UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

DESARROLLO DE UN SISTEMA DE DETECCIÓN DE INCENDIOS FORESTALES BASADO EN PROCESAMIENTO DIGITAL DE IMÁGENES CON UNA CÁMARA ESTACIONARIA EN EL SECTOR DE PICOL ORCCOMPUCYO DEL DISTRITO DE SAN JERÓNIMO-CUSCO

PRESENTADO POR:

Br. EDUARDO ELI CONDORHUAMAN QUISPE

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:

MSc. LUIS JIMÉNEZ TRONCOSO

CUSCO - PERÚ

2026

Universidad Nacional de San Antonio Abad del Cusco

INFORME DE SIMILITUD

(Aprobado por Resolución Nro.CU-321-2025-UNSAAC)

El que suscribe, el **Asesor** .....Luis Jiménez Troncoso..... ..... quien aplica el software de detección de similitud al trabajo de investigación/tesis titulada: ..." DESARROLLO DE UN SISTEMA DE DETECCIÓN DE INCENDIOS FORESTALES BASADO EN PROCESAMIENTO DIGITAL DE IMÁGENES CON UNA CÁMARA ESTACIONARIA EN EL SECTOR DE PICOL ORCCOMPUCYO DEL DISTRITO DE SAN JERÓNIMO-CUSCO"

Presentado por: ...EDUARDO ELI CONDORHUAMAN QUISPE. DNI N° ..... 70412279 .....;

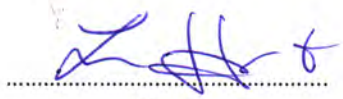
Para optar el título Profesional/Grado Académico de .....INGENIERO ELECTRÓNICO..... Informo que el trabajo de investigación ha sido sometido a revisión por ..... 2 .... veces, mediante el Software de Similitud, conforme al Art. 6° del ***Reglamento para Uso del Sistema Detección de Similitud en la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 7**%.**

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No sobrepasa el porcentaje aceptado de similitud.	X
Del 11 al 30 %	Devolver al usuario para las subsanaciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, conforme al reglamento, quien a su vez eleva el informe al Vicerrectorado de Investigación para que tome las acciones correspondientes; Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de Asesor, firmo el presente informe en señal de conformidad y **adjunto** las primeras páginas del reporte del Sistema de Detección de Similitud.

Cusco, 19 de enero de 2026



.....

Firma

Post firma...LUIS JIMÉNEZ TRONCOSO....

Nro. de DNI: 08275751.

ORCID del Asesor : 0000-0001-6414-9742

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema de Detección de Similitud: **oid: 27259:547399331**

Eduardo Eli Condorhuaman

tesis_Eduardo_Condorhuaman_con observaciones levantadas.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:547399331

Fecha de entrega
19 ene 2026, 12:43 p.m. GMT-5

Fecha de descarga
19 ene 2026, 12:48 p.m. GMT-5

Nombre del archivo
tesis_Eduardo_Condorhuaman_con observaciones levantadas.pdf

Tamaño del archivo
15.4 MB

230 páginas

36.691 palabras

205.686 caracteres

7% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Trabajos entregados

Fuentes principales

7%	Fuentes de Internet
2%	Publicaciones
0%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alerta de integridad para revisión

Texto oculto
647 caracteres sospechosos en N.º de páginas
El texto es alterado para mezclarse con el fondo blanco del documento.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Presentación

Estimado Dr. Lauro Enciso Rodas, decano de la Facultad de Ingeniería Eléctrica, Electrónica, Mecánica e Informática y honorables miembros del jurado evaluador.

Tengo el honor de presentar ante ustedes los resultados del proyecto de ingeniería, realizado en cumplimiento con los requisitos para optar al título profesional de ingeniero electrónico. Esta tesis, titulada "DESARROLLO DE UN SISTEMA DE DETECCIÓN DE INCENDIOS FORESTALES BASADO EN PROCESAMIENTO DIGITAL DE IMÁGENES CON UNA CÁMARA ESTACIONARIA EN EL SECTOR DE PICOL ORCCOMPUCYO DEL DISTRITO DE SAN JERÓNIMO-CUSCO", constituye un aporte para la seguridad y la conservación de nuestros recursos forestales.

El principal aporte del proyecto de ingeniería reside en la viabilidad técnica y social de la solución, utilizando dispositivos de bajo costo y herramientas computacionales de libre acceso. Asimismo, esta tesis documenta los resultados obtenidos a través de pruebas realizadas en campo, específicamente en la comunidad campesina de Picol Orccompucyo, demostrando su funcionalidad y las dificultades inherentes que se pueden encontrar en un despliegue real.

Convencido de que la aplicación estratégica de la tecnología accesible es una herramienta poderosa para proteger a las comunidades y sus recursos naturales, presento esta tesis para su consideración y evaluación.

Dedicatorias

A mi padre Isaac Condorhuaman, quien hoy vela por mi persona desde el cielo, y a mi madre María Quispe por su apoyo incondicional, paciencia, esfuerzo y cariño, que me ayudaron en este extenso proceso de formación académica. Quiero dedicar este trabajo a ustedes, porque este logro también es suyo.

A mis hermanas Gaby, Raquel y mi pequeña sobrina Samy, que son el apoyo emocional en situaciones difíciles de la vida y también por ser esa fuerza silenciosa y motivadora para cumplir mis objetivos.

A mis familiares en general tíos abuelos, tíos y primos por brindarme el apoyo moral y académico en este proceso de formación. Cada uno puso una parte, para este logro y siempre los tendré presente en mi corazón.

Eduardo Eli Condorhuaman Quispe

Agradecimientos

Expreso mi profundo agradecimiento a la comunidad campesina de Picol Orccompucyo, por permitir realizar las pruebas de campo en sus espacios territoriales, demostrando una disposición abierta con el proyecto de ingeniería. Además, expreso un reconocimiento al presidente de la comunidad Ing. Jose Tisoc, por proporcionarme las facilidades necesarias y por actuar como el intermediario entre mi persona y los comuneros de la zona.

A mis docentes por las enseñanzas impartidas y por compartir sus conocimientos. Además, quiero agradecer a mi asesor Mgt. Ing. Luis Jiménez, por haber aceptado acompañarme en el proceso de mi tesis, brindándome su tiempo y disposición

A mi compañero de aula y amigo Jhon Suma, por brindarme su tiempo y predisposición durante la etapa más complicada de mi tesis. Tu apoyo en las pruebas fue crucial para hacer esto posible.

Resumen

Hoy en día, los incendios forestales son una problemática que afecta a todo el mundo y en especial a la región del Cusco, la cual tiene el mayor número de emergencias históricas documentadas. Estos fuegos no controlados son en su mayoría generados de manera antrópica, causando la pérdida de grandes cantidades de flora y fauna.

Para abordar esto, el siguiente proyecto de ingeniería diseñó e implementó un prototipo para detectar humo (día) y fuego (noche). El prototipo usa una cámara fija para obtener video en tiempo real, procesado mediante algoritmos implementados en Python con librerías como OpenCV. Las principales técnicas son: detección de movimiento utilizando sustracción de fondo KNN, análisis mediante espacios de color HSV y técnicas de confirmación de eventos críticos, culminando con la notificación usando mensajería instantánea.

El hardware para el procesamiento del video es el sistema embebido Raspberry Pi 5 (8 GB de RAM). Para la adquisición de video se usaron dos cámaras: la Raspberry Pi Cam V3 (Wide) y la cámara IP Tapo C320WS. Las pruebas se realizaron simulando humo y fuego en el sector de Picol Orccompucyo, zona constantemente afectada por incendios.

Los resultados del prototipo demuestran: 73.9% de precisión, 85% de sensibilidad y 79% de balance general usando la cámara Raspberry Pi V3. Con el segundo modelo (Tapo C320WS), se obtuvo 76.9% de precisión, 100% de sensibilidad y 86.9% de balance general.

Palabras clave: Incendios forestales, Procesamiento digital de imágenes, Raspberry Pi, Algoritmos.

Abstract

Today, wildfires are a global issue, especially impacting the Cusco region of Peru, which has one of the highest numbers of documented historical emergencies. Most of these uncontrolled fires are anthropogenic (human-caused), leading to significant loss of flora and fauna.

To tackle this problem, this engineering project designed and implemented a prototype system to detect smoke (during the day) and fire (at night). The prototype uses a fixed camera to capture real-time video, which is processed using algorithms implemented in Python with libraries like OpenCV. Key techniques include: motion detection using KNN background subtraction, HSV color space analysis, and critical event confirmation, concluding with instant messaging notification.

The core processing hardware is an embedded Raspberry Pi 5 8 GB. For video acquisition, two cameras were used: the Raspberry Pi Cam V3 (Wide) and the Tapo C320WS IP camera. Testing involved simulating smoke and fire in the Picol Orccompucyo sector, an area constantly affected by wildfires.

The system's results demonstrate strong performance: 73.9% precision, 85% sensitivity, and 79% overall balance with the Raspberry Pi V3 camera. With the second model (Tapo C320WS), performance was even better, achieving 76.9% precision, 100% sensitivity, and 86.9% overall balance.

Keywords: Wildfires, Digital image processing, Raspberry Pi, Algorithms.

Introducción

El presente proyecto de ingeniería está estructurado en seis capítulos:

Capítulo I: Explica los aspectos generales del proyecto, enfatizando la problemática de los incendios forestales, desde el impacto global hasta la aplicación en la comunidad de Picol Orccompucyo. Se definen los objetivos, justificación, alcances y limitaciones del proyecto.

Capítulo II: Presenta el marco teórico, describiendo antecedentes de estudio y conceptos clave, para comprender los principios y tecnologías usadas.

Capítulo III: Describe la etapa de diseño del prototipo, destacando los requerimientos para la elección y desarrollo de hardware y software. También se justifican las decisiones tomadas en el diseño.

Capítulo IV: Describe la implementación del prototipo, cubriendo el ensamblado de componentes físicos, la codificación de algoritmos y la verificación de funcionamiento.

Capítulo V: Muestra las pruebas y los resultados obtenidos. Primero, presenta los resultados en condición simulada (videos de prueba), para finalizar con los resultados de las pruebas reales en el sector de Picol Orccompucyo.

Capítulo VI: Se realiza el análisis de costos de implementación del prototipo del sistema detector de incendios forestales.

El proyecto de ingeniería demuestra una propuesta innovadora en la que se utilizan equipos de bajo costo y herramientas computacionales de libre acceso, lo que facilitaría su implementación en distintos puntos vulnerables de la región.

Índice

Índice de Figuras

Índice de Tablas

Capítulo I: Aspectos Generales

1.1. Marco Referencial

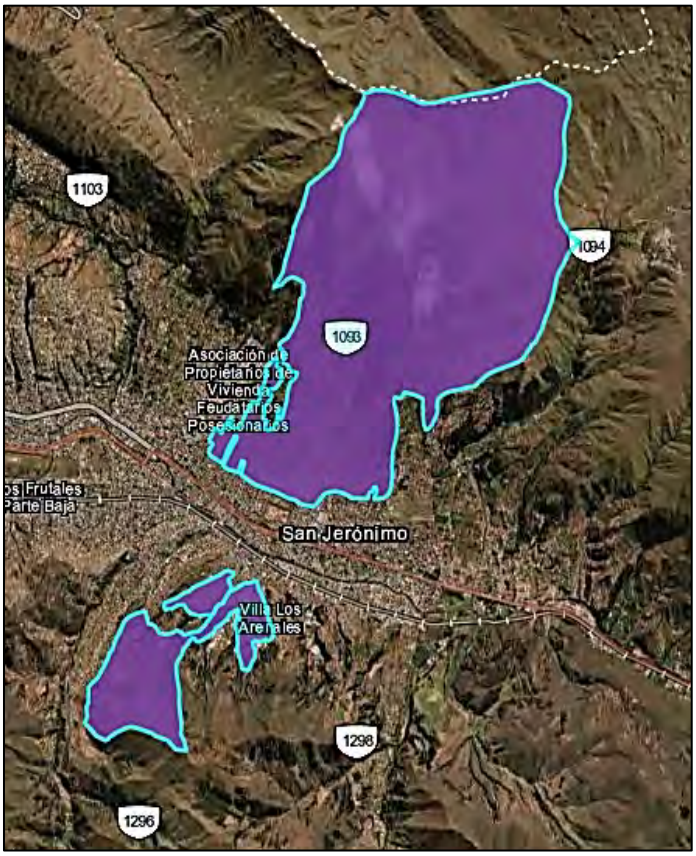
1.1.1. Ámbito de Aplicación

La comunidad campesina de Picol Orccompucyo pertenece al distrito de San Jerónimo, provincia de Cusco.

Cuenta con una extensión territorial de 1510.00 hectáreas siendo la organización territorial campesina más grande del distrito de San Jerónimo. Presenta una geomorfología del piso ecológico que comprende valles y laderas. Así mismo, presenta ecosistemas como zonas agrícolas, plantaciones forestales y pajonal de puna húmeda. (MD San Jeronimo, 2022).

Figura 1

Extensión territorial de la comunidad campesina de Picol Orccompucyo.



Fuente: Base de Datos de Pueblos indigenas u Originarios, s.f.

1.2. Planteamiento del Problema

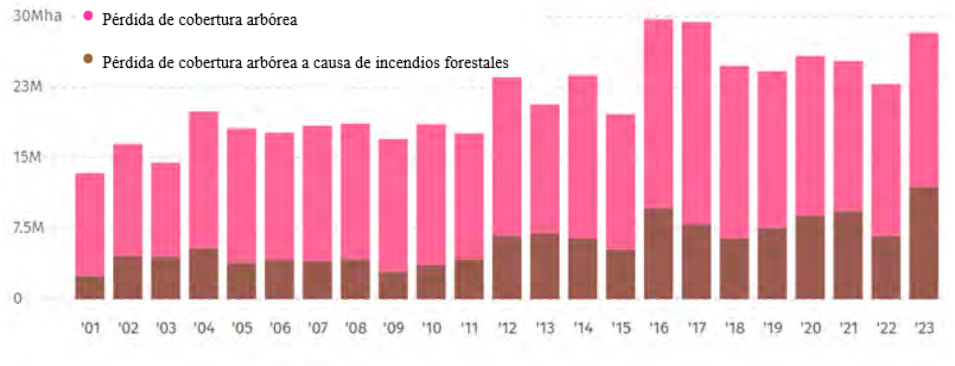
1.2.1. Situación Problemática

En la actualidad, los diferentes medios de comunicación reportan ocurrencias de incendios forestales de grandes proporciones en diferentes partes del planeta.

La plataforma en línea Global Forest Watch registró, durante el periodo de 2001 a 2023 un total de 138 Mha de cobertura arbórea extinta debido a los incendios forestales en todo el mundo. El año 2023 fue el periodo con la mayor disminución de cobertura arbórea, con un total de 11.9 Mha de territorio forestal devastado.

Figura 2

Pérdida de cobertura arbórea mundial en el periodo de 2001 a 2023



Nota. La figura 3 representa la pérdida global por año de la cobertura arbórea a causa de los incendios forestales. Tomado de Vizzuality, s.f.

En América Latina, la problemática es significativa debido a las consecuencias que conlleva la destrucción de los recursos naturales en toda la región. En Sudamérica, las incidencias han ido aumentando en mayor cantidad año tras año y se evidencia en los reportes que realizan distintas organizaciones de supervisión forestal.

El Instituto Brasileño de Investigaciones Espaciales (INPE), que emplea imágenes de satélite para calcular la cantidad de focos de calor en toda Sudamérica, registró hasta septiembre

del año 2024 un total de 409 099 incendios forestales, cifra que supera los totales anuales de los últimos tres años (EFE, 2024).

En Perú, los incendios forestales también constituyen una problemática. En 2023 se registró una cifra aproximada de 1432, superando las cifras anuales de los tres años anteriores a este. En el último reporte del MINAM (Ministerio del Ambiente), en 2024 se registró un total de 2 500 incendios forestales a nivel nacional. (CooperAcción, 2024)

En el aspecto local, el INDECI (Instituto Nacional de Defensa Civil) identificó al Cusco para el periodo de 1995 al 2015, como el departamento peruano con mayor registro de incendios forestales. La temporada de mayor riesgo va de junio a noviembre, teniendo su pico más alto en el mes de agosto" (Lau, 2017). Actualmente, el departamento del Cusco es considerado como la región con mayores incidencias de incendios forestales, registrando un total de 277 eventos en lo que va del año 2024 (Agencia Peruana de Noticias, 2024)

En la comunidad de Picol Orccompucyo se identificaron 28 hectáreas de área forestal destruidas por cuatro incendios forestales de gran magnitud y tres de moderada magnitud en el año 2020 durante la temporada seca. (Suarez, 2021). Actualmente, la comunidad campesina sigue siendo afectada por los incendios forestales. Según el último registro del COER, en agosto del 2024 se produjo un incendio forestal de gran magnitud, el cual afectó a comunidades cercanas (Diario el Sol del Cusco, 2024).

Aunque la problemática son los incendios forestales en toda la región del Cusco, el problema principal que abarcará el proyecto de ingeniería es la ausencia de sistemas de detección temprana ante incendios forestales en el sector de Picol Orccompucyo.

En la actualidad existen sistemas de alerta temprana ante incendios forestales implementadas alrededor de todo el planeta, utilizando distintos tipos de tecnologías como

estaciones terrenas de cámaras, drones y tecnología satelital. Incluso, algunos son soluciones comerciales, como ADELIE® de la empresa PARATRONIC e INSIGHTFD3 de la empresa INSIGHT ROBOTICS, los cuales son estaciones terrenas de cámaras visuales e infrarrojas con capacidad de realizar detecciones de eventos a grandes distancias (15 – 20 km). Otro producto comercial es el dron MAVIC 3 THERMAL de la empresa DJI, el cual oferta un dron especializado para estas tareas. Por supuesto, todas estas soluciones anteriormente mencionadas conllevan a un alto costo económico de compra e implementación. En el caso de INSIGHTFD3, puede llegar hasta los 101 730.77 USD **(**incluidos impuestos y por unidad de cámara) (Powertec Wireless Technology NZ, s.f.). Por otra parte, según la cotización del dron MAVIC 3 THERMAL, puede llegar a tener un costo de S/. 19 199.00. (Drone Solution Center, s.f.)

"El SERFOR (Servicio Nacional Forestal y de Fauna Silvestre) desde el año 2017, viene implementando la Unidad de Monitoreo Satelital (UMS) que genera reportes de focos de calor con fines de alerta, el cual realiza la detección y monitoreo de los incendios forestales" (Servicio Nacional Forestal y de Fauna Silvestre, 2019).

Según una reciente investigación realizada por el MINAM, IGP y el SERFOR en base a la efectividad de los satélites en la detección de focos de calor en la superficie terrestre de los andes del Cusco, esta revela que se tiene un nivel alto de confiabilidad de hasta un 97% en detección de incendios activos. Sin embargo, se encontró la limitación de la ausencia en detección de focos de calor de superficies pequeñas o de corta duración. Así mismo, indican que el sistema de detección de focos de calor es efectivo para áreas superiores a las 100 hectáreas (IGP, 2023).

Por otro lado, después de realizar la entrevista con el Ing. Mario Ramos Huamani, evaluador del Centro de Operaciones de Emergencia Regional Cusco (COER Cusco) y su equipo de trabajo, se identificó que actualmente la detección de incendios forestales se basa en el

monitoreo satelital mediante la plataforma UMS del SERFOR. Sin embargo, se indica que existen ciertas limitaciones en dicho sistema, entre ellas el tiempo de visualización de eventos, ya que en varios casos se generan reportes pasadas algunas horas, cuando el incendio se ha propagado, incluso cuando ya fue atendido. Al finalizar, se concluyó que no existe en las instituciones del COER e INDECI un sistema para detectar incendios forestales que cumpla con las especificaciones propuestas en el proyecto de ingeniería.

En resumen, en todo el mundo y en especial en el sector de Picol Orccompucyo, se tiene la problemática de los incendios forestales. También se tiene información sobre la existencia de algunas soluciones tecnológicas para detectar y mitigar incendios forestales, las cuales tienen un buen desempeño, pero con costos de implementación elevados. En Cusco, las instituciones dedicadas a la protección de entornos forestales y atención de emergencias regionales utilizan tecnología satelital con fines de detección de incendios forestales donde se evidencia algunas limitaciones que suelen estar relacionadas al tipo de tecnología utilizado, como la detección de incendios forestales cuando estos ya están en un nivel avanzado y cuando se han perdido grandes extensiones de cobertura forestal.

Como la provincia de Cusco es una de las más afectadas, es necesario implementar nuevas tecnologías que trabajen en sinergia con las soluciones ya establecidas, como la Unidad de Monitoreo Satelital, para lograr un control más rápido y efectivo de los incendios forestales.

1.2.2. Formulación del Problema

La detección de incendios forestales en la región de Cusco se basa en la localización de focos de calor mediante tecnología satelital, eficiente para detectar grandes extensiones, pero tardía en la identificación inicial de incendios, lo que retrasa la respuesta oportuna y aumenta el impacto ambiental.

1.3. Justificación

1.3.1. Justificación Social

El proyecto de ingeniería tiene el potencial para ayudar al cuerpo de bomberos forestales y a otras instituciones dedicadas al control de emergencias en desastres naturales, tal es el caso del COER, INDECI, SERFOR, entre otras. Esto se lograría facilitando las acciones de prevención y atención oportuna de incendios forestales.

1.3.2. Justificación Ambiental

El proyecto de ingeniería puede otorgar una mayor protección a todas las áreas forestales que alberga el sector de Picol Orccompucyo, de esta forma aportaría a la sostenibilidad ambiental lo cual, es uno de los objetivos mundiales vigentes.

1.3.3. Justificación Tecnológica

Un beneficio clave sería la utilización de tecnologías modernas aplicadas al entorno local y a los principales problemas identificados en la región, los incendios forestales son un ejemplo de ello.

1.3.4. Justificación Económica

Si bien existen soluciones comerciales en sistemas de detección de incendios forestales utilizando procesamiento de imágenes y visión artificial, estos tienen costos altos de implementación y su lógica de funcionamiento son patentadas y reservadas; Es por ello, que se plantea realizar el prototipo del sistema con dispositivos comerciales, de bajo costo y software de código abierto.

La implementación futura del proyecto de ingeniería puede disminuir gastos significativos, como: trabajos de neutralización de incendios forestales, reforestación y reparación de daños ambientales.

1.3.5. Justificación Metodológica

Se propone adaptar metodologías en detección de incendios forestales planteadas en investigaciones, teniendo en cuenta su desempeño, grado de efectividad y las conclusiones finales encontradas. Cabe resaltar, que ahora estas serán aplicadas a diferentes condiciones ambientales y geográficas, como la región del Cusco. De esta forma se evaluará su funcionamiento en un contexto diferente.

1.4.Objetivos

1.4.1. Objetivo General

Diseñar e Implementar un prototipo del sistema de detección de incendios forestales mediante procesamiento digital de imágenes obtenidas por una cámara estacionaria fija, con la finalidad de establecer la viabilidad para identificar incendios forestales en su etapa inicial, contribuyendo potencialmente a la disminución de áreas afectadas en el sector de Picol Orccompucyo del distrito de San Jerónimo-Cusco.

1.4.2. Objetivos Específicos

- Diseñar el sistema de detección de incendios forestales mediante el uso de una cámara estacionaria y procesamiento digital de imágenes.
- Implementar el prototipo de detección de incendios forestales que permita el monitoreo en tiempo real, ubicando una cámara estacionaria en una zona estratégica para asegurar cobertura amplia.
- Desarrollar el algoritmo de detección de incendios forestales utilizando el lenguaje de programación más óptimo en procesamiento de imágenes.
- Evaluar la eficiencia del prototipo del sistema mediante pruebas de campo, simulando eventos controlados.

1.5. Alcances

Dentro de los alcances del proyecto:

- Se desarrollará un prototipo del sistema de detección de incendios forestales utilizando una cámara estacionaria, provisto con tecnología de procesamiento de imágenes.
- El prototipo del sistema proporcionará la posibilidad de un monitoreo continuo y la capacidad de escalar en diferentes entornos.
- El prototipo del sistema proporcionará una comunicación rápida mediante envío de notificaciones de incendios forestales.

1.6. Limitaciones

El prototipo de detección de incendios forestales no abarcará todo el espacio geográfico de Picol Orccompucyo debido a la cobertura limitada de la cámara y los recursos disponibles. Se seleccionará un área específica para pruebas, que permita una evaluación controlada y facilite ajustes y mejoras.

El proyecto está limitado por un periodo de pruebas tentativo de dos meses, posibles errores de detección debido a condiciones ambientales, y la potencial negativa de permisos o apoyo de instituciones competentes.

1.7. Metodología

1.7.1. Enfoque y Alcance del Proyecto de Ingeniería

El proyecto de ingeniería es de carácter aplicado, orientado al desarrollo tecnológico. Así mismo, posee un enfoque basado en diseño de Ingeniería, porque realiza tareas de integración de conocimientos teóricos y técnicos para el desarrollo de un prototipo que introduce una mejora en las técnicas de atención de incendios forestales.

1.7.2. Identificación de Parámetros de Diseño

1.7.2.1. Parámetros de entrada

- **Secuencia de Fotogramas:** Flujo sucesivo de imágenes FPS (cuadros por segundo).
- **Resolución de Imagen:** Característica de una imagen que representa la cantidad de pixeles que posee (ancho x alto), mientras más pixeles tenga una imagen mejor será la calidad, estos detalles ayudarán a realizar un mejor procesamiento de la imagen.
- **Información Cromática:** Valor numérico de cada pixel, el cual representa el color del entorno forestal.

1.7.2.2. Parámetros de salida

- **Métricas de clasificación**

  Números de Falsos Positivos: Se produce cuando el sistema si detecta una alerta y en paralelo no se produce un incendio forestal.

  Números de Falsos Negativos: Se produce cuando el sistema no detecta una alerta y en paralelo si está ocurriendo un incendio forestal.

  Números de Verdaderos Positivos: Se produce cuando el sistema si detecta una alerta y en paralelo si está ocurriendo un incendio forestal.

  Números de Verdaderos Negativos: Se produce cuando el sistema no detecta una alerta y en paralelo no está ocurriendo un incendio forestal.

- **Tiempo de respuesta:** Tiempo trascurrido entre la detección y notificación.

Capítulo II: Marco Teórico

2.1. Estado del Arte

2.1.1. Antecedentes Internacionales

Un primer trabajo corresponde a la investigación **"A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing"** [Una revisión sobre los sistemas de detección temprana de incendios forestales mediante teledetección óptica] de Panagiotis Barmpoutis et al. (2020).

El estudio proporciona una revisión exhaustiva de las tecnologías de detección temprana de incendios forestales utilizando métodos de teledetección óptica. Compara sistemas basados en estaciones terrenas, vehículos aéreos no tripulados (UAV) y estaciones satelitales. El documento analiza algoritmos y técnicas utilizadas para la detección óptica de fuego y humo en distintos proyectos y trabajos de investigación realizados por diferentes autores ya sea utilizando métodos tradicionales como procesamiento de imágenes hasta los más complejos como inteligencia artificial.

La investigación proporciona información valiosa para decidir qué tipo de tecnología a utilizar ya sea por temas de costos, precisión, área de cobertura y tiempo de respuesta. Donde las principales conclusiones son:

- **Sistemas terrestres**: Tienden a ser más eficientes en términos de precisión, tiempo de respuesta, volumen de trabajo, capacidad para detectar pequeños eventos de fuego y una alta resolución espacio que nos brinda gracias a las cámaras utilizadas, sin embargo, una de sus deficiencias es que tiene una cobertura limitada y dedicada.
- **Vehículos aéreos no tripulados (UAV)**: Su mayor fortaleza el área de cobertura incluso en lugares inaccesibles, un potencial bueno para futuras aplicaciones,

capacidad para detectar pequeños eventos de fuego y la posibilidad de utilizar cámaras infrarrojas. Las desventajas son el tiempo de vuelo limitado, sensibilidad a condiciones climáticas y la necesidad de un operador constante.

- **Sistemas satelitales:** Tienen como ventaja el área de cobertura a detectar y un buen nivel de precisión, No obstante, su principal desventaja es la latencia y su escasa capacidad para transmitir grandes cantidades de datos.

Un segundo trabajo es el estudio **"Intelligent and visión-based fire detection systems: A survey"** [Estudio: Sistemas de detección de incendios basados en inteligencia y visión] de Fengju Bu, & Mohammad Samadi (2019).

El documento es una revisión de sistemas inteligentes de detección ante incendios basados en visión, categorizados en dos grupos: sistemas de detección inteligente para incendios forestales y sistemas de detección de incendios para todo tipo de entorno. Los sistemas utilizan diversas técnicas, como el procesamiento de imágenes en espacios de color, reglas y variación temporal, redes neuronales convolucionales (CNN) y lógica difusa. El documento evalúa el rendimiento de estos sistemas en términos de tasa de detección, precisión, tasa de verdaderos positivos y tasa de falsos positivos.

Según las conclusiones, los sistemas de detección basados en redes neuronales convolucionales profundas obtienen el mejor rendimiento en la mayoría de los resultados de evaluación, obteniendo una tasa de precisión y detección de 90% y una tasa de falsos positivos menores al 10%. Por otro lado, los sistemas basados en procesamiento de imágenes tienen una tasa de detección del 95%, y una tasa de falsas alarmas inferior al 30%, demostrando ser aun una buena tecnología para detección de incendios forestales.

Un tercer trabajo es el artículo de investigación **"Forest Fire Detection Using a Rule-Based Image Processing Algorithm and Temporal Variation"** de Mubarak & Honge (2018).

Los autores presentan un algoritmo para la detección de incendios forestales que consta de cuatro etapas principales:

- **Sustracción de fondo**: Aplicadas a regiones donde se detecta movimiento, esto debido a que los incendios forestales fluctúan continuamente.
- **Conversión de espacio de color**: Utilizan la conversión RGB al YCbCr para escoger regiones candidatas.
- **Aplicación de reglas**: Se utiliza cinco reglas para la detección de pixeles característicos del fuego.
- **Variación temporal**: Detección de pixeles cambiantes mediante la diferencia de fotogramas sucesivos.

Los resultados finales muestran la eficiencia de dicho algoritmo llegando a tener hasta un 92.59% de precisión. El autor propone al final combinar las reglas en diferentes espacios de color.

Un cuarto trabajo es el proyecto final de carrera **"Sistema de detección de incendios forestales utilizando técnicas de procesado de imagen."** del autor David Martin Borregon Domènech.

El propósito de esta investigación fue establecer un sistema de detección de incendio forestal mediante técnicas de procesamiento de imágenes; se proponen dos técnicas: uno que se basa en la detección de fuego y otro que se basa en la detección de humo.

- Para detección de fuego, se utiliza el modelo de color HSV (tono, saturación y brillo), obteniendo un 71% de clasificación de elementos etiquetados como fuego y 1,7% de clasificación errónea.

- Para detección de humo, el algoritmo aplica sustracción de fondo, filtración de elementos rápidos, detección por color, análisis de componentes conexas y por último aprendizaje de humo.

Aunque en el estudio no da un valor exacto de la precisión en detección de humo, se establece que en los videos de prueba los resultados son variables desde el 20% al 80% en detección de pixeles que representan humo. El autor atribuye a la opacidad del humo y velocidad de movimiento del humo.

2.1.2. Antecedentes Nacionales y Locales

En el transcurso de la investigación realizada para esta tesis, se realizó una revisión exhaustiva de trabajos académicos locales y nacionales relevantes para el tema del proyecto. Sin embargo, luego de una cuidadosa búsqueda en diversos repositorios y fuentes de información, no se encontraron antecedentes específicos locales o nacionales que aborden directamente el tema de diseño y validación de un sistema detector de incendios forestales con estaciones terrenas. Esto resalta la relevancia y originalidad de este trabajo, ya que ayudará a llenar los vacíos en el conocimiento existente y brindará nuevas perspectivas sobre las áreas involucradas.

2.2. Bases Teóricas

2.2.1. Incendios Forestales

Se entiende por incendio forestal a aquel fuego sin control que consume extensas áreas de bosques y zonas forestales utilizadas para producción o protección de recursos naturales y patrimoniales. (Moscovich, Ivandic, & Besold, 2014).

2.2.1.1. Causas para la Formación de Incendios Forestales

Los incendios forestales pueden surgir debido a diversas causas, tanto naturales como antropogénicas. Las causas naturales incluyen la ocurrencia de rayos durante periodos de sequía,

mientras que las causas antropogénicas se deben a actividades humanas, ya sea intencionales o accidentales.

2.2.1.2. Factores que Contribuyen a la Formación de Incendios Forestales

Los incendios forestales son influenciados por diferentes factores, entre ellos, los factores meteorológicos, topográficos y de combustible.

- **Factores meteorológicos**: hacen referencia a las condiciones climáticas, como la intensidad y dirección del viento, que pueden favorecer la propagación del fuego.
- **Factores topográficos**: están relacionados con la configuración del terreno donde ocurren los incendios. Por ejemplo, en zonas montañosas o con pendientes pronunciadas, los incendios tienden a propagarse con mayor rapidez debido al ascenso del calor y la dirección del viento.
- **Combustible**: se refiere a los materiales inflamables presentes en las áreas forestales, como pastos secos o árboles con resinas altamente inflamables. Estos combustibles pueden alimentar y acelerar la propagación del fuego.

2.2.1.3. Impactos Ambientales, Económicos y Sociales de Incendios Forestales

Los incendios forestales tienen múltiples impactos negativos en el medio ambiente, la economía y la sociedad. Desde el punto de vista ambiental, genera la pérdida de hábitats naturales, la destrucción de la biodiversidad y contaminación atmosférica por gases de efecto invernadero a gran escala. También provoca la degradación del suelo y la polución del agua. En términos económicos, causan pérdidas considerables en la industria forestal, la agricultura y el turismo, además de aumentar los costos asociados a la extinción del fuego y recuperación de los ecosistemas dañados. A nivel social, pueden desplazar comunidades, dañar infraestructuras y causar la pérdida de vidas humanas (Manta, 2017).

2.2.2. Sistemas de Alerta Temprana

2.2.2.1. Definición

Son herramientas y tecnologías cuyo objetivo principal es detectar y prevenir algún evento peligroso para comunicar de manera oportuna la aparición de estas situaciones inusuales, permitiendo así la toma de acciones.

2.2.2.2. Características

Según Glantz, M.H. & Ausbel, J.H. (1988) un sistema de alerta temprana esta caracterizado por las siguientes etapas:

- **Detección temprana**: captar indicios de amenaza
- **Monitoreo constante**: vigilancia continua de variables asociadas con la amenaza
- **Evaluación y análisis de datos**: adquisición y procesamiento de datos para evaluar gravedad y evolución de amenaza.
- **Comunicación rápida y efectiva**: mecanismo para transmitir la alerta a las personas u organizaciones pertinentes.
- **Respuesta y planificación**: plan de mitigación y coordinación entre organizaciones involucradas.

2.2.2.3. Tipos de Tecnologías Utilizadas en Sistemas de Alerta Temprana

Las tecnologías que se emplean en los sistemas de alerta temprana son:

- **Sistema de alerta temprana basados en sensores inalámbricos terrenos**: este tipo de tecnología mide parámetros ambiéntales tales como temperatura, humedad, presión atmosférica, nivel de agua, entre otras, de esta forma detectar eventos infrecuentes como inundaciones, huaycos, incendios urbanos-forestales, terremotos, oleajes anómalos, etc.

- **Sistemas de alerta temprana basados en visión**: este tipo de tecnología utiliza cámaras, los cuales pueden estar ubicadas en lugares fijos, drones o satélites. Estos combinados con técnicas de procesamiento digital de imágenes o visión artificial ayudan a la detección de eventos inusuales.
- **Sistemas de alerta temprana satelitales:** este tipo de tecnología utiliza la información recopilada de satélites con la finalidad de realizar la detección de eventos meteorológicos anormales como huracanes e incendios forestales a gran escala.
- **Sistema de alerta temprana utilizando modelos predictivos**: este tipo de tecnología utiliza básicamente bases de datos históricos de variables meteorológicas, combinado con aplicaciones de machine learning pueden predecir ocurrencias de eventos meteorológicos inusuales.

2.2.2.4. Sistemas de Detección Temprana Basados en Visión Comerciales

Hoy en día existen empresas dedicadas a la venta de soluciones de videovigilancia y detección temprana de incendios forestales, a continuación, se describirán algunos de ellos:

- **InsightFD3 Insight Robotics**

  Sistema de detección temprana de incendios forestales mediante uso de cámaras equipadas con sensores visuales e infrarrojos. El robot puede girar 360 grados y monitorear constantemente las 24 horas del día. El sistema consta de tres subsistemas principales: el primero es el robot InsightFD encargado de la captura de imágenes, el segundo es el subsistema AI-Engine el cual es un dispositivo de computación encargado de la detección automática de incendios forestales y el último es el Insight Glorecortebe el cual es la plataforma de gestión que genera la alerta en mapas 2D y 3D y proporcionar visualización de datos por terceros. (Inteccon, 2024).

Figura 3

InsightFD3. Insight Robotics

Fuente: Inteccon, 2024.

- **Sistema de videovigilancia de incendios ADELIE**

  Es un sistema orientado a monitorear, detectar, localizar y dar seguimiento a incendios forestales. Equipada con cámaras multiespectrales y tecnología de visión artificial. El sistema monitorea los espacios naturales las 24 horas del día y realiza un análisis de las áreas cada 3 minutos, activando una alerta cuando se localice un incendio forestal. Además, permite un control remoto para la visualización del área afectada y centraliza toda la información y las comparte a todos los centros de comandos para su respectiva atención. (Paratronic, 2025).

Figura 4

Sistema de videovigilancia de incendios ADELIE

Fuente: Paratronic, 2025.

2.3.Definición de Términos Básicos

2.3.1. Imagen Digital

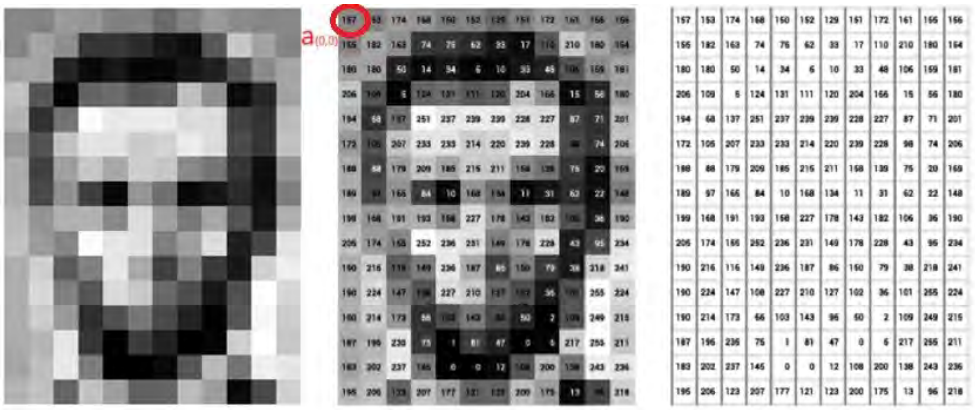
Las imágenes digitales son representaciones discretas de datos que contienen información de intensidad y están compuestas por píxeles. Estas imágenes son el resultado de capturar y convertir imágenes analógicas en formato digital utilizando dispositivos como cámaras digitales, escáneres y otros medios de adquisición de imágenes. (Muñoz Trujillo, 2023)

Una imagen puede ser representada mediante una matriz M x N:

$$I_{(x,y)} = \begin{bmatrix} a_{0,0} & \cdots & a_{0,N} \\ \vdots & \ddots & \vdots \\ a_{M,0} & \cdots & a_{M,N} \end{bmatrix} \qquad (1)$$

Figura 5

Representación de una imagen digital



Fuente: Melvin Wever, 2019.

2.3.1.1. Pixel

Unidad mínima de una imagen digital ubicada en un espacio bidimensional. Este contiene información de color e intensidad.

Tomando en cuenta la ecuación (1) un pixel representa a una de las coordenadas de la matriz por ejemplo $a_{0,0}$.

2.3.1.2. Muestreo y Cuantificación

El muestreo implica dividir la imagen analógica en segmentos reducidos, en esta situación, la imagen analógica se transforma en una matriz discreta de M x N pixeles.

Por otro lado, la cuantificación implica otorgar un grado de gris a cada muestra adquirida durante la fase de muestreo. Por ejemplo, en la figura 5 se puede apreciar que el primer pixel a $_{(0,0)}$ tiene el valor de 157 de un total de 256 posibles valores, donde 0 representa el color negro y 255 representa el color blanco.

2.3.1.3. Resolución

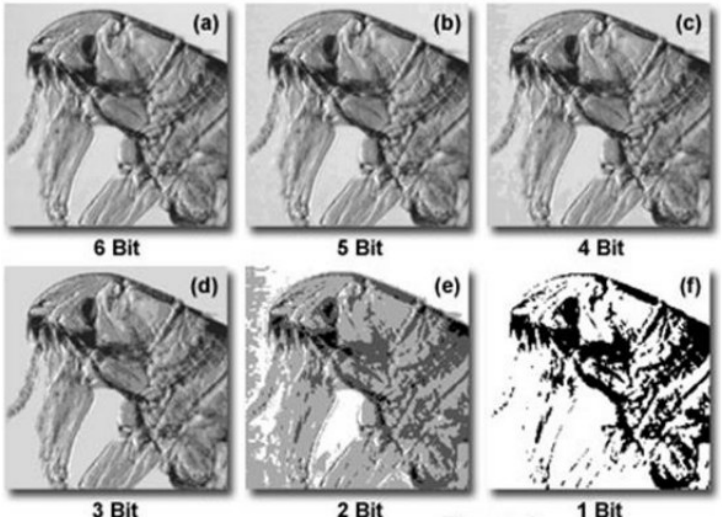
Una imagen se almacena en bits donde 1 bit es la mínima unidad de almacenamiento de una imagen.

$$Imagen\ de\ 1\ bit = 2^1 = 2\ colores$$

$$Imagen\ de\ 2\ bit = 2^2 = 4\ colores$$

Figura 6

Representación de resolución de una imagen



Fuente: Universidad de Sevilla.s.f.

Por lo tanto, podemos describir la resolución como el nivel de detalle perceptible en una imagen, fundamentado en dos factores:

- La resolución de intensidad: alude a la cantidad de niveles de grises que puede presentar la imagen, que depende del número de bits.
- La resolución espacial: hace referencia al muestreo matriz M x N o más conocido como la cantidad de pixeles existentes en la imagen.

2.3.1.4. Modelos de Color

Los modelos de color en imágenes digitales son sistemas que describen como se perciben y representan los colores. Son indispensables en diversas etapas del procesamiento de imágenes, ya que permiten capturar, procesar, visualizar y transmitir la información cromática de manera precisa. Algunos modelos de color mayormente utilizados en aplicaciones de procesamiento de imágenes son los siguientes:

A. **Modelo de color RGB:** El modelo de color RGB se fundamenta en la mezcla de colores primarios: rojo, verde y azul. Este modelo se basa en el principio del color aditivo, en el cual se mezclan proporciones de cada color para obtener una amplia gama de tonalidades y colores. Siendo el color blanco la combinación de los tres colores en su máxima intensidad y negro en la usencia de luz de estos colores. (Herrera, 2016)

   Matemáticamente un pixel de una imagen RGB se puede expresar:

$$I(x, y) = \big(R(x, y), G(x, y), B(x, y)\big)$$

   Donde:

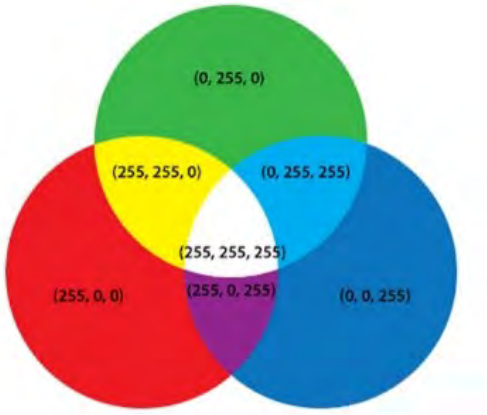
   R (x, y): valor canal rojo del pixel en las coordenadas (x, y) con valores (0 - 255)

   G (x, y): valor canal verde del pixel en las coordenadas (x, y) con valores (0 - 255)

   B (x, y): valor canal azul del pixel en las coordenadas (x, y) con valores (0 - 255)

Figura 7

Modelo de color RGB



Fuente: Abellan, 2019.

B. **Modelo de color HSV:** Modelo utilizado mayormente en aplicaciones de procesamiento de imágenes, según Gonzalez (2009) este modelo viene representado por las componentes:

- **Matiz (Hue)**: representado por una paleta hexagonal cromático en donde 0º representa el color rojo, 120º representa al color verde y 240 º al color azul.
- **Saturación:** representa a la intensidad de color y se mide en una escala de 0% a 100%. En este caso la saturación varia con los cambios en la luminosidad.
- **Valor (Value)**: representa a la luminosidad y está en una escala de 0% a 100%, en donde 0 representa al negro sin brillo y 100% representa al color blanco o máximo brillo.

Matemáticamente la conversión de color RGB a HSV viene representado por las siguientes ecuaciones:

$R' = \frac{R}{255}\ ;\ G' = \frac{G}{255};\ B' = \frac{B}{255}$ normalización R, G, B

$V = \max(R', G', B')$ componente Value

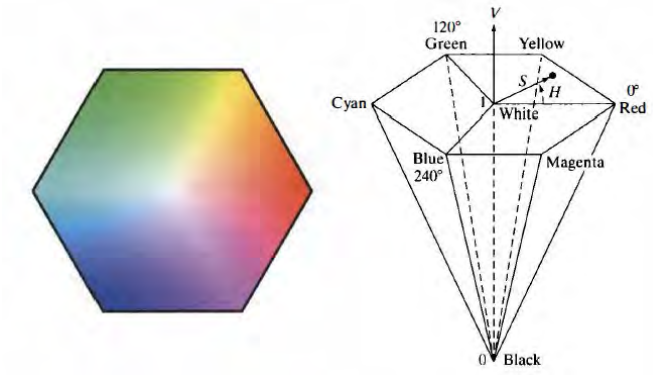
$\Delta = \ \mathrm{V} - \min(R', G', B')$ cálculo de máximo y mínimo

$$S = \begin{cases} 0 \ si\ V = 0 \\ \frac{\Delta}{V} \ si\ V \neq 0 \end{cases}$$ componente Saturación

$$H = \begin{cases} 0 & si\ \Delta = 0 \\ 60\ x\ \left(\frac{G'-B'}{\Delta}\ mod 6\right) & si\ V = R' \\ 60\ x\ \left(\frac{B'-R'}{\Delta} + 2\right) & si\ V = G' \\ 60\ x\ \left(\frac{R'-G'}{\Delta} + 4\right) & si\ V = B' \end{cases}$$ componente Matiz (HUE)

Figura 8

Modelo de color HSV



Fuente: Gonzalez R. W., 2009.

C. **Modelo de color YCbCr:** Es una familia del espacio de colores utilizado en los sistemas de video y compresión de imágenes.

- **Luminancia (Y):** representa a la información de brillo de una imagen en una escala de grises.
- **Crominancia Azul diferenciada (Cb):** captura la información de diferencia entre la componente azul y un valor de referencia.
- **Crominancia Roja diferenciada (Cr):** captura la información de diferencia entre la componente roja y un valor de referencia. (Gonzalez R. W., 2009)

Matemáticamente la conversión del modelo RGB al modelo YCbCr es estandarizada por ITU-R BT.601 mediante las fórmulas:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

$$Cb = 128 + (-0.168736 \times R - 0.331264 \times G + 0{,}5 \times B)$$

$$Cr = 128 + (0.5 \times R - 0.418688 \times G - 0.081312 \times B)$$

2.3.1.5. Formatos y Tipos de Imágenes Digitales

El formato de las imágenes digitales se refiere a la manera en que se almacenan y al tipo de información que representan. Según IONOS Digital Guide (2020) hay diferentes formatos utilizados según las aplicaciones requeridas. Algunos de los formatos más comunes son:

- **Formato JPEG (Joint Photographic Experts Group):** se utiliza para imágenes con tonos de colores continuos. El tamaño de datos de imagen se reduce mediante un algoritmo de compresión, eliminando información visualmente menos importante.
- **Formato PNG (Portable Network Graphics):** formato de imagen sin pérdida de información durante la compresión.

2.3.2. Procesamiento de Imágenes

2.3.2.1. Definición de Procesamiento Digital de Imágenes

Según Gonzalez et al. (2009) procesamiento digital de imágenes es la manipulación de imágenes digitales mediante el uso de algoritmos y técnicas computacionales. Cuyo objetivo principal es mejorar la calidad de las imágenes, extraer información, análisis cuantitativos y tareas de reconocimiento y clasificación.

2.3.2.2. Etapas de Procesamiento Digital de Imágenes

Gonzalez et al. (2009) indica el procesamiento digital de imágenes está conformado por 5 etapas los cuales se mencionarán a continuación:

- **Adquisición de imágenes:** etapa en la que se captura una imagen utilizando cámaras digitales, escáneres o sensores remotos. Convirtiendo la información óptica en un formato digital que se puede procesar y analizar.
- **Preprocesamiento:** Conjunto de técnicas para mejorar la calidad de las imágenes, como la corrección de contraste, ajuste de brillo y corrección de ruido de ruido (filtros).

Figura 9

Proceso de corrección de ruido en una imagen

Nota. a) imagen con ruido b) imagen suavizada o filtrada con máscara de promedio c) imagen suavizada con método OTSU. Fuente: Gonzalez R. , 2008.

