UNIVERSIDAD SANTO TOMÁS
SECCIONAL TUNJA

FACULTAD DE INGENIERÍA ELECTRÓNICA



Monitoreo autónomo de paneles solares mediante dron Parrot Anafi Thermal y visión artificial

Presentado por

Juan Camilo Riaño Pérez
Daniel Sebastian Benitez Paez

Para obtener el título profesional de

INGENIERO ELECTRÓNICO

Tunja Boyacá - Colombia
2026

UNIVERSIDAD SANTO TOMÁS
SECCIONAL TUNJA

FACULTAD DE INGENIERÍA ELECTRÓNICA



Monitoreo autónomo de paneles solares mediante dron Parrot Anafi Thermal y visión artificial

Trabajo de grado que presenta

Juan Camilo Riaño Perez
Daniel Sebastian Benitez Paez

Director del trabajo de grado:
Ing. Edgar Andrés Gutiérrez Cáceres

Codirector del trabajo de grado:
Ing. Camilo Ernesto Pardo Beainy

Tunja Boyacá - Colombia
2026

EXONERACIÓN

Nosotros, Juan Camilo Riaño Pérez y Daniel Sebastian Benitez Paez, autores del presente trabajo de grado titulado *Monitoreo autónomo de paneles solares mediante dron Parrot Anafi Thermal y visión artificial*, declaramos que el contenido desarrollado en este documento es de nuestra autoría, excepto en los casos en que se realizan las debidas referencias bibliográficas.

Asimismo, asumimos la responsabilidad por las opiniones, interpretaciones, resultados y conclusiones aquí expuestos, los cuales corresponden exclusivamente a los autores y no comprometen necesariamente a la Universidad Santo Tomás, a la Facultad de Ingeniería Electrónica ni a los directores o jurados del trabajo.

DEDICATORIA

Dedicamos este logro a todas las personas que, de una u otra manera, fueron sostén durante esta etapa y a quienes nos dieron motivos para seguir adelante, algo por lo cual esforzarnos, algo que defender y algo lo cual hacer. A quienes fueron impulso en los días difíciles, compañía en los días de trabajo y también refugio en aquellos momentos en los que fue necesario detenerse, descansar y recargar fuerzas.

Agradecemos a cada una de esas chispas que estuvieron presentes a lo largo de este proceso, por su apoyo, su paciencia, su afecto y su compañía. A quienes, con sus palabras, sus acciones y su presencia, ayudaron a hacer más llevadero este camino y dieron sentido a muchos de los esfuerzos que hicieron posible llegar hasta aquí.También agradecemos a quienes tendieron su mano cuando hizo falta, a quienes estuvieron cerca con respeto, con nobleza y con cariño sincero, y ayudaron a recoger el fruto de todo ese esfuerzo, sinceras gracias esas presencias que nos rodearon, nos sostuvieron y creyeron en nosotros durante esta etapa.

"De corazón a corazón, bendito sea el aforo La cama, la nana, la llama que expande mi ser y con la que acaloro A mi alrededor el honor y el respeto que dan esos seres que adoro".

AGRADECIMIENTO

Agradecemos sinceramente a todos aquellos ingenieros que, además de ser excelentes profesionales, también fueron docentes, compañeros y amigos durante el desarrollo de este trabajo, y que gracias a su tiempo, atención, compañía, aprecio y esfuerzo hicieron posible que este proyecto pudiera llevarse a cabo de la mejor manera, no solo desde lo técnico y académico, sino también desde lo humano.

Su disposición para orientar, compartir conocimientos, acompañar cada etapa del proceso y brindar apoyo constante fue fundamental para la construcción de este trabajo. Más allá de sus conocimientos profesionales, valoramos profundamente la cercanía, la confianza y la amistad con las que enriquecieron esta experiencia.

Asimismo, agradecemos a la universidad por permitir el uso de equipos avanzados, como el dron, así como por facilitar las instalaciones, espacios y recursos pertinentes para llevar a cabo este trabajo. Estos medios fueron esenciales para el desarrollo adecuado del proyecto.

Índice general

Resumen xi

Abstract xii

1 Introducción 1
1.1 Planteamiento del problema . . . 1
1.1.1 Pregunta de Investigación . . . 2
1.2 Objetivos . . . 3
1.2.1 Objetivo general . . . 3
1.2.2 Objetivos específicos . . . 3
1.3 Impactos esperados . . . 3
1.3.1 Corto plazo . . . 3
1.3.2 Mediano plazo . . . 3
1.3.3 Largo plazo . . . 4
1.4 Estructura del documento . . . 4

2 Estado del arte y análisis de vigilancia tecnológica 5
2.1 Arquitecturas de control y supervisión para UAV . . . 5
2.1.1 Arquitecturas Robóticas Distribuidas y Control Autónomo . . . 5
2.2 Visión por computador y análisis térmico para inspección fotovoltaica . . . 6
2.2.1 Sistemas de Inspección con UAV en Entornos Energéticos . . . 6
2.2.2 Termografía Infrarroja Aplicada a Sistemas Fotovoltaicos . . . 7
2.2.3 Herramientas Abiertas para Inspección Fotovoltaica . . . 8

3 Arquitectura de navegacion y ejecución de misión 10
3.1 Contexto teórico . . . 10
3.2 Propuesta metodológica . . . 11
3.2.1 Sistema de navegación base . . . 11
3.2.1.1 Papel de ROS 2 dentro del proyecto . . . 12
3.2.1.2 Nodos ROS 2 utilizados en la arquitectura . . . 13
3.2.2 Control de misión por waypoints . . . 14
3.2.2.1 Relación entre GUI, Olympe y el ejecutor de misión . . . 15
3.2.2.2 Control de alto nivel, PID y calibración dentro de la plataforma . . . 16
3.2.3 Resultados . . . 16

4 Procesamiento térmico y delimitación geométrica de la escena 18
4.1 Fundamentos teórico . . . 18
4.2 Desarrollo propuesto . . . 22

4.2.1 Delimitación geométrica y análisis térmico de la escena 22
4.2.2 Detección mediante umbrales usando sliders 24
4.2.3 Integración operativa y visualización dentro del sistema 25
4.3 Análisis de resultados 27
4.3.1 Análisis de curvas de entrenamiento, curvas de confianza y matriz de confusión 30
4.4 Manejo de la escena 35

5 Interfaz del usuario **36**
5.1 Requerimientos del computador 36
5.2 Instalación de Software 36
5.2.1 Instalación del sistema operativo 36
5.2.2 Entorno virtual e instalación Python 37
5.2.3 Instalación automática 37
5.2.4 Ejecución del proyecto 37
5.3 Manual de usuario por páginas 38
5.3.1 Página Sensores 39
5.3.1.1 Conexión y ciclo de actualización 40
5.3.1.2 Variables de telemetría utilizadas 40
5.3.1.3 Visualización 3D de trayectoria 41
5.3.2 Página Stream 41
5.3.2.1 Flujo de uso recomendado 42
5.3.2.2 Implementación del streaming RTSP 43
5.3.2.3 Control del gimbal 43
5.3.2.4 Bucle de reintento para establecer la conexión al stream 43
5.3.3 Página Galería RGB (gestión local de capturas) 44
5.3.3.1 Lectura de directorio y generación de miniaturas 45
5.3.3.2 Acciones de usuario 45
5.3.4 Página Galería Thermal 45
5.3.5 Página Errores (detección manual por umbrales y análisis por misión) 48
5.3.6 Página Misión de vuelo (editor JSON + ejecución de waypoints) . 51
5.3.7 Análisis comparativo de métricas térmicas, segmentación y altura de vuelo 55

6 Conclusiones y recomendaciones **61**
6.1 Conclusiones 61
6.2 Trabajos futuros 62
6.3 Recomendaciones 62

Referencias Bibliográficas **65**

Anexos **66**

Anexo A. Instalación del sistema operativo **67**

Anexo B. Plano del sistema fotovoltaico utilizado en las pruebas **78**

Anexo C. Secciones complementarias de instalación y configuración de la interfaz **79**

Anexo D. Preconfiguración del sistema **82**

Anexo E. Registro detallado del entrenamiento del modelo OBB **85**

Anexo F. Ejemplo de datos almacenados en una misión de vuelo **86**

Índice de figuras

Figura 3.2.1.1 Flujo de información del sistema base propuesto. Fuente: elaboración propia. 12
Figura 3.2.1.2 Grafo ROS 2 observado durante la ejecución de la misión. Se distinguen el nodo /anafi/mission_manager, el cliente /anafi_gui_mission_client y /anafi/mission. Fuente: elaboración propia. 14
Figura 3.2.2.1 Secuencia de navegación y control de misión por waypoints. Fuente: elaboración propia. 15
Figura 3.2.3.1 Grafo ROS 2 observado durante la ejecución de la misión. Se distinguen el nodo /anafi/mission_manager, el cliente /anafi_gui_mission_client y los tópicos de estado, evento y progreso dentro del espacio de nombres /anafi/mission. Fuente: elaboración propia. 17

Figura 4.1.1 Comparación entre detección de objetos tradicional mediante cajas rectangulares y detección mediante *Oriented Bounding Boxes* (OBB), donde la orientación del objeto se considera durante la localización. Fuente: adaptado de la documentación oficial de Ultralytics [1]. 19
Figura 4.2.1.1 *Setup* propuesto para la delimitación geométrica de la escena y el desarrollo de la Sección 4.2.1. Fuente: elaboración propia. 23
Figura 4.2.1.2 Targets térmicos implementados como referencia geométrica y térmica dentro de la escena evaluada. Fuente: elaboración propia. . 24
Figura 4.2.2.1 Interfaz de ajuste de umbrales mediante *trackbars* para la detección de regiones térmicas de interés. Fuente: elaboración propia. 25
Figura 4.2.3.1 Escena real (*setup* final) utilizada durante las pruebas finales y la comprobación del sistema. Fuente: elaboración propia. 26
Figura 4.2.3.2 Plano general de la escena evaluada para las pruebas del sistema fotovoltaico, correspondiente al Anexo B. Fuente: elaboración propia. 27
Figura 4.3.1 Comparación entre la captura original y la escena recortada a partir de la detección de *targets* térmicos mediante el algoritmo OBB entrenado. Fuente: elaboración propia. 30
Figura 4.3.1.1 Curvas generales de entrenamiento y validación del modelo a lo largo de las épocas. Fuente: elaboración propia. 31
Figura 4.3.1.2 Curva F1-score en función del umbral de confianza del detector. Fuente: elaboración propia. 31

Figura 4.3.1.3 Curva Precision–Recall obtenida durante la evaluación del modelo. Fuente: elaboración propia. 32
Figura 4.3.1.4 Matriz de confusión normalizada del modelo entrenado. Fuente: elaboración propia. 33
Figura 4.3.1.5 Visualización de una región térmica de interés dentro de la interfaz del sistema, donde se evidencia la detección de un punto caliente y su asociación con la información térmica reportada durante el análisis. Fuente: elaboración propia. 34

Figura 5.3.1 Dashboard lateral: indicadores de estado, botones de acción rápida y navegación por páginas. Fuente: elaboración propia. 39
Figura 5.3.1.1 Página Sensores: telemetría (IMU/velocidad/GPS), estado general y botón Conectar. Fuente: elaboración propia. 40
Figura 5.3.1.2 Gráfico 3D: trayectoria del dron en el marco ENU (E, N, Alt). Fuente: elaboración propia. 41
Figura 5.3.2.1 Página Stream: acceso a Autonomía ROS2 (rojo), control de streaming (naranja) y control de gimbal/pitch (azul). Fuente: elaboración propia. 42
Figura 5.3.2.2 Stream RGB en vivo durante un vuelo de prueba (despegue y estacionario). Fuente: elaboración propia. 43
Figura 5.3.3.1 Página Galería RGB: botón Refrescar (azul), Abrir carpeta (rojo) y acción Abrir por elemento (verde). Fuente: elaboración propia. . 44
Figura 5.3.4.1 Página *Galería Thermal*: actualización de la vista, apertura del directorio térmico y acceso a archivos individuales de la misión y del análisis. Fuente: elaboración propia. 46
Figura 5.3.4.2 Visor térmico detallado: archivo visual mapped, lectura por píxel, acceso a la carpeta de matrices y resumen de la matriz radiométrica asociada. Fuente: elaboración propia. 47
Figura 5.3.5.1 Página *Errores*: organización por misión, vista previa térmica, umbralización manual y ejecución del análisis de salida. Fuente: elaboración propia. 49
Figura 5.3.5.2 Ejemplo de operaciones morfológicas utilizadas en procesamiento de imágenes, incluyendo erosión, dilatación, apertura y cierre, aplicadas durante el posprocesamiento de la máscara de detección. Fuente: elaboración propia. 50
Figura 5.3.6.1 Interfaz principal de la página de misión de vuelo con identificación de sus bloques funcionales. Fuente: elaboración propia. 53
Figura 5.3.6.2 Panel de edición de waypoints, donde se ingresan los parámetros de desplazamiento, orientación, espera y acción asociada. Fuente: elaboración propia. 53
Figura 5.3.6.3 Ventana emergente para cargar una misión de vuelo almacenada en formato JSON. Fuente: elaboración propia. 54
Figura 5.3.6.4 Ventana emergente para guardar la configuración actual de la misión en formato JSON. Fuente: elaboración propia. 54
Figura 5.3.7.1 Región de interés observada en las seis muestras tras el recorte geométrico basado en *targets*. 57
Figura 5.3.7.2 Evolución térmica de la región detectada y comparación del tamaño aparente de la zona de interés entre muestras. 58

Figura 5.3.7.3 Comparación entre la altura solicitada y la altura efectivamente alcanzada durante las capturas. 59

Figura 6.3.1 Descarga de Ubuntu 22.04.2 LTS. Se selecciona la imagen ISO para desktop. Fuente: elaboración propia. 67
Figura 6.3.2 Opciones de descarga de balenaEtcher según sistema operativo y arquitectura. Fuente: elaboración propia. 68
Figura 6.3.3 Selección del archivo ISO desde balenaEtcher. Fuente: elaboración propia. 69
Figura 6.3.4 Selección de la memoria USB donde se escribirá la imagen. Fuente: elaboración propia. 69
Figura 6.3.5 Inicio del flasheo de la ISO en la memoria USB. Fuente: elaboración propia. 70
Figura 6.3.6 Ejemplo de configuración de arranque desde BIOS/UEFI. La ubicación de estas opciones varía según el fabricante. Fuente: elaboración propia. 71
Figura 6.3.7 Inicio del instalador al arrancar desde USB. Se selecciona *Try or Install Ubuntu*. Fuente: elaboración propia. 71
Figura 6.3.8 Pantalla de bienvenida del instalador. Se selecciona el idioma y la opción *Instalar Ubuntu*. Fuente: elaboración propia. 72
Figura 6.3.9 Conexión a una red Wi-Fi durante la instalación para habilitar descargas y configuración regional. Fuente: elaboración propia. . . 73
Figura 6.3.10 Selección de instalación normal, descarga de actualizaciones e instalación de software de terceros. Fuente: elaboración propia. . . 74
Figura 6.3.11 Selección del tipo de instalación: Ubuntu único o Dual Boot. Fuente: elaboración propia. 75
Figura 6.3.12 Creación del perfil de usuario durante la instalación de Ubuntu. Fuente: elaboración propia. 76
Figura 6.3.13 Proceso de instalación en ejecución. Al finalizar se solicita reiniciar y retirar el USB. Fuente: elaboración propia. 77
Figura 6.3.14 Menú de calibración disponible en la aplicación oficial *FreeFlight 6*, donde se presentan las opciones de corrección de horizonte, calibración de gimbal y calibración del magnetómetro. Fuente: elaboración propia. 83
Figura 6.3.15 Procedimiento de calibración manual del magnetómetro, en el cual el dron debe girarse siguiendo los tres ejes indicados por la aplicación para restablecer una referencia espacial consistente. Fuente: elaboración propia. 83
Figura 6.3.16 Interfaz de corrección manual del horizonte, utilizada para ajustar la referencia visual de la cámara y compensar desviaciones en la alineación horizontal de la escena. Fuente: elaboración propia. . . . 84
Figura 6.3.17 Proceso de calibración del gimbal, en el cual la cámara ejecuta movimientos automáticos o asistidos sobre sus ejes con el fin de restablecer su orientación mecánica. Fuente: elaboración propia. . 84

Índice de tablas

Tabla 3.2.1 Ganancias de referencia documentadas para el controlador PID adoptado desde anafi_autonomy. 16

Tabla 4.1.1 GSD teórico calculado para distintas alturas de vuelo utilizando el sensor FLIR Lepton 3.5. 21
Tabla 4.3.1 Especificaciones técnicas de la plataforma NVIDIA Jetson Orin Nano utilizada durante el desarrollo y validación del sistema. . . . 28
Tabla 4.3.2 Configuración general empleada para el entrenamiento del modelo de detección OBB. 29
Tabla 4.3.3 Métricas globales del modelo YOLOv8 al final del entrenamiento. 30

Tabla 5.1.1 Requisitos recomendados para ejecutar la interfaz. 36
Tabla 5.3.1 Parámetros radiométricos relevantes registrados para la captura thermal_20260313_142146. 48
Tabla 5.3.2 Información almacenada en el archivo de metadatos de misión. . . 55
Tabla 5.3.3 Configuración general de la misión de vuelo empleada para las capturas térmicas comparativas. 56

Tabla 6.1 Parámetros principales del archivo config.py. 81

Resumen

Se propone diseñar e implementar un sistema de inspección térmica autónoma para la evaluación de paneles solares empleando un dron Parrot ANAFI Thermal. El sistema ejecutará rutas de vuelo preprogramadas desde un computador en tierra y realizará la captura de imágenes RGB y térmicas radiométricas con el fin de identificar puntos calientes asociados a fallas operativas y necesidades de mantenimiento. El control de misión se desarrollará en Python mediante la integración del SDK Parrot Olympe con una arquitectura basada en ROS 2, mientras que la detección de anomalías se abordará mediante técnicas de visión por computador orientadas al análisis térmico. Como resultado, se entregará una interfaz gráfica de usuario para el monitoreo del vuelo y la revisión de los hallazgos obtenidos. El proyecto busca reducir los tiempos de diagnóstico, mejorar la seguridad operacional y aportar evidencia objetiva que apoye las labores de mantenimiento predictivo.

El presente documento se organiza en seis capítulos principales.

En el Capítulo 1 se presenta la introducción del proyecto, incluyendo el planteamiento del problema, los objetivos y la propuesta metodológica general.

El Capítulo 2 desarrolla el estado del arte y el análisis de vigilancia tecnológica.

El Capítulo 3 describe la arquitectura y funcionamiento del sistema desarrollado, junto con una guía general para su ejecución y las actividades de prueba realizadas.

El Capítulo 4 presenta los resultados obtenidos y su respectivo análisis.

El Capítulo 5 expone la integración operativa en la interfaz gráfica y las funciones de visualización, configuración y gestión del sistema.

Finalmente, el Capítulo 6 reúne las conclusiones y recomendaciones derivadas del trabajo.

Palabras clave: Inspección térmica, sistemas fotovoltaicos, vehículos aéreos no tripulados (UAV), termografía infrarroja, ROS 2, visión por computador, mantenimiento predictivo.

Abstract

This project proposes the design and implementation of an autonomous thermal inspection system for the evaluation of photovoltaic panels using a Parrot ANAFI Thermal drone. The system will execute preprogrammed flight paths from a ground control station and will acquire synchronized RGB and radiometric thermal images in order to identify hotspots associated with operational faults and maintenance needs. Mission control will be developed in Python through the integration of the Parrot Olympe SDK with a ROS 2–based architecture, enabling reliable flight control and data acquisition. Anomaly detection will be addressed using computer vision techniques oriented toward thermal image analysis. As a result, a graphical user interface will be provided for mission monitoring and the review of detected findings. The project aims to reduce inspection time, improve operational safety, and provide objective evidence to support predictive maintenance activities.

The present document is organized into six main chapters.

Chapter 1 presents the project introduction, including the problem statement, objectives, and the general methodological approach.

Chapter 2 develops the state of the art and the technological surveillance analysis.

Chapter 3 describes the architecture and operation of the developed system, along with a general guide for its execution and the testing activities performed.

Chapter 4 presents the obtained results and their discussion.

Chapter 5 presents the operational integration within the graphical user interface, including visualization, configuration, and system management functions.

Finally, Chapter 6 outlines the conclusions and recommendations derived from the work.

Palabras clave: Inspección térmica, sistemas fotovoltaicos, vehículos aéreos no tripulados (UAV), termografía infrarroja, ROS 2, visión por computador, mantenimiento predictivo.

Capítulo 1

Introducción

En el contexto actual de expansión de los sistemas fotovoltaicos, la inspección y el mantenimiento preventivo de estos sistemas se han convertido en un aspecto clave para garantizar su eficiencia, seguridad y vida útil. Una de las técnicas más utilizadas para la detección temprana de fallas es la termografía infrarroja, la cual permite identificar anomalías térmicas como puntos calientes, celdas defectuosas o problemas en conexiones eléctricas sin necesidad de intervenir físicamente el sistema. Sin embargo, en instalaciones de gran tamaño, los métodos de inspección tradicionales resultan lentos, costosos y dependientes de personal especializado.

Ante este escenario, el uso de vehículos aéreos no tripulados (UAV) equipados con cámaras térmicas se ha consolidado como una alternativa eficiente para la inspección de plantas fotovoltaicas, ya que permite realizar evaluaciones rápidas, seguras y con mayor cobertura espacial. No obstante, muchas de las soluciones disponibles actualmente corresponden a plataformas propietarias y de alto costo, que integran procesamiento en la nube y sensores térmicos de alta resolución, lo que limita su accesibilidad en contextos académicos, experimentales o de investigación aplicada.

En respuesta a esta problemática, el presente proyecto propone el desarrollo de un sistema de inspección térmica autónoma basado en el dron Parrot ANAFI Thermal, integrando herramientas de software abiertas como ROS 2 y el SDK Parrot Olympe para el control de misión, adquisición de datos y gestión de la información generada durante el vuelo. El sistema permitirá programar rutas de inspección mediante navegación por waypoints, capturar imágenes térmicas radiométricas y visualizar los resultados a través de una interfaz gráfica de usuario, facilitando la identificación preliminar de anomalías térmicas en sistemas fotovoltaicos.

De esta manera, el proyecto busca establecer una solución funcional y reproducible que permita explorar el uso de plataformas accesibles de inspección térmica con UAV, contribuyendo al desarrollo de herramientas abiertas orientadas a la investigación, la validación experimental y el apoyo a procesos de mantenimiento predictivo en sistemas fotovoltaicos.

1.1. Planteamiento del problema

La inspección de grandes superficies fotovoltaicas mediante métodos manuales resulta lenta, costosa y propensa a errores, dificultando la identificación precisa de anomalías críticas como puntos calientes, los cuales pueden ser causados por celdas dañadas, sombreado parcial, fallos en diodos o conexiones defectuosas.

Los largos tiempos de recorrido para la inspección, la autonomía limitada del dron y la variabilidad de las condiciones térmicas ambientales impiden la obtención de datos consistentes y comparables entre diferentes jornadas de trabajo. A esto se suma la dependencia de vuelos controlados manualmente, los cuales exigen la presencia de personal capacitado y con licencia, quien debe operar desde una ubicación segura o a una distancia que cumpla con la normativa vigente. Este requisito no solo incrementa los costos y el tiempo necesario para la inspección, sino que también introduce un riesgo operativo para el personal involucrado.

Además, la confiabilidad de la captura térmica no depende únicamente de la ejecución del vuelo, sino también del cumplimiento de condiciones mínimas de adquisición y calidad de imagen, tales como la distancia de inspección, la resolución espacial efectiva sobre el módulo, el ángulo de observación y la compensación de variables radiométricas como la emisividad, la temperatura ambiente, la humedad relativa y la temperatura reflejada aparente [2, 3, 4]. En este mismo sentido, la bibliografía técnica reciente reporta que la inspección térmica aérea de sistemas fotovoltaicos exige controlar la geometría de captura y la calidad radiométrica para obtener resultados comparables y útiles en el diagnóstico [5, 6].

Adicionalmente, para mejorar la confiabilidad de la medición térmica pueden emplearse procedimientos de calibración más robustos que las aproximaciones simplificadas comúnmente usadas en cámaras térmicas compactas, incorporando el valor *raw* de la captura, la emisividad de la superficie, la temperatura ambiente y la temperatura reflejada. Asimismo, puede emplearse el uso de *targets* o referencias térmicas que permitan establecer puntos de temperatura estáticos y repetibles dentro de la escena. Estos *targets* suelen estar compuestos por materiales de baja y alta emisividad, lo que facilita su identificación y permite utilizarlos como una guía de referencia para verificar la consistencia de la captura térmica y la correcta interpretación de los valores medidos [3].

Si bien existen normativas y lineamientos que pueden regir de manera particular distintas etapas del proceso, como la inspección termográfica exterior de módulos y plantas fotovoltaicas en operación o la operación de sistemas de aeronaves no tripuladas en Colombia, el presente trabajo, por su carácter académico, plantea el desarrollo de un sistema funcional y reproducible que toma estos criterios solo como referencia de diseño, dejando abierta su mejora futura hacia un cumplimiento más riguroso de estándares técnicos e industriales [2, 4, 7].

Por lo tanto, se hace evidente la necesidad de desarrollar un sistema automatizado y reproducible que no solo gestione la adquisición de datos térmicos y la detección de anomalías, sino que también permita un control centralizado y seguro desde un PC local, minimizando la intervención humana y los riesgos asociados.

1.1.1. Pregunta de Investigación

¿Cómo desarrollar un sistema automatizado de inspección térmica con dron para sistemas fotovoltaicos que permita adquirir imágenes térmicas de calidad y detectar anomalías de manera confiable?

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un prototipo de software para el control, adquisición y análisis de datos en un sistema de inspección térmica autónoma de sistemas fotovoltaicos, empleando hardware existente basado en el dron Parrot ANAFI Thermal.

1.2.2. Objetivos específicos

1. Establecer una arquitectura de software, basada en ROS 2 y el SDK Parrot Olympe, que permita el control confiable del dron desde una estación base y la adquisición de imágenes térmicas e imágenes RGB orientadas a la inspección de sistemas fotovoltaicos.

2. Garantizar la ejecución de misiones autónomas repetibles y seguras mediante la navegación por coordenadas de referencia (waypoints), permitiendo la revisión periódica y comparativa de sistemas fotovoltaicos estáticos a partir de puntos de inspección previamente establecidos.

3. Habilitar un flujo operativo de inspección térmica funcional que integre la delimitación de la zona de análisis mediante marcadores de referencia, la identificación de anomalías térmicas generales y la visualización y gestión de los resultados a través de una interfaz gráfica de usuario.

4. Validar el funcionamiento del sistema propuesto mediante su implementación en un sistema fotovoltaico simple con carga, utilizado como caso base experimental para la evaluación del proceso de inspección térmica.

1.3. Impactos esperados

1.3.1. Corto plazo

En el corto plazo, se espera una reducción significativa de los desplazamientos y de la exposición a riesgos durante las campañas de inspección de sistemas fotovoltaicos, al reemplazar parcialmente los recorridos manuales por misiones autónomas con UAV. Esto permitirá disminuir el número de horas-hombre requeridas en campo, así como la cantidad de personal directamente expuesto a condiciones operativas potencialmente peligrosas. El impacto se reflejará principalmente en una optimización del tiempo de inspección y en una mejora de la seguridad operacional durante las labores de diagnóstico.

1.3.2. Mediano plazo

En el mediano plazo, el uso sistemático del sistema de inspección térmica permitirá una mejora del rendimiento operativo y energético de las instalaciones fotovoltaicas, gracias a la detección temprana y a la priorización de anomalías térmicas generales. Esta capacidad facilitará la toma de decisiones oportunas en las labores de mantenimiento, contribuyendo

a la prevención de fallas mayores y a la reducción de pérdidas energéticas asociadas a sobrecalentamientos prolongados.

1.3.3. Largo plazo

En el largo plazo, se espera la consolidación de prácticas de mantenimiento predictivo apoyadas en la inspección térmica autónoma, así como la escalabilidad del sistema hacia otras instalaciones fotovoltaicas de características similares. El proyecto podrá servir como base para la adopción de procedimientos más seguros, sistemáticos y reproducibles, fortaleciendo la estandarización de las prácticas de inspección y mantenimiento en entornos académicos, institucionales o de pequeña y mediana escala.

1.4. Estructura del documento

Este documento se organiza en seis capítulos principales, estructurados de manera progresiva para presentar el problema, fundamentar la propuesta, describir el desarrollo técnico realizado y exponer las conclusiones derivadas del trabajo.

El Capítulo 1, Introducción, presenta la contextualización general del proyecto. En él se desarrolla el planteamiento del problema, la pregunta de investigación, los objetivos y los impactos esperados, estableciendo así la base conceptual desde la cual se orienta el trabajo de grado.

El Capítulo 2, Estado del arte y análisis de vigilancia tecnológica, reúne los antecedentes técnicos y el análisis de vigilancia tecnológica relacionados con la inspección térmica de sistemas fotovoltaicos, el uso de UAV y las herramientas de software empleadas. Este capítulo permite ubicar la propuesta dentro del estado actual del tema y justificar las decisiones de diseño adoptadas.

El Capítulo 3, Arquitectura de navegacion y ejecución de misión, describe la arquitectura del sistema orientada al control del dron, la comunicación entre procesos y la adquisición de datos. En esta parte se expone la integración entre ROS 2 y el SDK Parrot Olympe, así como la lógica implementada para la ejecución de misiones autónomas y la supervisión operativa del sistema.

El Capítulo 4, Procesamiento térmico y delimitación geométrica de la escena, aborda el componente de visión artificial del proyecto. Allí se explica el flujo de procesamiento aplicado a las imágenes térmicas y RGB, incluyendo la delimitación geométrica de la escena, la detección de regiones de interés y la recuperación de información térmica útil para el análisis.

El Capítulo 5, Interfaz del usuario, presenta la integración operativa del sistema dentro de la interfaz gráfica de usuario. En este capítulo se describen las funciones de visualización, configuración, monitoreo y gestión de resultados que permiten al operador interactuar con la suite desarrollada de manera centralizada.

Finalmente, el Capítulo 6, Conclusiones y recomendaciones, sintetiza los principales hallazgos obtenidos durante el desarrollo del proyecto, discute su alcance y plantea recomendaciones y posibles líneas de mejora para trabajos futuros.

Capítulo 2

Estado del arte y análisis de vigilancia tecnológica

2.1. Arquitecturas de control y supervisión para UAV

El desarrollo de sistemas aéreos no tripulados para aplicaciones de inspección demanda plataformas de software capaces de integrar, de manera ordenada, el control del vehículo, la gestión de sensores y el intercambio de información entre distintos módulos. En este contexto, *ROS 2 (Robot Operating System 2)* se ha posicionado como una de las bases tecnológicas más relevantes en robótica moderna, debido a su enfoque modular, distribuido y orientado a la interoperabilidad. Su estructura permite organizar procesos en nodos especializados que se comunican mediante tópicos, servicios y acciones, facilitando el diseño de arquitecturas escalables, mantenibles y adaptables a distintos escenarios de operación. Estas características han favorecido su adopción en sistemas autónomos y semiautónomos, donde la coordinación entre percepción, control y monitoreo resulta fundamental para garantizar un comportamiento robusto y confiable.

2.1.1. Arquitecturas Robóticas Distribuidas y Control Autónomo

Desde la perspectiva de la robótica y los sistemas autónomos, el desarrollo de sistemas de inspección aérea requiere arquitecturas de software que permitan separar claramente el control del vehículo, la adquisición de datos y el procesamiento de la información. En este sentido, el middleware *ROS 2 (Robot Operating System)* se ha consolidado como una plataforma de referencia para el desarrollo de sistemas robóticos distribuidos, gracias a su modelo basado en nodos, tópicos y servicios, así como a sus capacidades de comunicación en tiempo real y escalabilidad.

Para plataformas UAV comerciales, la disponibilidad de SDK abiertos, como *Parrot Olympe*, facilita la integración del control de vuelo con aplicaciones desarrolladas en Python. La combinación de Olympe con ROS 2 permite diseñar sistemas en los que el control de misión, el registro de telemetría y la captura de sensores se ejecutan de forma paralela, reduciendo latencias y mejorando la seguridad operacional. Este enfoque resulta particularmente relevante en aplicaciones de inspección térmica, donde la estabilidad del vuelo, la correcta sincronización de datos y la trazabilidad de la información son factores determinantes para la calidad del diagnóstico.

