UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE TELECOMUNICACIÓNES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"SISTEMA DE ALERTA UTILIZANDO VISIÓN ARTIFICIAL PARA DISPOSITIVOS DE RECURSOS COMPUTACIONALES LIMITADOS"

Trabajo de titulación previo a la obtención del título en Ingeniero en Telecomunicaciones

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:
Pablo David Muñoz Criollo

DIRECTOR:
MSc. Edgar Alberto Maya Olalla

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004904353		
APELLIDOS Y NOMBRES:	Muñoz Criollo Pablo David		
DIRECCIÓN:	Secundino Peñafiel 1-113 y Francisco Bonilla		
EMAIL:	pdmunozc@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0988146932

DATOS DE LA OBRA	
TÍTULO:	Sistema de Alerta utilizando Visión Artificial para dispositivos de recursos computacionales limitados
AUTOR (ES):	Muñoz Criollo Pablo David
FECHA: DD/MM/AAAA	23/02/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Telecomunicaciones
DIRECTOR:	MSc. Edgar Alberto Maya Olalla
ASESOR:	MSc. Luis Edilberto Suárez Zambrano

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 23 días, del mes de febrero de 2026

EL AUTOR:

Firmado electrónicamente por:
PABLO DAVID MUÑOZ CRIOLLO
Validar únicamente con FirmaEC

Pablo David Muñoz Criollo

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 23 de febrero de 2026

MSc. Edgar Alberto Maya Olalla

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Firmado electrónicamente por:
EDGAR ALBERTO MAYA OLALLA
Validar únicamente con FirmaEC

(f) ……
MSc. Edgar Alberto Maya Olalla
C.C.: 1002702197

DEDICATORIA

Este logro no me pertenece únicamente a mí, sino también a todas aquellas personas que estuvieron presentes a lo largo de este camino, acompañándome y aportando, cada una a su manera, a que pudiera llegar hasta aquí.

A mis padres, que han sido el pilar fundamental de mi vida. Gracias por su amor constante, por demostrarme con hechos que la constancia y la sencillez abren puertas, y por sostenerme incluso en los momentos en que el cansancio y la incertidumbre aparecieron. En cada avance, en cada meta cumplida, está reflejado su esfuerzo, su confianza en mí y el respaldo que nunca me faltó.

A las personas que estuvieron cerca, ofreciendo apoyo sincero, ya fuera con palabras de ánimo, una conversación o simplemente estando presentes cuando más lo necesitaba. Esos gestos, aunque a veces pequeños, hicieron una gran diferencia en los días difíciles.

También quiero reconocerme a mí mismo. A ese "yo" que enfrentó dudas, que atravesó obstáculos y momentos de agotamiento, pero que decidió continuar. A quien aprendió a creer en su propio potencial y hoy puede contemplar este resultado con satisfacción y gratitud.

- Pablo Muñoz

AGRADECIMIENTO

Quiero expresar, en primer lugar, mi gratitud a mi familia, por ser el cimiento sólido sobre el cual he construido cada uno de mis logros. A mis padres Juan Muñoz y Mónica Criollo, gracias por su cariño constante, por sus consejos oportunos y por inculcarme desde siempre la perseverancia frente a las dificultades. A mis amigos más cercanos Erik, Kenny, Milena, Sheyla y Aysha, que estuvieron presentes en esta etapa académica y social. Gracias por ser parte de este pequeño fragmento de mi vida.

Al MSc. Edgar Maya, director de esta tesis, le agradezco por su orientación, por compartir su conocimiento y por guiarme en cada fase de este proyecto.

Asimismo, expreso mi sincero agradecimiento al MSc. Luis Suárez, asesor de tesis, por su compromiso, su disposición de colaboración y sugerencias a este proyecto.

Extiendo también mi reconocimiento a todos los docentes que formaron parte de mi preparación profesional, en especial a quienes me motivaron a analizar con criterio, investigar con profundidad y aspirar siempre a mejorar como futuro ingeniero.

- Pablo Muñoz

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El presente trabajo de titulación consistió en el desarrollo e implementación de un sistema de alerta apoyado en visión artificial, pensado especialmente para funcionar en dispositivos con capacidad computacional limitada. Su finalidad fue identificar en tiempo real las distintas etapas del proceso de fritura de papas en la empresa "Delicias de mi Tierra". La necesidad que aborda esta problemática es la variación de los tiempos de cocción en el área de fritura de la empresa, esto es debido a que el operario debido a su experiencia visual es quien decide un punto óptimo de cocción del snack, ocasionando desperdicios de materia prima.

Como solución se diseña y entrena un modelo de visión artificial FOMO sobre una arquitectura MobileNetv2 capaz de detectar las etapas de cocción como son ebullición, deshidratación, cocción y sobrecocción. Este modelo fue implementado en un microcontrolador XIAO Seed ESP32-S3 Sense, para lograr los objetivos planteados se aplicaron técnicas de optimización de reducción de resolución y cuantización, este último utilizando INT8, permitiendo que el sistema de alerta opere sin infraestructura adicional. Posteriormente, se llevó a cabo un plan de pruebas el cual se tomaron muestras directamente en el área de fritura de la empresa Delicias de mi Tierra. Se obtuvo un rendimiento de 142 milisegundos, suficiente para garantizar el desempeño óptimo para generar las alertar visuales y auditivas. Adicional, el sistema presentó un mayor rendimiento en términos de precisión en el punto más crítico en la etapa de cocción, obteniendo una precisión superior al 90% lo que demuestra su utilidad como herramienta de apoyo para el operario de producción.

Finalmente, los resultados evidencian que técnicamente es viable integrar este tipo de modelo y dispositivo embebido en aplicaciones de término industrial a pequeña y mediana escala, debido a sus bajos costos y eficiencia de detección. Además, este proyecto permite la posibilidad de mejoras futuras, tanto de optimización del modelo, uso de otro tipo de dispositivo embebido según el enfoque final de otros procesos productivos.

Palabras clave: visión artificial, Edge Impulse, fritura, ESP32-S3, cocción, FOMO, microcontrolador.

ABSTRACT

This thesis consisted of the development and implementation of an artificial vision-based alert system, designed specifically to work on devices with limited computing capacity. Its purpose was to identify in real time the different stages of the potato frying process at the company "Delicias de mi Tierra." The need addressed by this problem is the variation in cooking times in the company's frying area, due to the fact that the operator, based on their visual experience, decides on the optimal cooking point for the snack, resulting in waste of raw materials.

As a solution, a FOMO artificial vision model was designed and trained on a MobileNetv2 architecture capable of detecting cooking stages such as boiling, dehydration, cooking, and overcooking. This model was implemented on a XIAO Seed ESP32-S3 Sense microcontroller. To achieve the objectives set, resolution reduction and quantization optimization techniques were applied, the latter using INT8, allowing the alert system to operate without additional infrastructure. Subsequently, a testing plan was carried out, with samples taken directly from the frying area of the Delicias de mi Tierra company. A performance of 142 milliseconds was obtained, sufficient to guarantee optimal performance for generating visual and audible alerts. In addition, the system performed better in terms of accuracy at the most critical point in the cooking stage, achieving an accuracy of over 90%, which demonstrates its usefulness as a support tool for production operators.

Finally, the results show that it is technically feasible to integrate this type of model and embedded device into small- and medium-scale industrial applications due to its low cost and detection efficiency. In addition, this project allows for the possibility of future improvements, both in terms of model optimization and the use of other types of embedded devices depending on the final approach of other production processes.

Keywords: computer vision, Edge Impulse, frying, ESP32S3, cooking, FOMO, microcontrollers

CAPITULO I: ANTECEDENTES

1.1.Tema

SISTEMA DE ALERTA UTILIZANDO VISIÓN ARTIFICIAL PARA DISPOSITIVOS DE RECURSOS COMPUTACIONALES LIMITADOS.

1.2. Problema

En un contexto global de rápida digitalización y avances tecnológicos, la adopción de tecnologías emergentes como la visión artificial se ha vuelto cada vez más relevante en diversas industrias, tanto en áreas de las telecomunicaciones como de Informática (Sinergy Research Group, 2019). Sin embargo, a medida que estas tecnologías avanzan, persisten desafíos significativos en términos de su implementación en entornos con recursos limitados, como dispositivos IoT (Internet de las cosas) y sistemas embebidos (Escobar, 2019).

Según (Gkillas & Lalos, 2023) menciona que los principales desafíos de la visión artificial es la optimización de algoritmos y modelos de aprendizaje para ser desplegados en dispositivos de recursos computacionales limitados, ya que estos mismos, tienen limitación de memoria, procesamiento y energía. Dicha optimización se convierte en el pilar fundamental del sistema en especial en aplicaciones en tiempo real donde la eficiencia es crucial (Greenpeace, 2017).