- **Mejora de imágenes**: Conjunto de técnicas y algoritmos con el objeto de realzar las características específicas de una imagen. Como mejora de nitidez, corrección del balance de color y la supresión de imperfecciones.
- **Segmentación**: Etapa donde se muestrea la imagen en partes más pequeñas y significativas. Se realiza mediante la identificación de bordes, diferenciación de colores o texturas, uso de técnicas de agrupamiento y umbralización (Glantz, 1988).

Figura 10

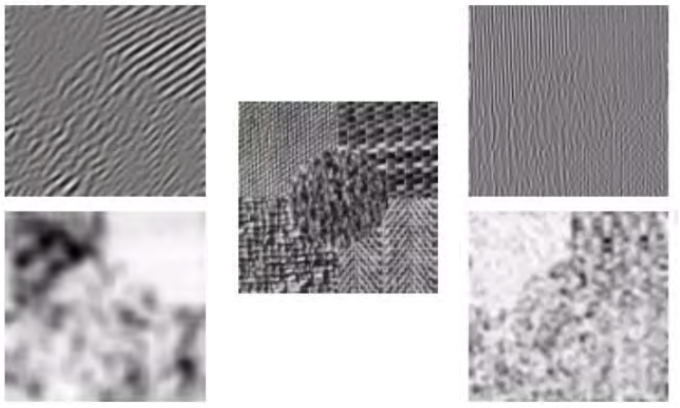
Proceso de segmentación de una imagen

Nota. a) Imagen original b) Imagen segmentada a partir de discontinuidades de intensidad c) Resultado de segmentación de imagen en dos regiones. Tomado de Gonzalez R. , 2008.

- **Extracción de características**: Implica la detección y cuantificación de atributos particulares en una imagen, como los límites, las texturas, las formas y los colores. Estas características son luego empleadas en actividades como la clasificación, el reconocimiento y el análisis de patrones.

Figura 11

Descriptor de texturas



Fuente: Multimedia and Vision Laboratory at Universidad del Valle, 2013.

2.3.2.3.Transformaciones Geométricas y Morfológicas en Procesamiento de Imágenes

Las transformaciones geométricas y morfológicas se utilizan para modificar la forma, la posición y la estructura de los objetos en las imágenes digitales.

2.3.2.3.1. Transformaciones Geométricas

Según Gonzalez et. al (2009) las transformaciones geométricas más utilizadas son:

A. **Rotación**: gira una imagen alrededor de un punto de referencia, cambiando así la orientación de la imagen.

B. **Escalado:** Modifica el tamaño de la imagen, aumentando o reduciendo el tamaño de la imagen.

C. **Traslación**: Esta transformación desplaza los pixeles de una imagen en una dirección determinada.

D. **Reflexión:** voltea la imagen en relación a un eje dado.

E. **Shearing**: distorsiona la forma de una imagen en cierta dirección.

2.3.2.3.2. Operaciones Morfológicas

Estas transformaciones se basan en operaciones de teoría de conjuntos mediante las cuales se pueden cambiar la forma y estructura de los elementos de la imagen. las conversiones más utilizadas son:

A. **Erosión**: Reduce el tamaño de los objetos de una imagen eliminando pixeles en bordes, pixeles aislados y solitarios y estructuras finas. Estos cambios se producen debido a la interacción con un elemento estructurante (también conocido como kernel). Este proceso se realiza mediante un ventaneo (windowing) a través de toda la imagen binaria de entrada. En donde el pixel de salida será primer plano o blanco si todo el

elemento estructurante cabe en una región donde todos los pixeles son de primer plano y en cualquier otro caso será fondo o pixel negro.

Matemáticamente la erosión se define como:

$A \ominus B = \{z \mid (B_z) \subseteq A\}$

$Donde$:

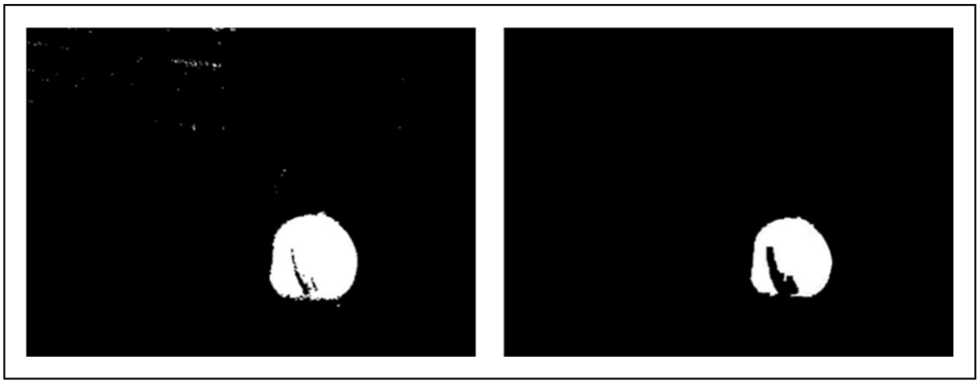
A: imagen binaria de entrada

B: elemento estructurante

B_z: traslación del elemento estructurante por el vector z

Figura 12

Operación morfológica - erosión en una imagen



Nota: proceso de aplicación de operación morfológica de erosión a una máscara binaria.

B. **Dilatación**: Amplía el tamaño de los objetos de interés en la imagen al agregar pixeles en los bordes, de esta forma ayuda a ampliar los objetos o cerrar los huecos no deseados en la imagen. Estos cambios se producen debido a la interacción con un elemento estructurante (también conocido como kernel). Este proceso se realiza mediante un ventaneo (windowing) a través de toda la imagen binaria de entrada. En donde el pixel de salida será primer plano o blanco si el elemento estructurante al ser superpuesto coincide con al menos un pixel de primer plano de la imagen binaria de entrada.

Matemáticamente la erosión se define como:

$A \oplus B = \{z \mid (B_z) \cup A \neq \emptyset\}$

$Donde$:

A: imagen binaria de entrada

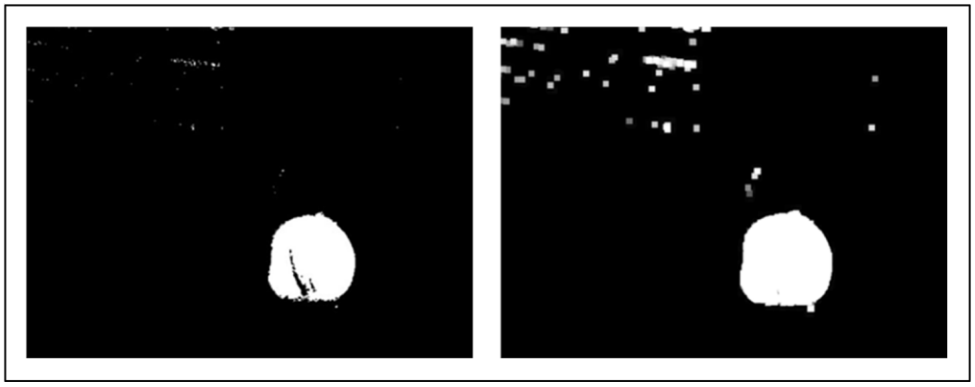
B: elemento estructurante

B_z: traslación del elemento estructurante por el vector z

$\emptyset$: $vacio$

Figura 13

Operación morfológica - dilatación en una imagen



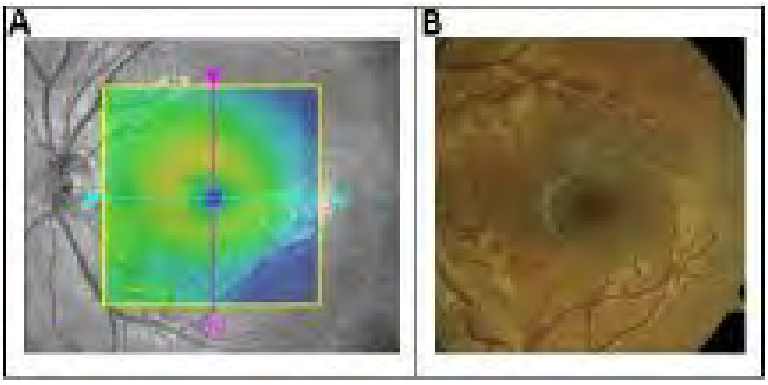
Nota: proceso de aplicación de operación morfológica de dilatación a una máscara binaria.

2.3.2.4. ROI (Region of Interesting)

En el ámbito del procesamiento de imágenes, la región de interés (ROI) hace referencia a la porción de una imagen escogidas para su análisis exhaustivo. Una ROI puede ser expresada como una máscara binaria. En la imagen de máscara, los píxeles que están dentro de la ROI se fijan en 1 y los que están fuera, en 0. (MathWorks, s.f.)

Figura 14

ROI (Region Of Interesting)



Nota. A) selección de ROI en Imagen original. B) ROI (Region of Interesting). Fuente: Díaz, 2015

2.3.3. Herramientas Computacionales

2.3.3.1. Python

En 1989, Guido Van Rossum desarrolló Python en el Cemtrum Wiskunde & Informática (CWI). Python es un lenguaje de programación de alto nivel, de código abierto, con tipos dinámicos, orientado a objetos e interpretado. Empleado extensamente en el desarrollo de software, programas web, ciencias de datos y aprendizaje automático. Python puede funcionar en varias plataformas y sistemas operativos (OS), entre los que se destacan Windows, Linux, macOS y Unix. De igual manera, Python cuenta con más de 137 000 bibliotecas que los desarrolladores utilizan para usarlas en diversas aplicaciones. Convirtiendo Python en un lenguaje eficaz y sencillo de aprender y entender. (Amazon Web Services, 2024)

2.3.3.2. Bibliotecas de Python

Una biblioteca o librería de Python es un conjunto de módulos que incluyen funciones, clases y procedimientos pre establecidos, creados para ampliar las capacidades del lenguaje de programación Python. Entre ellas tenemos:

- **Matplotlib**: Aplicado en usos científicos para realizar gráficos de data en dos y tres dimensiones.
- **Pandas**: Utilizado principalmente en labores de ciencia de datos, es eficiente para manejar datos de serie temporal y datos organizados como tablas y matrices.
- **NumPy**: Utilizado para efectuar operaciones y algebra lineal
- **Requests**: Utilizada en desarrollo web, ya que utiliza solicitudes HTTP,
- **OpenCV-Python**: Utilizado para desarrolladores de aplicaciones de procesamiento de imágenes y visión artificial.
- **OS**: Librería que permite el manejo de archivos, directorios y procesos del sistema operativo.
- **Datetime:** Librería que permite el uso de la fecha y hora del sistema operativo.
- **Picamera2:** Librería que permite uso y configuración de cámaras oficiales de Raspberry.

2.3.3.3. OpenCV

OpenCV (Open Source Computer Vision) es una librería de código abierto que ofrece un amplio conjunto de herramientas y algoritmos para el procesamiento digital de imágenes y el análisis de video. Intel impulsó su desarrollo inicial en el año 2000, con la finalidad de simplificar la implementación de aplicaciones de visión computacional en tiempo real. OpenCV se emplea en programas como:

- **Procesamiento de imágenes**: Se llevan a cabo acciones en las imágenes como filtros, transformaciones geométricas y segmentación, que resultan cruciales para extraer información pertinente de las mismas.

- **Visión por computadora**: Es capaz de llevar a cabo funciones complejas como el reconocimiento facial, la identificación de objetos y el monitoreo de movimiento.
- **Interacción hombre maquina**: Facilita la creación de interfaces en las que el usuario interactúa con sistemas informáticos.
- **Interacción con técnicas avanzadas**: Emplea métodos de aprendizaje automático y Deep learning para dar solución a problemas complejos de visión artificial.

2.3.3.4. Algoritmos de Análisis de Movimiento en OpenCV

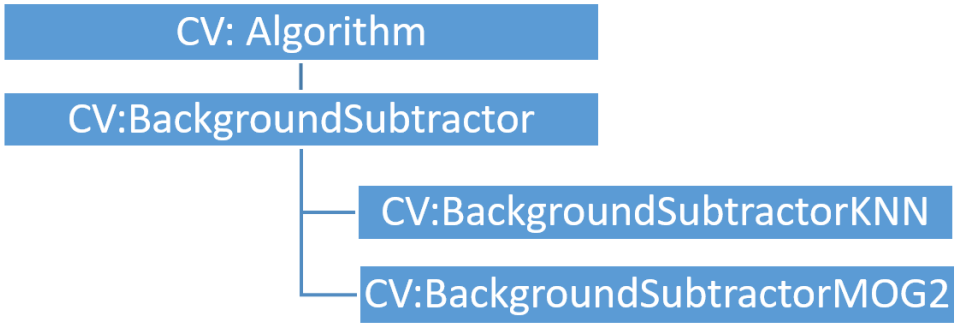
OpenCV dispone de algoritmos para realizar análisis de video en tiempo real, esto con la finalidad de realizar detección de eventos de interés tales como presencia y movimiento. Los algoritmos para realizar análisis de movimiento son: sustracción de fondo, flujo óptico.

2.3.3.4.1. Técnicas de Sustracción de Fondo

La sustracción de fondo es una técnica ampliamente utilizada en procesamiento de video con la finalidad de identificar elementos en movimiento. La sustracción de fondo divide los objetos en movimiento del fondo estático en un primer plano binario (los elementos que se mueven son blancos y el fondo es negro). OpenCV dispone de algoritmos para sustracción de fondo entre las principales esta MOG2 y KNN.

Figura 15

Diagrama de herencia para CV: BackgroundSubtractor



Nota. Diagrama de herencia de algoritmos BackgroundSubtractor de OpenCV. Fuente: OpenCV, 2025.

Los algoritmos MOG2 y KNN están basados en el trabajo de investigación de Zivkovic y Van der Heijden (2006). Donde el principal objetivo del trabajo de investigación fue separar los objetos en movimiento de un fondo estático. Para resolver este problema presentaron dos enfoques estadísticos avanzados, el primero basado en el modelo de mixturas Gaussianas y el segundo basado en estimación de densidad no paramétrica con kernel variable.

A. Algoritmo MOG2 (Mixture of Gaussians 2)

Este algoritmo modela el valor de cada pixel como una mezcla de varias distribuciones Gaussianas, donde cada distribución representa un posible estado o valor del pixel. Mediante el uso de criterios bayesianos, este algoritmo decide automáticamente cuál de las distribuciones gaussianas se asemeja mejor al valor del pixel y en caso de no existir coincidencia con alguna distribución, este pixel es puesto en primer plano considerado movimiento. En OpenCV el algoritmo se aplica mediante uso del comando cv2.createBackgroundSubtractorMOG2() y sus parámetros se presentan en la tabla 1.

Tabla 1

Parámetros de programación algoritmo cv2.createBackgroundSubtractorMOG2

cv2.createBackgroundSubtractorMOG2(Parámetros)	
Parámetros	**Descripción**
History	Número de frames, utilizado para actualizar el modelo de fondo.
varThreshold	El Umbral de decisión establece si un píxel pertenece a una distribución gaussiana. Para determinar la distribución apropiada, contrasta la distancia cuadrática del valor del píxel con la media de las distribuciones, ponderada según la varianza.
detectShadows	Permite detectar sombras como parte del movimiento.
Nmixtures	Número de distribuciones gaussianas usadas.
backgroundRatio	Umbral para decidir que distribución gaussiana representa el fondo.

Shadowthreshold	Umbral para considerar que un pixel es considerado sombra u objeto en movimiento.

B. Algoritmo KNN (K-Nearest Neighbors)

El algoritmo KNN es un algoritmo más simple y rápido que las distribuciones gaussianas usadas en MOG2. El algoritmo KNN utiliza estimación de densidad no paramétrica, esto quiere decir que no utiliza una forma o ecuación definida para decidir si un pixel pertenece a un primer plano o fondo. Entonces, lo que hace KNN es modelar un pixel en base a todo un conjunto de muestras. Para dividir la imagen en primer plano y fondo utiliza un kernel de tamaño variable alrededor del conjunto de muestras.

Tabla 2

Parámetros de programación algoritmo cv2.createBackgroundSubtractorKNN

cv2.createBackgroundSubtractorKNN(Parámetros)	
Parámetros	**Descripción**
History	Número de frames que utiliza el algoritmo para calcular el modelo de fondo, en este caso guarda valores anteriores del pixel en esa posición.
dis2Threshold	Umbral de distancia cuadrada que decide si el nuevo valor del pixel pertenece al fondo. Similar al valor de la varianza en una distribución gaussiana donde le da un rango para decidir si pertenece al fondo o al primer plano.
detectShadows	Permite detectar sombras como parte del movimiento.

Para un mejor entendimiento a continuación se presenta la lógica del algoritmo:

Dado el fotograma de imagen representada por la matriz 3x3 en el instante T:

$$Imagen\ RGB_T = \begin{matrix} P_{0,0} & P_{0,1} & P_{0,2} \\ [P_{1,0} & P_{1,1} & P_{1,2}]_T \\ P_{2,0} & P_{2,1} & P_{2,2} \end{matrix}$$

Analizando solo para el pixel $P_{i,j=}[R_{i,j}G_{i,j}B_{i,j}]$ el cual, por métodos de ejemplificación, el historial H para el pixel (1,1) es el conjunto de sus valores RGB en 4 fotogramas anteriores:

$$H_{1,1} = \left[P_{1,1}^{T-4}, P_{1,1}^{T-3}, P_{1,1}^{T-2}, P_{1,1}^{T-1}\right] \quad donde: \left|H_{1,1}\right| = 4$$

La heurística k vecino más cercano se define por la raíz cuadrada del tamaño del historial:

$$k = \sqrt{\left|H_{1,1}\right|} = \sqrt{4} = 2$$

Entonces para que el pixel nuevo sea considerado fondo, el número de muestras dentro del umbral de distancia cuadrada debe ser como mínimo 2.

Calculando las distancias cuadradas del valor del pixel con su historial:

$$d_i = \left(\left(R_{1,1}\right)_T - \left(R_{1,1}\right)_{T-i}\right)^2 + \left(\left(G_{1,1}\right)_T - \left(G_{1,1}\right)_{T-i}\right)^2 + \left(\left(B_{1,1}\right)_T - \left(B_{1,1}\right)_{T-i}\right)^2$$

$$d_i = \sum_{c \in \{R,G,B\}} \left(c_{1,1}^{T} - c_{1,1}^{T-i}\right)^2 \qquad para: i = 1 \ldots .4$$

Finalmente se analiza:

$$Si \sum_{i=1}^{N} \mathbb{I}(d_i \leq dis^2 Threshold) \geq K$$

Donde las distancias que se ajusten a esta condición estarán dentro del umbral y si la cantidad de distancias es igual o mayor al valor k, el pixel es considerado fondo. En caso contrario, será considerado pixel cambiante o representación de movimiento.

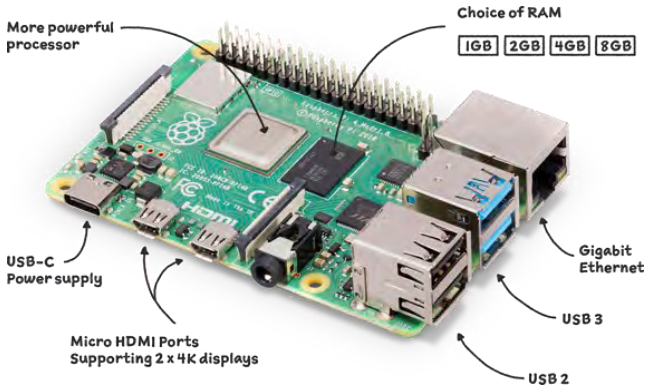
2.3.4. Sistema Embebido Raspberry Pi

Es un mini - ordenador de placa única de bajo costo y alto rendimiento, desarrollado por la Fundación Raspberry Pi y lanzado al mercado en el año 2012. Basado en arquitectura ARM, sistema operativo Linux, capacidad gráfica avanzada y diferentes capacidades de memoria RAM, que puede llegar hasta los 8 GB en los últimos modelos.

La Raspberry es un dispositivo caracterizado por su versatilidad ya que no solo es un miniordenador, sino que se puede utilizar en una amplia gama de aplicaciones como proyectos de robótica, automatización e internet de las cosas (IoT), gracias a que la Raspberry incluye puertos de entrada y salida de propósito general (GPIO). (Raspberry Pi, 2025)

Figura 16

Tarjeta electrónica Raspberry PI



Fuente: Raspberrypi, 2024.

2.3.5. Cámaras Raspberry Pi

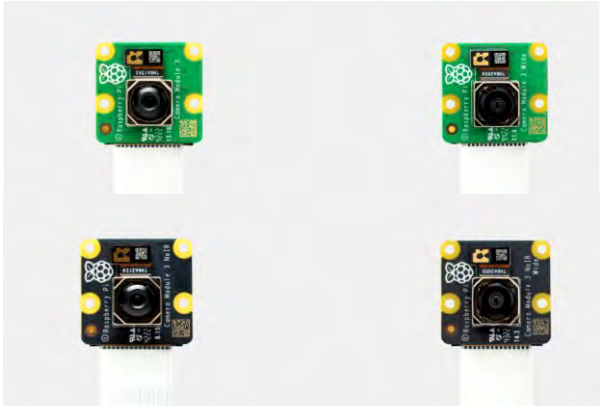
Raspberry Pi ofrece varios módulos de cámaras oficiales compatibles con los sistemas embebidos fabricados por la misma empresa. El primer modelo de 5 megapíxeles fue lanzado el año 2013, para el año 2016 lanzaron los nuevos módulos de cámaras en su versión 2 con 8 megapíxeles. Los modelos más recientes son los módulos de cámaras versión 3 de 12 megapíxeles lanzado el año 2023. Estos últimos modelos fueron producidos en cuatro variantes:

- Estándar de espectro visible,
- Amplio campo de visión de espectro visible,
- Infrarrojo estándar
- Infrarrojo de amplio campo de visión.

Actualmente se vienen desarrollando nuevos modelos como la cámara HQ (High Quality) los cuales tiene la opción de usar monturas de lentes externas para mejorar la calidad de las imágenes y el modelo de cámara IA el cual tiene capacidad de baja latencia y un alto rendimiento para uso en aplicaciones de redes neuronales.

Figura 17

Módulos de cámaras Raspberry PI V·3



Nota: 04 modelos de módulos de cámaras Raspberry Pi V3. Fuente: Raspberrypi, s.f.

2.3.6. Cámaras de Seguridad IP

Dispositivo electrónico de monitoreo que captura y envía video en forma digitalizada mediante una red IP (protocolo de internet). (Valades, 2024)

2.3.6.1. Funcionamiento

Según Valades (2024) la operación de una cámara IP se fundamenta en 4 etapas:

- **Captura de imágenes**: Procedimiento en el que se captura el video a través de un sensor de captura de imagen.
- **Compresión de video**: Etapa en la que el video es comprimido en estándares de codificación de video H.264 o H.265 para luego ser enviado por la red de manera eficiente.

- **Transmisión a través de una red**: El video comprimido es enviado a través de la red mediate una conexión ethernet o mediante conexión inalámbrica Wi-Fi.
- **Recepción y visualización**: Etapa final donde el video es descomprimido y mostrado al receptor mediante una pantalla.

2.3.6.2. Aspectos Técnicos de Cámaras IP

A continuación, se definirán algunas especificaciones técnicas a tener en cuenta al momento de utilizar cámara IP.

- **Sensor**: Hay dos clases, el CCD (Dispositivo de carga acoplada) y el CMOS (Semiconductor complementario de oxido metálico), cuyo contraste se basa en la sensibilidad a la luz y la rapidez con que se convierten señales lumínicas en señales eléctricas. El CMOS es el más asequible y más utilizado en cámaras de videovigilancia.
- **Lente**: componente óptico encargado de enfocar la luz exterior al sensor de imagen.
- **Distancia focal**: Es la separación entre la lente y el sensor, expresada en milímetros como unidad de medición, este indicador establece el nivel de acercamiento o amplitud de la imagen.
- **Apertura**: La apertura del diafragma controla la cantidad de luz que entra al sensor, una apertura amplia resulta ventajosa en situaciones de escasa iluminación, mientras que una apertura reducida conserva los elementos más enfocados.
- **Resolución**: Hace referencia al número de pixeles que alberga la imagen capturada, lo que determina la nitidez y calidad del video. Se categoriza en normas como la HD (720p), la Full HD (1080p) y la 4K (2160p).

2.3.7. OTT

Una OTT (Over-The-Top) hace referencia a servicios disponibles en internet que evitan el uso de plataformas convencionales de telecomunicaciones. Estos servicios comprenden aplicaciones de video bajo demanda, servicios de mensajería y videoconferencias, tales como Netflix, WhatsApp e Instagram. (Mamushiane, 2023)

2.3.8. API

Se puede interpretar la API (Application Programming Interface) como las características que proporciona un servicio de software específico, permitiendo que otros puedan emplearlas para optimizar sus resultados. Usualmente no es un producto en sí mismo, sino que actúa como un enlace entre un software ya desarrollado y otro al que puede ser beneficioso, lo que se denominaría una interacción "software-to-software". (Plaza Estévez, 2016)

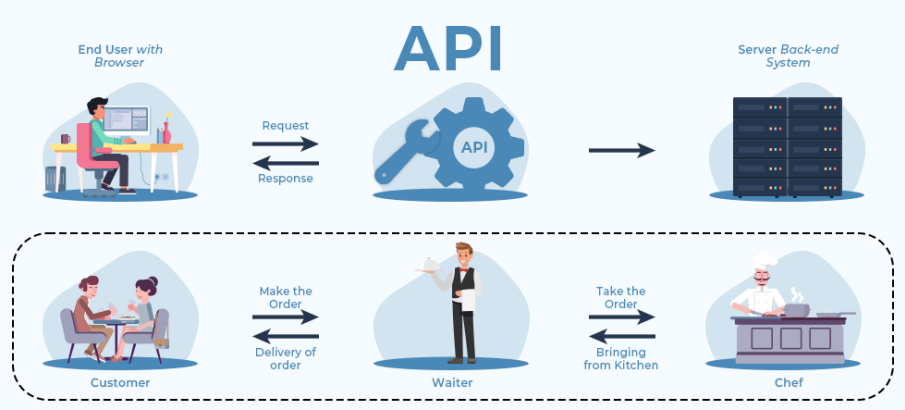
2.3.8.1. Funcionamiento de API

Según (Geeksforgeeks, 2023) el funcionamiento de API (Application Programming Interface) se puede explicar en 4 pasos:

- El cliente inicia una solicitud a través de una URI (Identificador Uniforme de Recursos) de las APIs.
- El API realiza una llamada al servidor.
- Después el servidor envía una respuesta al API con la información solicitada.
- Finalmente, la API transfiere la información al cliente

Figura 18

Funcionamiento de API



Nota: Comparativa de funcionamiento de API con un ejemplo cotidiano. Fuente: geeksforgeeks, 2023

2.3.9. Bot

Es una aplicación de software automatizado que lleva a cabo actividades recurrentes en una red. Esta aplicación sigue directrices concretas para replicar el comportamiento humano. Un bot informático sigue normas e indicaciones para llevar a cabo una tarea programada. Una vez activados puede interactuar entre sí o con los seres humanos mediante protocolos de comunicación de red. Una petición HTTP es un protocolo de comunicación empleados por los navegadores para enviar y recibir información. (Amazon Web Services, s.f.)

2.3.10. API Bot de Telegram

Son aplicaciones que se ejecutan en el programa de Telegram. La API de bots es una interface basada en HTTP creada para programadores interesados en desarrollar bots para Telegram con la finalidad de realizar actividades como: remplazo de sitios web, administración de negocios, creación de herramientas personalizadas, integración con servicios y dispositivos entre otras. (Telegram, s.f.)

2.3.10.1. Proceso de Integración de API Bot de Telegram

El proceso de integración según documentación de Telegram sigue los siguientes pasos:

- Autorizar el Bot: la creación y autorización del bot se lleva a cabo a través de la herramienta de Telegram denominado @BotFather, el cual es el encargado de crear y administrar los bots de Telegram. Este brindará el token que es el identificador de bot con autorización para comunicarse con los servidores de Telegram.
- Realizar solicitudes: las peticiones o consultas a las API de bots de Telegram deben ser enviadas a través de HTTPS en el siguiente formato, https://api.telegram.org/bot<token>/METHOD_NAME donde el campo METHOD_NAME representa los métodos de HTTP GET y POST, los cuales son empleados para enviar información entre un cliente y un servidor.

Telegram ofrece una serie de métodos, cada uno diseñado para fines particulares, los cuales se pueden consultar en la documentación oficial de la Telegram Bot API.

Capítulo III: Diseño del Sistema

3.1. Estudio Preliminar del Lugar de Instalación

En la presente sección se examina el entorno de aplicación con el fin de validar la necesidad y respaldar las decisiones del diseño del prototipo del sistema. A continuación, se presenta las condiciones históricas, climáticas y geográficas del sector de Picol Orccompucyo.

3.1.1. Documentación Histórica y Causas

Aunque los reportes de las instituciones dedicadas al registro de incidencias de incendios forestales como el COER e INDECI, estos catalogan a los incendios forestales como incidencias a nivel distrital o regional, dificultando el registro a escala comunitaria. Se ha verificado la recurrencia de incendios forestales en la zona de estudio,

Trabajos académicos como la de Suarez (2021) documenta eventos en el año 2020, en los meses de octubre y noviembre, llegando a la conclusión que estos son causados por factores antrópicos y malas prácticas de quemas.

Esta situación se refuerza con informes documentados por los diaros locales, donde el 27 de julio de 2022 se produjo un incendio forestal que devasto hasta 5 hectareas en el sector de Picol Orccompucyo, cerca al centro arqueológico Raqayraqayniyuq. Asimismo, el 16 de agosto de 2024, se produjo un incendio de gran magnitud en la zona de Picol, que devastó hasta 400 hectareas de superficie forestal, que fue atribuido a la actividad humana.

Para reforzar la idea de las causas de los incendios forestales, según el MINAM (Ministerio del Ambiente) indica que el 98% de incendios forestales son causados por la acción humana, con actividades relacionadas a las quemas agrícolas, quemas con fines de pastoreo y quemas accidentales.

3.1.2. Temporadas de Riesgo y Cronología

Según Suarez (2021) la comunidad campesina de Picol Orccompucyo presenta una caracteristica climatica bimodal, es decir temporada seca de abril a octubre y otra temporada de lluvias de noviembre a marzo. Siendo los meses de junio a noviembre donde se presenta temperaturas mas altas, vientos fuertes y acompañado a las bajas precipitaciones de lluvia lo cual pone a la comunidad en un alto riesgo para la ocurrencia de un incendio forestal.

3.1.3. Condiciones Climáticas y Geográficas del Sector

Los incendios forestales pueden agravarse en condiciones climáticas específicos: déficit hídrico, altas temperaturas, estaciones secas y fuertes vientos, que favorecen la extensión del fuego abarcando grandes áreas. (CENEPRED, 2021)

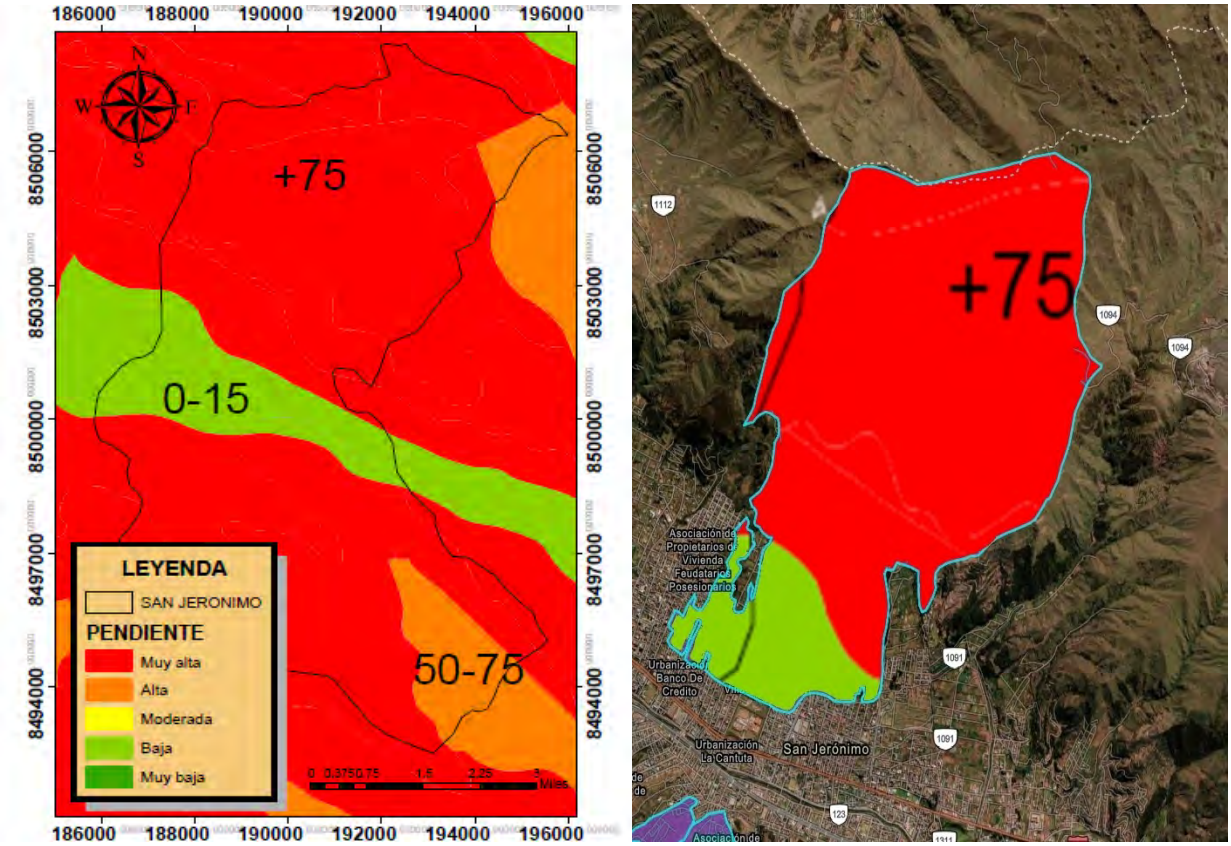
Si bien no existe un estudio específico del comportamiento de los vientos en la comunidad campesina de Picol Orccompucyo, la referencia de CENEPRED (Centro Nacional de Estimación, Prevención y Reducción del Riesgo de Desastres) sobre el mapa de velocidades medias de vientos en la región de Cusco (obtenidos a 10 metros de la superficie) permite inferir que la velocidad máxima es de 5.4 m/s lo que corresponde a un grado de fuerza 3 en la escala Beaufort, catalogado como brisa floja.

Las direcciones de los vientos en la región de Cusco varían y están influenciados por la geomorfología local pero los patrones indica que en los meses de junio a noviembre prevalecen del norte y del noroeste. (Weather Spark, s.f.)

Otro factor que contribuye favorablemente a la expansión de un incendio forestal es la forma de terreno. Los suelos con pendientes pronunciadas propagan el fuego de forma rápida. Según Suarez (2021) la comunidad Campesina de Picol Orccompucyo presenta pendientes catalogadas desde bajas hasta muy altas tal como se muestra en la figura 19.

Figura 19

Morfología de pisos de comunidad campesina de Picol Orccompucyo



Fuente: Suarez, 2021

3.1.4. Características de la Flora como Combustible

Según Suarez (2021) la comunidad de Picol Orccompucyo presenta zonas forestales con suelos de pajonales espinosos y matorrales naturalmente secos. Este tipo de vegetación es un factor primordial para la expansión del fuego.

Así mismo indica que en la comunidad campesina predomina el eucalipto, una especie de árbol pirófila cuyas hojas están compuestas por aceites volátiles, lo cual actúa como combustible para estos fuegos no controlados.

3.1.5. Comportamiento del Humo

El comportamiento del humo depende de las estaciones del viento en la zona, generalmente el humo tiene un comportamiento ascendente esto debido a que: el calor generado por las quemas produce un fenómeno de convección del aire, haciendo que el aire caliente (menos denso) asciende de forma vertical. Los vientos al tener un comportamiento de movimiento horizontal interactúan con el humo, lo que produce que la columna vertical del humo se incline (Andrews, 1986).

El color de humo depende de factores entre ellos la proporción de concentración de vapor de agua y hollín (carbono no quemado). El humo blanco indica la etapa inicial de un incendio forestal ya que existe un calentamiento con alta humedad o combustible ligero. A medida que la combustión se intensifica y sumado con la reducción de oxígeno y el aumento de hollín traerá como consecuencia el humo negro u oscuro lo que indica una intensidad fuerte de fuego y combustión seca (Krzeczkowska, 2024).

En base estos conceptos y los patrones de viento en la zona (sección 3.1.3) se infiere que el comportamiento del humo forestal en el sector de Picol Orccompucyo tendrá un comportamiento horizontal. En condiciones de vientos fuertes, la columna de humo tendera a inclinarse predominantemente hacia el sur y el suroeste. En cuestión de color a considerar se tendrá en cuenta los colores claros (blanco a gris claro) ya que este indica una combustión inicial de la flora.

3.2. Metodología de Diseño y Justificación del Prototipo

El proyecto se centra en el desarrollo y validación de un prototipo funcional, adoptando una técnica de diseño que sustenta la viabilidad y escalabilidad.

3.2.1. Metodología de Diseño del Sistema

Dada la delimitación del alcance del proyecto de ingeniería respecto al desarrollo y validación del sistema con una sola cámara, se optó por una metodología bottom – up.

Este enfoque se centró en el desarrollo de un prototipo funcional, permitiendo la validación de la idea antes de implementar un sistema a gran escala.

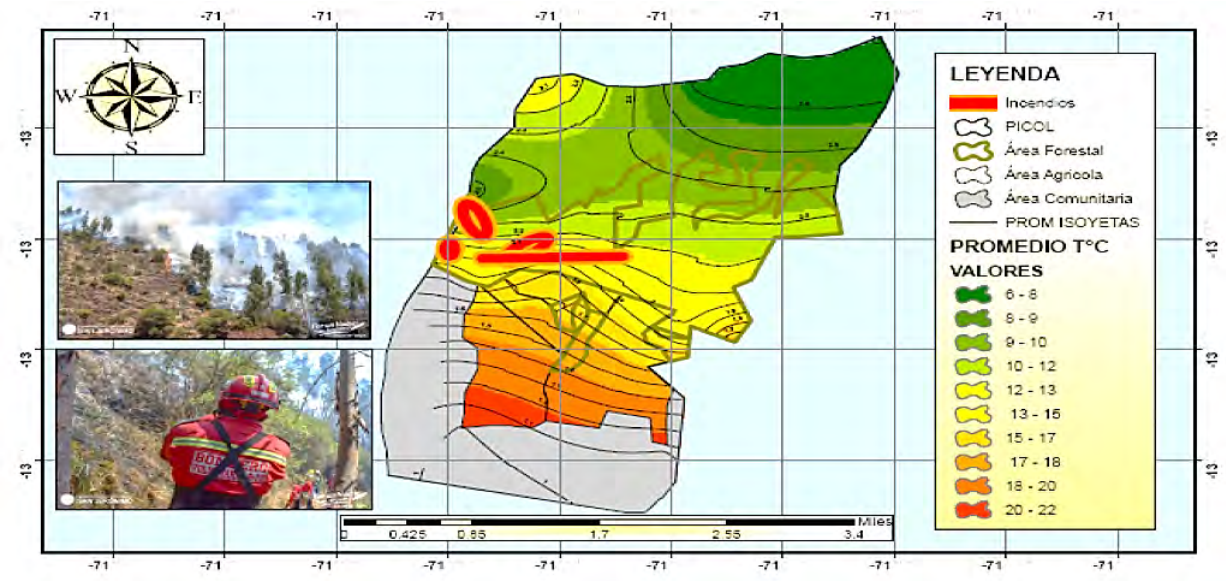
3.2.2. Justificación del Área de Monitoreo

Según Suarez (2021), las áreas de intersección entre zona agrícola con zonas forestales son los sectores donde se inician los incendios forestales, siendo las quemas de rastrojos son los principales detonantes de fuego.

Si bien el mapa de riesgo histórico del año 2020 se concentra hacia el oeste del sector y en el año 2022 se registró otro evento hacia el este, la probabilidad de la ocurrencia de un incendio forestal persiste en toda esta zona de intersección.

Figura 20

Mapa de riesgo potencial de incendios forestales.



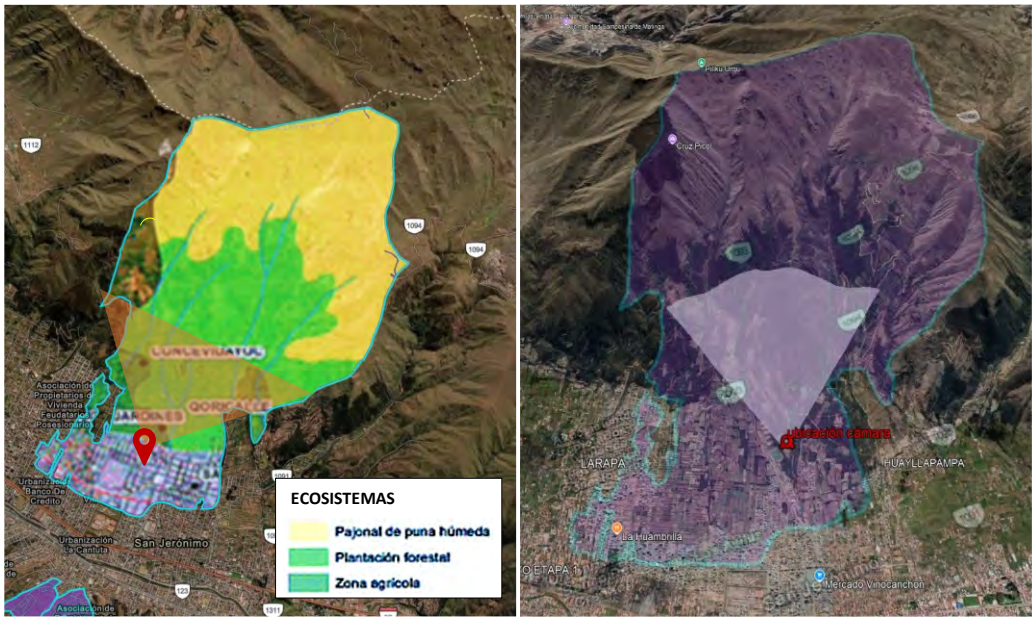
Fuente: Suarez (2021)

Para la fase de pruebas del prototipo, se tendrá en cuenta un punto de localización que cumpla con los siguientes requerimientos: representatividad del ecosistema, viabilidad para las pruebas controladas, línea de visión a la zona problemática y factibilidad técnica de instalación.

En la figura 21 se muestra el punto de prueba con mayor potencial por la existencia de infraestructura cercana donde instalar el prototipo, como el salón comunal de Picol Orccompucyo y viviendas aledañas, los cuales tienen buena línea de visión a estos entornos problemáticos.

Figura 21

Mapa de cobertura de cámara instalada



Nota: La Figura muestra la ubicación de la cámara en zonas de transición agrícola y forestal. Modificación de mapa de ecosistemas San Jerónimo Fuente: MD San Jeronimo, 2022

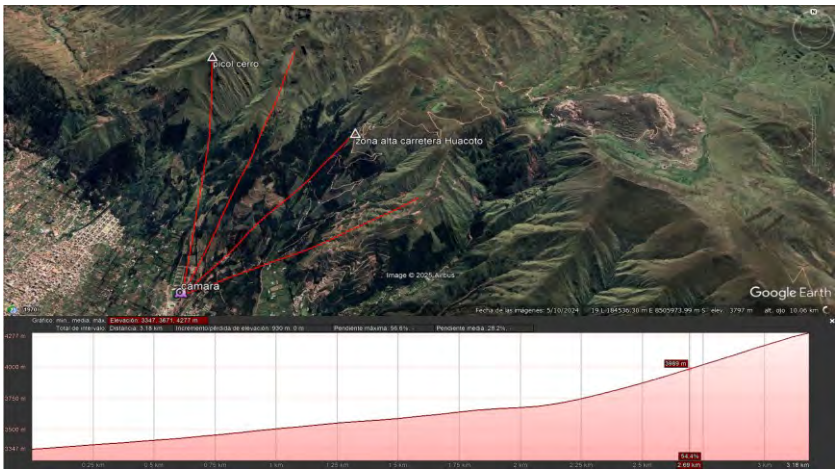
Figura 22

Evidencia fotográfica de campo

El análisis de línea de visión y los perfiles de elevación mostrado en las figuras 23 y 24 muestra que el punto de localización potencial tiene un campo de visión que abarca la mayor parte de ecosistemas desde la zona agrícola terminando en las zonas más altas como pajonal de puna.

Figura 23

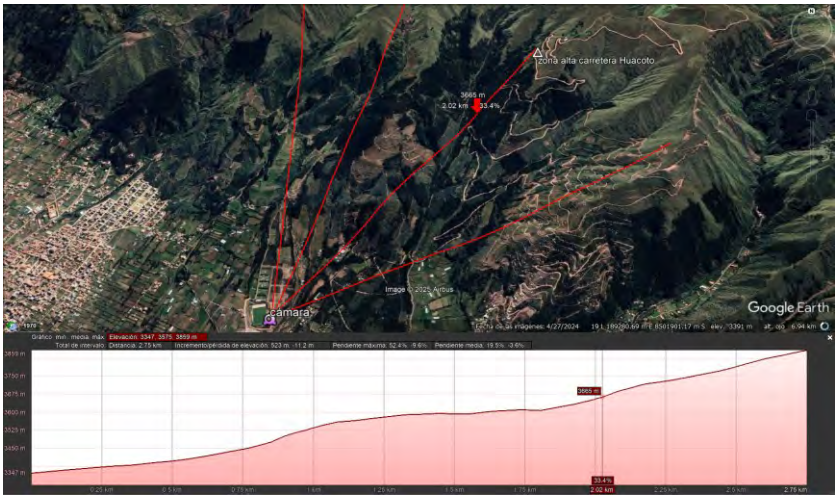
Perfil de elevación del terreno primer punto alto sector Picol Orccompucyo



Fuente: Google Earth Pro.

Figura 24

Perfil de elevación de terreno segundo punto de elevación Picol Orccompucyo



Fuente: Google Earth Pro.

3.3. Especificaciones de Requerimientos

El diseño del prototipo del sistema de detección de incendios forestales utilizando técnicas de procesamiento digital de imágenes, implica una buena elección de herramientas de implementación, que incluyen: dispositivo de captura de imágenes, dispositivo electrónico embebido, programas y algoritmos necesarios para hacer la detección de eventos y envío de notificaciones de alertas de los incendios detectados.

3.3.1. Requerimientos Funcionales

El proyecto de ingeniería debe cumplir con tareas operacionales especificas:

- **Monitoreo constante:** La captura de video debe ser en tiempo real.
- **Precisión en detección:** El prototipo debe tener un alto grado en detección de humo o fuego según sea el caso.
- **Rapidez en tiempo de respuesta:** La generación de alertas debe ser automática e inmediata mediante el envío de notificaciones.

3.3.2. Requerimientos Técnicos

Adicionalmente algunas especificaciones técnicas que debe cumplir el proyecto de ingeniería:

- **Software:** Debe ser capaz de procesar imagen en tiempo real además de incluir librerías para hacer tareas de visión artificial.
- **Sistema operativo:** El software debe ser compatible con sistemas operativos como Windows o Linux.
- **Cámara:** La cámara debe ser capaz de obtener imágenes de buena resolución y operar con protocolos compatibles con la unidad electrónica de procesamiento.

- **Unidad de procesamiento:** El sistema embebido seleccionado debe ser capaz de ejecutar programas de procesamiento de imágenes en tiempo real. Compatible con el sistema operativo Windows o Linux, lo cual será indispensable para implementar algoritmos de análisis de imágenes.
- **Robustez:** El sistema debe operar de manera continua, adaptándose a cambios de iluminación en el entorno forestal.
- **Escalable:** El sistema debe ser capaz de implementarse en diferentes entornos forestales y debe ser ajustable en sus parámetros de programación.
- **Fuente de energía:** El proyecto de ingeniería requiere una fuente de energía ininterrumpida, para energizar los componentes durante las pruebas.
- **Conexión a internet:** El sistema requiere conexión a internet para enviar las notificaciones de alertas detectadas.

3.4. Arquitectura del Sistema

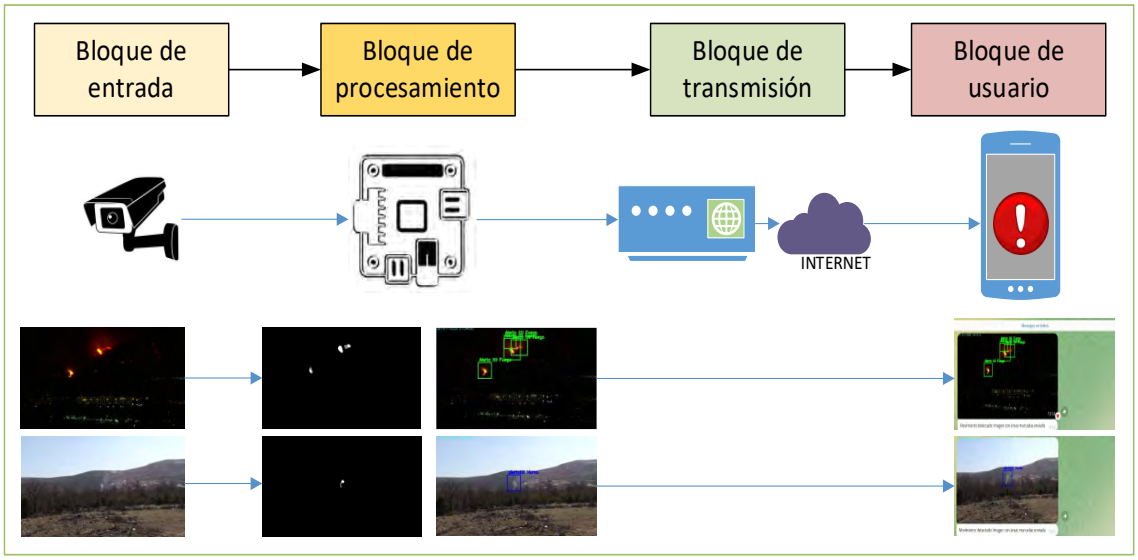
En la presente sección se describe la estructura de los elementos del sistema y su interrelación para asegurar el cumplimiento de los objetivos propuestos. El sistema fue diseñado para funcionar en tiempo real y producir alertas automáticas frente a sucesos, como detección de humo o fuego.