En este contexto se ubica el presente proyecto, que toma como referencia los avances

académicos y comerciales existentes, pero propone una solución local, abierta y reproducible, orientada al desarrollo de un sistema de inspección térmica autónoma con UAV que pueda ser utilizado en entornos académicos y de investigación aplicada.

2.2. Visión por computador y análisis térmico para inspección fotovoltaica

La visión por computador constituye uno de los pilares tecnológicos en los sistemas modernos de inspección automatizada, al permitir la adquisición, procesamiento e interpretación de información visual para apoyar tareas de monitoreo, detección y diagnóstico. En aplicaciones con UAV, estas técnicas resultan especialmente relevantes porque facilitan la integración de imágenes visibles y térmicas dentro de un mismo flujo de análisis, haciendo posible no solo la localización de elementos de interés, sino también la identificación de patrones asociados a anomalías operativas. En el contexto de la inspección fotovoltaica, la visión artificial proporciona las bases para segmentar módulos, resaltar regiones de interés y relacionar la información geométrica y radiométrica con el estado funcional de los paneles, consolidándose como un componente clave en el desarrollo de herramientas de diagnóstico más precisas, reproducibles y escalables.

2.2.1. Sistemas de Inspección con UAV en Entornos Energéticos

El uso de vehículos aéreos no tripulados (UAV) en tareas de inspección técnica ha crecido de manera sostenida en los últimos años, particularmente en sectores como la energía, la infraestructura civil y las telecomunicaciones. Esta adopción se explica por la capacidad de los UAV para acceder a zonas extensas o de difícil acceso, reducir tiempos de operación y minimizar la exposición del personal a riesgos operativos. En el ámbito de los sistemas fotovoltaicos, los UAV se han consolidado como una herramienta eficaz para la evaluación visual y térmica de paneles solares, permitiendo detectar anomalías asociadas a fallas eléctricas, degradación de materiales o condiciones operativas no uniformes.

La inspección térmica de sistemas fotovoltaicos mediante vehículos aéreos no tripulados (UAV) se ha consolidado como una práctica eficiente para el diagnóstico rápido de fallas y la planificación de mantenimiento, especialmente en instalaciones de gran escala. En este contexto, la bibliografía y el desarrollo tecnológico coinciden en que la calidad del diagnóstico depende principalmente de la captura térmica radiométrica, la resolución espacial alcanzable (GSD) y la correcta asociación entre los datos térmicos y cada módulo inspeccionado [8, 9].

En paralelo al desarrollo académico, han surgido soluciones comerciales de extremo a extremo (E2E) que dominan actualmente el mercado de la inspección fotovoltaica. Plataformas como *Raptor Maps* y *Sitemark* integran la ingestión de imágenes RGB e infrarrojas, flujos automatizados de análisis y generación de reportes, apoyándose en arquitecturas propietarias y, en la mayoría de los casos, en procesamiento en la nube. Estas soluciones publican recomendaciones técnicas relacionadas con parámetros de captura, como rangos de resolución espacial expresados en centímetros por píxel, con el fin de estandarizar la calidad mínima de los datos adquiridos.

De forma complementaria, guías técnicas y literatura reciente resaltan que la utilidad del diagnóstico térmico no depende únicamente de la detección de anomalías, sino

también de la calidad radiométrica de la captura, de la resolución espacial efectiva sobre el módulo y del control de variables como el ángulo de observación, la irradiancia, la emisividad y la temperatura reflejada [8]. No obstante, dichas recomendaciones no siempre resultan directamente aplicables en contextos académicos o experimentales, especialmente cuando se emplea hardware de bajo costo o sensores térmicos de resolución limitada. En este sentido, los valores sugeridos de cm/px se consideran en el presente proyecto únicamente como una guía para la calibración y planificación de la captura, sin asumir el cumplimiento estricto de normativas técnicas, dada la complejidad operativa que esto implica en escenarios reales de prueba.

Desde el punto de vista del hardware, el sensor térmico y su resolución condicionan directamente el GSD alcanzable y, por ende, la altitud de vuelo y la seguridad de la operación. El *Parrot ANAFI Thermal*, equipado con un sensor FLIR Lepton 3.5 de resolución 160 × 120 píxeles, representa una plataforma con disponibilidad de un ecosistema abierto de desarrollo mediante *Parrot Olympe*, aunque requiere volar a alturas significativamente menores para alcanzar resoluciones espaciales comparables a las exigidas por plataformas comerciales [8, 9]. En contraste, drones como el *DJI Mavic 3T* o el *Autel EVO II 640T*, con sensores térmicos de 640 × 512 píxeles, permiten operar a mayores alturas manteniendo un nivel de detalle suficiente, lo que explica su adopción predominante en soluciones industriales [9]. Diversas guías técnicas coinciden en que, con sensores de baja resolución, es indispensable fijar un GSD objetivo y derivar a partir de este la altitud, velocidad y solapes de vuelo para garantizar reproducibilidad en la captura [9].

2.2.2. Termografía Infrarroja Aplicada a Sistemas Fotovoltaicos

La termografía infrarroja es una técnica ampliamente utilizada para el diagnóstico no invasivo de sistemas eléctricos y electrónicos, ya que permite visualizar la distribución de temperatura en una superficie sin contacto directo. En sistemas fotovoltaicos, las imágenes térmicas radiométricas facilitan la detección de puntos calientes, conexiones defectuosas, suciedad localizada y otros fenómenos que se manifiestan como variaciones térmicas respecto a un entorno de referencia.

En el ámbito académico, se han publicado datasets y estudios recientes que respaldan el uso de imágenes térmicas capturadas por UAV para el análisis de fallas fotovoltaicas. El dataset *PVF-10*, publicado en *Applied Energy*, proporciona imágenes térmicas de alta resolución con anotaciones para clasificación de fallas, mientras que conjuntos de datos más recientes disponibles en repositorios científicos incluyen capturas realizadas con drones comerciales y sensores térmicos de 640 × 512 píxeles, orientadas a detección de paneles y anomalías [8]. Asimismo, múltiples trabajos exploran *pipelines* de detección y segmentación de paneles combinados con análisis térmico, demostrando la viabilidad de asociar métricas térmicas a cada módulo cuando se controla adecuadamente la geometría de captura [8]. Estudios recientes también muestran que el monitoreo térmico basado en UAV permite no solo identificar anomalías, sino además comparar el comportamiento térmico de módulos y apoyar análisis de eficiencia en plantas solares [8].

Desde una perspectiva práctica, numerosos trabajos han demostrado que para tareas de diagnóstico preliminar no es estrictamente necesario emplear algoritmos complejos de aprendizaje profundo. En muchos casos, técnicas clásicas de procesamiento de imágenes, como la umbralización de temperatura y el análisis de regiones de interés, resultan

suficientes para identificar anomalías térmicas evidentes, siempre que la captura se realice bajo condiciones controladas y repetibles. Este enfoque reduce la complejidad computacional del sistema y facilita su implementación en entornos académicos o con hardware accesible.

Desde el punto de vista de la adquisición de datos, la calidad de una inspección térmica no depende únicamente de los algoritmos de análisis utilizados, sino también de las características físicas del sensor empleado durante la captura. Una de las plataformas más difundidas en entornos académicos y proyectos de investigación es el *Parrot ANAFI Thermal*, el cual incorpora un sensor FLIR Lepton 3.5 con resolución térmica de 160 × 120 píxeles, longitud focal efectiva aproximada de 1.93 mm y un campo de visión cercano a 57° × 42° [10]. Aunque esta plataforma ofrece ventajas relacionadas con su bajo costo, portabilidad y disponibilidad de herramientas abiertas de desarrollo como *Parrot Olympe*, la resolución disponible impone limitaciones sobre el nivel de detalle espacial que puede obtenerse durante una misión aérea.

Diversos trabajos relacionados con agricultura de precisión, inspección de infraestructura y monitoreo fotovoltaico mediante UAV coinciden en que la planificación de la captura debe realizarse a partir de una resolución espacial objetivo o *Ground Sample Distance* (GSD), parámetro que determina el tamaño mínimo de detalle observable sobre la superficie inspeccionada [11]. En aplicaciones donde se requiere identificar defectos localizados o anomalías térmicas de pequeñas dimensiones, suelen recomendarse resoluciones espaciales del orden de pocos centímetros por píxel, mientras que tareas de monitoreo general o vigilancia admiten valores superiores. Como consecuencia, la altura de vuelo, la velocidad de desplazamiento y los porcentajes de solape entre imágenes deben definirse a partir del GSD requerido para garantizar la repetibilidad y calidad de la inspección.

La evolución reciente de las plataformas térmicas comerciales ha impulsado la adopción de sensores de mayor resolución. Equipos como el DJI Mavic 3T y el Autel EVO Max 4T incorporan sensores térmicos de 640 × 512 píxeles, considerados actualmente una referencia dentro del sector industrial para aplicaciones de inspección, búsqueda y rescate o monitoreo de infraestructura [12, 13]. Frente a sensores de menor resolución como el FLIR Lepton 3.5, estas plataformas permiten mantener niveles de detalle equivalentes operando a mayores alturas de vuelo, incrementando simultáneamente la cobertura de terreno por imagen y los márgenes de seguridad operacional. En consecuencia, la resolución térmica y las características ópticas asociadas al sensor se han convertido en factores determinantes para la selección de plataformas destinadas a tareas de inspección técnica mediante UAV.

2.2.3. Herramientas Abiertas para Inspección Fotovoltaica

En cuanto a herramientas abiertas, existen repositorios y librerías que abordan parcialmente la inspección fotovoltaica con UAV, como *PVHawk* o proyectos basados en *OpenDroneMap/WebODM*, así como utilidades para la conversión y análisis radiométrico de imágenes térmicas mediante herramientas como *QGIS*, *ExifTool* o *thermal parsers* [9]. Sin embargo, estas soluciones suelen requerir adaptaciones significativas y no ofrecen un flujo completamente integrado que combine control de vuelo, adquisición de datos y análisis térmico orientado específicamente a sensores de baja resolución.

De forma paralela, plataformas comerciales ampliamente utilizadas en inspección

fotovoltaica, como *Raptor Maps*, *Sitemark*, *DroneDeploy* o *Pix4D*, han demostrado el valor operativo de integrar la captura aérea con procesos de análisis, generación de reportes y gestión centralizada de resultados. Estas soluciones se encuentran principalmente orientadas a entornos productivos y grandes instalaciones, donde la automatización y la escalabilidad constituyen requisitos prioritarios. No obstante, en contextos académicos y de investigación continúa siendo relevante disponer de arquitecturas abiertas que permitan modificar el flujo de adquisición, procesamiento y análisis de los datos, así como evaluar nuevas metodologías sobre plataformas experimentales o sensores térmicos con características distintas a las utilizadas habitualmente en entornos industriales [14, 15, 16, 17].

Capítulo 3

Arquitectura de navegacion y ejecución de misión

La arquitectura software de este trabajo se apoya en dos capas complementarias. La primera es Olympe, el SDK oficial de Parrot para el control programático de drones ANAFI desde Python sobre Linux, empleado para conexión, movimiento relativo, manejo de cámara y acceso a telemetría, la segunda es ROS 2, utilizado como capa de integración entre la interfaz gráfica, la lógica de misión y los procesos que supervisan o reutilizan la información del dron. En este proyecto ROS 2 no reemplaza al autopiloto del ANAFI Thermal; su papel consiste en organizar la comunicación, desacoplar responsabilidades y permitir que la GUI funcione como cliente de una arquitectura distribuida.

3.1. Contexto teórico

En este trabajo, ROS 2 se entiende como la infraestructura de coordinación de alto nivel sobre la cual se articulan los componentes software del sistema. A diferencia de un programa monolítico, ROS 2 permite organizar la aplicación en procesos especializados con responsabilidades diferenciadas, capaces de intercambiar información mediante mecanismos estandarizados como topics, services y, cuando es necesario, actions. Dentro del proyecto desarrollado, esta capacidad de desacoplamiento permite separar la interfaz gráfica, la publicación de telemetría y la ejecución de misiones en módulos con funciones claramente definidas. En consecuencia, ROS 2 no se utiliza únicamente como un canal de mensajería, sino como una capa de integración que facilita la modularidad, la observabilidad del sistema y la reutilización de información por parte de otros procesos. Gracias a ello, estados de misión, eventos de ejecución y datos de telemetría pueden ser supervisados desde la GUI o desde herramientas externas del ecosistema ROS 2 sin alterar la lógica principal de control.

Por su parte, Olympe constituye la capa de comunicación directa con el dron Parrot ANAFI Thermal, este SDK proporciona acceso programático a capacidades nativas de la plataforma, tales como la conexión con el vehículo, la ejecución de movimientos relativos, el control de cámara y gimbal, la consulta de estados y la lectura de telemetría. Bajo esta perspectiva, la relación entre ROS 2 y Olympe no es de reemplazo, sino de complementariedad: mientras ROS 2 organiza la arquitectura distribuida del sistema, Olympe materializa la interacción efectiva con el dron y sus subsistemas.

Esta separación de responsabilidades resulta conveniente para el objetivo del proyecto, ya que permite que la lógica de misión y supervisión permanezca desacoplada de la

comunicación de bajo nivel con la plataforma aérea, de esta manera, la interfaz gráfica puede asumir el papel de estación de planificación y monitoreo, ROS 2 puede encargarse de la interoperabilidad entre procesos, y Olympe puede concentrarse en la ejecución de las operaciones específicas del vehículo. En conjunto, esta organización proporciona una base coherente para integrar control de misión, captura de medios, telemetría y análisis posterior dentro de una misma suite de trabajo.

3.2. Propuesta metodológica

3.2.1. Sistema de navegación base

Para la implementación del proyecto se emplearon equipos y herramientas que permitan desarrollar un sistema de inspección térmica autónoma con UAV, garantizando la adquisición de datos térmicos y la ejecución segura del vuelo. La plataforma aérea seleccionada es un dron Parrot ANAFI Thermal, equipado con un sensor térmico radiométrico FLIR Lepton de resolución 160×120 píxeles y una cámara RGB integrada. Este dron fue seleccionado debido a su disponibilidad dentro del contexto académico del proyecto y por contar con un ecosistema abierto de desarrollo, el cual permite la integración con herramientas externas de control y programación. Como estación de control en tierra se utilizó un computador portátil con sistema operativo Ubuntu 22.04 LTS, con el fin de soportar la visualización y el procesamiento básico de los datos adquiridos.

El desarrollo del sistema se realizó sobre un ecosistema de software de código abierto, empleando Python 3.10 o superior, el SDK Parrot Olympe (versión 7.5.x) para el control del dron, y una arquitectura distribuida basada en ROS 2 (Humble o equivalente) para la gestión de la comunicación entre procesos. Adicionalmente, se utilizaron librerías de procesamiento y análisis de datos como OpenCV, NumPy, Pandas y Matplotlib, así como herramientas auxiliares para la visualización y gestión multimedia. Se entrenó un modelo de aprendizaje profundo únicamente para la detección de targets térmicos; las demás tareas de visión del sistema se apoyaron en técnicas clásicas de procesamiento de imágenes.

Dentro de este enfoque se implementó la arquitectura de control de misión del sistema. Como se observa en la Figura 3.2.1.1, el flujo de información para este objetivo fue el siguiente: la GUI envió la misión al nodo mission_manager, este coordinó la ejecución sobre el dron mediante Olympe y, finalmente, retornó estados, eventos, progreso y datos capturados para su visualización y análisis. Para ello se definió un nodo principal en ROS 2 denominado mission_manager, encargado de recibir la misión enviada desde la interfaz gráfica, establecer la conexión con el dron mediante el SDK Olympe y ejecutar los comandos de vuelo correspondientes (despegue, navegación, captura de imágenes y aterrizaje). Este nodo publicó el estado de la misión a través de los tópicos status, event y progress, los cuales permitieron informar el estado general de la misión, los eventos relevantes durante la ejecución y el progreso de la misma.



Figura 3.2.1.1: Flujo de información del sistema base propuesto. Fuente: elaboración propia.

De forma complementaria, la interfaz gráfica del sistema se implementó mediante el nodo anafi_gui_mission_client, encargado de enviar las misiones al mission_manager y de suscribirse a los tópicos de estado, eventos y progreso para actualizar la información mostrada al usuario durante la ejecución de la misión. Esta estructura permite separar el control del dron de la capa de interacción con el usuario, facilitando la supervisión del proceso de inspección desde la estación de control en tierra.

3.2.1.1. Papel de ROS 2 dentro del proyecto

La suite implementada quedó organizada alrededor de una raíz única, anafi_suite_ros2, que contiene simultáneamente la GUI, el *workspace* de ROS 2, el directorio de ejecución runtime/ y los scripts de arranque. Esta decisión tiene una consecuencia directa sobre la lógica del sistema: la misión deja de residir exclusivamente en memoria dentro de la interfaz y pasa a existir también como un recurso compartido entre procesos. Cuando el operador diseña una trayectoria desde la página *Misión de vuelo*, la interfaz puede serializarla hacia runtime/current_mission.json después, un nodo ROS 2 especializado lee ese archivo y realiza la ejecución física. Gracias a esta organización, la GUI se concentra en edición, visualización y supervisión, mientras que ROS 2 asume el papel de infraestructura de despacho y monitoreo.

En el proyecto, ROS 2 se manifiesta principalmente a través de dos funciones. La primera es la publicación de telemetría, implementada mediante el objeto Ros2TelemetryBridge, que toma datos leídos por Olympe y los reexpone en tópicos dentro del *namespace* /anafi. La segunda es la orquestación de misiones, resuelta mediante el nodo /anafi/mission_manager y el cliente de interfaz anafi_gui_mission_client. En otras palabras, ROS 2 no solo sirve para compartir estados del dron, sino también para desacoplar la fase de planificación visual de la fase de ejecución real del vuelo. Esta separación aporta varias ventajas. Primero, mejora la modularidad, porque los componentes de interfaz, bridge, ejecución y registro pueden evolucionar sin estar acoplados en una sola clase monolítica. Segundo, incrementa la observabilidad, ya que el estado del sistema puede inspeccionarse desde herramientas estándar como rqt_graph, ros2 topic echo o los archivos de log de runtime/logs.

Tercero, favorece la reutilización, porque la misma telemetría y el mismo estado de misión pueden alimentar nodos externos de autonomía o inspección sin modificar la GUI. Finalmente, reduce conflictos de conexión, ya que la arquitectura permite transferir la propiedad de la sesión de control Olympe desde la interfaz hacia el ejecutor de misión antes de iniciar el vuelo.

3.2.1.2. Nodos ROS 2 utilizados en la arquitectura

En la práctica, la arquitectura implementada se apoya en un conjunto acotado de nodos ROS 2 con responsabilidades diferenciadas. El nodo más importante para la ejecución de trayectorias es /anafi/mission_manager, implementado en ros2_ws/src/anafi_mission_manager/anafi_mission_manager/mission_manager_node.py, este nodo declara parámetros como mission_file y suite_root, que mantiene un estado interno de misión y expone tres servicios del tipo Trigger: /anafi/mission/ping, /anafi/mission/start y /anafi/mission/stop. Además, publica información serializada en formato JSON sobre los tópicos /anafi/mission/status, /anafi/mission/progress y /anafi/mission/event, de esta manera, el mission manager actúa como intermediario entre la descripción abstracta de una misión y su ejecución efectiva sobre el dron.

El segundo nodo relevante es anafi_gui_mission_client, creado desde app/interfaz/ros2_mission_client.py, a diferencia del mission manager, este nodo no ejecuta vuelo; su función es suscribirse a /anafi/mission/status, verificar la disponibilidad de los servicios de misión y realizar las llamadas necesarias para iniciar o detener una trayectoria, en la lógica de la GUI, este cliente se construye bajo demanda y se mantiene como puente ligero entre la página de misión y la infraestructura ROS 2. Gracias a este diseño, la interfaz puede comportarse como cliente de misión sin cargar dentro de sí toda la lógica de ejecución.

Un tercer nodo utilizado por el proyecto es /anafi/olympe_telemetry_bridge, creado desde app/ros2_bridge/telemetry_bridge.py, su objetivo es publicar hacia ROS 2 una réplica de la telemetría obtenida por Olympe: altitud, estado de vuelo, estabilidad, batería, orientación, velocidades y datos GPS. Este nodo mantiene compatibilidad semántica con la nomenclatura usual del ecosistema anafi_ros, de manera que otros nodos o herramientas ROS pueden consumir los tópicos sin depender directamente de la interfaz gráfica. A continuación, en la Figura 3.3.1.1 se pueden observar los tres nodos principales dentro de la infraestructura de ROS 2.

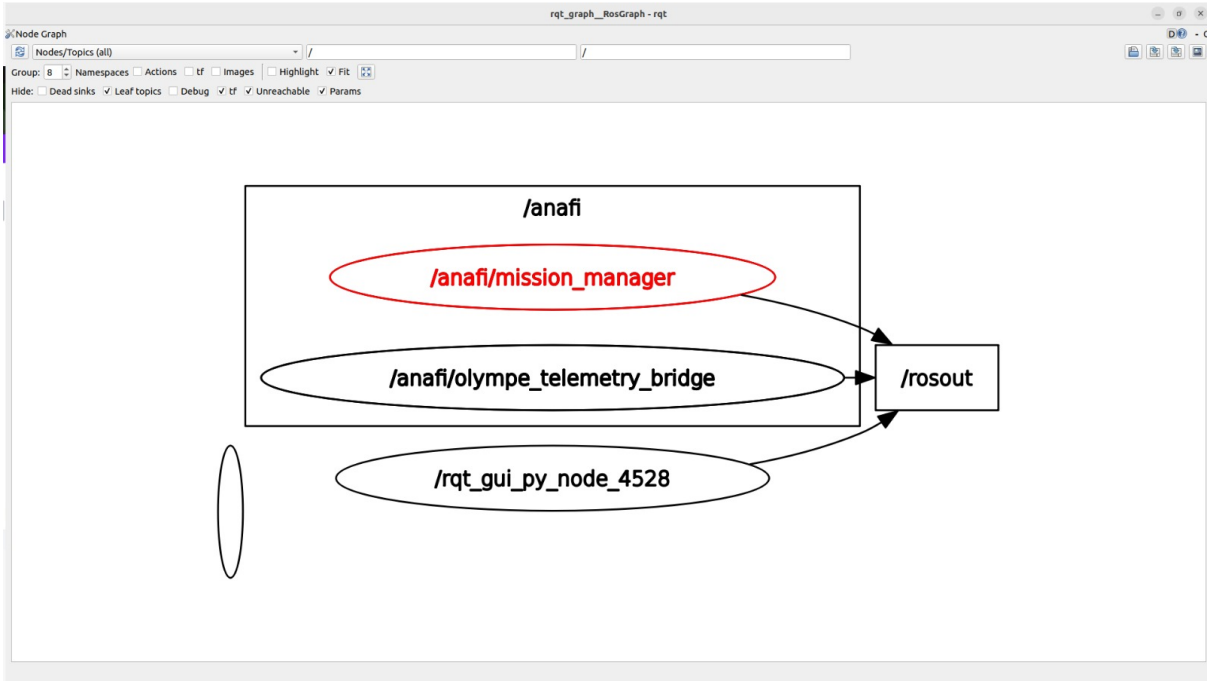


Figura 3.2.1.2: Grafo ROS 2 observado durante la ejecución de la misión. Se distinguen el nodo /anafi/mission_manager, el cliente /anafi_gui_mission_client y /anafi/mission. Fuente: elaboración propia.

3.2.2. Control de misión por waypoints

En consecuencia la implementación del control de misión autónoma mediante waypoints se presenta en la Figura 3.2.2.1, la misión fue ejecutada mediante comandos del SDK Olympe, siguiendo la secuencia de despegue, navegación por waypoints y aterrizaje, incorporando funciones de seguridad como el retorno automático al punto de inicio en caso de batería baja o pérdida de señal. Además, los comandos de navegación se enviaron de manera ordenada para cada waypoint a seguir, garantizando la continuidad de la ruta planificada. Durante toda la misión se registraron de forma continua la altura, la velocidad y la posición del dron, asociando cada captura RGB e infrarroja con su correspondiente estado de vuelo.

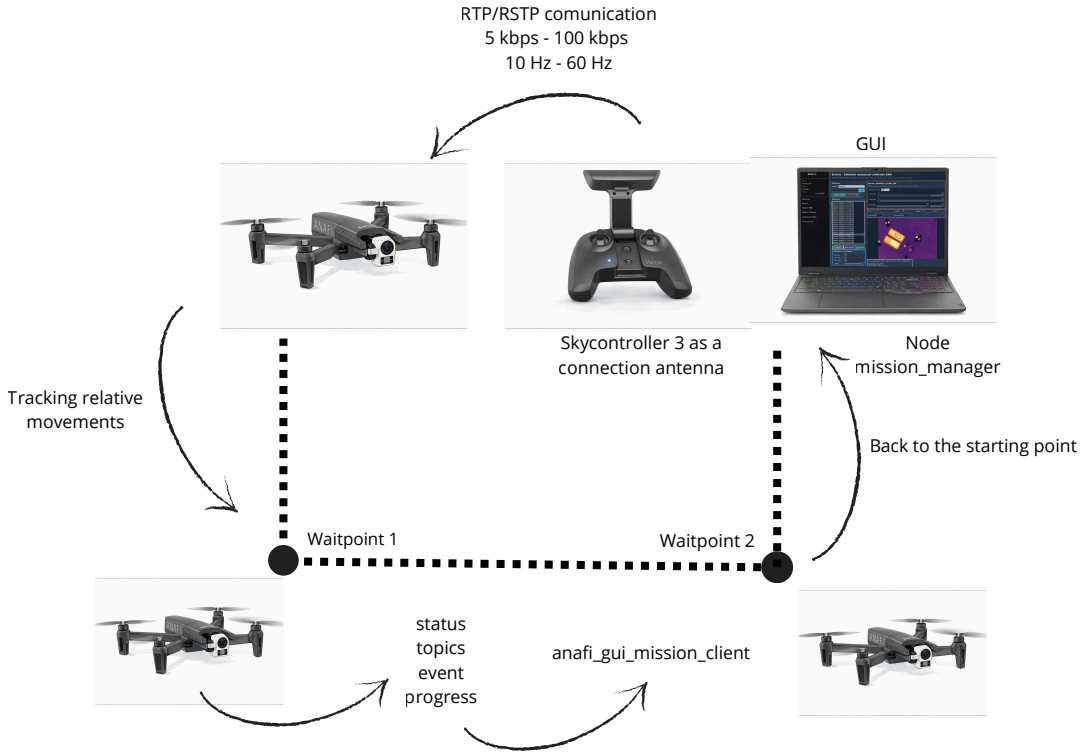


Figura 3.2.2.1: Secuencia de navegación y control de misión por waypoints. Fuente: elaboración propia.

3.2.2.1. Relación entre GUI, Olympe y el ejecutor de misión

Desde el punto de vista de implementación, Olympe sigue siendo la capa que resuelve la comunicación efectiva con el ANAFI Thermal, tanto la GUI como el mission manager dependen de esta biblioteca para abrir conexión, emitir movimientos relativos, controlar cámara y gimbal o registrar información de vuelo. Sin embargo, la ejecución física de la misión no queda concentrada dentro de la página visual. El archivo ros2_ws/src/anafi_suite_core/anafi_suite_core/mission_executor.py encapsula esa lógica en una clase reutilizable, MissionExecutor, utilizada por el nodo mission_manager. Esta clase se encarga de: establecer una conexión exclusiva con el dron, aplicar el pitch del gimbal, enviar desplazamientos con moveBy o extended_move_by, iniciar o detener video, capturar fotografías RGB y almacenar metadatos del vuelo ejecutado.

Este detalle es importante porque cambia el papel de la GUI dentro del proyecto, la interfaz no es simplemente un programa que se conecta al dron y ejecuta de inmediato cada botón; más bien, funciona como estación de planificación y despacho. Cuando el operador presiona Ejecutar, la página Misión de vuelo valida la información, libera la propiedad de la conexión Olympe que pudiera estar retenida por módulos de sensores o bridge, guarda la misión en runtime/current_mission.json y solicita al mission manager que la ejecute. A partir de ese momento, el seguimiento de la misión se realiza por ROS 2 mediante mensajes de estado y progreso, mientras la lógica de movimiento reside fuera del hilo principal de la interfaz.

3.2.2.2. Control de alto nivel, PID y calibración dentro de la plataforma

En este proyecto, la lógica implementada por la GUI y por ROS 2 trabaja sobre órdenes de alto nivel, esto implica que la estabilización rápida del vehículo, la corrección de actitud y los lazos internos equivalentes al comportamiento PID permanecen dentro del piloto embebido del ANAFI Thermal. Metodológicamente, se decidió no reescribir esos lazos ni introducir una capa experimental de control de bajo nivel, en cambio, se conservó la lógica de control por defecto compatible con el ecosistema anafi_ros/anafi_autonomy, de modo que el aporte del trabajo se concentrara en misión, interfaz, captura y trazabilidad de datos.

Como referencia documental del comportamiento de control adoptado durante las pruebas, se tomaron los valores publicados en el archivo params_controller_pid.yaml de la librería anafi_autonomy para ROS 2, cuya parametrización fue replicada como base de ajuste en la plataforma de ensayo [18]. Aunque estos lazos no fueron reescritos dentro de la GUI.

Tabla 3.2.1: Ganancias de referencia documentadas para el controlador PID adoptado desde anafi_autonomy.

Lazo	Parámetro	Valor
Posición	p	2.0
Posición	i	2.0
Posición	d	0.5
Posición	max_i	0.1
Velocidad	p	20.0
Velocidad	i	1.0
Velocidad	d	3.0
Velocidad	max_i	0.5
Guiñada (yaw)	p	70.0

En términos prácticos, esta configuración muestra que el control asociado a velocidad se mantuvo con una ganancia proporcional más alta que la empleada en posición, mientras que la guiñada se documentó con una respuesta proporcional elevada para orientar con rapidez el dron alrededor de su eje vertical, dentro del alcance del proyecto esto no implicó modificar el piloto interno del dron, sino conservar una base de control ya validada para que las pruebas de navegación, captura y supervisión desde la interfaz se ejecutaran sobre un comportamiento consistente del dron.

3.2.3. Resultados

Finalmente, durante la inspección y depuración del sistema aparecen nodos auxiliares del ecosistema ROS, como /rqt_gui_py_node_*, asociados a rqt, y _ros2cli_daemon_*, asociados a utilidades de línea de comandos. Estos nodos no implementan lógica de vuelo, pero sí son importantes para el proceso experimental porque permiten visualizar relaciones entre tópicos, inspeccionar el grafo y verificar si la comunicación entre GUI y ejecutor de misión ocurre de forma correcta.