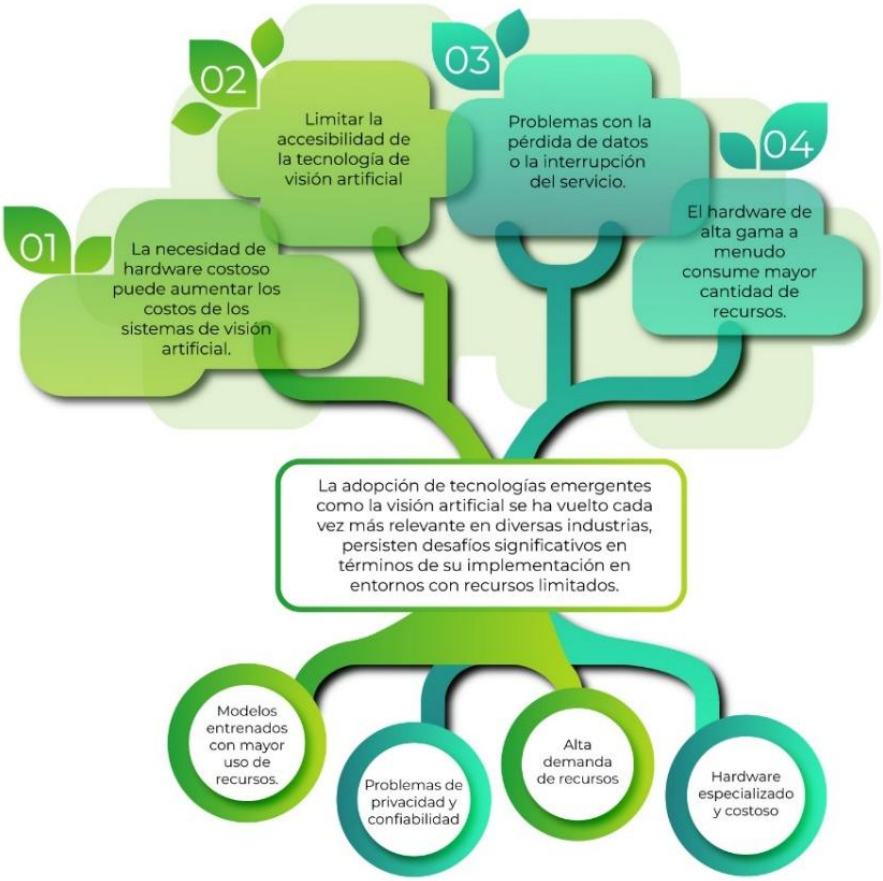
En una entrevista realizada al jefe de producción de la empresa Delicias de mi Tierra en la ciudad de Ibarra (Ruales, 2024), se constató que el área de fritura de la empresa tiene un problema. La inconsistencia en los tiempos de cocción de las papas (snacks) resulta en productos de baja calidad e incluso desperdicio de materia prima. A pesar de las capacitaciones proporcionadas al personal encargado de esta área, los errores humanos persisten y afectan

negativamente el producto final, por lo que el propietario de la empresa necesita un sistema de alerta que pueda detectar el punto óptimo de cocción.

Por lo tanto, es necesario implementar una solución basada en visión artificial que pueda operar en dispositivos con limitaciones computaciones, de esta forma funcionando como apoyo al operario del área de fritura, optimizando los recursos y disminuyendo el error humano, de esta manera logrando un sistema más ágil, autónomo y adaptado a las condiciones reales del entorno de trabajo.

Figura 1

Árbol de Problemas



Nota. La Figura muestra el árbol de problemas de Causas y Efectos del proyecto.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar un sistema de alerta utilizando tecnología de visión artificial para operar de manera eficiente en dispositivos con recursos computacionales limitados con el propósito de detectar en tiempo real el nivel de fritura de papas.

1.3.2. Objetivos Específicos

- Analizar los principios fundamentales de la visión artificial, los diferentes tipos de modelos de entrenamiento y cómo se pueden aplicar estos conceptos en un dispositivo con recursos limitados.
- Entrenar un modelo de reconocimiento de imágenes basado en visión artificial para la detección de fritura de papas.
- Realizar pruebas del prototipo para evaluar el rendimiento, la precisión y la eficacia del sistema en la detección de fritura de papas.
- Implementar el sistema de alerta con visión artificial en un dispositivo de recursos limitados.

1.4. Alcance

El proyecto actual se llevará a cabo siguiendo una metodología en V. Este enfoque estructurado asegura un desarrollo óptimo del proyecto, alineado con las recomendaciones del estudio de (Hadida, 2020).

De acuerdo con el primer objetivo específico se llevará a cabo una investigación sobre los conceptos de detección de imágenes en visión artificial, incluyendo los principios y técnicas fundamentales de la detección de objetos en imágenes digitales utilizando enfoques de visión artificial. Además, se analizarán los diferentes modelos y arquitecturas de aprendizaje

automático, evaluando sus fortalezas, debilidades y aplicabilidad en el contexto del proyecto. También se investigarán las características y limitaciones de los dispositivos de cómputo de bajo consumo y recursos limitados, con el fin de comprender los desafíos asociados a la implementación de modelos de aprendizaje profundo en estos entornos. El objetivo es adquirir los conocimientos necesarios para abordar los desafíos de implementar el modelo de aprendizaje entrenado en dispositivos con recursos computacionales limitados y seleccionar el modelo óptimo y adecuado para trabajar con dispositivos de las características mencionadas anteriormente.

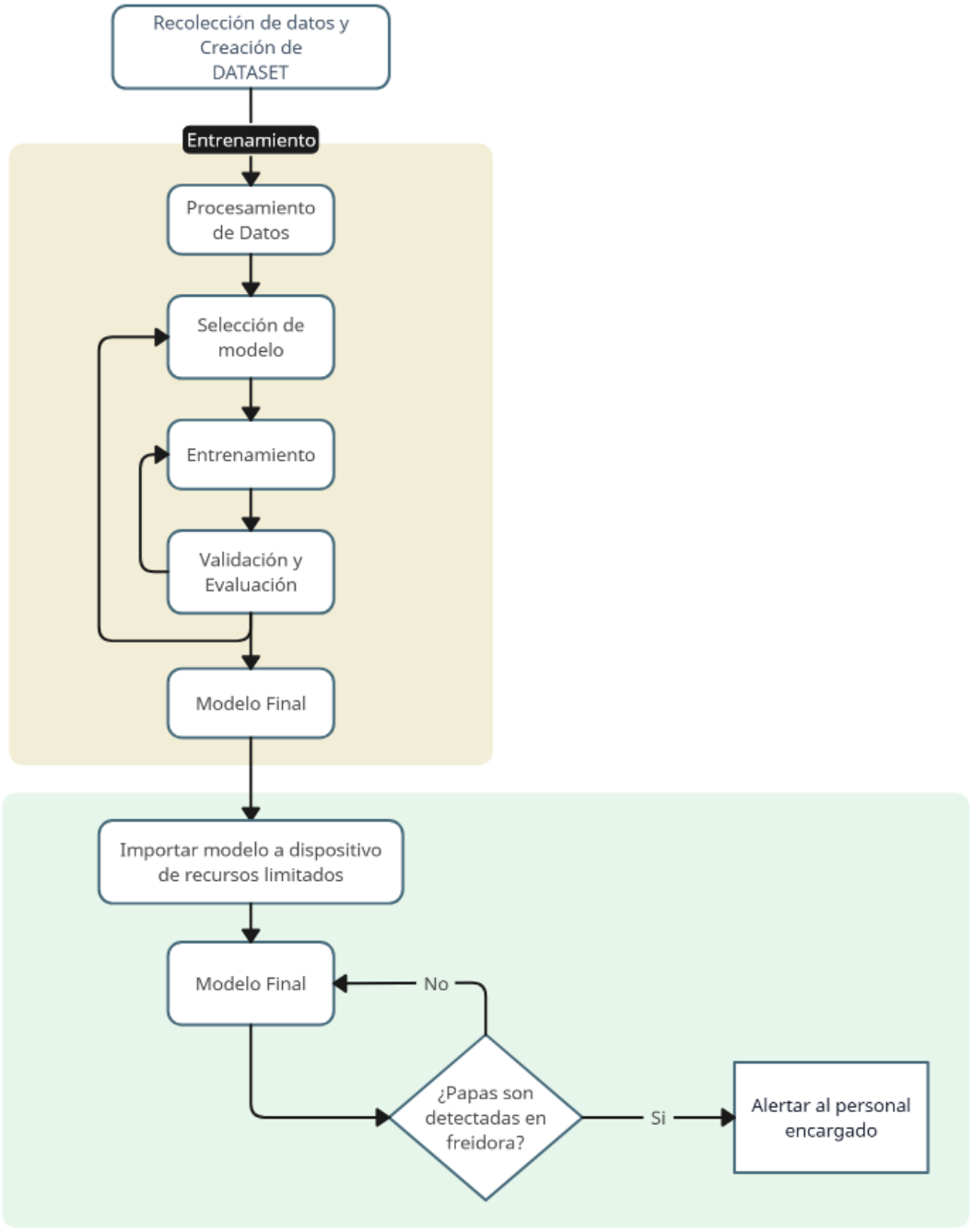
El segundo objetivo específico se llevará a cabo siguiendo el diagrama de flujo presentado en la Figura 2. Para este proyecto, se creará un dataset que contendrá información sobre las papas en el proceso de fritura. Estas imágenes etiquetadas se almacenarán y tabularán en el dataset. En la sección de la arquitectura de visión artificial se llevará a cabo el entrenamiento del modelo utilizando el dataset creado.

El tercer objetivo específico se desarrollarán las pruebas de funcionamiento en el área de fritura de la empresa “Delicias de mi Tierra”. El sistema de alerta desarrollado detectará la etapa óptima en el proceso de fritura y notificará al operario mediante una alarma. En esta etapa se evaluarán métricas de rendimiento, precisión y eficacia en la detección.

El cuarto objetivo específico implica la implementación del modelo entrenado en un dispositivo con recursos computacionales limitados para la empresa Delicias de mi Tierra, con la posterior evaluación de los resultados obtenidos.

Figura 2

Diagrama de flujo del sistema de alerta



Nota. La Figura muestra el diagrama de flujo del funcionamiento del proyecto. Autoría Propia.

1.5. Justificación

El proyecto actual se basa en la necesidad de afrontar los desafíos relacionados con el procesamiento de datos generados, y en la búsqueda de soluciones eficaces y asequibles para la detección precisa de objetos de interés en entornos industriales para pequeñas empresas, en contraste con los sistemas convencionales. Esta iniciativa se enfocará en optimizar el procesamiento de datos y en disminuir los errores humanos en aplicaciones industriales, fomentando la eficiencia en diversos sectores de forma accesible, lo que impulsará la productividad y la competitividad en el mercado (Prieto et al., 2008).