3.4.1. Diagrama de Bloques del Sistema

El sistema está conformado por diferentes bloques interconectados para realizar actividades fundamentales como: captura, procesamiento y análisis, transmisión de data y para finalizar la visualización de la alerta en un dispositivo móvil del usuario. La figura 25 presenta el diagrama de bloques, donde se puede visualizar los elementos clave y su interacción:

Figura 25

Diagrama de bloques del sistema



Nota: Diagrama de bloques de equipos físicos.

- **Bloque de entrada (Captura):** Constituido principalmente por la cámara, que captura el video del entorno forestal en tiempo real y envía la data digitalizada al sistema embebido para su procesamiento.
- **Bloque de Procesamiento:** El bloque recibe el flujo de video digitalizado, realiza el procesamiento de la secuencia de imágenes mediante algoritmos desarrollados en software como: extracción de regiones de interés (ROIs), detección de movimiento, filtrado para eliminar ruido de movimientos pequeños, detección por análisis de color, detección de movimiento continuo, para finalizar la segmentación del área de interés y genera la alerta de detección de humo y/o fuego si se cumple los criterios definidos.
- **Bloque de transmisión:** Bloque donde se realiza envío de las alertas, incluyendo la imagen segmentada, a través de internet mediante el uso de servicios OTT (Over The Top), como Telegram o WhatsApp.

- **Bloque de usuario:** Mediante el uso de un smartphone o PC con accesibilidad a aplicativos de mensajería instantánea, visualiza las alertas generadas por el sistema embebido.

3.5. Diseño de Hardware

El diseño de hardware es esencial para el presente proyecto de ingeniería, una elección inadecuada de estos componentes físicos puede ocasionar gastos innecesarios debido al mal dimensionamiento en las especificaciones técnicas o incompatibilidad de integración entre dispositivos.

3.5.1. Selección y Especificaciones de Dispositivos

En la presente sección se detalla las especificaciones técnicas de los dispositivos candidatos a utilizar, así como la justificación de la elección final del dispositivo.

3.5.1.1. Selección y Justificación de la Cámara

Para la selección de la cámara se evaluó tres cámaras de la empresa Raspberry, teniendo en cuenta la compatibilidad que estos tienen con los sistemas embebidos comerciales.

Tabla 3

Módulos de cámaras de la empresa Raspberry Pi

	Cámara 1	**Cámara 2**	**Camara3**
Modelo	Raspberry Pi V3 Standard (12 MP)	Raspberry Pi V3 Wide (12 MP)	Raspberry Pi HQ Camera (12.3 MP)
Sensor	Sony IMX708	Sony IMX708	Sony IMX477
Formato de imagen	Hasta 4608 x 2592 píxeles.	Hasta 4608 x 2592 píxeles.	Hasta 4056 x 3040 píxeles.
Resolución de video	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2028 × 1080p50, 2028 × 1520p40 1332 × 990p120

Enfoque	Autoenfoque por detección de fase (PDAF)	Autoenfoque por detección de fase (PDAF)	Enfoque manual dependiente de la lente.
Focal-ratio	F1.8	F2.2	Depende del lente
Campo de visión horizontal	66°	108°	Dependiente de la lente.
Campo de visión vertical	41º	63º	Dependiente de la lente.
Campo de visión diagonal	76º	120º	Dependiente de la lente.
Interfaz	Puerto CSI-2	Puerto CSI-2	Puerto CSI
Costo	S/. 149	S/. 207	S/. 640

Al comparar las tres cámaras y en base a las características se optó por seleccionar la cámara Raspberry Pi V3 (Wide) por sus características: capacidad para utilizar distintos valores de resolución de video, FPS, capacidad de autoenfoque y sobre todo por el amplio campo de visión (Wide Field of View), el cual es un parámetro importante para capturar video de grandes escenas en la zona de riesgo. Asimismo, se seleccionó este modelo porque permite operar en una resolución de 1920 x 1080, la cual, según Bu y Samadi (2019), es la resolución con mejores resultados logrando precisión de 97% en detección de incendios. Aunque el modelo HQ camera posee un sensor de 12.3 MP, la V3 ofrece paridad técnica en resolución de video operativa con un menor costo de compra y mayor facilidad de configuración.

El modelo de cámara dispone de la interfaz MIPI CSI-2, diseñado para transferir datos a una alta velocidad, mediante la transmisión de bits de datos desde la cámara directamente al procesador de los sistemas embebidos compatibles o que disponen dicha interfaz.

El modelo seleccionado dispone de documentación y librerías compatibles para el uso con OpenCV.

3.5.1.2. Selección y Justificación del Sistema Embebido

Para la selección del sistema embebido se tomó en cuenta la capacidad de realizar tareas de procesamiento de imágenes en tiempo real, bajo consumo para utilizarlos como estaciones de monitoreo y la accesibilidad en el mercado local y nacional.

Entre los microcomputadores con los requerimientos de accesibilidad y precio se tienen los siguientes:

Tabla 4

Sistemas embebidos propuestos

	Raspberry Pi 5	**Raspberry Pi 4B**	**NVIDIA JetsonNano**
Procesador	Quad-core ARM Cortex-A76 (2.4 GHz)	Quad-core ARM Cortex-A72 (1.5 GHz)	Quad-core ARM Cortex-A57 (1.43 GHz)
GPU	VideoCore VII	VideoCore VI	NVIDIA Maxwell con 128 núcleos CUDA
RAM	8 GB LPDDR4X	8 GB LPDDR4	4 GB LPDDR4
Almacenamiento	MicroSD + eMMC (opcional)	MicroSD	MicroSD
Interfaz de cámaras	Doble puerto CSI	Doble puerto CSI	Un puerto CSI
Conectividad	Gigabit Ethernet, Wi-Fi 6, Bluetooth 5.0	Gigabit Ethernet, Wi-Fi 5, Bluetooth 5.0	Gigabit Ethernet, sin Wi-Fi ni Bluetooth
Aceleración de IA	No	No	TensorRT y aceleración CUDA
Alimentación	USB-C (5V, 5A)	USB-C (5V, 3A)	Barrel Jack (5V, 4A)
Costo	S/. 579	S/. 509	S/.1490

Al comparar los tres sistemas embebidos con capacidad de realizar tareas de procesamiento de imágenes en tiempo real, se decidió elegir a la Raspberry Pi 5 por tener una mayor potencia de procesamiento en el CPU, lo cual es ideal para el proyecto de procesamiento de imágenes en tiempo real y ofrece margen para futuras ampliaciones o mejoras de software. Dispone de interfaces de conectividad necesarias para el proyecto de ingeniería y sobre todo por disponer la interfaz de cámaras compatibles con las propuestas en pasos anteriores.

En cuanto al costo se puede apreciar que las Raspberry Pi 5 es más económica que la Jetson Nano y con capacidad suficiente para el proyecto de ingeniería.

3.5.1.3. Selección de Dispositivo de Alimentación Eléctrica

El modelo de Raspberry Pi 5 al ser uno de los últimos sistemas embebidos que salió en el mercado, con capacidades superior a sus predecesores para realizar tareas de procesamiento de imágenes en tiempo real. Conlleva a que el sistema embebido requiera más potencia para realizar estas tareas.

Según fabrica, la Raspberry Pi 5 debe utilizar una fuente de alimentación con las siguientes especificaciones mostradas en la siguiente tabla:

Tabla 5

Especificaciones técnicas de fuente de energía de Raspberry Pi 5

C**aracterísticas**	**Especificaciones**
Voltaje de entrada	220 V AC
Voltaje de salida	5.0 V DC
Corriente máx. salida	5.0 amperios.
Potencia total	27 watts
Conector	USB-C (Power Delivery)

3.5.1.4. Selección de Dispositivos de Enfriamiento del Sistema

Los sistemas de enfriamiento y ventilación son esenciales para los sistemas embebidos y de esto depende que el sistema funcione de manera eficiente y evita que los dispositivos internos se sobrecalienten.

La Raspberry Pi 5 según características técnicas tiene una operación óptima de funcionamiento entre los 0º C a 80º C. A partir de los 85º C se produce una reducción de rendimiento o más conocido como thermal throttling.

Para el proyecto de ingeniería se utilizó un Active Cooler aprovechando que la Raspberry Pi 5 posee un conector dedicado para ventilación, en la tabla 6 se muestra las características del Active Cooler.

Tabla 6

Especificaciones técnicas de Active Cooler Raspberry Pi

Características	**Especificaciones**
Tipo	Activo (ventilación más disipadora)
Velocidad	4000-8000 RPM
Consumo de energía	5 V DC / 0,2W-1W
Control de temperatura	PWM controlado por Raspberry

3.5.1.5. Selección del Modem de Internet

Uno de los requerimientos del sistema es el acceso a internet en una zona donde no se dispone de infraestructura de red fija. Por ello que se seleccionó un modem 4G LTE cuyas características principales son la autonomía energética para el periodo de pruebas y su portabilidad permitió trasladar a diferentes puntos. Se utilizó este dispositivo para optimizar recursos ya que se disponía de este hardware, de esta forma validando la viabilidad del prototipo antes de implementar

un sistema de comunicación autónomo para el sistema. El modem seleccionado es el ZTE MF920U y las características principales se muestra en la siguiente tabla:

Tabla 7

Características modem Wi-Fi 4G LTE

Características	**Especificaciones**
Tecnología	LTE CAT 4G
Banda Wi-Fi	2.4 GHz
Sistemas operativos	Soporta WINDOWS / MAC OS/ LINUX
Batería	Batería Li-ion de 2000 mAh
Tiempo de trabajo	8 horas
Consumo energético	4W

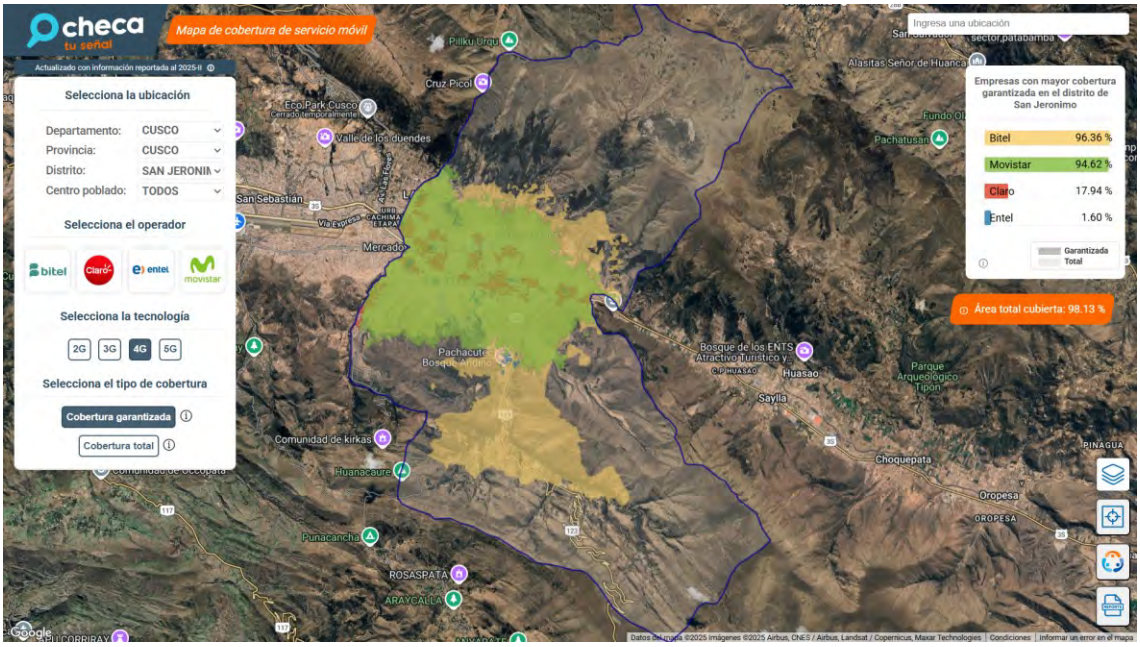
3.5.1.6. Selección de Operador y Análisis de Cobertura Celular 4G

Una vez definida la tecnología de comunicación, se procedió a un análisis de cobertura celular en la ubicación seleccionada, sirviendo de base para futuros despliegues. La herramienta de OSIPTEL se utilizó para verificar la cobertura, identificando a Movistar, Bitel y Entel como las operadoras con mayor presencia 4G en la zona (figura 26 y 27).

Adicionalmente, se evaluó el RSRP (Reference Signal Received Power), o potencia media de la señal recibida, mediante Network Cell Info lite. El análisis mostró que Claro (Banda 28/700 MHz) superó a Bitel (Banda 7/2600 MHz) en potencia y calidad en la ubicación de interés. Claro registró una señal más fuerte, alcanzando -94dBm, y una calidad superior, -16 dBm. A pesar de que Bitel tiene más antenas cercanas, esta diferencia se atribuye a la frecuencia de sus bandas. Estas mediciones serán cruciales para seleccionar la operadora a utilizar en campo.

Figura 26

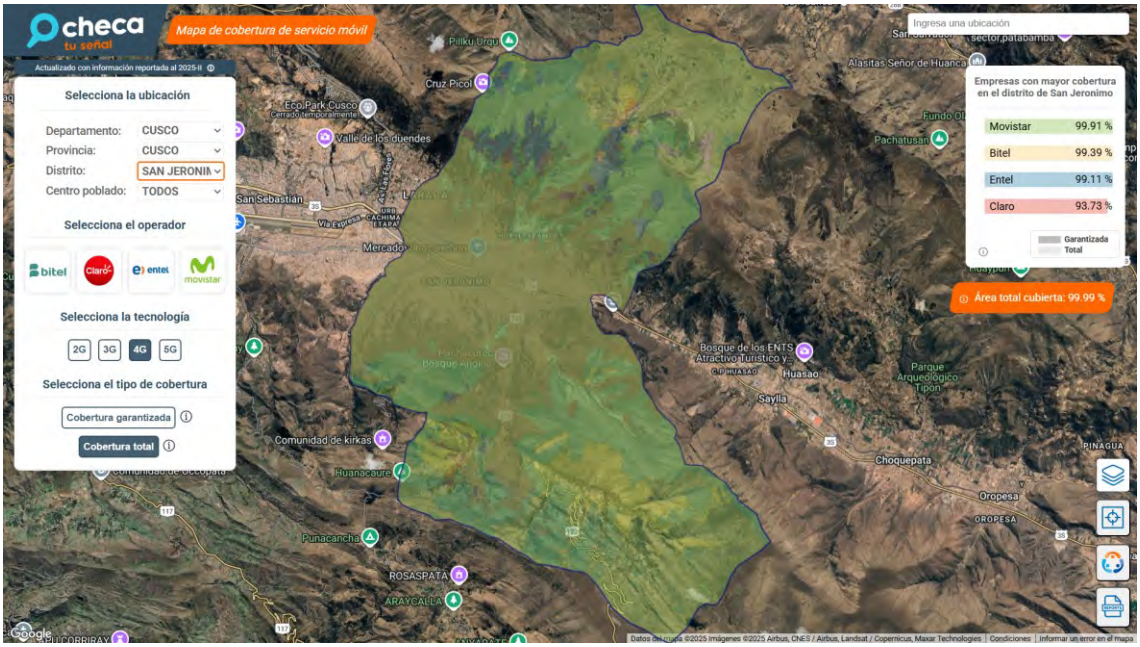
Mapa de cobertura celular 4G operadores en sector de San Jeronimo Cusco. (Garantizado)



Nota: cobertura garantizada de cobertura móvil 4G Fuente: Checa tu señal - OSIPTEL, s.f.

Figura 27

Mapa de cobertura celular 4G operadores en sector San Jeronimo Cusco. (Desempeño variable)



Nota: Cobertura Total con desempeño variable de cobertura móvil 4G Fuente: Checa tu señal - OSIPTEL, s.f.

Figura 28

Medición calidad de servicio (QoS) y rendimiento red LTE/4G operadoras Claro y Bitel.

Fuente: Aplicativo movil Network Cell Info Lite

Figura 29

Mapa de cobertura celular 4G operadora Claro y Bitel

Fuente: Aplicativo movil Network Cell Info Lite

3.5.2. Diagrama de Conexión Lógica y Disposición Física

En la figura 30 se muestra el diagrama de conexión lógica de los componentes físicos seleccionados, el flujo de datos y el tipo de señales que los conectan. A continuación, se describe los bloques principales:

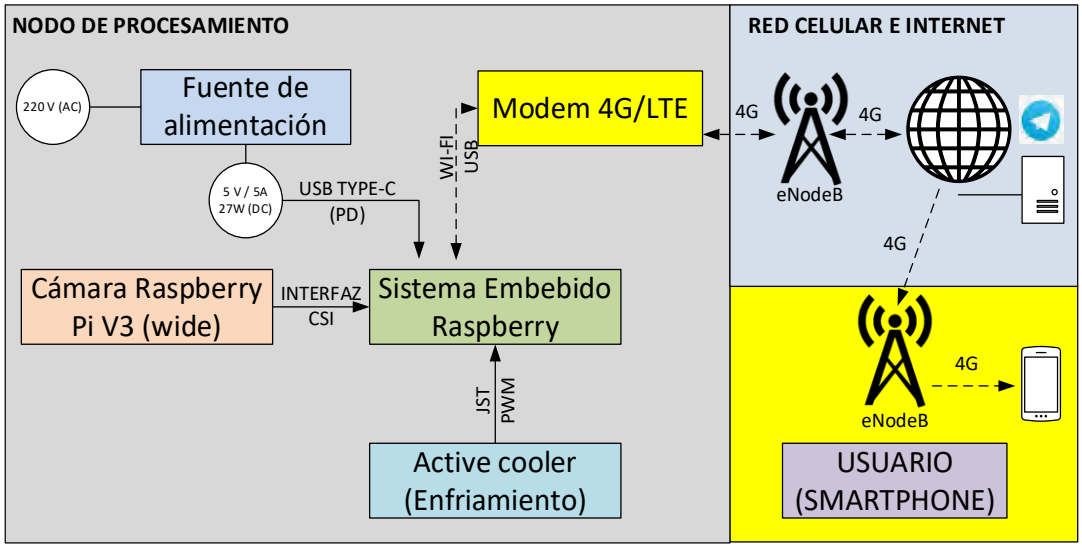
- **Nodo de procesamiento:** Representa al núcleo central del sistema. Se basa en la arquitectura del sistema embebido Raspberry Pi, el cual es la unidad de procesamiento central, recibe la señal de video digital desde la cámara Raspberry Pi V3 (Wide) a través de la interfaz MIPI CSI-2, el cual es un protocolo optimizado para transferencia de datos desde sensores de imagen a una gran velocidad.

  El tema de regulación térmica del procesador se logra mediante un control en lazo cerrado del mismo hardware, donde una señal de ancho de pulso (PWM) es emitida por la Raspberry para regular la velocidad del Active Cooler y de esta forma disipando el calor.

  Tras el procesamiento interno de la Raspberry, los datos de texto e imagen de la detección son serializadas y transmitidas hacia un modem 4G mediante una interfaz de red Wi-Fi. El modem representa la capa de enlace de datos, encapsulando la información en paquetes de internet para su enrutamiento hacia el servidor de Telegram.
- **Red celular e internet:** la información se envía a través de la red celular 4G, el cual accede a internet a través de la red de acceso a radio (RAN). Los paquetes son enrutados en Internet específicamente a los servidores de Telegram.
- **Usuario:** A través del aplicativo en el dispositivo del cliente se puede establecer la conexión con el servidor de Telegram para decodificar y visualizar en tiempo real la alerta generada por el nodo de procesamiento.

Figura 30

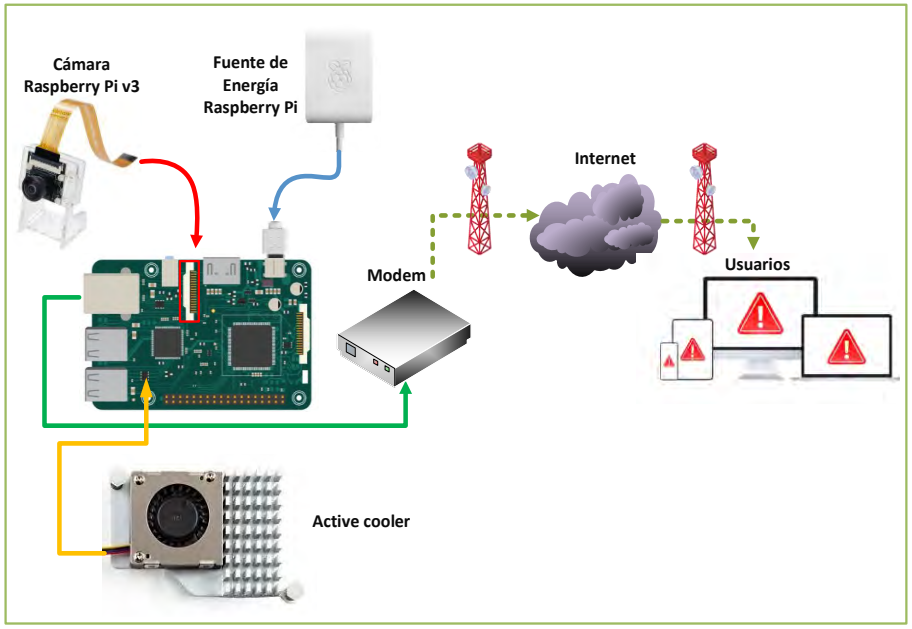
Diagrama de conexión lógica



Adicionalmente en la figura 31 se presenta el diagrama físico de conexión de los dispositivos principales seleccionados.

Figura 31

Disposición física de los componentes del sistema



3.6. Diseño de Software

3.6.1. Algoritmos Planificados

En la presente sección se describe las técnicas y algoritmos planificados para la implementación del prototipo del sistema detector de incendios forestales. En este sentido, se empleó métodos que previamente fueron usados en distintos trabajos académicos donde se tiene datos sobre la efectividad de cada método.

Tabla 8

Comparación de sistemas de detección de fuego mediante metodologías de detección de color, movimiento y forma

Tipo	Método	Autores	Exactitud (%)	Falsos Positivos (%)	Falsos Negativos (%)
Single Expert	Ce	Celik and Demirel	83.87	29.41	0
	Me	Foggia et al.	71.43	53.33	0
	Sv	-	53.57	66.67	21.85
Multi Expert	Ce + Sv	-	88.29	13.33	9.74
	Ce + Me	Di Lassio et al	92.86	13.33	0
	Ce + Me + Sv	Foggia et al.	93.55	11.76	0
Others	RGB + Shape + Motion	Rafiee et al.	74.20	41.18	7.14
	Yuv + Shape + Motion	Rafiee et al.	87.10	17.65	7.14
	Color + Shape + Motion	Habiboğlu et al.	90.32	5.88	14.29
	Color + Shape + Motion	Chen et al.	87.10	11.76	14.29

Nota: Ce: Color expert (color); Me: Motion expert (movimiento); Sv: Shape variation (forma). Fuente: Bu, 2019.

En la tabla 8 se puede apreciar que los sistemas expertos basados en detección de color combinado con métodos de detección de movimiento tienen una exactitud de 92.86%, mientras que fusionada con técnicas en detección de forma aumentan hasta una exactitud de 93.55%.

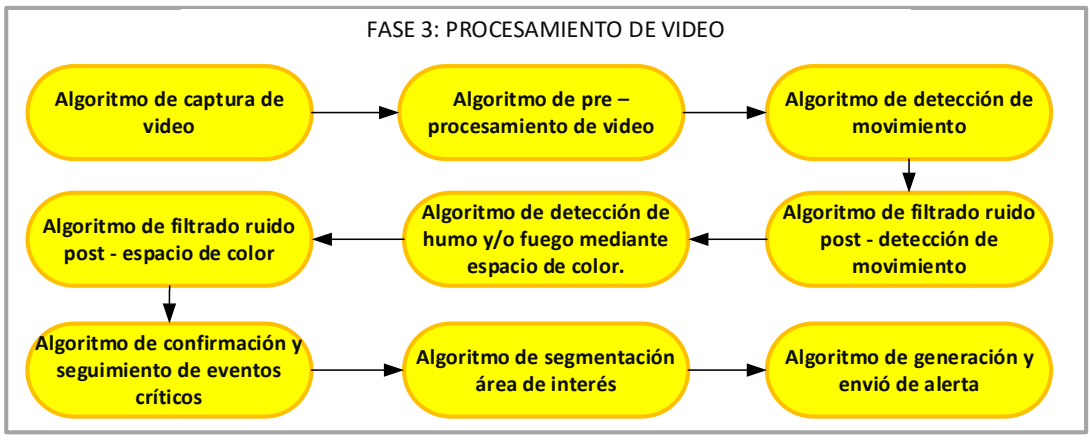
Sin embargo, para el presente proyecto de ingeniería se optó por una estrategia que fusiona movimiento y color, esto debido a que el trabajo al que hace referencia detección mediante características de forma lo utiliza únicamente para detectar fuego.

3.6.2. Algoritmos del Sistema

La presente sección está enfocada en el bloque procesamiento del video (sección 3.4.1.) Esta fase esta divida en secuencia de algoritmos donde se describe toda la lógica del programa realizado para cumplir con el objetivo planteado. En la figura 32 se muestra la representación de los algoritmos unidos secuencialmente que describen toda la lógica del sistema de detección de incendios forestales.

Figura 32

Algoritmos del sistema detector de incendios forestal



3.6.2.1. Algoritmo de Captura de Video

El algoritmo tiene como objetivo establecer la conexión y obtener el flujo de video (secuencia de frames) en tiempo real desde la cámara para proseguir con la etapa de pre –

procesamiento de cada frame. En caso de no establecer una conexión adecuada con la cámara, se muestra el error y termina la ejecución del código.

3.6.2.2. Algoritmo de Pre – Procesamiento de Video

Este algoritmo prepara cada frame de video para su respectivo procesamiento. Se emplean métodos de redimensionamiento de cada frame para su análisis. Esto con la finalidad de mitigar errores en detección y reducir la carga computacional en el sistema embebido. Además, se realiza tareas de extracción de región de interés (ROI), con la finalidad de examinar únicamente las áreas donde existe mayor probabilidad de ocurrencia de incendios forestales.

3.6.2.3. Algoritmo de Detección de Movimiento

Tras el acondicionamiento del frame, se aplica un algoritmo especializado en detección de movimiento. Este algoritmo se basa en comparaciones de valores de pixeles entre fotogramas consecutivos, identificando y mostrando en primer plano todos los pixeles que experimentaron modificaciones. De esta forma generando una primera máscara binaria de movimiento.

3.6.2.4. Algoritmo de Filtrado Post – Detección de Movimiento

Este algoritmo se aplica a la máscara resultante del algoritmo detector de movimiento con la función de eliminar el ruido causado por movimientos rápidos y de menor intensidad a través del empleo de filtros de suavizado y morfológicos.

3.6.2.5. Algoritmo de Detección de Humo y/o Fuego Mediante Espacio de Color

Este algoritmo se fundamenta en la comparación de umbrales en el espacio de color (ej. HSV, RGB o YCbCr). Para ello se toma de referencia los rangos de valores del color de humo y fuego para posteriormente crear una máscara binaria con los pixeles de valores iguales.

3.6.2.6. Algoritmo de Filtrado Post – Espacio de Color

Este algoritmo tiene la función de eliminar el ruido de pequeños elementos que cumplan

condiciones de color similares al humo o fuego. La reducción de ruido se logra mediante el uso de filtros de suavizado y filtros morfológicos

3.6.2.7. Algoritmo de Confirmación y Seguimiento de Eventos Críticos

Este algoritmo fusiona ambas máscaras: movimiento y color.

- **Confirmación inicial:** Se confirma un evento valido si existe intersección de pixeles en ambas máscaras, es decir, es un pixel con tonalidad de humo/fuego y tiene movimiento.
- **Seguimiento temporal:** El segundo algoritmo confirma si la detección perdura a lo largo del tiempo. Para ello, se tiene un número mínimo de fotogramas de análisis consecutivo que confirma finalmente el evento antes de generar la alerta.

3.6.2.8. Algoritmo de Segmentación Área de Interés

El algoritmo de segmentación identifica el conjunto de pixeles que cumplen con las tres condiciones clave (movimiento, espacio de color y la persistencia de la detección). Y realiza un último análisis de regiones conexas cuya finalidad es segmentar las agrupaciones de pixeles que pasen un umbral, de esta forma se resalta solo movimiento relevante. Luego se identifica el área de interés mediante uso de etiquetas y marcos, ya sea de humo o fuego.

3.6.2.9. Algoritmo de Generación y Envío de Alerta

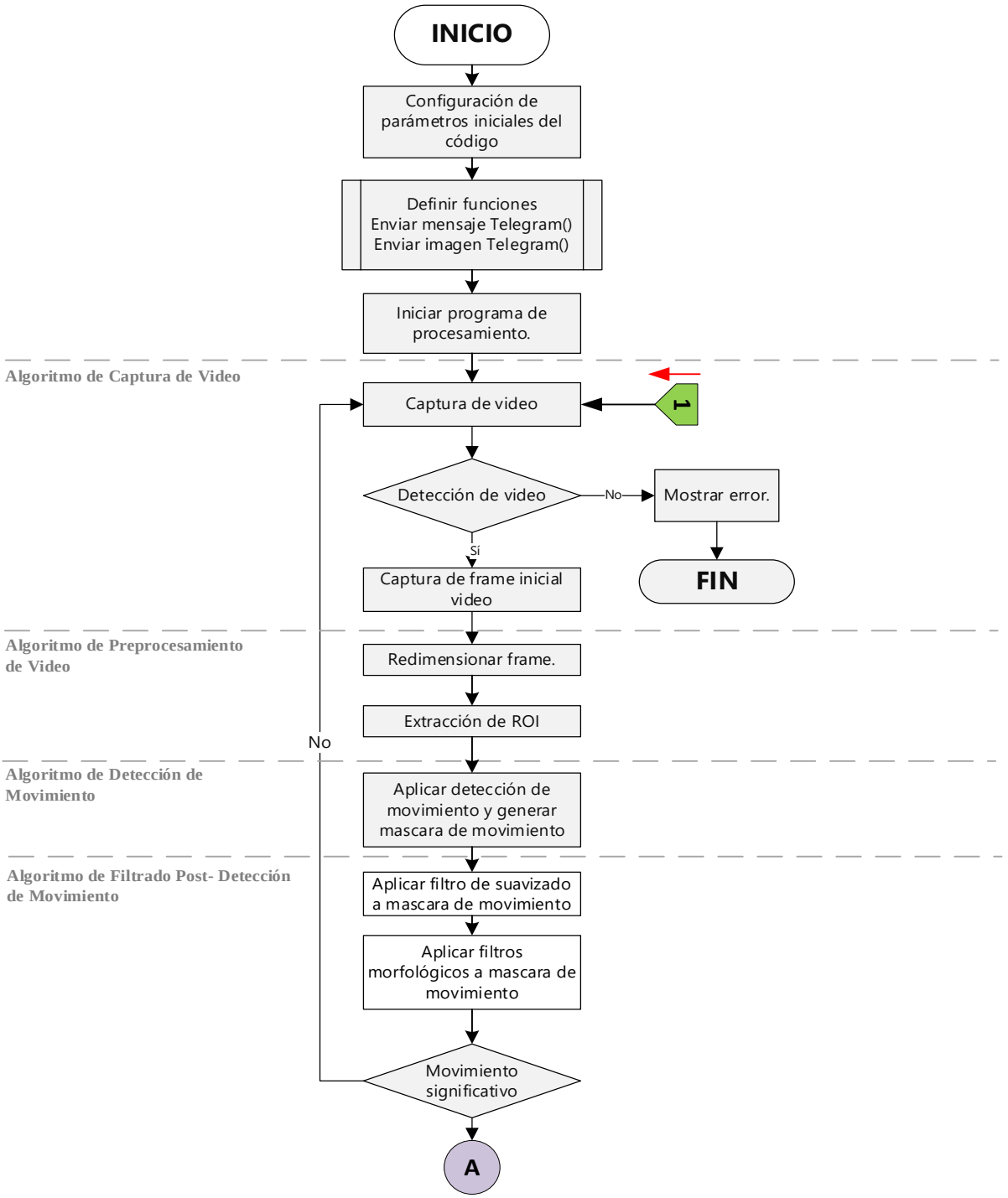
El algoritmo final se encarga de generar la imagen final (con el bounding box aplicado). Esta imagen será guardada momentáneamente, para luego proceder con el envío de la alerta mediante mensajería instantánea como WhatsApp o Telegram.

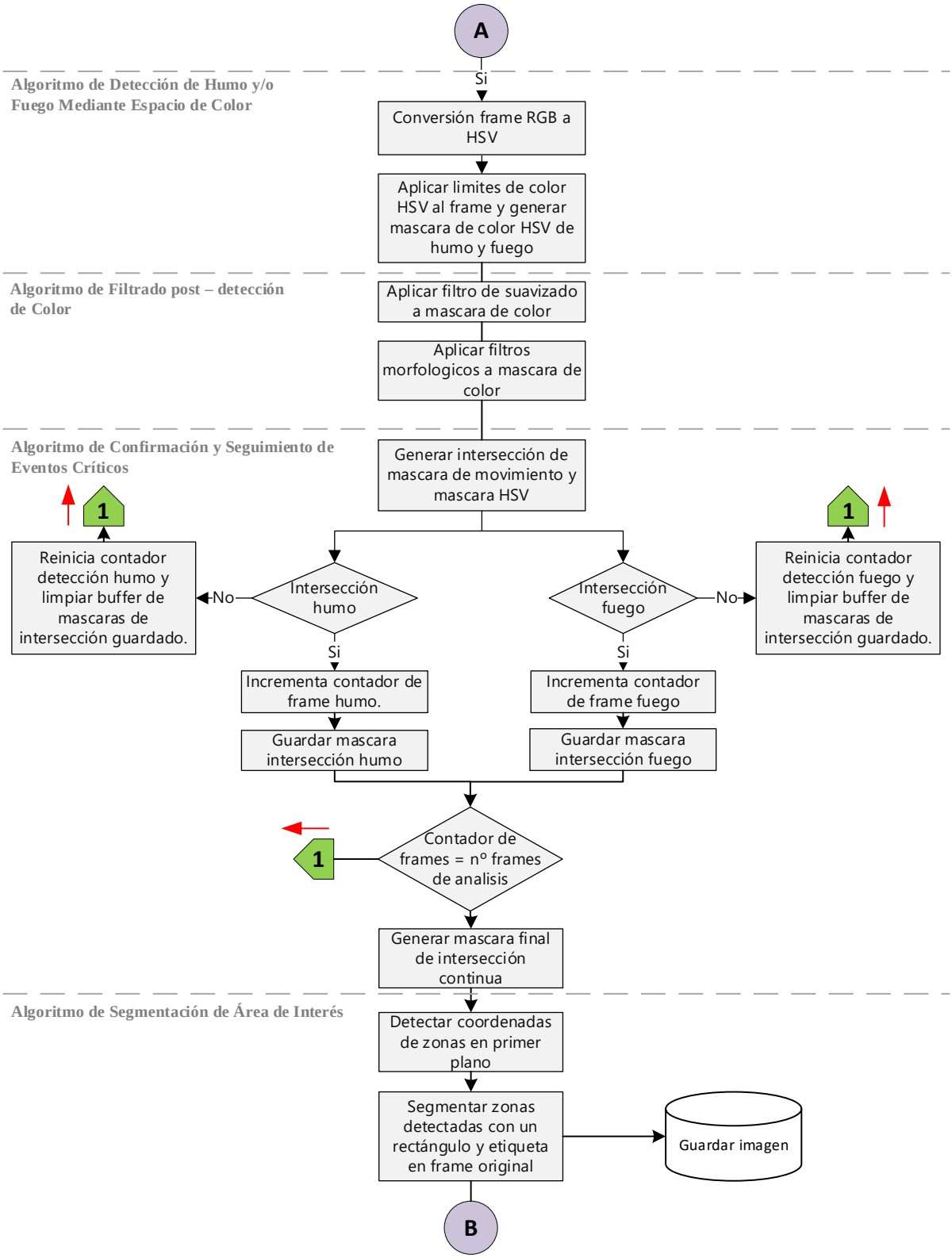
3.6.3. Diagrama de Flujo de Programación

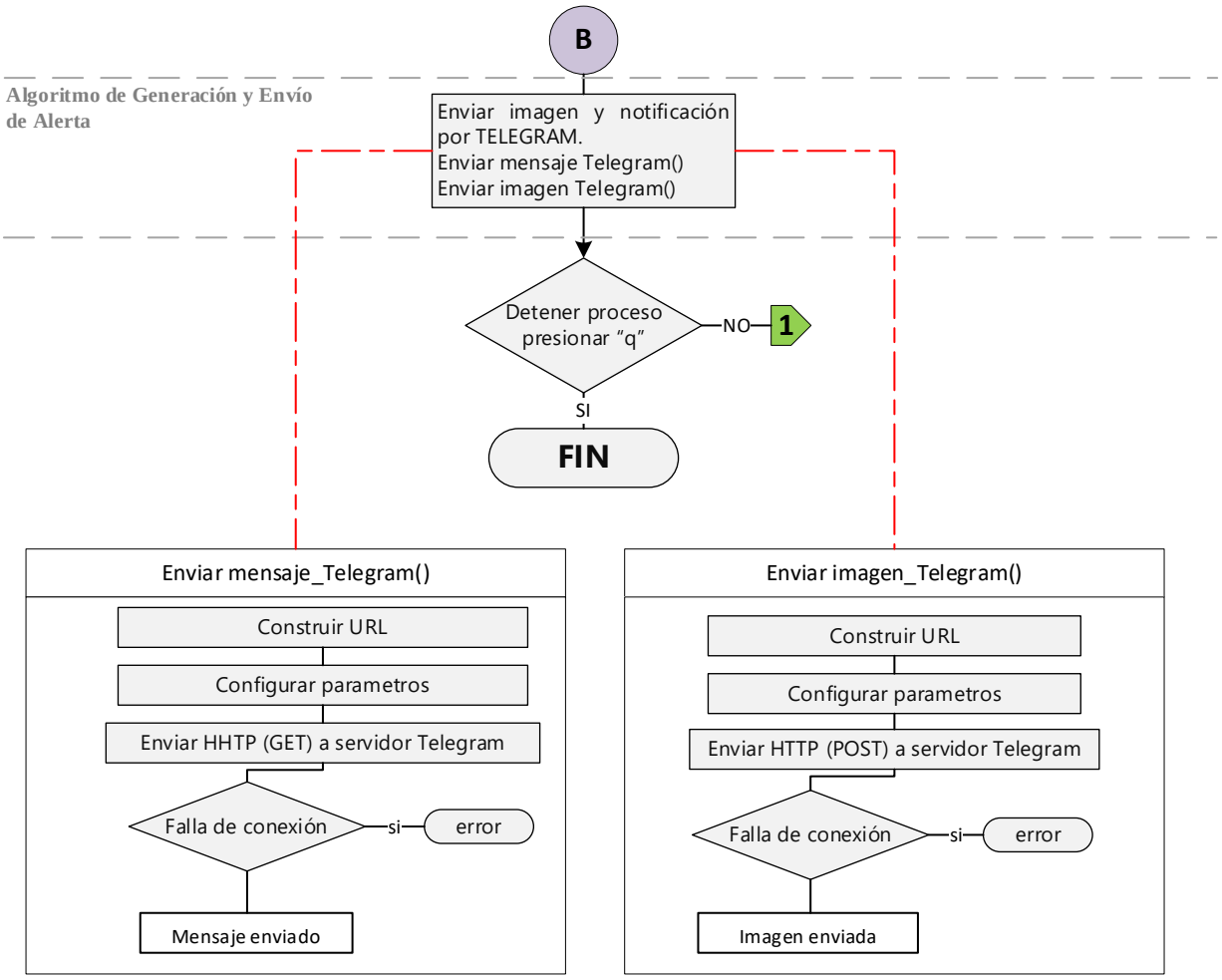
El diagrama de flujo del sistema detector de incendios forestales se muestra en la figura 33, donde se representa de forma visual y secuencial los algoritmos anteriormente conceptualizados en la presente sección.

Figura 33

Diagrama de flujo del sistema detector de incendios forestales




A
Si
Algoritmo de Detección de Humo y/o Fuego Mediante Espacio de Color
Conversión frame RGB a HSV
Aplicar limites de color HSV al frame y generar mascara de color HSV de humo y fuego
Algoritmo de Filtrado post – detección de Color
Aplicar filtro de suavizado a mascara de color
Aplicar filtros morfologicos a mascara de color
Algoritmo de Confirmación y Seguimiento de Eventos Críticos
Generar intersección de mascara de movimiento y mascara HSV
1
Reinicia contador detección humo y limpiar buffer de mascaras de intersección guardado.
No
Intersección humo
Si
Incrementa contador de frame humo.
Guardar mascara intersección humo
Intersección fuego
No
1
Reinicia contador detección fuego y limpiar buffer de mascaras de intersección guardado.
Si
Incrementa contador de frame fuego
Guardar mascara intersección fuego
1
Contador de frames = nº frames de analisis
Generar mascara final de intersección continua
Algoritmo de Segmentación de Área de Interés
Detectar coordenadas de zonas en primer plano
Segmentar zonas detectadas con un rectángulo y etiqueta en frame original
Guardar imagen
B



3.6.4. Herramientas y Librerías Seleccionadas

Para el desarrollo e implementación de los algoritmos propuestos, se requirió la integración de software de terceros y librerías especializadas. Estas herramientas se emplearon para tareas como la configuración del hardware, conexión remota al sistema embebido, codificación de algoritmos y la instalación de dependencias necesarias.

A continuación, en la tabla 9 se detalla los programas necesarios instalados en el computador personal (PC), para tareas puntuales como la configuración inicial de configuración de hardware y establecer la interfaz de programación remota con la Raspberry.

Tabla 9

Software de PC seleccionados para el proyecto de ingeniería

Software PC	Función
Raspberry Pi Imager	Descargar e instalar el sistema operativo en MicroSD para su uso en los sistemas embebidos de Raspberry
PuTTY	Acceder remotamente a terminal de comandos del sistema embebido Raspberry.
Real VNC Viewer	Acceder a la interfaz gráfica del sistema operativo del Raspberry.

Las tablas 10 y 11 presentan los software y librerías necesarios instaladas directamente en el sistema embebido Raspberry Pi 5. Este conjunto de herramientas sirvió para realizar la conexión remota, codificación del algoritmo, instalación de dependencias y la ejecución de los algoritmos desarrollados.

Tabla 10

Software de Raspberry seleccionados para el proyecto de ingeniería

Software de Raspberry	Función
Thonny	Codificar todos los algoritmos planteados en lenguajes de programación Python.
Real VNC Server	Permitir que otros dispositivos se conecten y puedan controlar la interfaz gráfica del sistema embebido.

Tabla 11

Librerías de Python seleccionadas para el proyecto de ingeniería

Librerías de Python	Función
cv2 (OpenCV)	Procesamiento de imágenes y aplicación de algoritmos de visión artificial.
Numpy	Manejo de operaciones numéricas y matriciales.
Requests	Envío de solicitudes bajo el protocolo HTTP (Hipertext Transfer Protocol).
Os	Gestión de archivos, directorios y procesos del sistema operativo.
Datetime	Obtención de fecha y hora del sistema operativo.
Picamera2	Configuración de cámara oficial de Raspberry en Python.

3.7. Justificación de Decisiones del Diseño

Para el proceso de prueba inicial del sistema, se planteó utilizar videos pregrabados en los que existe actividad de humo y fuego. Estos videos fueron descargados de dos bases de datos académico.

- "Wildfire Observers and Smoke Recognition Homepage": Portal web establecido por el centro de investigación de incendios forestales de la Universidad de Split Croacia, Este portal colabora con bases de datos de fotos, videos, y artículos científicos donde se desarrollan algoritmos para detección de humo. (Administrator, s.f.).
- Portal web desarrollada por el Profesor A. Enis Cetin: Este es una página web donde se presenta resultados de su trabajo de investigación, que consistió en el desarrollo de un software para detección de incendios basados en visión computacional. (Cetin, s.f.)

El uso de estas bases de datos permitió evaluar el algoritmo planteado en condiciones controlados, ajustar y calibrar parámetros del sistema en las pruebas repetitivas sin depender de condiciones ambientales reales. A continuación, se justifica el uso de algunos parámetros de diseño los cuales fueron usados en la etapa de implementación.

3.7.1. Resolución

Según Bu y Samadi (2019) en su trabajo de investigación "Sistemas de detección de incendios basados en inteligencia y visión" realizan la comparativa de resolución utilizados en distintos proyectos de investigación. Aunque las configuraciones de resoluciones están orientadas a sistemas de detección de fuego usando vehículos aéreos (UAV) en lugar de tecnología terrestre, estos fueron usados como referencia basándose en sus métricas de desempeño.

En los resultados del trabajo de investigación se observa que mientras mayor es la resolución mayor es la precisión. Obteniendo una precisión de 96.47% para resoluciones de 240 x 135 y una precisión de 97.39% para una resolución de 1920 x 1080. Por lo tanto, se considera estos resultados para la etapa del diseño en caso se requiera modificar la resolución en el algoritmo final.

3.7.2. Algoritmos de Detección de Movimiento y Calibración de Parámetros

OpenCV dispone de algoritmos para detección de movimiento en videos o secuencia de fotogramas entre los más importante en base a su precisión y últimos avances tenemos a MOG2 y KNN por su precisión.

Adinanta et al. (2021) en su trabajo de investigación realiza una comparativa de precisión en detección de personas, donde se utilizaron los métodos anteriormente mencionados, concluyendo que el método de KNN es el método con una mayor precisión en detección de elementos en movimiento.

Adicionalmente, el trabajo de investigación al que hace referencia la documentación de OpenCV realizado por Zivkovic et al. (2006) indica que el método no paramétrico KNN es más simple de implementar y presenta mejores resultados en escenas dinámicas complejas, mientras que el modelo de distribuciones gaussianas (MOG) es más adecuado para escenas estáticas.

Se descarta algoritmos como Random Forest, XGBoost o SVM para esta etapa de sustracción de fondo para detectar movimiento, debido a que estos son modelos de aprendizaje supervisado que requiere un etiquetado previo exhaustivo y alto costo computacional para la segmentación de pixeles en tiempo real. En contraste a KNN o MOG2 que son métodos no supervisados que permite la segmentación automática del fondo y movimiento sin necesidad de un entrenamiento previo.

Por los motivos mencionados (simplicidad, robustez y desempeño) se decidió utilizar el segundo método KNN para la sustracción de fondo. Esta elección se diferencia de los enfoques de estado del arte revisados que utilizan distribuciones gaussianas con mayor predominancia.