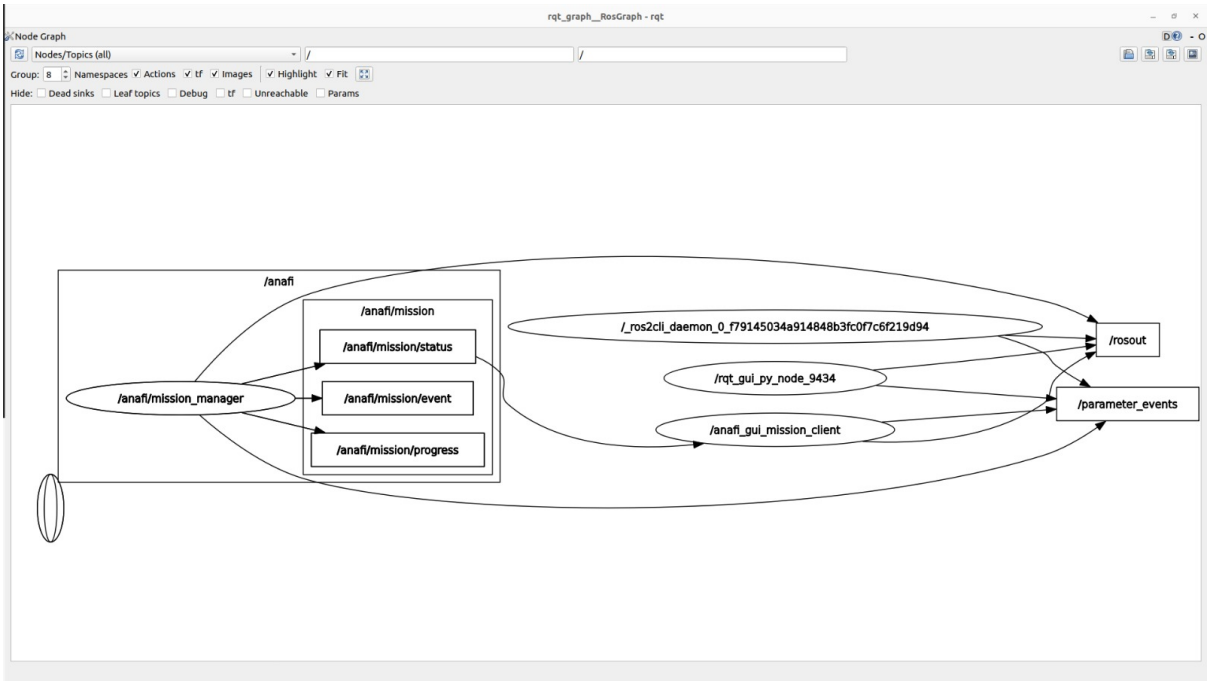


Figura 3.2.3.1: Grafo ROS 2 observado durante la ejecución de la misión. Se distinguen el nodo /anafi/mission_manager, el cliente /anafi_gui_mission_client y los tópicos de estado, evento y progreso dentro del espacio de nombres /anafi/mission. Fuente: elaboración propia.

La Figura 3.2.3.1 confirma esta organización, en ella se observa que el nodo /anafi/mission_manager publica sobre tres tópicos: /anafi/mission/status, /anafi/mission/event y /anafi/mission/progress, el cliente de la GUI consume principalmente el canal de estado para actualizar la barra de estado de la página Misión de vuelo. Aunque los servicios /anafi/mission/ping, /anafi/mission/start y /anafi/mission/stop no aparecen representados en rqt_graph de la misma forma que los tópicos, son parte esencial de la interacción entre ambos nodos: la interfaz llama a ping para verificar disponibilidad, a start para despachar la misión serializada y a stop para solicitar cancelación.

La preconfiguración operativa del sistema, incluyendo las restricciones prácticas de movimiento, estabilización, calibración del magnetómetro, corrección del horizonte y calibración del gimbal, se presenta como material complementario en el Anexo D.

En síntesis, lo expuesto en esta sección permite abordar de los objetivos específicos 1 y 2 del proyecto. Esto debido a que, por una parte, la integración entre ROS 2 y el SDK Parrot Olympe define una arquitectura de software coherente para el control del dron desde una estación base, la coordinación entre procesos y la gestión de la adquisición de datos; y, por otra, la implementación del nodo *mission_manager*, la serialización de misiones y el esquema de navegación por *waypoints* proporcionan una base operativa para ejecutar trayectorias autónomas de forma organizada, repetible y supervisable.

Capítulo 4

Procesamiento térmico y delimitación geométrica de la escena

La visión artificial constituye la capa encargada de transformar la información capturada por el sistema en una base útil para el análisis técnico de la inspección. En este trabajo, su integración no se limita únicamente al procesamiento posterior de imágenes, sino que forma parte de un flujo más amplio que inicia con la adquisición de evidencia RGB y térmica, continúa con la organización de los archivos asociados a cada misión y culmina con la identificación de regiones de interés y la recuperación de información térmica significativa. Desde esta perspectiva, el capítulo reúne los fundamentos conceptuales y las decisiones de implementación que permiten comprender cómo se articula el tratamiento de la imagen dentro del sistema desarrollado.

Aunque en aplicaciones recientes de visión artificial es frecuente recurrir a modelos de detección y segmentación cada vez más complejos, en escenarios de inspección también siguen siendo válidos enfoques más simples, siempre que respondan de forma adecuada a las necesidades del problema. Entre ellos se encuentra la detección por umbrales mediante máscaras, en la cual se seleccionan rangos de valores sobre determinados canales de color o intensidad para resaltar regiones con un comportamiento visual o térmico particular. Este tipo de estrategia, lejos de ser un recurso elemental sin valor técnico, puede resultar conveniente cuando se requiere bajo costo computacional, alta interpretabilidad y control directo por parte del operador sobre los criterios de detección.

4.1. Fundamentos teórico

El tratamiento de escenas térmicas y RGB dentro de un sistema de inspección puede abordarse desde distintos niveles de complejidad. En una primera aproximación, la detección de objetos suele realizarse mediante cajas rectangulares convencionales, adecuadas cuando la orientación del elemento a localizar no tiene un efecto importante sobre la decisión posterior. Sin embargo, cuando la posición angular del objeto sí altera la forma en que se delimita la escena o se estima una región de interés, resultan más convenientes estrategias basadas en *Oriented Bounding Boxes* (OBB). Este tipo de representación no solo indica la localización del objeto, sino también su orientación dentro del plano de la imagen, lo cual mejora la consistencia geométrica del recorte y reduce errores asociados a cajas excesivamente grandes o desalineadas. En el contexto de esta tesis, dicha característica adquiere relevancia debido a que los *targets* térmicos pueden ser observados con cambios de perspectiva, inclinación o rotación durante la

captura aérea. Frente a detectores tradicionales de cajas horizontales, el uso de OBB aporta una referencia más precisa para ordenar centroides, estimar homografías y construir una escena corregida con menor incertidumbre geométrica.

No obstante, la detección basada en OBB no representa la única alternativa dentro del campo de la visión artificial. Existen también detectores convencionales de objetos, segmentación semántica, segmentación por instancias y métodos clásicos apoyados en umbrales, bordes, textura o transformaciones geométricas. La elección entre uno u otro enfoque depende del equilibrio requerido entre precisión, interpretabilidad, costo computacional y facilidad de implementación. En aplicaciones donde interesa aislar visualmente una zona caliente o una anomalía evidente dentro de una escena previamente acotada, la segmentación por umbrales puede ofrecer una solución práctica y robusta. En cambio, cuando se requiere reconocer estructuras complejas, separar múltiples clases simultáneamente o adaptarse a escenarios altamente variables, suele ser necesario incorporar modelos más elaborados. Por ello, dentro del desarrollo propuesto se combinan ambos enfoques: un detector orientado para establecer la geometría de la escena y una segmentación más simple para resaltar regiones térmicas de interés sobre una visual mapeada.

La Figura 4.1.1 ilustra la diferencia entre una detección convencional mediante cajas rectangulares y una detección basada en OBB. Esta comparación permite apreciar por qué, en escenas donde la orientación de los objetos condiciona el recorte geométrico posterior, la caja orientada ofrece una representación más consistente para establecer referencias dentro de la imagen.



Figura 4.1.1: Comparación entre detección de objetos tradicional mediante cajas rectangulares y detección mediante *Oriented Bounding Boxes* (OBB), donde la orientación del objeto se considera durante la localización. Fuente: adaptado de la documentación oficial de Ultralytics [1].

Desde el punto de vista de la representación digital, una imagen puede entenderse como una matriz cuyos elementos almacenan valores asociados a intensidad, color o radiometría. Archivos de mapa de bits ampliamente difundidos, como JPG o PNG, permiten representar la información visual mediante arreglos bidimensionales o tridimensionales que luego pueden descomponerse en canales. En el caso de una imagen RGB, por ejemplo, es posible separar su contenido en tres matrices correspondientes a los componentes rojo, verde y azul. Esta descomposición resulta útil para aplicar operaciones independientes sobre cada canal, definir máscaras, construir falsos colores o convertir la imagen a otros espacios como HSV o LAB. Aun cuando formatos como JPG son adecuados para visualización y almacenamiento ligero, su naturaleza comprimida puede introducir pérdidas en la información. Por esta razón, en aplicaciones técnicas no

se consideran suficientes para preservar por sí solos el dato radiométrico original.

En contraste, archivos de naturaleza más cercana al dato crudo, como DNG u otros formatos radiométricos, conservan una estructura menos orientada a la visualización final y más vinculada al valor medido por el sensor. Este tipo de archivos puede almacenar matrices de mayor profundidad y menor compresión, lo que los hace apropiados para análisis térmico o fotométrico posterior. Además, la imagen visible no siempre concentra toda la información relevante de una captura: es común que el sistema de adquisición genere archivos auxiliares o metadatos asociados que complementan la imagen principal, por ejemplo parámetros de captura, escalas, rangos de temperatura, georreferencias o datos radiométricos exportados en formatos independientes. En esa lógica, un archivo JPG puede formar parte del conjunto de salida de una captura, pero el contenido técnico de interés puede residir en matrices externas o metadatos vinculados, y no únicamente en la imagen visualizada.

El concepto de dato *raw* se refiere, precisamente, a una medida todavía cercana a la respuesta original del sensor. En cámaras térmicas, estos valores no equivalen directamente a una temperatura interpretable en grados Celsius o Kelvin, sino a una magnitud que debe transformarse mediante una relación de calibración. Dependiendo del fabricante, del tipo de sensor y del flujo de exportación, la conversión desde el valor *raw* hasta un valor funcional puede involucrar parámetros de ganancia, *offset*, correcciones internas, emisividad, compensación atmosférica o ecuaciones específicas del dispositivo. Esto implica que la imagen mapeada a color no debe confundirse con la temperatura real, ya que el color representa solo una visualización perceptual. La estimación térmica confiable debe recuperarse siempre desde la matriz radiométrica original y a partir del procedimiento de conversión correspondiente.

En términos de implementación, el manejo de estos datos requiere formatos que permitan conservar matrices y metadatos de forma eficiente. Dentro del ecosistema de Python, los archivos `.npy` resultan especialmente útiles porque almacenan arreglos numéricos de `NumPy` sin necesidad de reconstruir manualmente su estructura en cada carga. Esto facilita trabajar con matrices radiométricas, máscaras binarias, regiones rectificadas y otras representaciones intermedias del procesamiento. Por su parte, los archivos `.json` ofrecen una alternativa ligera, legible y ampliamente compatible para almacenar metadatos, configuraciones, resultados parciales, parámetros de una misión o información descriptiva asociada a una captura. Su principal ventaja radica en la facilidad con la que pueden intercambiarse entre distintos módulos del sistema, lenguajes de programación o herramientas externas. En consecuencia, la combinación de una imagen visual, una matriz numérica y un archivo de metadatos constituye una estrategia coherente para mantener separadas la capa perceptual, la capa radiométrica y la capa descriptiva del sistema.

La resolución espacial alcanzable durante una captura aérea no depende únicamente de la resolución nominal del sensor, sino también de la relación geométrica existente entre la cámara y la escena observada. Uno de los parámetros más utilizados para cuantificar esta relación es el *Ground Sample Distance* (GSD), definido como la dimensión real del terreno representada por cada píxel de la imagen. Este parámetro constituye una referencia fundamental para determinar el nivel de detalle que puede capturarse durante una inspección térmica, ya que establece el tamaño mínimo observable de un objeto o anomalía sobre la superficie analizada.

Desde una perspectiva geométrica, el GSD puede estimarse a partir de la altura de vuelo, el tamaño físico del píxel del sensor y la distancia focal efectiva de la óptica

utilizada. Bajo esta aproximación, la relación puede expresarse mediante la ecuación:

$$GSD = \frac{H \cdot p}{f} \tag{4.1.1}$$

donde H corresponde a la altura entre la cámara y la superficie observada, p representa el tamaño físico de cada píxel del sensor y f la distancia focal efectiva de la lente. La ecuación muestra que el GSD aumenta proporcionalmente con la altura de vuelo y disminuye al emplear ópticas de mayor distancia focal o sensores con menor tamaño de píxel. En consecuencia, para una misma cámara, el incremento de la altitud implica una reducción progresiva del detalle espacial disponible en la imagen.

En el caso específico del sistema utilizado en esta investigación, el sensor FLIR Lepton 3.5 integrado en el *Parrot ANAFI Thermal* dispone de una resolución térmica de 160×120 píxeles, un tamaño de píxel de aproximadamente $12, \mu m$ y una distancia focal efectiva cercana a 1.93 mm [10]. A partir de estos parámetros es posible estimar el GSD teórico alcanzable para distintas alturas de operación, permitiendo relacionar directamente la planificación de vuelo con la capacidad de detección de anomalías térmicas.

La Tabla 4.1.1 presenta los valores de GSD calculados para diferentes alturas de vuelo considerando las especificaciones del sensor empleado. Estos resultados permiten apreciar cómo pequeñas variaciones en la altitud generan cambios significativos en la resolución espacial disponible, aspecto particularmente relevante cuando se pretende detectar objetivos de dimensiones reducidas o realizar comparaciones térmicas entre regiones específicas de una escena.

Tabla 4.1.1: GSD teórico calculado para distintas alturas de vuelo utilizando el sensor FLIR Lepton 3.5.

Altura de vuelo (m)	**GSD (cm/píxel)**
3	1.87
5	3.11
7	4.35
10	6.22
20	12.44

Los valores obtenidos permiten relacionar directamente las capacidades del sensor FLIR Lepton 3.5 con recomendaciones ampliamente utilizadas en termografía aérea. Diversas guías técnicas señalan que la confiabilidad de una medición térmica depende de la cantidad de píxeles que representan el objetivo observado, recomendándose que los elementos de interés ocupen varios píxeles dentro de la imagen para garantizar una interpretación adecuada de la información radiométrica. Asimismo, para aplicaciones de inspección de detalle suelen requerirse resoluciones espaciales reducidas, asociadas a valores de GSD del orden de pocos centímetros por píxel [19].

Al comparar estas recomendaciones con los valores calculados para el sensor empleado, se observa que alturas de vuelo comprendidas aproximadamente entre 5 y 10 m producen GSD entre 3.11 y 6.22 cm/píxel. Considerando que el objetivo de esta investigación consiste en identificar y caracterizar anomalías térmicas localizadas, resulta conveniente operar dentro de este rango de alturas, ya que permite mantener un equilibrio entre cobertura de terreno y capacidad de discriminación espacial. Aunque aplicaciones de inspección general pueden admitir valores de GSD superiores, la detección de detalles térmicos de menor tamaño requiere resoluciones espaciales más exigentes, favoreciendo

alturas de vuelo moderadas para conservar una representación suficiente del objetivo dentro de la imagen.

4.2. Desarrollo propuesto

El desarrollo de la capa de visión artificial se organizó como una secuencia progresiva orientada a conservar trazabilidad radiométrica, simplificar la interacción con el operador y mantener coherencia entre la geometría de la escena y la interpretación térmica posterior. En lugar de plantear un análisis monolítico sobre la captura completa, se definió una estructura de procesamiento en etapas, donde primero se estabiliza la región de trabajo y luego se realiza la búsqueda de comportamientos térmicos anómalos sobre una visual que favorece la interpretación. Esta forma de organizar el procesamiento no solo reduce el costo computacional, sino que también permite que cada resultado intermedio quede documentado y pueda reutilizarse en etapas posteriores del sistema.

4.2.1. Delimitación geométrica y análisis térmico de la escena

La primera parte del flujo se centra en establecer una región de interés homogénea a partir de referencias visibles dentro de la captura. Para ello, se emplean *targets* térmicos cuya detección permite reconocer una geometría conocida y, a partir de ella, definir un recorte controlado de la escena. Esta lógica puede apreciarse en la Figura 4.2.1.1, donde se presenta el *setup* propuesto para esta etapa y la disposición de los elementos que permiten estabilizar geométricamente la escena antes del análisis térmico. Esta decisión responde a la necesidad de evitar que el análisis térmico se vea afectado por zonas periféricas sin utilidad diagnóstica, variaciones de perspectiva o fragmentos del entorno que no pertenecen al arreglo fotovoltaico. Una vez identificadas estas referencias, se calculan sus centroides, se ordenan espacialmente y se utiliza dicha información para construir una transformación geométrica que corrige inclinaciones y genera una vista rectificada del área bajo inspección.

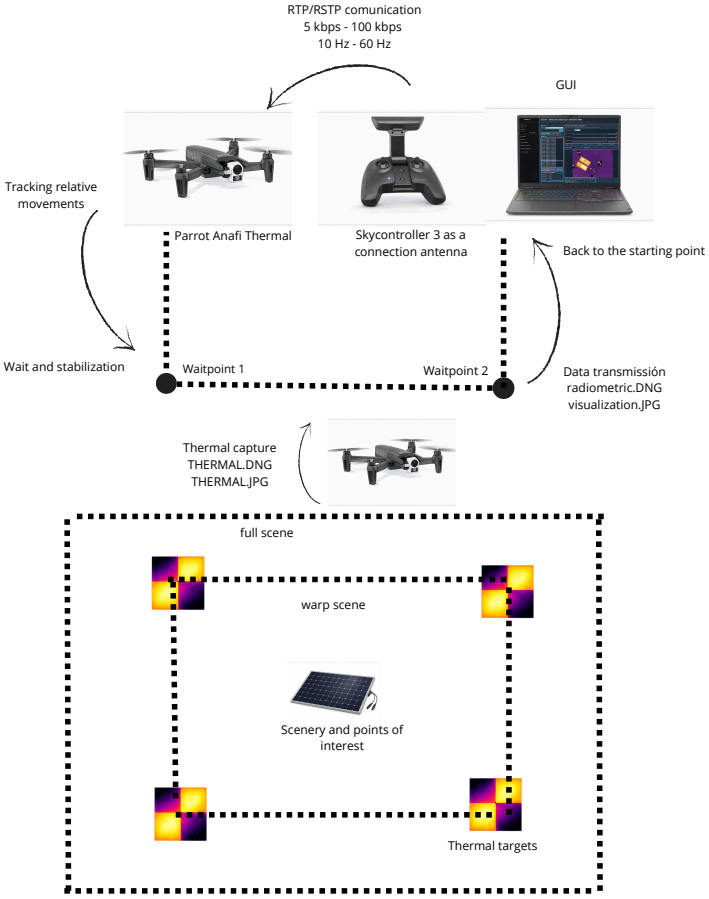


Figura 4.2.1.1: *Setup* propuesto para la delimitación geométrica de la escena y el desarrollo de la Sección 4.2.1. Fuente: elaboración propia.

La conveniencia de trabajar sobre una escena previamente delimitada se manifiesta en varios niveles. En primer lugar, disminuye la cantidad de píxeles a procesar, con lo cual se reduce el tiempo de cálculo y se facilita una respuesta más ágil en la interfaz. En segundo lugar, permite comparar con mayor consistencia distintas capturas pertenecientes a una misma misión, dado que todas quedan reducidas a un marco geométrico semejante. En tercer lugar, mantiene una referencia térmica y espacial dentro del recorte, lo que mejora la repetibilidad del procedimiento durante las pruebas. Como guía para el diseño de estos elementos de referencia se consideraron antecedentes reportados en [11], donde se describen superficies de control térmicamente detectables empleadas para facilitar su identificación en imágenes adquiridas desde UAV.



Figura 4.2.1.2: Targets térmicos implementados como referencia geométrica y térmica dentro de la escena evaluada. Fuente: elaboración propia.

4.2.2. Detección mediante umbrales usando sliders

Con la escena ya rectificada, el procesamiento continúa mediante una estrategia de segmentación por umbrales ajustables sobre la imagen de visualización. Esta decisión se adoptó por su sencillez de implementación, su bajo costo computacional y su facilidad de interpretación para el usuario. En lugar de delegar toda la identificación de anomalías a modelos más complejos, el operador puede seleccionar intervalos mínimos y máximos sobre los canales del espacio de color escogido para resaltar zonas térmicamente atípicas. El uso de máscaras sobre espacios como RGB, HSV o LAB hace posible distinguir visualmente regiones calientes o irregulares con una lógica clara y controlable. Posteriormente, operaciones morfológicas como *blur*, erosión, dilatación y cierre se emplean para limpiar la máscara obtenida y conservar únicamente regiones con tamaño y forma coherentes para el análisis.



Figura 4.2.2.1: Interfaz de ajuste de umbrales mediante *trackbars* para la detección de regiones térmicas de interés. Fuente: elaboración propia.

Es importante señalar que esta detección se realiza sobre la imagen de visualización y sobre sus matrices de *colorspace*, no directamente sobre la radiometría cruda. Esta separación resulta ventajosa desde el punto de vista operativo, ya que el usuario puede interactuar con una representación perceptualmente más intuitiva, mientras que el sistema conserva intacta la posibilidad de regresar al dato físico original. Dado que el mapeo a color no es uniforme ni lineal en magnitudes térmicas, el valor real de temperatura de la región detectada debe recuperarse nuevamente desde la matriz radiométrica original de 16 bits y su correspondiente conversión. De esta forma, la etapa visual cumple la función de localizar candidatos y la etapa radiométrica la función de cuantificar el fenómeno térmico con validez técnica.

Como resultado de este flujo, el sistema genera productos intermedios y finales que incluyen una visual térmica recortada, matrices numéricas asociadas a temperatura y archivos descriptivos con metadatos del análisis. Entre ellos, archivos como `ROI_warp.jpg` y `ROI_warp_tempC.npy` representan, respectivamente, la imagen térmica rectificada para visualización y la matriz térmica correspondiente en grados Celsius.

4.2.3. Integración operativa y visualización dentro del sistema

La segunda parte del desarrollo se orienta a incorporar este flujo dentro de una interfaz gráfica de usuario que articule configuración, monitoreo y análisis. En este contexto, la GUI no se concibe como un componente accesorio, sino como el entorno operativo desde el cual se administra la misión, se observan las capturas y se presentan los resultados derivados del procesamiento. Su implementación en Python, apoyada en herramientas como Qt, OpenCV, Pillow, *ffmpeg* y utilidades del sistema, permite consolidar en un mismo entorno funciones de visualización, parametrización y exportación de resultados.

Desde la interfaz es posible mostrar telemetría básica, imágenes o secuencias de captura y una vista preliminar de la escena térmica con distintos *colorspaces* orientados exclusivamente a la visualización. Esta decisión facilita que el operador realice una

inspección exploratoria rápida antes de profundizar en el análisis radiométrico. A su vez, la posibilidad de exportar información en formatos como CSV, PDF o JSON contribuye a mantener trazabilidad y a documentar los hallazgos de forma compatible con etapas posteriores de revisión. La incorporación de esta capa de interacción establece de manera natural el vínculo con el capítulo siguiente, en el cual se detalla la arquitectura, organización y comportamiento de la interfaz desarrollada, cuyo flujo final se representa en la Figura 4.2.3.1, la cual muestra el setup sobre el cual se ejecutaron las pruebas finales y la comprobación integral del sistema.

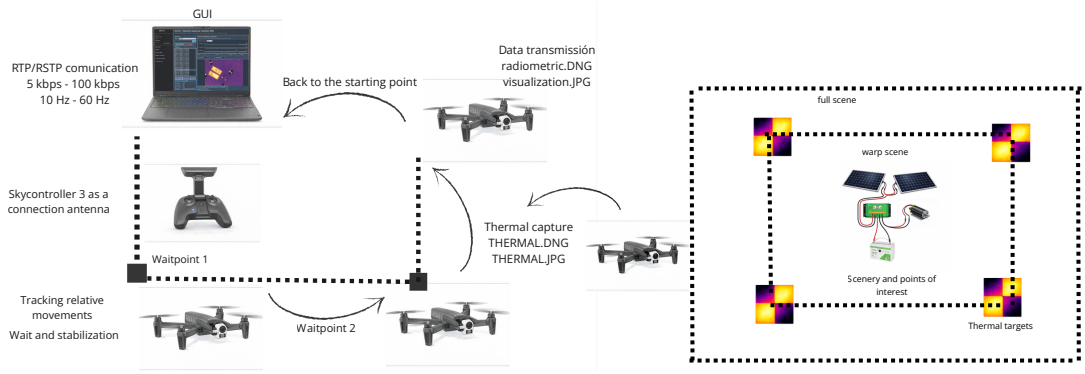


Figura 4.2.3.1: Escena real (*setup* final) utilizada durante las pruebas finales y la comprobación del sistema. Fuente: elaboración propia.

Con el fin de validar el flujo de captura térmica y análisis propuesto en este trabajo, se dispuso una escena de prueba basada en un sistema fotovoltaico a pequeña escala. Esta escena fue concebida como un montaje experimental controlado, suficiente para verificar el comportamiento general del sistema desarrollado y facilitar la aparición de diferencias térmicas observables dentro del arreglo evaluado. Como se presenta en el plano del Anexo B (Figura 4.2.3.2), la escena está compuesta por dos paneles solares conectados en paralelo, un regulador de carga, una batería y un motor DC empleado como carga del sistema. La presencia de esta carga permite que exista consumo eléctrico dentro del montaje, lo cual resulta conveniente para provocar un funcionamiento más representativo del arreglo y favorecer la observación de variaciones térmicas o puntos calientes sobre los módulos durante la inspección, cabe recalcar que la configuración utilizada no busca reproducir la complejidad de una planta fotovoltaica real, sino disponer de una escena instrumentada y controlada que permita evaluar de manera práctica la captura térmica con el dron, la delimitación del área de interés y la interpretación básica de anomalías térmicas dentro de los paneles observados.

Como complemento al cumplimiento normativo descrito, el RAC 100 establece en su artículo 100.315 que las operaciones en categoría abierta con fines recreativos no requieren licencia ni autorización especial ante la Aerocivil, siempre que la aeronave se mantenga dentro de una distancia horizontal máxima de 750 metros respecto al operador y se conserve en todo momento la línea de vista visual directa (VLOS) [20]. En este marco, el dron Parrot ANAFI Thermal empleado en el proyecto opera bajo categoría abierta y no requirió ningún tipo de certificación adicional para su uso. Cabe destacar que, según las especificaciones técnicas del fabricante, el ANAFI Thermal con control SkyController 3 dispone de una distancia máxima de transmisión de hasta 4 km [8]; sin embargo, este alcance no pudo ser demostrado experimentalmente durante las pruebas, dado que las dimensiones físicas del campus universitario constituyeron la

principal limitación operativa. En consecuencia, la distancia horizontal máxima demostrada durante los vuelos fue de 120 metros, y la distancia vertical máxima alcanzada fue de 60 metros, ambas muy por debajo del límite normativo de 750 m y del alcance técnico del equipo, pero suficientes para validar de forma íntegra la funcionalidad del sistema de inspección térmica en las condiciones reales del entorno de pruebas.

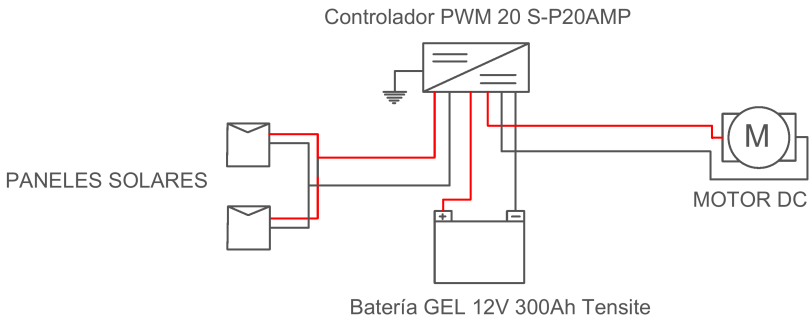


Figura 4.2.3.2: Plano general de la escena evaluada para las pruebas del sistema fotovoltaico, correspondiente al Anexo B. Fuente: elaboración propia.

4.3. Análisis de resultados

Los resultados asociados a la capa de visión artificial se presentan en dos niveles complementarios. Por una parte, se evalúa el desempeño del detector empleado para localizar los *targets* térmicos de referencia, ya que de esta etapa depende la consistencia geométrica del recorte posterior. Por otra, se muestran evidencias del comportamiento del flujo de segmentación y recuperación térmica dentro de la interfaz, con el fin de ilustrar cómo la detección visual se convierte finalmente en una medición térmica interpretable para el operador.

Para sustentar el desempeño reportado en las métricas, resulta pertinente describir el proceso de construcción del conjunto de datos y la configuración empleada durante el entrenamiento. El conjunto base estuvo conformado por 679 imágenes adquiridas con anterioridad al desarrollo de la interfaz definitiva, capturadas a lo largo de un rango de alturas que abarcó desde la distancia mínima necesaria para la identificación visual de los targets hasta aproximadamente 50 metros, con variaciones deliberadas en la posición de los elementos de referencia, el ángulo del gimbal, cantidad de targets en la escena y las condiciones ambientales del entorno. La adquisición fue un proceso progresivo: se inició empleando la aplicación nativa del dron, se continuó con versiones tempranas de la interfaz que aún no disponían de integración completa con los nodos de ROS2 y solo permitían la captura de la imagen térmica sin asociar la matriz radiométrica correspondiente, y se finalizó con la interfaz en su estado completo. Este carácter gradual fue intencional, ya que garantizar diversidad real en las condiciones de captura exigió avanzar de forma pausada, asegurando que cada sesión aportara variabilidad útil al conjunto. Las imágenes resultantes fueron etiquetadas manualmente mediante

Roboflow, plataforma ampliamente reconocida en la comunidad de visión artificial por sus herramientas de anotación y por integrar un módulo de aumento de datos que permitió aplicar transformaciones sobre parámetros propios del sensado fotográfico, tales como brillo, contraste, saturación, desenfoque y ruido, entre otros, expandiendo el conjunto hasta alcanzar un total de 1154 imágenes distribuidas en subconjuntos de entrenamiento, validación y prueba. El entrenamiento se ejecutó durante 100 épocas. El Anexo D contiene el registro detallado de la evolución del proceso de entrenamiento del modelo a lo largo de todas las épocas ejecutadas. En este archivo se encuentran las métricas asociadas al desempeño del modelo durante las fases de entrenamiento y validación, incluyendo las funciones de pérdida empleadas para evaluar la localización y clasificación de los objetos, así como indicadores de rendimiento utilizados para medir la capacidad de detección. Adicionalmente, se registran parámetros relacionados con el proceso de optimización, tales como la tasa de aprendizaje aplicada en cada etapa. Este entrenamiento fue realizado sobre la plataforma Jetson Orin Nano facilitada por la universidad, lo que permitió disponer de un entorno con suficiente capacidad de cómputo, continuidad en la ejecución y un uso eficiente de los recursos disponibles. Las prestaciones generales de este equipo pueden observarse en la Tabla 4.3.1.

Tabla 4.3.1: Especificaciones técnicas de la plataforma NVIDIA Jetson Orin Nano utilizada durante el desarrollo y validación del sistema.

Característica	**Especificación**
Modelo	NVIDIA Jetson Orin Nano 8 GB Developer Kit
CPU	6 núcleos ARM Cortex-A78AE de 64 bits a 1.7 GHz
GPU	NVIDIA Ampere con 1024 núcleos CUDA y 32 Tensor Cores
Memoria RAM	8 GB LPDDR5
Ancho de banda de memoria	68 GB/s
Rendimiento de IA	Hasta 67 TOPS
Almacenamiento	Tarjeta microSD y soporte para SSD NVMe mediante interfaz M.2
Interfaces de cámara	2 interfaces MIPI CSI-2
Conectividad	1 puerto Gigabit Ethernet
Puertos USB	4 puertos USB 3.2 Gen 2
Salida de video	DisplayPort 1.2
Expansión	Cabecera GPIO de 40 pines compatible con periféricos embebidos
Consumo energético	Configurable entre 7 W y 15 W
Sistema operativo	Ubuntu Linux mediante NVIDIA JetPack SDK

La configuración empleada utilizó el modelo base `yolov8n-obb.pt` como punto de partida, con un tamaño de imagen de entrada de 640×640 píxeles, un tamaño de lote (*batch size*) de 16 y una tasa de aprendizaje inicial de 1×10^{-3}. La configuración asignada para el entrenamiento se resume en la Tabla 4.3.2.

Tabla 4.3.2: Configuración general empleada para el entrenamiento del modelo de detección OBB.

Parámetro	**Valor o descripción**
Modelo base	`yolov8n-obb.pt`
Tipo de detección	Detección de objetos mediante *Oriented Bounding Boxes* (OBB).
Número de clases	1 clase (*Target térmico*).
Número total de imágenes	1154 imágenes.
Distribución del conjunto de datos	80 % entrenamiento, 15 % validación y 5 % prueba.
Número de épocas	100 épocas de entrenamiento.
Tamaño de imagen	640×640 píxeles.
Tamaño de lote	16 imágenes (*batch size*).
Tasa de aprendizaje inicial	1×10^{-3}.
Optimizador	`SGD`.
Inicialización de pesos	Pesos preentrenados `yolov8n-obb.pt`.