El planteamiento y construcción de un sistema de alerta apoyado en visión artificial surge como una alternativa práctica y moderna frente a la problemática identificada. Al funcionar en dispositivos con capacidad de procesamiento reducida, este sistema podrá identificar el estado de las papas durante la fritura sin necesidad de infraestructura compleja. Con ello, se busca no solo mejorar el control del proceso, sino también aprovechar mejor los recursos disponibles y aportar mayor estabilidad y uniformidad a la producción (Mellit et al., 2023). Además, la adopción de una metodología en V garantiza un enfoque sistemático y estructurado en el desarrollo del proyecto. Desde la investigación inicial hasta la implementación y pruebas finales (Duque Quevedo, 2024).

Se espera que el Sistema de Alerta basado en Visión Artificial, una vez desarrollado y probado, contribuya a un uso más eficiente de los recursos y a una mejora tangible en la calidad dentro de distintos procesos industriales. Además, el hecho de que pueda funcionar en equipos con capacidades limitadas lo convierte en una solución que amplia las posibilidades de aplicación, así como también en contextos donde la inversión tecnológica no siempre es viable, haciendo que se convierta en una alternativa accesible que permita mejoras sin requerir de una gran inversión (Sougey Medialdea, 2023).

En concordancia con el objetivo de Desarrollo Sostenible número 9, "Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales", propuesto por las Naciones Unidas, este proyecto se centra en el desarrollo de visión artificial en dispositivos con recursos limitados, el cual es una innovación tecnológica y sostenible que permite optimizar recursos y energía. Al ser accesible para empresas del sector industrial, este sistema tiene la posibilidad de mejorar procesos tradicionales, siendo más eficiente y reduciendo costos por desperdicio de materia prima, contribuyendo a la sostenibilidad y crecimiento económico de la empresa (UNDP, 2024).

El proyecto se desarrollará en un entorno real, empleando el área de fritura de la empresa “Delicias de mi Tierra” como escenario para llevar a cabo las pruebas y validar el sistema propuesto. Al realizarse en un entorno industrial, se asegurará la aplicabilidad directa de los resultados obtenidos, permitiendo validar el sistema en esta área específica y facilitando su implementación en la industria alimentaria (Lovato, 2020).

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

En este capítulo se presenta el marco teórico que sustenta el desarrollo del sistema de alerta mediante visión artificial para la detección del nivel de fritura de papas, utilizando dispositivos de recursos computacionales limitados. Se abordan conceptos clave sobre sistemas embebidos, microcontroladores de bajo consumo, modelos de inteligencia artificial optimizados, y fundamentos del proceso de cocción de la papa. Esta revisión permite comprender las bases técnicas y científicas necesarias para el diseño, implementación y validación del prototipo propuesto.

2.1. Sistemas Embebidos y Dispositivos de Recursos Computacionales Limitados

2.1.1. Definición y características

Los sistemas embebidos son equipos electrónicos creados para cumplir funciones concretas, combinando un hardware adaptado a la tarea y un software desarrollado específicamente para ese propósito. A diferencia de una computadora, este tipo de sistemas tienen mayores limitaciones en términos de memoria, consumo de energía y capacidad de procesamiento (Marwedel, 2021).

Este tipo de dispositivos de recursos computacionales limitados

son una subcategoría de sistemas embebidos que presentan capacidades restringidas de memoria (generalmente menos de 512 KB de RAM), procesadores de baja frecuencia (por debajo de 200 MHz) y opciones limitadas de conectividad y energía. Su popularidad ha crecido en entornos donde la eficiencia energética y el bajo costo son más importantes que el alto rendimiento computacional (Zhou et al., 2019)

Además, el auge de la inteligencia artificial en el borde (Edge AI) ha generado la necesidad de ejecutar modelos ligeros directamente en estos dispositivos, lo cual plantea desafíos adicionales en cuanto a procesamiento, consumo y almacenamiento (Lane et al., 2016). A pesar de ello, herramientas como TensorFlow Lite, TinyML y Edge Impulse han

hecho posible que tareas como reconocimiento de imágenes y análisis de señales se realicen de forma local sin depender de la nube.

2.1.2. Aplicaciones en la industria y el IoT

Los sistemas embebidos, que combinan componentes físicos y programación para cumplir tareas puntuales, han evolucionado significativamente en los últimos años hasta convertirse en piezas clave dentro de múltiples sectores industriales y del ecosistema del Internet de las Cosas (IoT). Su diseño les permite trabajar con recursos reducidos y consumir poca energía, lo que los hace especialmente adecuados para aplicaciones donde se necesita respuesta en tiempo real y capacidad de comunicación constante (Yalli et al., 2024).

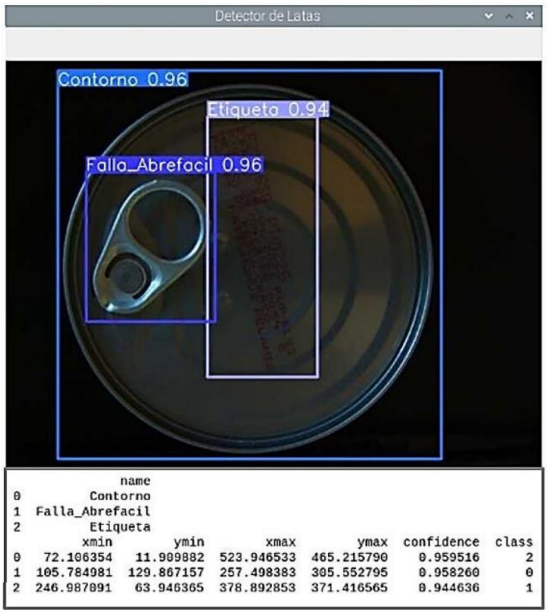
Algunas aplicaciones reales de sistemas embebidos e Internet de las Cosas (IoT) en la industria son:

- **Mantenimiento Predictivo en la Industria Electromecánica:** Se destaca que el mantenimiento predictivo representa el 46.52% de las aplicaciones de IoT en la industria electromecánica. Estas implementaciones han demostrado mejoras significativas en la eficiencia operativa, reducción de costos y sostenibilidad (Morales Tamayo, 2025).
- **Monitoreo de Procesos en la Industria Alimentaria:** En la empresa ECOLAC, ubicada en Ecuador, se implementó un prototipo de sistema embebido con tecnología IoT orientado al control de la temperatura en los tanques utilizados para la pasteurización de leche. El sistema emplea sensores tipo termocupla K para captar las variaciones térmicas y envía la información a una plataforma web, desde donde puede ser visualizada por los operadores. Gracias a este monitoreo continuo, es posible vigilar el proceso en tiempo real, lo que contribuye a mantener estándares adecuados de calidad y a disminuir posibles fallas o riesgos durante la producción (Gonzáles, 2021).

- **Control de Calidad Automatizado en la Producción de Atún Enlatado:** Un estudio reciente describe la implementación de un sistema automatizado de control de calidad en la producción de atún enlatado, utilizando visión artificial e IoT. En la Figura 3 el sistema emplea una cámara para inspeccionar las latas en una cinta transportadora, detectando defectos como problemas en las tapas o en el anillo de apertura fácil mediante el modelo YOLOv5, permitiendo una supervisión en tiempo real y mejorando la eficiencia del proceso (Vera et al., 2024).

Figura 3

Inspección de resultados de pruebas en enlatados



Nota. Imagen recuperada de (Vera et al., 2024)

2.1.3. Desafíos de diseño y operación

El diseño e implementación de sistemas embebidos en dispositivos con recursos computacionales limitados presenta múltiples desafíos técnicos que deben ser abordados cuidadosamente para garantizar un funcionamiento eficiente, confiable y seguro. Estos desafíos surgen principalmente de las restricciones de procesamiento, memoria, consumo energético y conectividad inherentes a este tipo de plataformas (Marwedel, 2021).

Uno de los retos más relevantes es la optimización del uso de recursos, ya que los microcontroladores utilizados suelen contar con capacidades de almacenamiento y procesamiento reducidas. Esto obliga a los desarrolladores a emplear algoritmos ligeros y técnicas de programación eficientes que minimicen el uso de RAM y ciclos de CPU (Lane et al., 2016). Asimismo, el uso de modelos de inteligencia artificial en estos entornos requiere cuantización, poda o compresión de redes neuronales para asegurar su ejecución local sin comprometer el rendimiento general del sistema (Banbury et al., 2020).

Un punto clave en este tipo de desarrollos es el control del consumo de energía. Cuando se trata de sistemas embebidos que deben funcionar de manera constante o durante largos periodos sin intervención técnica, es indispensable incorporar mecanismos que optimicen el uso energético, como modos de bajo consumo (deep sleep) o activación únicamente cuando ocurre un evento específico. Este enfoque no solo prolonga la vida útil del dispositivo, sino que también facilita su implementación en escenarios reales, especialmente cuando se alimenta con baterías o en lugares donde el acceso a la energía es limitado.

Por otro lado, también resulta esencial lograr una integración adecuada entre el hardware y el software. Esto implica una correcta gestión de los periféricos, temporizadores e interrupciones, así como garantizar que el firmware mantenga su estabilidad frente a variaciones del entorno o cambios internos del sistema. Además, la verificación del funcionamiento correcto y la posibilidad de realizar actualizaciones remotas se convierten en desafíos relevantes para asegurar la operación continua y confiable de estos dispositivos (Zhou et al., 2019).