Para hallar los parámetros más eficientes del algoritmo KNN los cuales son: history, dis2Threshold y detectShadows se utilizó una metodología de muestreo de Verdad Fundamental (Ground Truth). Para ello, se analizó cuatro videos representativos de la base de datos de prueba. La metodología usada fue:

a) **Análisis fotogramas clave**: identificación de frames que represente inicio de movimiento, movimiento relevante y un fotograma sin movimiento.
b) **Aplicar el algoritmo KNN (K-Nearest Neighbor)**: variando distintos valores de history y dis2Threshold se genera las máscaras de detección en los fotogramas claves.

c) **Creación de máscara de verdad fundamental:** utilizando un editor de fotos se segmenta las partes que representan humo o fuego. Para luego binarizar y generar la máscara binaria manual.

d) **Cálculo de desempeño (Índice de Jaccard):** Una vez obtenida las dos máscaras se utilizará un análisis matemático para hallar el índice de Jaccard el cual tiene por finalidad encontrar la proporción de área de intersección de las dos máscaras, tanto la máscara de verdad fundamental (A) y la máscara generada por el algoritmo de movimiento (B). La ecuación representativa de este análisis esta dado por:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = Índice\ de\ Jaccard$$

Adicionalmente se realizó el cálculo de métricas de desempeño del algoritmo para identificar la cantidad de pixeles que representa humo (verdaderos positivos), pixeles que no representan humo (falsos positivos) y precisión.

$$VP = A \cap B$$

$$FP = B - VP$$

$$FN = A - VP$$

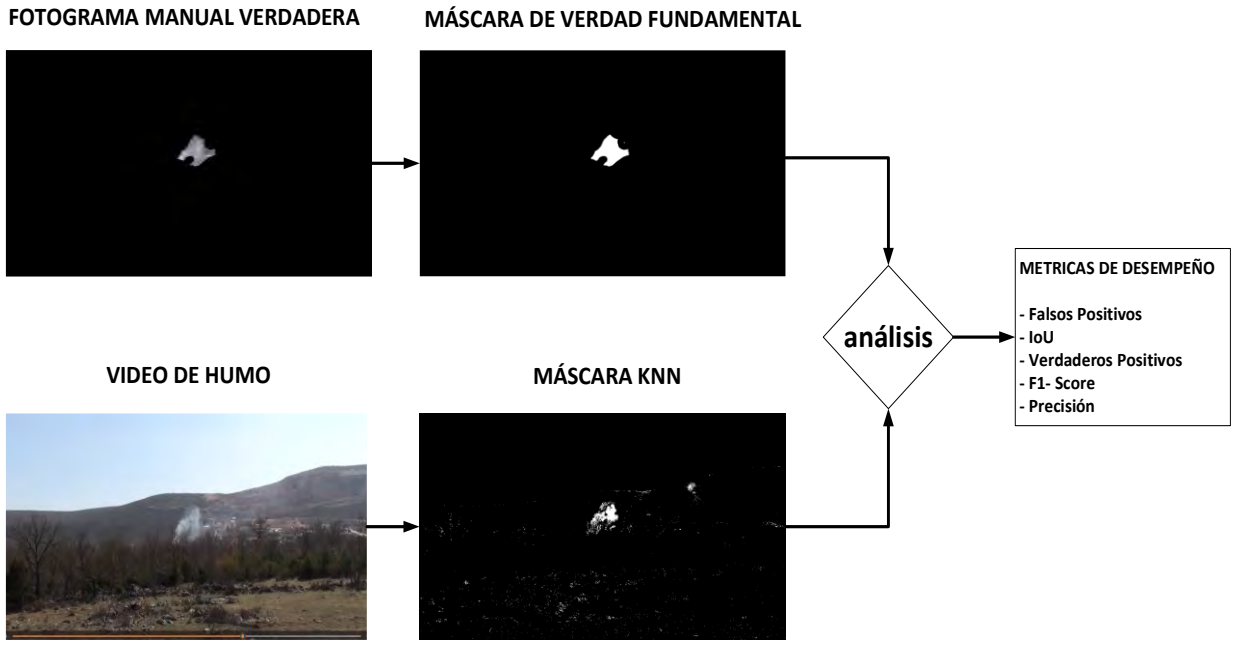
$$Precisión = \frac{VP}{VP + FP}$$

e) **Automatización del proceso**: Se desarrolló un código en Python (Anexos 01). Este código proporciona todo el análisis mencionado incluida el cálculo matemático para encontrar estas métricas que demuestre los parámetros con mejor desempeño.

En la figura 34 se muestra la representación gráfica del método utilizado.

Figura 34

Representación gráfica del algoritmo para calcular parámetros del algoritmo KNN

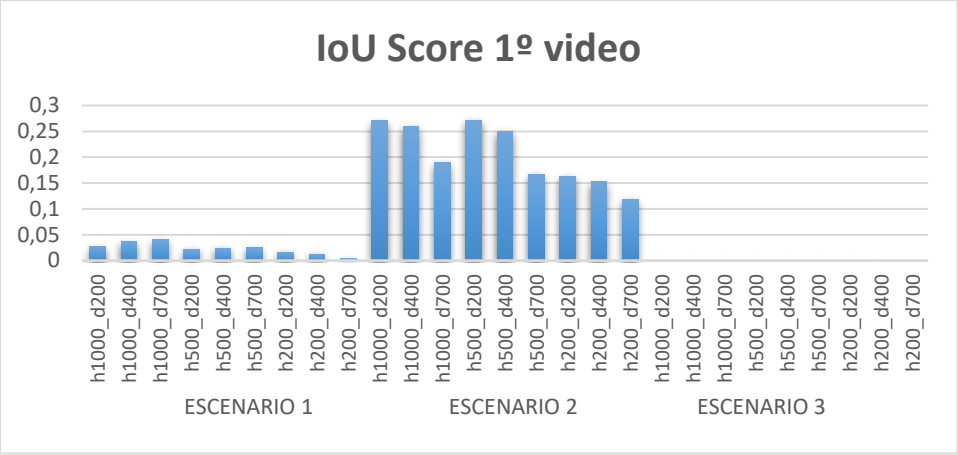


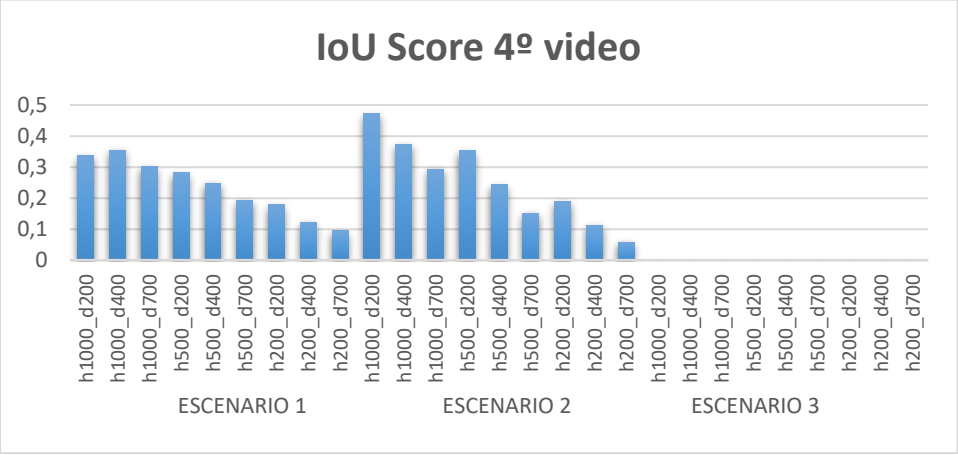
En base a los resultados mostrados en las tablas de Anexo 01 se puede argumentar que los valores donde se obtuvo el mejor desempeño, evidenciado por el índice de Jaccard (IoU):

- History (Historial de muestras): Un valor alto, cercano a 1000, permite que el algoritmo de sustracción de fondo sea más estable y robusto, mejorando la confiabilidad para detectar el movimiento de pixeles que representen humo, ya que en todas las pruebas que se ha demostrado consistentemente que los valores con el mayor historial están asociados a un mejor rendimiento para la detección de movimiento que corresponde a la pluma de humo. La figura 35 ilustra dos escenarios comparativos como ejemplo del hallazgo.

Figura 35

Representación en gráficos del IoU Score de dos videos usados de prueba



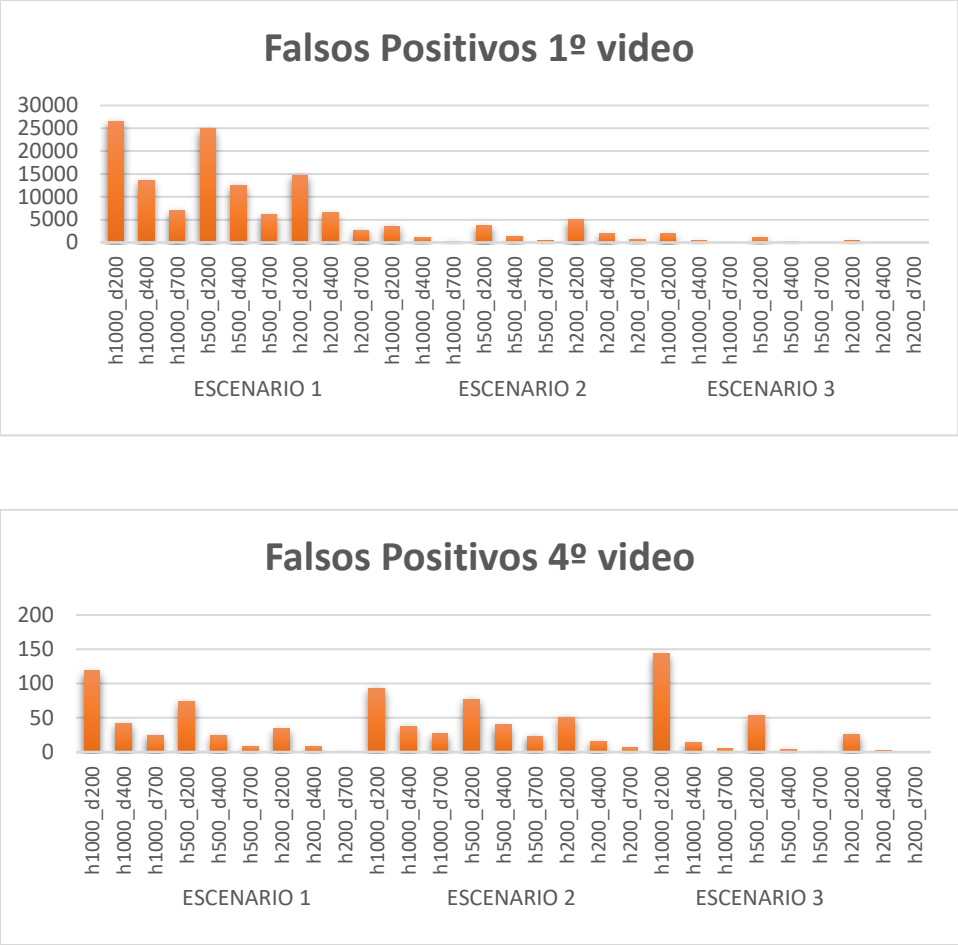


Nota: Los gráficos demuestran que los valores con un History alto y Dist2threshold bajo tienen un mejor desempeño.

- Dist2Threshold: un valor bajo cercano a 200 demostró tener mejor desempeño para detectar pequeñas variaciones de movimiento, lo que ayuda a la detección de humo sutil o de volumen pequeño en su etapa inicial. Sin embargo, este valor capturó la mayor cantidad de falsos positivos (pixeles que no eran humo) y con valores cercanos a 700 se disminuye esta cantidad de pixeles que representen falsos positivos, aunque a expensas de reducir la sensibilidad. Por lo tanto, este parámetro requerirá un umbral de calibración más amplio para encontrar el balance perfecto entre sensibilidad y precisión.

Figura 36

Representación en gráficos de la cantidad de falsos positivos por combinación de parámetros.

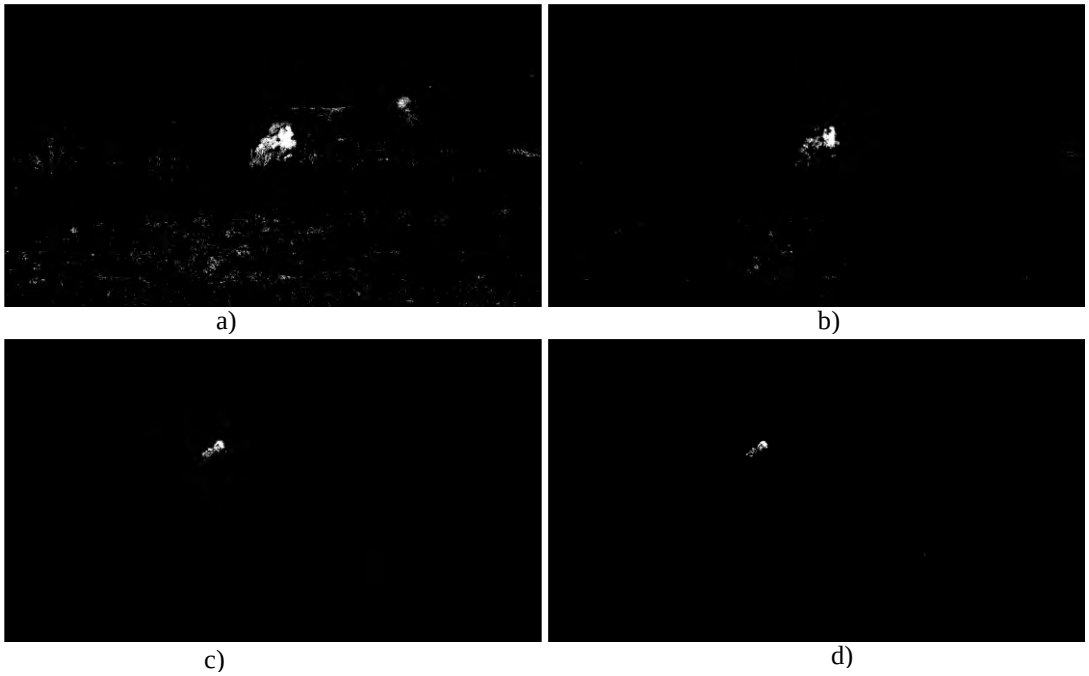


Nota: Mientras el Dist2Threshold es más bajo permitirá detección de pixeles que no son humo.

Este hallazgo se corroboró mediante la visualización de las máscaras generadas, lo cual permitió observar los resultados para cada combinación de parámetros. La figura 37 se muestra las máscaras resultantes tras aplicar el algoritmo de sustracción de fondo KNN.

Figura 37

Representación de máscaras generadas con algoritmo KNN subtraction



Nota: Las máscaras a) y c) representan ejemplares de máscaras generadas con parámetros de history = 1000 y Dist2Threshold= 200 mientras que las máscaras b) y d) con history = 1000 y Dist2Threshold= 700

3.7.3. Elección de Espacio de Color y Cálculo de Rangos

Esta sección describe la metodología utilizada para hallar rangos mínimos y máximos de colores HSV (Tonalidad, Saturación, Valor) en los que está comprendido el humo y el fuego. Para el presente proyecto de ingeniería se llevó a cabo siguiendo estos pasos:

a) **Adquisición de muestras:** Descarga de múltiples imágenes donde se pueda apreciar incendios forestales en diversas condiciones.

b) **Desarrollo de herramienta de calibración**: Codificar el algoritmo para calcular rangos máximos y mínimos de color HSV para humo y fuego.

c) **Análisis estadístico:** Aplicación de análisis estadístico para encontrar los rangos de color HSV del humo y fuego que son utilizados en la implementación final del sistema.

En las figuras 38 y 39 se puede apreciar las imágenes de prueba que se utilizaron para realizar el cálculo de los rangos de espacio de color HSV de humo y fuego respectivamente.

Figura 38

Imágenes de prueba cálculo parámetros HSV – humo

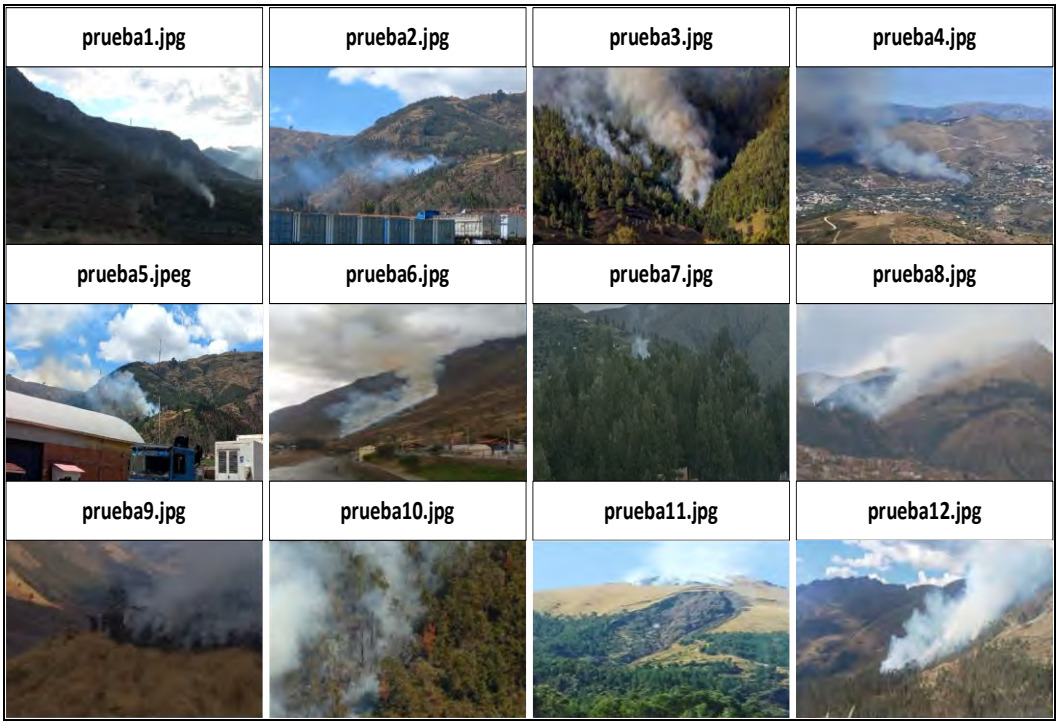
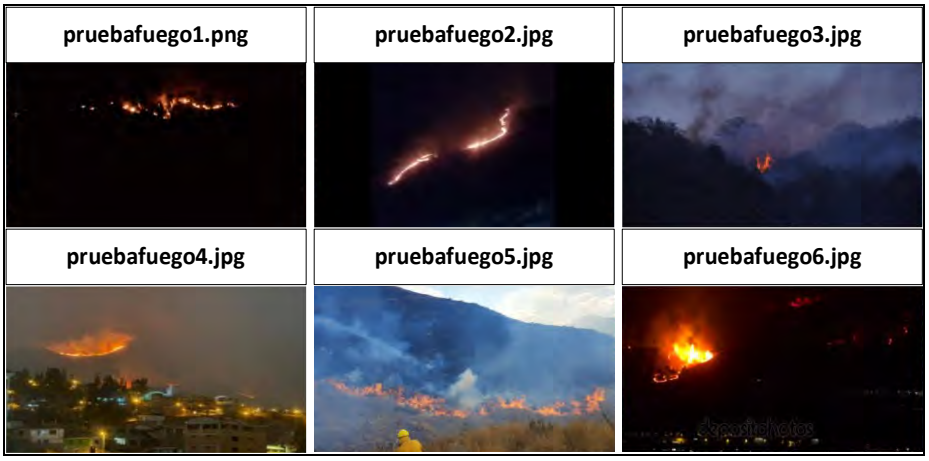


Figura 39

Imágenes de prueba cálculo parámetros HSV – fuego



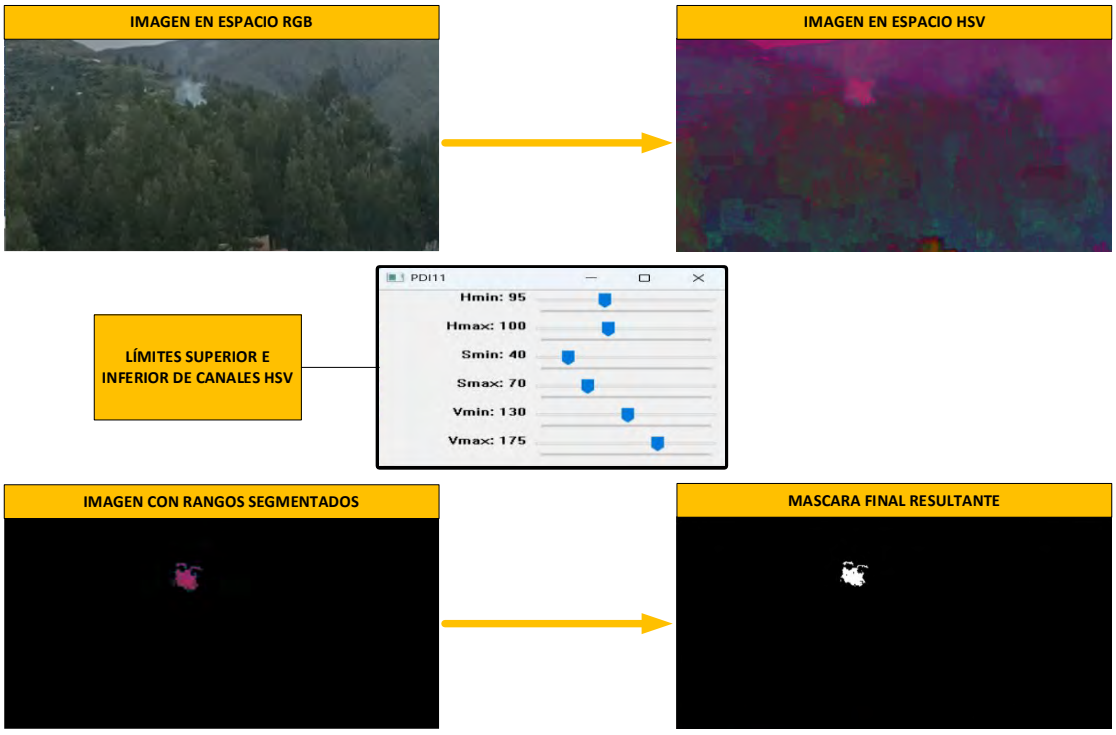
A continuación, se presenta el algoritmo para calcular el rango de color HSV del humo y fuego, los cálculos de los valores se realizaron deslizando un cursor en una barra gráfica comprendida entre valores 0 a 255 (8 bits). Este procedimiento facilitó el ajuste manual de los valores de HSV en tiempo real para segmentar la región de interés.

Al finalizar los ajustes en las imágenes de prueba, se determinó los valores mínimos y máximos preliminares del modelo de color HSV donde se encuentra el humo y fuego.

En la figura 40 se muestra el diagrama de funcionamiento del algoritmo.

Figura 40

Representación gráfica del algoritmo para calcular parámetros de color HSV

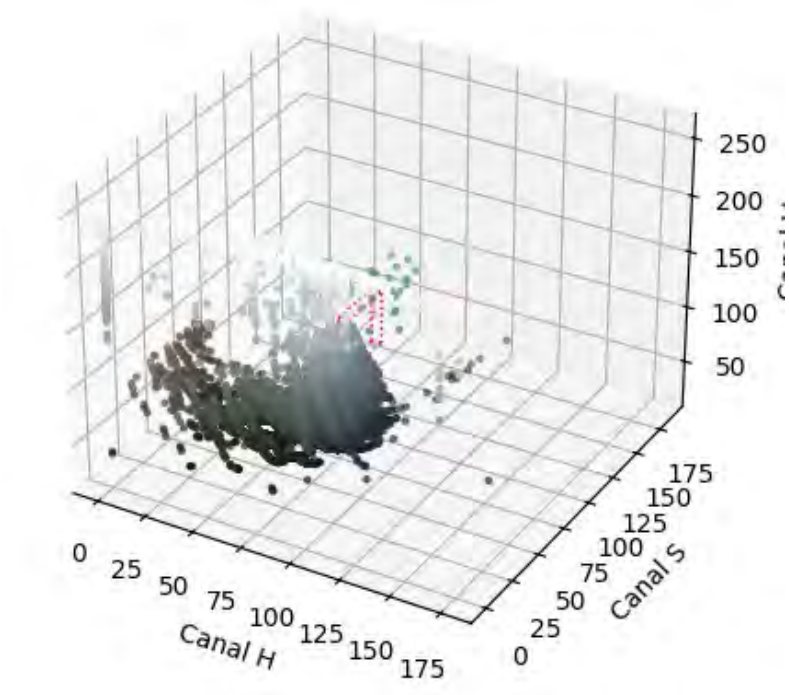


Nota: Cálculos manuales de rangos de color HSV par humo y fuego.

El gráfico 3D representa la distribución de los colores en el espacio HSV, donde el volumen representado por el polígono rojo interior aísla los pixeles cuyos valores H, S y V están dentro de los umbrales hallados en el paso anterior.

Figura 41

Imagen en el espacio HSV



Nota: el polígono rojo muestra los pixeles que están en el rango hallado manualmente en pasos anteriores.

Para el ejemplo mostrado en la figura 40, se utilizó la muestra “prueba7.jpg” y tras realizar los ajustes manuales arrojaron el siguiente rango de color HSV para el humo: HSV_{MAX} = [100,70,175] y HSV_{MIN} = [95,40,130]. Estos valores nos muestran el rango de valores HSV de los pixeles que representan humo.

Esta misma metodología se utilizó para hallar los rangos de valores HSV de todas las muestras tanto de humo y fuego. El código realizado para desarrollar esta herramienta de calibración se encuentra en Anexo 02. Para finalizar con los cálculos de rangos de color HSV de todas las muestras de humo y fuego, se utilizó un análisis estadístico para hallar los valores finales que fueron usados en la implementación del sistema. En las tablas 12 y 13 se muestran los resultados del análisis de las imágenes humo y fuego respectivamente.

Tabla 12

Parámetros HSV de humo de las imágenes de prueba

Parámetros HSV de humo		
Imagen de prueba	**HSV_{MIN}**	**HSV_{MAX}**
prueba1.jpg	[79,0,80]	[104,71,149]
prueba2.jpg	[86,82,185]	[110,110,255]
prueba3.jpg	[0,5,135]	[120,55,210]
prueba4.jpg	[105,50,155]	[109,85,190]
prueba5.jpg	[95,5,140]	[105,85,195]
prueba6.jpg	[25,0,105]	[110,55,160]
prueba7.jpg	[95,40,130]	[100,70,175]
prueba8.jpg	[0,0,175]	[180,15,230]
prueba9.jpg	[100,20,95]	[120,45,125]
prueba10.jpg	[55,5,140]	[100,35,235]
prueba11.jpg	[75,25,235]	[105,35,255]
prueba12.jpg	[20,0,205]	[115,45,240]

Tabla 13

Parámetros HSV de fuego de las imágenes de prueba

Parámetros HSV de fuego		
Imagen de prueba	HSV_{MIN}	HSV_{MAX}
pruebafuego1.png	[5,100,60]	[25,250,255]
pruebafuego2.jpg	[5,50,75]	[15,135,255]
pruebafuego3.jpg	[0,135,135]	[255,205,235]
pruebafuego4.jpg	[10,165,185]	[15,250,230]
pruebafuego5.jpg	[5,100,145]	[15,160,255]
pruebafuego6.jpg	[10,145,215]	[30,255,255]

Para hallar un único rango que represente a todas las muestras se calculó la media y desviación estándar del conjunto de datos.

Los rangos de color final vienen expresados por las ecuaciones:

$$HSV_{MIN} = [\mu(H_{MIN}) \pm \sigma(H_{MIN}),\ \mu(S_{MIN}) \pm \sigma(S_{MIN}), \mu(V_{MIN}) \pm \sigma(V_{MIN})]$$

$$HSV_{MAX} = [\mu(H_{MAX}) \pm \sigma(H_{MAX}), \mu(S_{MAX}) \pm \sigma(S_{MAX}), \mu(V_{MAX}) \pm \sigma(V_{MAX})]$$

La media viene representada por la ecuación:

$$\mu = \frac{1}{N}\sum x_i$$

La desviación estándar está representada por la ecuación:

$$\sigma = \sqrt{\frac{1}{N}\sum(x_i - \mu)^2}$$

Finalmente, los valores hallados para umbralización son los que se muestran a continuación:

Tabla 14

Rangos de parámetros finales HSV para humo y fuego

	HSV_{MIN}	**HSV_{MAX}**
Humo	$[61 \pm 38 \quad 19 \pm 25 \quad 148 \pm 44]$	$[115 \pm 21 \quad 59 \pm 25 \quad 201 \pm 41]$
Fuego	$[6 \pm 3 \quad 116 \pm 38 \quad 136 \pm 55]$	$[59 \pm 88 \quad 209 \pm 47 \quad 248 \pm 11]$

Estos valores fueron utilizados al momento de realizar la implementación del sistema en el algoritmo de detección mediante espacios de color, donde los valores medios serán los iniciales y la varianza sirvió para calibrar los parámetros de color del sistema de detección de incendios forestales.

Capítulo IV: Implementación del Sistema

4.1.Introducción

En la presente sección se detalla el proceso de implementación del prototipo del sistema, siguiendo los pasos que previamente fueron planteados en el Capítulo III (Diseño). El capítulo IV está conformado por dos partes principales: la primera se centra en el desarrollo e instalación de hardware y software del prototipo del sistema y la segunda parte está conformada por la integración.

4.2. Proceso de Implementación del Hardware

Como se puede apreciar en el diagrama de conexión y disposición física en el Capítulo III (Diseño del Sistema). Existe ciertos componentes que serán necesarios para la implementación del prototipo del sistema detector de incendios forestales en esta sección se describe cada uno de ellos y la configuración inicial antes de hacer la implementación.

4.2.1. Ensamblado de Componentes

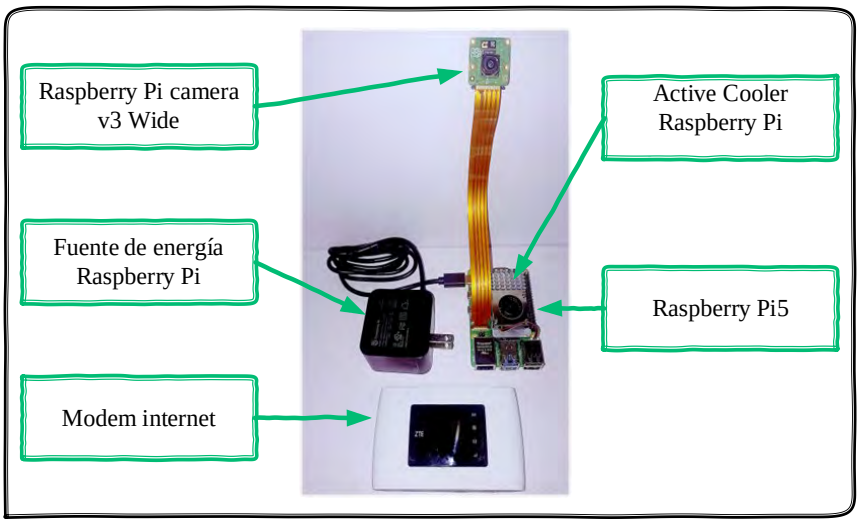
El primer paso fue la adquisición y ensamblaje de los componentes físicos, cuyas especificaciones técnicas fueron justificadas en el Capítulo III:

- **Unidad de procesamiento**: Raspberry Pi 5 (8 GB RAM)
- **Captura de video:** Cámara Raspberry Pi v3 (Wide)
- **Alimentación:** Fuente de poder para Raspberry Pi 5
- **Refrigeración**: Active Cooler de Raspberry Pi
- **Conectividad:** Modem internet

La figura 42 muestra los dispositivos seleccionados para la implementación física del prototipo.

Figura 42

Componentes del sistema detector de incendios forestales



4.2.2. Configuración del Sistema Embebido

Una vez adquiridos los componentes de hardware, se procedió con la configuración del sistema embebido Raspberry Pi 5. Es importante destacar que todo este proceso se ejecutó en una computadora portátil externa, que sirvió para descargar y grabar el sistema operativo al Raspberry Pi 5, también sirvió para acceder de manera remota al sistema embebido y de esta forma realizar las configuraciones iniciales.

a) **Instalación del sistema operativo (SO):** Se utilizó el programa Raspberry Pi Imager para descargar y grabar la imagen Raspberry Pi OS (64-bit) en una tarjeta microSD (se recomienda de una capacidad superior a 32 GB).

b) **Personalización y acceso remoto:** En el proceso de grabación del sistema operativo, se aprovechó la función de personalización de la imagen para:

- Definir nombre de usuario y contraseña.

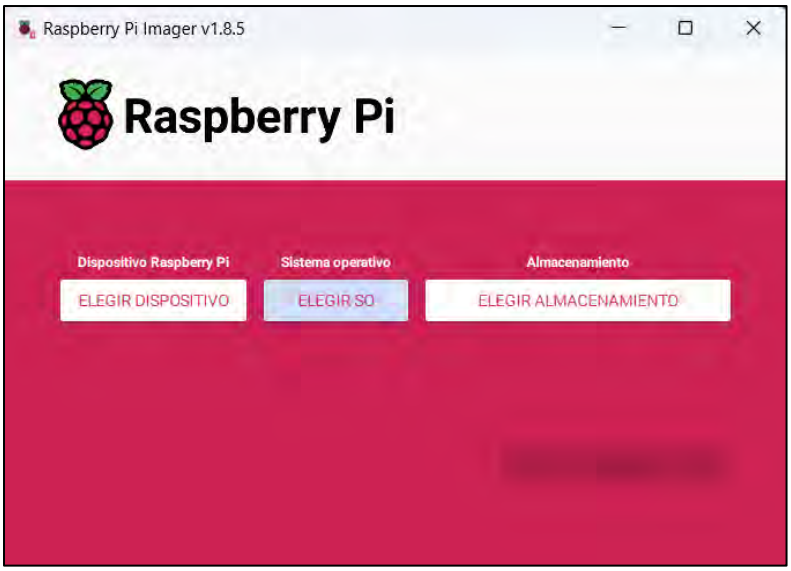
- Configuración de red LAN inalámbrica (Wi-Fi).

- Habilitar servicio de conexión remota SSH (Secure Shell), de esta forma eliminando la necesidad de utilizar periféricos adicionales como monitor, teclado y mouse al sistema embebido.

La Figura 43 muestra la interfaz del software Raspberry Pi Imager en la etapa de personalización.

Figura 43

Software Raspberry Pi Imager



c) **Verificación de conexión remota:** Después de instalar la tarjeta microSD al dispositivo Raspberry Pi 5, se verificó su funcionalidad correcta mediante conexión remota siguiendo los siguientes pasos:

- Determinación de la dirección IP: Se pudo identificar la dirección IP del sistema embebido utilizando herramientas de escaneo de red o mediante el comando arp en el CMD de Windows. (figuras 44 y 45).

Figura 44

Software Advanced IP Scanner

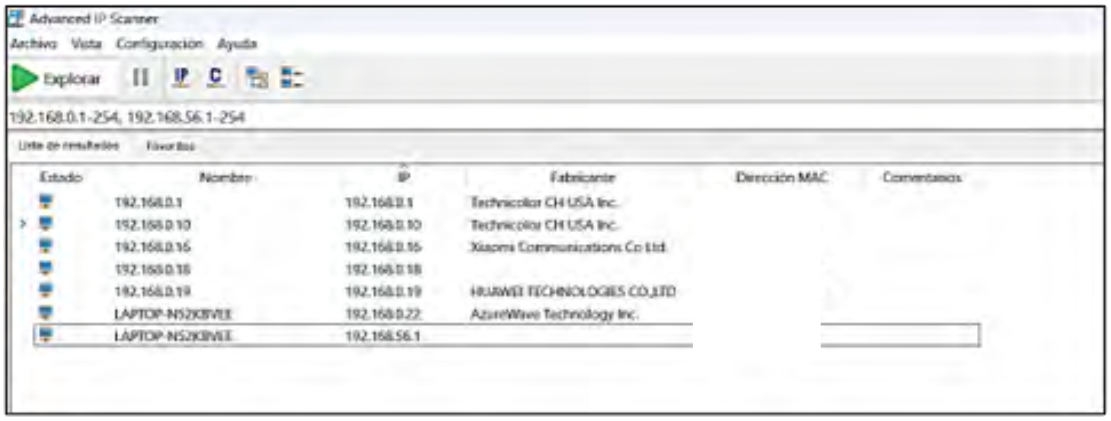
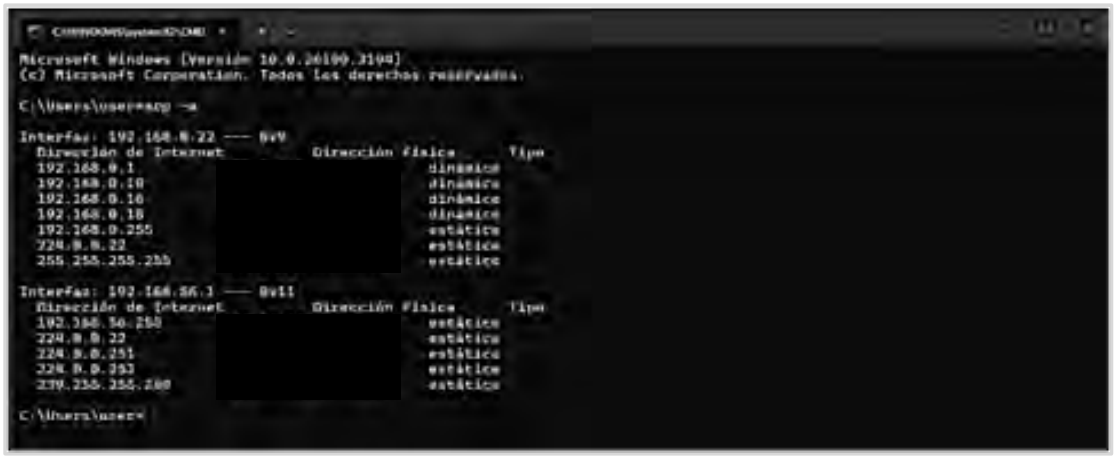


Figura 45

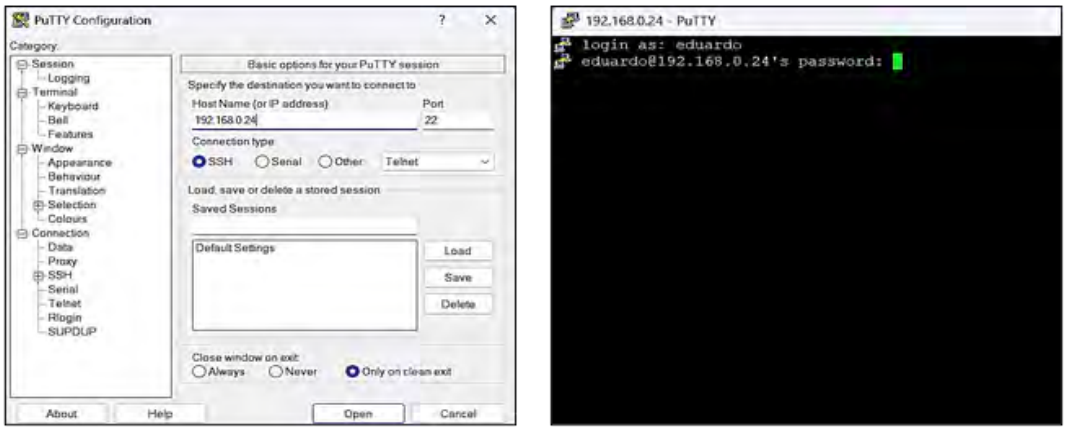
Comando ARP en Windows



d) **Acceso SSH:** Para realizar la conexión remota se utilizó el emulador de terminal PuTTY. Para la conexión correcta se requirió la dirección IP, el nombre de usuario y la contraseña configurados en la etapa de personalización del Sistema Operativo (figura 46).

Figura 46

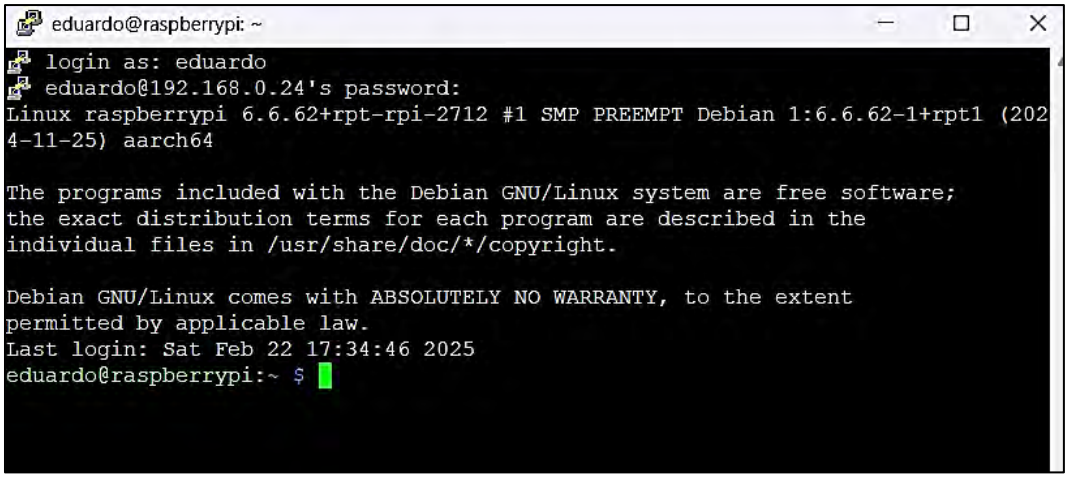
Conexión a Raspberry Pi mediante Software PuTTY



Si la autenticación fue correcta, se podrá acceder a la línea de comandos Linux (figura 47), asegurando que el sistema embebido está operativo y listo para su uso.

Figura 47

Conexión a interfaz de líneas de comandos Raspberry Pi



4.2.3. Configuración de la Cámara Raspberry Pi V3 (Wide)

Una vez configurado el sistema embebido, se procedió a configurar y verificar la conexión entre el Raspberry Pi 5 y la cámara Raspberry Pi V3 (Wide). Para ello se realizó el siguiente procedimiento:

a) **Conexión y actualización:** Verificar la conexión física del módulo de la cámara al puerto CSI del Raspberry Pi 5, después ingresar al terminal del Raspberry y verificar si está instalado el framework libcamera (controlador para cámaras oficiales de Raspberry):

   - $ *sudo apt update && sudo apt install -y libcamera-apps*

b) **Verificación de operatividad**: Se verificó la operatividad de la cámara mediante el comando:

   - $ *libcamera-hello*

   El emulador de código PuTTy no tiene un visualizador de gráficos por ello no se muestra la imagen, pero si se puede comprobar la funcionalidad si se muestra el número de frames, los FPS, la exposición y la ganancia digital de la imagen. con este paso, se confirma la funcionalidad de los dispositivos esenciales.

4.3.Proceso de Implementación del Software

El sistema de detección de incendios forestales tiene como núcleo de funcionamiento el desarrollo de software, que integra los algoritmos para la detección de estos eventos y la generación de alertas de manera eficiente. Esta sección detalla la configuración del entorno de programación, el desarrollo del código y para finalizar con la explicación de los algoritmos desarrollados.

4.3.1. Configuración del Entorno de Desarrollo

Esta sección explica la preparación del entorno de programación en el sistema embebido Raspberry Pi 5, incluyendo la verificación e instalación de software de programación, librerías y demás dependencias.

4.3.1.1.Preparación del Entorno Gráfico (VNC)

Aunque la herramienta PuTTy fue útil para la configuración inicial. Su limitación a un entorno basado en texto imposibilitaba la visualización de gráficos y ventanas. Por ello, se requirió utilizar la herramienta Real VNC Server para acceder de manera remota al entorno gráfico de la Raspberry PI OS.

El acceso se realizó en una laptop externa, que actúa únicamente como una interfaz de programación y configuración, sin participar del procesamiento del sistema. A continuación, se explica a mejor detalle el proceso de acceso a la interfaz gráfica del sistema embebido Raspberry Pi 5.

a) **Acceso a VNC**: Como primer paso se requirió instalar el software Real VNC Server en la computadora portátil. Mientras en el Raspberry Pi 5 ya dispone de este software de forma nativa.

b) **Conexión:** Una vez Instalado el programa Real VNC Server se procedió a realizar una nueva conexión y para ello se necesitó conocer la dirección IP del Raspberry Pi 5, el cual se obtuvo en la configuración inicial y personalización del sistema embebido.

   Tras introducir la contraseña, se obtuvo acceso al entorno gráfico completo (figura 48), confirmando el acceso a todos los recursos del sistema operativo.

Figura 48

Interfaz Gráfica de Raspberry Pi

4.3.1.2.Instalación de Herramientas y Dependencias

En la sección de diseño se estableció las herramientas y dependencias necesarias para realizar el código de programación.

a) **Verificación de Python e IDE**: Se verificó que el lenguaje de programación Python v.3 este preinstalado introduciendo el siguiente comando:

- **$** *python3 –versión*

- Respuesta: *Python 3.11.2*

b) **Instalación de entorno de desarrollo (IDE):** Se verificó que el entorno de escritura, depuración y ejecución de código Thony este preinstalado introduciendo el siguiente comando en la terminal:

- **$** *thonny –versión*

- Respuesta: *12:14:30.201 [MainThread] INFO thonny: Thonny version: 4.1.4*

c) **Instalación de librerías:** Se procedió a instalar y actualizar las librerías especializadas y necesarias, para ello se introduce los siguientes comandos en la terminal:

- **$** *sudo apt install libopencv-dev python3-opencv -y*
- **$** *pip3 install numpy*
- **$** *pip3 install requests*
- **$** *pip3 install picamera2*

d) **Verificación de versiones:** Una vez instalada todas las librerías se procedió a verificar sus versiones para garantizar la compatibilidad:

- **$** *python3 -c "import cv2; print(cv2.__version__)"*
- *Respuesta: 4.6.0*
- **$** *pip3 freeze | grep -E "numpy|requests|picamera2"*
- *Respuesta: numpy==1.24.2*
- *Respuesta: picamera2==0.3.24*

En esta parte se completó la fase de implementación de hardware y preparación para la codificación.

4.3.2. Desarrollo del Código

Esta sección tiene como objetivos desarrollar la lógica del prototipo del sistema detector de incendios forestales utilizando lenguaje de programación Python, implementando los 9 algoritmos planteados en el punto 3.6.2. del Capítulo III diseño del sistema.

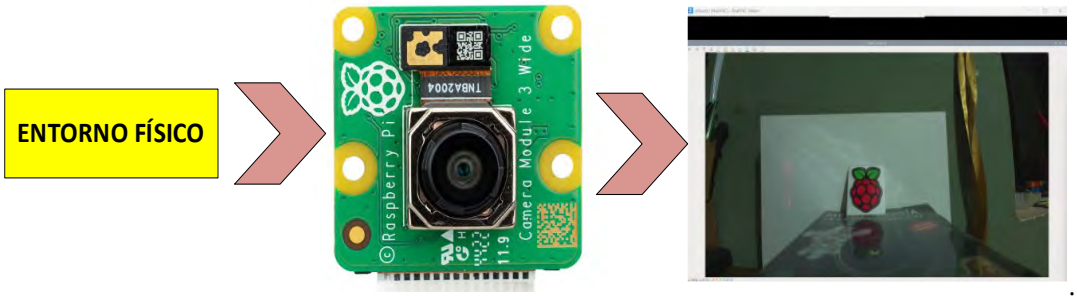
4.3.2.1. Adquisición del Video

La adquisición del video en tiempo real se realiza a través de la cámara Raspberry Pi V3 (Wide), para su integración se requiere utilizar la librería Picamera2, la cual proporciona una interfaz flexible para la configuración y control de la cámara.

El resultado del algoritmo se muestra en la figura 49 y el procedimiento para adquirir el video en tiempo real se muestra en el Pseudocódigo 01.

Figura 49

Representación gráfica de algoritmo adquisición de video mediante módulo de cámara Raspberry Pi V·3



Pseudocódigo 01

Adquisición de video con Python

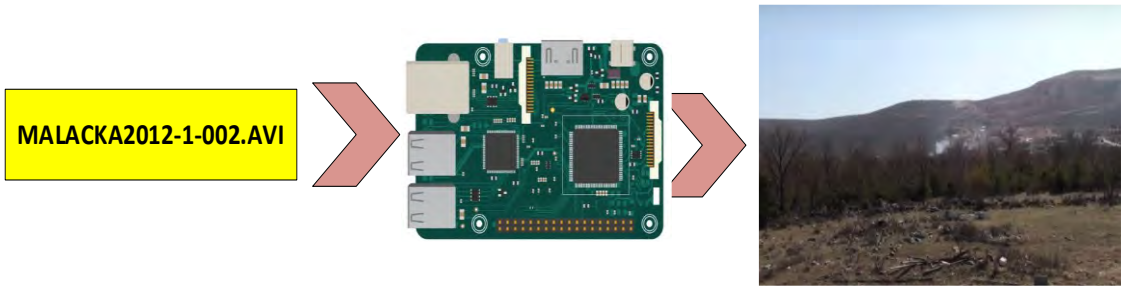
Pseudocódigo 01: Adquisición de video	
01	**INICIAR** objeto Picamera2
02	**CONFIGURAR** objeto Picamera2 (Resolución, FPS, configuraciones predeterminadas)
03	**ACTIVAR** cámara (Picam2.start()).
04	**SI** cámara no detectada **ENTONCES**
05	**MOSTRAR** error **SALIR**
06	**BUCLE PRINCIPAL MIENTRAS** Picamera2 Activo:
07	**CAPTURAR** fotograma (frame)
08	**CONTINUAR** pseudocódigo *03 (pre - procesamiento de video)*
09	**FIN BUCLE PRINCIPAL** (presionar tecla “q”)
10	**CERRAR** liberar cámara

Tal como se explicó en el capítulo III diseño del sistema en la sección de justificación de decisiones del diseño. Para realizar las pruebas iniciales se utilizó videos pregrabados, en este caso para la adquisición del video se utilizó otra metodología, se modificó el código donde se realiza la

captura de video en tiempo real por lectura de archivos de video. Sin cambiar la lógica de detección de humo y fuego. El resultado del algoritmo se muestra en la figura 50 y el procedimiento para capturar archivos de video se muestra a continuación en Pseudocódigo 02:

Figura 50

Representación gráfica del algoritmo adquisición de video pregrabado



Pseudocódigo 02

Adquisición de video pregrabado con Python

Pseudocódigo 02: Adquisición de video pre - grabado	
01	**ABRIR** ruta de video (cv2.VideoCapture)
02	**SI** ruta de video no abre **ENTONCES**
03	**MOSTRAR** error y **SALIR**
04	**BUCLE PRINCIPAL MIENTRAS** ruta de video abierto:
05	**LEER** fotograma (frame)
06	**SI** no existe fotogramas **ENTONCES** salir bucle
07	**CONTINUAR** pseudocódigo *03 (pre - procesamiento de video)*
08	**FIN BUCLE** (presionar tecla "q")
09	**CERRAR** lectura de video y ventanas activas.