El seguimiento del entrenamiento época por época, documentado en el Anexo D, muestra cómo las pérdidas decrecen de forma sostenida durante las primeras etapas y cómo, hacia las épocas finales, comienzan a aparecer indicios de sobreajuste sobre el conjunto de entrenamiento. El punto de corte adoptado responde precisamente a este criterio: detener el proceso en una época en que el modelo ya exhibe métricas de validación sólidas y estables, sin comprometer recursos computacionales adicionales ni arriesgar una degradación en la capacidad de generalización. Con ello, el modelo resultante queda listo para integrarse al flujo de detección y análisis térmico con un comportamiento confiable y trazable dentro del contexto experimental del proyecto.

En la etapa de detección de *targets*, el desempeño del modelo se evaluó mediante métricas estándar en detección de objetos: precisión (*Precision*), exhaustividad (*Recall*) y media de precisión promedio (*mAP*). La precisión expresa qué proporción de las detecciones realizadas por el modelo corresponde efectivamente a objetos correctos, mientras que el *recall* indica qué proporción de los objetos presentes en las imágenes fue detectada. A su vez, la métrica *mAP* (*mean Average Precision*) resume el desempeño global del detector considerando simultáneamente la capacidad de clasificación y la calidad de localización del *bounding box*. En términos generales, para cada clase se calcula el *Average Precision* (AP) integrando el área bajo la curva Precision–Recall, y luego el *mAP* corresponde al promedio de los valores AP entre clases.

La Tabla 4.3.3 presenta los valores globales obtenidos al final del entrenamiento. En conjunto, estos resultados evidencian un comportamiento alto y estable del modelo, con valores elevados tanto en precisión como en *recall*, así como en *mAP*. Para el contexto controlado de esta tesis, dicho desempeño resulta consistente con la diferenciación visual de las clases y con la necesidad de contar con una referencia geométrica confiable para el recorte de la escena.

Tabla 4.3.3: Métricas globales del modelo YOLOv8 al final del entrenamiento.

Métrica	**Valor**
Precision (B)	0.9968
Recall (B)	0.9983
mAP50 (B)	0.9950
mAP50-95 (B)	0.9808

La Figura 4.3.1 presenta una evidencia representativa del funcionamiento del algoritmo OBB entrenado sobre una escena real de inspección. En esta imagen se puede apreciar cómo la detección de los *targets* térmicos no solo permite localizar los elementos de referencia dentro de la captura, sino también estimar su orientación espacial, lo que hace posible definir con mayor precisión el recorte del área útil de análisis. Este resultado es relevante porque muestra una respuesta positiva del detector en condiciones reales, donde la escena puede presentar variaciones de perspectiva, inclinación o distribución no perfectamente alineada. De manera complementaria, la misma figura permite contrastar la captura original con la región ya recortada, evidenciando la utilidad práctica de esta etapa dentro del flujo propuesto: mientras la imagen completa conserva información del entorno que no aporta al diagnóstico, la versión recortada concentra el análisis sobre la zona de interés, facilitando la rectificación geométrica y el procesamiento térmico posterior.

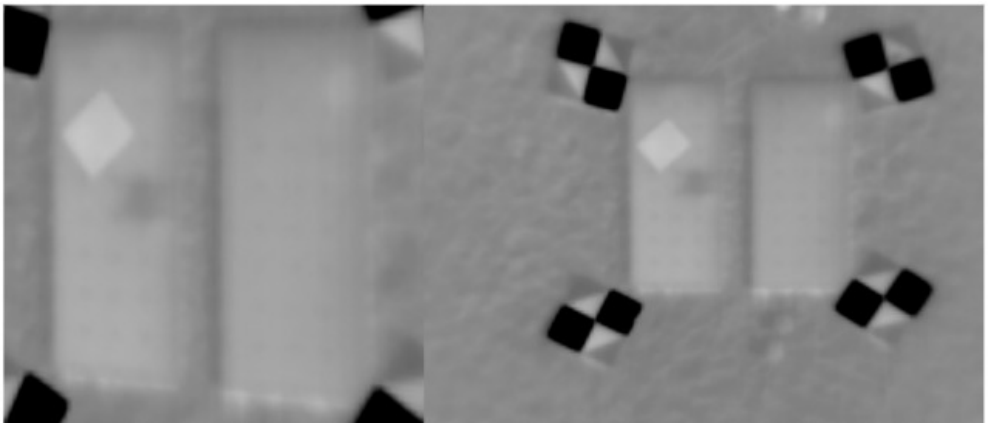


Figura 4.3.1: Comparación entre la captura original y la escena recortada a partir de la detección de *targets* térmicos mediante el algoritmo OBB entrenado. Fuente: elaboración propia.

4.3.1. Análisis de curvas de entrenamiento, curvas de confianza y matriz de confusión

La Figura 4.3.1.1 resume el comportamiento del entrenamiento y la validación a lo largo de las épocas. En ella se observan, de manera general, las pérdidas asociadas a la localización y la clasificación, las cuales presentan una tendencia decreciente, así como el aumento progresivo de las métricas de validación hasta alcanzar una zona de estabilización. En este proyecto, al tratarse de clases con diferencias visuales marcadas, era esperable una convergencia acompañada de métricas elevadas.

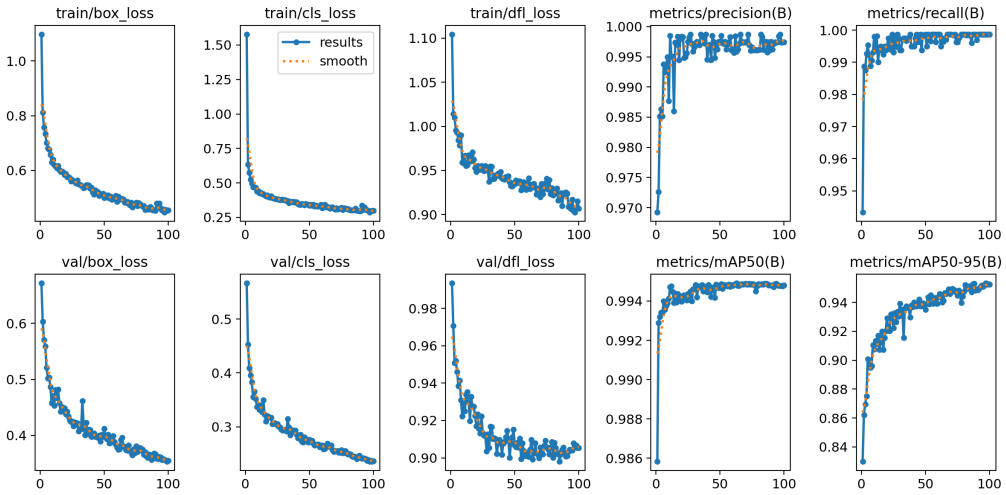


Figura 4.3.1.1: Curvas generales de entrenamiento y validación del modelo a lo largo de las épocas. Fuente: elaboración propia.

La Figura 4.3.1.2 muestra el comportamiento del *F1-score* en función del umbral de confianza. Dado que el *F1-score* es la media armónica entre precisión y *recall*, esta curva resulta útil para identificar un punto de operación que mantenga un equilibrio razonable entre falsos positivos y falsos negativos. En este trabajo, el umbral operativo se definió considerando dicho balance, fijando un umbral de confianza del 87 % ($conf \geq 0{,}87$) para aceptar detecciones.

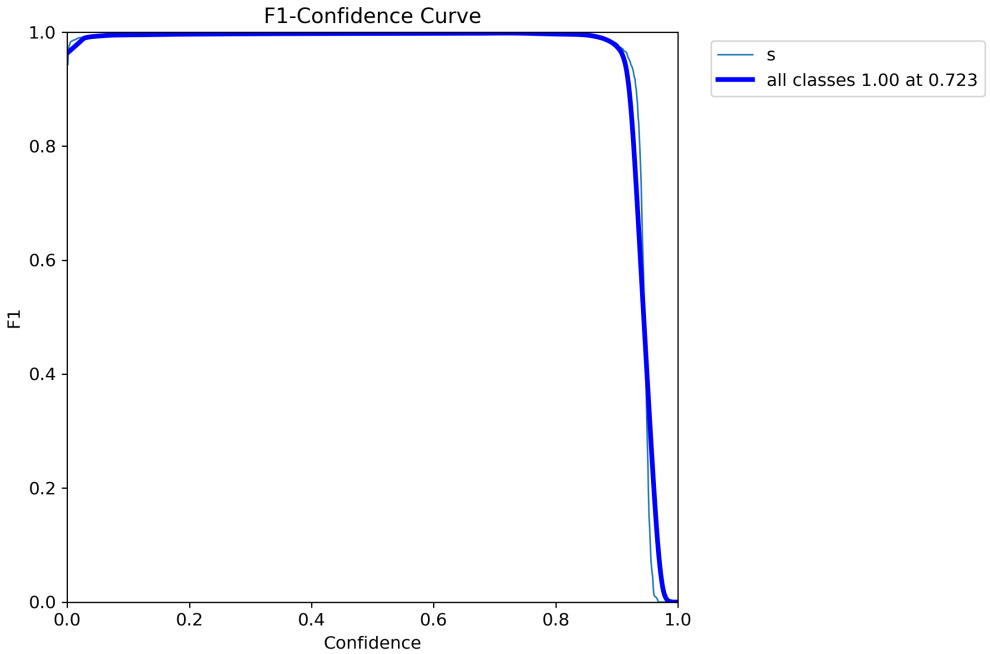


Figura 4.3.1.2: Curva F1-score en función del umbral de confianza del detector. Fuente: elaboración propia.

La Figura 4.3.1.3 presenta la relación entre precisión y *recall* para distintos umbrales de confianza. Un comportamiento favorable se refleja en una curva cercana a la esquina superior derecha, lo que sugiere que el detector mantiene un equilibrio sólido entre la

proporción de aciertos y la cobertura de objetos presentes. En el modelo entrenado, la forma de la curva es coherente con el alto desempeño global reportado en las métricas finales.

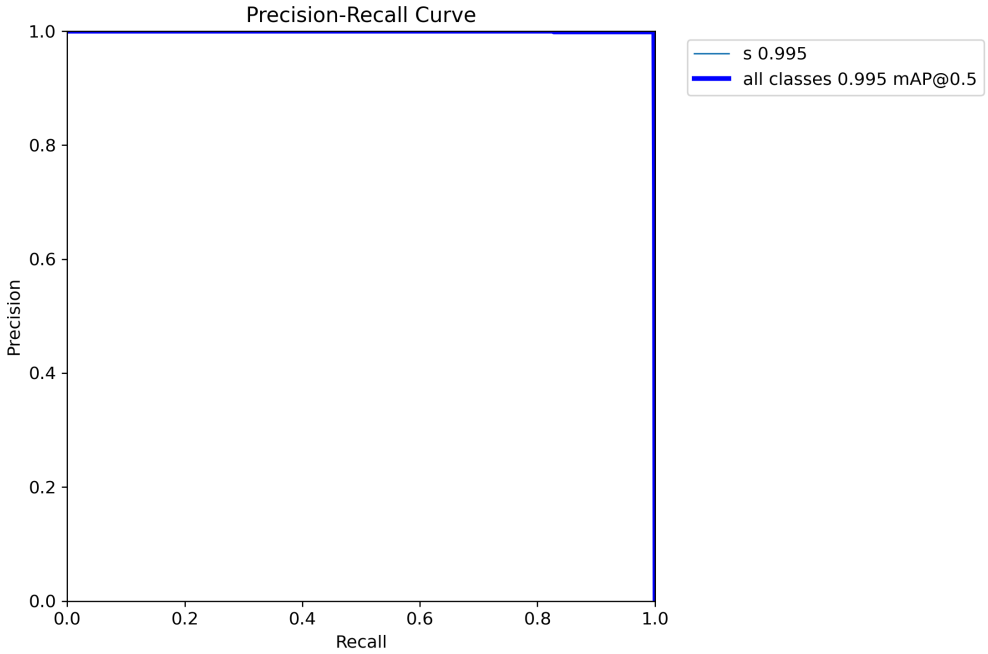


Figura 4.3.1.3: Curva Precision–Recall obtenida durante la evaluación del modelo. Fuente: elaboración propia.

La Figura 4.3.1.4 muestra la matriz de confusión normalizada. Esta representación permite observar cómo se distribuyen las predicciones del modelo y verificar si la mayor parte de los resultados se concentra en la diagonal principal, como sería esperable en un clasificador con baja confusión entre clases. En este caso, el comportamiento observado no se asocia únicamente a un escenario rígidamente controlado, sino también a que el proceso de entrenamiento incluyó variaciones relevantes en las condiciones de captura, tales como cambios de iluminación, presencia de sombras, diferencias de contraste, giros en la orientación de los *targets*, variaciones en la distancia de observación y cambios en la altura de captura. La inclusión de estas condiciones buscó dotar al modelo de mayor solidez frente a perturbaciones razonables del entorno y mejorar su capacidad de generalización dentro del contexto experimental planteado.

Aun así, el hecho de obtener una matriz de confusión casi perfecta y métricas cercanas a la unidad permite considerar la posibilidad de un cierto grado de sobreajuste. Sin embargo, esta situación no resulta inesperada ni invalida los resultados obtenidos, dado que durante el entrenamiento no se contaba con una referencia previa suficientemente clara sobre el número mínimo de épocas necesarias para alcanzar convergencia en este problema particular. Por esta razón, se optó por un entrenamiento suficientemente prolongado, privilegiando la estabilidad del aprendizaje y la certeza de obtener un modelo con desempeño robusto sobre la tarea propuesta. En consecuencia, los resultados alcanzados pueden interpretarse como consistentes con un conjunto de clases visualmente distinguibles, una estrategia de entrenamiento orientada a la robustez y una etapa de validación cuyo propósito principal fue garantizar el correcto funcionamiento del sistema de detección antes de su integración completa con el flujo de

percepción y análisis térmico.

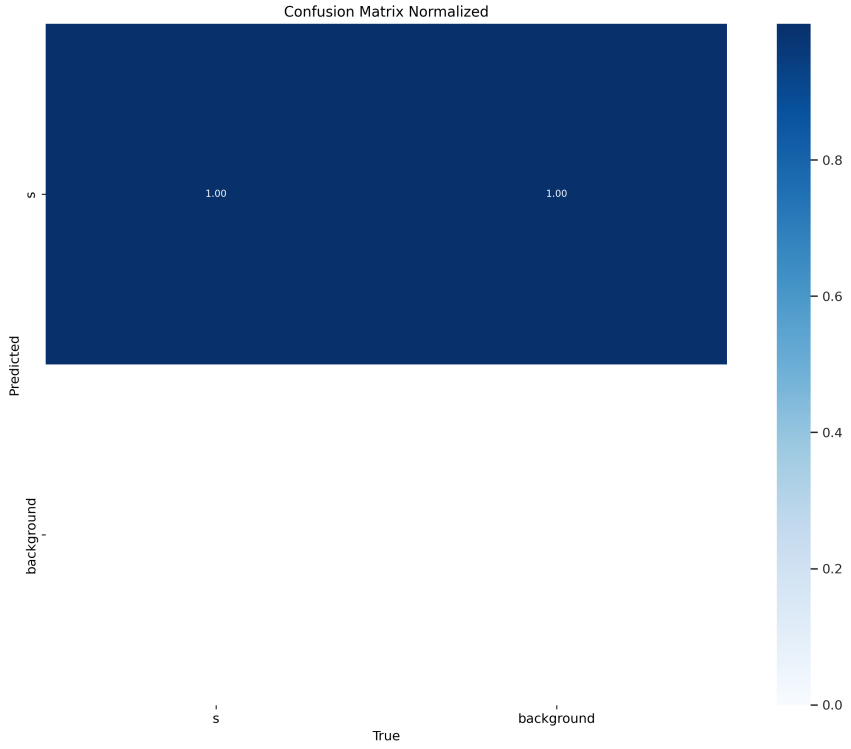


Figura 4.3.1.4: Matriz de confusión normalizada del modelo entrenado. Fuente: elaboración propia.

Además del desempeño alcanzado en la detección de *targets*, resulta igualmente importante evidenciar el comportamiento del flujo de análisis térmico una vez este se integra dentro de la interfaz del sistema. La Figura 4.3.1.5 muestra un ejemplo representativo en el que se aplica la segmentación por umbrales sobre la visual térmica y se logra resaltar una región asociada a un punto caliente dentro del área evaluada. Esta evidencia permite observar cómo la detección visual realizada sobre la imagen mapeada se articula con la información numérica presentada en la GUI, de modo que el operador no solo identifica de forma intuitiva la anomalía térmica, sino que también dispone de una referencia cuantitativa obtenida a partir de la matriz radiométrica original.

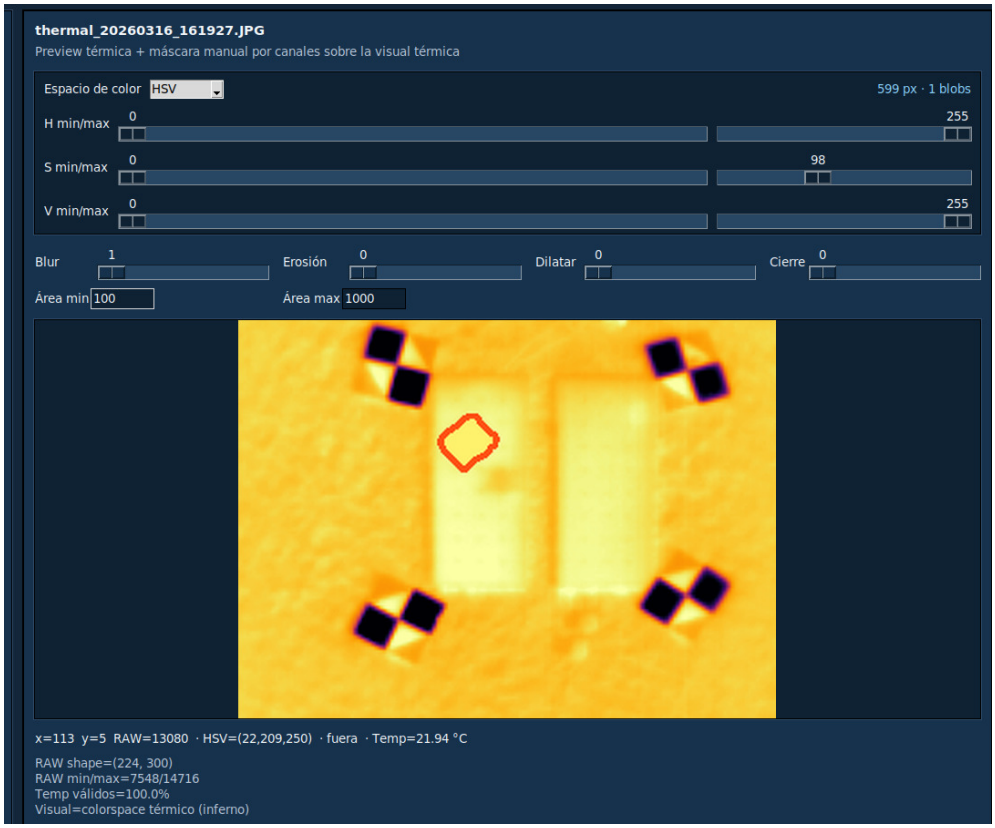


Figura 4.3.1.5: Visualización de una región térmica de interés dentro de la interfaz del sistema, donde se evidencia la detección de un punto caliente y su asociación con la información térmica reportada durante el análisis. Fuente: elaboración propia.

A nivel de implementación, la temperatura no se calcula únicamente a partir del valor *RAW* leído en pantalla. Primero se corrige el valor del píxel según la emisividad de la escena, la transmisión atmosférica, la contribución de la temperatura reflejada, la temperatura atmosférica y la transmisión de la ventana infrarroja. Después de esta corrección se aplica la ecuación de conversión basada en los parámetros de calibración Planck del sensor:

$$T_{^\circ\mathrm{C}} = \frac{B}{\ln\left(\frac{R_1}{R_2\,(raw_{obj}+O)} + F\right)} - 273{,}15 \tag{4.3.1}$$

En esta ecuación, se puede entender que raw_{obj} representa el valor radiométrico corregido del objeto, mientras que R_1, R_2, B, F y O corresponden a los parámetros de calibración propios de la cámara térmica. Este procedimiento permite que la lectura por píxel presentada en la interfaz sea consistente con la radiometría almacenada en los archivos _raw.npy, _tempC.npy y _meta.json, dejando una trazabilidad completa entre visualización y dato físico.

En síntesis, lo expuesto en esta sección permite abordar de manera directa los objetivos específicos 3 y 4 del proyecto. Esto se debe a que, por una parte, la estrategia planteada integra un flujo operativo de visión artificial que comprende la delimitación geométrica de la escena mediante *targets* térmicos, la rectificación del área de análisis, la segmentación de anomalías sobre la visual térmica y la recuperación de la temperatura desde la radiometría original, respondiendo así al objetivo de habilitar un proceso funcional de inspección térmica dentro del sistema. Por otra parte, la incorporación de este flujo en una escena

experimental instrumentada permiten comprobar el funcionamiento del sistema sobre un caso base real, lo que da soporte al objetivo de validar experimentalmente la propuesta en un arreglo fotovoltaico sencillo bajo condiciones controladas.

4.4. Manejo de la escena

Resulta importante señalar que el desempeño del sistema no depende únicamente del algoritmo de análisis implementado, sino también de las condiciones físicas bajo las cuales se realiza la captura térmica. En este sentido, una de las principales limitaciones identificadas corresponde a la resolución del sensor térmico integrado en el dron Parrot ANAFI Thermal. Debido a que se trata de una cámara de resolución relativamente baja frente al tamaño y detalle de los elementos inspeccionados, la altura de vuelo influye de manera directa sobre la calidad del análisis. En las pruebas realizadas se observó que un rango aproximado entre 1 y 8 metros permite conservar un nivel de detalle aceptable sobre la escena; por encima de este intervalo, la huella espacial de cada píxel aumenta y la información térmica pierde definición, dificultando la identificación precisa de regiones anómalas o puntos calientes de tamaño reducido.

De igual manera, las condiciones climáticas tienen un efecto considerable sobre la confiabilidad de la captura. Para este tipo de inspección, las condiciones más favorables corresponden a ambientes soleados o levemente nublados, en los cuales existe suficiente contraste térmico sin generar una radiación excesiva sobre la superficie observada. Cuando la incidencia solar directa es demasiado intensa, la escena puede presentar una emisividad aparente elevada y un comportamiento térmico menos uniforme. Esto dificulta la interpretación de la imagen y puede introducir respuestas no deseadas en la cámara. Además, bajo estas condiciones el sensor térmico puede representar con menor claridad las diferencias locales de temperatura, afectando la identificación de las anomalías de interés.

Otro factor relevante es la temperatura ambiental, ya que interviene en la conversión del valor radiométrico a temperatura física. La estimación térmica final no depende únicamente del valor *RAW* registrado por el sensor, sino también de parámetros asociados al entorno y a la escena, tales como la temperatura ambiente, la emisividad asumida del objeto y otras condiciones de captura almacenadas en el archivo `.json`. Estos datos permiten realizar la conversión y obtener valores de temperatura más precisos. Por ello, los cambios significativos en la temperatura del entorno pueden modificar la interpretación térmica de una misma escena y afectar la comparabilidad entre inspecciones realizadas en distintos momentos del día o bajo diferentes condiciones meteorológicas. En consecuencia, se recomienda efectuar las capturas bajo condiciones ambientales lo más estables posible y registrar, en la medida de lo posible, las variables del entorno que puedan influir en la conversión de temperatura.

Capítulo 5

Interfaz del usuario

5.1. Requerimientos del computador

Para la implementación del proyecto se emplea una estación de control en tierra (computador portátil o de escritorio) bajo arquitectura x86_64. La carga principal del sistema proviene del refresco de telemetría y gráficos embebidos para la vision artificial (deteccion de puntos calientes y anomalias termicas en paneles solares), decodificación y visualización de video RTSP en tiempo real (stream en vivo del dron), y ejecución de procesos auxiliares como de telemetria, grabacion del RTSP y guardado de metadatos.

Tabla 5.1.1: Requisitos recomendados para ejecutar la interfaz.

Componente	Mínimo funcional	Recomendado
CPU	4 núcleos (Intel i5 / Ryzen 5 o equivalente)	6–8 núcleos para multitarea
RAM	8 GB	16 GB o más
Almacenamiento	20 GB libres	SSD con 50–100 GB libres
Red	Wi-Fi estable	Wi-Fi 5/6, baja latencia.
GPU	No obligatoria	Útil si se ejecuta procesamiento adicional

En este proyecto se utilizó un computador con Ubuntu 22.04.2 LTS, conectividad Wi-Fi 6 a 2.4/5 GHz, procesador Ryzen 5 7535HS, 16 GB de memoria RAM y almacenamiento SSD (512 GB), lo cual permitió ejecutar en paralelo la GUI, el registro de telemetría y pruebas con ROS 2 sin degradación del rendimiento, permitiendo funciones de multitareas dentro de la interfaz.

5.2. Instalación de Software

Se describe y se da una guía detallada del software necesario para correr la interfaz al completo, iniciando por la instalación del sistema operativo hasta la instalación del framework ROS 2 Humble y librerías necesarias para la GUI.

5.2.1. Instalación del sistema operativo

La sección detallada de instalación del sistema operativo se encuentra en los anexos para consulta (ver Anexo A). En cuanto a software, los requerimientos están planteados

en el archivo denominado `requirements.txt`, el cual, al ejecutarse, realizará la descarga, descompresión e instalación de todos los paquetes necesarios para poder ejecutar esta interfaz.

- Python: Python 3.10 (incluido por defecto en Ubuntu 22.04).
- SDK del dron: Parrot Olympe version 7.5.
- Librerías Python: numpy, matplotlib, opencv-python, Pillow.
- Dependencias del sistema: python3-tk (Tkinter), ffmpeg (grabación RTSP), librerías gráficas base (libgl1).
- ROS 2 Humble.

La preparación del sistema y la instalación de ROS 2 Humble se trasladaron al Anexo C para consulta detallada, de igual forma la instalación de ROS 2 Humble, la creación del workspace y demás parámetros de configuración se trasladaron al Anexo D.

5.2.2. Entorno virtual e instalación Python

El proyecto usa un entorno virtual para aislar dependencias Python de la GUI, visión por computador, exportación de metadatos y control del dron. El archivo `requirements.txt` contiene únicamente los paquetes Python requeridos por el proyecto.

```
cd anafi_suite_ros2
python3 -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
pip install -r requirements.txt
```

Listing 5.1: Creación del entorno virtual e instalación de dependencias Python.

La construcción del workspace ROS 2 se encuentra detallada en el Anexo C.

5.2.3. Instalación automática

Para automatizar todo el proceso anterior se puede ejecutar el script `install_all_dependencies.sh` desde la raíz del proyecto.

```
cd anafi_suite_ros2
chmod +x install_all_dependencies.sh
./install_all_dependencies.sh
```

Listing 5.2: Instalación automática completa.

5.2.4. Ejecución del proyecto

Después de instalar dependencias y compilar el workspace, la ejecución se realiza cargando ROS 2, activando el entorno virtual y lanzando los scripts principales del proyecto.

```
cd anafi_suite_ros2
source /opt/ros/humble/setup.bash
source .venv/bin/activate
bash 00_PRIMERO_construir_ros2.bash
bash 03_iniciar_todo.bash
```

Listing 5.3: Ejecución de la GUI y del entorno ROS 2.

La verificación de conectividad con el dron se trasladó al Anexo C.

5.3. Manual de usuario por páginas

La interfaz se organiza en una barra lateral (sidebar) que permite navegar por las páginas del sistema. La ventana principal y la lógica de navegación se implementan en interfaz/app_shell.py mediante la clase AppShell, la cual instancia cada página según la selección del usuario. El dashboard lateral resume el estado global del sistema y concentra acciones rápidas. Como se observa en la Figura 5.3.1, el panel Estado muestra indicadores de: conexión de sensores, estado del puente ROS 2, estado de ejecución (Runner), disponibilidad del dron (identificador o guiones si no está conectado) y conteo de mensajes recibidos.

Adicionalmente, el dashboard incorpora dos botones de uso frecuente:

- Conectar: habilita la lectura de telemetría (equivalente al botón de conexión dentro de la página Sensores).
- Iniciar ROS2: habilita el módulo de publicación hacia ROS 2 cuando el entorno se ejecuta con Humble.

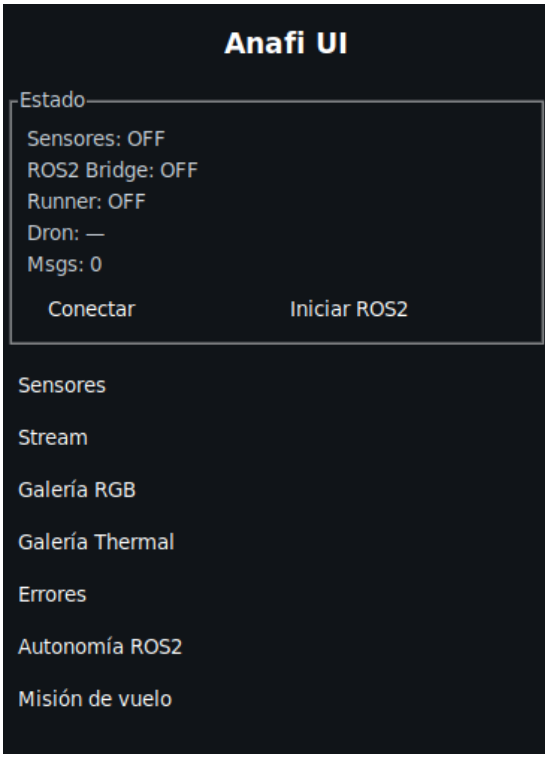


Figura 5.3.1: Dashboard lateral: indicadores de estado, botones de acción rápida y navegación por páginas. Fuente: elaboración propia.

En la parte inferior del panel se listan las páginas disponibles: Sensores, Stream, Galería RGB, Galería Thermal, Misión de vuelo y Errores. Esta separación por páginas reduce la carga computacional durante el vuelo, ya que organiza las funciones por objetivo en hilos individuales (telemetría, video, gestión de medios, ROS 2, misiones y modelo de errores).

5.3.1. Página Sensores

La página Sensores (interfaz/pages/sensores_page.py) es el punto de entrada operativo del sistema, es la página inicial de la GUI al ejecutar el programa, concentra la conexión al dron, la lectura de telemetría, la visualización de estado y el gráfico 3D del recorrido realizado por el dron. La Figura 5.3.1.1 muestra el diseño general: una columna de variables IMU/velocidad/GPS, un panel principal con el gráfico 3D, y un botón de Conectar en la esquina superior derecha.

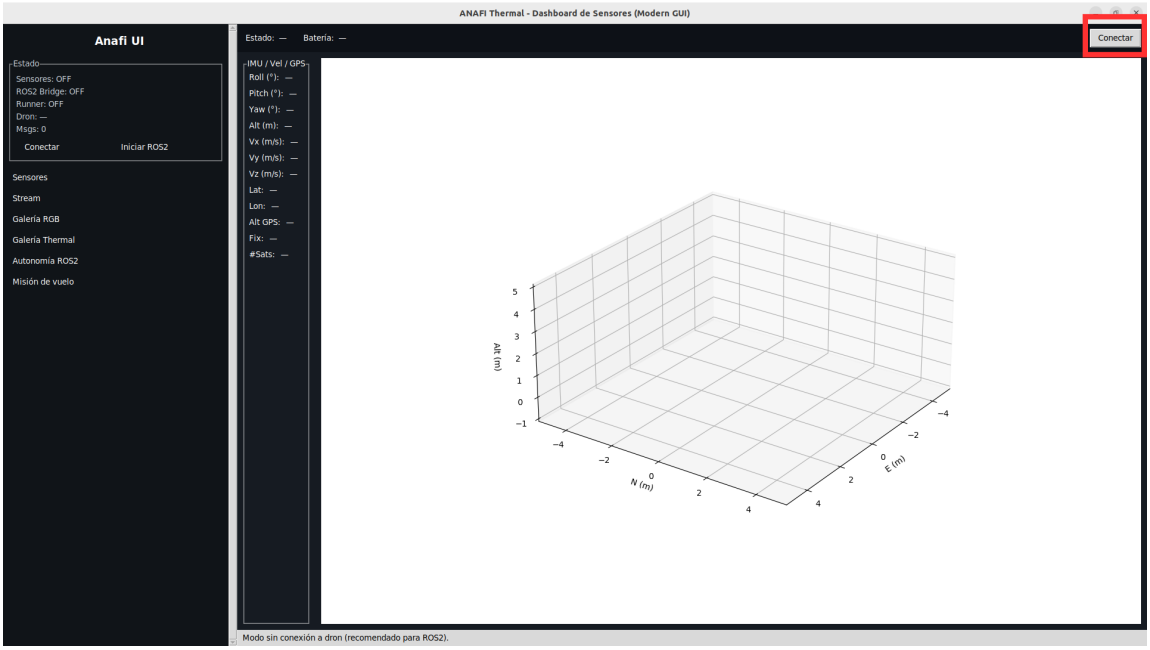


Figura 5.3.1.1: Página Sensores: telemetría (IMU/velocidad/GPS), estado general y botón Conectar. Fuente: elaboración propia.