2.2. Arquitectura de Microcontroladores para Dispositivos de Bajo Consumo

2.2.1. Arquitecturas de microcontroladores para visión artificial embebida

El desarrollo de soluciones de visión artificial en sistemas embebidos exige el uso de microcontroladores diseñados para mantener un balance entre capacidad de procesamiento, eficiencia energética y compatibilidad con entornos de inteligencia artificial. No se trata solo de tener potencia, sino de lograr que el dispositivo pueda ejecutar modelos de aprendizaje automático sin comprometer el consumo ni la estabilidad del sistema.

Dentro de este panorama, sobresalen tres arquitecturas que se han consolidado en microcontroladores actuales: Xtensa, RISC-V y ARM Cortex-M. Cada una ofrece características particulares, con fortalezas y también ciertas limitaciones, por lo que la elección del hardware debe realizarse de forma cuidadosa. Esta decisión es clave cuando se busca implementar modelos de aprendizaje automático directamente en el borde (edge computing), donde los recursos son más restringidos y la eficiencia es determinante.

La arquitectura Xtensa, desarrollada por Cadence y utilizada en los microcontroladores ESP32 y ESP32-S3, está diseñada para ofrecer alto rendimiento y eficiencia en el procesamiento de tareas digitales complejas. En particular, el núcleo Xtensa LX7, presente en el ESP32-S3, ofrece soporte para operaciones vectoriales (SIMD), ejecución dual-core a 240 MHz y optimizaciones específicas para tareas de procesamiento de señales e inferencia ligera (Espressif, 2023). Esta arquitectura es ideal para proyectos que requieren procesamiento de imágenes en tiempo real con consumo energético moderado y recursos de memoria controlados.

Por su parte, la arquitectura RISC-V ha ido ganando terreno en el ámbito de los sistemas embebidos gracias a su enfoque abierto y flexible. Esta característica permite adaptar los diseños según las necesidades del proyecto. Un ejemplo claro es el ESP32-C3, que incorpora esta arquitectura y ofrece soluciones eficientes en consumo energético y personalización.

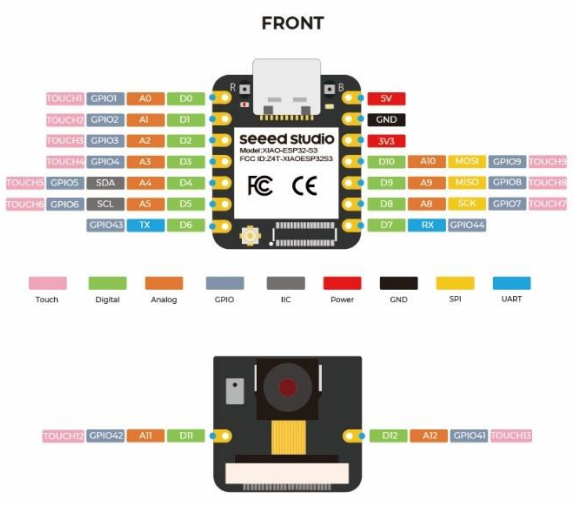
Si bien su potencia de procesamiento puede ser menor en comparación con arquitecturas como Xtensa o ARM Cortex-M, presenta ventajas interesantes como su bajo costo y la incorporación de mecanismos de seguridad por hardware, entre ellos cifrado AES y funciones SHA. Además, el creciente respaldo en plataformas de inteligencia artificial como Edge Impulse refuerza su utilidad. Por estas razones, se convierte en una alternativa adecuada para aplicaciones de inferencia básica dentro de entornos IoT donde el ahorro energético es una prioridad (Zha & Zhang, 2021).

2.2.2. Características del microcontrolador ESP32-S3 (XIAO)

El microcontrolador Seeed Studio XIAO ESP32-S3 Sense es una placa embebida basada en el chip ESP32-S3 de doble núcleo como muestra la Figura 4, diseñada para aplicaciones que requieren procesamiento local de inteligencia artificial, visión artificial y conectividad inalámbrica en espacios reducidos. Este dispositivo combina un rendimiento eficiente con un diseño físico optimizado para proyectos embebidos de bajo consumo energético, como el que se plantea en este trabajo.

Figura 4

XIAO ESP32S3 Sense



Nota. En la imagen se muestra como está conformado la placa ESP32-S3, recuperado de (Seeed Studio, 2025).

Una de las características más relevantes de este dispositivo es que incorpora de fábrica una cámara digital OV2640 y un micrófono PDM, lo que permite capturar información visual y sonora sin necesidad de añadir módulos adicionales. Esta integración simplifica considerablemente el desarrollo de aplicaciones completas, como sistemas de reconocimiento de imágenes, detección de eventos o clasificación de objetos, utilizando modelos previamente entrenados en plataformas como Edge Impulse (Arciniegas et al., 2025).

Asimismo, cuenta con una arquitectura Xtensa de doble núcleo que trabaja a 240 MHz y dispone de 8 MB de memoria PSRAM externa, lo que le otorga la capacidad suficiente para almacenar y procesar imágenes en tiempo real. Este aspecto es especialmente importante en aplicaciones de visión artificial embebida, donde la rapidez de procesamiento es determinante. Por otro lado, gracias a su conectividad Wi-Fi y Bluetooth 5, el XIAO ESP32-S3 puede integrarse sin dificultad en entornos IoT, permitiendo enviar resultados, generar notificaciones o activar alertas a través de la red (Arciniegas et al., 2025).

La siguiente Tabla 1 resume las especificaciones técnicas más relevantes de esta placa:

Tabla 1

Especificaciones técnicas del Seeed Studio XIAO ESP32-S3 Sense

Parámetro	Valor / Descripción
Arquitectura	Xtensa Dual-Core 240 MHz
Memoria SRAM	512 KB integrados
Memoria PSRAM externa	8 MB
Almacenamiento flash	8 MB
Sensor de cámara	OV2640, resolución VGA (1600x1200)
Micrófono	PDM digital

Conectividad inalámbrica	Wi-Fi 802.11 b/g/n, Bluetooth 5 (BLE)
Puertos y pines GPIO	11 pines multifuncionales (ADC, PWM, UART, I2C, SPI)
Conversores ADC	ADC de 12 bits (2 canales disponibles)
Tamaño de la placa	21.0 mm × 17.5 mm
Peso aproximado	< 4 g
Modos de bajo consumo	Deep sleep, Light sleep
Soporte IA (Edge Impulse)	Compatible con modelos TinyML y TensorFlow Lite Micro

Fuente. Autoría propia.

2.3. Visión Artificial

La visión por computadora puede entenderse como una rama tecnológica que permite a los sistemas informáticos interpretar y actuar a partir de información visual, buscando imitar, hasta cierto punto, la manera en que las personas perciben su entorno. Para lograrlo, sigue un proceso organizado que comienza con la captura de imágenes, continúa con el preprocesamiento de los datos y la segmentación de los elementos relevantes, y avanza hacia la identificación de características y patrones. Inicialmente, el sistema captura imágenes mediante sensores ópticos, como cámaras digitales, generando datos en bruto que luego deben ser refinados. A través de técnicas de preprocesamiento, se mejora la calidad de estas imágenes aplicando filtros, correcciones de iluminación y normalización, lo cual facilita la detección precisa de objetos o patrones visuales relevantes (Barrios-Chinchayhuara et al., 2022)

2.3.1. Funcionamiento de la Visión Artificial

Una vez que la información visual ha sido capturada e interpretada, es necesario pasarla por una fase de preprocesamiento antes de realizar cualquier análisis profundo. Esta etapa cumple un papel fundamental, ya que permite mejorar la calidad de las imágenes, corrigiendo

posibles ruidos, imperfecciones o distorsiones que podrían afectar los resultados. Al optimizar los datos, se facilita la identificación de características importantes y la detección de objetos relevantes dentro de la escena. Posteriormente, la información procesada se examina con mayor detalle para generar conclusiones o decisiones fundamentadas. En muchos casos, los resultados obtenidos no solo quedan en el análisis, sino que también se utilizan para enviar retroalimentación o activar acciones dentro de otros sistemas, integrando así el proceso de visión con mecanismos de control y automatización. Para llevar a cabo estas tareas, se emplean algoritmos avanzados de aprendizaje automático y técnicas específicas de procesamiento de imágenes, lo que permite interpretar visualmente el entorno (Yandrapati et al., 2023).

2.3.1.1. Adquisición de Datos.

En cualquier sistema de visión artificial, el punto de partida siempre es la obtención de los datos visuales. La calidad y fidelidad de estas imágenes son determinantes, ya que influyen directamente en el rendimiento y la exactitud del modelo que se desarrollará posteriormente. Si la información capturada no es adecuada, el análisis posterior difícilmente será confiable. Esta fase implica registrar imágenes mediante sensores ópticos, comúnmente cámaras digitales, que permiten capturar escenas del entorno ya sea en condiciones controladas o bajo variaciones propias del ambiente. En el caso de los sistemas embebidos, suelen emplearse cámaras compactas y de bajo consumo energético. Esto implica trabajar con ciertas restricciones, como una resolución limitada, velocidades de captura moderadas y posibles variaciones en la iluminación, aspectos que deben considerarse cuidadosamente durante el diseño del sistema. Según (Barrios-Chinchayhuara et al., 2022) la resolución de la imagen debe elegirse con equilibrio: lo suficientemente alta para conservar los detalles importantes, pero sin llegar a saturar la memoria o los recursos del dispositivo. Al mismo tiempo, contar con una iluminación homogénea ayuda a disminuir imperfecciones visuales y facilita que las etapas posteriores, como la segmentación, se realicen con mayor precisión y estabilidad.

