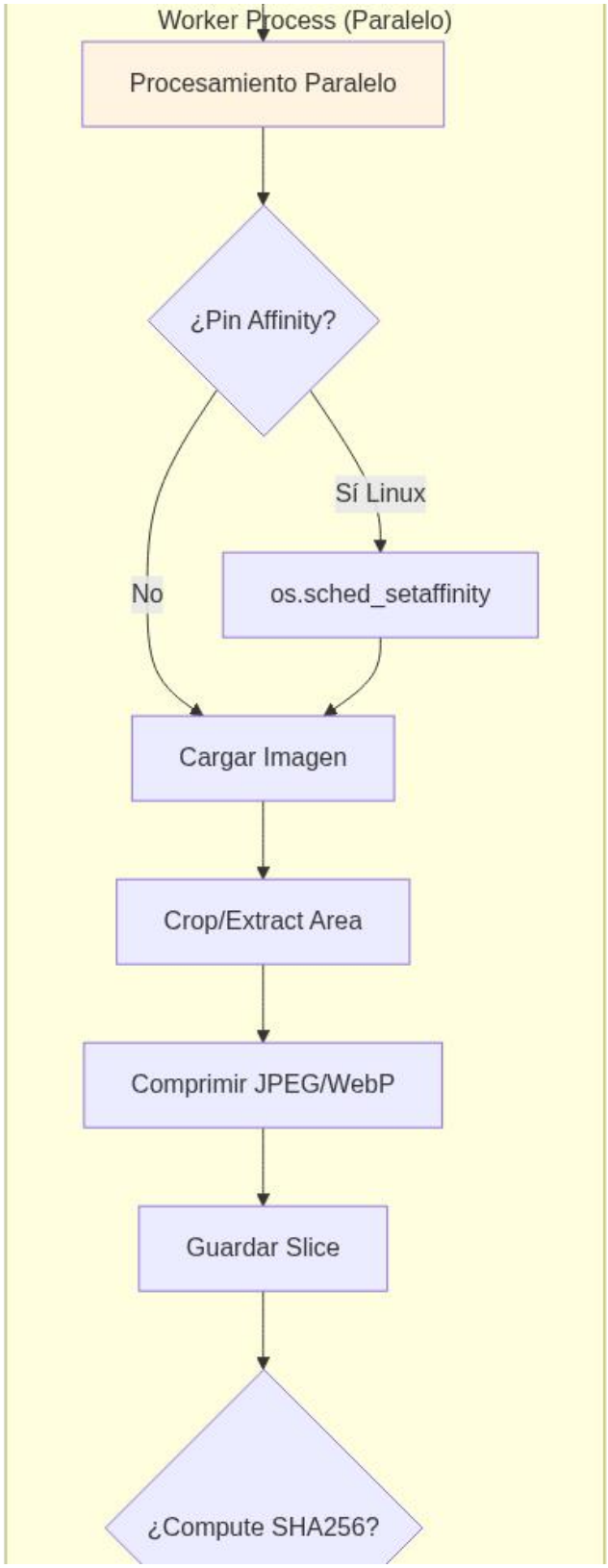
Smith, O. (2021). *Raspberry Pi Zero 2 W with Ubuntu Server 21.10 support is here*. Ubuntu. https://ubuntu.com/blog/raspberry-pi-zero-2-w-with-ubuntu-server-support-is-here

Students Win First Ever Canadian CanSat Competition with Design Incorporating FLIR Lepton. (2021). https://oem.flir.com/learn/discover/students-win-first-ever-canadian-cansat-competition-with-design-incorporating-flir-lepton/?utm_source=chatgpt.com

Umbaugh, S. E. (2023). *Digital Image Processing and Analysis: Digital Image Enhancement, Restoration and Compression*.

ANEXOS


Inicio
Cargar Configuración YAML
Aplicar Performance Profile
Listar Imágenes en Input Dir
Resolver Workers/Executor/Engine
Crear Pool de Procesos/Threads
¿Más imágenes?
Sí
No
process_image
Escribir Reporte CSV


Worker Process (Paralelo)
Procesamiento Paralelo
¿Pin Affinity?
Sí Linux
No
os.sched_setaffinity
Cargar Imagen
Crop/Extract Area
Comprimir JPEG/WebP
Guardar Slice
¿Compute SHA256?