5.3.1.1. Conexión y ciclo de actualización

El botón Conectar ejecuta la función toggle_connect(), la cual crea/inicializa el objeto SensorStream (definido en sensores/streams.py). Este flujo establece una única sesión con Olympe hacia el host configurado (DRONE_IP) y lanza un hilo de lectura a una frecuencia de 20 Hz (SENSOR_HZ) para independizar los hilos de entrada del dron. La actualización de etiquetas y gráficos se realiza cada 500 (REFRESH_MS) milisegundos, para mantener fluidez visual sin saturar CPU.

5.3.1.2. Variables de telemetría utilizadas

La interfaz presenta las variables más relevantes para el control y diagnóstico durante vuelo. En particular:

- Actitud (IMU): Roll, Pitch y Yaw (°), útiles para validar estabilidad y orientación.
- Altitud: altura estimada (m), usada también como eje vertical del gráfico 3D.
- Velocidades: Vx, Vy, Vz (m/s), relevantes para maniobras y verificación de comandos de movimiento.
- GPS: latitud y longitud (grados), altitud GPS (m), Fix (si se cumple con el minimo de satelites = 4) y número de satélites (#Sats).
- Estado general: estado de conexión, estado de vuelo y batería (porcentaje), visibles en la parte superior.

Estas variables se consultan en segundo plano y se exponen en un snapshot que la GUI consume para refrescar etiquetas dentro de la página. Cuando el dron no está conectado, la interfaz mantiene valores en guiones (–) para indicar ausencia de datos.

5.3.1.3. Visualización 3D de trayectoria

El panel central integra un gráfico 3D basado en Matplotlib (implementado en interfaz/plot3d.py). La Figura 5.3.1.2 ilustra el comportamiento: se grafica la posición del dron en un marco local ENU (East–North–Up), donde E y N representan desplazamientos horizontales (m), y el eje Alt corresponde a la altitud (m), para poder tomar estos valores se necesita Fix = True ya que estos se ayudan de los valores del GPS.

Cada punto azul representa una muestra del estado de posición; la lista de puntos se mantiene con un tamaño máximo definido por PATH_MAX_POINTS (200) para evitar crecimiento indefinido. De forma complementaria, la orientación instantánea se representa con un cubo (o frame) que se actualiza con los ángulos de actitud (roll/pitch/yaw), facilitando interpretar cambios de orientación mientras se avanza en la trayectoria de vuelo.

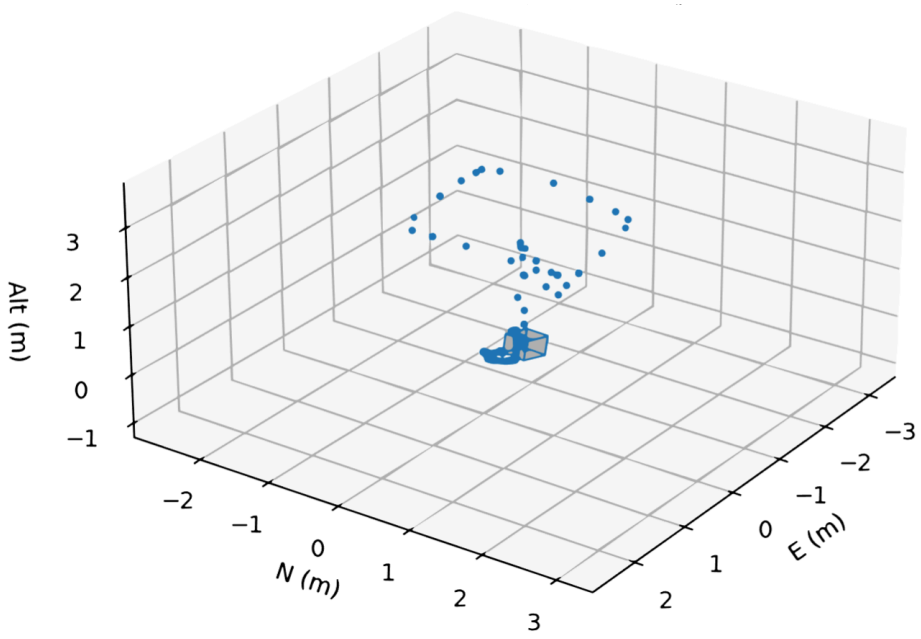


Figura 5.3.1.2: Gráfico 3D: trayectoria del dron en el marco ENU (E, N, Alt). Fuente: elaboración propia.

5.3.2. Página Stream

La página Stream (interfaz/pages/stream_page.py) está orientada a la visualización en tiempo real del video RGB del dron mediante un RTSP (Real Time Streaming Protocol), y a la orientación de la cámara a través del control del pitch del gimbal. Su objetivo es proporcionar retroalimentación visual durante el vuelo (encuadre, altura aproximada por referencia visual, verificación de maniobras) y habilitar acciones rápidas para integración con autonomía/ROS2.

La Figura 5.3.2.1 muestra la interfaz de la página de Stream y resalta tres grupos de controles: el botón Abrir Autonomía ROS2 (marcado en rojo), los botones Iniciar stream y Detener stream (marcados en naranja), y los controles del gimbal (marcados en azul), junto al indicador del ángulo objetivo de Pitch.

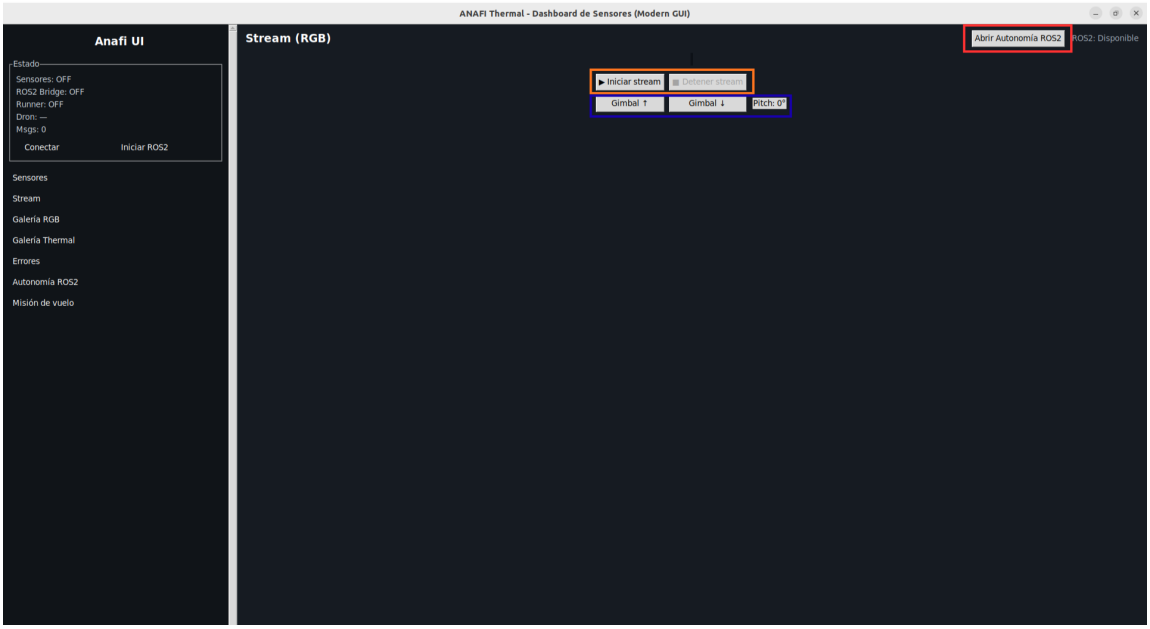


Figura 5.3.2.1: Página Stream: acceso a Autonomía ROS2 (rojo), control de streaming (naranja) y control de gimbal/pitch (azul). Fuente: elaboración propia.

5.3.2.1. Flujo de uso recomendado

En pruebas de campo, el flujo típico de operación fue:

1. Conectar Sensores desde la página Sensores (habilita telemetría, estados y sesión Olympe).
2. Cambiar a Stream e iniciar el video con Iniciar stream.
3. Ajustar el gimbal para orientar el encuadre de la cámara (por ejemplo, inspección frontal o nadir).
4. (Opcional) Abrir Autonomía ROS2 para ejecutar/verificar el puente ROS2 y rutinas de autonomía.

Cuando el stream se encuentra activo, los frames se actualizan en el panel principal y se mantiene la interacción con los botones de control. La Figura 5.3.2.2 muestra un ejemplo del stream en vivo durante un vuelo corto de prueba (takeoff y hovering).

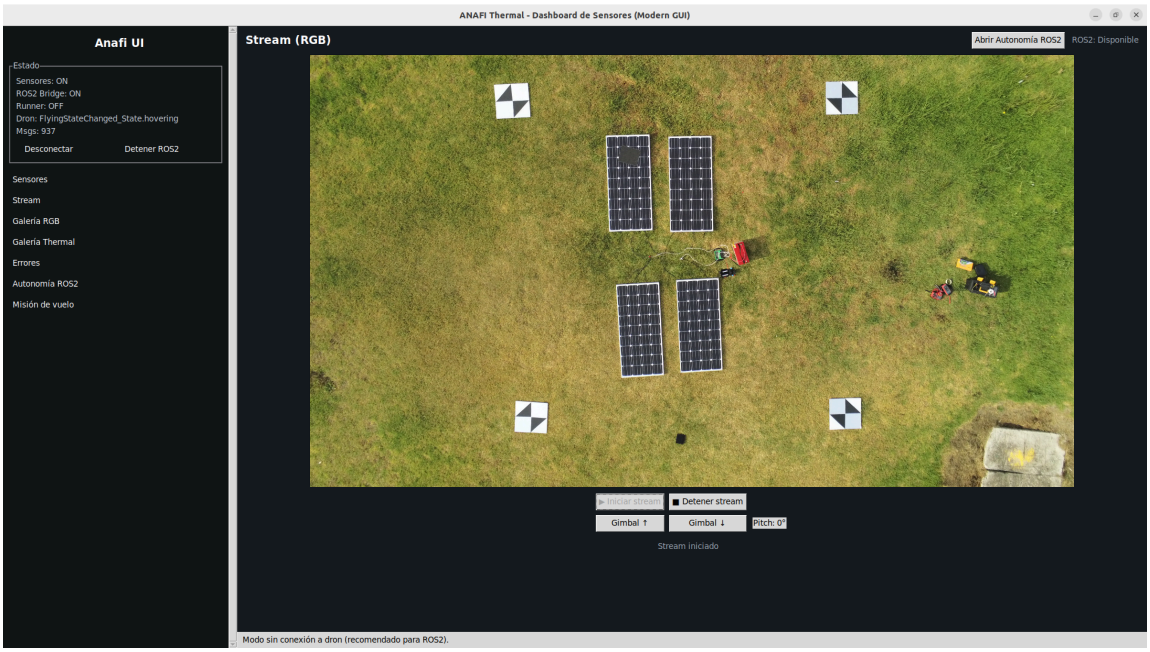


Figura 5.3.2.2: Stream RGB en vivo durante un vuelo de prueba (despegue y estacionario). Fuente: elaboración propia.

5.3.2.2. Implementación del streaming RTSP

El video se consume desde la URL definida en config.py (DRONE_RTSP_URL). Para evitar bloquear la GUI, la lectura del stream se ejecuta en un hilo dedicado mediante la clase RtspThreaded (ubicada en sensores/rtsp_streaming.py). En cada ciclo, el frame decodificado con OpenCV se transforma a PhotoImage con Pillow para ser renderizado en el componente gráfico de Tkinter, mostrando así el stream en vivo fotograma a fotograma. Este proceso limita los fps del streaming a un rango aproximado de 20 a 30 fps.

- Iniciar stream: _start_stream() crea/activa el hilo RTSP y habilita la actualización del panel de streaming dentro de la interfaz.
- Detener stream: _stop_stream() finaliza el hilo, libera recursos y detiene el procesamiento y refresco de frames.

5.3.2.3. Control del gimbal

Los botones Gimbal ↑ y Gimbal ↓ implementan un control incremental del pitch en una secuencia de 5 grados. Internamente se usa una función de ajuste (_nudge_gimbal()) que actualiza el valor objetivo y envía un comando Olympe (gimbal.set_target). El indicador Pitch, refleja el último valor objetivo aplicado, permitiendo al operador llevar control del encuadre en el streaming.

5.3.2.4. Bucle de reintento para establecer la conexión al stream

Durante la operación, el endpoint RTSP puede no estar disponible inmediatamente por inicialización del encoder, latencia de red o cambios de enlace. Para mejorar la robustez, la función _start_stream() implementa un loop de reintento antes de declarar error al conectar stream:

- Se intenta abrir el stream y leer el primer frame válido.
- Si la conexión falla (por ejemplo, timeout o lectura nula), el sistema cierra el intento, espera un intervalo corto y reintenta.
- El ciclo finaliza al recibir el primer frame válido (stream establecido) o al alcanzar un máximo de intentos (1000), informando el estado al usuario sin congelar la interfaz.

Esta estrategia reduce fallos por condiciones transitorias y evita que el operador tenga que reiniciar la aplicación cuando el stream tarda en iniciar o directamente no inicia. Además, al ejecutarse en un hilo dedicado, la GUI permanece funcional mientras se completa la conexión, en caso de no establecer una conexión, la interfaz puede ejecutar el resto de páginas, ya que estas no son dependientes entre sí.

5.3.3. Página Galería RGB (gestión local de capturas)

Archivo principal: interfaz/pages/gallery_page.py.
La página Galería RGB organiza y visualiza las fotografías capturadas por la cámara visible del dron en el directorio local media/rgb/. Su propósito dentro del flujo operativo es doble: permitir una verificación rápida del material almacenado (fotos tomadas durante pruebas, inspecciones o misiones), y facilitar la recolección de evidencias para el informe final sin depender de herramientas externas.

En la Figura 5.3.3.1 se muestran los controles principales resaltados por color: Refrescar(azul) reconstruye la grilla de miniaturas leyendo nuevamente el contenido de la carpeta; Abrir carpeta (rojo) abre el directorio de trabajo en el explorador de archivos del sistema; y Abrir (verde) permite visualizar el archivo seleccionado (imagen) utilizando el visor definido por la aplicación.



Figura 5.3.3.1: Página Galería RGB: botón Refrescar (azul), Abrir carpeta (rojo) y acción Abrir por elemento (verde). Fuente: elaboración propia.

5.3.3.1. Lectura de directorio y generación de miniaturas

Al iniciar la página, o al presionar Refrescar, la interfaz ejecuta un procedimiento de indexación local:

1. Escanea media/rgb/y filtra extensiones soportadas (p. ej. jpg, .jpeg ,.png).
2. Ordena los archivos por fecha (metadatos del sistema o por convención de nombre basada en tiempo).
3. Genera miniaturas para cada elemento (redimensionamiento) con Pillow para mantener tiempos de carga bajos.
4. Renderiza las miniaturas en un recuadro dentro de un contenedor desplazable (scroll), evitando saturar la ventana principal.

Esta estrategia permite explorar rápidamente decenas de capturas sin bloquear la GUI, dado que las imágenes completas sólo se cargan cuando el usuario presiona Abrir Para mantener trazabilidad entre pruebas de vuelo y material almacenado, se recomienda una convención de nombres basada en timestamp, por ejemplo rgb_YYYYMMDD_HHMMSS.jpg. Esta convención facilita: ordenar cronológicamente capturas, asociar archivos con sesiones de telemetría/ROS2 y documentar resultados en el capítulo de pruebas y anexos..

5.3.3.2. Acciones de usuario

Las acciones principales quedan trazadas a métodos dentro de gallery_page.py:

- Refrescar (azul): refresh() limpia el contenedor gráfico y reconstruye la lista de miniaturas a partir del estado actual del directorio.
- Abrir carpeta (rojo): _open_folder() invoca el explorador de archivos del sistema (en Linux típicamente xdg-open) para inspección manual.
- Abrir (verde): _open_media(path) abre la captura seleccionada; para imágenes, se utiliza un visor embebido o una ventana secundaria.

5.3.4. Página Galería Thermal

Archivo principal: interfaz/pages/gallery_thermal_page.py.
La página *Galería Thermal* amplía la lógica de la galería RGB y la transforma en un visor de análisis preliminar para capturas térmicas. A nivel operativo, esta página se alimenta del directorio media/thermal/ y permite recorrer tanto las fotografías térmicas originales como los archivos derivados producidos después del análisis, por ejemplo ROI_warp.jpg, hotspots_overlay.jpg y hotspots_mask.png.

La Figura 5.3.4.1 muestra la vista general de esta página. En ella, el botón Refrescar, marcado en amarillo, reconstruye la grilla de miniaturas una vez termina una misión de vuelo o un análisis térmico, incorporando los nuevos archivos guardados por el sistema. A su derecha, el botón Abrir carpeta, resaltado en rojo, abre el directorio media/thermal/, permitiendo navegar manualmente por sus subcarpetas y acceder a las capturas térmicas, a la carpeta de matrices radiométricas y a las salidas del módulo de errores. Finalmente,

el botón Abrir, señalado en azul sobre cada tarjeta, permite cargar el archivo seleccionado dentro del visor térmico detallado.

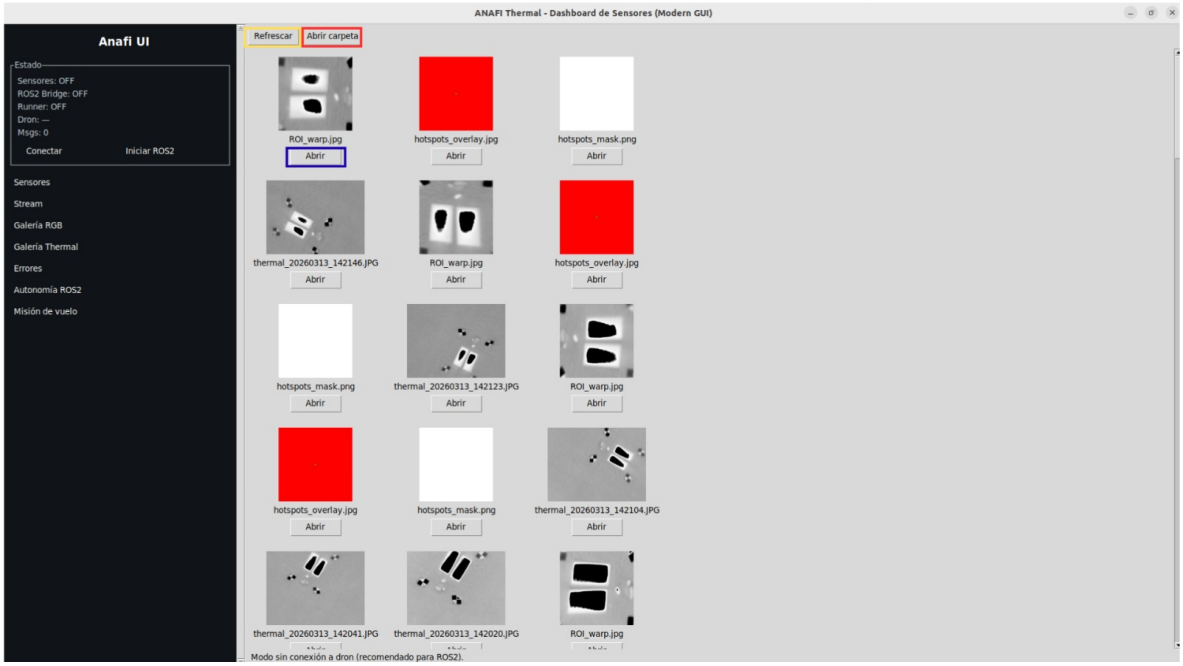


Figura 5.3.4.1: Página *Galería Thermal*: actualización de la vista, apertura del directorio térmico y acceso a archivos individuales de la misión y del análisis. Fuente: elaboración propia.

Cuando el operador abre una captura térmica radiométrica, la interfaz intenta localizar la fotografía original y, de forma complementaria, generar o reutilizar una versión _mapped.png ubicada en la carpeta mapped/. Esta versión se colorea con el mapa *inferno* para facilitar la interpretación visual de contrastes térmicos. Sin embargo, tal como se aprecia en la Figura 5.3.4.2, el archivo mapped se utiliza únicamente para visualización humana. El procesamiento no se realiza sobre esta visual a color, dado que el mapeo no es uniforme en términos físicos y puede exagerar o comprimir diferencias de temperatura según el rango escogido para la normalización.

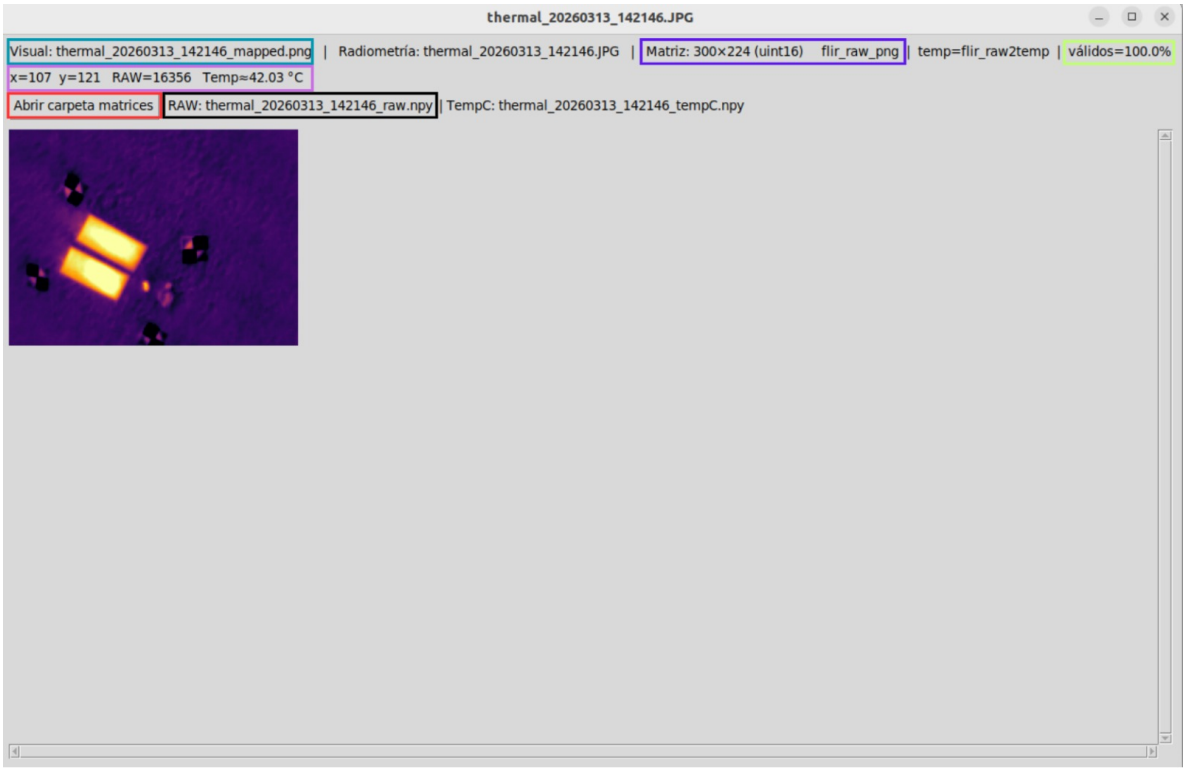


Figura 5.3.4.2: Visor térmico detallado: archivo visual mapped, lectura por píxel, acceso a la carpeta de matrices y resumen de la matriz radiométrica asociada. Fuente: elaboración propia.

En la barra superior del visor se distinguen varios elementos funcionales. El recuadro azul informa el nombre del archivo visualizado, por ejemplo thermal_20260313_142146_mapped.png; esta convención permite diferenciar la vista coloreada del archivo radiométrico fuente. El recuadro fucsia inferior presenta la información de *pixel-hover*, es decir, la coordenada x, y del cursor, el valor RAW asociado a la matriz térmica y la temperatura estimada en grados Celsius para dicho punto. En el ejemplo mostrado, el píxel ubicado en $x = 107$, $y = 121$ presenta un valor RAW=16356 y una temperatura aproximada de 42,03 °C. Este mecanismo resulta útil para comprobar rápidamente si un valor parece físicamente coherente o si, por el contrario, se encuentra cerca de valores sospechosos, saturados o inconsistentes para la escena.

El botón Abrir carpeta matrices, marcado en rojo, abre la ubicación donde se almacenan los archivos .json, _raw.npy y _tempC.npy asociados a la imagen seleccionada. A su derecha, el recuadro negro indica el nombre exacto de la matriz radiométrica utilizada por el visor, mientras que el recuadro morado resume las dimensiones y el formato de dicha matriz. En el ejemplo documentado, la interfaz reporta una matriz de 300 × 224 en formato uint16, extraída mediante el método flir_raw_png. El recuadro verde claro, por su parte, presenta el porcentaje de valores válidos convertidos a temperatura. Dentro del proyecto, un valor válido corresponde a un píxel que permanece dentro del rango radiométrico utilizable de la captura y que, además, puede convertirse a temperatura sin producir valores indefinidos. Por ello, un indicador como válidos=100.0 % significa que toda la matriz pudo ser interpretada correctamente para esa imagen.

Con el fin de dejar trazabilidad de esta conversión, el sistema genera para cada imagen un archivo de metadatos, por ejemplo thermal_20260313_142146_meta.json. En este archivo se conservan tanto los datos básicos de la matriz como los parámetros radiométricos utilizados durante la conversión. La Tabla 5.3.1 resume los valores más representativos de la captura mostrada en la Figura 5.3.4.2.

Parámetro	**Valor registrado**
Dimensión de la matriz térmica	300×224 píxeles
Tipo de dato *RAW*	uint16
Rango *RAW* observado	8228 a 16776
Temperatura mínima y máxima convertida	$-19{,}46\,^{\circ}\mathrm{C}$ a $44{,}38\,^{\circ}\mathrm{C}$
Temperatura mediana	$24{,}19\,^{\circ}\mathrm{C}$
Porcentaje de valores válidos	$100{,}0\,\%$
Emisividad	1.0
Temperatura reflejada aparente	$22{,}0\,^{\circ}\mathrm{C}$
Temperatura atmosférica	$20{,}0\,^{\circ}\mathrm{C}$
Temperatura de ventana IR	$27{,}97\,^{\circ}\mathrm{C}$
Transmisión de ventana IR	0.80
Humedad relativa	$50{,}0\,\%$

Tabla 5.3.1: Parámetros radiométricos relevantes registrados para la captura thermal_20260313_142146.

5.3.5. Página Errores (detección manual por umbrales y análisis por misión)

Archivo principal: interfaz/pages/errors_page.py.
La página *Errores* concentra la etapa de inspección manual y análisis por lote de las imágenes térmicas pertenecientes a una misión. Su propósito no es reemplazar la radiometría original, sino brindar al operador una herramienta de exploración controlada para ajustar umbrales, revisar una previsualización térmica y ejecutar después el análisis sobre todas las imágenes seleccionadas bajo una misma configuración. En este sentido, la página se apoya tanto en las fotografías térmicas como en sus matrices RAW y TempC, y además integra el modelo OBB empleado para localizar los *targets* de referencia.

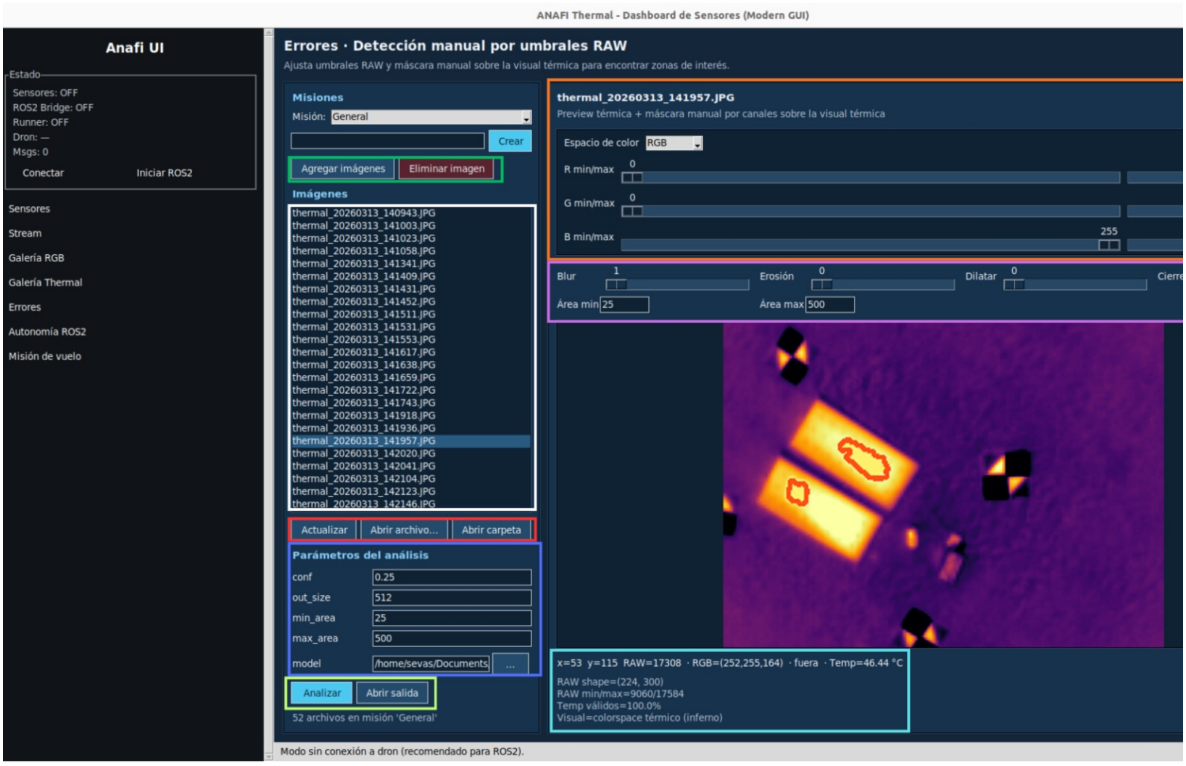


Figura 5.3.5.1: Página *Errores*: organización por misión, vista previa térmica, umbralización manual y ejecución del análisis de salida. Fuente: elaboración propia.

En el panel izquierdo, el bloque verde superior reúne las acciones Agregar imágenes y Eliminar imagen. Estas permiten construir o depurar la misión térmica actual, incorporando nuevas capturas o retirando archivos repetidos e innecesarios. Debajo de este bloque se encuentra el recuadro blanco de Imágenes, donde se listan los nombres de todas las capturas pertenecientes a la misión seleccionada. Al hacer clic sobre cualquiera de ellas, la interfaz carga la vista previa correspondiente y habilita el ajuste de parámetros sobre una imagen individual. Esta lógica es importante porque, aunque el operador prueba visualmente sobre una sola imagen, los parámetros fijados en esta página quedan concebidos para ser aplicados al conjunto completo de archivos de la misión.