2.3.1.2. Representación de una Imagen en Visión Artificial.

En el ámbito de la visión artificial, las imágenes dejan de ser fotos para convertirse en estructuras de datos. El sistema las entiende como matrices multidimensionales donde la unidad mínima, el píxel, no es más que un valor numérico. Dicho valor cuantifica qué tan intensa es la luz en uno o varios espectros cromáticos. Según (Banzi -Arduino, 2021) en imágenes en escala de grises, esta estructura es una matriz bidimensional como muestra la Ecuación (1).

Ecuación (1)

$$I \in \mathbb{R}^{HxW}$$

Donde H es la altura y W el ancho de la imagen. Mientras que en la Ecuación (2) para imágenes a color (RGB), se emplea un tensor tridimensional donde el tercer eje representa los tres canales de color: rojo, verde y azul.

Ecuación (2)

$$I \in \mathbb{R}^{H\times W\times 3}$$

Durante el entrenamiento de modelos de aprendizaje profundo, estas imágenes se normalizan y redimensionan a formatos estándar, como 96x96 píxeles, para reducir la carga computacional sin perder información relevante. Estas matrices constituyen la entrada (input tensor) del modelo, que posteriormente transforma estos datos a través de múltiples capas para extraer patrones jerárquicos. En arquitecturas como MobileNetV2, utilizadas por algoritmos como FOMO, esta representación permite generar salidas en forma de mapas de activación que sintetizan la presencia y ubicación de objetos o clases en la imagen, optimizando el uso de memoria en dispositivos de bajos recursos (Banzi -Arduino, 2021).

2.3.1.3. Preprocesamiento de Imágenes.

Para que el etiquetado sea efectivo, primero es necesario 'limpiar' y optimizar el material gráfico. En esta fase de preprocesamiento, se busca potenciar las características críticas de la imagen. Para ello, solemos recurrir a la ecualización del contraste, la eliminación

de interferencias (ruido), la aplicación de filtros específicos y la segmentación de las áreas de interés.

Las imágenes capturadas inicialmente suelen almacenarse en formato RGB, por ejemplo con dimensiones de 96x96x3. Sin embargo, en muchos casos pueden convertirse a escala de grises (96x96x1), lo que disminuye considerablemente el consumo de memoria, algo especialmente útil cuando se trabaja con dispositivos que tienen recursos limitados. Además de esta conversión, es común aplicar ciertos ajustes para mejorar la calidad de la imagen antes del análisis. Entre ellos se encuentra el filtrado gaussiano, que ayuda a suavizar el ruido, así como la modificación del contraste para resaltar mejor los bordes. En determinadas aplicaciones también se realiza una segmentación previa, con el fin de separar y enfocarse únicamente en las zonas de interés dentro de la imagen. Según (Mellit et al., 2023) estos procedimientos no solo mejoran el rendimiento durante la inferencia, sino que también ayudan a reducir el riesgo de sobreajuste. Al simplificar y depurar la información visual, se favorece que el modelo aprenda patrones más generales en lugar de depender de detalles muy específicos, lo que contribuye a que su comportamiento sea más estable frente a nuevas imágenes.

Según (Eki et al., 2023) las técnicas más comunes con un gran resultado son:

- **Filtrado:** Aplicar filtros como el gaussiano para suavizar la imagen y reducir el ruido.
- **Conversión a Escala de Grises:** Con el fin de aligerar la carga de procesamiento, hemos optado por trabajar con imágenes convertidas a escala de grises. Este tipo de preprocesamiento de imágenes será la metodología adoptada en este proyecto y se implementará en la plataforma Edge Impulse.
- **Segmentación:** Separar la imagen en regiones de interés para facilitar la identificación de objetos.

2.3.1.4. Etiquetado de Imágenes.

El proceso de etiquetado consiste en asignar una clase o categoría a determinadas zonas dentro de cada imagen. Esta tarea es fundamental, ya que permite construir un conjunto de datos organizado que servirá como base para que el modelo de aprendizaje automático identifique y aprenda los patrones visuales correspondientes (Barrios-Chinchayhuara et al., 2022).

La definición de etiquetas establece las categorías de objetos a detectar, luego se etiqueta manualmente las imágenes identificando las regiones de interés (ROI) utilizando herramientas de software como Roboflow o Edge Impulse y finalmente se revisan y validan las etiquetas para asegurar su precisión y consistencia. En el caso de Edge Impulse cuenta con una biblioteca existente de modelos de detección de objetos preentrenados de YOLOv5 (entrenados con el conjunto de datos COCO), los objetos comunes en las imágenes se pueden identificar y etiquetar rápidamente en segundos sin necesidad de escribir ningún código (Arun et al., 2023).

2.3.1.5. Creación del Conjunto de Datos.

Una vez que las imágenes han sido etiquetadas, el siguiente paso consiste en organizar toda la información dentro de un conjunto de datos estructurado. Generalmente, este se divide en tres partes con funciones específicas. En primer lugar, la mayor cantidad de datos se destina al entrenamiento, ya que es en esta etapa donde el modelo aprende a identificar los patrones presentes en las imágenes. Por otra parte, se reserva un grupo adicional para la validación, el cual permite ajustar y afinar los parámetros internos durante el proceso de aprendizaje. Gracias a esta separación, es posible mejorar el desempeño del sistema sin que simplemente memorice la información inicial. Finalmente, se mantiene un tercer subconjunto completamente independiente, utilizado exclusivamente para la fase de prueba. Este último permite medir el

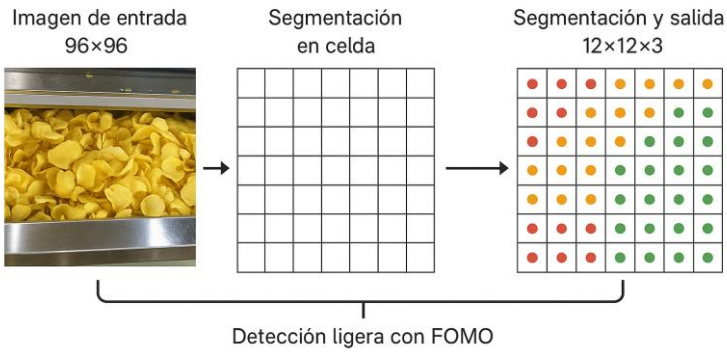
rendimiento real del modelo frente a datos que no ha visto antes, asegurando así que su comportamiento sea confiable y generalizable (Bayar et al., 2023).

2.3.1.6. Segmentación y Detección.

La segmentación y detección son fases esenciales en sistemas de visión artificial, encargadas de identificar y localizar regiones significativas dentro de una imagen. En este proyecto, se implementa el algoritmo FOMO (Faster Objects, More Objects), una arquitectura ligera diseñada específicamente para sistemas embebidos que permite la detección de objetos sin necesidad de utilizar bounding boxes. A diferencia de modelos convencionales como YOLO o SSD, que delimitan objetos mediante cajas rectangulares, Como se observa en la Figura 5, FOMO realiza una segmentación gruesa al dividir la imagen en una cuadrícula regular y analizar cada celda de forma individual para determinar la probabilidad de contener un objeto de interés (Banzi -Arduino, 2021).

Figura 5

Segmentación en celdas



Fuente. Auditoría propia.

Según (Eki et al., 2023) este enfoque convierte la salida del modelo en un mapa tridimensional, donde cada celda representa una región de 8x8 píxeles de la imagen original y predice la clase del objeto o fondo presente. Esta estructura permite reducir significativamente el consumo de memoria, ya que evita la necesidad de calcular las coordenadas exactas de los

bordes de cada objeto. Para el caso del presente proyecto, la imagen de entrada de 96x96 píxeles se transforma en una matriz de 12x12 posiciones, donde cada celda se evalúa por su contenido visual asociado a una de las clases entrenadas (ebullición, deshidratación o cocción).

2.3.1.7. Análisis y Clasificación.

A continuación, es necesario examinar las regiones previamente identificadas con el fin de obtener información relevante que permita su posterior clasificación. Para lograrlo, se realiza un proceso de análisis más detallado que facilita distinguir cada elemento según sus particularidades. Este proceso implica:

- Extracción de Características: Cálculo de atributos como textura, color, forma, etc.
- Clasificación: Se emplean diversos modelos de aprendizaje computacional con el fin de asignar una identidad específica a los elementos previamente identificados. En lugar de procesar datos aislados, el sistema se apoya en arquitecturas robustas.

2.3.1.8. Toma de Decisiones.

En la etapa final, el sistema de visión artificial emplea los datos analizados para generar una acción o respuesta concreta. A partir de la información procesada, el sistema puede ejecutar decisiones de manera autónoma. Por ejemplo, puede activar una alarma cuando detecta una anomalía, descartar un producto que no cumple con los estándares en una línea de producción o incluso dirigir a un robot para que realice una tarea determinada. De esta forma, el análisis visual se convierte en una herramienta práctica que impacta directamente en el funcionamiento del entorno donde se aplica (Barrios-Chinchayhuara et al., 2022).

2.4. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales, conocidas como CNN, forman parte de los modelos de aprendizaje profundo más empleados en el ámbito de la visión por computadora.

Esto se debe a su habilidad para identificar patrones tanto espaciales como jerárquicos dentro de las imágenes, lo que les permite comprender mejor la información visual (Ma et al., 2018). Su arquitectura está inspirada en el funcionamiento del sistema visual humano, especialmente en la manera en que el cerebro procesa estímulos visuales, lo que explica su eficacia en este tipo de tareas. Gracias a estas características, las CNN han demostrado un desempeño destacado en diferentes aplicaciones, como la clasificación de imágenes, la segmentación semántica y el reconocimiento facial, entre muchas otras áreas donde el análisis visual es fundamental (Ma et al., 2018).