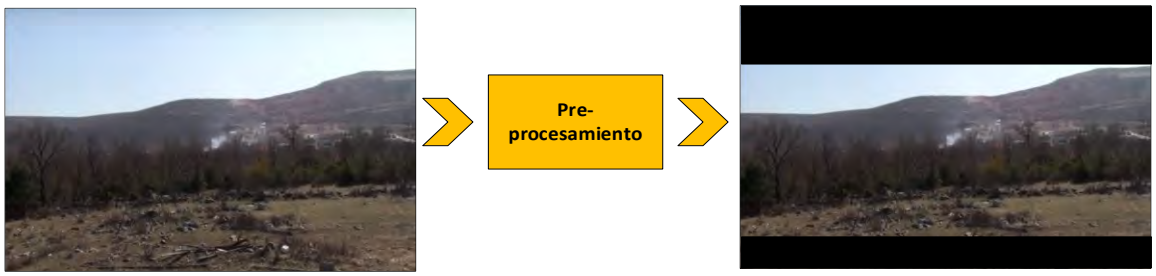
4.3.2.2. Pre - Procesamiento de Video

El pre – procesamiento del video se encarga de redimensionar cada frame de entrada a 960 x 540 y de segmentar el ROI (Región of Interesting) de esta forma eliminado zonas donde no se originan incendios forestales. En la etapa de diseño se justifica el uso de dicha resolución. Cabe

resaltar que la librería Picamera2 tiene opciones para cambio de resolución, pero estos están definidos para cámaras oficiales de Raspberry Pi. En consecuencia, esta línea de código se encarga de redimensionar la imagen si en algún momento se decide utilizar otro tipo de cámara con una mayor resolución. El resultado del algoritmo se muestra en la figura 51 y el procedimiento para realizar el algoritmo se muestra a continuación en el Pseudocódigo 03:

Figura 51

Representación gráfica del algoritmo de pre – procesamiento de video



Pseudocódigo 03

Algoritmo pre - procesamiento de video

Pseudocódigo 03: Pre – procesamiento de video	
01	**DEFINIR** puntos de extracción (ROI) *configuración inicial única*
02	----------
03	**RECIBIR** fotograma (pseudocódigo 01 o 02)
04	**REDIMENSIONAR** fotograma (960, 540)
05	**EXTRAER** porción definida ⟶ ROI
06	**ENVIAR** ROI al pseudocódigo 04 *(detector de movimiento)*

4.3.2.3. Detección de Movimiento

OpenCV dispone de algoritmos para detectar movimiento en secuencia de imágenes o video. Para el proyecto de ingeniería se utilizó el método de detección de movimiento cv2.createBackgroundSubstractorKNN(). En la sección de diseño se presentó el concepto de los

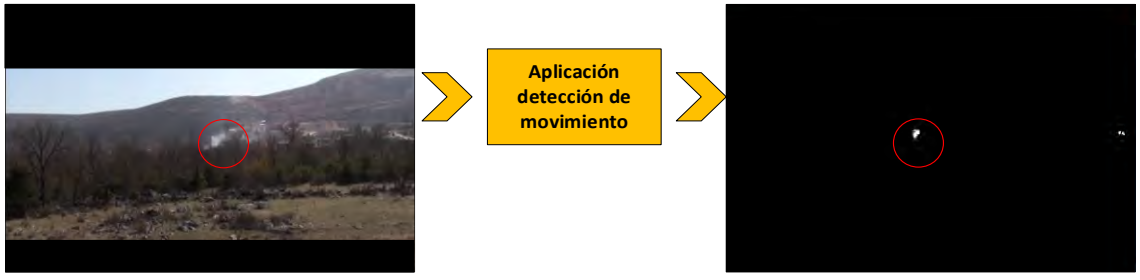
parámetros del algoritmo de detección de movimiento y el cálculo de valores óptimos. Para este caso se utilizó los siguientes valores:

- history = 900 tal como se halló los parámetros óptimos en la etapa de diseño se decidió utilizar un history alto cercano a 1000 ya que con estos valores se obtuvo el mejor desempeño del algoritmo. Sin embargo, para mejorar ante cambios rápidos de iluminación, se decidió reducir hasta 900 ya que con este valor se pudo observar una mejor respuesta.
- dist2Threshold = 700 se optó por utilizar este valor ya que con este se redujo la cantidad de pixeles que no representaban humo y los movimientos rápidos de vegetación si bien para 200 se encontró el mejor desempeño para detectar la mayor cantidad de pixeles que representaba humo, esto ocasionaba la detección excesiva de movimientos irrelevantes. Por ello, a costa de reducir una porción de pixeles de humo para eliminar estos falsos positivos que representaban un ruido visual en la máscara resultante y teniendo en cuenta que el humo tiene un comportamiento expansivo y en la máscara generada lograba ocupar porciones grandes de pixeles.

La representación gráfica del algoritmo en la figura 52 y el procedimiento para realizar la detección de movimiento se muestra a continuación en Pseudocódigo 04:

Figura 52

Representación gráfica del algoritmo detector de movimiento



Pseudocódigo 04

Algoritmo detección de movimiento con KNN subtraction

Pseudocódigo 04: Detección de movimiento	
01	**DEFINIR** parámetros de sustracción de fondo KNN (history = 900, Dis2threshold = 700)
02	----------
03	**RECIBIR** ROI (pseudocódigo 03)
04	**APLICAR** sustracción de fondo KNN a ROI ⟶ *MÁSCARA MOVIMIENTO*
05	**ENVIAR** *MÁSCARA MOVIMIENTO* al pseudocódigo 05 *(post - detección de movimiento)*

4.3.2.4. Post – Detección de Movimiento

Esta sección de código se encarga de reducir el ruido de fondo que logró pasar en la primera máscara de movimiento y que no representaba humo o fuego, causado por distintas razones como lo son: movimientos rápidos en árboles, polvo, aparición de aves en movimiento, cambios de iluminación hasta los ruidos producidos en la captura de imagen por parte de la cámara. Por los motivos mencionados se decidió aplicar filtros de suavizado y operaciones morfológicas como erosión y dilatación.

El filtrado de suavizado (filtro de mediana) permitió reducir o eliminar en la mayor parte el ruido impulsivo o más conocido como tipo sal y pimienta.

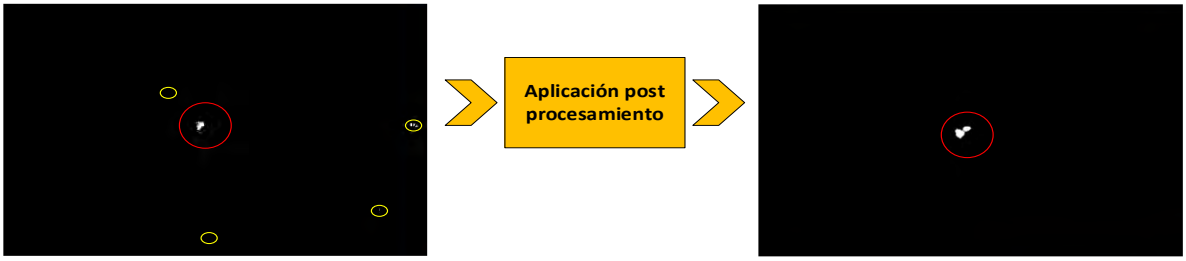
Posteriormente las operaciones morfológicas de erosión y dilatación sirvieron para eliminar los pixeles dispersos que persistían tras la aplicación de filtros iniciales. Se aplicó la operación de erosión para eliminar estos pixeles pequeños y aislados. Sin embargo, esto provocó perder también información de pixeles agrupados de humo o fuego.

Para contrarrestar esta pérdida de información se utilizó la operación de dilatación en dos iteraciones para recuperar los pixeles perdidos a causa de los filtros y operaciones aplicados anteriormente.

El procedimiento para realizar el filtrado post- detección de movimiento se muestra a continuación en Pseudocódigo 05 y la representación gráfica del algoritmo en la figura 53:

Figura 53

Representación gráfica el algoritmo post – detección de movimiento



Pseudocódigo 05

Algoritmo post – detección de movimiento

Pseudocódigo 05: Post - detección de movimiento	
01	**RECIBIR** *MÁSCARA MOVIMIENTO* (pseudocódigo 04)
02	**APLICAR** filtro de mediana (Ventana 3x3) a *MÁSCARA MOVIMIENTO*
03	**DEFINIR** elemento kernel 3x3 para operaciones morfológicos
04	**APLICAR** filtro morfológico erosión a *MÁSCARA MOVIMIENTO*
05	**APLICAR** filtro morfológico dilatación a *MÁSCARA MOVIMIENTO en 2 iteraciones*
06	**SI** en *MÁSCARA MOVIMIENTO mayor 2000 px. (movimiento relevante)* ***ENTONCES***
07	**ALMACENAR** *MÁSCARA MOVIMIENTO*
08	**CONTINUAR** pseudocódigo 06 *(detección mediante espacio de color)*
09	**CASO CONTRARIO** *(no existe movimiento relevante)*
10	**SALTAR** al **BUCLE PRINCIPAL** (leer siguiente fotograma)

4.3.2.5. Detección de Humo y/o Fuego Mediante Espacio de Color

Después de haber detectado movimiento y haber creado la máscara de las regiones donde el movimiento es evidente, es necesario utilizar otro algoritmo ampliamente utilizado para realizar detección de humo y fuego. En este caso fue mediante espacios de color. Para el proyecto se usó

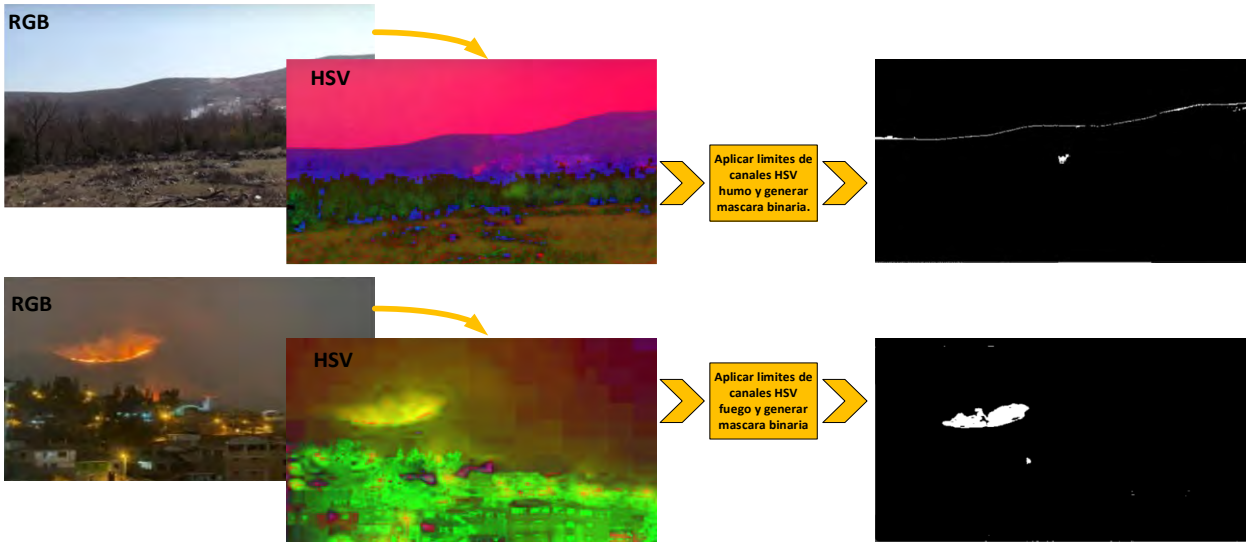
el espacio de color HSV y los rangos a utilizados fueron calculados en la sección 3.7.3. del capítulo III diseño el sistema.

El algoritmo propuesto lleva a cabo una comparación del valor de color de los pixeles del fotograma actual con el rango cromático que tiene el humo y el fuego hallados en secciones previas, mediante el análisis de color en el espacio HSV. Esta comparativa facilita la identificación todos los pixeles del fotograma actual que se encuentran en estos rangos. Por lo tanto, el algoritmo genera una máscara binaria en la que los pixeles que cumplieron el requisito se muestran en primer plano vinculado a la presencia de humo o fuego.

El procedimiento para realizar la detección de humo y/o fuego mediante espacio de color se muestra a continuación en Pseudocódigo 06 y la representación gráfica del algoritmo en figura 54:

Figura 54

Representación gráfica de algoritmo detector de humo y fuego mediante espacio de color HSV



Pseudocódigo 6

Algoritmo detector de humo y fuego mediante espacio de color HSV

Pseudocódigo 06: Detección mediante espacio de color	
01	**DEFINIR** parámetros limites MAX. y MIN. de humo y fuego en espacio de color HSV
02	----------
03	**RECIBIR** ROI (pseudocódigo 03)
04	**CONVERTIR** ROI (RGB) a espacio de color HSV.
05	**APLICAR** umbral de rangos MAX. y MIN. de humo ⟶ *MÁSCARA HUMO*
06	**APLICAR** umbral de rangos MAX. y MIN. de fuego ⟶ *MÁSCARA FUEGO*
07	**CONTINUAR** pseudocódigo 07 *(post procesamiento espacio de color)*

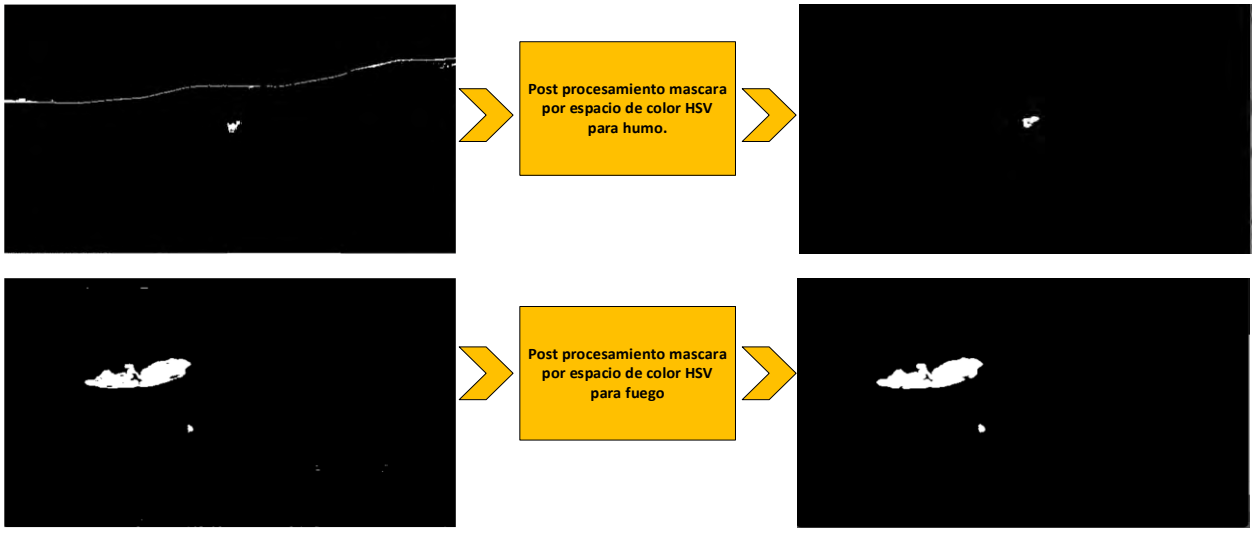
4.3.2.6. Post - Procesamiento Espacio de Color

Esta sección de código se encarga de reducir el ruido de fondo causado por objetos pequeños en escena con la misma tonalidad de humo o fuego, el cual se presenta en la máscara de espacio de color como ruido. Es importante mencionar que el humo y fuego tienden a extenderse y al realizar este filtrado no perjudica la detección. Por el motivo mencionado y de la misma forma que se realizó para la máscara de movimiento, se aplicó filtros de suavizado y filtros morfológicos para reducir este ruido.

El procedimiento para realizar la detección de movimiento se muestra a continuación en Pseudocódigo 07 y la representación gráfica del algoritmo en la figura 55:

Figura 55

Representación gráfica de algoritmo post procesamiento espacio de color



Pseudocódigo 7

Algoritmo post procesamiento espacio de color

Pseudocódigo 07: post – procesamiento espacio de color	
01	**RECIBIR** *MÁSCARA HUMO* (pseudocódigo 06)
02	**RECIBIR** *MÁSCARA FUEGO* (pseudocódigo 06)
03	**APLICAR** filtro de mediana (Ventana 3x3) a *MÁSCARA HUMO Y MÁSCARA FUEGO*
04	**DEFINIR** kernel 3x3 para operaciones de filtros morfológicos
05	**APLICAR** filtro morfológico erosión a *MÁSCARA HUMO Y MÁSCARA FUEGO*
06	**APLICAR** filtro morfológico dilatación a *MÁSCARA HUMO Y MÁSCARA FUEGO* *2 iteraciones.*
07	**ALMACENAR** *MÁSCARA HUMO Y MÁSCARA FUEGO*
08	**CONTINUAR** pseudocódigo 08 *(confirmación y seguimiento de eventos críticos)*

4.3.2.7. Confirmación y Seguimiento de Eventos Críticos

Esta es la última parte de la lógica para validar si existe presencia de humo o fuego en la escena. En primera instancia se utilizó las máscaras de movimiento y la de espacio de color,

después se realizó una intersección de ambas máscaras (bitwise_AND), de esta forma hallando todos los conjuntos de pixeles que cumplan ambas condiciones y que confirme el verdadero positivo.

Como segundo paso, se realizó un algoritmo que confirme si la actividad perdura en el tiempo. Para ello, se realiza un análisis a las máscaras binarias resultantes de la intersección de las máscaras de movimiento y color. Si dichas máscaras se mantienen activas durante un número determinado de frames de análisis, definido como el umbral de análisis temporal, entonces se considera que existe regiones persistentes.

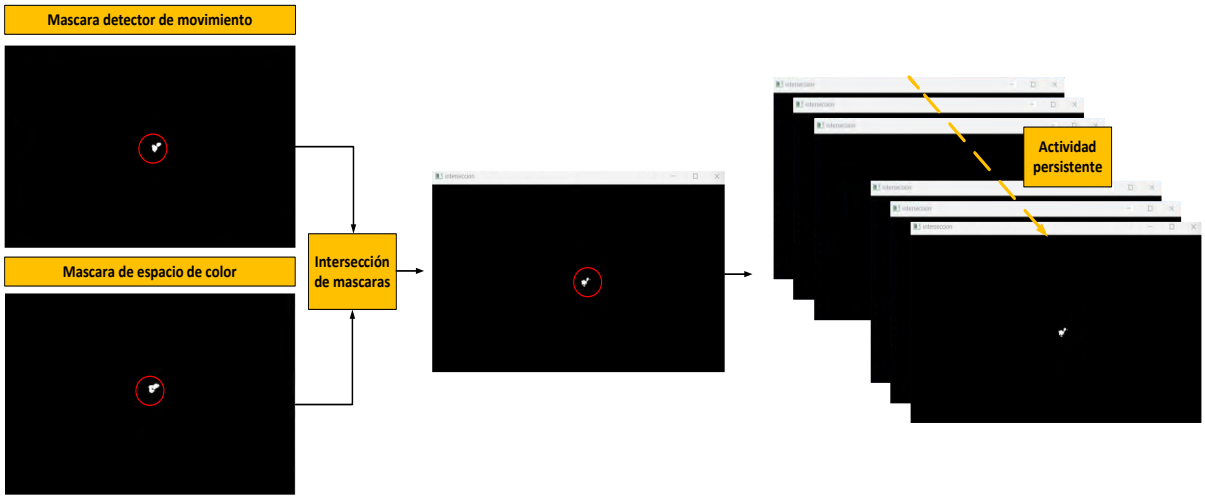
Luego se utilizó una metodología donde se genera una última máscara donde se muestran en primer plano todos los pixeles que estuvieron presentes en un 70% de la cantidad de frames de análisis, eliminando los pixeles que aparecieron de forma temprana, intermedia o tardía en pocos frames de análisis y que en la mayor parte representan elementos con un movimiento rápido y efímero. Para aplicar esta metodología, previamente se almacenó las máscaras binarias consecutivas de intersección, donde se analiza la frecuencia de activación por pixel en todas las máscaras y de esta forma generando una máscara representativa.

Para culminar se utilizó un análisis de regiones conexas el cual filtra regiones de pixeles que no cumplan con un umbral de tamaño establecido, de esta forma eliminando pequeños puntos en escena que perduran en el tiempo, ya que el humo tiende a abarcar grandes áreas en la imagen.

El procedimiento para realizar la confirmación y seguimiento de eventos críticos se muestra en Pseudocódigo 08 y la representación gráfica del algoritmo en la figura 56:

Figura 56

Representación gráfica del algoritmo confirmación y seguimiento de eventos críticos



Pseudocódigo 08

Algoritmo de confirmación y seguimiento de eventos críticos

Pseudocódigo 08: confirmación y seguimiento de eventos críticos

```
01  RECIBIR MÁSCARA HUMO, MÁSCARA FUEGO y MÁSCARA MOVIMIENTO
02  PROCESAR MÁSCARA HUMO y MÁSCARA MOVIMIENTO ⟶INTERSECCIÓN HUMO
03  PROCESAR MÁSCARA FUEGO y MÁSCARA MOVIMIENTO ⟶INTERSECCIÓN FUEGO
04  SI INTERSECCIÓN HUMO > umbral mínimo (500 px) ENTONCES
05      INCREMENTAR contador de intersección humo.
06      ALMACENAR INTERSECCIÓN HUMO en una lista.
07  CASO CONTRARIO reiniciar contador y lista de almacén humo
08  SI INTERSECCIÓN FUEGO > umbral mínimo (500 px) ENTONCES
09      INCREMENTAR contador de intersección fuego.
10      ALMACENAR INTERSECCIÓN FUEGO en una lista.
11  CASO CONTRARIO reiniciar contador y lista de almacén fuego
12  SI contador intersección humo >= f_análisis O contador intersección fuego >= f_análisis
13    SI contador intersección humo > f_análisis Y len(lista INTERSECCIÓN HUMO)> f_análisis
14        SUMAR pixeles activos en posición (x,y) de INTERSECCIÓN HUMO total acumuladas
15        DEFINIR umbral presencia final ⟶ 70% (f_análisis)
16        GENERAR MÁSCARA FINAL HUMO con pixeles > umbral presencia final
```

17 **ALMACENAR** *MÁSCARA FINAL HUMO*

18 **SI** contador intersección fuego>f_análisis **Y** len(lista *INTERSECCIÓN FUEGO)*>f_análisis

19 **SUMAR** pixeles activos en posición (x,y) de *INTERSECCIÓN FUEGO total acumuladas*

20 **DEFINIR** umbral presencia final ⟶ 70% (f_análisis)

21 **GENERAR** *MÁSCARA FINAL FUEGO* con pixeles > umbral presencia final

22 **ALMACENAR** *MÁSCARA FINAL FUEGO*

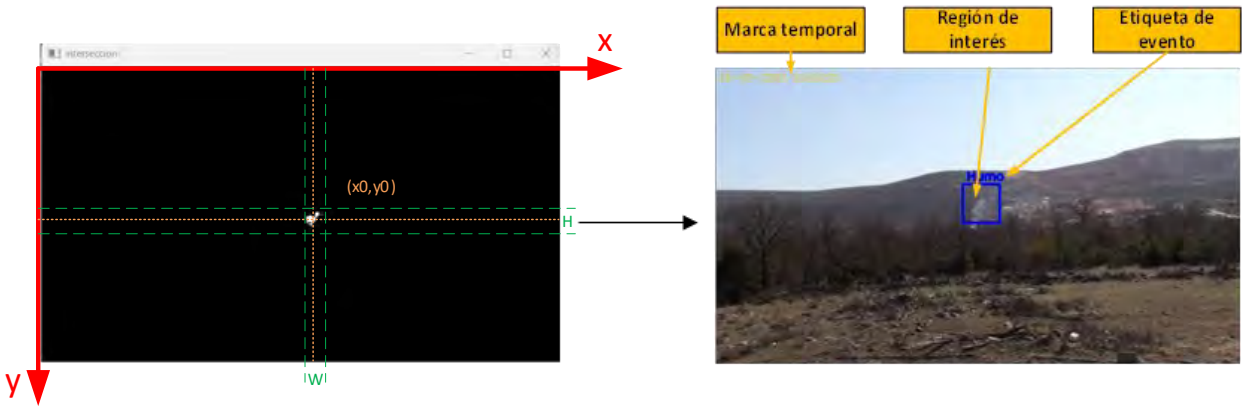
23 **CONTINUAR** pseudocódigo 09 *(segmentación área de interés)*

4.3.2.8. Segmentación de Área de Interés

En este punto el sistema realizó la detección humo o fuego. Ahora es de suma importancia sobreponer los resultados en la imagen original para demostrar de forma gráfica la región donde se identificó la actividad relevante. En esta sección de código se realiza las actividades de superposición de información como inserción de rectángulos (bounding boxes) alrededor de áreas detectadas, inserción de etiquetas y la incorporación de marca temporal (timestamp). El procedimiento para realizar la segmentación de área de interés se muestra a continuación en Pseudocódigo 09 y la representación gráfica del algoritmo en la figura 57:

Figura 57

Representación gráfica de segmentación de área de interés



Pseudocódigo 9

Algoritmo de segmentación de área de interés

Pseudocódigo 09: segmentación de área de interés

```
01    RECIBIR MÁSCARA FINAL HUMO y MÁSCARA FINAL FUEGO
02    BUCLE para MÁSCARA FINAL HUMO y MÁSCARA FINAL FUEGO HACER
03      SI MÁSCARA FINAL HUMO O MÁSCARA FINAL FUEGO no está vacía ENTONCES
04        ENCONTRAR contornos en MÁSCARA FINAL HUMO y MÁSCARA FINAL FUEGO
05        BUCLE para contornos encontrados HACER
06            SI áreas de contornos > 10 (eliminar regiones conexas pequeñas) ENTONCES
07              DIBUJAR Rectángulo delimitador (Bounding Box) en ROI
08              AGREGAR etiqueta "HUMO" o "FUEGO" en ROI
09              AGREGAR marca temporal (timestamp) ROI
10              GUARDAR imagen final etiquetada con la detección en ruta_imagen.jpg.
11              CONTINUAR pseudocódigo 10 (generación y envío de alerta)
12        FIN BUCLE
13      FIN SI
14    FIN BUCLE
```

4.3.2.9. Generación y Envío de Alertas

Es fundamental para el sistema de detección de incendios forestales, alertar en tiempo real los eventos detectados, para ello se utilizó servicios OTT (Over The Top), que permiten una comunicación eficaz, automática y de bajo costo mediante uso de APIs (Interfaz de Programación de Aplicaciones).

Para el proyecto de ingeniería se utilizó la API de Telegram, aprovechando que existe documentación oficial de APIs de Telegram y Python, siguiendo el procedimiento para la configuración del bot:

a) **Creación del Bot:**

- Buscar el bot oficial @BotFather en Telegram.

- Iniciar un chat con el comando /newbot, donde se establecera un nombre (ej. Tesis) y nombre de usuario (ej. IncendioCusco_bot).

- Telegram proporcionará un Bot_Token que será la llave de autenticación de la API.

b) Obtención Chat ID:

- Acceder a la dirección web en el navegador:

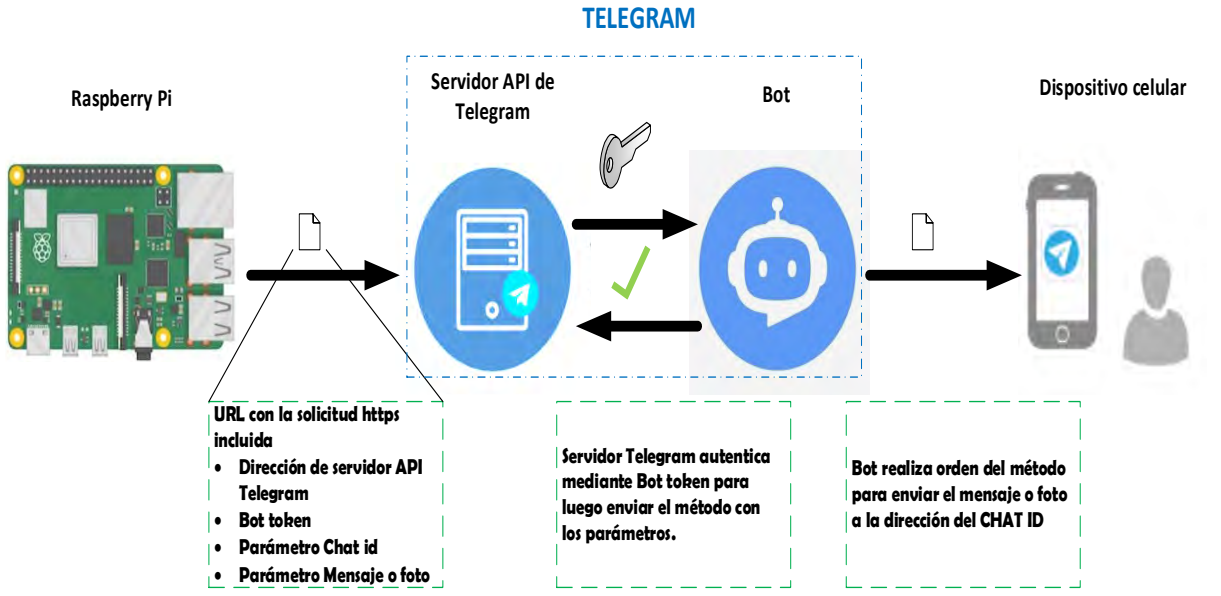
  https://api.telegram.org/bot<Bot_Token>/getUpdates

- Servidor de Telegram envía una respuesta JSON (JavaScript Object) donde se podrá visualizar el Chat ID.

Una vez obtenidos el Bot_Token y el chat ID, se procedió con la implementación del algoritmo para enviar notificaciones mediante Python cumpliendo una de las funcionalidades del prototipo del sistema. El procedimiento para realizar envío de notificaciones se muestra a continuación en Pseudocódigo 10 y la representación gráfica del algoritmo en la figura 58:

Figura 58

Representación gráfica de algoritmo generación y envío de notificaciones mediante Telegram



Pseudocódigo 10

Algoritmo de generación y envío de alertas

Pseudocódigo 10: generación y envío de alerta

```
01  FUNCION enviar mensaje ()
02      CONSTRUIR URL de la API de Telegram con Bot_Token y método SendMessage
03      CONFIGURAR parámetros (chat ID y texto)
04      ENVIAR solicitud HTTPS (GET) al servidor de Telegram
05      SI falla conexión ENTONCES mostrar error
06  FIN FUNCION
07  ----------------------------------------------------------------------------------------------------
08  FUNCION enviar imagen ()
09      SI no existe ruta de imagen ENTONCES mostrar error y salir
10      CONSTRUIR URL de la API de Telegram con Bot_Token y método SendPhoto
11      CONFIGURAR parámetros (chat ID e imagen)
12      ENVIAR solicitud HTTPS (POST) al servidor de Telegram
13      SI falla conexión ENTONCES mostrar error
14  FIN FUNCION
15  ----------------------------------------------------------------------------------------------------
16      CONTINUACIÓN pseudocódigo 09
17      LLAMAR función enviar mensaje (alerta)
18      LLAMAR función enviar imagen (ruta_imagen.jpg)
19  RETORNA al BLUQUE PRINCIPAL
```

4.4. Integración del Sistema

Esta sección final del capítulo IV, describe el proceso de integración físico y lógico del prototipo final. Inicia con el montaje del hardware, luego con la unión y puesta en marcha de los módulos de software.

4.4.1. Montaje de Hardware

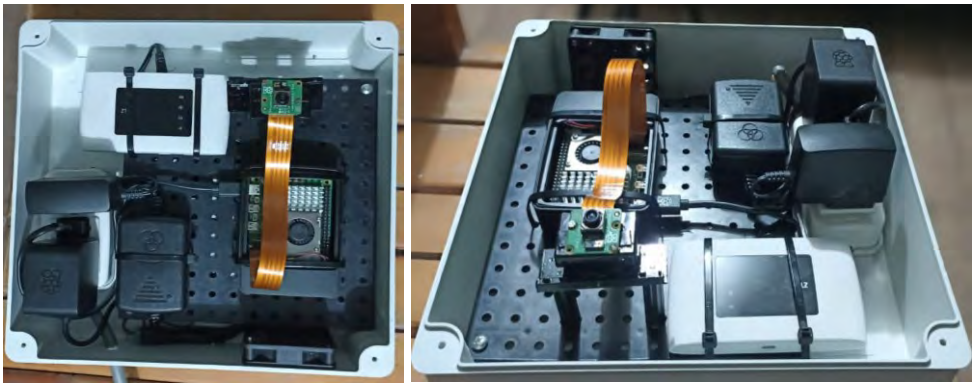
En esta sección se priorizo la protección componentes electrónicos en el entorno de pruebas ante las condiciones ambientales.

- **Alojamiento del sistema:** los componentes electrónicos y fuentes de alimentación eléctrica se resguardaron en una caja hermética con dimensiones 25 x 25 x 10 cm.
- **Montaje de cámara:** La cámara Raspberry Pi V3 se adaptó a una estructura dinámica, diseñada para permitir la manipulación del ángulo y dirección de la cámara.
- **Sistema de extracción:** Adicional al sistema de ventilación propia de la Raspberry Pi 5 se instaló un ventilador de 12v, permitiendo de esta forma la extracción del aire caliente generado por los dispositivos que fueron instalados dentro de la caja hermética

En la figura 59 se muestra la disposición y el montaje de los dispositivos dentro de la caja hermética.

Figura 59

Montaje del sistema detector de incendios forestales



4.4.1.1. Adaptación del Suministro Eléctrico

Uno de los desafíos logísticos al momento de realizar las pruebas en campo del prototipo del sistema fue el suministro de energía eléctrica. Para garantizar la autonomía operativa durante los periodos de evaluación se utilizó un kit solar portátil, el cual permitió tener suministro eléctrico (220V CA) de manera sostenida por el tiempo en que se requiera hacer las pruebas. Las especificaciones del kit solar portátil se muestran en la siguiente tabla.

Tabla 15

Especificaciones técnicas de kit solar portátil

Características	Especificaciones
Capacidad de batería	12V / 9Ah
Tipo de batería	Lead – Batería acida
Salida AC	220V / 50 Hz / 300W
Salida USB	5V / 2.4 A
Entrada DC (carga)	18V / 3 A

4.4.2. Integración del Software

La integración del software se logró al unificar todos los scripts modulares propuestos en la etapa de desarrollo del código (4.3.2) en un único ejecutable. La principal modificación fue el reemplazo de adquisición de video: se reemplazó la lectura de videos pre grabados por el fragmento de código 01: adquisición del video mediante la cámara Raspberry Pi V3 (Wide).

El código final, que integra todos los modulo, se guardó en una carpeta de archivos de la Raspberry Pi 5 y se detalla por completo en Anexo 03.

4.4.3. Verificación de Funcionalidad Conjunta

Para realizar la verificación de funcionalidad, primero se conectó el sistema embebido Raspberry Pi a Internet mediante el uso del modem inalámbrico 4G, pues este permite que el sistema embebido pueda realizar el envío de notificaciones por Telegram, caso contrario detectara un error de funcionamiento.

Una vez establecida la conectividad, se ejecutó el código final en la Raspberry Pi 5 y si después de la ejecución del código se inicia sin generar errores, se puede afirmar que el sistema está operando con total normalidad y listo para la fase de pruebas.

Capítulo V: Pruebas y Resultados

5.1. Introducción

En esta sección se detallan las metodologías de pruebas implementadas para validar el prototipo del sistema y se presenta los resultados obtenidos para evaluar su desempeño en la detección de incendios forestales.

Para el proceso de validación del proyecto de ingeniería se dividió en dos fases principales, los cuales permitieron calibrar el software.

- **Pruebas en condición controlada (simulación):** Esta fase fue como un banco de pruebas preliminares, empleando videos pre – grabados, donde se puede apreciar en escena la formación de humo y fuego forestal. Se logró hacer algunas calibraciones en los parámetros del prototipo (resolución, umbrales KNN, rangos de color HSV, etc.).
- **Pruebas de campo (entorno real):** En esta fase se verificó la viabilidad operativa del prototipo al trasladarlo al sector de Picol Orccompucyo. Esta fase se dividió en dos etapas, una para validar los parámetros iniciales y la segunda para afinar el código y obtener las métricas de desempeño final.

Es de suma importancia mencionar el rigor ético y social de las pruebas en campo. se realizaron diligencias para solicitar la autorización de la comunidad campesina de Picol Orccompucyo. Donde la solicitud fue aprobada en asamblea, previa exposición de la metodología a usar en las pruebas y garantizando la seguridad del entorno, respetando la propiedad comunal.

5.2. Pruebas y Resultados en Condición Simulada

En esta sección se presenta las pruebas y los resultados obtenidos en condición de simulación. Estas pruebas fueron repetitivas para evaluar el código diseñado y en caso requiera modificar los parámetros del prototipo.

5.2.1. Pruebas en Condición Simulada

Tal como se argumentó en la etapa de justificación del diseño en la sección 3.7 justificación de decisiones del diseño, para las primeras pruebas se utilizó videos pregrabados. Con la finalidad de evaluar el algoritmo planteado y ajustar parámetros del sistema.

Se utilizó 19 videos pregrabados donde se aprecia la existencia de humo y fuego en entornos forestales. Los videos se muestran en la tabla 16.

Tabla 16

Lista de bases de datos de videos pregrabados

Nº	**Nombre del video**	**Duración**	**Resolución**	**Fps**
01	kamenolom 002.avi	00:12:10	1200 x 676	25
02	Kozjak2012-2-001.avi	00:11:51	1200 x 676	50
03	Kozjak2012-2-002.avi	00:06:10	1200 x 676	50
04	Malacka2012-1-002.avi	00:06:18	1200 x 676	50
05	Trnbusi2012-1-001.avi	00:02:16	1200 x 676	50
06	Krivodol1.avi	00:03:41	1200 x 676	50
07	Kastela2012-1-001.avi	00:06:23	1200 x 676	50
08	NOCHE1.mp4	00:00:20	608 x 342	24
09	fuego1.mp4	00:00:21	1280 x 672	30
10	fuego2.mov	00:00:15	1280 x 720	30
11	fuego3.mp4	00:00:14	1920 x 1080	30
12	fuego4.mp4	00:00:14	1280 x 720	30
13	20090409ManavgatTEst_1.mp4	00:11:02	424 x 320	25
14	2000770817_Aksehir_Duman_Test5.mp4	00:13:31	600 x 480	30
15	Pelco_Colakli_1.mp4	00:02:01	600 x 480	25
16	Smoke_Manavgat_Raw.avi	00:04:01	352 x 288	25
17	forest1.avi	00:00:13	400 x 256	15
18	forest5.avi	00:00:14	400 x 256	15
19	ForestFire1.avi	00:00:13	400 x 256	15

Figura 60

Videos pregrabados de base de datos Profesor A. Enis Cetin

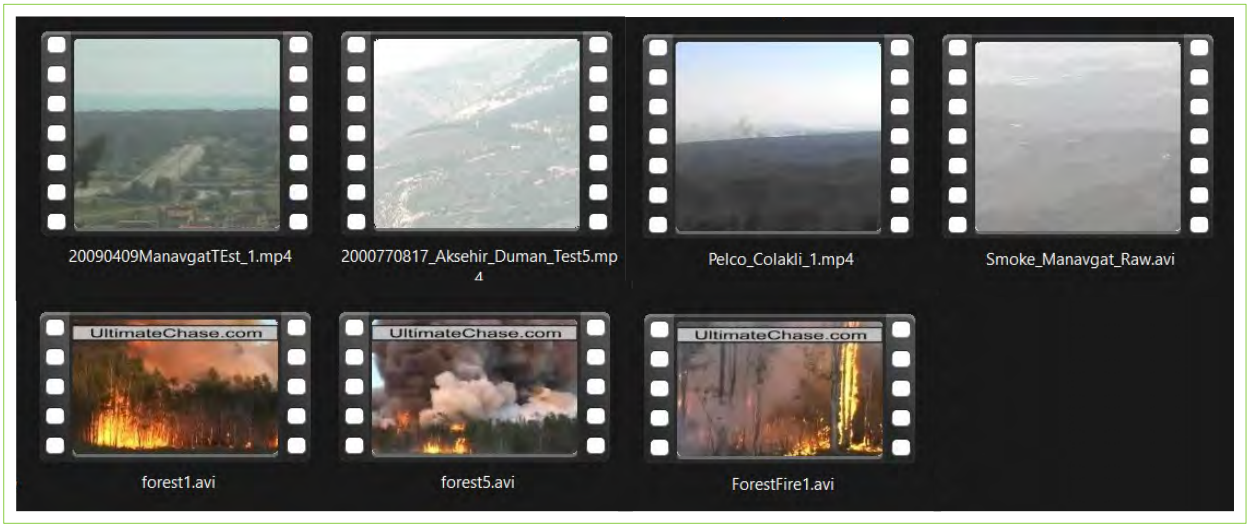


Figura 61

Videos pregrabados de base de datos "Wildfire Observers and Smoke Recognition Homepage"

Adicionalmente se utilizaron 04 videos que fueron grabados en el sector de Picol Orccompucyo con la cámara de un smartphone. Estos videos fueron grabados en las fechas 09-02-

2025, 11-03-2025 y 16-03-2025 donde se puede apreciar la aparición de humo en el entorno de aplicación.

Tabla 17

Lista de videos pregrabados en sector Picol Orccompucyo

Nº	Nombre del video	Duración	Resolución	Fps
1	picol-09022025.mp4	00:03:21	1280 x 1028	30
2	picol-09022025-2.mp4	00:01:40	1280 x 896	30
3	Picol humo tarde.mp4	00:02:19	1980 x 720	30
4	picol11032025-3.mp4	00:04:34	1920 x 1080	30

Figura 62

Videos pregrabados Picol Orccompucyo

5.2.2. Resultados en Condición Simulada

Para evaluar los resultados utilizando los videos pregrabados, se implementó un análisis de detección binaria de un evento a nivel del video. Se definieron los siguientes criterios para clasificar los resultados:

- **Verdadero positivo (VP):** El prototipo del sistema detectó por lo menos una vez la existencia de humo o fuego real durante el video.

- **Falsos positivos (FP):** El prototipo del sistema realizo detecciones de humo o fuego y estos no corresponden a humo o fuego real (falsa alarma).
- **Falso negativo (FN):** El prototipo no ogro detectar humo o fuego real en el video donde si había un evento presente.

En la siguiente tabla se presenta todos los posibles resultados de evaluación del prototipo del sistema.

Tabla 18

Matriz de evaluación de eventos

Situación	Resultado
Detección de humo o fuego real.	Verdadero Positivo (VP)
Detección de humo o fuego real y otras zonas falsas.	Verdadero Positivo (VP) + Falso Positivo (FP)
No se detectó humo o fuego real, pero si zonas falsas.	Falso Negativo (FN) + Falso Positivo (FP)
No se detectó humo o fuego real.	Falso Negativo (FN)

Después de realizar las evaluaciones a los 23 videos de prueba se obtuvo los siguientes resultados y se muestra en la tabla 19:

Tabla 19

Resultados de videos pregrabados

Nº	Video	VP	FP	FN
01	kamenolom 002.avi	SI	NO	NO
02	Kozjak2012-2-001.avi	SI	NO	NO
03	Kozjak2012-2-002.avi	NO	NO	SI
04	Malacka2012-1-002.avi	SI	NO	NO
05	Trnbusi2012-1-001.avi	SI	NO	NO
06	Krivodol1.avi	SI	NO	NO

07	Kastela2012-1-001.avi	SI	NO	NO
08	NOCHE1.mp4	SI	NO	NO
09	fuego1.mp4	SI	NO	NO
10	fuego2.mov	SI	NO	NO
11	fuego3.mp4	SI	NO	NO
12	fuego4.mp4	SI	NO	NO
13	20090409ManavgatTEst_1.mp4	SI	SI	NO
14	2000770817_Aksehir_Duman_Test5.mp4	SI	SI	NO
15	Pelco_Colakli_1.mp4	SI	NO	NO
16	Smoke_Manavgat_Raw.avi	SI	NO	NO
17	forest1.avi	SI	NO	NO
18	forest5.avi	SI	NO	NO
19	ForestFire1.avi	SI	NO	NO
20	picol-09022025.mp4	SI	NO	NO
21	picol-09022025-2.mp4	SI	NO	NO
22	Picol humo tarde.mp4	SI	NO	NO
23	picol11032025-3.mp4	SI	NO	NO
SUMATORIA DE DETECCIONES		22	2	1

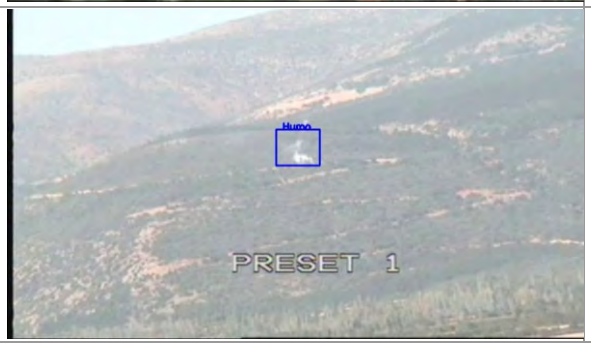
La tabla 20 presenta ejemplares de las detecciones realizadas por el sistema. Es importante mencionar que durante la ejecución del código se realizaron más detecciones de humo o fuego (verdaderos positivos) y las mostradas son representaciones de ello. La cantidad de falsos positivos y falsos negativos se colocan en su totalidad. Se presenta la siguiente tabla con el fin de ilustrar el desempeño del sistema de forma gráfica.

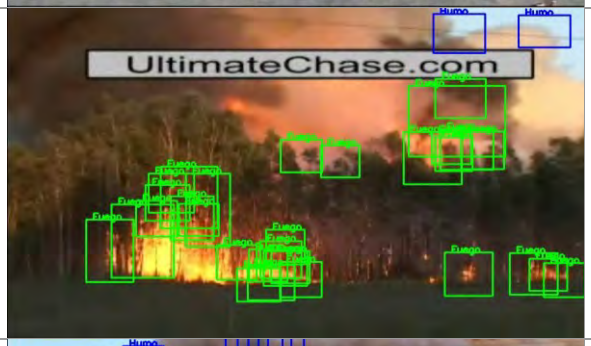
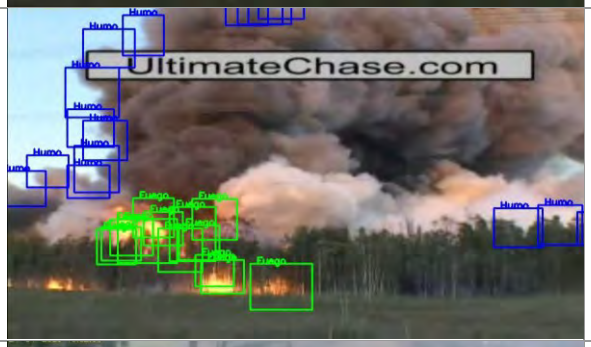
Tabla 20

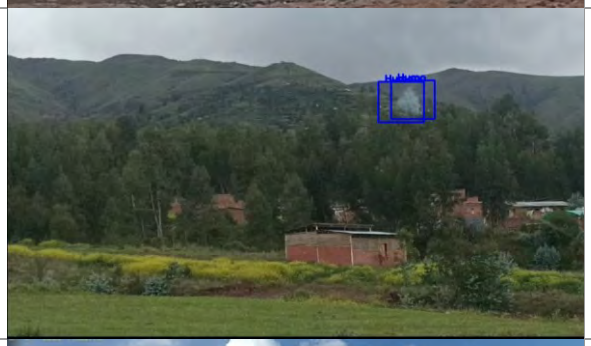
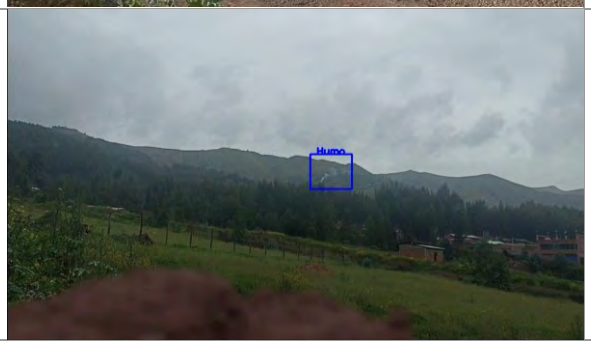
Resultados por video: verdaderos positivos, falsos positivos, falso negativos

N.° VIDEO	VERDADERO POSITIVOS	FALSOS POSITIVO
01		No detectó falsos positivos.
02		No detectó falsos positivos.
03	No detecto verdaderos positivos	No detectó falsos positivos.
04		No detectó falsos positivos.

05		No detectó falsos positivos.
06		No detectó falsos positivos.
07		No se detectó falsos positivos.
08		No detectó falsos positivos.
09		No detectó falsos positivos.

10		No detectó falsos positivos.
11		No detectó falsos positivos.
12		No detectó falsos positivos.
13		
14		

15		No detectó falsos positivos.
16		No detectó falsos positivos.
17		No detectó falsos positivos.
18		No detectó falsos positivos.
19		No detectó falsos positivos.

20		No detectó falsos positivos.
21		No detectó falsos positivos.
22		No detectó falsos positivos.
23		No detectó falsos positivos.

Para hallar las métricas primero se contabilizó la cantidad de verdaderos positivos, falsos positivos y falsos negativos, estos valores permitió saber que tan eficiente es el sistema para hacer detección de humo o fuego en tiempo real.

Para hallar la precisión, la sensibilidad y balance general F1 Score del sistema se utilizó las siguientes formulas.