En la parte inferior izquierda se observa el bloque rojo, compuesto por los botones Actualizar, Abrir archivo y Abrir carpeta. Estos controles permiten refrescar el contenido de la misión, abrir una imagen específica desde el gestor de archivos del sistema o acceder directamente a la carpeta donde se almacenan los datos seleccionados. Más abajo aparece el bloque azul Parámetros del análisis, donde se consignan los parámetros globales asociados al sistema OBB y al procesamiento posterior. Dentro de estos valores, conf representa la confianza mínima aceptada para la detección del modelo; aunque la interfaz permite valores exploratorios menores, en pruebas orientadas a priorizar detecciones robustas se recomienda trabajar por encima de 0.75. El parámetro out_size define la resolución de salida del recorte rectificado: 512 píxeles se asumió como valor de trabajo casi estándar, 256 puede emplearse cuando el volumen de imágenes es muy alto y se busca reducir carga computacional, mientras que 1024 resulta útil cuando se requiere preservar mayor detalle espacial. Los parámetros min_area y max_area establecen el rango de área aceptado para las regiones detectadas, evitando

conservar blobs demasiado pequeños o exageradamente grandes dentro de la máscara resultante.

El panel derecho de la página se dedica al análisis visual de una imagen seleccionada. En la parte superior, el recuadro anaranjado agrupa los *trackbars* de detección por umbrales. Allí el operador escoge un espacio de color, como RGB, HSV, LAB o escala de grises, y ajusta los valores mínimos y máximos de cada canal. Este ajuste se realiza sobre la visual térmica mapeada y permite aislar, de manera simple y de bajo costo computacional, las zonas cuya apariencia cromática se asocia con un punto caliente o con un comportamiento térmico distinto al del resto de la escena. La lógica detrás de este diseño consiste en aprovechar que el usuario puede reconocer con mayor facilidad una anomalía sobre una visual coloreada, aun cuando el cálculo final de temperatura deba recuperarse después desde la matriz radiométrica original.

Debajo de este recuadro aparece la zona morada inferior, donde se ubican los parámetros de posprocesamiento: *blur*, erosión, dilatación y cierre. Estos controles complementan la etapa principal de umbralización y permiten suavizar el ruido presente en la imagen, consolidar regiones conectadas y limpiar contornos irregulares en la máscara resultante. Estas operaciones corresponden a transformaciones morfológicas clásicas en procesamiento de imágenes, las cuales permiten mejorar la calidad de la segmentación al eliminar pequeñas regiones aisladas o al reforzar las zonas de interés detectadas, como se puede evidenciar en las operaciones morfológicas mostradas en la Figura 5.3.5.2.

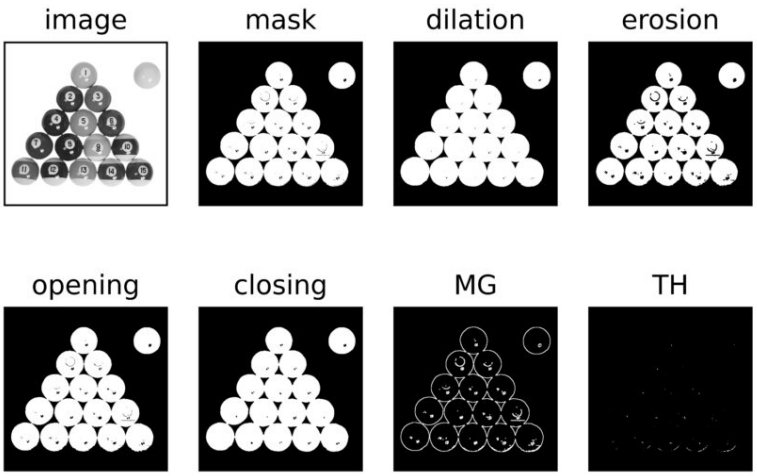


Figura 5.3.5.2: Ejemplo de operaciones morfológicas utilizadas en procesamiento de imágenes, incluyendo erosión, dilatación, apertura y cierre, aplicadas durante el posprocesamiento de la máscara de detección. Fuente: elaboración propia.

En el mismo bloque se incluyen además los parámetros *Área min* y *Área max*, los cuales acotan la cantidad de píxeles que puede tener una región detectada. Esta restricción resulta útil porque, dentro de una escena térmica, pueden existir otros puntos con temperaturas similares a la del punto caliente real. Limitar el tamaño de las regiones detectadas permite descartar detecciones innecesarias o excesivamente extensas

que no correspondan realmente a la zona de interés analizada.

Finalmente, el bloque lima inferior izquierdo, formado por los botones Analizar y Abrir salida, ejecuta el análisis sobre la imagen seleccionada o abre la carpeta de resultados generados. Cuando se dispara el análisis, el sistema recurre a la matriz tempC.npy real, detecta los cuatro *targets*, rectifica la región de interés y genera artefactos como ROI_warp.jpg, ROI_warp_tempC.npy, hotspots_mask.png, hotspots_overlay.jpg y debug.json. Con ello se mantiene la separación metodológica planteada en esta tesis: la detección preliminar se apoya en una visual térmica intuitiva para el operador, pero la cuantificación final de la anomalía siempre se fundamenta en la matriz radiométrica real y en la temperatura físicamente calculada para cada píxel.

5.3.6. Página Misión de vuelo (editor JSON + ejecución de waypoints)

La página Misión de vuelo constituye el módulo de planificación, edición, almacenamiento y despacho de trayectorias para el dron ANAFI Thermal. La interfaz permite definir una misión basada en waypoints, validar la coherencia de los parámetros, guardar o recuperar trayectorias en formato JSON y ejecutar la misión desde la propia GUI. En la Figura 5.3.6.1 esta funcionalidad se organiza mediante bloques resaltados por color, lo que permite distinguir con claridad las zonas de configuración, gestión, ejecución y edición detallada.

En la parte superior aparece el bloque amarillo, donde se agrupan los parámetros globales de la misión: nombre, tipo, sistema de referencia, número de waypoints, velocidad e inclinación del gimbal. Estos valores alimentan directamente la estructura serializable generada por _collect_mission_dict(), por lo que no son solo campos visuales, sino la cabecera operativa de la misión. A su lado se ubica el bloque gris, que activa despegue y aterrizaje automáticos, estas banderas son interpretadas posteriormente por el ejecutor de misión, ya sea cuando la ejecución ocurre de forma local o cuando la misión se delega al nodo ROS 2 encargado de ejecutarla.

Debajo de estos controles se ubican tres grupos de acciones. El bloque azul concentra la gestión documental de la misión: crear una nueva, cargar una trayectoria JSON, guardar la configuración actual y validar los datos de la tabla y del editor, esta zona se apoya en _new_mission(), _load_json(), _save_json(), _collect_mission_dict(), _apply_mission_dict() y _validate_ui(), con lo cual la interfaz mantiene sincronía entre los valores visibles y la representación persistente en archivo. A continuación, el bloque rojo reúne la exportación y limpieza de metadatos, administrados por PhotoMetadataStore, este bloque es relevante porque la misión no solo ordena movimientos, sino que también puede asociar capturas RGB o térmicas a una corrida específica y exportar después esos registros a formatos tabulares. El bloque naranja corresponde al control de ejecución, mediante los botones Ejecutar y Detener. Finalmente, el bloque rosado presenta el estado resumido de la misión, por ejemplo Listo, Ejecutando, Cancelada o mensajes de error.

En la zona central izquierda se encuentra el bloque blanco, correspondiente a la tabla de waypoints, cada fila contiene índice, desplazamientos relativos dx, dy, dz, cambio de guiñada dyaw_deg, tiempo de espera, acción, sensor y modo. La tabla se sincroniza con el número total de puntos por medio de _sync_n_waypoints(), mientras que _on_select() transfiere al editor el contenido del punto marcado por el usuario. En la parte inferior del panel izquierdo aparece el bloque celeste, destinado a reorganizar la secuencia. Los

botones Subir, Bajar, Insertar y Eliminar modifican directamente la estructura de la tabla mediante _move_selected(), _insert_after() y _delete_selected().

El bloque morado, ubicado a la derecha, implementa el editor detallado del waypoint, en él se ingresan desplazamientos relativos, dyaw, tiempo de espera, acción, sensor y modo. El método _apply_to_selected() escribe estos valores sobre la fila activa de la tabla. Una característica importante de esta sección es que dyaw se expresa en grados para simplificar la interacción del operador, pero al momento de ejecutar la misión ese valor se convierte a radianes, de acuerdo con la convención utilizada por moveBy y extended_move_by de Olympe.

Las Figuras 5.3.6.3 y 5.3.6.4 muestran las ventanas emergentes para cargar y guardar misiones en formato JSON, estas ventanas extienden funcionalmente el bloque azul, ya que permiten conservar trayectorias reutilizables por campaña o tipo de prueba. En la organización real de la suite, las misiones de ejemplo se almacenan dentro de app/media/missions/, mientras que la misión activa despachada a ROS 2 se copia a runtime/current_mission.json.

Desde el punto de vista operativo, la lógica de ejecución de esta página combina interfaz local y orquestación ROS 2,cuando el operador pulsa Ejecutar, el método _run_async() valida primero la misión y luego intenta transferirla al cliente Ros2MissionClient, si el servicio ROS 2 está disponible, la GUI libera previamente la propiedad de la conexión Olympe mediante _release_gui_drone_ownership(), detiene los módulos de sensores y bridge que puedan estar activos, guarda la misión en runtime/current_mission.json y realiza una llamada al servicio /anafi/mission/start. En esta modalidad, la GUI asume un rol de edición y supervisión, mientras la ejecución física del vuelo queda a cargo del nodo ROS 2 /anafi/mission_manager, el estado de esa ejecución es seguido desde la interfaz mediante _ros2_tick(), que consulta y muestra el avance publicado en /anafi/mission/status. Si el mission manager no está disponible, la misma página conserva una ruta de ejecución local en un hilo independiente para no bloquear la GUI.

Las opciones de acción por waypoint permiten seleccionar entre no ejecutar ninguna acción, tomar fotografía o controlar la grabación de video. La captura RGB se resuelve mediante _capture_rgb_snapshot(), mientras que la grabación de video se coordina con _video_start() y _video_stop(), el ajuste de orientación de cámara se realiza con _try_set_gimbal_pitch(). Para la traslación del dron, la página utiliza _move_by(), que intenta aprovechar extended_move_by cuando se dispone de una velocidad objetivo y, en caso contrario, recurre a moveBy, esta decisión hace que la misión conserve compatibilidad con distintas capacidades del entorno Olympe.

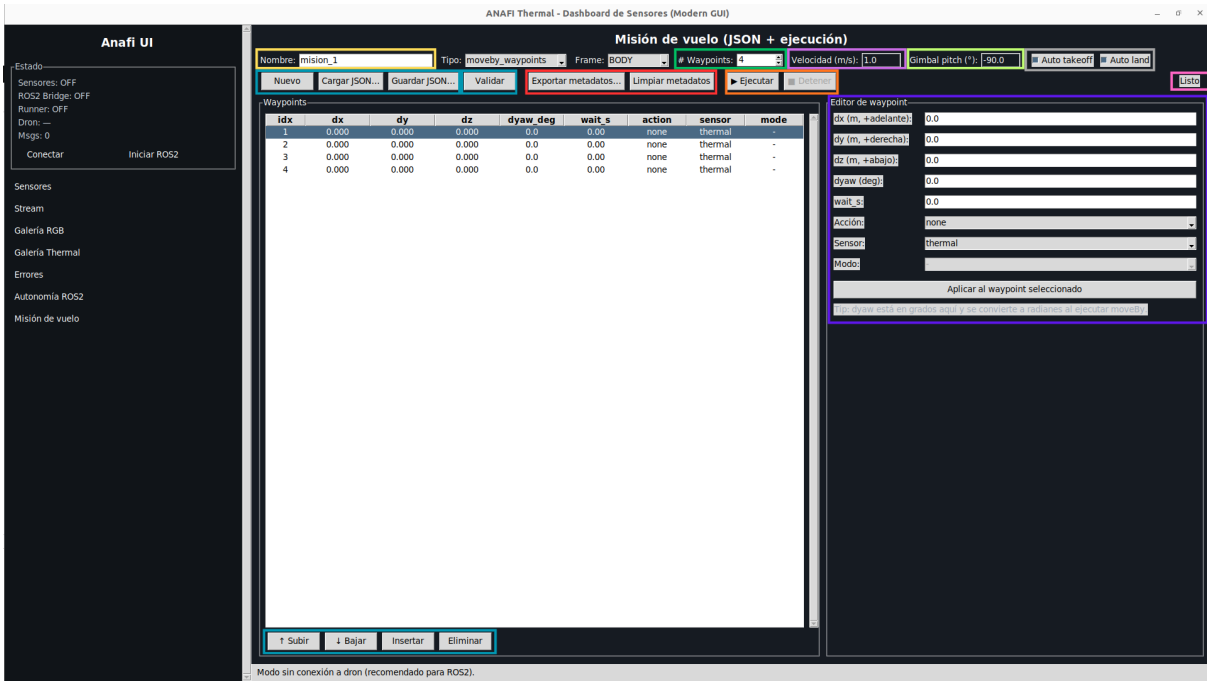


Figura 5.3.6.1: Interfaz principal de la página de misión de vuelo con identificación de sus bloques funcionales. Fuente: elaboración propia.

La Figura 5.3.6.2 muestra el editor detallado del bloque morado. Allí se observa la convención de signos empleada en la interfaz: dx positivo hacia adelante, dy positivo hacia la derecha y dz positivo hacia abajo, esta convención es consistente con la manera en que luego se genera la orden relativa para Olympe.

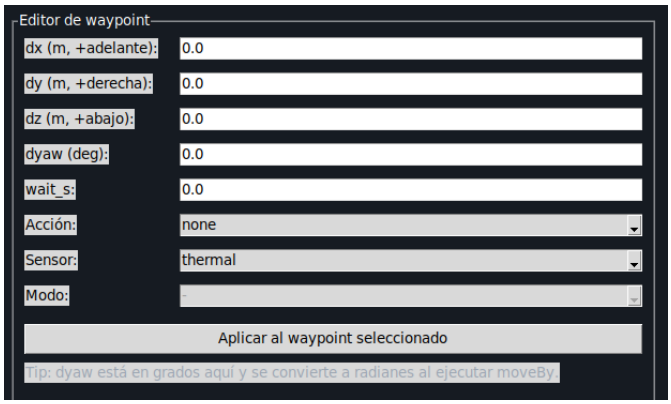


Figura 5.3.6.2: Panel de edición de waypoints, donde se ingresan los parámetros de desplazamiento, orientación, espera y acción asociada. Fuente: elaboración propia.

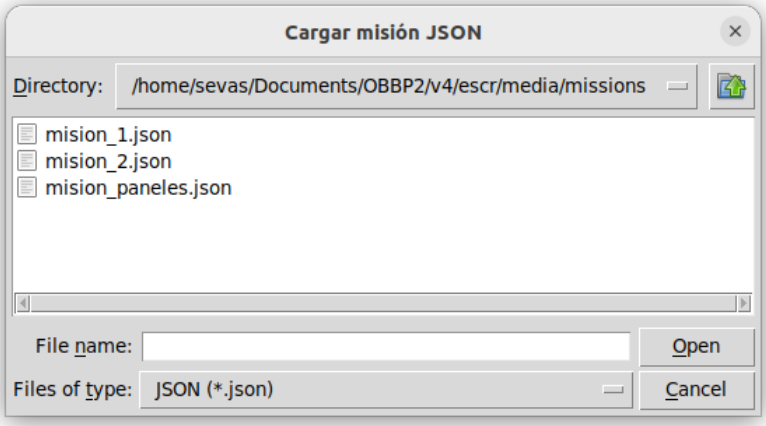


Figura 5.3.6.3: Ventana emergente para cargar una misión de vuelo almacenada en formato JSON. Fuente: elaboración propia.



Figura 5.3.6.4: Ventana emergente para guardar la configuración actual de la misión en formato JSON. Fuente: elaboración propia.

La página de misión incorpora la función de guardado de metadatos disponible en la carpeta media, su propósito es consolidar en un único archivo .xlsx la información asociada a cada captura realizada durante la misión, de manera que cada fila represente una imagen y permita relacionar la captura con el punto de vuelo, la marca temporal, la posición geográfica, la actitud del dron y las rutas de los archivos radiométricos y de análisis generados por el sistema. La integración de esta función puede verse al terminar una misión con una ventana emergente que permite guardar estos datos o se puede realizar mediante un botón Guardar metadatos, ubicado dentro de la página de misión, al ejecutarse, este botón exporta un archivo con una convención de nombre del tipo genai_{mision}_{fecha}.xlsx, facilitando la trazabilidad de los datos recolectados en cada operación, como ejemplo de estos datos recolectados en una de las pruebas se dispone el Anexo F.

Tabla 5.3.2: Información almacenada en el archivo de metadatos de misión.

Grupo	**Información almacenada**
Identificación de captura	Nombre de misión, identificador de captura, waypoint, sensor y modo de adquisición.
Tiempo de captura	Fecha y hora de captura en tiempo local y en UTC.
Posición del dron	Latitud, longitud, altitud GPS y altitud relativa.
Estado de vuelo	Roll, pitch, yaw, porcentaje de batería, estado de GPS y número de satélites.
Archivos asociados	Nombre y ruta de la imagen, matriz radiométrica y archivos derivados generados por el sistema.
Resultados de análisis térmico	Región rectificada, máscara, imagen de superposición y resumen de regiones detectadas, cuando exista análisis previo.

La Tabla 5.3.2 resume la información que se almacena en el archivo de metadatos por cada captura realizada.

5.3.7. Análisis comparativo de métricas térmicas, segmentación y altura de vuelo

Con el propósito de validar integralmente el sistema antes de proceder al análisis cuantitativo, se llevaron a cabo un total de veinte vuelos de prueba que abarcaron distintos contextos y condiciones de operación. Estas pruebas permitieron verificar que tanto el workspace como el workflow definidos resultaban correctos, funcionales y adecuados para los objetivos del proyecto. Entre los escenarios evaluados se incluyeron capturas sobre superficies variadas, como pavimento con pintura de cancha de microfútbol, ensayos sin paneles solares con el único propósito de comprobar el recorte geométrico, y pruebas exclusivamente con los targets de referencia sin anomalía térmica presente, con el fin de confirmar que el sistema respeta la condición de requerir los cuatro puntos de referencia para delimitar correctamente la región de captura y habilitar el análisis posterior. A partir de esta etapa exploratoria, y habiendo establecido la robustez general del sistema, el análisis que se presenta a continuación se concentra en una prueba real, única y de carácter lineal, en la que se conserva una misma escena con un mismo panel fotovoltaico a lo largo de un único día de medición. Durante esta sesión se mantuvieron condiciones medioambientales relativamente estables, propias de la dinámica solar y de nubosidad del entorno, lo que permitió estudiar de forma aislada cómo variaciones leves en las condiciones de iluminación —ausencia de nubes frente a nubosidad parcial— y en la altura de vuelo inciden sobre los resultados obtenidos, conservando en todo momento un mismo punto caliente como referencia de detección.

La configuración utilizada fue la siguiente y se resume en la Tabla 5.3.3.

Tabla 5.3.3: Configuración general de la misión de vuelo empleada para las capturas térmicas comparativas.

Waypoint	**Altura**	**Tiempo**	**Acción a realizar**
WP1	1 m	10 s	Estabilizar el dron frente a la escena y realizar la primera captura térmica de referencia.
WP2	2 m	10 s	Mantener la orientación hacia el panel fotovoltaico y capturar imagen RGB, térmica y metadatos.
WP3	3 m	10 s	Repetir la captura sobre la misma región de interés para comparar el cambio de escala.
WP4	5 m	10 s	Registrar la muestra usada como altura de referencia para el ajuste del tamaño aparente del punto caliente.
WP5	7 m	10 s	Evaluar la respuesta del sistema ante una mayor distancia de adquisición.
WP6	8 m	10 s	Realizar la captura final y verificar la persistencia de la anomalía térmica detectada.

Una vez realizada la misión de vuelo y obtenidas las capturas térmicas con sus matrices asociadas y metadatos, se prosiguió con el análisis. Para esto, se realizó un análisis comparativo sobre una misma zona de interés observada en seis capturas representativas, adquiridas a distintas alturas y bajo condiciones cercanas de operación, con lo cual, para cada muestra se conservó la misma lógica general de procesamiento: primero se localizó la región útil mediante el recorte geométrico basado en los *targets* de referencia, detectados sobre la imagen en escala de grises; posteriormente, una vez obtenida la región recortada y corregida en perspectiva, la segmentación térmica se efectuó nuevamente sobre la visual térmica mapeada, ya que es en dicha representación donde los *trackbars* y los umbrales definidos en la interfaz conservan coherencia radiométrica y visual.

La Figura 5.3.7.1 muestra precisamente esta secuencia de observación sobre el mismo punto caliente en las seis imágenes analizadas. En ella se presentan las vistas ya recortadas a partir de la detección geométrica en escala de grises, usando los cuatro *targets* como referencia para normalizar la región inspeccionada. No obstante, aunque la imagen que pasa por el algoritmo de detección OBB se encuentra en escala de grises, la detección final de la anomalía térmica se realiza nuevamente sobre la imagen mapeada, empleando el espacio de color deseado, que fue el utilizado en la configuración mostrada por la interfaz durante el ajuste de umbrales

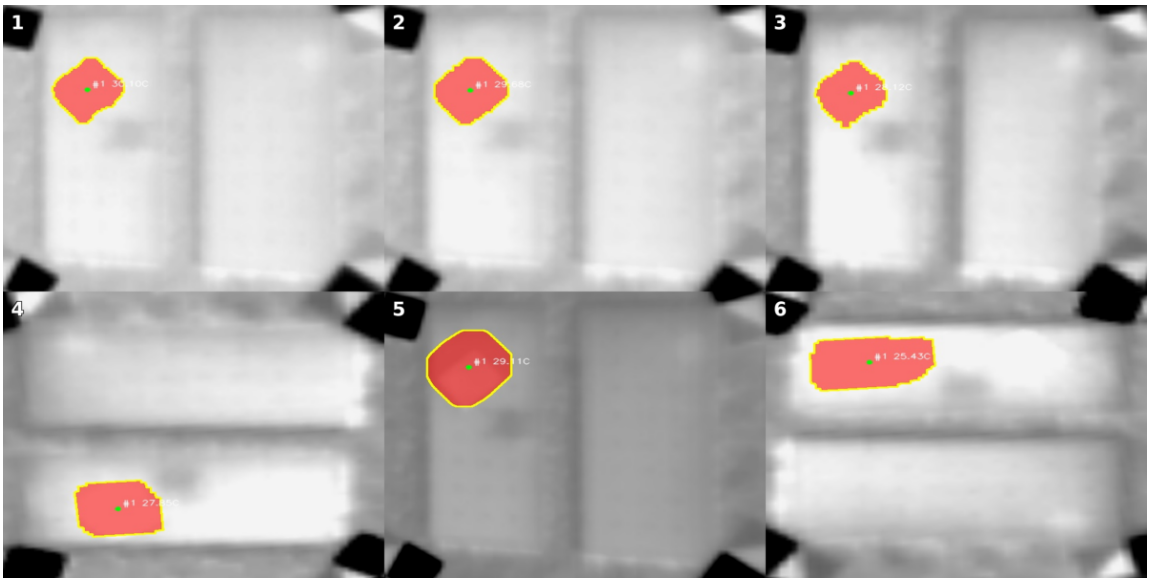


Figura 5.3.7.1: Región de interés observada en las seis muestras tras el recorte geométrico basado en *targets*.

Una vez definida la región de interés para cada captura, se extrajeron las temperaturas mínima, promedio y máxima de la zona segmentada, así como el tamaño aparente de la anomalía en número de píxeles, lo que se puede visualizar en la Figura 5.3.7.2. En el panel superior se muestra la evolución térmica de la región detectada, mientras que en el panel inferior se compara el tamaño aparente de la zona de interés expresado en píxeles con una versión ajustada a una altura de referencia de 5 m.

En este contexto, la magnitud denominada *detected hotspot size* corresponde al número de píxeles clasificados como pertenecientes a la región caliente detectada, es decir, al área aparente ocupada por la anomalía dentro del plano imagen, lo cual no representa directamente un área física en unidades métricas, sino una superficie proyectada cuya escala depende de la distancia entre el sensor y la escena. Por esta razón, para facilitar una comparación más directa entre capturas tomadas a distintas alturas, se empleó una compensación relativa a una altura de referencia de 5 m, dada por:

$$A_{5m} = A_{px} \left(\frac{h_{\text{eff}}}{5} \right)^2 \tag{5.3.1}$$

donde A_{px} es el número de píxeles detectados en la región, h_{eff} es la altura efectiva de adquisición y A_{5m} representa el tamaño ajustado a una condición equivalente de 5 m. Esta relación se fundamenta en el comportamiento de escala proyectiva y se relaciona conceptualmente con el *Ground Sampling Distance* (GSD), entendido como la relación entre una distancia física en la escena y su representación en píxeles.

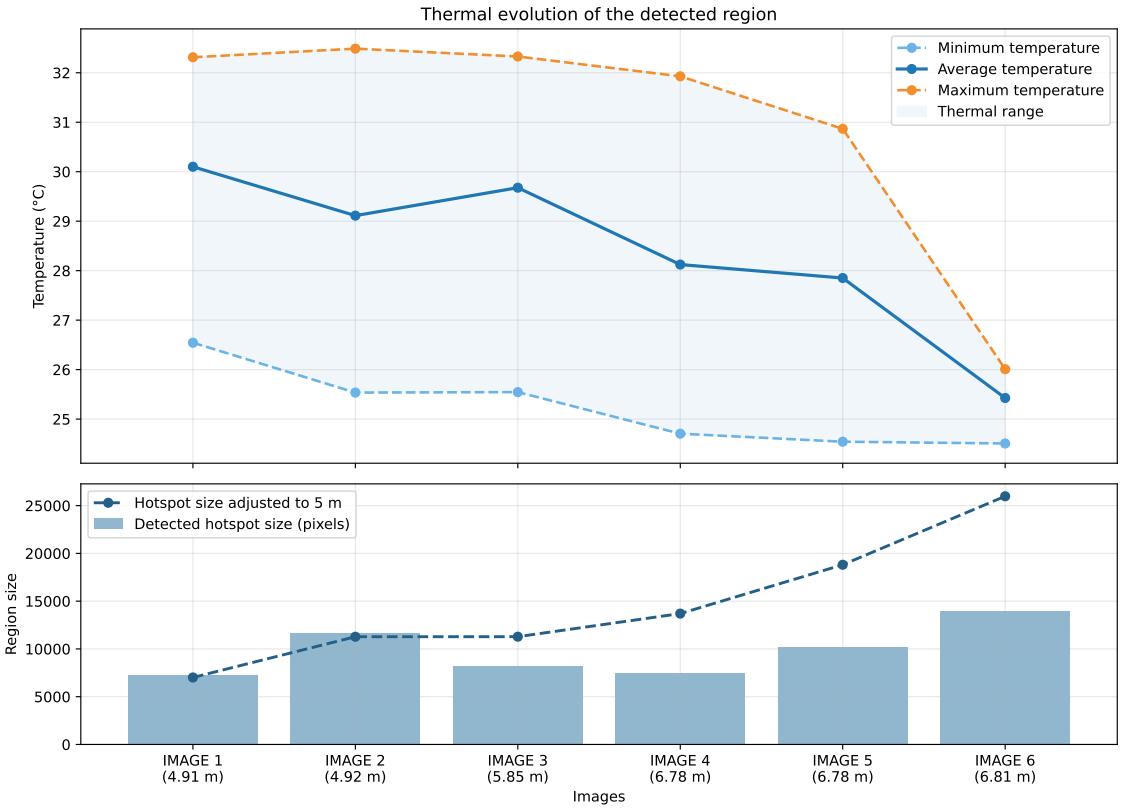


Figura 5.3.7.2: Evolución térmica de la región detectada y comparación del tamaño aparente de la zona de interés entre muestras.

En el panel inferior de la Figura 5.3.7.2, la línea discontinua representa precisamente esta magnitud ajustada a 5 m y su crecimiento marcado en las últimas muestras no debe interpretarse directamente como una expansión física real y abrupta del punto caliente, sino como el efecto combinado entre el conteo de píxeles detectado, la variación de altura y la sensibilidad del proceso de segmentación, obteniendo que el factor de compensación depende cuadráticamente de la altura, pequeñas diferencias en la distancia de adquisición pueden amplificar visualmente la curva ajustada.

Respecto al comportamiento térmico mostrado en el panel superior de la Figura 5.3.7.2, las lecturas obtenidas son globalmente consistentes, ya que en todas las muestras se conserva la relación esperada entre temperatura mínima, promedio y máxima, y la evolución general de la curva no presenta discontinuidades incompatibles con el fenómeno físico observado. Sin embargo, también se aprecian pequeñas inconsistencias locales entre muestras, particularmente en la temperatura mínima y en la temperatura máxima de las últimas capturas. Este comportamiento es razonable si se considera que la detección de la región se realizó mediante umbrales sobre la imagen térmica mapeada y que dicha segmentación depende de parámetros como el desenfoque, la erosión y la dilatación, lo cual se entiende como que el procesamiento está expuesto a que cualquier variación leve pueda añadir o eliminar algunos píxeles del contorno segmentado, alterando así el conjunto exacto de valores térmicos que entra en el cálculo estadístico.

En otras palabras, un píxel que en una imagen queda incluido dentro de la región segmentada puede quedar excluido en la siguiente, aun cuando visualmente el punto caliente sea prácticamente el mismo. Esta sensibilidad se vuelve más relevante al

aumentar la altura de adquisición, dado que la sensibilidad del sensor aumenta y cualquier reflejo del sol puede llegar a generar una aparente región de interés casi igual a un punto caliente y, por lo tanto, el efecto pocos píxeles o centímetros adquiere un peso relativo mayor. Por ello, a partir de aproximadamente 7 m puede observarse un deterioro en la temperatura mínima de la región, incorporándose valores más cercanos al entorno térmico normal dentro del área detectada. De manera similar, la disminución apreciable de la temperatura máxima en la última muestra, así como la reducción asociada en la temperatura promedio, puede explicarse por la combinación de varios factores: sensibilidad de la segmentación a pequeñas variaciones de borde, tiempo transcurrido con el sensor encendido, posible deriva térmica del sistema y una modificación sutil de las condiciones ambientales, particularmente por el paso leve de una nube que alteró el contraste radiométrico de la escena.

Por su parte, la Figura 5.3.7.3 compara la altura solicitada para cada captura frente a la altura efectivamente alcanzada durante la ejecución y la diferencia entre ambas curvas no debe interpretarse como una falla estricta del sistema, sino como una consecuencia natural del modo en que la misión fue ejecutada. Aunque el control del dron dispone de mecanismos de estabilización basados en controladores tipo PID, la referencia vertical no se impone mediante un setpoint georreferenciado rígido que obligue al vehículo a mantenerse exactamente sobre una altura absoluta, sino mediante comandos de desplazamiento aplicados en lazo abierto. Bajo estas condiciones, resulta esperable que existan pequeñas discrepancias entre la altura teórica y la altura realmente registrada.

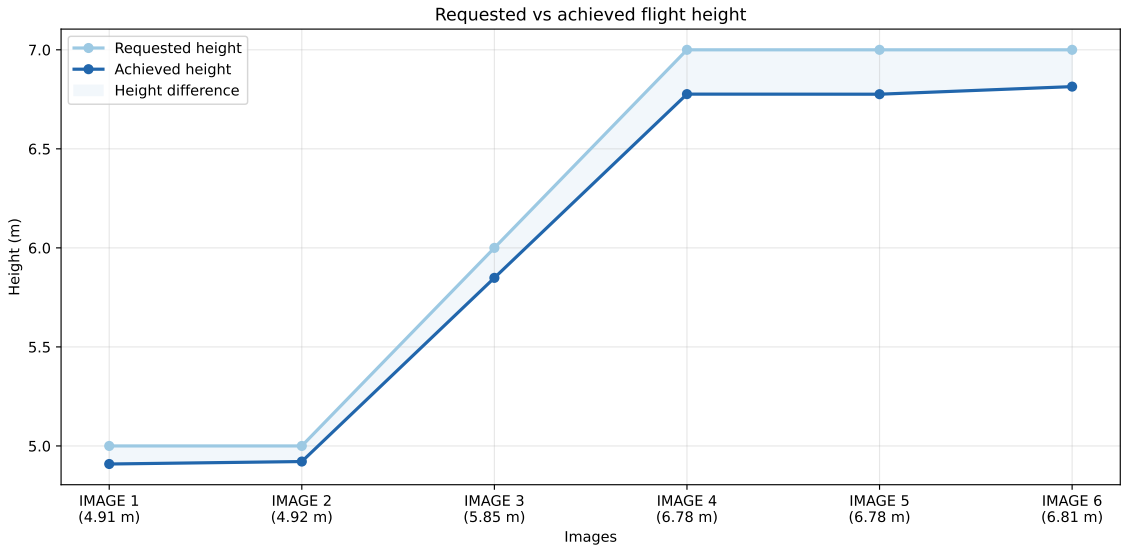


Figura 5.3.7.3: Comparación entre la altura solicitada y la altura efectivamente alcanzada durante las capturas.