2.4.1. Arquitectura y funcionamiento

La arquitectura típica de una CNN está compuesta por una secuencia de capas convolucionales, capas de activación, capas de agrupamiento (pooling) y capas completamente conectadas (fully connected) (Ma et al., 2018). En los estratos iniciales del modelo, la atención se centra en la detección de elementos primordiales, tales como la orientación de bordes o patrones de textura superficial. Sin embargo, a medida que los datos transitan hacia las capas de mayor profundidad, el sistema logra una síntesis superior, transformando esos rasgos básicos en representaciones abstractas que permiten identificar siluetas complejas y estructuras íntegras de los objetos. Finalmente, las capas densas realizan la clasificación o regresión del input procesado.

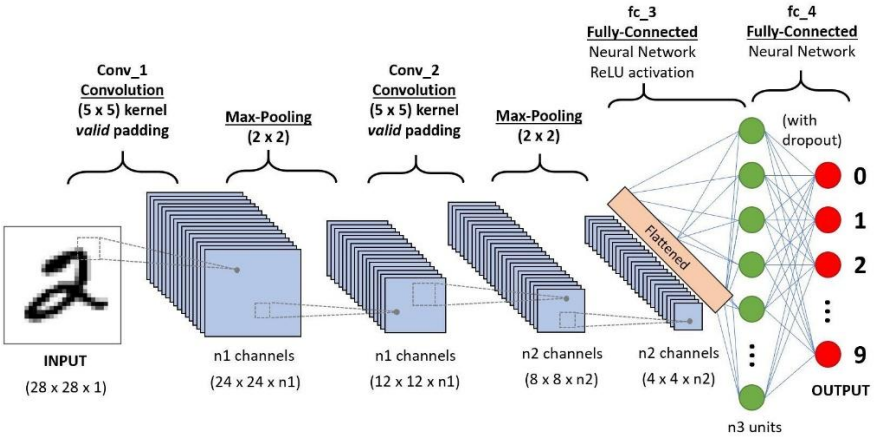
Según (Zhou et al., 2019), las CNN modernas integran también técnicas como normalización por lotes (Batch Normalization) y regularización mediante Dropout, lo que mejora la generalización del modelo y reduce el sobreajuste. La arquitectura ResNet, por ejemplo, introdujo conexiones residuales que permiten entrenar redes muy profundas sin degradación del rendimiento, abriendo paso a aplicaciones más complejas en entornos de visión artificial industrial y médica.

2.4.2. Operaciones fundamentales (convolución, pooling, activación)

Convolución: La etapa de convolución se lleva a cabo mediante la aplicación de pequeños filtros, también conocidos como kernels, que recorren de manera ordenada la matriz de entrada. A medida que estos filtros se desplazan sobre la imagen, van resaltando determinados rasgos visuales. Por ejemplo, como se observa en la Figura 6, algunos pueden identificar líneas horizontales, otros bordes definidos o incluso esquinas. Es importante señalar que estos filtros no son fijos desde el inicio, sino que se ajustan durante el proceso de entrenamiento. De esta manera, el modelo aprende qué características son más relevantes y logra capturar las propiedades espaciales más significativas de los datos de entrada. (Rawat & Wang, 2017).

Figura 6

Arquitectura de red neuronal convolucional



Fuente. Recuperado de https://datapeaker.com/wp-content/uploads/2021/08/90650dnn2-5115625.jpeg

Pooling: Esta operación reduce dimensionalidad y complejidad computacional, conservando las características más relevantes. El agrupamiento máximo o max pooling se destaca como la variante predominante. Esta operación mejora la invariancia a traslaciones y robustez frente a distorsiones.

Funciones de activación: Estas funciones incorporan no linealidades indispensables para que la red pueda aprender relaciones complejas dentro de los datos. Sin este componente, el modelo se limitaría a comportamientos lineales y perdería capacidad de representación. La función más utilizada es ReLU (Rectified Linear Unit), debido a su simplicidad y eficiencia computacional (Liu et al., 2016). No obstante, con el tiempo se han propuesto variantes como Leaky ReLU, ELU y Swish, las cuales buscan corregir ciertos inconvenientes, como el fenómeno conocido como "muerte de neuronas", que ocurre cuando algunas unidades dejan de activarse durante el entrenamiento. Gracias a estas alternativas, es posible mantener un flujo de aprendizaje más estable y mejorar el rendimiento del modelo en distintos escenarios.

Investigaciones como la de (Rawat & Wang, 2017) confirman que estas operaciones combinadas permiten a las CNN construir representaciones profundas, eficientes y escalables para distintas tareas de visión computacional.

2.4.3. Representación de características y mapas de activación

Los llamados mapas de activación no son más que la huella que dejan los filtros al recorrer una imagen. En otras palabras, muestran cómo reacciona cada filtro frente a distintas zonas del contenido visual. A medida que la información avanza por la red, estas representaciones van cambiando: en las primeras capas suelen resaltar patrones sencillos, como bordes o texturas, mientras que en niveles más profundos comienzan a reflejar estructuras mucho más abstractas y con mayor significado (Zhou et al., 2016). Ahora bien, estas salidas no solo sirven para que la red "funcione", sino también para comprender qué está observando realmente el modelo cuando toma una decisión. Herramientas como Class Activation Mapping (CAM) o Grad-CAM permiten visualizar qué regiones influyen más en la clasificación final. Este tipo de análisis resulta especialmente valioso en aplicaciones delicadas, por ejemplo, en sistemas de apoyo al diagnóstico médico, donde no basta con obtener un resultado: también es necesario entender por qué el modelo llegó a esa conclusión.

Según (Zhou et al., 2016) demostró que estas visualizaciones no solo ayudan en la interpretabilidad, sino que también pueden utilizarse para optimizar modelos mediante atención guiada y regularización espacial.

2.5. Modelos de Entrenamiento para Detección de Objetos

La detección de objetos es una tarea fundamental en la visión por computadora, Según (Rawat & Wang, 2017) implica identificar y localizar instancias de objetos dentro de una imagen o secuencia de video. Para lograrlo, se emplean diversos modelos de entrenamiento que varían según el enfoque de aprendizaje y la representación de los objetos.

2.5.1. Modelos de entrenamiento en visión artificial

2.5.1.1. Aprendizaje supervisado.

Dentro de las técnicas utilizadas para la detección de objetos, el aprendizaje supervisado es el más común. Este enfoque se basa en entrenar al modelo con imágenes previamente etiquetadas, es decir, cada imagen ya contiene información sobre qué objetos aparecen y en qué parte se encuentran. Por lo general, estas ubicaciones se indican mediante cuadros delimitadores que enmarcan cada elemento de interés. Gracias a este proceso guiado, el modelo aprende a relacionar ciertas características visuales con categorías específicas. Con el tiempo, logra reconocer esos mismos patrones en imágenes nuevas y no vistas, pudiendo no solo identificar el objeto, sino también señalar su posición dentro de la escena. (Szeliski, 2022).

2.5.1.2. Aprendizaje no supervisado.

Aunque no es el enfoque más utilizado en tareas de detección de objetos, el aprendizaje no supervisado ha despertado un interés creciente en los últimos años. A diferencia del aprendizaje supervisado, aquí el modelo no cuenta con etiquetas que le indiquen qué debe aprender. En su lugar, intenta encontrar por sí mismo patrones, similitudes o estructuras dentro de los datos disponibles. Por ejemplo, mediante técnicas como el agrupamiento o clustering, es posible identificar conjuntos de datos que comparten características similares, lo que podría

relacionarse con distintas categorías de objetos. Sin embargo, a pesar de su potencial, este tipo de métodos todavía se encuentra en evolución cuando se trata de aplicaciones específicas como la detección precisa y localización de objetos, donde el nivel de exactitud requerido es alto (Szeliski, 2022).

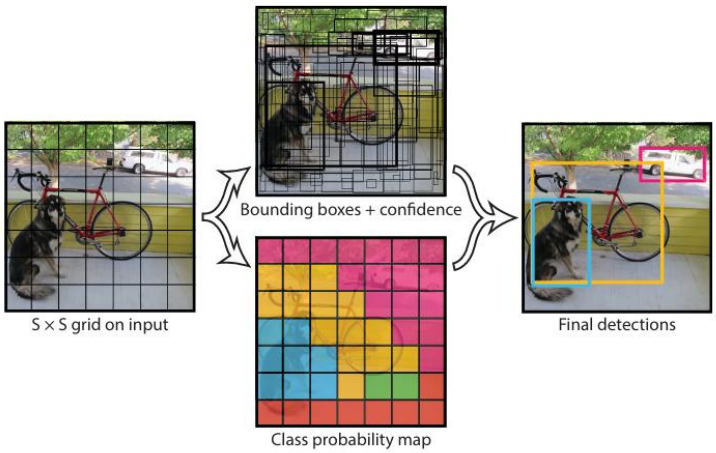
2.5.2. Modelos basados en bounding boxes: YOLO, SSD

YOLO (You Only Look Once)

Dentro del ámbito de la visión computacional, la familia de modelos YOLO se distingue por abordar el reconocimiento de objetos como un desafío de regresión unificado. A diferencia de los métodos multietapa, esta arquitectura calcula de manera simultánea tanto las coordenadas de los recuadros delimitadores como las probabilidades de pertenencia a una categoría mediante un flujo de datos único a través de la red. Dicha metodología de procesamiento integral es la que garantiza una capacidad de respuesta en tiempo real sin sacrificar los niveles de exactitud en la identificación. YOLO divide la imagen en una cuadrícula como se ve en la Figura 7 y, para cada celda, predice un número fijo de cuadros delimitadores junto con las probabilidades de clase correspondientes (Redmon, 2016).