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ positivos + Falsos\ Positivos}$$

$$Sensibilidad = \frac{Verdaderos\ Positivos}{Verdaderos\ positivos + Falsos\ Negativos}$$

$$F1 = 2 * \frac{precisión * \ sensibilidad}{precisión + sensibilidad}$$

Para el análisis de resultados se tomaron 23 videos de prueba donde existe humo o fuego real en cada escena. Del total en 22 escenarios se tuvo detecciones humo y fuego detectado (Verdadero Positivo). 02 detecciones de elementos que no fueron humo o fuego (Falsos Positivos) y solo en un escenario no se detectó humo real (Falso Negativo)

$$Precisión = \frac{22}{22 + 2} = 0.916 = 91.6\%$$

$$Sensibibilidad = \frac{22}{22 + 1} = 0.956 = 95.6\%$$

$$F1 = 2 * \frac{0.916 * \ 0.956}{0.916 + 0.956} = 0.935 = 93.5\%$$

Los resultados muestran que el sistema hace una buena detección de humo y fuego real, ya que no pudo detectar humo real en uno de los videos de prueba, esto debido a la baja iluminación, baja concentración y la corta duración del humo en la escena. En caso de la precisión se observa que hubo falsas detecciones, esto debido a movimientos bruscos de la cámara y aparición de objetos en movimiento con cromaticidad parecida al humo.

Para finalizar el balance general nos muestra que el sistema es bueno obteniendo un 93.5%, demostrando que existe un alto grado en detección de humo y fuego real y un bajo grado de localización de falsas detecciones.

5.2.3. Parámetros Óptimos del Sistema en Condición Simulada

En la tabla 21 se muestra los valores para los cuales se obtuvieron los resultados de desempeño anteriormente presentados.

Tabla 21

Parámetros óptimos en condición simulada

Parámetros de diseño	**Valores**
Nº de frame análisis (movimiento continuo)	Depende al video (aprox. 7 seg.)
Rango HSV para humo	lower_hsv1 = np.array([33, 5, 120]) upper_hsv1 = np.array([165, 60, 220])
Rango HSV para fuego	lower_hsv2 = np.array([0, 143, 30]) upper_hsv2 = np.array([17, 255, 255])
Parámetros algoritmo KNN (K-Nearest Neighbor)	history= 900, dist2Threshold= 700, detectShadows=False
FPS del video	Promedio de 30
Resolución de imagen	960 x 540

Con estos valores se asegura que el sistema tenga un rendimiento similar en el lugar de aplicación real.

5.3. Pruebas Iniciales del Sistema en un Entorno Real (Cámara Raspberry Pi V3 Wide)

En esta sección se describe las pruebas realizadas en entorno real utilizando la cámara Raspberry Pi V3 (wide). Con estas pruebas se demuestra el funcionamiento en condiciones reales y se valida los requerimientos funcionales planteados, tales como monitoreo constante, precisión en detección y rapidez en tiempo de respuesta.

Las pruebas en entornos reales fueron planificadas para ejecutarse en 15 días, considerando distintos horarios y ubicaciones a fin de evaluar el desempeño del sistema en diversas condiciones.

La planificación se dividió en dos etapas:

- **Etapa de ajuste (4 días):** Destinada a detectar variables que podrían intervenir en el mal funcionamiento del prototipo y afinar los parámetros óptimos en un entorno real.
- **Etapa de evaluación final (11 días):** Destinados para realizar la evaluación final del sistema.

Cabe recalcar que, durante la fase de ejecución de pruebas se presentaron limitaciones para la obtención de permisos en los predios urbanos que contaban con estructuras elevadas, puntos cercanos al seleccionado en la sección de diseño. Sin embargo, se optó por ubicar el prototipo en terreno libre que, aunque a menor altura, mantuvo la línea de visión a la zona de interés (intersección área agrícola - forestal), este punto también facilitó la ejecución de las pruebas controladas mediante el uso de bengalas de humo. El punto de monitoreo final se georreferenció en la Zona UTM 19L, con coordenadas 187,588.00 m E (Este) y 8,501,930.00 m N (Norte).

Figura 63

Prototipo del sistema detector de incendios forestales.

En la tabla 22 se presenta el cronograma de pruebas incluida las horas de prueba y cantidad de simulaciones de humo en escena.

Tabla 22

Cronograma de pruebas iniciales – primera cámara

Fecha	Hora inicio	Hora fin	N.º pruebas	Observaciones
12-03-2025	09:00	12:00	02	Humo
18-03-2025	09:00	13:00	03	Humo
28-03-2025	16:00	18:00	01	Humo
05-04-2025	11:00	13:30	01	Humo

En la tabla 23 se presenta los resultados obtenidos con la cámara Raspberry Pi 5, donde se presenta las cantidades de verdaderos positivos, falsos positivos y falsos negativos.

Tabla 23

Resultados de pruebas iniciales – primera cámara

Fecha	Prueba N. º	Hora inicio	Hora final	VP	FP	FN
12-03-2025	01	09:00	10:30	1	10	0
12-03-2025	S/P	10:30	11:30	0	06	0
12-03-2025	02	11:30	12:00	1	34	0
18-03-2025	01	10:30	10:55	1	02	0
18-03-2025	02	10:56	11:07	1	0	0
18-03-2025	03	11:08	12:00	1	0	0
28-03-2025	01	16:00	18:00	0	0	1
05-04-2025	01	11:00	13:30	1	5	0

Como se puede observar el primer día de análisis se obtuvo muchos falsos positivos y estos fueron disminuyendo con el afinamiento en los parámetros del código y ajustes físicos del sistema.

El tercer día de análisis se realizó una prueba en horas de 4 pm a 6 pm donde las condiciones de iluminación fueron bajas y se pudo apreciar un falso negativo.

Con estas primeras pruebas se pudo detectar la presencia de varios falsos positivos y falsos negativos, esto debido a condiciones ambientales como vientos fuertes, nubes con tonalidades similares a humo y cambios extremos de iluminación en escena.

También se modificó los rangos de color HSV ya que existía elementos que cumplían ciertas condiciones de análisis de detección de humo y fuego como: cromaticidad de eucaliptos retoños, efectos estroboscópicos por parte de elementos reflejantes y vegetación en movimiento que cumplían con las condiciones de análisis.

En las figuras 64 y 65 se presenta los ejemplares de estas detecciones:

Figura 64

Verdaderas detecciones primeras pruebas en campo

Figura 65

Falsas detecciones primeras pruebas en campo

5.4. Validación del Sistema con Otro Tipo de Cámara

Si bien el prototipo fue diseñado para funcionar con el módulo de cámara Raspberry Pi V3 (Wide). No obstante, durante la etapa de pruebas de ajuste realizadas en condiciones de baja iluminación (aproximadamente 18:00 horas), se apreció que el sensor de la cámara Raspberry Pi V3 alcanzaba su umbral máximo de subexposición, tonos cercanos al negro, impidiendo que el algoritmo capture imágenes útiles.

Ante esta limitación operativa, se procedió a realizar una validación cruzada sustituyendo la fuente de captura de imágenes por una cámara IP Tapo C320WS. Es importante mencionar que esta validación no se realizó con ambas cámaras funcionando simultáneamente, sino mediante un intercambio físico del periférico de entrada.

Este cambio permitió demostrar el desempeño del sistema en diversas circunstancias de imagen, calidad de video y conectividad. Evidenciando la versatilidad del código con diferentes flujos de video.

Las especificaciones técnicas y la comparativa con la cámara seleccionada inicialmente se muestra a continuación la tabla 24.

Tabla 24

Especificaciones técnicas de cámara Tapo C320WS

	Camara TAPO C320WS	**Raspberry Pi V3 Wide**
Definición de video / resolución	2K QHD (2560 x 1440 px)	Max. 2304 x 1296 px
Sensor de imagen	1/3” CMOS con sensor Starlight	Sony IMX708 (1/3”)
Focal - ratio	f/1.61	F2.2
Campo de visión (H)	108 º	No especificado (standard)
Fps	15	Max. 120 fps
Conectividad	Wi-Fi 2.4 GHz / RJ45	Puerto CSI - 2
Protocolo de video	RTSP, ONVIF	Lib camera 2
Consumo energético	9.0 V / 0.6 A	Dependiente a Raspberry pi
Costo Referencial	S/. 207.00	S/. 210.00

Adicionando a este paso el código realizado para las pruebas con la segunda cámara se muestra en Anexo 04.

5.4.1. Pruebas Iniciales del Sistema en un Entorno Real con la Segunda Cámara

De la misma forma que se realizó con la cámara Raspberry Pi V3 (Wide), se configuró la segunda cámara con los parámetros óptimos obtenidos en la fase de simulación. En este caso, se pudo observar que el segundo dispositivo requirió menos modificaciones de los parámetros iniciales, ya que con estos se obtuvieron buenos resultados.

La diferencia de focal – ratio (f/2.2 a f/1.61) permitió que este segundo modelo de cámara capte mayor entrada de luz, manteniendo la visibilidad al anochecer. Sumado a esto, la tecnología Starligth optimizó la captura de imágenes a color incluso en condiciones de baja iluminación.

Asimismo, la interoperabilidad del sistema permitió que el código no sufriera modificaciones estructurales, solo cambiando la dirección de fuente de video mediante protocolo RTSP. En relación costo beneficio permitió integrar una cámara con mejores prestaciones a un costo similar. En la tabla 25 se muestra los días que se realizó las pruebas, periodo de pruebas y cantidades de pruebas realizadas. En la tabla 26 se presenta la cantidad de verdaderos positivos y falsos positivos que se obtuvieron con la cámara Tapo C320WS.

Tabla 25

Cronograma de pruebas iniciales – segunda cámara

Fecha	Hora inicio	Hora fin	N.º pruebas	Observaciones
18-03-2025	11:00	12:00	01	Humo
28-03-2025	16:00	18:00	01	Humo
05-04-2025	11:00	13:30	01	Humo

Tabla 26

Resultados de pruebas iniciales – segunda cámara

Fecha	Prueba N. º	Hora inicio	Hora final	VP	FP	FN
18-03-2025	01	11:00	12:00	1	0	0
28-03-2025	01	16:00	18:00	1	1	0
05-04-2025	01	11:00	13:30	1	7	0

Las detecciones de segunda cámara Tapo C320WS se muestran en las figuras 66 y 67.

Figura 66

Verdaderas detecciones primeras pruebas en campo segunda cámara.

Figura 67

Falsas detecciones primeras pruebas en campo segunda cámara.

5.5. Conclusiones Preliminares de Primeras Pruebas

Las primeras pruebas (Fase de ajuste) fueron fundamentales para modificar y optimizar algunos sectores del código que inicialmente fue desarrollado. En la versión 01 donde se aplicó detección de movimiento, detección mediante espacios de color, intersección de máscaras y persistencia de movimiento. Los resultados obtenidos en los videos de prueba fueron prometedores, lo cual daba un buen indicio del funcionamiento en un entorno real.

Sin embargo, en las pruebas de campo se evidenció otra realidad donde se detectó muchos falsos positivos tal como se muestra en los resultados de las primeras pruebas. Estos errores se solucionaron aplicando las siguientes mejoras:

- **Segmentación de la imagen a una región de interés (ROI):** Se segmento la imagen para excluir áreas problemáticas, ya que la mayoría de falsos positivos se originaban en las nubes, que cumplían condiciones de movimiento y cromaticidad.
- **Mitigación de la inestabilidad física**: Las ráfagas de viento causaban inestabilidad en el prototipo, esto se solucionó reubicando el prototipo a una zona más protegida y añadiendo un contrapeso al soporte de la cámara y una caja cortavientos a los laterales del montaje.
- **Implementación de filtros (reducción de ruido):** Se detectó elementos cercanos a la cámara los cuales generaban mucho ruido a causa del movimiento constante y sumado a su tonalidad similar al humo o fuego, ocasionaban estas falsas detecciones, es por ello que se utilizaron técnicas de filtrado a cada máscara resultante y filtrado de regiones pequeñas.
- **Reducción de resolución:** Se determinó que la resolución inicial de 1280 x 720 px captaban demasiados detalles no deseados. Por ello se tuvo que reducir en un 25%

llegando a utilizar una resolución de 960 x 540 px, lo que permitió generalizar la detección de grandes áreas y esto también produjo indirectamente la reducción de consumo de recursos computacionales.

Con todas estas observaciones se diseñó la versión final del código, el cual fue puesto en prueba para obtener las métricas finales de desempeño del prototipo del sistema.

5.6. Resultados Finales en un Entorno Real

Esta sección presenta los resultados cuantitativos y cualitativos obtenidos con la versión final del código, calibrado en el entorno de aplicación, minimizando las falsas detecciones.

Para esta etapa final de evaluación los parámetros fueron ajustados y diferenciados para cada cámara. La variación entre rangos de color HSV entre las cámaras es atribuible a las diferencias intrínsecas en sus sensores de imagen y los algoritmos de procesamiento internos, que afecta directamente el equilibrio de blancos y la representación cromática.

Tabla 27

Parámetros de diseño final para cámara Raspberry Pi V3 (Wide)

Parámetros de diseño	**Valores**
Nº de frame análisis	300 (equivalente a 10 seg. a 30 FPS)
Rango HSV para humo	lower_hsv1 = np.array([61, 10, 180]) upper_hsv1 = np.array([133, 55, 255])
Rango HSV para fuego	lower_hsv2 = np.array([0, 5, 220]) upper_hsv2 = np.array([40, 255, 255])
Parámetros algoritmo KNN (K-Nearest Neighbor)	history=900, dist2Threshold=700, detectShadows=False
FPS captura de video	30 fps
Resolución de imagen	960 x 540 px

Tabla 28

Parámetros de diseño final para cámara IP TapoC320WS

Parámetros de diseño	**Valores**
Nº de frame análisis	150 (equivalente a 10 seg. A 15 FPS)
Rango HSV para humo	lower_hsv1 = np.array([42, 5, 160]) upper_hsv1 = np.array([142, 63, 205])
Rango HSV para fuego	lower_hsv2 = np.array([0, 3, 235]) upper_hsv2 = np.array([55, 255, 255])
Parámetros algoritmo KNN (K-Nearest Neighbor)	history=900, dist2Threshold=700, detectShadows=False
FPS captura de video	15 fps
Resolución de imagen	960 x 540 px

5.6.1. Resultados Final con Cámara Raspberry Pi V3 (Wide)

En la presente sección se muestra los resultados obtenidos por ambas cámaras y las métricas de precisión, sensibilidad y balance general. Cabe mencionar que desde este punto se tomaron datos para hallar la eficiencia del sistema en el entorno de aplicación.

A continuación, se muestra las tablas 29 y 30 donde se describen las fechas en las que se hicieron las pruebas respectivas y las cantidades de las detecciones encontradas, todas ellas obtenidas mediante el uso del módulo de cámara Raspberry Pi V3 (Wide).

Adicionalmente, en la tabla 31 se muestra los ejemplares de las verdaderas y falsas detecciones en cada prueba.

Tabla 29

Cronograma de pruebas finales

Fecha	Hora inicio	Hora fin	N.º pruebas	Observaciones
09-04-2025	09:00	11:00	01	Humo
10-04-2025	09:00	12:00	02	Humo
11-04-2025	11:00	13:00	01	Humo
12-04-2025	08:30	11:30	04	Humo
07-05-2025	13:30	15:20	01	Humo
11-05-2025	18:00	19:57	02	Fuego
12-05-2025	17:40	18:00	02	Fuego
13-05-2025	14:00	15:00	01	Humo
14-05-2025	14:30	16:30	04	Humo
18-05-2025	18:30	19:30	02	Fuego

Tabla 30

Resultados de pruebas finales – primera cámara

Fecha	Prueba N. º	Distancia	Hora inicio	Hora final	VP	FP	FN
09-04-2025	01	~120 m	09:00	11:00	1	0	0
10-04-2025	01	~353 m	09:00	10:20	1	0	0
10-04-2025	02	~493 m	10:20	12:00	1	1	0
11-04-2025	01	~353 m	11:00	13:00	1	0	0
12-04-2025	01	~069 m	08:00	09:00	1	0	0
12-04-2025	02	~136 m	09:00	09:05	1	0	0
12-04-2025	03	~353 m	09:05	09:10	1	1	0
12-04-2025	04	~493 m	09:10	11:30	1	1	0
07-05-2025	01	~150 m	13:30	15:20	1	3	0
11-05-2025	01	~005 m	18:00	18:40	1	0	0
11-05-2025	02	~009 m	18:40	19:57	1	0	0

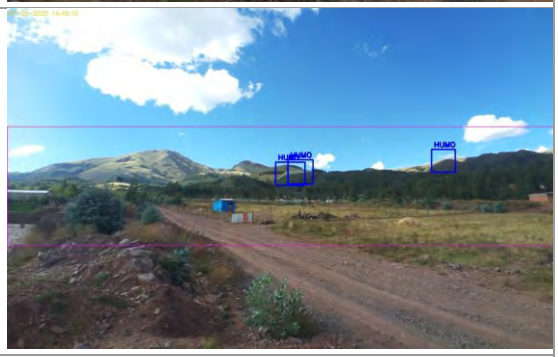
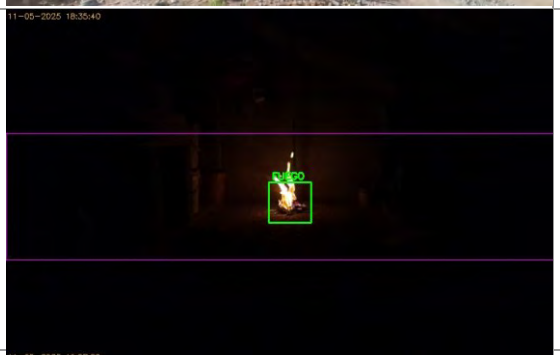
12-05-2025	01	~005 m	17:40	17:50	1	0	0
12-05-2025	02	~009 m	17:50	18:00	1	0	0
13-05-2025	01	~098 m	14:00	15:00	0	0	1
14-05-2025	01	~040 m	14:30	15:06	1	0	0
14-05-2025	02	~136 m	15:06	15:13	1	0	0
14-05-2025	03	~353 m	15:13	15:20	1	0	0
14-05-2025	04	~493 m	15:20	16:30	1	0	0
18-05-2025	01	~70 m	18:30	19:00	0	0	1
18-05-2025	02	~150 m	19:00	19:30	0	0	1
Sumatoria de detecciones					**17**	**6**	**3**

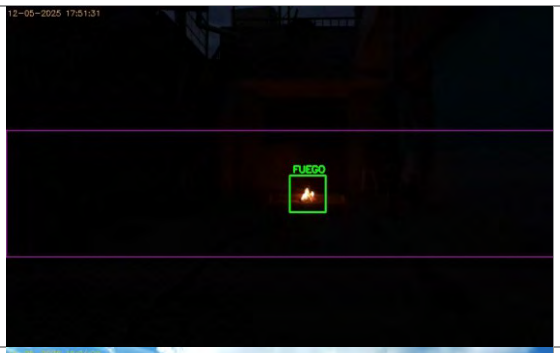
Tabla 31

Resultados por video: verdaderos positivos, falsos positivos, falso negativos – pruebas finales primera cámara

FECHA	N.º PRUEBA	VERDADERO POSITIVOS	FALSOS POSITIVO
09-04-2025	**01**		No se detectó falsos positivos.
10-04-2025	**01**		No se detectó falsos positivos.

	02		
11-04-2025	**01**		No se detectó falsos positivos.
12-04-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.
	03		

	04		
07-05-2025	**01**		
11-05-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.
12-05-2025	**01**		No se detectó falsos positivos.

	02		No se detectó falsos positivos.
14-05-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.
	03		No se detectó falsos positivos.
	04		No se detectó falsos positivos.

De la misma forma que se realizó en condición simulada para hallar las métricas primero se contabilizó la cantidad de verdaderos positivos, falsos positivos y falsos negativos, estas cantidades permitió conocer la eficiencia del sistema para hacer detección de humo o fuego en el sector de aplicación.

Se analizaron 20 escenarios de pruebas donde existe humo o fuego real en cada escena. Del total en 17 escenarios se tuvo detecciones de humo y fuego (Verdadero Positivo), 06 detecciones de elementos que no fueron humo o fuego (Falsos Positivos) y en 03 escenario no se detectó humo y fuego real (Falso Negativo).

Aplicando las fórmulas para hallar la precisión, sensibilidad y el balance general se obtuvo los siguientes resultados:

$$Precisión = \frac{17}{17 + 06} = \ 0.739 = 73.9\%$$

$$Sensibibilidad = \frac{17}{17 + 03} = \ 0.85 = 85\%$$

$$F1 = 2 * \frac{0.739 * 0.85}{0.739 + 0.85} = 0{,}79 \ = 79\%$$

Los resultados finales en entorno real, a pesar de las correcciones de código, muestran un rendimiento esperado, demostrando que la complejidad de las variaciones ambientales impacta en las métricas de desempeño. La prueba realizada el día 13-05-2025 no detectó humo esto debido a la baja concentración de humo, mientras que la prueba realizada el 18-05-2025 no logró detección de fuego en las 02 pruebas esto debido a la baja capacidad por parte de la cámara al capturar imágenes en condición de baja iluminación. En caso de la precisión se observa que hubo falsas detecciones, esto debido al movimiento rápido de nubes en la escena, especialmente en horas pasados el meridiano cuando los vientos son más intensos y los cambios repentinos de iluminación

especialmente cuando la escena era opacada repentinamente por sombras de nubes y el efecto estroboscópico de calaminas.

El balance general muestra que el sistema es bueno obteniendo un 79%, demostrando una buena capacidad en detección, pero también resalta las limitaciones inherentes a la selección del hardware y variables ambientales.

Para finalizar, es importante mencionar que el uso del módulo de cámara Raspberry Pi v3 (Wide) tiene un grado mayor de dificultad en su uso, debido a que sus principales parámetros (contraste, nitidez, exposición y balance de blancos) no están preconfigurados por defecto y estos deben calibrarse en código para encontrar los valores exactos que den una imagen con la cromaticidad real del entorno, justificando por que los rangos de colores HSV finales difieren de los utilizados en simulación. Una segunda observación es que la cámara Raspberry Pi V3 (Wide) tiene limitaciones en condiciones de baja iluminación esto debido al sensor de imagen que posee dicha cámara. Aunque en las especificaciones técnicas la cámara puede activar la funcionalidad HDR (High Dynamic Range), el cual permite mejorar la calidad de las imágenes capturadas en condiciones de baja iluminación o alto contraste. Sin embargo, se detectó limitaciones en cuanto a su compatibilidad con la librería Picamera2 de OpenCV y a la resolución utilizada.

5.6.2. Resultados Final con Cámara TAPO C320WS

Para viabilizar las pruebas en el mismo punto y evitar complicaciones de conexión física en la Raspberry Pi 5, se optó por ejecutar por ejecutar el código utilizando una laptop con sistema operativo Linux. Esta configuración actuó como un entorno de ejecución equivalente, permitiendo procesar el flujo de video de la cámara TAPO mediante protocolo RTSP, una interfaz que el primer modelo de cámara (Raspberry Pi V3 wide) no permite de forma nativa.

El uso de la laptop como estación de procesamiento tuvo un objetivo metodológico clave: garantizar la integridad de los resultados finales sin duplicar las pruebas de humo y fuego. Dado que previamente se comprobó el correcto funcionamiento de la cámara TAPO en las pruebas de ajuste preliminar, esta configuración permitió validar el algoritmo final frente al mismo estímulo.

En lugar de realizar pruebas de humo o fuego por separado para cada cámara, en el entorno de la laptop permitió capturar y procesar eventos de validación de la cámara TAPO mientras el sistema embebido Raspberry Pi 5 mantenía su operatividad usando la primera cámara. De esta forma, se optimizó los recursos de campo y demostró que el software es interoperable.

A continuación, se muestra los resultados utilizando la segunda cámara, en la tabla 32 se muestra las fechas en las que se realizaron las pruebas y las detecciones encontradas. Adicionalmente en la tabla 33 se muestra los ejemplares de las verdaderas y falsas detecciones que fueron enviadas mediante mensajería instantánea.

Tabla 32

Resultados de pruebas finales – segunda cámara

Fecha	Prueba N. º	Distancia	Hora inicio	Hora final	VP	FP	FN
09-04-2025	01	~120 m	09:00	11:00	1	0	0
10-04-2025	01	~353 m	09:00	10:20	1	0	0
10-04-2025	02	~493 m	10:20	12:00	1	0	0
11-04-2025	01	~353 m	11:00	13:00	1	0	0
12-04-2025	01	~069 m	08:00	09:00	1	0	0
12-04-2025	02	~136 m	09:00	09:05	1	0	0
12-04-2025	03	~353 m	09:05	09:10	1	0	0
12-04-2025	04	~493 m	09:10	11:30	1	0	0
07-05-2025	01	~150 m	13:30	15:20	1	2	0
11-05-2025	01	~005 m	18:00	18:40	1	0	0
11-05-2025	02	~009 m	18:40	19:57	1	0	0

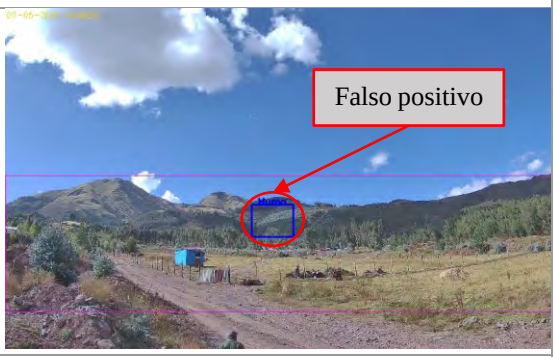
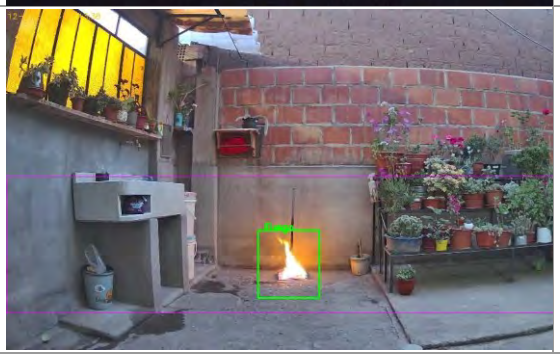
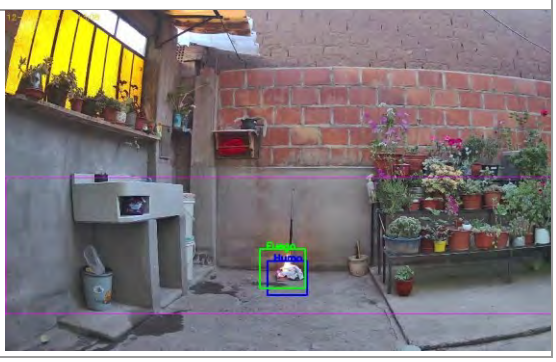
12-05-2025	01	~005 m	17:40	17:50	1	1	0
12-05-2025	02	~009 m	17:50	18:00	1	0	0
13-05-2025	01	~098 m	14:00	15:00	1	2	0
14-05-2025	01	~040 m	14:30	15:06	1	0	0
14-05-2025	02	~136 m	15:06	15:13	1	0	0
14-05-2025	03	~353 m	15:13	15:20	1	0	0
14-05-2025	04	~493 m	15:20	16:30	1	1	0
18-05-2025	01	~070 m	18:30	19:00	1	0	0
18-05-2025	02	~150 m	19:00	19:30	1	0	0
Sumatoria de detecciones					**20**	**6**	**0**

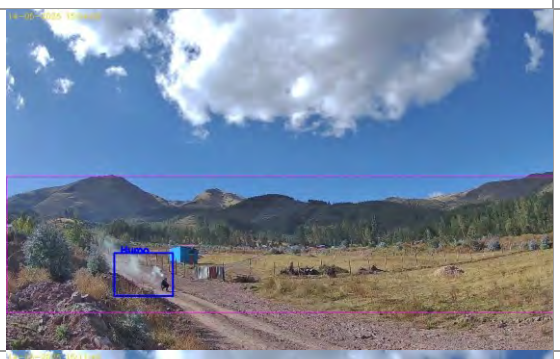
Tabla 33

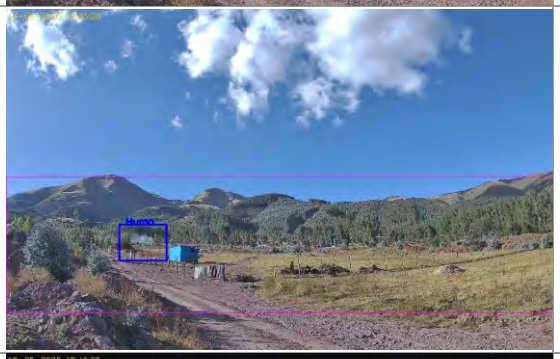
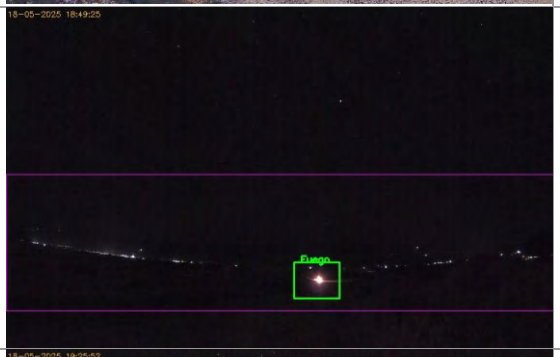
Resultados por video: verdaderos positivos, falsos positivos, falso negativos – pruebas finales segunda cámara

FECHA	N.º PRUEBA	VERDADERO POSITIVOS	FALSOS POSITIVO
09-04-2025	**01**		No se detectó falsos positivos.
10-04-2025	**01**		No se detectó falsos positivos.

	02		No se detectó falsos positivos.
11-04-2025	**01**		No se detectó falsos positivos.
12-04-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.
	03		No se detectó falsos positivos.

	04		No se detectó falsos positivos.
07-05-2025	**01**		
11-05-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.
12-05-2025	**01**		

	02		No se detectó falsos positivos.
13-05-2025	**01**		 
14-05-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.

	03		No se detectó falsos positivos.
	04		
18-05-2025	**01**		No se detectó falsos positivos.
	02		No se detectó falsos positivos.

Con la segunda cámara en total se realizó 20 pruebas donde existió humo simulado o fuego real en cada escena. En todas las pruebas se obtuvo detecciones de humo y fuego (Verdadero Positivo). 06 detecciones de elementos que no fueron humo o fuego (Falsos Positivos) y ningún falso negativo.

$$Precisión = \frac{20}{20 + 06} = 0.769 \ = 76.9\%$$

$$Sensibibilidad = \frac{20}{20 + 00} = \ 1 = 100\%$$

$$F1 = 2 * \frac{0.769 * 1}{0.769 + 1} = 0.869 \ = 86.9\%$$

Los resultados muestran que el uso de la segunda cámara tiene un mejor desempeño en comparación a la cámara Raspberry Pi V3 (Wide) esto debido a las mejores capacidades que tiene la cámara de videovigilancia. Entre las principales se pudo observar:

- **Eliminación de FN:** La cámara Tapo tiene una mejor fidelidad de color y un mejor rendimiento en condiciones de baja iluminación inherentes a la cámara Tapo, logrando un Recall perfecto.
- **Procesamiento integrado:** La cámara incluye procesamiento integrado y parámetros configurados por fabrica y mediante el uso del protocolo RTSP (Real-Time Streaming Protocol) permite envío de imágenes en tiempo real sin necesidad de modificar otros parámetros.
- **Eficiencia a menor FPS:** Aunque solo funciona a 15 fps el desempeño fue mejor en comparación con la primera cámara.

Para finalizar el balance general nos muestra que el sistema es bueno obteniendo un 86.9%, demostrando que existe un alto grado en detección de humo y fuego real y un bajo grado de localización de falsas detecciones.

5.7. Comparativa con Métricas Encontradas en los Antecedentes Académicos

En la tabla 34 se detalla las métricas obtenidas en el presente trabajo con los resultados reportados en los antecedentes académicos de referencia.

Tabla 34

Comparativa de métricas de rendimiento frente a los antecedentes académicos.

Autor / Trabajo	Método Principal	Tipo de Prueba	Precisión	Sensibilidad (Recall)	F1-Score / Exactitud
Condorhuaman (Tesis)	KNN + HSV	Real (Campo - Tapo)	76.9%	**100%**	86.9%
Condorhuaman (Tesis)	KNN + HSV	Real (Campo - RPi V3)	73.9%	85.0%	79.0%
Condorhuaman (Tesis)	KNN + HSV	Simulada (Bases datos)	91.6%	**95.6%**	**93.5%**
Mubarak & Honge	YCbCr + Temporal	Simulada (Internet)	92.59%	93.13%	92.86%
Thou-Ho et al.	RGB + Crecimiento	Simulada (Internet)	87.10%	88.40%	87.75%
Di Lascio et al. (Fenju Bu et al)	Color + Movimiento	Simulada (Mix)	(N/A)	(N/A)	92.86%
Domènech (Fuego)	HSV	Simulada (Dataset)	**98.3%**	71.0%	(N/A)
Domènech (Humo)	PDI Mixto	Simulada (Dataset)	(Variable)	20% - 80%	(Variable)

La evaluación de los algoritmos KNN + HSV se realizó en un enfoque progresivo, donde los resultados demuestran una solidez técnica competitiva a los antecedentes utilizados, incluso superando en métricas como sensibilidad y F1- Score, validando el algoritmo final en condiciones similares (videos de prueba).

Al cambiar a las pruebas en campo, la implementación con la cámara IP Tapo obtuvo una sensibilidad de 100% cifra que superó a los resultados obtenidos en los demás trabajos académicos.

Si bien la exactitud en los escenarios reales presenta una disminución natural debido a las variables climáticas del sector de Picol Orccompucyo.

5.8. Evaluación del Consumo de Recursos del Sistema

En la ejecución de pruebas reales del sistema se realizó la medición de parámetros principales esto con el objetivo de no solo determinar la eficiencia en detecciones, sino también determinar que lo haga de forma estable sin necesidad de consumir muchos recursos computacionales y energéticos.

5.8.1. Consumo Energético

El consumo energético máximo que se calculó en base a las especificaciones técnicas de los dispositivos utilizados y se muestra a continuación en la siguiente tabla.

Tabla 35

Consumo energético del prototipo del sistema

Dispositivos	Consumo
Raspberry Pi 5 8Gb (incluida fuente)	12.5 watts a procesos intensivos (Max.)
Active cooler	1 watts (Max.)
Cámara Tapo C320WS (incluida fuente)	600 mA a 9v. (5.4 watts)
Ventilador 12 voltios (incluida fuente)	130 mA a 12v (1.56 watts)
Modem ZTE 4G LTE (incluida fuente)	4 watts (Max.)
Total	**24.46 watts**

El valor de 24.46 W representa el consumo teórico máximo si todos los dispositivos funcionaran simultáneamente a su capacidad límite. Sin embargo, el consumo real es menor. Esto se confirma con la medición en la etapa de alimentación Alterna (CA) de la fuente de energía, donde el consumo promedio registrado fue de 16.5 W (validado con la pinza amperimétrica).

Aunque esta medición en CA incluye las pérdidas por eficiencia de todos los adaptadores (cargadores AC/DC), Esto reduce el consumo de potencia útil (DC) real del sistema.

Es importante mencionar que este consumo energético se realizó cuando se estaba utilizando el segundo modelo de cámara (Tapo C320WS), el cual tiene un consumo energético mayor, con el sistema embebido Raspberry Pi 5.

Figura 68

Consumo energético real del sistema

5.8.2. Consumo de Recursos Computacionales

El consumo de recursos se midió en el sistema embebido usando el comando de Linux htop. Esta herramienta proporcionó datos sobre CPU, RAM, procesos y tiempo de ejecución (Tablas 36 y 37).

Tabla 36

Consumo de recursos computacional del sistema usando cámara Raspberry Pi V3 (wide)

Parámetros	**Valor**
CPU (procesador)	120% (carga distribuida en 4 núcleos)
Memoria RAM	725 M
Tiempo de ejecución	Valor registrado en 56 min de ejecución de script

Tabla 37

Consumo de recursos computacionales de Raspberry Pi con cámara Tapo C320WS

Parámetros	Valor
CPU (procesador)	162.5% (carga distribuida en 4 núcleos)
Memoria RAM	910 M
Tiempo de ejecución	Valor registrado en 31 min de ejecución de script

Figura 69

Consumo de recursos computacionales de Raspberry Pi con cámara Tapo C320WS

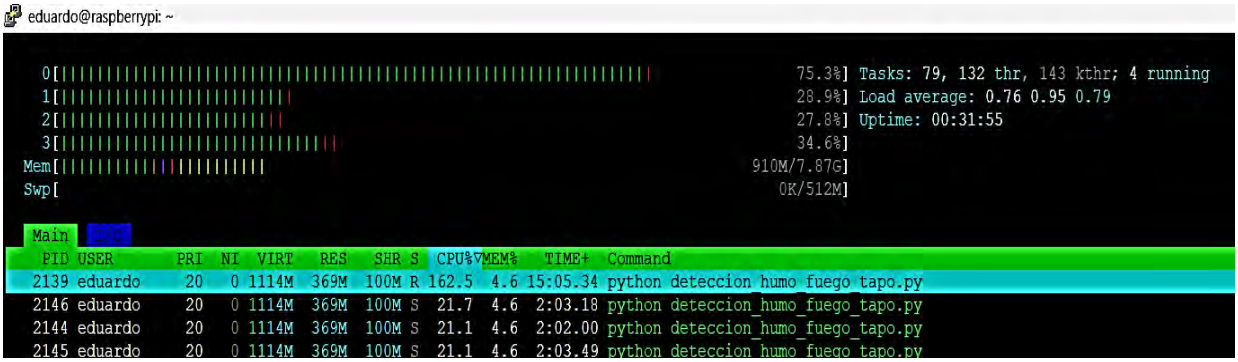
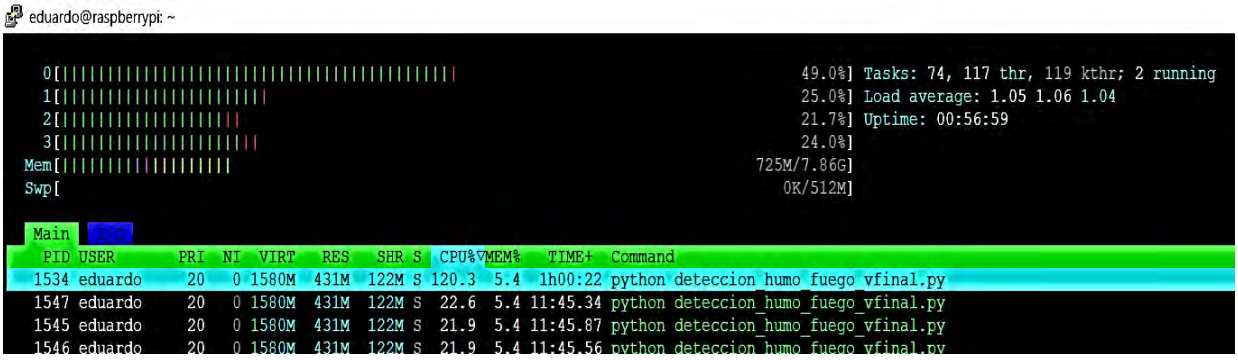


Figura 70

Consumo de recursos computacionales de Raspberry Pi con cámara Raspberry Pi V3 (wide)



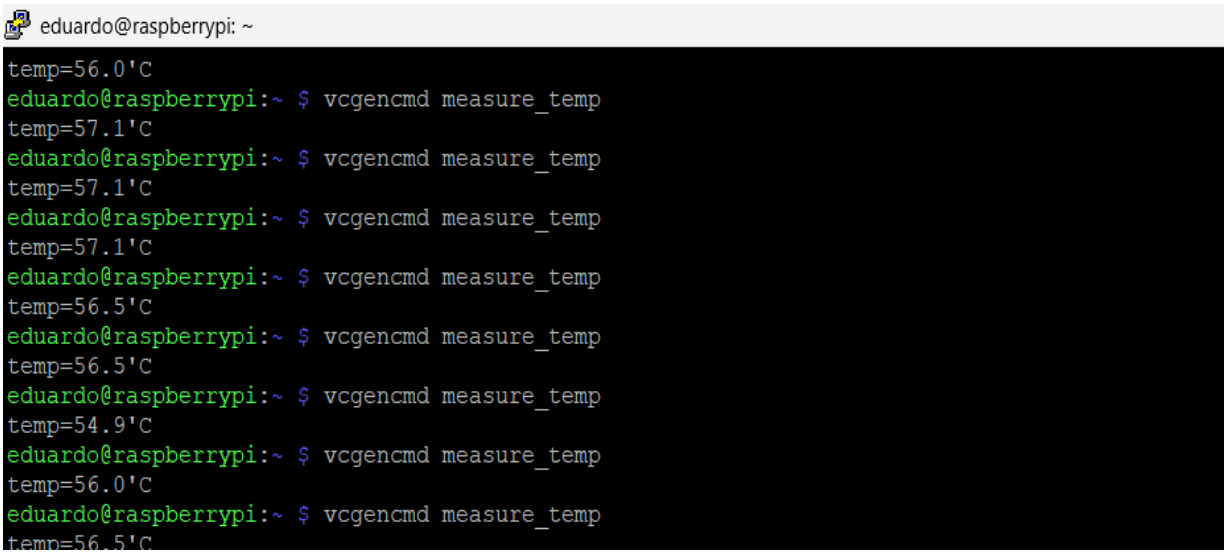
5.8.3. Temperatura de CPU

El monitoreo de la temperatura del CPU es importante ya que este parámetro sirve para verificar la estabilidad térmica del sistema durante los periodos de pruebas y también sirve para justificar la decisión de utilizar una refrigeración en el sistema, mediante el uso de un Active Cooler y el ventilador para extraer el aire caliente del sistema.

La temperatura del CPU de la Raspberry Pi 5 fue en promedio 55 º C usando el segundo modelo de cámara y se muestra en la siguiente imagen.

Figura 71

Temperatura de CPU del Raspberry Pi



Capítulo VI: Costos de Implementación del Sistema

El presente capitulo tiene por finalidad realizar el cálculo de los costos necesarios para realizar la implementación del prototipo del sistema de detección de incendios forestales.

6.1. Costo de Hardware

6.1.1. Costos de Hardware Adquirido

En la presente sección se detalla los gastos realizados en la adquisición de dispositivos esenciales para el desarrollo del proyecto de ingeniería.

Tabla 38

Costos de Hardware Adquirido

Ítem	cantidad	Costo unitario	Sub total
Raspberry Pi 5 (8Gb)	01	S/. 579.00	S/. 579.00
Active Cooler para Raspberry Pi	01	S/. 29.90	S/. 29.90
Fuente Raspberry Pi 27 Watts	01	S/. 67.00	S/. 67.00
Cámara Raspberry Pi V3 (wide)	01	S/. 207.00	S/. 207.00
Cámara IP Tapo C320WS	01	S/. 199.00	S/. 199.00
Ventilador 12v	01	S/. 18.00	S/. 18.00
Fuente 12 voltios (2 A)	01	S/. 16.90	S/. 16.90
Total			**S/. 1 116.80**

6.1.2. Costos de Depreciación de Equipos Propios

Durante la etapa de implementación y pruebas del sistema se utilizó equipos propios como: laptop, kit solar y modem de internet, para valorar el uso de estos recursos se presenta la depreciación del equipo.

Tabla 39

Costos de depreciación de equipos utilizados

Ítem	Precio nuevo	Vida útil	Costo mensual	Total, por 5 meses
Laptop ASUS TUF Gaming F15	S/. 3099.00	36	S/. 86.08	S/. 430.40
Power Station LIDIMI 300 Watts	S/. 525.00	36	S/. 14.58	S/. 72.90
Modem 4G LTE ZTE	S/. 129.90	36	S/. 3.60	S/. 18.00
Total				**S/. 521.30**

6.1.3. Costos de Materiales de Implementación

Para la implementación del sistema se requirió materiales para el armado del prototipo los cuales se presentan en la siguiente tabla.

Tabla 40

Costos de materiales de implementación

Ítem	cantidad	Costo unitario	Sub total
Caja hermética PVC	01 unid.	S/. 32.90.	S/. 32.90
Tomacorriente triple universal	01 unid.	S/. 32.00	S/. 32.00
6m cable vulcanizado exterior 3 x12 AWG	01 unid.	S/. 72.00	S/. 72.00
Enchufe 2p + T	01 unid.	S/. 15.00	S/. 15.00
Trípode telescopio móvil para cámara.	01 unid.	S/. 120.00	S/. 120.00
Total			**S/. 271.90**

6.1.4. Costo Total de Hardware y Materiales de Implementación

A continuación, se muestra el costo total de materiales físicos del sistema.

Tabla 41

Costo total de hardware y materiales de implementación

Descripción	**Costo**
Costo hardware adquirido	S/. 1 116.80
Costo depreciación de equipos propios	S/. 521.30
Costo de materiales de implementación	S/. 271.90
Costo total	**S/. 1 910.00**

6.2. Costo de Software

Para el desarrollo del sistema no se realizó gastos en temas de licencias o compras de software de terceros ya que las herramientas seleccionadas son de código abierto como Python, OpenCV y demás librerías.

Tabla 42

Costo de software

Ítem	**Detalle**	**Costo**
Software de desarrollo	Python, OpenCV, librerías.	S/. 0.00
Plataforma mensajería	Bot Telegram	S/. 0.00
Total		**S/. 0.00**

6.3. Costo de Implementación y Pruebas

6.3.1. Costos de Mano de Obra

El desarrollo del proyecto de ingeniería fue desarrollado con la finalidad de la obtención del título profesional. Como tesista, se realizó el proyecto desde la etapa inicial que incluye la investigación, diseño, implementación de software y hardware hasta la etapa final que es las pruebas del sistema. Si bien no se recibió remuneración económica, se ha considerado realizar un análisis de costo horas hombre por el tiempo invertido, de esta forma se estima el costo real del proyecto. Para realizar los costos en total fueron 6 meses desde la etapa inicial de investigación hasta la etapa final de pruebas del sistema. Se estima 4 horas de trabajo en 5 días a la semana por el periodo de los 6 meses.

Tabla 43

Costos de mano de obra

Rol	**Actividades realizadas**	**Horas estimadas**	**Tarifa (por hora)**	**Sub total**
Tesista	Diseño, implementación de software y hardware y pruebas.	480 horas	S/. 15.00	S/. 7 200.00
Apoyo	Asistencia en pruebas de campo	40 horas	S/. 10.00	S/. 400.00
Total				**S/. 7 600.00**

6.3.2. Costos de Pruebas de Campo

Para la ejecución de pruebas se realizaron gastos en temas de movilidad, paquetes de internet, consumibles para simulación de eventos de humo y fuego. A continuación, se detalla los gastos realizados.

Tabla 44

Costos de pruebas de campo

Ítem	cantidad	Costo unitario	Sub total
Bengalas de Humo	27 unid	S/. 10.00	S/. 270.00
Servicio de internet móvil 4G LTE	4 meses	S/. 30.00	S/. 120.00
Movilidad 02 personas	20 días	S/. 20.00	S/. 400.00
Total			**S/. 790.00**

6.3.3. Costo Total de Implementación y Pruebas

A continuación, se detalla los costos totales de implementación y los gastos realizados en las pruebas.

Tabla 45

Costo total de implementación y pruebas

Descripción	Costo
Costos mano de obra	S/. 7 600.00
Costos de Pruebas de Campo	S/. 790.00
Costo total	**S/. 8 390.00**

6.4. Resumen de Costos Totales

A continuación, se muestra el resumen de los costos totales en hardware, software, implementación y pruebas.

Tabla 46

Resumen de costos totales

Descripción	**Costo**
Costos de hardware	**S/. 1 910.00**
Costos de software	**S/. 0.00**
Costo de implementación y pruebas	**S/. 8 390.00**
Costo total	**S/. 10 300.00**

Conclusiones

- El sistema fue desarrollado y puesto en marcha, cumpliendo con la finalidad de detectar eventos simulados de incendios, se evidenció su factibilidad técnica y operativa en zonas cercanas a los entornos forestales del sector de Picol Orccompucyo. El prototipo modular permite que el sistema tenga futuras ampliaciones como la integración de nuevas tecnologías.
- Se ha diseñado un sistema de detección de incendios forestales mediante el uso de una cámara estacionaria y utilizando técnicas de procesamiento de imágenes, cumpliendo los requerimientos definidos en los alcances como: monitoreo continuo, detección de humo y/o fuego y envío de notificaciones automáticas. Para el diseño se tomó en cuenta la factibilidad de funcionamiento en la comunidad campesina de Picol Orccompucyo, a través de uso de hardware y herramientas computacionales accesibles.
- Se logró implementar el prototipo del sistema de detección de incendios forestales en el sector de Picol Orccompucyo, para ello se utilizó el sistema embebido Raspberry Pi 5 con memoria de 8 GB. Para la captura de imágenes se evaluó dos tipos de cámaras: la Raspberry Pi Cam V3 (Wide) y la cámara IP Tapo C320WS. El prototipo fue montado sobre una estructura portátil lo que permitió su fácil transporte e instalación en la zona de pruebas.
- Se desarrolló el algoritmo capaz de realizar detecciones de humo en escenas diurnas y fuego en escenarios nocturnos. La codificación del algoritmo se realizó en Python, aprovechando su compatibilidad con librerías especializada en visión computacional como OpenCV. El algoritmo principal está compuesto por sub - algoritmos destacando entre ellos la detección de movimiento utilizando KNN-subtraction para luego seguir por segmentación de humo o fuego mediante espacio de color HSV y finalizar con algoritmos de persistencia para la confirmación de eventos.