Las diferencias observadas en la Figura 5.3.7.3 pueden asociarse a fluctuaciones del clima en la escena, variaciones del viento, así como al calentamiento progresivo del sistema durante la misión. En consecuencia, el vehículo puede experimentar desplazamientos leves en cualquiera de sus ejes, pequeñas caídas o sobrecorrecciones antes de recuperar una condición cuasi estable. Por esta razón, una altura nominal de 5 m puede traducirse experimentalmente en valores cercanos, por ejemplo 4,91 m, mientras que los niveles superiores presentan también desviaciones moderadas respecto

al objetivo solicitado. Estas diferencias, aunque acotadas, afectan directamente la escala de la imagen y justifican la necesidad de compensar el tamaño aparente de la región detectada cuando se comparan muestras térmicas obtenidas a distintas alturas si se quiere realizar un análisis comparativo entre las diversas capturas.

En conjunto, las Figuras 5.3.7.1, 5.3.7.2 y 5.3.7.3 evidencian que la interfaz permite no solo ejecutar y registrar misiones de forma consistente, sino también analizar cuantitativamente cómo pequeñas variaciones de adquisición, segmentación y condiciones ambientales afectan la interpretación térmica posterior. Aun cuando el proceso de detección por umbrales introduce sensibilidad frente a cambios de escala, ruido radiométrico y perturbaciones externas, los resultados conservan coherencia global y permiten establecer tendencias útiles para la inspección. Con ello, se concluye que la interfaz desarrollada constituye una herramienta válida tanto para la operación del sistema como para el análisis experimental de campañas térmicas comparables.

Como cierre del capítulo, puede afirmarse que la interfaz propuesta cumple una doble función dentro del proyecto: por un lado, actúa como medio de interacción y despacho operacional sobre el dron; por otro, ofrece una base suficientemente trazable para evaluar experimentalmente la relación entre condiciones de adquisición, comportamiento de vuelo y respuesta térmica observada en la escena inspeccionada.

Capítulo 6

Conclusiones y recomendaciones

6.1 Conclusiones

En relación con el primer objetivo específico, se logró establecer una arquitectura de software que integra ROS 2 con el SDK Parrot Olympe para el control del dron desde una estación base y la adquisición de imágenes RGB y térmicas. Esta integración permitió construir una solución operativa capaz de coordinar la navegación, la captura de datos y su organización para la inspección. Sin embargo, la arquitectura aún depende parcialmente de la lógica propia del fabricante, por lo que no se ha alcanzado una migración completa hacia una estructura totalmente desacoplada y centrada en ROS 2.

Respecto al segundo objetivo específico, se consiguió habilitar la ejecución de misiones autónomas repetibles mediante navegación por puntos de referencia, lo que permitió recorrer trayectorias predefinidas y mantener un comportamiento de vuelo suficientemente estable para tareas de inspección en escenarios controlados. Esto constituye una base importante para la revisión periódica y comparativa de sistemas fotovoltaicos estáticos.

En cuanto al tercer objetivo específico, se implementó un flujo funcional de inspección térmica que integra la delimitación de la zona de análisis mediante *targets* de referencia, la rectificación de la escena, la detección de anomalías térmicas generales por umbralización y el análisis de resultados a través de una interfaz gráfica de usuario. Esta estrategia permitió desarrollar una línea de análisis de bajo costo computacional y comprensible para el operador. No obstante, el sistema se orienta principalmente a la detección de anomalías térmicas generales y no a la clasificación específica de fallas sobre módulos fotovoltaicos.

Con respecto al cuarto objetivo específico, el sistema fue validado sobre un arreglo fotovoltaico simple con carga, empleado como caso base experimental para evaluar el proceso completo de visión artificial enfocado en fotovoltaica. Aun así, los resultados deben interpretarse dentro de las condiciones particulares del montaje experimental, dado que la escala del sistema evaluado y la relativa simplicidad de la escena limitan su extrapolación directa a instalaciones de mayor tamaño o complejidad.

De manera general, se concluye que el proyecto alcanzó los objetivos planteados y demostró la factibilidad de emplear un dron térmico comercial como plataforma de apoyo para tareas de inspección autónoma en sistemas fotovoltaicos. No obstante, también se evidenció que variables como la altura de operación, la resolución térmica disponible y las condiciones particulares de captura influyen de forma importante en la calidad del análisis. En consecuencia, el trabajo desarrollado constituye una base funcional y

escalable, pero aún requiere mejoras técnicas para ampliar su robustez y aplicabilidad en escenarios reales.

6.2 Trabajos futuros

Como trabajo futuro, se recomienda fortalecer el componente de visión por computador mediante algoritmos orientados al análisis específico de módulos fotovoltaicos, de manera que el sistema pueda diferenciar con mayor precisión distintos tipos de anomalías y reducir la dependencia de la interpretación manual.

Asimismo, se recomienda avanzar hacia una migración más completa de la arquitectura de control a ROS 2, con el fin de unificar la lógica del sistema en un entorno más modular, mantenible e interoperable.

En cuanto a la interfaz gráfica, conviene realizar mejoras orientadas a la claridad visual, la fluidez de uso y la seguridad operativa, especialmente en la organización de controles y en la validación de acciones críticas.

A su vez, se podría realizar un control más robusto, donde se asocie tanto la lectura del GPS para reafimar o realizar una corrección mucho más robusta del movimiento del dron, lo cual permitiría realizar capturas replicables sin tener en cuenta los cambios de ubicación que puede llegar a tener el dron cuando se encuentre afectado por corrientes de viento durante una misión.

Finalmente, se recomienda ampliar la validación experimental hacia escenarios más cercanos a condiciones reales de operación, considerando distintas alturas de vuelo, variaciones ambientales y configuraciones más complejas de arreglos fotovoltaicos.

6.3 Recomendaciones

En caso de replicar el proyecto, es importante considerar que el desempeño del sistema depende de una adecuada integración entre hardware, software y condiciones de captura. En primer lugar, se debe contar con una plataforma aérea compatible con el ecosistema de desarrollo utilizado, particularmente un dron que permita acceso programático al control de vuelo, a la telemetría y a la captura de imágenes térmicas y RGB.

De igual manera, se recomienda disponer de un entorno de ejecución estable en Ubuntu 22.04 LTS o un sistema equivalente, con soporte para Python, ROS 2 y las librerías necesarias para procesamiento de imágenes, visualización e integración con la interfaz gráfica.

También se debe prestar especial atención a la calidad de la captura térmica, ya que factores como la altura de vuelo, la resolución del sensor, la estabilidad del dron y las condiciones ambientales influyen directamente en la utilidad del análisis posterior. En particular, con sensores térmicos de baja resolución, conviene trabajar a alturas controladas y bajo condiciones ambientales lo más estables posible para conservar detalle suficiente en la escena.

Asimismo, se recomienda verificar previamente la calibración del dron y de sus subsistemas, especialmente el magnetómetro, el gimbal y la referencia de horizonte, debido a que estas condiciones afectan de forma directa la estabilidad del vuelo, la orientación de la cámara y la repetibilidad de las capturas.

Referencias Bibliográficas

[1] Ultralytics, "Yolo oriented bounding boxes (obb)," 2024. Consultado en 2026.

[2] International Electrotechnical Commission, "IEC TS 62446-3:2017 – Photovoltaic systems – Requirements for testing, documentation and maintenance – Part 3: Outdoor infrared thermography of PV modules and plants in operation," tech. rep., IEC, Geneva, Switzerland, 2017. [En línea]. Disponible: https://webstore.iec.ch/en/publication/28628.

[3] International Electrotechnical Commission, "IEC TS 62446-3: Photovoltaic (PV) systems – Requirements for testing, documentation and maintenance – Part 3: Photovoltaic modules and plants – Infrared thermography," tech. rep., IEC, Geneva, Switzerland, 2017.

[4] M. A. Alajmi, A. Al-Shamma'a, and S. Addali, "A review of infrared thermography for photovoltaic fault detection and performance monitoring," *Energies*, vol. 15, no. 6, pp. 1–23, 2022.

[5] Y. Zhang *et al.*, "Drone-based infrared thermography for fault detection in photovoltaic plants: A review of methods and datasets," *Renewable and Sustainable Energy Reviews*, vol. 189, 2024.

[6] Hikmicro, "Inspección de paneles fotovoltaicos con cámaras termográficas." Hikmicro Industrial Blog, 2023. [Online]. Available: https://www.hikmicrotech.com/es/explore/industrial-blog/inspeccion-de-paneles-fotovoltaicos-con-camarastermograficas/.

[7] SkyVisor, "Solar thermography: inspection of photovoltaic plants using drones." SkyVisor Technical Resources, 2022. [Online]. Available: https://www.skyvisor.ai/es/thermography/.

[8] Parrot, "Anafi thermal – technical specifications." Parrot Drones SAS, 2023. [En línea]. Disponible: https://www.parrot.com/en/drones/anafi-thermal/technical-specifications.

[9] Parrot, "Olympe sdk – python interface for parrot drones." Parrot Drones SAS, 2023. [En línea]. Disponible: https://developer.parrot.com/docs/olympe/.

[10] FLIR Systems, "Lepton engineering datasheet," 2023. Consultado en 2026.

[11] G. Messina and G. Modica, "Applications of uav thermal imagery in precision agriculture: State of the art and future research outlook," *Remote Sensing*, vol. 12, no. 9, p. 1491, 2020. [En línea]. Disponible: https://doi.org/10.3390/rs12091491.

[12] DJI, "Dji mavic 3 thermal specifications," 2024. Consultado en 2026.

[13] Autel Robotics, "Autel evo max 4t specifications," 2024. Consultado en 2026.

[14] Raptor Maps, "Solar asset management platform," 2025. Consultado en 2026.

[15] DroneDeploy, "Dronedeploy solar inspection solutions," 2025. Consultado en 2026.

[16] Pix4D, "Pix4dmapper and pix4dinspect," 2025. Consultado en 2026.

[17] Sitemark, "Automated solar pv thermography inspections," 2025. Consultado en 2026.

[18] marturiel00, "params_controller_pid.yaml in anafi_autonomy (branch ros2)." GitHub repository. Archivo de configuración consultado en la rama ros2 del repositorio anafi_autonomy.

[19] J. Seidel, "Drone thermography specification tables." LibreTexts Workforce Library, 2024. Consultado en 2026.

[20] Unidad Administrativa Especial de Aeronáutica Civil, "Rac 100: Operación de sistemas de aeronaves no tripuladas uas," 2023. Consultado en 2026.

[21] S. Tsanakas, L. Ha, and C. Buerhop, "Fault detection in photovoltaic modules using infrared thermography," *Renewable and Sustainable Energy Reviews*, vol. 62, pp. 695–709, 2016. [En línea]. Disponible: https://doi.org/10.1016/j.rser.2016.04.043.

[22] R. Szeliski, *Computer Vision: Algorithms and Applications*. Cham, Switzerland: Springer, 2nd ed., 2022. [En línea]. Disponible: https://link.springer.com/book/10.1007/978-3-030-34372-9.

[23] Raptor Maps, "Solar pv inspection software using uavs." Raptor Maps Inc., 2020. [En línea]. Disponible: https://www.raptormaps.com/solar-inspection-software.

[24] Sitemark, "Automated solar pv thermography inspections." Sitemark AG, 2023. [En línea]. Disponible: https://www.sitemark.com/solutions/solar-thermography-inspections.

[25] Pix4D, "Solar farm inspection with drones and thermal mapping." Pix4D SA, 2022. [En línea]. Disponible: https://www.pix4d.com/blog/solar-farm-inspection-thermal-mapping.

[26] DroneDeploy, "Thermal live map for drone inspections." DroneDeploy Inc., 2023. [En línea]. Disponible: https://www.dronedeploy.com/product/thermal/.

[27] DJI, "Mavic 3 thermal – specifications." DJI Technology Co., Ltd., 2023. [En línea]. Disponible: https://www.dji.com/mavic-3-enterprise/specs.

[28] Autel Robotics, "Evo ii dual 640t – specifications." Autel Robotics, 2023. [En línea]. Disponible: https://www.autelrobotics.com/products/evo-ii-dual-640t.

[29] B. Wang *et al.*, "Pvf-10: A high-resolution unmanned aerial vehicle thermal infrared image dataset for fine-grained photovoltaic fault classification," *Applied Energy*, vol. 376, p. 124187, 2024. [En línea]. Disponible: https://doi.org/10.1016/j.apenergy.2024.124187.

[30] Zenodo, “Thermal uav imagery dataset for photovoltaic inspection.” Zenodo Repository, 2025. [En línea]. Disponible: https://zenodo.org.

[31] OpenDroneMap Contributors, “Opendronemap documentation,” 2024. [En línea]. Disponible: https://www.opendronemap.org.

[32] QGIS Development Team, “Qgis geographic information system.” Open Source Geospatial Foundation, 2024. [En línea]. Disponible: https://www.qgis.org.

[33] Open Robotics, “Ros 2 documentation (humble hawksbill),” 2023. [En línea]. Disponible: https://docs.ros.org/en/humble/index.html.

Anexos

Anexo A. Instalación del sistema operativo

Es nesesario hacer una correcta instalacion del sistema operativo de Ubunto 22.04.2 LTS pra no tener inconvenientes futuros a la hora de instalar librerias, frameworks y aplicaciones que se utilizan para el correcto funcionamiento de la interfaz.

Paso 1. Descarga de la imagen ISO

Para iniciar la instalacion del sistema operativo es nesesario ingresar a la siguiente Url: https://cdimage.ubuntu.com/releases/jammy/release/inteliot/ que corresponde al repositorio oficial de imágenes de Ubuntu Jammy Jellyfish donde encontramos principalmente 2 imagenes ISO una para desktop y server, en este caso de estudio utlizamos la version desktop. En la Figura 6.3.1 se resalta la opción de descarga utilizada durante las pruebas del proyecto.

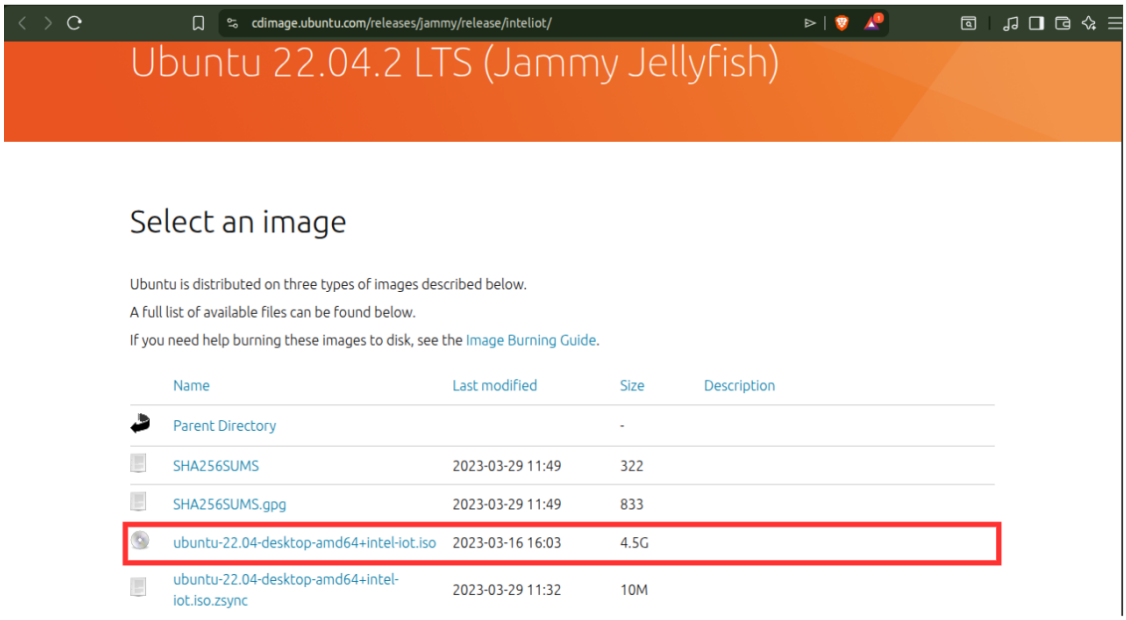


Figura 6.3.1: Descarga de Ubuntu 22.04.2 LTS. Se selecciona la imagen ISO para desktop. Fuente: elaboración propia.

Repositorio consultado: `https://cdimage.ubuntu.com/releases/jammy/release/inteliot/`.

Paso 2. Descarga de balenaEtcher

Teniendo la imagen ISO del sistema operativo se debe crear el USB booteable, se empleó *balenaEtcher*, el cual está disponible para distintos sistemas operativos y arquitecturas, se debe descargar la version que tenga el dispositivo en el cual se vaya a crear el USB booteable. La Figura 6.3.2 muestra la sección de descarga, donde se elige la opción correspondiente al equipo utilizado.

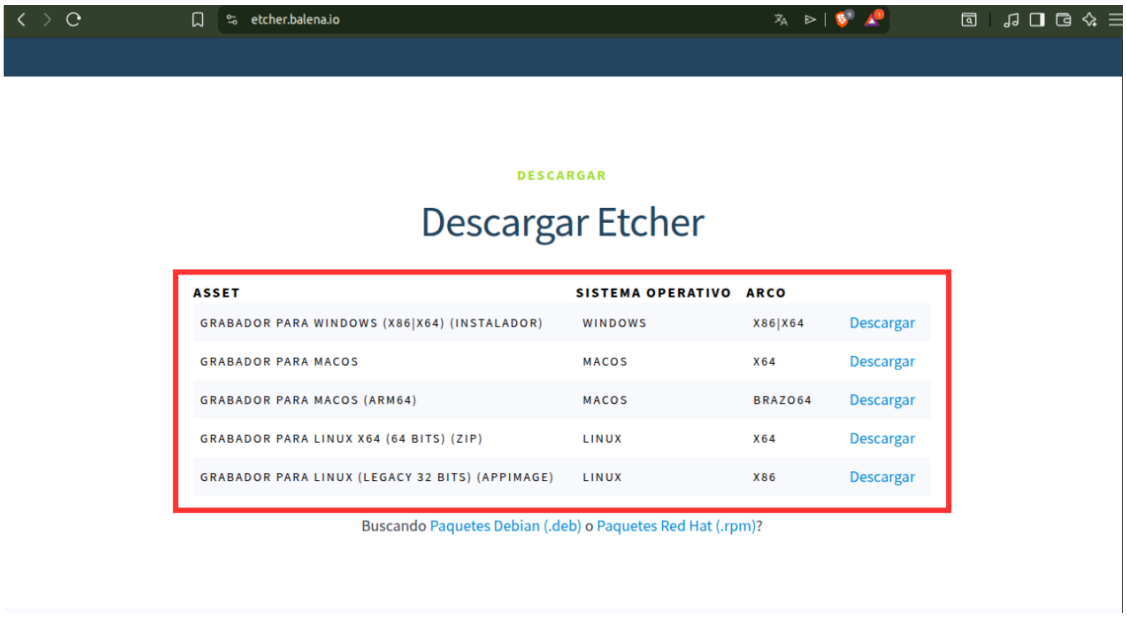


Figura 6.3.2: Opciones de descarga de balenaEtcher según sistema operativo y arquitectura. Fuente: elaboración propia.

Sitio oficial: `https://etcher.balena.io/`.

Paso 3. Creación del USB booteable (flasheo)

Una vez instalado y ejecutado balenaEtcher, se realiza el flasheo en tres acciones principales:

(a) Seleccionar la imagen ISO. Se presiona *Flash from file* y se busca el archivo .iso descargado de Ubuntu 22.04.2 LTS dentro de la carpeta de descargas (Figura 6.3.3).

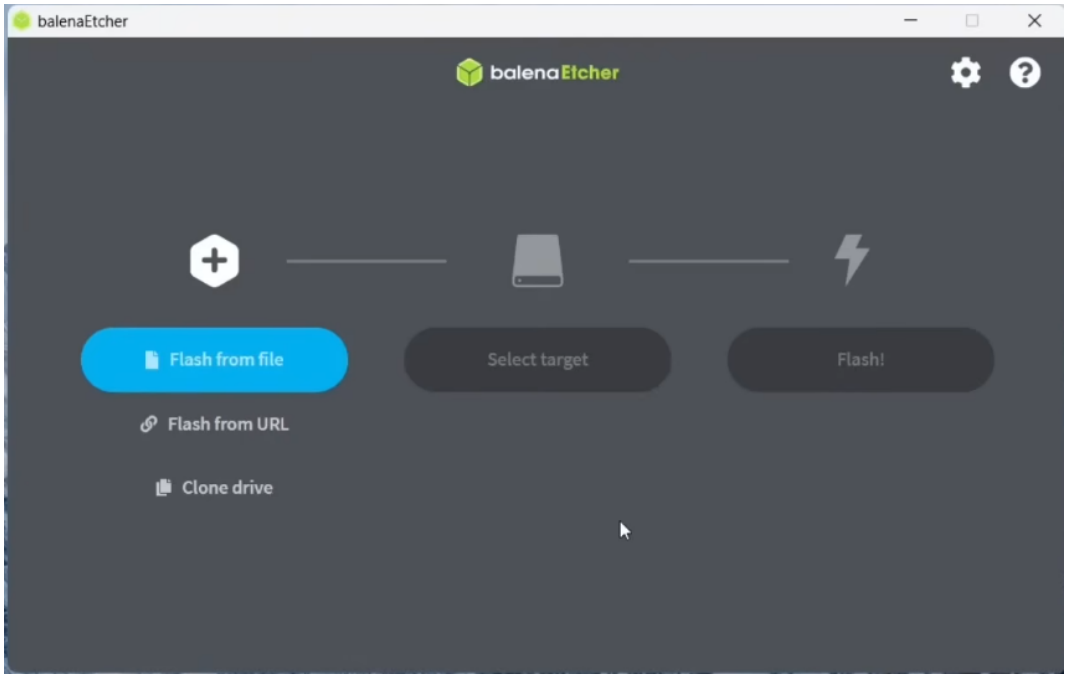


Figura 6.3.3: Selección del archivo ISO desde balenaEtcher. Fuente: elaboración propia.

(b) Seleccionar el dispositivo USB destino. Se presiona *Select target* y se elige la memoria USB que previamente debe ser formateada ya que se precisa de una USB totalmente vacia, de no ser asi el contenido previo sera totalemnte eliminado de la USB. (Figura 6.3.4).

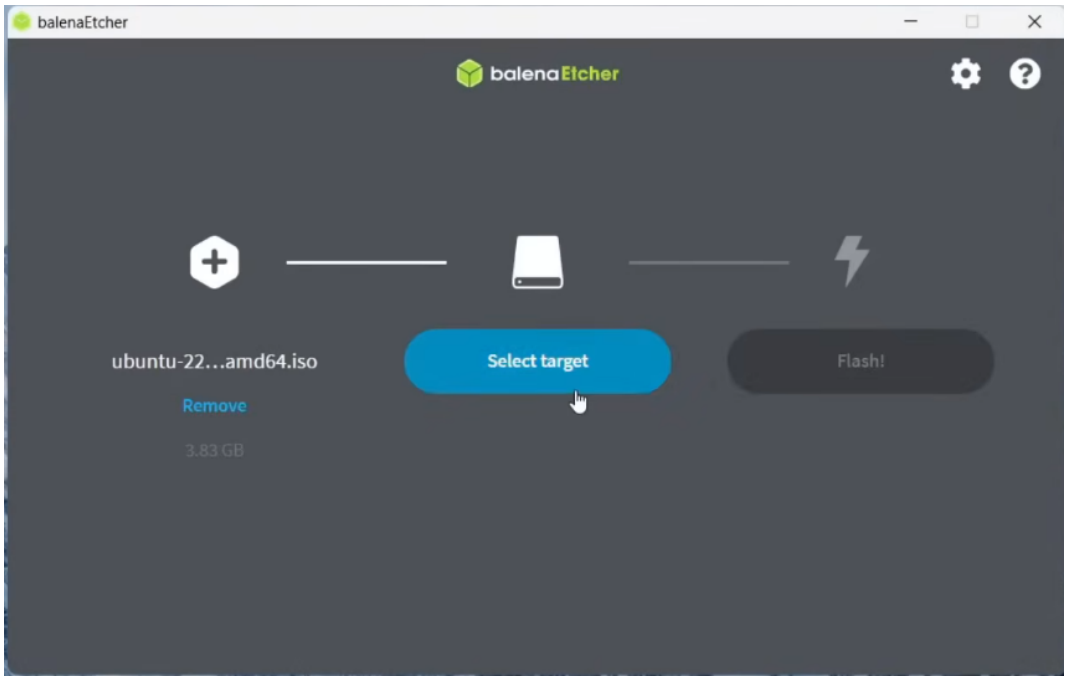


Figura 6.3.4: Selección de la memoria USB donde se escribirá la imagen. Fuente: elaboración propia.

(c) Iniciar el flasheo. Finalmente, se presiona el botón *Flash!*, generalmente este proceso puede tomar de 5 min a 20 min, Etcher realiza la escritura y luego una verificación automática.(Figura 6.3.5).



Figura 6.3.5: Inicio del flasheo de la ISO en la memoria USB. Fuente: elaboración propia.

Paso 4. Configuración de arranque (BIOS/UEFI)

Con el USB ya flasheado, el siguiente paso es iniciar el equipo desde esa memoria. Este procedimiento puede variar según la marca del computador (Lenovo, HP, Dell, ASUS, entre otros); sin embargo, en la mayoría de casos se ingresa a la BIOS/UEFI o al *Boot Menu* presionando una tecla al encender (por ejemplo: F2, F10, F12, ESC o DEL). En la Figura 6.3.6 se muestra un ejemplo de configuración donde se prioriza el arranque por USB.

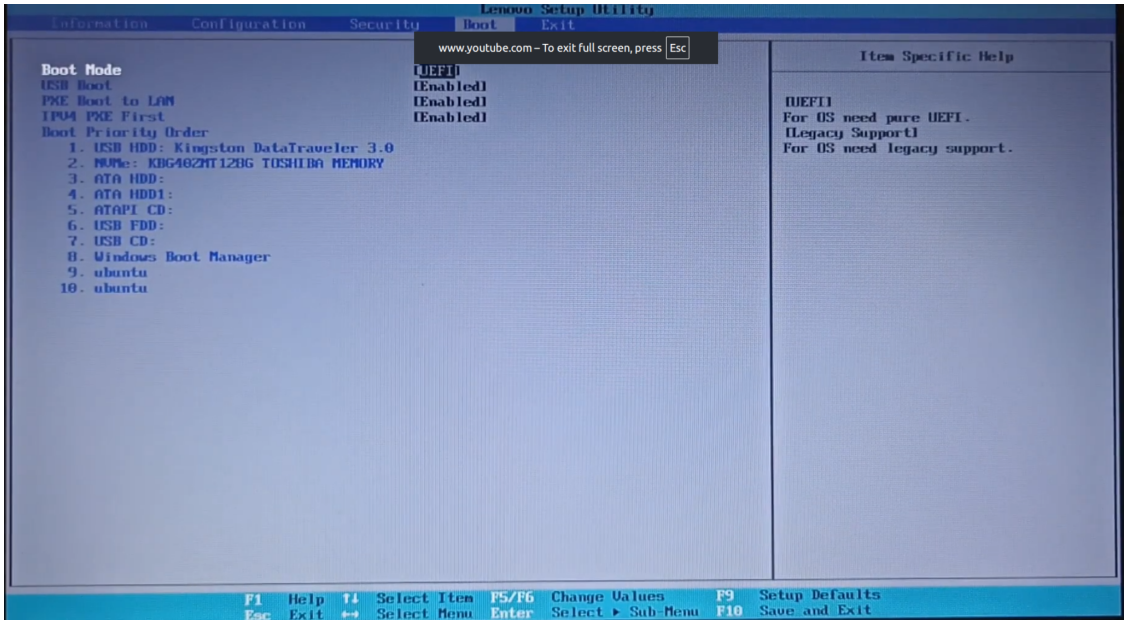


Figura 6.3.6: Ejemplo de configuración de arranque desde BIOS/UEFI. La ubicación de estas opciones varía según el fabricante. Fuente: elaboración propia.

Nota de compatibilidad: si el USB no aparece o no inicia, revisar *Secure Boot* (desactivar temporalmente si bloquea el arranque) y confirmar el modo de inicio (UEFI/Legacy).

Paso 5. Inicio desde el USB (menú GRUB)

Al arrancar desde la memoria USB, se mostrará el menú inicial de Ubuntu (GRUB). En la Figura 6.3.7 se observa la opción *Try or Install Ubuntu*, la cual carga el instalador del sistema.

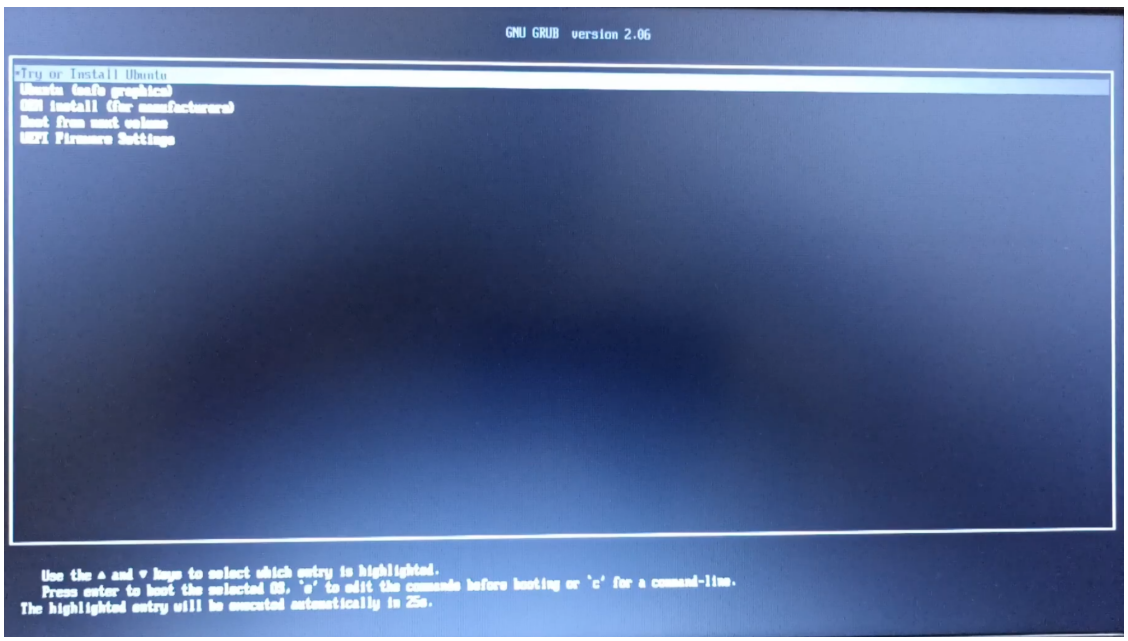


Figura 6.3.7: Inicio del instalador al arrancar desde USB. Se selecciona *Try or Install Ubuntu*. Fuente: elaboración propia.

Paso 6. Selección de idioma e inicio del instalador

Una vez cargado el entorno, se selecciona el idioma de preferencia y se elige la opción *Instalar Ubuntu*. La Figura 6.3.8 muestra la pantalla de bienvenida con ambas alternativas (*Probar Ubuntu* o *Instalar Ubuntu*).



Figura 6.3.8: Pantalla de bienvenida del instalador. Se selecciona el idioma y la opción *Instalar Ubuntu*. Fuente: elaboración propia.

Paso 7. Conexión a red inalámbrica

Para permitir la descarga de actualizaciones, la configuración automática de zona horaria y la instalación completa del paquete de idioma, se recomienda conectarse a una red Wi-Fi durante la instalación. La Figura 6.3.9 presenta el listado de redes disponibles y el proceso de conexión.