Figura 7

Funcionamiento de YOLO



Fuente. Recuperado de (Redmon, 2016)

SSD (Single Shot MultiBox Detector)

El modelo SSD se distingue por su capacidad de procesar la información visual a través de una jerarquía de mapas de características con resoluciones variables. A diferencia de otros sistemas, esta metodología permite capturar entidades de diversas dimensiones al asociar cada nivel de activación con una serie de marcos de referencia preestablecidos (Liu et al., 2016).

2.5.3. Modelos por segmentación: Mask R-CNN

Mask R-CNN puede entenderse como una mejora sobre la arquitectura Faster R-CNN, ya que amplía sus capacidades al incorporar segmentación de instancias. Es decir, no solo identifica y clasifica los objetos dentro de una imagen, sino que además es capaz de delinearlos con mayor detalle, generando una máscara precisa para cada uno.

A diferencia de los métodos que se limitan a dibujar un cuadro alrededor del objeto, este modelo permite definir su contorno a nivel de píxel. Esta característica resulta especialmente útil en situaciones donde la forma exacta del elemento es importante, ya que ofrece una representación mucho más fiel y completa que la simple detección basada en bounding boxes (Deepak & Bhat, 2025).

2.5.4. Modelos por celdas: FOMO (Faster Objects, More Objects)

FOMO es un modelo de detección de objetos diseñado para dispositivos con recursos computacionales limitados. A diferencia de los modelos tradicionales que predicen cuadros delimitadores, FOMO divide la imagen en una matriz de celdas y predice la presencia y clase de objetos en cada celda. Este enfoque reduce significativamente la carga computacional y permite la detección en tiempo real en dispositivos de baja potencia, como microcontroladores (Mellit et al., 2023).

2.6. FOMO: Principios y Aplicación en Dispositivos Edge

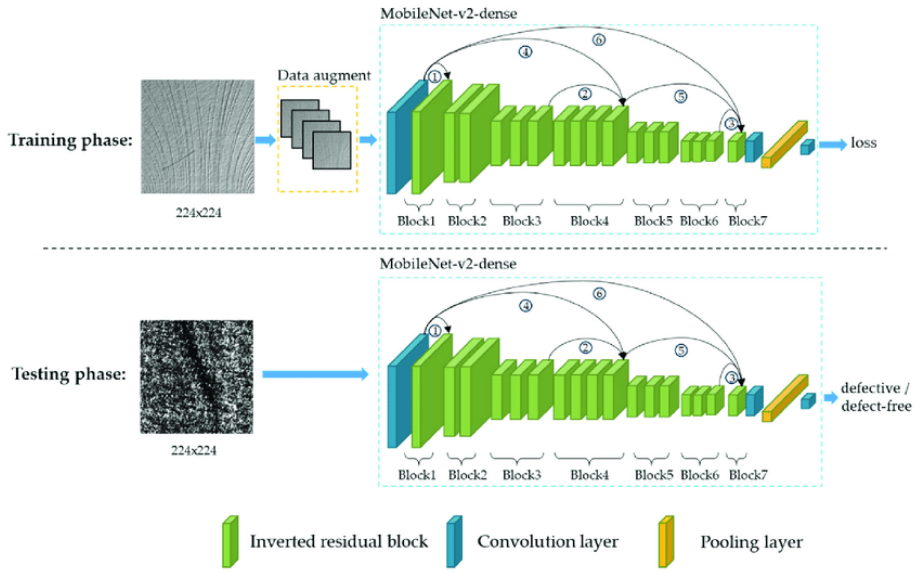
2.6.1. Arquitectura general de FOMO y su base en MobileNetV2

La arquitectura MobileNet-v2 se fundamenta en la búsqueda de la eficiencia operativa, consolidándose como una solución idónea para entornos con capacidades de hardware restringidas, tales como sistemas embebidos o dispositivos móviles. Como se muestra en la Figura 8, el núcleo de su diseño radica en la sustitución de las operaciones convencionales por convoluciones separables en profundidad. Esta estrategia técnica permite una reducción drástica en la volumetría de parámetros y en la carga de procesamiento, sin comprometer severamente el rendimiento del modelo en tareas de visión artificial.

- **Convoluciones separables en profundidad (Depthwise Separable Convolutions):** Este tipo de operación descompone la convolución convencional en dos pasos más sencillos: primero se aplica una convolución depthwise, que actúa sobre cada canal de manera independiente, y luego una convolución pointwise, que combina la información resultante. Gracias a esta división, se reduce considerablemente la cantidad de parámetros y el número de cálculos necesarios, lo que mejora la eficiencia sin sacrificar demasiado rendimiento.
- **Capas Base:** La arquitectura principal de la red está formada por una secuencia organizada de capas convolucionales acompañadas de etapas de *pooling*. En conjunto, estas capas permiten extraer progresivamente patrones relevantes de la imagen, filtrando la información hasta quedarse con aquellas características que resultan más significativas para la tarea que se desea realizar.

Figura 8

Arquitectura MobileNet-v2



Nota. La Figura muestra los bloques de concatenación del modelo MobileNet-v2.

Este algoritmo se caracteriza por ser liviano en comparación con otros modelos más complejos, ya que utiliza menos parámetros y demanda menos operaciones durante su ejecución. Gracias a esto, resulta especialmente adecuado para dispositivos con capacidad de procesamiento reducida, donde cada recurso cuenta. Además, permite ajustar el equilibrio entre rapidez de respuesta y nivel de precisión mediante configuraciones relacionadas con el ancho de la red y la resolución de entrada. Esta flexibilidad facilita adaptarlo según las necesidades del proyecto. Sumado a ello, su bajo consumo energético lo convierte en una alternativa conveniente para aplicaciones que deben operar durante largos periodos sin un gasto elevado de energía (Lane et al., 2016).

2.7. Proceso de producción de snack

La producción de snacks de papas fritas comienza con la selección y recepción de las papas crudas, siendo la calidad de la materia prima crucial para determinar la calidad del producto final. Según (Fiallos, 2023) la calidad alimentaria es importante para verificar que un producto sea sano, sabroso y seguro, ya que el consumidor es quien juzga su valor. Por ello, la

Procesadora de Alimentos "Delicias de mi Tierra" utiliza la papa (Solanum tuberosum L.) en su variedad denominada "Chola" como muestra la Figura 9, un tubérculo mediano con forma oval-elíptica, seleccionada por su bajo contenido de azúcar y alto contenido de almidón. Además, según (Guzman et al., 2024) menciona que contiene un alto contenido de carbohidratos, vitaminas y minerales, las cuales aportan de manera positiva nutricionalmente al ser humano. Esta variedad es la más adecuada para la producción de snacks de papas fritas debido a sus propiedades, lo que garantiza un producto de alta calidad que satisface las expectativas del consumidor (Pumisacho & Velasquez, 2009).

Figura 9

Papa (Solanum tuberosum L.) en su variedad Chola

Nota. La figura muestra la variedad de papa Chola con propiedades de piel rosada áspera, y amarilla alrededor de los ojo. Adaptado de *Obtención de snack alimenticio a partir de cuatro variedades de papa (Solanum tuberosum L), utilizando aceites esenciales de rizomas* (Guzman et al., 2024).

2.7.1. Lavado, Pelado y Corte

Una vez seleccionadas, las papas se someten a un proceso de lavado para eliminar tierra, piedras y otros residuos utilizando una máquina industrial fabricada en acero inoxidable como se observa en la Figura 10, donde son sometidas a cepillos de goma y cerda especial que pelan cada papa, acumulando la piel o cáscara en un filtro que se vacía posteriormente. Este proceso

de lavado y pelado es crucial para garantizar la higiene y seguridad alimentaria del producto final, cumpliendo con estrictos estándares de calidad y normativas sanitarias.

Figura 10

Proceso de lavado y pelado de papas

Nota. La figura muestra el área de lavado y pelado de la Procesadora de alimentos "Delicias de mi Tierra". Autoría Propia.

Posteriormente, las papas peladas se cortan en rodajas de un grosor uniforme utilizando una máquina cortadora especializada, siendo el más utilizado un grosor de 2 mm debido a su mejor resistencia una vez empacado, según el Jefe de producción (Ruales, 2024) esta uniformidad en el grosor de las rodajas, que generalmente varía entre 1.2 mm y 2 mm, es esencial para asegurar una cocción pareja y evitar que algunas rodajas queden crudas mientras otras puedan quemarse, lo que se traduce en un producto final de alta calidad y consistencia. Además, mantener un espesor constante en las rodajas no solo facilita un empaquetado más ordenado, sino que también mejora la presentación final del producto. Esta uniformidad visual resulta más atractiva para el consumidor, ya que transmite calidad y cuidado en el proceso de elaboración.

2.7.2. Lavado Posterior y Secado

Luego del corte, las rodajas de papa se someten a un segundo lavado para eliminar el exceso de almidón superficial, lo que evita que las rodajas se peguen durante la fritura. Este lavado se realiza mediante un sistema de inmersión durante varios minutos, el cual bloquea la actividad enzimática y remueve azúcares, preparando así las rodajas para la siguiente etapa del proceso (Castillo & Colorado, 2022). Posteriormente, las rodajas se secan utilizando una línea de banda con un rodillo de centrifugado, proceso crucial para obtener una textura crujiente durante la fritura. La eliminación del exceso de humedad mediante este sistema de secado permite que las rodajas absorban menos aceite durante la fritura, resultando en un producto final más saludable y con una textura crocante y agradable al paladar. Además, el secado uniforme de las rodajas asegura una fritura pareja y evita la formación de burbujas o ampollas en la superficie, lo que mejora la apariencia y calidad del producto final (Posligua Bran et al., 2009).