- En las pruebas realizadas, se pudo evaluar la eficiencia del sistema detector de incendios forestales en dos condiciones diferentes:

  - La primera prueba fue en condiciones simuladas obteniendo métricas de desempeño elevadas (precisión de 91.6%, sensibilidad de 95.6% y un balance general de 93.3%), comparables con los antecedentes de investigación estudiados.

  - En el entorno real de aplicación utilizando el sistema embebido y la cámara Raspberry Pi Cam V3 se obtuvo un desempeño aceptable, precisión de 73.9%, sensibilidad de 85% y un balance general de 79%, esta reducción del rendimiento se debió a factores ambientales y el desempeño de la cámara en condiciones de baja iluminación.

  - Usando la segunda cámara se obtuvo un mejor desempeño con una precisión de 76.9%, sensibilidad de 100% y un balance general de 86.9%. Estos resultados muestran un buen rendimiento generando una base para futuras mejoras y aplicaciones en otros entornos de la región.

Recomendaciones

- Basándose en las pruebas realizadas en campo en un entorno real, se recomienda mejorar la estabilidad de la estructura de montaje, especialmente de la cámara, ante la ocurrencia de vientos intensos. con la finalidad de reducir los desenfoques de cámara que generaban falsas detecciones.
- Dado que el análisis del cielo y nubes fue un factor que generó gran parte de falsos positivos, se recomienda segmentar la escena y realizar un análisis solo a zonas terrestres donde la probabilidad de ocurrencia de incendios forestales es mucho mayor.
- Se recomienda realizar futuras pruebas en distancias más amplias, de esta forma se replica las condiciones de los videos de prueba iniciales y validar el rendimiento a mayor alcance. Como referencia, en Anexo 06 se presenta registros de detecciones obtenidas desde una ubicación urbana lejana. Estas detecciones no formaron parte de la planificación inicial, pero se adjunta como evidencia para futuras aplicaciones.
- Para garantizar un monitoreo continuo las (24/7) durante los meses más propensos de ocurrencia de estos eventos y teniendo en cuenta las mediciones efectuadas al consumo eléctrico del prototipo (~17 W), se recomienda diseñar e implementar un sistema de alimentación ininterrumpido mediante energía solar.
- Se recomienda utilizar los módulos HAT 4G de Raspberry Pi, para asegurar la conexión 4G LTE dedicada, garantizando la conexión a internet.
- Con el objetivo de alcanzar unos mejores resultados de precisión, se propone llevar el proyecto a una segunda fase que incorpore Inteligencia Artificial. De esta forma convirtiendo al trabajo en una base para una solución más avanzada y personalizada para la región.

Recomendaciones para un Futuro Despliegue de Red de Cámaras

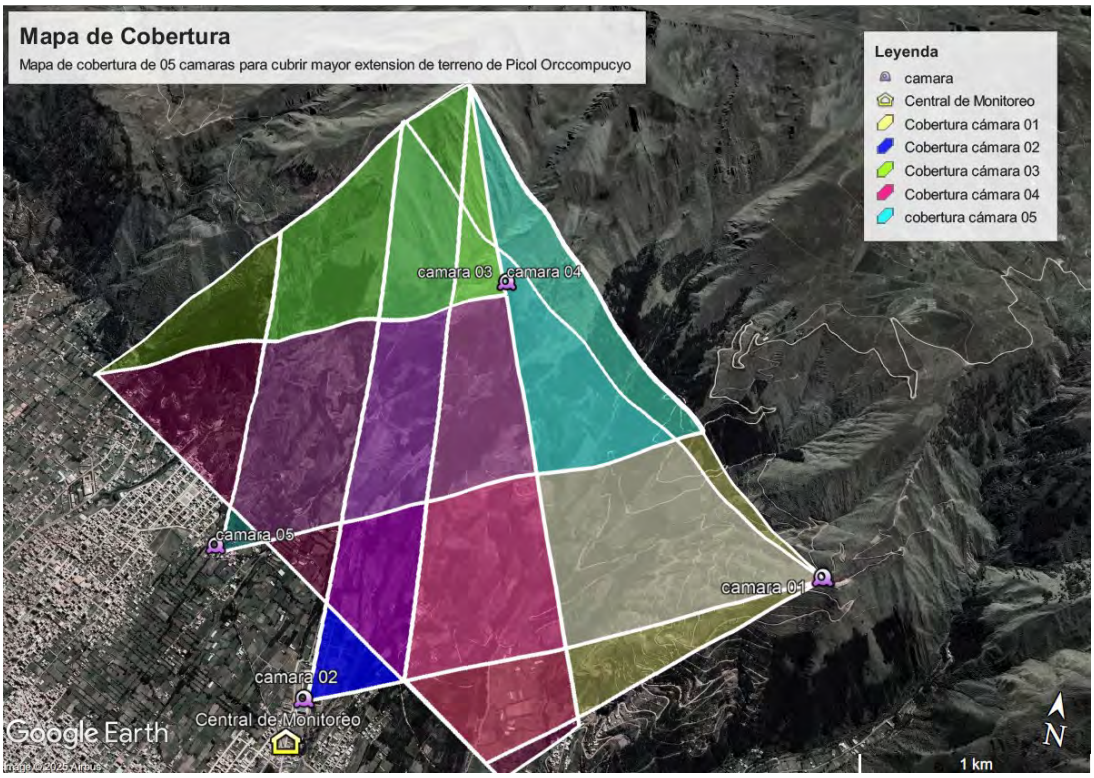
Propuesta de Ubicaciones para una Red de Cámaras Futuras

Como parte de la visión de un sistema escalable bajo una metodología top-down, se realizó un análisis preliminar de la comunidad campesina de Picol Orccompucyo para identificar posibles ubicaciones estratégicas para futuras cámaras. Basándose en los datos históricos y los mapas de pendientes, se ha observado que los incendios forestales tienen origen antrópico y tienden a generarse en áreas de transición entre agrícolas y forestales.

En un proyecto a mayor escala, se propone que las cámaras sean ubicadas en zonas estratégicas para dar mayor cobertura a zonas vulnerables tal como se muestra en el siguiente análisis mostrado en la figura 72.

Figura 72

propuesta de ubicaciones de cámaras en Picol Orccompucyo



Nota: diseño realizado en Google Earth Pro.

Los nodos seleccionados fueron elegidos en base a la geomorfología del sitio (puntos altos) y la facilidad de acceso ya que existen vías de acceso a dichos puntos que facilitarían la implementación y futuros mantenimientos y en caso de los cámaras ubicados en las zonas bajas por que existe infraestructura donde instalar soportes o postes donde ubicar las cámaras, tales como el salón comunal de Picol Orccompucyo y del sector de Machu Picol

Los polígonos de colores representan los campos de visión aproximados de cada cámara y la superposición de las áreas serviría para:

- Reducir los puntos ciegos y de esta forma garantizar cubrir la mayor cantidad de zonas forestales de la comunidad campesina de Picol Orccompucyo.
- Permitir una triangulación de información, de esta manera haciendo un análisis más exhaustivo. Por ejemplo, enviando los datos a un servidor central que correlacione las alertas de múltiples cámaras para confirmar la existencia del incendio.

Estudio de Cobertura Celular en los Nodos Seleccionados

Para realizar un despliegue y basándose en el funcionamiento del prototipo se debe considerar el medio de comunicación y teniendo en cuenta que el prototipo se conectó a internet a través de un modem 4G LTE. Entonces en base al estudio realizado en la parte de diseño se optaría por utilizar la red de Bitel, Movistar y Entel los cuales tienen mayor cobertura en las zonas más altas.

Propuesta de Arquitectura del Sistema

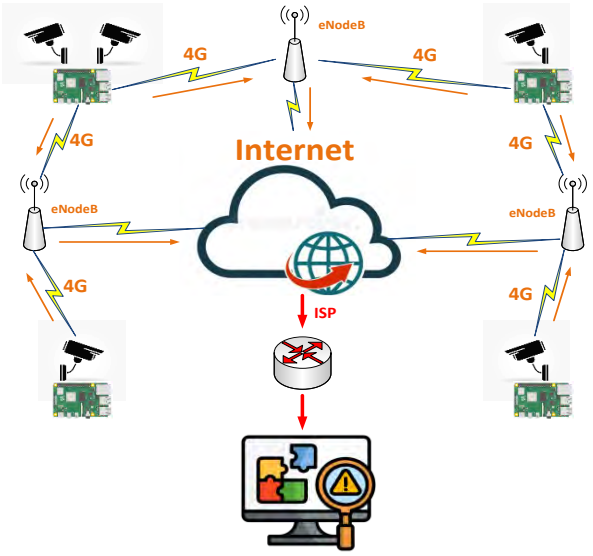
- **Primera arquitectura planteada basado en un servidor físico**

Considerando que cada nodo tendrá conexión a internet se plantea utilizar una arquitectura “Hub and Spoke” (centro y periferia) esto quiere decir todos los nodos están conectados a un servidor central. Cada nodo (Raspberry con cámara y modem 4G) funcionara de manera autónoma

comunicándose directamente al servidor físico, donde también se podría realizar un centro de monitoreo en el cual se pueda visualizar las alertas e imágenes. Adicionalmente esta arquitectura otorga mayor robustez al sistema ya que si un nodo falla no se compromete la funcionalidad de todo el sistema

Figura 73

Arquitectura basada en un servidor físico.



El fujo operativo de la arquitectura propuesta consta de tres principales fases los cuales se describen a continuación:

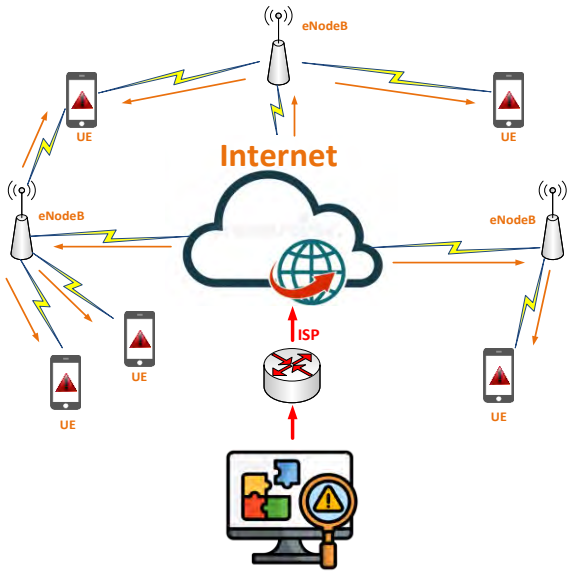
- **Procesamiento en los nodos**: Los nodos de captura y procesamiento, tendrán el mismo principio de funcionamiento del prototipo diseñado, con la única diferencia que ahora las notificaciones serán enviadas a la dirección IP pública estática del centro de monitoreo a través de solicitudes HTTPS POST, el router de la central se encargara de dirigir todas las notificaciones a la IP Privada del servidor de análisis.
- **Análisis centralizado:** Una PC física en un centro de monitoreo el cual trabajará también como un servidor, recibe las alertas y hace el trabajo de "cerebro" y visualización. En el

servidor es donde se realizará la lógica de triangulación de información, mediante un análisis espacio - temporal. La alerta se puede catalogar como primordial en caso dos o más cámaras mandan una alerta de áreas en intersección, caso contrario se le considera una alerta simple. Esta metodología reduciría drásticamente los falsos positivos y priorizaría las alertas de mayor relevancia.

- **Notificación y entrega**: Una vez validado la alerta en el servidor, se deberá integrar con la API de Telegram tal como se realizó en el prototipo con la única diferencia que ahora se integraran múltiples dispositivos. Esto se logrará mediante la agregación de CHAT_ID o identificadores únicos, garantizando que las alertas sean entregadas al personal responsable. En la figura 74 se muestra el proceso descrito en el presente punto.

Figura 74

Proceso de notificación y entrega de alertas primera arquitectura.



- **Segunda arquitectura planteada basado en un servidor agregado en la nube.**

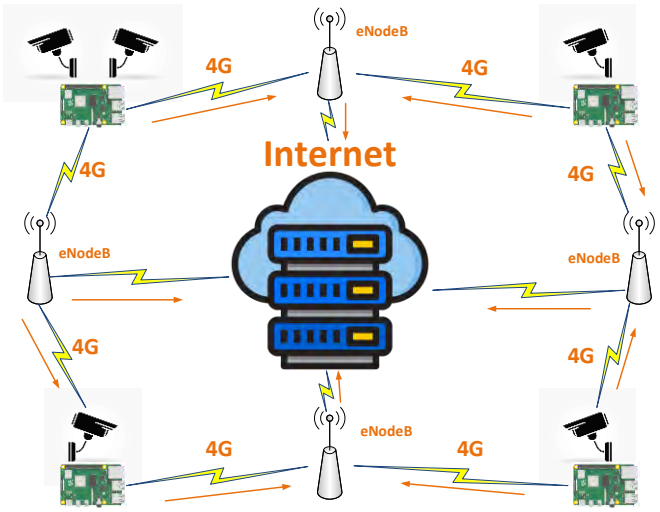
Esta arquitectura permitirá escalabilidad al sistema en caso de que se requiera adicionar más cámaras y en lugar de que cada cámara envié las detecciones a un servidor físico cada nodo

(Raspberry con cámara y modem 4G) funcionará de manera autónoma comunicándose directamente al servidor agregado en la nube.

En la figura 75 se muestra la arquitectura propuesta en donde se tiene los cuatro nodos conectados al servidor de agregación central mediante conectividad celular 4G.

Figura 75

Arquitectura basada en un servidor en la nube.



El fujo operativo de la arquitectura propuesta consta de tres principales fases los cuales se describen a continuación:

- **Procesamiento en los nodos**: Los nodos de captura y procesamiento, tendrán el mismo principio de funcionamiento del prototipo diseñado, con la única diferencia que ahora las notificaciones serán enviadas al servidor de agregación en la nube, mediante solicitudes HTTPS POST. Este método garantiza una comunicación segura estandarizada en un formato JSON.
- **Agregación y Análisis centralizado:** El servidor de agregación, el cual puede ser un servidor en la nube como las que ofrece AWS EC2, Google Cloud Platform entre otras. Este servidor virtual deberá disponer de un sistema operativo virtual como Ubuntu Server,

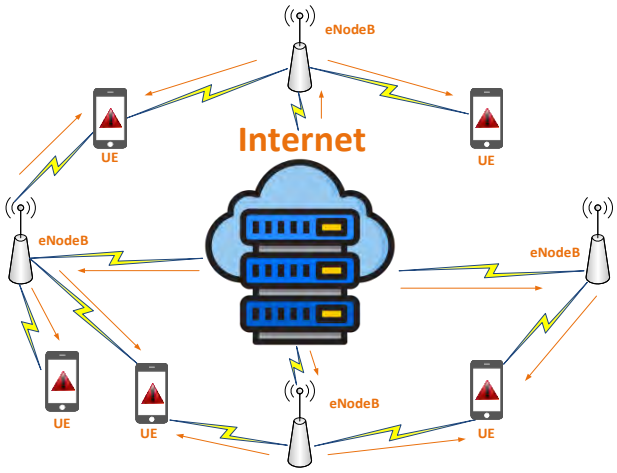
donde se ejecutará un software basado en Python y librerías necesarias para la interacción con la API de Telegram.

En el servidor es donde se realizará la lógica de triangulación de información, mediante un análisis espacio - temporal. La alerta se puede catalogar como primordial en caso dos o más cámaras mandan una alerta de áreas en intersección, caso contrario se le considera una alerta simple. Esta metodología reduciría drásticamente los falsos positivos y priorizaría las alertas de mayor relevancia.

- **Notificación y entrega**: Una vez validado la alerta en el servidor de agregación, se deberá integrar con la API de Telegram tal como se realizó en el prototipo con la única diferencia que ahora se integraran múltiples dispositivos. Esto se logrará mediante la agregación de CHAT_ID o identificadores únicos, garantizando que las alertas sean entregadas al personal responsable. En la figura 76 se muestra el proceso descrito en el presente punto.

Figura 76

Proceso de notificación y entrega de alertas segunda arquitectura.



Referencias

Amazon Web Services. (s.f.). ¿Qué es un bot? - Explicación sobre los tipos de bots - AWS. Obtenido de Amazon Web Services, Inc.: https://aws.amazon.com/es/what-is/bot/#:~:text=Un%20bot%20es%20una%20aplicaci%C3%B3n,forma%20independiente%20sin%20intervenci%C3%B3n%20humana.

Abellan, M. A. (2019). Led RGB del robot mBot con mBlock. Obtenido de Sum, Asociación Programo Ergo: https://raw.githubusercontent.com/ProgramoErgoSum/Tutoriales/master/v1/led-rgb-del-robot-mbot/img/colores-rgb.jpg

Administrator. (s.f.). Image and video databases. Obtenido de Welcome to the Wildfire Observers and Smoke Recognition Homepage: http://wildfire.fesb.hr/index.php?option=com_content&view=article&id=62:image-and-video-databases&catid=35:articles&Itemid=72

Agencia Peruana de Noticias. (09 de 19 de 2024). Cusco suma 277 incendios forestales este año y siniestros han arrasado 10,000 hectáreas. Obtenido de Andina.pe: https://andina.pe/agencia/noticia-cusco-suma-277-incendios-forestales-este-ano-y-siniestros-han-arrasado-10000-hectareas-1000873.aspx

Amazon Web Services. (2024). ¿Qué es Python? Recuperado el 03 de 01 de 2025, de Amazon.com: https://aws.amazon.com/es/what-is/python/

Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. Sensors. doi:https://doi.org/10.3390/s20226442

Base de Datos de Pueblos indigenas u Originarios. (s.f.). Mapa interactivo BDPI. Recuperado el 28 de julio de 2025, de https://bdpi.cultura.gob.pe/mapa-interactivo

Bu, F. G. (2019). Intelligent and vision-based fire detection systems: A survey. mage And Vision Computing 91. doi:https://doi.org/10.1016/j.imavis.2019.08.007

CENEPRED. (2021). Escenario de Riesgo por Incendios Forestales de la Region de Cusco. Lima.

Cetin, A. E. (s.f.). Computer Vision based fire detection software. Obtenido de http://signal.ee.bilkent.edu.tr/VisiFire/

CooperAcción. (09 de septiembre de 2024). Los incendios forestales de cada año. Obtenido de CooperAcción: https://cooperaccion.org.pe/opinion/los-incendios-forestales-de-cada-ano/#:~:text=Las%20cifras%20son%20contundentes%20y,cifra%20similar%20el%20a%C3%B1o%202023.

Diario el Sol del Cusco. (17 de agosto de 2024). Incendio forestal de grandes proporciones afecta flora y fauna de cerro Picol - Diario el Sol del Cusco. Obtenido de Diario el Sol del Cusco: https://diarioelsolcusco.pe/2024/08/17/incendio-forestal-de-grandes-proporciones-afecta-flora-y-fauna-de-cerro-picol/

Díaz, R. F. (2015). Identificación de casos potenciales de edema macular diabético en imágenes de fondo de ojo utilizando técnicas de segmentación dinámica y clasificadores basados en redes neuronales. doi:10.13140/RG.2.1.4773.0646

Domènech, D. M. (2012). Sistema de detección de incendios forestales utilizando tecnicas de procesado de imagen. proyecto final de carrera. Universitat Politecnica de Catalunya.

Drone Solution Center. (s.f.). DJI Mavic 3 Enterprise Thermal (3T). Obtenido de https://dronesolution.pe/producto/dji-mavic-3-enterprise-thermal-3t/

EFE. (1 de octubre de 2024). Sudamérica acumula hasta septiembre el mayor número de incendios forestales en 14 años. SWI swissinfo.ch. Obtenido de https://www.swissinfo.ch/

spa/sudam%C3%A9rica-acumula-hasta-septiembre-el-mayor-n%C3%BAmero-de-incendios-forestales-en-14-a%C3%B1os/87654226

Geeksforgeeks. (16 de febrero de 2023). What is an API (Application Programming Interface). Obtenido de GeeksforGeeks: https://www.geeksforgeeks.org/what-is-an-api/

Glantz, M. H. (1988). Impact assessment by analogy: Comparing the impacts of the Ogallala aquifer depletion and CO2-induced climate change. Societal responses to regional climatic change: Forecasting by analogy.

Gonzalez, R. (2008). Digital image processing. Pearson Prentice Hall.

Gonzalez, R. W. (2009). Digital Image Processing using MATLAB. United States of America: Gatesmark Publishing.

H. Adinanta, E. K. (14 de mayo de 2021). Physical distancing monitoring with background subtraction methods. 442, 45-50. doi:10.1109/icramet51080.2020.9298687

Herrera, A. (19 de junio de 2016). Modelos de color (RGB, CMYK, HSV/HSL). Obtenido de https://ahenav.wordpress.com/2014/04/09/modelos-de-color/

Huillca, F. (s.f.). Checa tu señal - OSIPTEL. Recuperado el 24 de 08 de 2025, de OSIPTEL: https://checatusenal.osiptel.gob.pe/

IGP. (10 de agosto de 2023). Nota Cientifica| Estudio revela que el uso de datos satelitales es eficaz para detectar incendios forestales en los Andes peruanos. Obtenido de Gob.pe: https://www.gob.pe/institucion/igp/noticias/816599-nota-cientifica-estudio-revela-que-el-uso-de-datos-satelitales-es-eficaz-para-detectar-incendios-forestales-en-los-andes-peruanos

Inteccon. (24 de enero de 2024). InsightFD3. Insight Robotics, Insight Robotics Products, Inteccon. Obtenido de Inteccon: https://inteccon.com/es/productos/camaras-de-deteccion-de-incendios-forestales-robotics-cats/insightfd3-roboticscats/

IONOS Digital Guide. (2020). Obtenido de Los formatos de imagen más importantes en Internet: https://www.ionos.com/es-us/digitalguide/paginas-web/diseno-web/cuales-son-los-formatos-de-imagen-mas-importantes/

Krzeczkowska, P. (20 de 11 de 2024). SmokeD. Obtenido de Fire Smoke Color Interpretation: What You Need to Know: https://smokedsystem.com/fire-smoke-color-meaning/

Lau, J. (2017). Utilizacion del Índice Meteorologico de Incendios Forestales (Fire Weather Index) en el Departamento de Cusco. (Tesis para obtar el titulo profesional de Ingenieria Meteorologica). Universidad Nacional Agraria la Molina, Lima.

Mamushiane. (2023). 5G Network Slice Resource Overbooking: An Opportunity for Telcos to Boost their Revenue. University of Cape Town (UCT), Cape Town, South Africa.

Manta, M. I. (2017). Contribución al conocimiento de la prevención de los incendios forestales en la sierra peruana. Lima-Perú: Editora Gráfica Vega S.A.C.

MathWorks, I. (s.f.). Procesamiento basado en ROI - MATLAB & Simulink. Recuperado el 05 de 12 de 2025, de https://la.mathworks.com/help/images/roi-based-processing.html

MD San Jeronimo. (Julio de 2022). Plan de Prevencion y reduccion del riesgo de desastres del distrito de San Jeronimo 2022-2026. Obtenido de CENEPRED: https://sigrid.cenepred.gob.pe/sigridv3/documento/15541

Melvin Wever, T. S. (8 de enero de 2019). The Visual Digital turn: using neural networks to study historical images. Digital Scholarship in the Humanities. doi:10.1093/llc/fqy085

Moscovich, F. A., Ivandic, F., & Besold, L. C. (2014). Manual de combate de incendios forestales y manejo de fuego. . Argentina: Ediciones INTA. Obtenido de http://hdl.handle.net/20.500.12123/5780

Mubarak A. I. Mahmoud & Honge Ren, 2. (2018). Forest Fire Detection Using a Rule-Based Image Processing Algorithm and Temporal Variation. Mathematical Problems in Engineering. doi:10.1155/2018/7612487

Multimedia and Vision Laboratory at Universidad del Valle. (20 de mayo de 2013). Descriptores de textura. Obtenido de SlideShare: https://www.scribbr.es/citar/generador/folders/aKyIZECPOc2XvGY7D0fY4/lists/6p8MRqotSU5qKwkcioPYYF/fuentes/2TmzT8KhBBno7jS1sgJ7YX/editar/

Muñoz Trujillo, J. C. (2023). Encriptación de imágenes digitales.

OpenCV. (21 de febrero de 2025). OpenCV - Open Computer Vision Library. Obtenido de OpenCV: https://opencv.org/

Paratronic. (25 de febrero de 2025). Paratronic surveillance feux de forêt ADELIE. Obtenido de Paratronic: https://www.paratronic.com/es/produit/surveillance-feux-de-foret-systeme-adelie

Plaza Estévez, S. R. (2016). API de servicios web orientados a accesibilidad. Obtenido de https://docta.ucm.es/entities/publication/a975df8e-2ff1-4a74-a50b-f1eba62cb50b

Powertec Wireless Technology NZ. (s.f.). Obtenido de Insight Robotics Thermal + Smoke Wildfire Detection System [INSIGHT FD 3]: https://powertec.co.nz/insight-robotics-thermal-smoke-wildfire-detection-system-insight-fd-3/

Raspberry Pi. (03 de enero de 2025). We are Raspberry Pi. We make computers. Obtenido de Raspberry Pi: https://www.raspberrypi.com/about/

Raspberrypi. (25 de marzo de 2024). Raspberrypi. Obtenido de Raspberrypi.com: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Raspberrypi. (s.f.). Camera About the Camera Modules. Obtenido de Raspberrypi documentation: https://www.raspberrypi.com/documentation/accessories/camera.html

SERFOR. (2019). Sistema de Alerta Temprana y Monitoreo de Incendios Forestales. Obtenido de http://repositorio.serfor.gob.pe/handle/SERFOR/727

Servicio Nacional Forestal y de Fauna Silvestre. (2019). Sistema de Alerta Temprana y Monitoreo de Incendios Forestales. Obtenido de http://repositorio.serfor.gob.pe/handle/SERFOR/727

Suarez, M. W. (2021). Mapa de riesgo potencial de incendios forestales en la comunidad campesina Picol Orcompugio - Distrito de San Jerónimo – Cusco – 2020. (Tesis para obtener el titulo profesional de Ingeniera Ambiental). Universidad Cesar vallejo.

Telegram. (s.f.). Bots: An introduction for developers. Recuperado el 31 de 07 de 2025, de https://core.telegram.org/bots

Universidad de Sevilla. (s.f.). INTRODUCCIÓN A LAS IMÁGENES DIGITALES. Recuperado el 24 de mayo de 2025, de https://asignatura.us.es/imagendigital/Tema1.pdf

Valades, B. (05 de octubre de 2024). ¿Qué es una cámara de seguridad IP, cómo funciona y cuáles son sus ventajas? Obtenido de Segurilatam: https://www.segurilatam.com/actualidad/camara-de-seguridad-ip-que-es-como-funciona-y-ventajas_20240510.html

Vizzuality. (s.f.). Obtenido de Global Deforestation Rates & Statistics by Country GFW: https://www.globalforestwatch.org/dashboards/global/

Weather Spark. (s.f.). Cusco Climate, weather by month, Average temperature (Peru) - Weather Spark. Obtenido de Weather Spark: https://s.weatherspark.com/y/25926/Clima-promedio-en-Cuzco-Per%C3%BA-durante-todo-el-a%C3%B1o

Zoran Zivkovic, F. v. (07 de enero de 2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. 27. doi:10.1016/j.patrec.2005.11.005

Anexos

Anexo 01: Código Cálculo de Métricas Desempeño KNN Background Subtractor

```
import cv2
import numpy as np
ruta_video = 'Malacka2012-1-002.avi'
tiempo_muestreo = ['0:50', '1:54', '6:08']
combinaciones = [
    {'history': 1000, 'dist2Threshold': 200.0, 'detectShadows': False},
    {'history': 1000, 'dist2Threshold': 400.0, 'detectShadows': False},
    {'history': 1000, 'dist2Threshold': 700.0, 'detectShadows': False},]
tiempo_segundos = []
try:
    for tiempo_1 in tiempo_muestreo:
        minutos,segundos = map(int, tiempo_1.split(':'))
        total_segundos = minutos * 60 + segundos
        tiempo_segundos.append(total_segundos)
except ValueError:
    print("Error: El formato de tiempo no válido. Usa 'minutos:segundos'.")
    exit()
for params in combinaciones:
    cap = cv2.VideoCapture(video_file)
    if not cap.isOpened():
        print("Error: No se pudo abrir el archivo de video.")
        break
    substraccion_KNN = cv2.createBackgroundSubtractorKNN(
        history=params['history'],
        dist2Threshold=params['dist2Threshold'],
        detectShadows=params['detectShadows'])
    captured = np.zeros(len(tiempo_segundos), dtype=bool)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        video_ms = cap.get(cv2.CAP_PROP_POS_MSEC)
        video_seg = video_ms / 1000.0
        fg_mask = substraccion_KNN.apply(frame)
        for i, tiempo_objetivo in enumerate(tiempo_segundos):
            if video_seg >= tiempo_objetivo and not captured[i]:
                parametros_1 = f"h{params['history']}_d{int(params['dist2Threshold'])}"
                tiempo_1 = tiempo_muestreo[i].replace(":", "m")
                mascara = f'mascara_{tiempo_1}s_{parametros_1}.png'
                frame_real = f'fotograma_{tiempo_1}s_{parametros_1}.png'
                cv2.imwrite(mascara, fg_mask)
                cv2.imwrite(frame_real, frame)
                captured[i] = True
        if np.all(captured):
            break
    cap.release()
cv2.destroyAllWindows()
```

```
import cv2
import numpy as np
import os
máscara_verdad = 'fotograma_6m08s_h1000_d400.png'
carpeta_mascaras_KNN = '6_08'
masc_verd_gris = cv2.imread(máscara_verdad, cv2.IMREAD_GRAYSCALE)
if masc_verd_gris is None:
    print(f"Error: No se pudo cargar la máscara de verdad fundamental")
    exit()
_,masc_verd_binario=cv2.threshold(masc_verd_gris,0,255,cv2.THRESH_BINARY/cv2.THRESH_OTSU)
masc_verd_bool = masc_verd_binario.astype(bool)
results = []

for filename in os.listdir(carpeta_mascaras_KNN):
    if filename.endswith(".png"):
        mascaras_knn = os.path.join(carpeta_mascaras_KNN, filename)
        mascaras_knn_gris = cv2.imread(mascaras_knn, cv2.IMREAD_GRAYSCALE)
        if mascaras_knn_gris is not None:
            _, mascaras_knn_binario = cv2.threshold(mascaras_knn_gris, 0, 255,
               cv2.THRESH_BINARY / cv2.THRESH_OTSU)
            mascaras_knn_bool = mascaras_knn_binario.astype(bool)
            if masc_verd_bool.shape == mascaras_knn_bool.shape:
                TP = np.logical_and(masc_verd_bool, mascaras_knn_bool).sum()
                FP = mascaras_knn_bool.sum() - TP
                FN = masc_verd_bool.sum() - TP
                union = TP + FP + FN
                iou_score = TP / union if union != 0 else 0.0
                results.append((filename, TP, FP, FN, iou_score))
            else:
                print("dimensiones incorrectas")
# --- Mostrar los resultados en formato de tabla ---
print("\n" + "="*80)
print("      Análisis de Métricas      ".center(80))
print("="*80)
print(f"{'Archivo':<20}{'TP':<10}{'FP':<10}{'FN':<10}{'IoU Score':<15}")
print("-"*80)
for filename, tp, fp, fn, iou_score in sorted(results, key=lambda x: x[4], reverse=True):
    print(f"{filename:<20}{tp:<10}{fp:<10}{fn:<10}{iou_score:<15.4f}")
print("="*80)
```

Primer video analizado

Escenario 1: 2m38s (Inicio de Humo)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d700	363	6946	1590	0.0408	0.050	0.186	0.080
h1000_d400	568	13588	1385	0.0365	0.040	0.291	0.069
h1000_d200	761	26558	1192	0.0267	0.028	0.389	0.053
h500_d700	198	6062	1755	0.0247	0.032	0.101	0.048
h500_d400	352	12550	1601	0.0243	0.027	0.180	0.046
h500_d200	577	24905	1376	0.0215	0.023	0.295	0.042
h200_d200	272	14683	1681	0.0164	0.018	0.139	0.031
h200_d400	105	6541	1848	0.0124	0.016	0.054	0.024
h200_d700	22	2718	1931	0.0047	0.008	0.011	0.009

Escenario 2 : 8m04s (Humo Cargado)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h500_d200	2499	3689	3034	0.2710	0.404	0.451	0.426
h1000_d200	2445	3511	3088	0.2703	0.410	0.441	0.425
h1000_d400	1717	1074	3816	0.2599	0.615	0.310	0.412
h500_d400	1692	1224	3841	0.2504	0.580	0.306	0.401
h1000_d700	1110	320	4423	0.1896	0.776	0.200	0.319
h500_d700	986	368	4547	0.1671	0.728	0.178	0.286
h200_d200	1716	5027	3817	0.1625	0.254	0.310	0.280
h200_d400	1137	1934	4396	0.1523	0.370	0.206	0.264
h200_d700	741	743	4792	0.1181	0.499	0.134	0.212

Escenario 3: 1m15s (Sin Movimiento)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	0	1930	0	0.0000	0.00	0.00	0.00
h1000_d400	0	421	0	0.0000	0.00	0.00	0.00
h1000_d700	0	89	0	0.0000	0.00	0.00	0.00
h200_d200	0	373	0	0.0000	0.00	0.00	0.00
h200_d400	0	51	0	0.0000	0.00	0.00	0.00
h200_d700	0	12	0	0.0000	0.00	0.00	0.00
h500_d200	0	995	0	0.0000	0.00	0.00	0.00
h500_d400	0	180	0	0.0000	0.00	0.00	0.00
h500_d700	0	33	0	0.0000	0.00	0.00	0.00

Segundo video analizado

Escenario 1: 3m20s (Inicio de Humo)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d400	36	439	279	0.0477	0.076	0.114	0.091
h1000_d700	16	126	299	0.0363	0.113	0.051	0.070
h500_d200	43	1001	272	0.0327	0.041	0.137	0.063
h500_d400	19	315	296	0.0302	0.057	0.060	0.058
h1000_d200	50	1366	265	0.0297	0.035	0.159	0.058
h200_d400	17	308	298	0.0273	0.052	0.054	0.053
h500_d700	9	108	306	0.0213	0.077	0.029	0.042
h200_d200	24	916	291	0.0195	0.026	0.076	0.039
h200_d700	8	96	307	0.0195	0.077	0.025	0.038

Escenario 2: 0m27s (Humo Cargado)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	110	9213	1028	0.0106	0.012	0.097	0.021
h1000_d400	67	5215	1071	0.0105	0.013	0.059	0.022
h1000_d700	43	3044	1095	0.0103	0.014	0.038	0.021
h500_d200	72	9707	1066	0.0066	0.007	0.063	0.013
h500_d400	37	5839	1101	0.0053	0.006	0.032	0.011
h500_d700	18	3536	1120	0.0039	0.005	0.016	0.008
h200_d200	38	####	1100	0.0031	0.003	0.033	0.006
h200_d400	16	7258	1122	0.0019	0.002	0.014	0.004
h200_d700	7	4717	1131	0.0012	0.001	0.006	0.002

Escenario 3: 4m32s (Sin Movimiento)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	0	1238	0	0.0000	0.00	0.00	0.00
h1000_d400	0	291	0	0.0000	0.00	0.00	0.00
h1000_d700	0	77	0	0.0000	0.00	0.00	0.00
h200_d200	0	107	0	0.0000	0.00	0.00	0.00
h200_d400	0	11	0	0.0000	0.00	0.00	0.00
h200_d700	0	2	0	0.0000	0.00	0.00	0.00
h500_d200	0	266	0	0.0000	0.00	0.00	0.00
h500_d400	0	36	0	0.0000	0.00	0.00	0.00
h500_d700	0	6	0	0.0000	0.00	0.00	0.00

Tercer video analizado

Escenario 1: 0m45s (Inicio de Humo)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	20	285	108	0.0484	0.066	0.156	0.093
h500_d200	5	270	123	0.0126	0.018	0.039	0.025
h1000_d400	4	210	124	0.0118	0.019	0.031	0.024
h500_d400	2	197	126	0.0062	0.010	0.016	0.012
h1000_d700	1	164	127	0.0034	0.006	0.008	0.007
h200_d200	0	266	128	0.0000	0.00	0.00	0.00
h200_d400	0	195	128	0.0000	0.00	0.00	0.00
h200_d700	0	159	128	0.0000	0.00	0.00	0.00
h500_d700	0	148	128	0.0000	0.00	0.00	0.00

Escenario 2: 1m52s (Humo Cargado)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	941	354	430	0.5455	0.726	0.686	0.706
h1000_d400	812	134	559	0.5395	0.858	0.592	0.700
h1000_d700	715	53	656	0.5021	0.931	0.521	0.666
h500_d200	740	321	631	0.4374	0.697	0.540	0.609
h500_d400	600	135	771	0.3984	0.816	0.438	0.570
h500_d700	475	64	896	0.3310	0.881	0.346	0.496
h200_d200	416	210	955	0.2631	0.664	0.304	0.416
h200_d400	266	79	1105	0.1834	0.771	0.194	0.310
h200_d700	151	24	1220	0.1082	0.863	0.110	0.195

Escenario 3: 5m24s (Sin Humo)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	0	94	0	0.0000	0.00	0.00	0.00
h1000_d400	0	48	0	0.0000	0.00	0.00	0.00
h1000_d700	0	27	0	0.0000	0.00	0.00	0.00
h200_d200	0	84	0	0.0000	0.00	0.00	0.00
h200_d400	0	41	0	0.0000	0.00	0.00	0.00
h200_d700	0	23	0	0.0000	0.00	0.00	0.00
h500_d200	0	91	0	0.0000	0.00	0.00	0.00
h500_d400	0	58	0	0.0000	0.00	0.00	0.00
h500_d700	0	25	0	0.0000	0.00	0.00	0.00

Cuarto video analizado

Escenario 1: 0m50s (Humo Cargado)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d400	82	42	107	0.3550	0.661	0.434	0.522
h1000_d200	104	119	85	0.3377	0.466	0.550	0.504
h1000_d700	64	24	125	0.3005	0.727	0.339	0.463
h500_d200	74	74	115	0.2814	0.500	0.391	0.440
h500_d400	53	24	136	0.2488	0.688	0.280	0.398
h500_d700	38	8	151	0.1929	0.826	0.201	0.324
h200_d200	40	34	149	0.1794	0.541	0.212	0.305
h200_d400	24	8	165	0.1218	0.750	0.127	0.218
h200_d700	18	1	171	0.0947	0.947	0.095	0.173

Escenario 2: 1m54s (Humo Cargado)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	408	93	363	0.4722	0.814	0.529	0.641
h1000_d400	302	38	469	0.3733	0.888	0.392	0.543
h500_d200	299	77	472	0.3526	0.795	0.388	0.521
h1000_d700	234	27	537	0.2932	0.897	0.304	0.453
h500_d400	197	40	574	0.2429	0.831	0.256	0.391
h200_d200	155	51	616	0.1886	0.752	0.201	0.317
h500_d700	120	23	651	0.1511	0.839	0.155	0.262
h200_d400	89	16	682	0.1131	0.847	0.116	0.204
h200_d700	45	6	726	0.0579	0.882	0.058	0.109

Escenario 3: 6m08s (Sin Humo)

Parámetros	TP	FP	FN	IoU Score	Precisión	Sensibilidad	F1-Score
h1000_d200	0	144	0	0.0000	0.00	0.00	0.00
h1000_d400	0	14	0	0.0000	0.00	0.00	0.00
h1000_d700	0	5	0	0.0000	0.00	0.00	0.00
h200_d200	0	26	0	0.0000	0.00	0.00	0.00
h200_d400	0	2	0	0.0000	0.00	0.00	0.00
h200_d700	0	0	0	0.0000	0.00	0.00	0.00
h500_d200	0	54	0	0.0000	0.00	0.00	0.00
h500_d400	0	4	0	0.0000	0.00	0.00	0.00
h500_d700	0	1	0	0.0000	0.00	0.00	0.00

Anexo 02: Código Cálculo de Parámetros HSV Para Humo y Fuego

```
import cv2 #importar librería open cv
import numpy as np #importar librería Numpy
img = cv2.imread('prueba1.jpg') #importar imagen
# creación de función barras deslizantes
def nothing(x):
pass
cv2.namedWindow('PDI11') #creación de ventana llamada PDI11
cv2.createTrackbar('Hmin','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255
cv2.createTrackbar('Hmax','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255
cv2.createTrackbar('Smin','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255
cv2.createTrackbar('Smax','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255
cv2.createTrackbar('Vmin','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255
cv2.createTrackbar('Vmax','PDI11',0,255,nothing) #barras deslizantes en PDI11 de 0-255

while True:
hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV) #conversion de escala de color RGB a HSV
Hmin = cv2.getTrackbarPos('Hmin','PDI11') #obtener valor de barra deslizante
Hmax = cv2.getTrackbarPos('Hmax','PDI11') #obtener valor de barra deslizante
Smin = cv2.getTrackbarPos('Smin','PDI11') #obtener valor de barra deslizante
Smax = cv2.getTrackbarPos('Smax','PDI11') #obtener valor de barra deslizante
Vmin = cv2.getTrackbarPos('Vmin','PDI11') #obtener valor de barra deslizante
Vmax = cv2.getTrackbarPos('Vmax','PDI11') #obtener valor de barra deslizante
lower = np.array([Hmin,Smin,Vmin]) #variable lower con valores bajos de HSV
upper = np.array([Hmax,Smax,Vmax]) #variable upper con valores altos de HSV
mask = cv2.inRange(hsv,lower,upper) #mascara con rango de valores HSV
mask_colored = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR) #mascara de escala gris a color RGB
# Aplicar la máscara sobre la imagen original
resultado = cv2.bitwise_and(img, mask_colored)
cv2.imshow('mask',mask) #mostrar mascara HSV
cv2.imshow('real',img) #mostrar imagen original
cv2.imshow('Resultado', resultado) #mostrar resultado de operación AND
k= cv2.waitKey(1)
if k==27:
break
cv2.destroyAllWindows() #cerrar todas las ventanas activas
```

Anexo 03: Código Sistema Detector de Incendios Forestales Usando Raspberry Pi Cam V3

```
# Importar librerias necesarias para implementacion
import cv2
import numpy as np
import requests
import os
import datetime
from picamera2 import Picamera2
# Configuración para la API de Telegram
bot_token = 'bot Token personal'
chat_id = 'chat_id personal'
# funcion para enviar mensaje via telegram
def enviar_mensaje_telegram(mensaje):
    """Envía un mensaje de texto a Telegram."""
    url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
    params = {"chat_id": chat_id, "text": mensaje}
    try:
        requests.get(url, params=params, timeout=5)
    except requests.RequestException as e:
        print(f"Error al enviar el mensaje: {e}")
# funcion para enviar imagen via telegram
def enviar_imagen_telegram(ruta_imagen):
    """Envía una imagen a Telegram."""
    if not os.path.exists(ruta_imagen):
        print("La imagen no existe.")
        return
    url = f"https://api.telegram.org/bot{bot_token}/sendPhoto"
    with open(ruta_imagen, "rb") as file:
    files = {"photo": file}
    try:
        requests.post(url, files=files, data={"chat_id": chat_id}, timeout=5)
    except requests.RequestException as e:
        print(f"Error al enviar la imagen: {e}")
# Configuracion de camara Raspberry v3
picam2 = Picamera2()
config = picam2.create_preview_configuration(main={"format": "RGB888", "size":(960,540)})
config["controls"]={
    "FrameRate":30,
    "NoiseReductionMode":2,
    "AeEnable": True,
    "AeMeteringMode":2,
    "ExposureValue":0,
    "Contrast":1,
     }
```

```
# aplicacion de configuraciones
picam2.configure(config)
# Inicializacion de camara
picam2.start()
# verificacion de estado de camara
if not picam2:
    print("Error al abrir el video.")
    exit()
# puntos de recorte para extraccion de ROI
pts = np.array(
    [[0, 220], [44,210], [89,209], [126,201], [140,201],[161,199],[182,199],
    [194,203],[202,203],[220,214],[245,226],[268,235],[284,235],[294,238],
    [301,237],[309,231],[314,229],[321,225],[332,224],[343,229],[347,229],
    [367,236],[393,232],[436,242],[462,247],[514,246],[520,244],[533,244],
    [540,247],[573,247],[633,234],[684,232],[691,228],[735,228],[745,223],
    [767,230],[807,219],[864,215],[896,205],[938,209],[947,210],[960,204],
    [960,370],[0,370]], np.int32)
pts = pts.reshape((-1, 1, 2))
x1 = 0
y1 = 197
w1 = 960
h1 = 200
# Configuración del sustractor de fondo y rangos HSV
knn_subtractor = cv2.createBackgroundSubtractorKNN(
    history=900, dist2Threshold= 700, detectShadows=False)
#Rango HSV para humo iluminacion baja en escena
#lower_hsv1 = np.array([61, 10, 120])
#upper_hsv1 = np.array([133, 55, 215])
# Rango HSV para humo buena iluminacion en escena
lower_hsv1 = np.array([61, 10, 180])
upper_hsv1 = np.array([133, 55, 240]
# Rango HSV para fuego
lower_hsv2 = np.array([0, 5, 220])
upper_hsv2 = np.array([40, 255, 255])
# Parámetros de detección
NUM_FRAMES_ANALISIS = 300  #equivalente a 10 seg.
movimiento_frames_hsv1 = 0
movimiento_frames_hsv2 = 0
mascaras_acumuladas1 = []
mascaras_acumuladas2 = []
while True:
    # captura de frame
    frame_a = picam2.capture_array("main")
    # cambio de resolucion
    frame_a = cv2.resize(frame_a, (960, 540))
    # extraccion del ROI para su analisis
    mask = np.zeros(frame_a.shape[:2], dtype=np.uint8)
    cv2.fillPoly(mask, [pts], 255)
    roi = cv2.bitwise_and(frame_a, frame_a, mask=mask)
```

```
    # creacion de mascara de deteccion de movimento
    fg_mask = knn_subtractor.apply(roi)
    # Filtrado de ruido medinate suavisado y morfologico
    fg_mask = cv2.medianBlur(fg_mask, 3)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
    fg_mask = cv2.dilate(fg_mask,kernel, iterations=2)
    # si en mascara hay movimientos grandes inicia analisis por color
    if np.sum(fg_mask) > 2000:
        # conversion de Roi a Rango de color HSV
        hsv_image = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
        # creacion de mascara donde los pixeles que cumplan con rango HVS de humo en primer plano
        mask_hsv1 = cv2.inRange(hsv_image, lower_hsv1, upper_hsv1)
        # aplicacion de filtros de suavizado y morfologico a mascara de color humo
        mask_hsv1 = cv2.medianBlur(mask_hsv1, 3)
        mask_hsv1 = cv2.erode(mask_hsv1,kernel, iterations=1)
        mask_hsv1 = cv2.dilate(mask_hsv1,kernel, iterations=2)
        # creacion de mascara donde los pixeles que cumplan con rango HVS de fuego en primer plano
        mask_hsv2 = cv2.inRange(hsv_image, lower_hsv2, upper_hsv2)
        # aplicacion de filtros de suavizado y morfologico a mascara de color fuego
        mask_hsv2 = cv2.medianBlur(mask_hsv2, 3)
        mask_hsv2 = cv2.erode(mask_hsv2,kernel, iterations=1)
        mask_hsv2 = cv2.dilate(mask_hsv2,kernel, iterations=2)
        # interseccion de mascara de movimiento y mascara de color HSV humo
        interseccion_hsv_movimiento1=cv2.bitwise_and(fg_mask, mask_hsv1)
        interseccion_hsv_movimiento1=cv2.dilate(interseccion_hsv_movimiento1,kernel,iterations=2)
        # interseccion de mascara de movimiento y mascara de color HSV fuego
        interseccion_hsv_movimiento2=cv2.bitwise_and(fg_mask, mask_hsv2)
        interseccion_hsv_movimiento2=cv2.dilate(interseccion_hsv_movimiento2,kernel,iterations=2)
        # si mascara de humo resultante es mayor a 500 px
        if np.sum(interseccion_hsv_movimiento1) > 500:
            # inicia conteo de frames y guarda mascara de interseccion humo
            movimiento_frames_hsv1 += 1
            mascaras_acumuladas1.append(interseccion_hsv_movimiento1.copy())
        else:
            movimiento_frames_hsv1 = 0
            mascaras_acumuladas1 = []
        # si mascara de fuego resultante es mayor a 500 px inicia conteo de frames
        if np.sum(interseccion_hsv_movimiento2) > 500:
            # inicia conteo de frames y guarda mascara de interseccion fuego
            movimiento_frames_hsv2 += 1
            mascaras_acumuladas2.append(interseccion_hsv_movimiento2.copy())
        else:
            movimiento_frames_hsv2 = 0
            mascaras_acumuladas2 = []
        resultado_filtrado1=np.zeros_like(mascaras_acumuladas1[0],dtype=np.uint8) if
mascaras_acumuladas1 else None
        resultado_filtrado2 = np.zeros_like(mascaras_acumuladas2[0], dtype=np.uint8) if
mascaras_acumuladas2 else None
```

```
        condicion_humo = movimiento_frames_hsv1 >= NUM_FRAMES_ANALISIS and
len(mascaras_acumuladas1) >= NUM_FRAMES_ANALISIS
        condicion_fuego = movimiento_frames_hsv2 >= NUM_FRAMES_ANALISIS and
len(mascaras_acumuladas2) >= NUM_FRAMES_ANALISIS
        # si cantidad de frames de interseccion de humo o fuego es igual a NUM_FRAMES_ANALISIS
        if condicion_humo or condicion_fuego:
            # analisis persistencia humo
            if movimiento_frames_hsv1>=NUM_FRAMES_ANALISIS and len(mascaras_acumuladas1)>=
NUM_FRAMES_ANALISIS:
                # crear mascara con ceros con el tamaño de mascaras analisadas
                suma_mascaras1 = np.zeros_like(mascaras_acumuladas1[0], dtype=np.uint16)
                # suma veces que estuvo activo pixel a lo largo de todos los frames de analisis
                for m in mascaras_acumuladas1:
                    suma_mascaras1 += (m > 0).astype(np.uint16)
                    # pixeles que estuvieron activos en un 70% o mas en frames de analisis
                    umbral_presencia = int(NUM_FRAMES_ANALISIS*0.7)
                    resultado_filtrado1=np.where(suma_mascaras1>=
umbral_presencia,255,0).astype(np.uint8)
                    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
                    resultado_filtrado1 = cv2.dilate(resultado_filtrado1,kernel, iterations=1)
            # analisis persistencia fuego
            if movimiento_frames_hsv2 >= NUM_FRAMES_ANALISIS and len(mascaras_acumuladas2)>=
NUM_FRAMES_ANALISIS:
                # crear mascara con ceros con tamaño de mascaras analisadas
                suma_mascaras2 = np.zeros_like(mascaras_acumuladas2[0], dtype=np.uint16)
                # suma veces que estuvo activo un pixel a lo largo de todos los frames de analisis
                for m in mascaras_acumuladas2:
                    suma_mascaras2 += (m > 0).astype(np.uint16)
                    # pixeles que estuvieron activos en un 70% o mas en  frames de analisis
                    umbral_presencia = int(NUM_FRAMES_ANALISIS*0.7)
                    resultado_filtrado2 = np.where(suma_mascaras2 >=
umbral_presencia,255,0).astype(np.uint8)
                    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
                    resultado_filtrado2 = cv2.dilate(resultado_filtrado2,kernel, iterations=1)
            # segmentar areas detectadas
            for mask, etiqueta, color in [(
resultado_filtrado1 if 'resultado_filtrado1' in locals() else None,"HUMO",(255,0,0)),
(resultado_filtrado2 if 'resultado_filtrado2' in locals() else None,"FUEGO",(0,255,0))]:
                # deteccion de contornos
                if mask is not None:
                    contornos, _=cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
                    for c in contornos:
                        if cv2.contourArea(c) > 5:
                            # coordenadas del rectangulo sobre los contornos
                            x,y,w,h = cv2.boundingRect(c)
                            # dibujar el rectangulo sobre la imagen original
                            cv2.rectangle(frame_a, (x-20,y-20), (x+w+20,y+h+20), color,2)
                            # colocar la etiqueta sobre la imagen original
                            cv2.putText(frame_a,etiqueta,(x-15,y-25),
cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
```

```
            # lineas indicadoras del ROI
            cv2.rectangle(frame_a, (x1,y1), (x1+w1,y1+h1), (255,0,255),1)
            # fecha y hora de deteccion
            cv2.putText(frame_a,datetime.datetime.now().strftime("%d-%m-%Y %H:%M:%S"),
(2,15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0,204,255),1)
            # Guardar imagen y enviar alerta
            ruta_imagen = "deteccion_final.jpg"
            cv2.imwrite(ruta_imagen, frame_a)
            enviar_imagen_telegram(ruta_imagen)
            enviar_mensaje_telegram(f"!!!!ALERTA rpi!!!!")
            movimiento_frames_hsv1 = 0
            movimiento_frames_hsv2 = 0
            mascaras_acumuladas1 = []
            mascaras_acumuladas2 = []
    # linea para cancelar ejecucion de codigo
    if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cv2.destroyAllWindows()
picam2.stop()
```