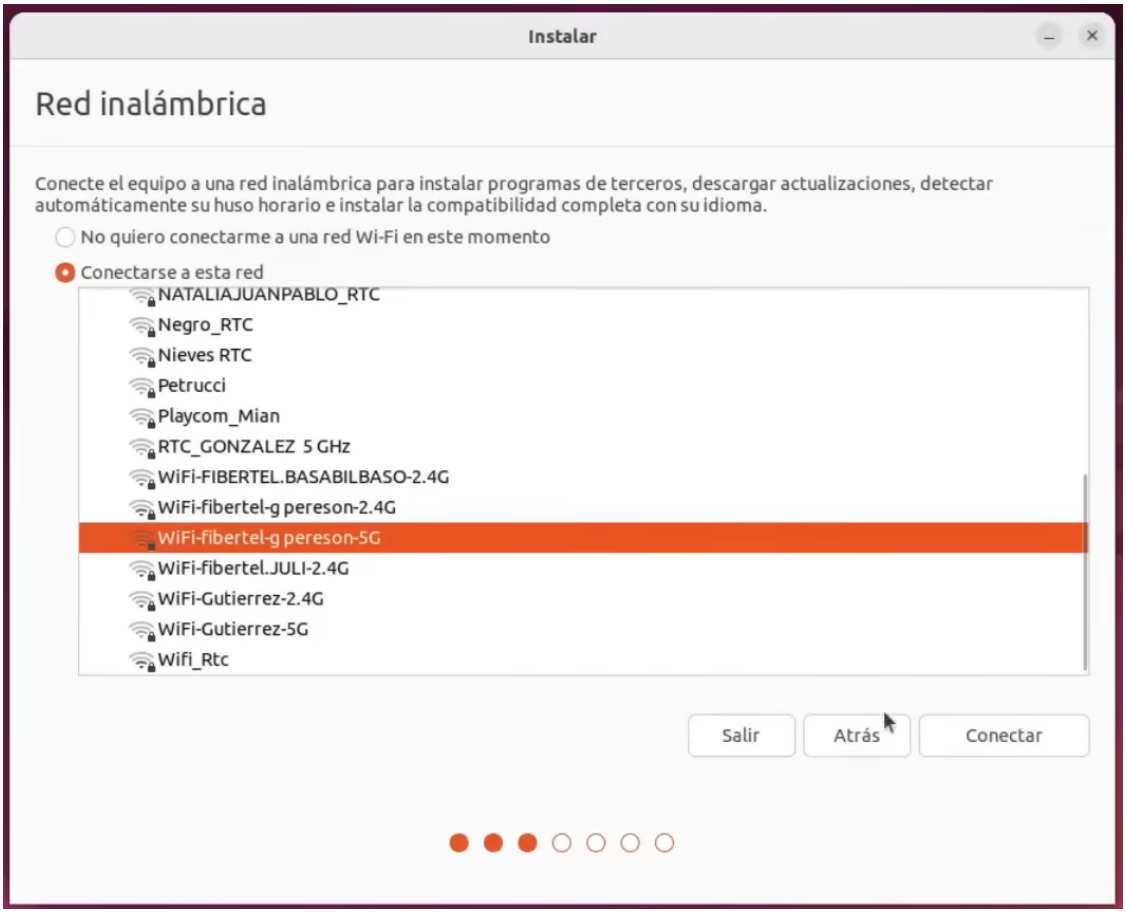


Figura 6.3.9: Conexión a una red Wi-Fi durante la instalación para habilitar descargas y configuración regional. Fuente: elaboración propia.

Paso 8. Selección de actualizaciones y software de terceros

En esta pantalla se define el perfil de instalación. Para un entorno de desarrollo (drivers, códecs y compatibilidad multimedia), se recomienda: (i) *Instalación normal*, (ii) descargar actualizaciones durante la instalación, y (iii) instalar software de terceros (drivers de gráficos, Wi-Fi y formatos multimedia), como se observa en la Figura 6.3.10.

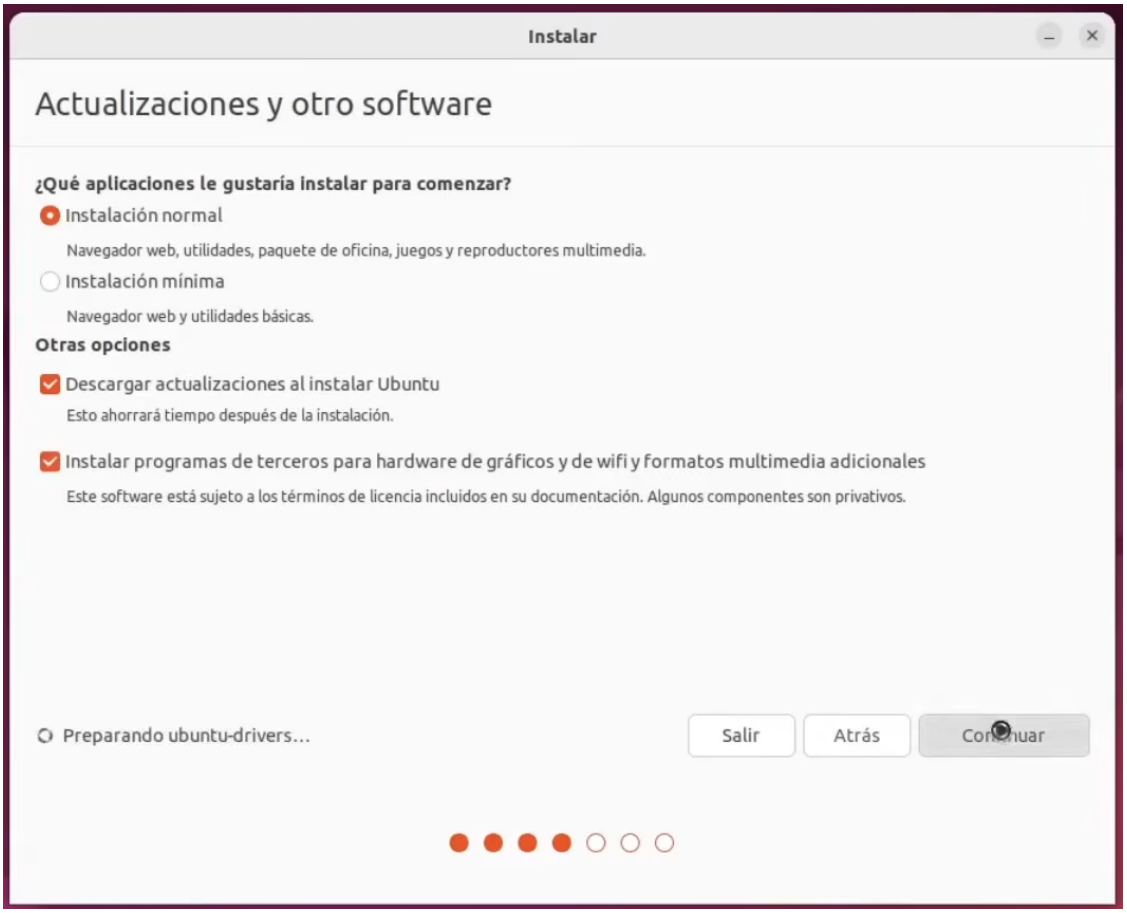


Figura 6.3.10: Selección de instalación normal, descarga de actualizaciones e instalación de software de terceros. Fuente: elaboración propia.

Paso 9. Tipo de instalación (Ubuntu único o Dual Boot)

El instalador permite distintos escenarios: instalar Ubuntu junto a un sistema existente (Dual Boot) o utilizar un único sistema operativo. Si el equipo se dedicará exclusivamente al proyecto, se recomienda *Borrar disco e instalar Ubuntu*. En caso de requerir conservar Windows, se puede seleccionar *Instalar Ubuntu junto a Windows Boot Manager*. La Figura 6.3.11 muestra estas alternativas.

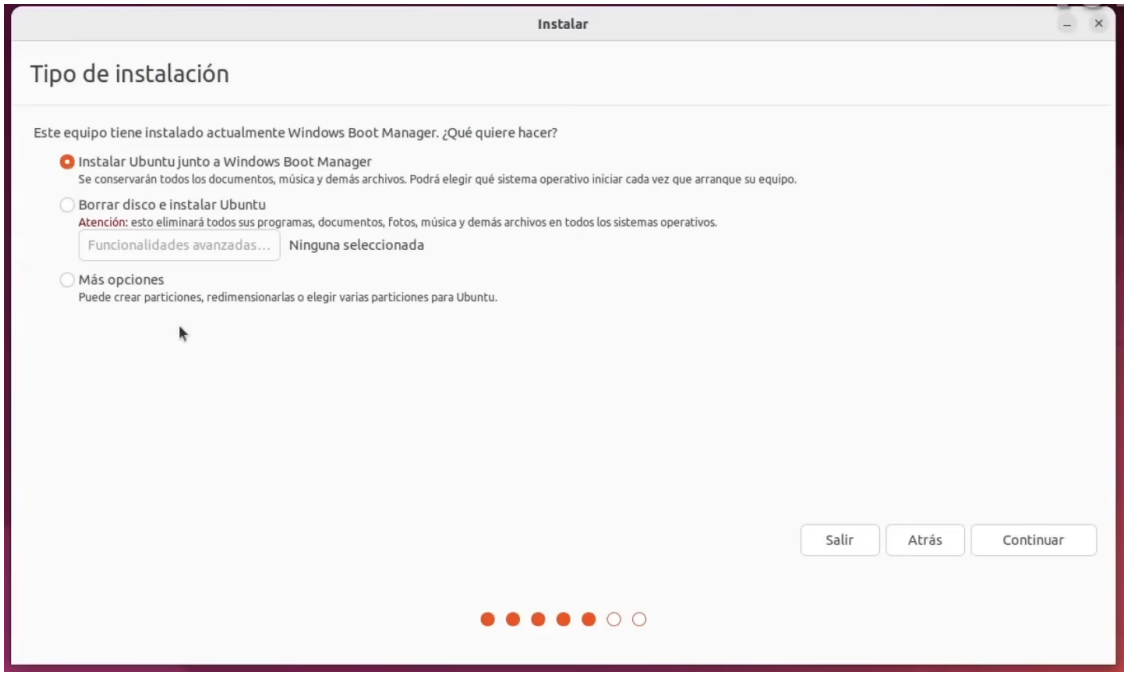


Figura 6.3.11: Selección del tipo de instalación: Ubuntu único o Dual Boot. Fuente: elaboración propia.

Advertencia: la opción de borrado elimina la información del disco. Se recomienda realizar copia de seguridad antes de continuar.

Paso 10. Creación del perfil de usuario

A continuación se crea el usuario principal del sistema (nombre, nombre del equipo, nombre de usuario y contraseña). Esta configuración habilita el inicio de sesión y permisos administrativos mediante sudo. La Figura 6.3.12 muestra un ejemplo de esta pantalla.

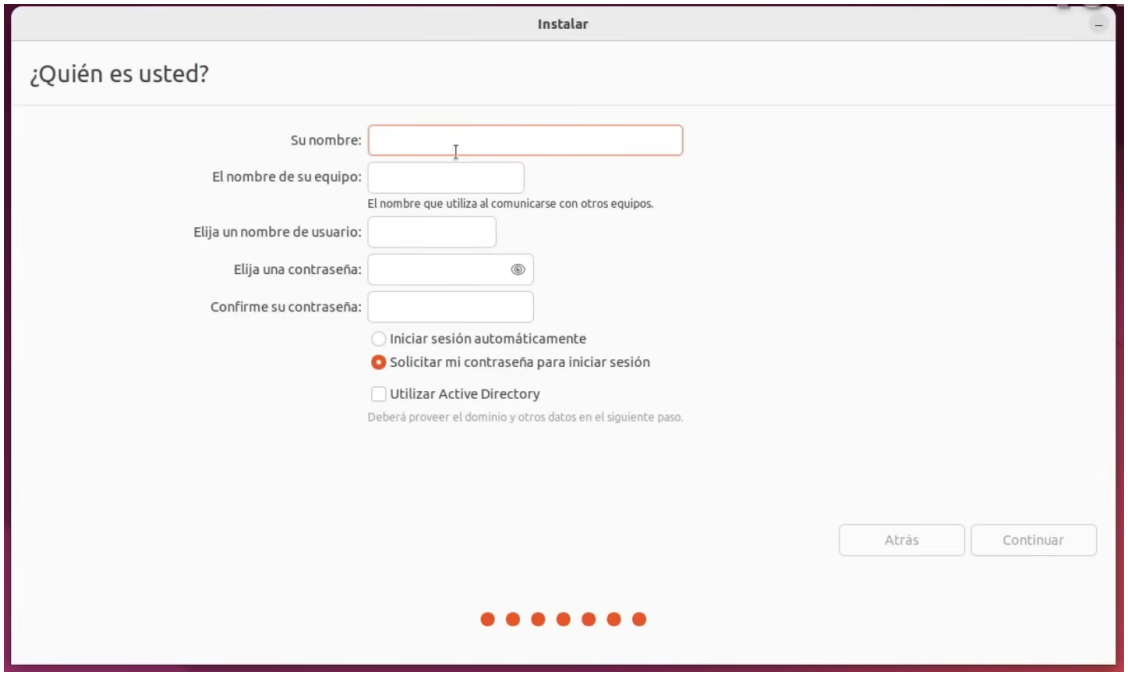


Figura 6.3.12: Creación del perfil de usuario durante la instalación de Ubuntu. Fuente: elaboración propia.

Paso 11. Proceso de instalación

Finalmente, el instalador copia archivos, aplica configuraciones y deja el sistema listo para reiniciar. En la Figura 6.3.13 se observa el progreso del proceso. Al finalizar, el asistente solicita reiniciar y retirar el USB.

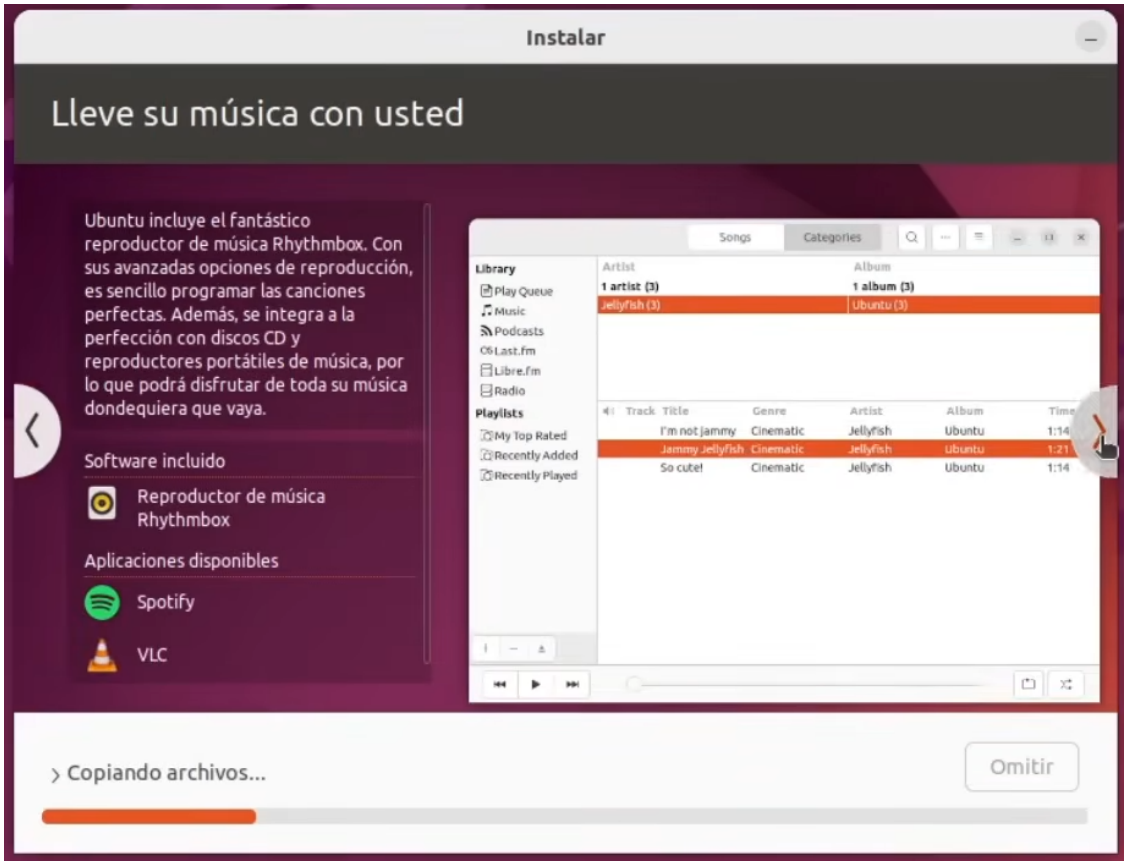


Figura 6.3.13: Proceso de instalación en ejecución. Al finalizar se solicita reiniciar y retirar el USB. Fuente: elaboración propia.

Anexo B. Repositorio con el proyecto y Plano del sistema fotovoltaico utilizado en las pruebas

https://github.com/juancryyy/Anafi$_{t}hermal_{v}ision_{s}ystem$

Anexo C. Secciones complementarias de instalación y configuración de la interfaz

C.1 Preparación del sistema

Con Ubuntu 22.04 actualizado, primero se instalan las dependencias base del sistema, Python y compilación necesarias para la interfaz, Olympe y el workspace ROS 2 Humble.

```
sudo apt update
sudo apt install -y locales software-properties-common curl git
   ffmpeg libgl1 python3-pip python3-venv python3-tk python3-
   rosdep python3-colcon-common-extensions build-essential cmake
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
```

Listing 6.1: Dependencias base del sistema.

C.2 Instalación de ROS 2 Humble

ROS 2 Humble mantiene soporte para Ubuntu 22.04 y su instalación recomendada en Linux continúa siendo mediante paquetes `deb`. La configuración del repositorio oficial y la instalación base se realiza con los comandos del Listado 6.2.

```
sudo add-apt-repository universe -y
sudo apt update
export ROS_APT_SOURCE_VERSION=$(curl -s https://api.github.com/
   repos/ros-infrastructure/ros-apt-source/releases/latest | grep
   -F 'tag_name' | awk -F '"' '{print $4}')
curl -L -o ros2-apt-source.deb "https://github.com/ros-
   infrastructure/ros-apt-source/releases/download/${
   ROS_APT_SOURCE_VERSION}/ros2-apt-source_${
   ROS_APT_SOURCE_VERSION}.$(. /etc/os-release && echo ${
   UBUNTU_CODENAME:-${VERSION_CODENAME}})_all.deb"
sudo dpkg -i ros2-apt-source.deb
sudo apt update
sudo apt install -y ros-humble-ros-base ros-dev-tools
sudo rosdep init
rosdep update
```

Listing 6.2: Instalación de ROS 2 Humble.

C.3 Construcción del workspace ROS 2

Una vez instalado ROS 2 y activado el entorno virtual, se resuelven dependencias declaradas en los paquetes del workspace y luego se compila con `colcon`.

```
cd anafi_suite_ros2
source opt/ros/humble/setup.bash
cd ros2_ws
rosdep install --from-paths src --ignore-src -r -y --rosdistro
   humble
colcon build
```

Listing 6.3: Resolución de dependencias y compilación del workspace ROS 2.

C.4 Verificación de conectividad con el dron

Antes de abrir la interfaz conviene validar conectividad con el dron o con el SkyController 3.

```
ping 192.168.53.1
ping 192.168.42.1
```

Listing 6.4: Verificación básica de conectividad.

C.5 Parámetros de configuración y atributos del sistema

Los parámetros de operación se centralizan en el archivo de config.py. La Tabla 6.1 resume los más relevantes.

El archivo de configuración se usa para adaptar la ejecución a diferentes escenarios de prueba sin modificar el código fuente. En particular, DRONE_IP y las URLs RTSP permiten alternar entre conexión directa al dron o conexión mediante SkyController, mientras que REFRESH_MS y SENSOR_HZ controlan el equilibrio entre fluidez visual y carga computacional de los hilos utilizados en la interfaz. Por último, parámetros como PATH_MAX_POINTS limitan el tamaño de la trayectoria mostrada en el gráfico 3D para mantener un uso estable de memoria durante vuelos prolongados, ya que luego de vuelos prolongados estos valores generan una mayor cagra computacional innecesaria.

Tabla 6.1: Parámetros principales del archivo config.py.

Parámetro	Valor típico	Descripción
DRONE_IP	192.168.53.1	Host objetivo para conexión Olympe. Alternativa directa: 192.168.42.1.
REFRESH_MS	500	Periodo de refresco (ms) para actualización de variables UI.
SENSOR_HZ	20	Frecuencia del hilo de lectura de telemetría en sensores/streams.py.
DRONE_RTSP_URL	rtsp://.../live	URL RTSP principal para stream RGB en interfaz/pages/stream_page.py.
RTSP_TRANSPORT	udp	Transporte RTSP.
PATH_MAX_POINTS	2000	Máximo de puntos para trayectoria 3D en SensoresPage.

Anexo D. Preconfiguración del sistema

Durante las pruebas realizadas se observó que la ejecución autónoma de misiones con el dron no depende únicamente de la validez geométrica de los waypoints, sino también de un conjunto de restricciones prácticas asociadas al comportamiento físico de la plataforma y a la forma en que Olympe acepta y ejecuta los comandos de movimiento, en este sentido, aunque la misión sea descrita desde la GUI como una secuencia continua de desplazamientos relativos, en operación real existen límites temporales y espaciales que deben considerarse para evitar fallos de ejecución, rechazos de comandos o degradación en la calidad de la captura.

Una de las limitaciones más evidentes aparece cuando se programan desplazamientos demasiado largos dentro de un mismo waypoint, en estos casos, la ejecución de un único movimiento extenso puede resultar menos robusta que una trayectoria equivalente dividida en tramos más pequeños. Por esta razón, dentro de la lógica de misión se adoptó la segmentación automática de recorridos largos, de modo que un desplazamiento grande pueda ejecutarse como una sucesión de movimientos relativos más cortos. Esta decisión mejora la estabilidad de la misión y reduce la probabilidad de error durante el seguimiento de la trayectoria. No obstante, también incrementa el tiempo total de ejecución, por lo que una alternativa práctica para compensar este efecto consiste en ajustar adecuadamente la velocidad de operación del dron, siempre que dicha velocidad siga siendo compatible con la estabilidad del vuelo y con la calidad de las capturas requeridas.

Otra consideración operativa importante está relacionada con el gimbal. Cuando la misión requiere modificar el ángulo de inclinación de la cámara, no resulta conveniente iniciar de inmediato la captura o el siguiente desplazamiento justo después del comando de reposicionamiento. En las pruebas se observó que es recomendable incorporar un tiempo de espera inicial o de estabilización de minimo 5 segundos luego de variar el pitch del gimbal, con el fin de permitir que la cámara alcance su nueva orientación de manera estable antes de registrar información térmica o visual, de lo contrario, la imagen obtenida puede verse afectada por transitorios mecánicos, pequeñas oscilaciones o desajustes momentáneos en la orientación de la escena.

De forma similar, el comportamiento del dron frente a la posición debe interpretarse como una estabilización relativa y no como una permanencia absolutamente rígida en un punto fijo del espacio. Durante estados de espera o de captura, el dron mantiene una posición suficientemente estable para la operación prevista; sin embargo, esta puede presentar pequeñas desviaciones debidas a perturbaciones externas, entre ellas ráfagas de viento, irregularidades del entorno o variaciones propias de la estimación del movimiento. Aunque el control interno del dron corrige una parte importante de estas perturbaciones para conservar una actitud estable y una referencia espacial razonable, en condiciones de viento fuerte o disturbios persistentes pueden presentarse desviaciones que el comportamiento equivalente al PID embebido no logra compensar

completamente. Esto repercute tanto en la repetibilidad geométrica de la misión como en la consistencia de la captura, especialmente cuando se trabaja a baja altura o con sensores de resolución limitada como es el caso. De igual forma, durante las pruebas se observó que la condición previa del dron afecta directamente la ejecución del despegue y, por ende, la capacidad de realizar misiones automáticas de manera confiable. Por esta razón, la calibración del magnetómetro se asumió como una tarea de preparación del sistema y se realizó mediante la aplicación oficial *FreeFlight 6* de Parrot. En esta herramienta, el procedimiento exige girar el dron sobre sus tres ejes principales, siguiendo la secuencia indicada por la aplicación, tal como se ilustra en la Figura 6.3.14 y en la Figura 6.3.15. Cuando el dron no aceptaba correctamente el comando de *takeoff* o cuando se detectaban inconsistencias en la referencia espacial, se identificó que una causa frecuente correspondía a una calibración deficiente del magnetómetro.

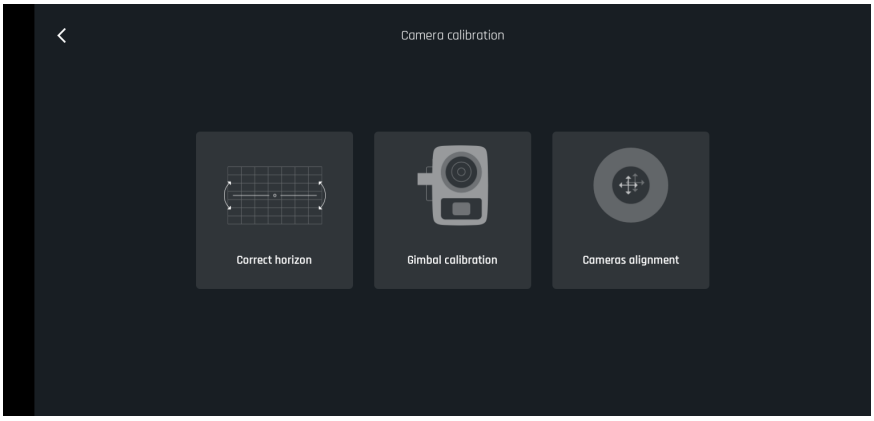


Figura 6.3.14: Menú de calibración disponible en la aplicación oficial *FreeFlight 6*, donde se presentan las opciones de corrección de horizonte, calibración de gimbal y calibración del magnetómetro. Fuente: elaboración propia.

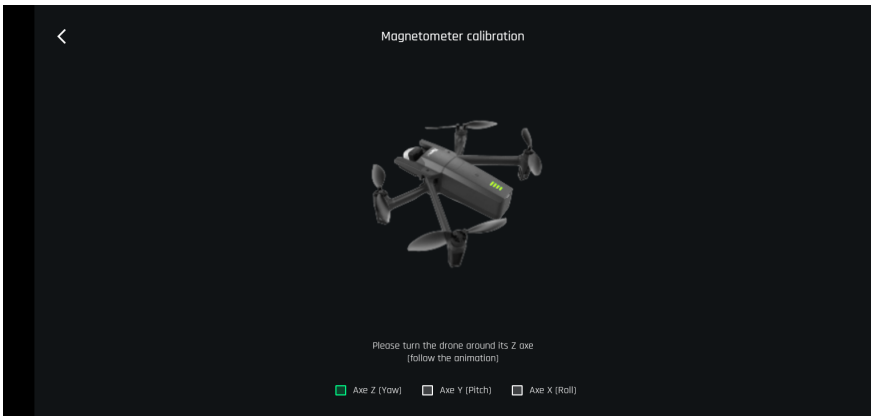


Figura 6.3.15: Procedimiento de calibración manual del magnetómetro, en el cual el dron debe girarse siguiendo los tres ejes indicados por la aplicación para restablecer una referencia espacial consistente. Fuente: elaboración propia.

De manera complementaria, la calibración del gimbal, la corrección del horizonte y la alineación entre las cámaras visible y térmica también se mantuvieron dentro del ecosistema oficial del fabricante. Esta decisión permitió trabajar sobre una plataforma

previamente acondicionada para vuelo, dejando a la suite desarrollada la responsabilidad de la operación y la orquestación de la misión, pero no de la calibración física del vehículo. Como se muestra en la Figura 6.3.16, la corrección manual del horizonte permite ajustar la referencia visual de la cámara frente a pequeñas desviaciones en la horizontalidad percibida, mientras que la Figura 6.3.17 muestra el proceso de calibración del gimbal, en el cual la cámara realiza movimientos controlados sobre sus ejes para restablecer su orientación mecánica. Esta separación entre preparación del dron y orquestación *software* favorece la repetibilidad experimental y evita atribuir a la GUI problemas que en realidad corresponden a la condición física, mecánica o de calibración de la plataforma aérea.

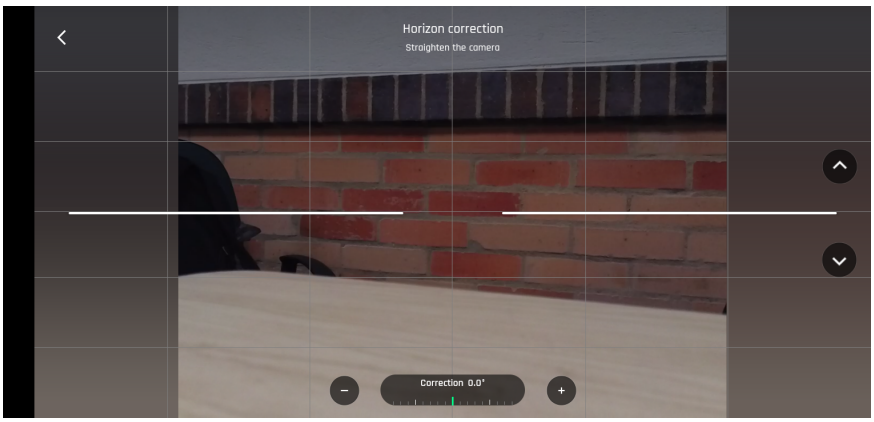


Figura 6.3.16: Interfaz de corrección manual del horizonte, utilizada para ajustar la referencia visual de la cámara y compensar desviaciones en la alineación horizontal de la escena. Fuente: elaboración propia.

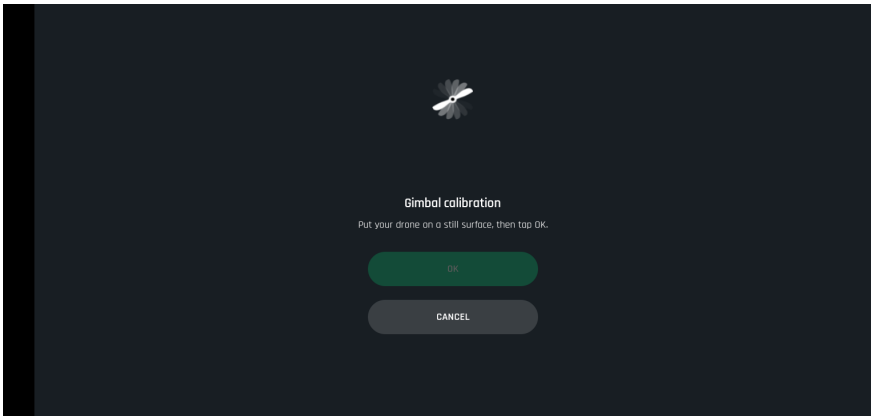


Figura 6.3.17: Proceso de calibración del gimbal, en el cual la cámara ejecuta movimientos automáticos o asistidos sobre sus ejes con el fin de restablecer su orientación mecánica. Fuente: elaboración propia.

Anexo E. Registro detallado del entrenamiento del modelo OBB

Este anexo contiene el enlace al documento complementario donde se registra la evolución del proceso de entrenamiento del modelo de detección OBB. En dicho documento se incluyen las métricas obtenidas durante las fases de entrenamiento y validación, las funciones de pérdida, los indicadores de desempeño y los parámetros asociados al proceso de optimización.

Enlace al documento: `https://docs.google.com/spreadsheets/d/14lzVezV-goEKQJiATEO__fAUw4vO5znpY61x4BX6giA/edit?usp=sharing`

Anexo F. Ejemplo de datos almacenados en una misión de vuelo

Este anexo contiene un ejemplo de los datos almacenados en un escenario real de funcionamiento de una misión de vuelo. El archivo CSV contiene los datos de vuelo, sensores, GPS, configuración de cámara y ubicación de los archivos descargados desde el dron.

Enlace al documento: `https://docs.google.com/spreadsheets/d/1IpAPHWJmi9o_g33FtsJjbrq62AD6cgp8hlxRhZtwJM8/edit?usp=sharing`