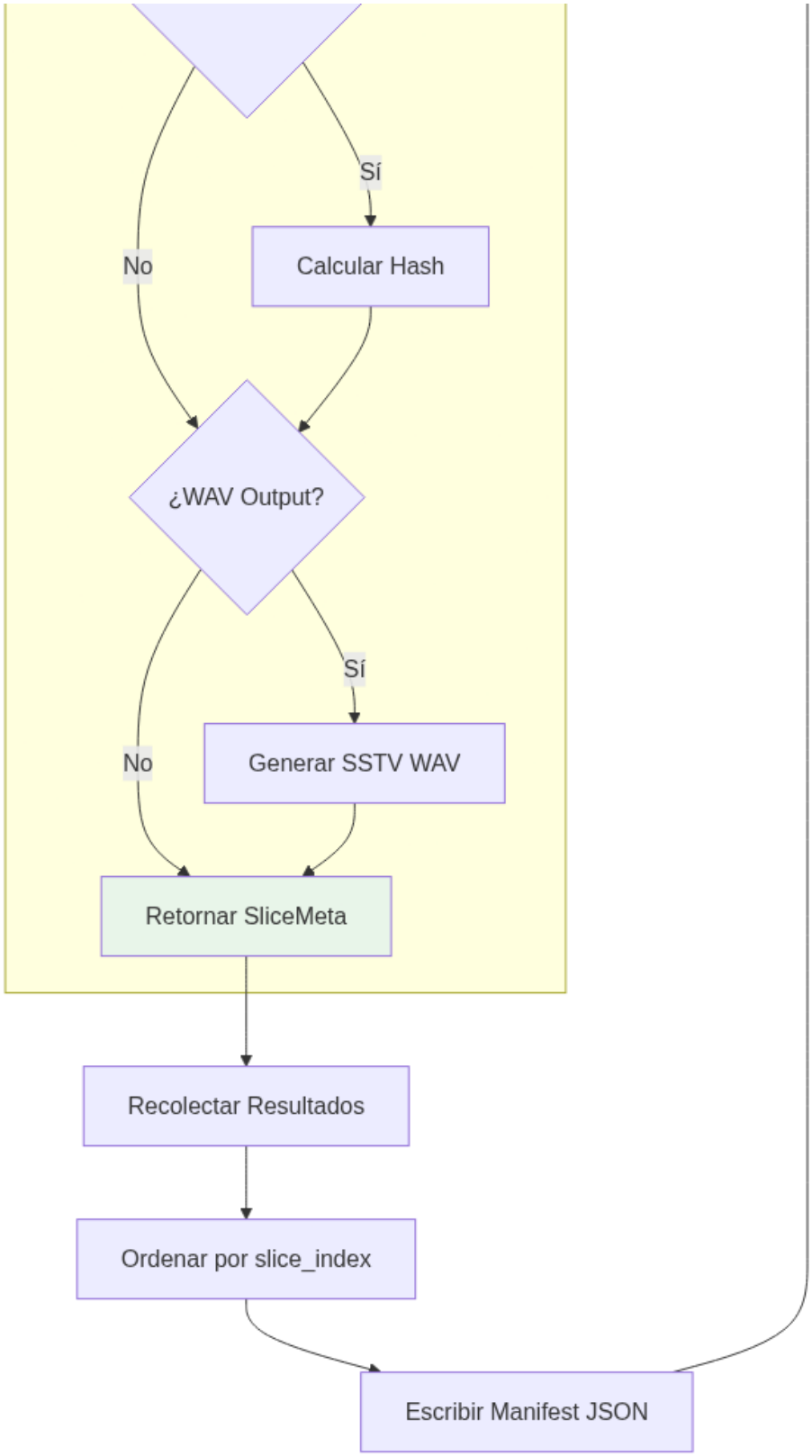


Ilustración 6. Diagrama de flujo del programa

Diagrama de secuencia

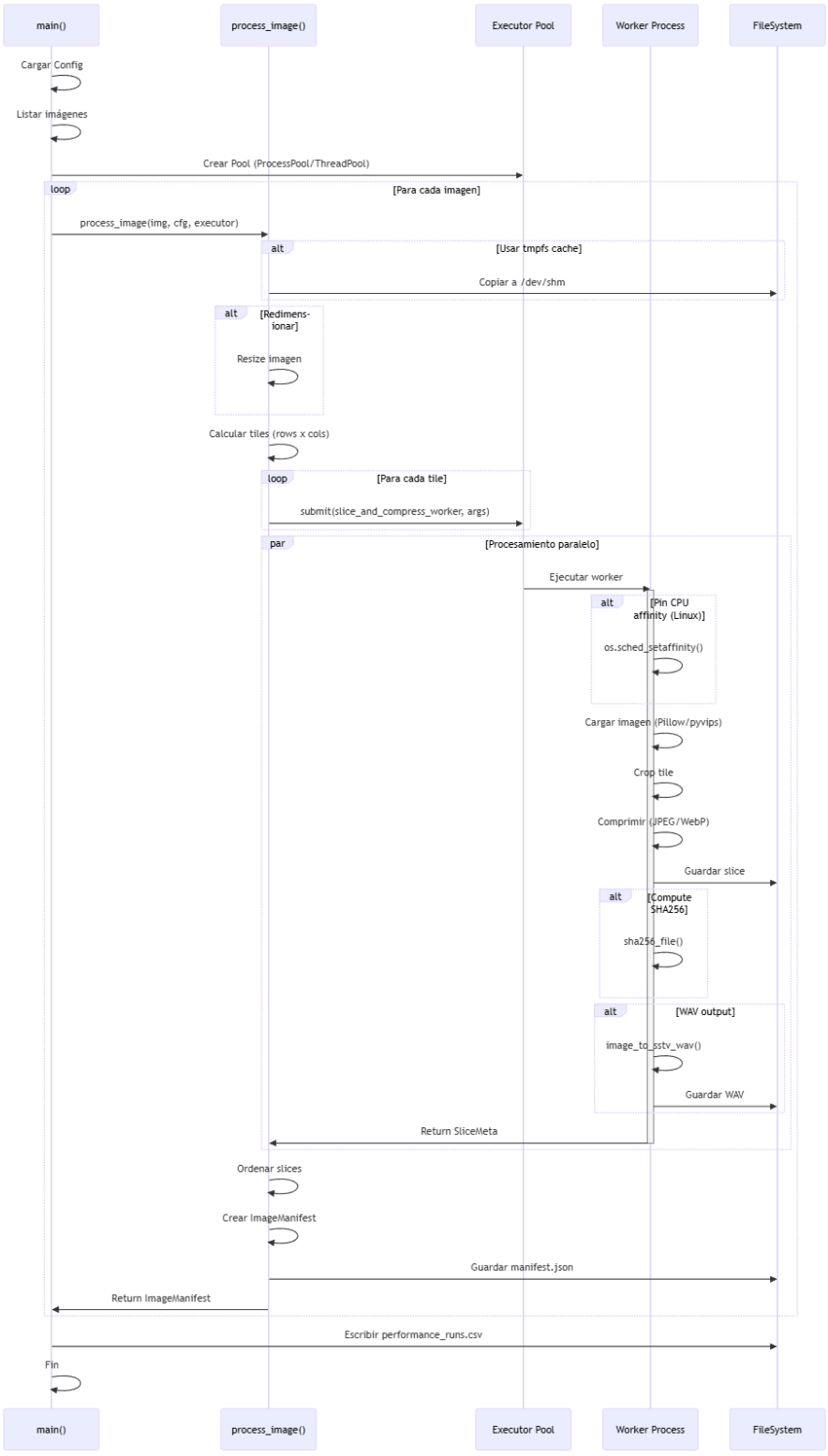


Ilustración 7. Diagrama de secuencia

URL del repositorio: https://github.com/MarksSecretCode/cansatsPrototypes.git

URL con los resultados de las pruebas:
https://github.com/MarksSecretCode/cansatsPrototypes/blob/main/performance_runs.csv