Anexo 04: Código Sistema Detector de Incendios Forestales Usando Cámara TapoC320WS

```
# Importar librerias necesarias para implementacion
import cv2 # importar opencv
import numpy as np # importar libreria numpy
import requests #importamos la libreria request para utilizar API telegram
import os # gestionar archivos
import datetime # importamos libreria para leer hora y fecha en tiempo real
# Configuración para la API de Telegram
bot_token = 'bot token personal'
chat_id = 'chat_id personal'
# funcion para enviar mensaje via telegram
def enviar_mensaje_telegram(mensaje):
    url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
    params = {"chat_id": chat_id, "text": mensaje}
    try:
        requests.get(url, params=params, timeout=5)
    except requests.RequestException as e:
        print(f"Error al enviar el mensaje: {e}")
# funcion para enviar imagen via telegram
def enviar_imagen_telegram(ruta_imagen):
    if not os.path.exists(ruta_imagen):
        print("La imagen no existe.")
    return
    url = f"https://api.telegram.org/bot{bot_token}/sendPhoto"
    with open(ruta_imagen, "rb") as file:
    files = {"photo": file}
    try:
        requests.post(url, files=files, data={"chat_id": chat_id}, timeout=5)
    except requests.RequestException as e:
    print(f"Error al enviar la imagen: {e}")
# Configuración del sustractor de fondo
knn_subtractor = cv2.createBackgroundSubtractorKNN(
    history=900, dist2Threshold=700, detectShadows=False)
# Rango HSV para humo buena iluminacion
lower_hsv1 = np.array([42, 5, 160])
upper_hsv1 = np.array([142, 63, 205])
# Rango HSV para humo baja iluminacion o nublado
# lower_hsv1 = np.array([42, 5, 125])
# upper_hsv1 = np.array([142, 63, 205])
# Rango HSV para fuego
lower_hsv2 = np.array([0, 3, 235])
upper_hsv2 = np.array([55, 255, 255])
# Parámetros de detección
f_analisis = 150 #equivalente a 10 seg.
```

```
mov_f_hsv1 = 0
mov_f_hsv2 = 0
mascaras_acumuladas1 = []
mascaras_acumuladas2 = []
# iniciamos con la lectura de camara y/o stream de video
url =                Link Rstp://
# puntos de recorte para extraccion de ROI
pts = np.array(
[[0, 312], [19,302], [56,292], [80,289], [90,288],[119,278],[127,276],
[138,272],[195,273],[172,269],[183,274],[187,274],[224,296],[249,304],
[258,309],[298,311],[309,300],[321,293],[334,289],[352,292],[352,296],
[386,305],[423,297],[443,298],[489,306],[538,309],[601,305],[621,299],
[642,302],[671,301],[767,274],[827,266],[836,260],[887,252],[910,256],
[938,240],[960,241],[960,479],[0,479]], np.int32)
pts = pts.reshape((-1, 1, 2))
x1, y1, w1, h1 = 0, 265, 960, 216
# obtener flujo de video
cap = cv2.VideoCapture(url)
# verificacion de estado de camara
if not cap.isOpened():
    print("Error al abrir el video.")
    exit()
while True:
    # captura de frame
    ret, frame = cap.read()
    if not ret:
        break
    # Reducir resolución
    frame = cv2.resize(frame, (960, 540))
    # extraccion del ROI para su analisis
    mask = np.zeros(frame.shape[:2], dtype=np.uint8)
    cv2.fillPoly(mask, [pts], 255)
    roi = cv2.bitwise_and(frame, frame, mask=mask)
    # Aplicamos deteccion de movimiento mediante algoritmo knn substractor
    fg_mask = knn_subtractor.apply(roi)
    # Filtrado de ruido medinate suavisado y morfologico
    fg_mask = cv2.medianBlur(fg_mask, 3)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
    fg_mask = cv2.dilate(fg_mask, kernel, iterations=2)
    # si en mascara hay movimientos grandes inicia analisis por color
    if np.sum(fg_mask) > 2000:
        # convertimos cada frame leido de escala RGB a escala de color HSV
        hsv_image = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
        #creamos una mascara donde los pixeles que cumplan con rango HVS de humo
        mask_hsv1 = cv2.inRange(hsv_image, lower_hsv1, upper_hsv1)
        mask_hsv1 = cv2.medianBlur(mask_hsv1, 3)
        mask_hsv1 = cv2.erode(mask_hsv1, kernel, iterations=1)
        mask_hsv1 = cv2.dilate(mask_hsv1, kernel, iterations=2)
```

```
        #creamos una mascara donde los pixeles que cumplan con rango HVS de fuego
        mask_hsv2 = cv2.inRange(hsv_image, lower_hsv2, upper_hsv2)
        mask_hsv2 = cv2.medianBlur(mask_hsv2, 3)
        mask_hsv2 = cv2.erode(mask_hsv2, kernel, iterations=1)
        mask_hsv2 = cv2.dilate(mask_hsv2, kernel, iterations=2)
        #interseccion que cumplan con las dos condiciones de movimiento y color de humo
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
        interseccion_hsv_movimiento1 = cv2.bitwise_and(fg_mask, mask_hsv1)
        interseccion_hsv_movimiento1=cv2.dilate(interseccion_hsv_movimiento1,kernel,iterations=2)
        #interseccion que cumplan con las dos condiciones de movimiento y color de fuego
        interseccion_hsv_movimiento2 = cv2.bitwise_and(fg_mask, mask_hsv2)
        interseccion_hsv_movimiento2=cv2.dilate(interseccion_hsv_movimiento2,kernel,iterations=2)
        #si en la interseccion hay mas de 500 pixeles empieza el conteo de frames
        if np.sum(interseccion_hsv_movimiento1) > 500:
            # inicia conteo de frames y guarda mascara de interseccion humo
            mov_f_hsv1 += 1
            mascaras_acumuladas1.append( interseccion_hsv_movimiento1.copy())
        else:
            mov_f_hsv1 = 0
            mascaras_acumuladas1 = []
        #si en la interseccion hay mas de 700 pixeles empieza el conteo de frames
        if np.sum(interseccion_hsv_movimiento2) > 500:
            # inicia conteo de frames y guarda mascara de interseccion fuego
            mov_f_hsv2 += 1
            mascaras_acumuladas2.append( interseccion_hsv_movimiento2.copy())
        else:
            mov_f_hsv2 = 0
            mascaras_acumuladas2 = []
        resultado_filtrado1 = np.zeros_like(mascaras_acumuladas1[0], dtype=np.uint8) if
mascaras_acumuladas1 else None
        resultado_filtrado2 = np.zeros_like(mascaras_acumuladas2[0], dtype=np.uint8) if
mascaras_acumuladas2 else None
        # si cantidad de frames de interseccion de humo o fuego es igual a NUM_FRAMES_ANALISIS
        if (mov_f_hsv1 >= f_analisis and len(mascaras_acumuladas1) >= f_analisis) or (mov_f_hsv2
>= f_analisis and len(mascaras_acumuladas2) >= f_analisis):
            # analisis persistencia humo
            if mov_f_hsv1 >= f_analisis and len(mascaras_acumuladas1) >= f_analisis:
                # crear mascara con ceros con el tamaño de mascaras analisadas
                suma_mascaras1 = np.zeros_like(mascaras_acumuladas1[0], dtype=np.uint16)
                # sumatoria de veces que estuvo activo un pixel a lo largo de frames de analisis
                for m in mascaras_acumuladas1:
                    suma_mascaras1 += (m > 0).astype(np.uint16)
                    # pixeles que estuvieron activos en un 70% o mas en frames de analisis
                    umbral_presencia = int(f_analisis * 0.7)
                    resultado_filtrado1 = np.where(suma_mascaras1 >= umbral_presencia, 255,
0).astype(np.uint8)
            # analisis persistencia fuego
            if mov_f_hsv2 >= f_analisis and len(mascaras_acumuladas2) >= f_analisis:
                # crear mascara con ceros con tamaño de mascaras analisadas
                suma_mascaras2 = np.zeros_like(mascaras_acumuladas2[0], dtype=np.uint16)
                # sumatoria de veces que estuvo activo un pixel a lo largo de frames de analisis
```

```
                for m in mascaras_acumuladas2:
                    suma_mascaras2 += (m > 0).astype(np.uint16)
                    # pixeles que estuvieron activos en un 70% o mas en frames de analisis
                    umbral_presencia = int(f_analisis * 0.7)
                    resultado_filtrado2 = np.where(suma_mascaras2 >= umbral_presencia, 255,
0).astype(np.uint8)
            # segmentar areas detectadas
            for mask, etiqueta, color in [(
resultado_filtrado1 if 'resultado_filtrado1' in locals() else None, "Humo",(255, 0, 0)),
(resultado_filtrado2 if 'resultado_filtrado2' in locals() else None, "Fuego",(0, 255, 0))]:
                if mask is not None:
                    contornos, _= cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
                    for c in contornos:
                        if cv2.contourArea(c) > 10:
                            # Obtener rectángulo delimitador
                            x, y, w, h = cv2.boundingRect(c)
                            # Dibujar rectángulo con margen
                            cv2.rectangle(frame, (x-30, y-20), (x+w+25, y+h+25),color, 2)
                            # Agregar etiqueta (Humo o Fuego)
                            cv2.putText(frame,etiqueta,(x-20,y-20),cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color,2)
            # Agregar lineas delimitadora de ROI y la fecha y hora actual en la imagen
            cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1), (255,0,255), 1)
            cv2.putText(frame, datetime.datetime.now().strftime("%d-%m-%Y %H:%M:%S"),
(2, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 204, 255), 1)
            # Guardar imagen y enviar alerta
            ruta_imagen = "deteccion_final.jpg"
            cv2.imwrite(ruta_imagen, frame)
            enviar_imagen_telegram(ruta_imagen)
            enviar_mensaje_telegram(f"Advertencia tapo!!!!:")
            mov_f_hsv1 = 0
            mov_f_hsv2 = 0
            mascaras_acumuladas1 = []
            mascaras_acumuladas2 = []
    if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

Anexo 05: Carta de Autorización de la Comunidad Campesina de Picol Orccompucyo

Comunidad Campesina Picol Orccompucyo

San Jerónimo – Cusco

RECONOCIDO POR R.S. N.° 0163 del 19-04-1965

INSCRITO EN SUNARP FICHAS 10309-10310-10311

"Año de la recuperación y consolidación de la economía peruana"

AUTORIZACIÓN PARA REALIZACIÓN DE TESIS

Por medio del presente, yo, **JOSE TISOC FLOREZ**, identificado con DNI N.º **23893796**, en calidad de **PRESIDENTE DE LA COMUNIDAD CAMPESINA DE PICOL ORCCOMPUCYO**, distrito de San Jerónimo, provincia de Cusco, autorizo al estudiante:

EDUARDO ELI CONDORHUAMAN QUISPE, identificado con DNI N.º **70412279,** en condición de bachiller de la carrera profesional de **INGENIERÍA ELECTRÓNICA** de la **UNIVERSIDAD SAN ANTONIO ABAD DEL CUSCO.**

Para que pueda realizar pruebas de campo e implementar actividades relacionadas con su tesis titulada: **"DESARROLLO DE UN SISTEMA DE DETECCIÓN DE INCENDIOS FORESTALES BASADO EN PROCESAMIENTO DIGITAL DE IMÁGENES CON UNA CÁMARA ESTACIONARIA EN EL SECTOR DE PICOL ORCCOMPUCYO DEL DISTRITO DE SAN JERÓNIMO-CUSCO"**, la cual tiene por finalidad el diseño y desarrollo de un sistema detector de incendios forestales basados en técnicas de procesado de imágenes

La autorización fue formalmente acordada el 20 de marzo del presente año tras una reunión previa con los representantes y autoridades de la comunidad campesina de Picol Orccompucyo. Si bien el consentimiento fue otorgado inicialmente de forma verbal, este documento tiene como finalidad constatar y formalizar dicha autorización para los fines del proyecto. Se expide la presente a solicitud del interesado, para los fines que estime conveniente.

San Jerónimo Cusco, 04 de julio de 2025

COMUNIDAD CAMPESINA DE PICOL ORCCOMPUCYO
JOSE TISOC FLOREZ
DNI 23893796
PRESIDENTE

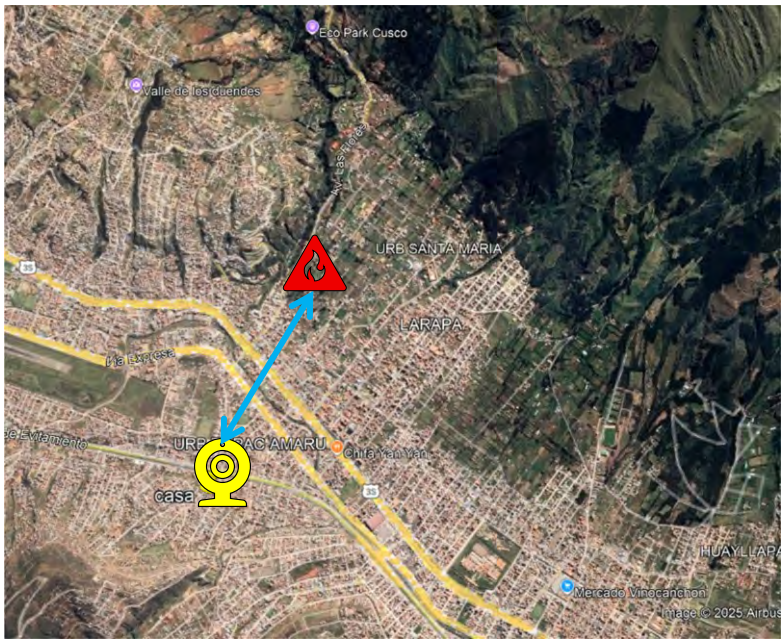
Presidente de comunidad campesina
de Picol Orccompucyo

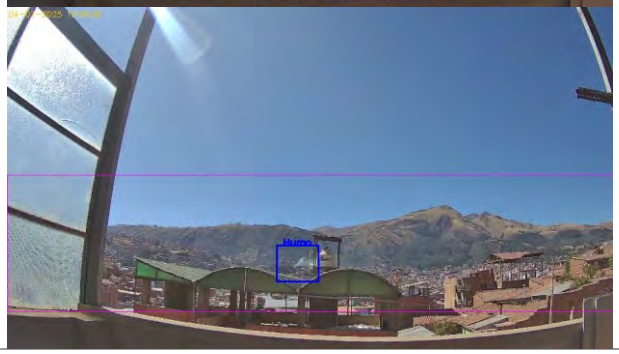
AV. ALMUDENA – CIRCUNVALACIÓN NORTE S/N. LOCAL CENTRAL SALÓN COMUNAL

Anexo 06: Registro de Detecciones Automáticas del Sistema Desde un Entorno Urbano Post – Etapa de Validación

En el anexo presento detecciones automáticas de humo realizadas por el sistema mientras funcionaba de forma constante desde un entorno urbano utilizando la cámara IP Tapo C320WS, apuntando hacia el sector de Picol. Estas detecciones no se incluyeron en el conjunto de pruebas programadas, ni fueron provocadas intencionalmente, pero simbolizan sucesos reales de humo. Se incluye estos registros ilustrativos evidenciando el funcionamiento del sistema en condiciones reales y a largas distancias.

El día 04 de julio de 2025 se detectó un evento de humo a una distancia aproximada de 1.6 Km tal como se muestra en la siguiente figura.

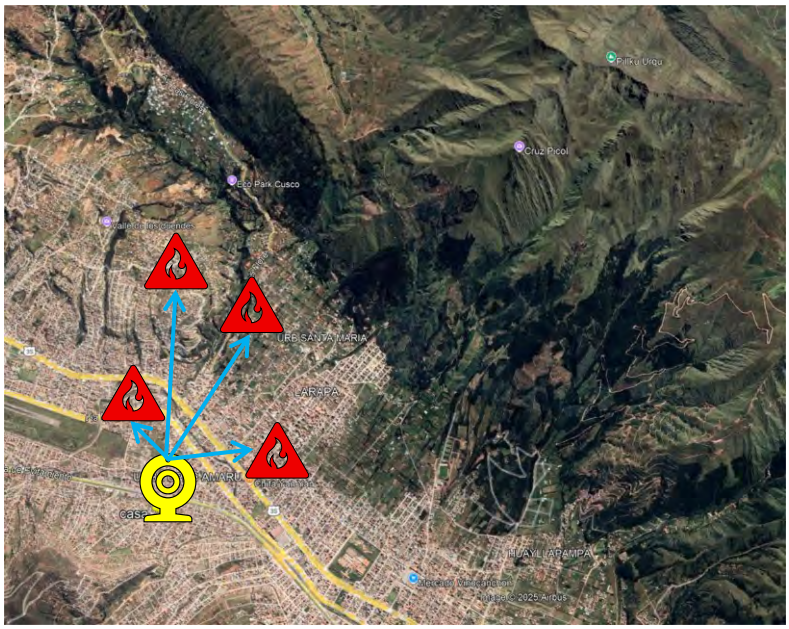


FECHA	VERDADERO POSITIVOS	FALSOS POSITIVO
04-07-2025	   	No se detectó falsos positivos.

El día 07 de julio de 2025 se detectó dos eventos de humo el primero a una distancia aproximada de 1.9 Km y el segundo que fue un evento real de un incendio forestal que se produjo a espaldas del cerro Picol por el sector de Ccorao a una distancia aproximada de 4.9 Km.

FECHA	VERDADERO POSITIVOS	FALSOS POSITIVO
07-07-2025		

El día 10 de julio se detectó múltiples eventos de humo en escena, observando el evento más lejano el producido por el sector de Alto Qosqo a 1.9 Km aproximadamente. A continuación, se muestran las detecciones realizadas por el sistema.



FECHA	VERDADERO POSITIVOS	FALSOS POSITIVO
10-07-2025	   	


Humo
Humo
Humo

Anexo 07: Evidencias Fotográficas del Prototipo y las Pruebas en Campo

Implementación física de dispositivos

Prototipo inicial

Primeras pruebas de campo

Primeras pruebas de campo

Simulación de humo con bengalas.

Picol Orccompucyo

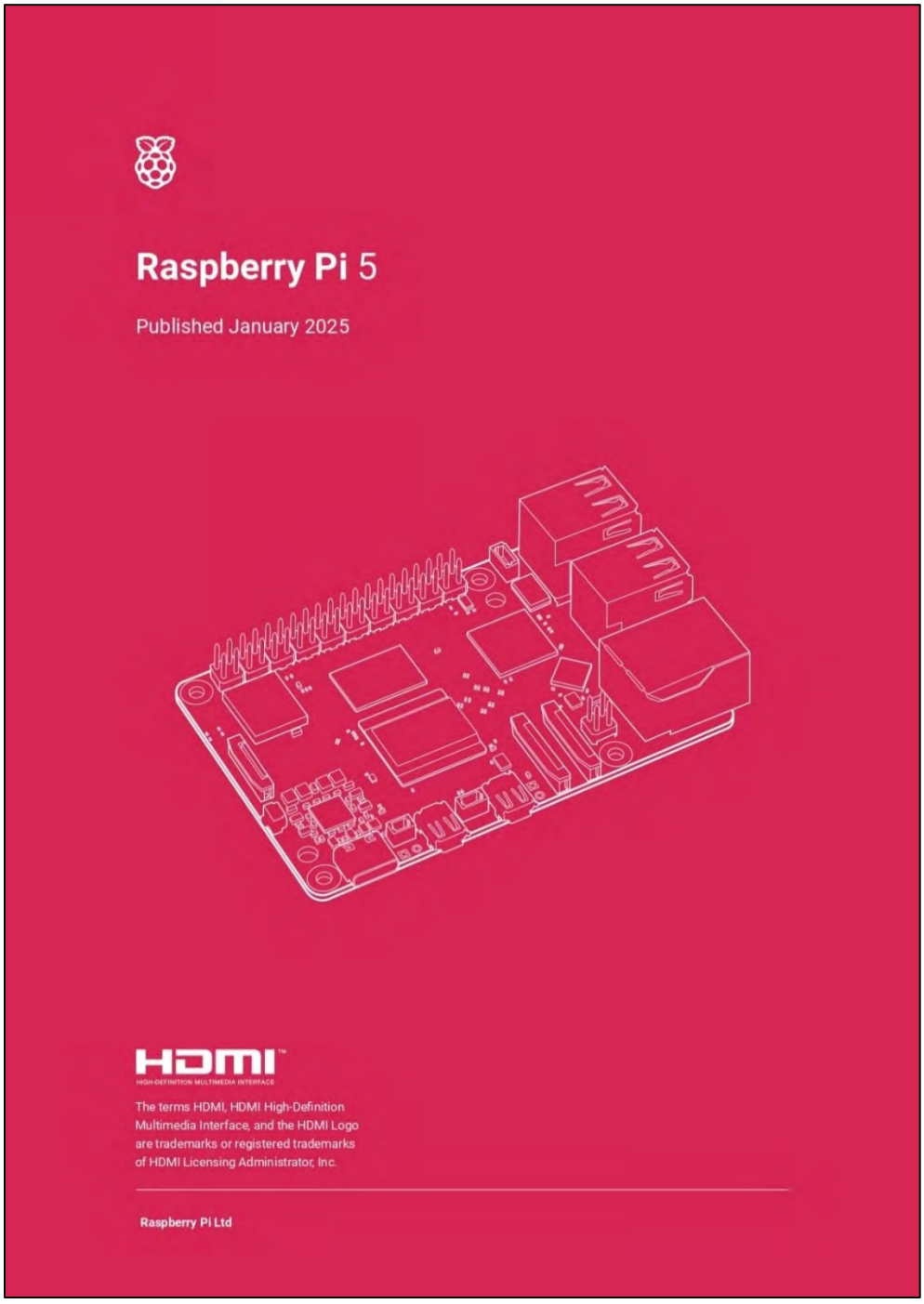
Pruebas detección de fuego

Pruebas detección de fuego

Picol Orccompucyo

Picol Orccompucyo

Anexo 08: Datasheet Raspberry Pi 5



Overview

Welcome to the latest generation of Raspberry Pi: the everything computer.

Featuring a 64-bit quad-core Arm Cortex-A76 processor running at 2.4GHz, Raspberry Pi 5 delivers a 2–3× increase in CPU performance relative to Raspberry Pi 4. Alongside a substantial uplift in graphics performance from an 800MHz VideoCore VII GPU; dual 4Kp60 display output over HDMI; and state-of-the-art camera support from a rearchitected Raspberry Pi Image Signal Processor, it provides a smooth desktop experience for consumers, and opens the door to new applications for industrial customers.

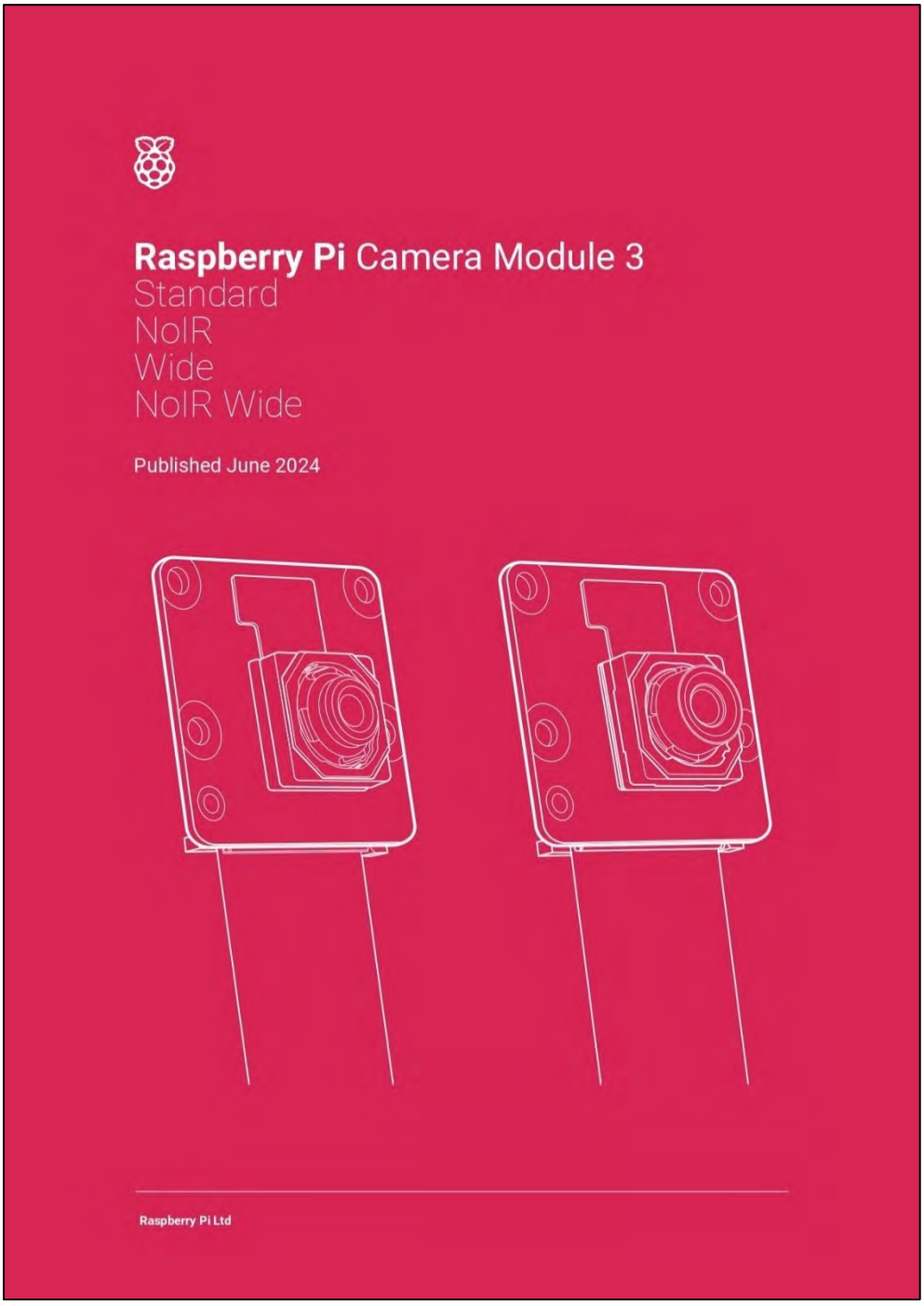
For the first time, this is a full-size Raspberry Pi computer using silicon built in-house at Raspberry Pi. The RP1 "southbridge" provides the bulk of the I/O capabilities for Raspberry Pi 5, and delivers a step change in peripheral performance and functionality. Aggregate USB bandwidth is more than doubled, yielding faster transfer speeds to external UAS drives and other high-speed peripherals; the dedicated two-lane 1Gbps MIPI camera and display interfaces present on earlier models have been replaced by a pair of four-lane 1.5Gbps MIPI transceivers, tripling total bandwidth, and supporting any combination of up to two cameras or displays; peak SD card performance is doubled, through support for the SDR104 high-speed mode; and for the first time the platform exposes a single-lane PCI Express 2.0 interface, providing support for high-bandwidth peripherals.

Specification

Processor	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with Cryptographic Extension, 512KB per-core L2 caches, and a 2MB shared L3 cache
Features:	• VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2 • Dual 4Kp60 HDMI® display output with HDR support • 4Kp60 HEVC decoder • LPDDR4X-4267 SDRAM (options for 2GB, 4GB, 8GB and 16GB) • Dual-band 802.11ac Wi-Fi® • Bluetooth 5.0 / Bluetooth Low Energy (BLE) • microSD card slot, with support for high-speed SDR104 mode • 2 × USB 3.0 ports, supporting simultaneous 5Gbps operation • 2 × USB 2.0 ports • Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT) • 2 × 4-lane MIPI camera/display transceivers • PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter) • 5V/5A DC power via USB-C, with Power Delivery support • Raspberry Pi standard 40-pin header • Real-time clock (RTC), powered from external battery • Power button
MTBF[1] Ground Benign:	93 800 hours
Production lifetime:	Raspberry Pi 5 will remain in production until at least January 2036
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com
List price:	2GB **$50** 4GB **$60** 8GB **$80** 16GB **$120**

[1] Mean Time Between Failure

Anexo 09: Datasheet Raspberry Camera V3



Overview

Raspberry Pi Camera Module 3 is a compact camera from Raspberry Pi. It offers an IMX708 12-megapixel sensor with HDR, and features phase detection autofocus. Camera Module 3 is available in standard and wide-angle variants, both of which are available with or without an infrared cut filter.

Camera Module 3 can be used to take full HD video as well as stills photographs, and features an HDR mode up to 3 megapixels. Its operation is fully supported by the libcamera library, including Camera Module 3's rapid autofocus feature: this makes it easy for beginners to use, while offering plenty for advanced users. Camera Module 3 is compatible with all Raspberry Pi computers.[1]

The PCB size and mounting holes remain the same as for Camera Module 2. The Z dimension differs: due to the improved optics, Camera Module 3 is several millimetres taller than Camera Module 2.

All variants of Camera Module 3 feature:

- Back-illuminated and stacked CMOS 12-megapixel image sensor (Sony IMX708)
- High signal-to-noise ratio (SNR)
- Built-in 2D Dynamic Defect Pixel Correction (DPC)
- Phase Detection Autofocus (PDAF) for rapid autofocus
- QBC Re-mosaic function
- HDR mode (up to 3 megapixel output)
- CSI-2 serial data output
- 2-wire serial communication (supports I2C fast mode and fast-mode plus)
- 2-wire serial control of focus mechanism

[1] Excluding early Raspberry Pi Zero models, which lack the necessary FPC connector. Later Raspberry Pi Zero models require an adapter FPC, sold separately.

Specification

Sensor:	Sony IMX708
Resolution:	11.9 megapixels
Sensor size:	7.4mm sensor diagonal
Pixel size:	1.4µm × 1.4µm
Horizontal/vertical:	4608 × 2592 pixels
Common video modes:	1080p50, 720p100, 480p120
Output:	RAW10
IR cut filter:	Integrated in standard variants; not present in NoIR variants
Autofocus system:	Phase Detection Autofocus
Dimensions:	25 × 24 × 11.5mm (12.4mm height for Wide variants)
Ribbon cable length:	200mm
Cable connector:	15 × 1mm FPC
Operating temperature:	0°C to 50°C
Compliance:	FCC 47 CFR Part 15, Subpart B, Class B Digital Device Electromagnetic Compatibility Directive (EMC) 2014/30/EU Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU
Production lifetime:	Raspberry Pi Camera Module 3 will remain in production until at least January 2030

Variants

	Camera Module 3	Camera Module 3 NoIR	Camera Module 3 Wide	Camera Module 3 Wide NoIR
Focus range	10cm–∞	10cm–∞	5cm–∞	5cm–∞
Focal length	4.74mm	4.74mm	2.75mm	2.75mm
Diagonal field of view	75 degrees	75 degrees	120 degrees	120 degrees
Horizontal field of view	66 degrees	66 degrees	102 degrees	102 degrees
Vertical field of view	41 degrees	41 degrees	67 degrees	67 degrees
Focal ratio (F-stop)	F1.8	F1.8	F2.2	F2.2
Infrared-sensitive	No	Yes	No	Yes

WARNINGS

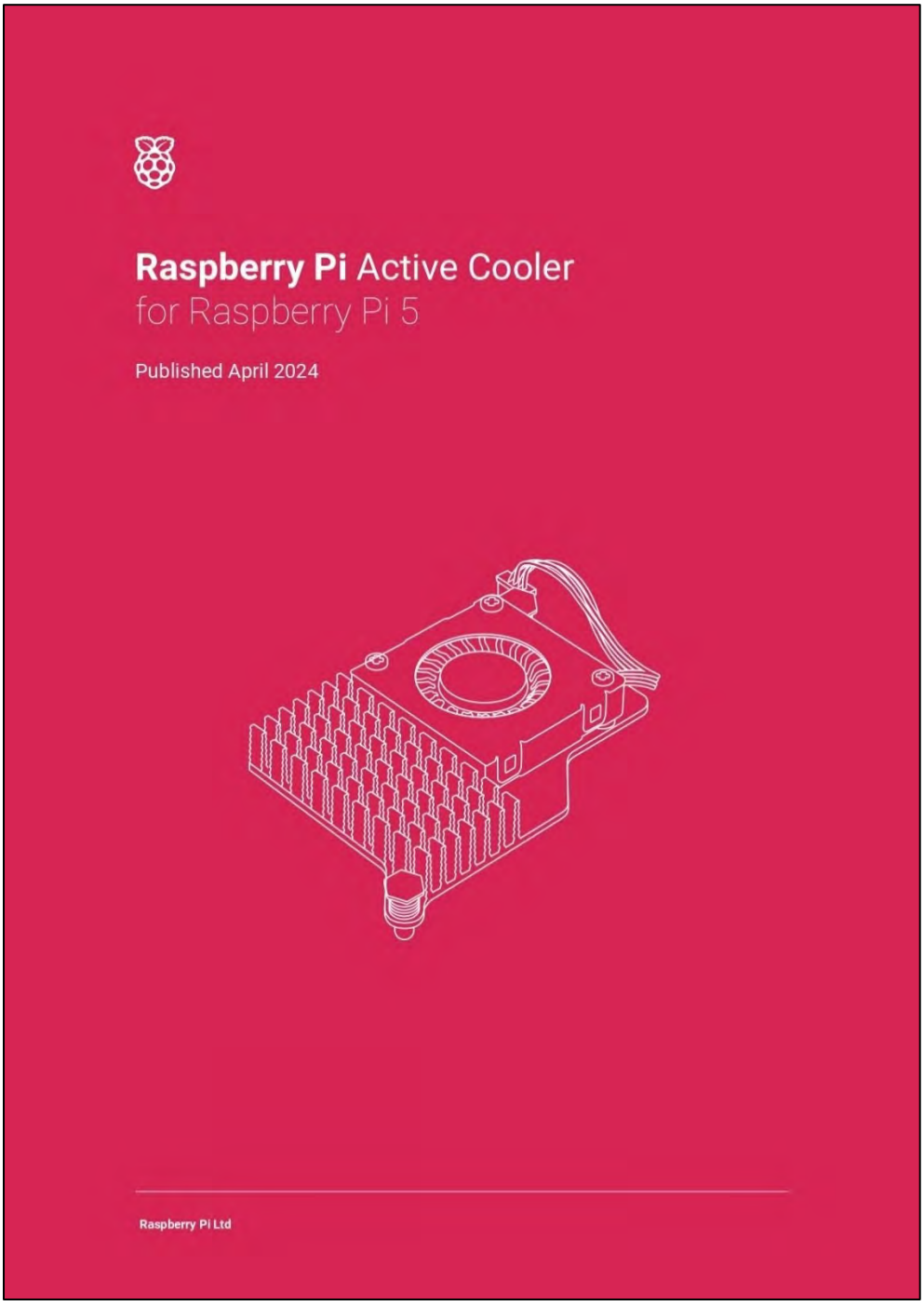
- This product should be operated in a well ventilated environment, and if used inside a case, the case should not be covered.
- Whilst in use, this product should be firmly secured or should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Camera Module 3 may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Important: Before connecting this device, shut down your Raspberry Pi computer and disconnect it from external power.
- If the cable becomes detached, first pull forward the locking mechanism on the connector, then insert the ribbon cable ensuring that the metal contacts face towards the circuit board, and finally push the locking mechanism back into place.
- This device should be operated in a dry environment at 0–50°C.
- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; Raspberry Pi Camera Module 3 is designed for reliable operation at normal ambient temperatures.
- Store in a cool, dry location.
- Avoid rapid changes of temperature, which can cause moisture to build up in the device, affecting image quality.
- Take care not to fold or strain the ribbon cable.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimise the risk of electrostatic discharge damage.

Anexo 10: Datasheet Active Cooler



Overview

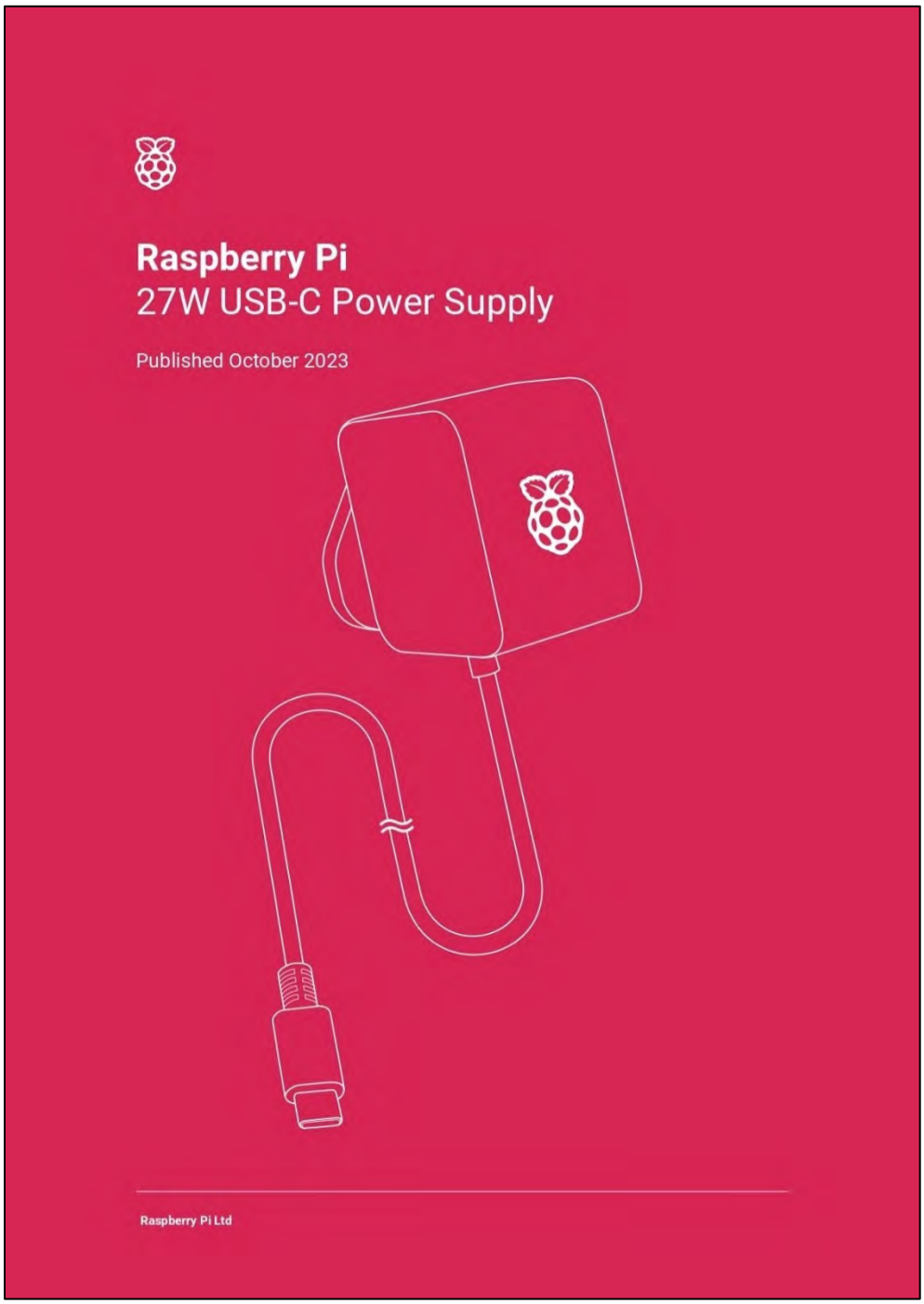
The Raspberry Pi Active Cooler for Raspberry Pi 5 is a dedicated, permanent clip-on cooling solution for Raspberry Pi 5. It combines an aluminium heatsink with a temperature-controlled blower fan to keep your Raspberry Pi 5 at a comfortable operating temperature even under heavy loads.

- Single-piece anodised aluminium heatsink
- Heatsink-mounted, temperature-controlled blower fan attached by three screws
- Spring-loaded push pins for mounting onto Raspberry Pi 5
- Pre-applied thermal pads for heat transfer

Specification

Input voltage:	5V DC supplied via four-pin fan header on Raspberry Pi 5
Fan speed control:	Pulse width modulation control with tachometer
Maximum airflow:	1.09 CFM
Maximum fan speed:	8000 RPM +/- 15%
Product material:	Anodised aluminium
Production lifetime:	The Raspberry Pi Active Cooler for Raspberry Pi 5 will remain in production until at least January 2036
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com

Anexo 11: Datasheet Power Supply



Raspberry Pi
27W USB-C Power Supply

Published October 2023

Raspberry Pi Ltd

Overview

The Raspberry Pi 27W USB-C Power Supply is an ideal power supply for Raspberry Pi 5, especially for users who wish to drive high-power peripherals, such as hard drives and SSDs, from Raspberry Pi 5's four Type A USB ports.

Delivering a maximum of 5.1V, 5A, it supports USB PD (Power Delivery), so Raspberry Pi 5 can communicate with it and select the most appropriate power profile. This enables Raspberry Pi 5 to increase the USB current limit automatically from the default 600mA to 1.6A, in order to provide extra power for devices connected to the four Type A USB ports.

Additional built-in power profiles mean the Raspberry Pi 27W USB-C Power Supply is also an excellent option for powering third-party PD-compatible products. The available profiles are 9V, 3A; 12V, 2.25A; and 15V, 1.8A, all limited to a maximum of 27W.

Specification

Input:	100 – 240Vac
Output:	5.1V, 5A; 9V, 3A; 12V, 2.25A; 15V, 1.8A (Power Delivery)
Connector:	USB-C
Cable:	1.2m 17AWG, white or black
Plug types:	• US, Canada (type A) • Europe (type C) • India (type D) • UK (type G) • Australia, New Zealand (type I)
Production lifetime:	The Raspberry Pi 27W USB-C Power Supply will remain in production until at least January 2035
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com

Anexo 12: Datasheet Cámara Tapo C320WS

Features

2K QHD Live View[Δ]
Now with 1.7 × more pixels than 1080p, providing clearer videos and more crisp photos.

IP66 Weatherproof
Helps Tapo C320WS perform well even in harsh environments with rain and dust.

Dual Powerful Antennas
Provides wider Wi-Fi coverage and more stable wireless connection with dual external antennas.

Person/Vehicle Detection
Smart AI identifies people and vehicles, notifying users as needed.

Activity Zones
Set up activity zones to only receive alerts that matter.

Sound and Light Alarm
Triggers sound and light effects to frighten away unwanted visitors.

Local[†] & Cloud[‡] Storage
Save recorded videos to a microSD card (up to 512GB)[†] or by using Tapo Care[‡] cloud storage services.

Wired/Wireless Networking
Connect the camera to your network through Ethernet or Wi-Fi for more flexible installation.

Two-Way Audio
Enables communication through a built-in microphone and speaker.

[†]microSD card needs to be purchased separately.
[Δ]Compatible with Alexa and Google Assistant supported displays. User experience is dependent on the connected device's specifications.
[‡]Subscribe for cloud storage at https://www.tapo.com/tapocare/

Specifications

Camera

- Image Sensor: 1/3"
- Resolution: 2K QHD 4MP (2560 × 1440 px)
- Night Vision: 850 nm IR LED (up to 98 ft), Starlight Color Night Vision
- Lens: F/NO: 1.61±10%; Focal Length: 3.18mm±5%

Audio

- Audio Communication: 2-way audio
- Audio Input & Output: Built-in microphone and speaker

Video

- Video Compression: H.264
- Frame Rate: 15fps
- Video Streaming: 2560 × 1440 px

Activity Notifications

- Input Trigger: Motion Detection, Person Detection, Vehicle Detection, Linecrossing Detection, Intrusion Detection, Tamper Detection
- Output Notification: Push Notification, Snapshot (Tapo Care), Clips (Tapo Care)

Network

- Wireless Protocols: IEEE 802.11b/g/n, 2.4 GHz, 2T2R
- Frequency: 2.4 GHz
- Wireless Security: WEP, WPA/WPA2-PSK
- Security: 128 bit AES encryption with SSL/TLS
- Protocol: TCP/IP, DHCP, ARP, ICMP ,DNS, NTP, HTTP, HTTPS, UDP, ONVIF, RTSP

Specifications

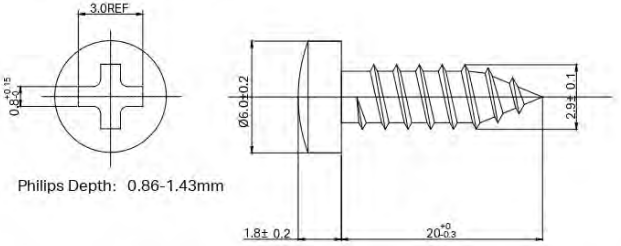
Hardware

- Port: Ethernet through RJ45 Port
- Button: RESET button
- LED: System LED
- Adapter Input: 100-240 V, 50/60 Hz, 0.3 A
- Adapter Output: 9.0 V, 0.6 A (DC Power)
- Typical Power Consumption: 5 W
- Maximum Power Consumption: 5.4 W
- External Storage: microSD card slot (Supports up to 512GB)
- Dimensions (W x D x H): 5.6*4.1*2.5 in. (142.3*103.4*64.3 mm)
- Weatherproof Rating: IP66
- Power Cord Length: 3 m

Others

- Certification: CE, RCM, RoHS, NCC, BSMI
- Package Contents:

Tapo C320WS Camera × 1
Power Adapter × 1
Waterproof Seal × 1
Mounting Anchors × 3
Mounting Screws × 3
Mounting Template × 1
Waterproof Cable Attachments × 1
Quick Start Guide



- Operating Temperature: -20 °C ~ 45 °C (-4°F ~ 113°F)
- Operating Humidity: 10%~90%RH, Non-condensing