2.7.3. Fritura

La fritura es una etapa crítica en la producción de papas fritas en esta empresa, donde las rodajas de papa se sumergen en aceite vegetal a una temperatura constante de 180 grados centígrados. Este proceso de fritura puede durar entre 2 y 3 minutos, dependiendo del grosor de las rodajas y el tipo de aceite utilizado, según el jefe de producción (Ruales, 2024) las freidoras utilizadas en esta etapa sumergen completamente las rodajas en el aceite caliente, asegurando una cocción uniforme y evitando que algunas partes queden crudas mientras otras se queman. Sin embargo, debido a que el personal encargado es quien detecta el nivel óptimo de fritura, el tiempo de cocción puede variar ligeramente. Una vez que las rodajas alcanzan el punto perfecto de dorado y crujencia, son retiradas de las freidoras y trasladadas a un área de secado y saborizado para eliminar el exceso de aceite y aplicar los condimentos deseados. Este proceso de fritura a alta temperatura y corta duración es fundamental para lograr la textura

característica de las papas fritas, con un exterior crujiente y dorado, satisfaciendo las expectativas de los consumidores (Castillo & Colorado, 2022).

2.7.3.1. Proceso de Fritura.

Según (Montes O. et al., 2016) la fritura de alimentos implica procesos químicos y celulares complejos que ocurren a nivel microscópico. Durante este proceso, el agua contenida en el alimento se calienta rápidamente, convirtiéndose en vapor como se observa en la Figura 11. Este vapor necesita escapar del interior del alimento, buscando puntos débiles en su estructura celular, como las uniones entre células. A medida que el vapor se abre paso, crea pequeñas cavidades o poros en la superficie del alimento, lo que permite que el aceite de fritura penetre en su interior. Por lo tanto, es crucial controlar cuidadosamente los parámetros de fritura, como la temperatura, el tiempo y la calidad del aceite, para obtener alimentos con una textura crujiente y un sabor óptimo (Castillo & Colorado, 2022).

Además, según (Montes O. et al., 2016) existen cuatro etapas de este proceso:

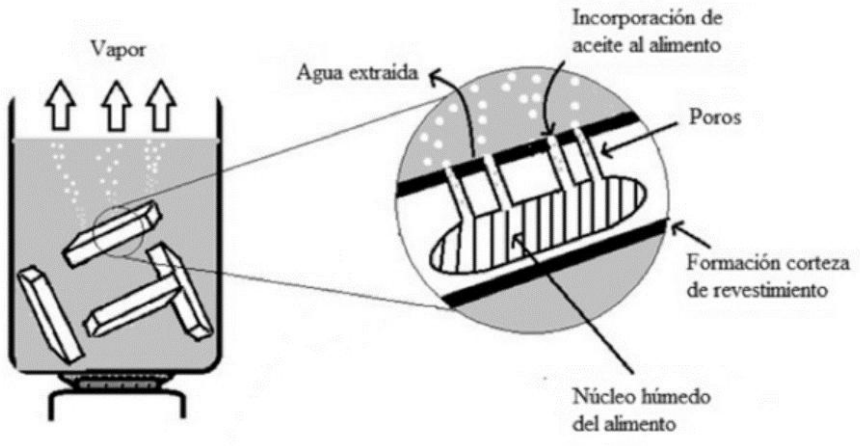
- **Etapa de Inmersión Inicial:** Se observa que la inmersión inicial genera un choque térmico casi instantáneo, elevando la temperatura superficial de las rodajas de forma drástica durante los primeros segundos de exposición.
- **Etapa de Deshidratación:** A medida que la fritura progresa, se manifiesta un fenómeno de deshidratación proporcional al calor del aceite, lo que induce la liberación vigorosa de vapor de agua. Esta pérdida de humedad es la precursora del proceso de tostado y rigidización superficial, desencadenado por el complejo químico de Maillard. En este punto, la síntesis entre aminoácidos y azúcares presentes en el tubérculo define las propiedades organolépticas finales mediante la aplicación de energía térmica sostenida.
- **Etapa de Cocción:** Es la etapa más larga del proceso, donde se desarrollan las características organolépticas deseadas en las papas fritas, como el sabor, el

color y la textura. Durante esta fase, el interior de las rodajas de papa se cocina completamente, mientras que la corteza se forma y se vuelve más crujiente. A medida que el calor penetra en el interior del tubérculo, las moléculas de almidón se hinchan y gelatinizan, dando como resultado una textura suave y esponjosa. La formación de la corteza crujiente se debe a la deshidratación de la superficie del alimento, donde el agua se evapora rápidamente, dejando una capa exterior crocante y apetecible.

- **Etapa final o "punto final de burbujeo":** En esta etapa final del proceso de fritura, se completa la pérdida de humedad en las rodajas de papa, ya que éstas dejan de mantener líquido en su interior o se reduce significativamente la transferencia de calor entre la corteza y el interior del tubérculo. Una vez que se alcanza el nivel de cocción deseado, determinado por factores como el grosor de las rodajas, la temperatura del aceite y el tiempo de fritura, el alimento se retira cuidadosamente del aceite(Montes O. et al., 2016).

Figura 11

Proceso de fritura



Nota. La figura muestra las etapas de fritura de una papa. Adaptado de *Oil absorption in fried foods* (Montes O. et al., 2016).

CAPÍTULO III: DISEÑO Y ENTRENAMIENTO DE VISIÓN ARTIFICIAL

En este capítulo se presenta la metodología aplicada para el desarrollo del sistema de alerta basado en visión artificial. Se utilizará la metodología en V, la cual permite estructurar el proceso en etapas de análisis, diseño e implementación. Aquí se abordará la elección del hardware y software necesario, así como las fases de entrenamiento del modelo, integración del sistema embebido y validación del prototipo en condiciones simuladas de uso real.

3.1. Metodología

La metodología en V aplicada en este proyecto permite dividir el desarrollo en dos fases principales: una línea descendente orientada al análisis y diseño del sistema, y una línea ascendente enfocada en la implementación, pruebas y validación. Esta estructura asegura que cada componente del sistema, tanto de hardware como de software, sea diseñado y verificado de forma coherente con los requerimientos establecidos para el sistema de visión artificial embebido.

3.1.1. Modelo en V

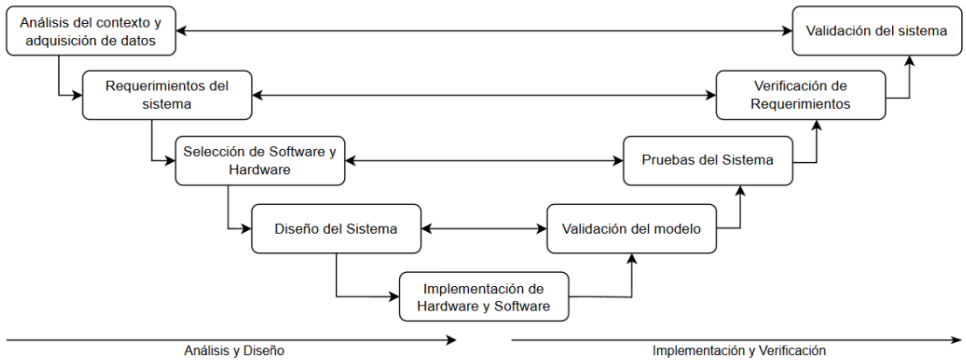
Según (Wu, 2023) la metodología en V es un modelo de desarrollo ampliamente utilizado en ingeniería de sistemas embebidos, caracterizado por su estructura simétrica que vincula cada fase de diseño con una correspondiente etapa de verificación o validación, por otro lado en el contexto de sistemas habilitados con inteligencia artificial, esta metodología ha sido objeto de adaptaciones que buscan preservar su trazabilidad, mientras se incorporan elementos como la experimentación iterativa y el ajuste de modelos.

En la Figura 12 se muestra el modelo en V adaptado para este proyecto, el cual se basa en la propuesta de Wu (2023), la cual introduce una estructura extendida para proyectos que integran componentes con aprendizaje automático. No obstante, para este proyecto de visión artificial en sistemas embebidos, se han incorporado únicamente las fases pertinentes,

manteniendo la trazabilidad y verificación entre diseño e implementación que caracteriza a este enfoque.

Figura 12

Etapas del modelo en V



Nota. Este modelo está adaptado al contexto de la creación de sistemas basados en visión artificial de (Wu, 2023).

En la primera fase del modelo en V de la Figura 12 se contempla el análisis del contexto y adquisición de datos, etapa fundamental para definir el problema y su situación actual para la obtención de más información. A partir de este análisis, se establecen los requerimientos del sistema, los cuales guían la selección de hardware y software, y posteriormente, el diseño estructurado del sistema, incluyendo tanto la arquitectura física como lógica, y la integración del modelo de visión artificial.

La segunda fase del modelo en V, correspondiente a la fase de implementación y verificación, inicia con la implementación del hardware y software, seguida por la validación del modelo entrenado, evaluando su precisión y comportamiento embebido. Según (Wu, 2023), las siguientes etapas comprenden la verificación de requerimientos, pruebas del sistema y la validación final, asegurando que cada módulo y el sistema completo cumplen con las funciones previstas y se comportan de forma consistente bajo condiciones reales.

Como lo señala (Sosa-Savedra et al., 2023), cada fase del modelo en V cumple un rol clave dentro del proceso de validación progresiva, permitiendo identificar errores de diseño

desde etapas tempranas y garantizando trazabilidad entre los requisitos planteados y la solución final. Esta metodología resulta adecuada para el desarrollo del sistema de alerta mediante visión artificial propuesto, ya que permite integrar de manera controlada los componentes físicos, el procesamiento de datos visuales y la lógica de inferencia del modelo de inteligencia artificial.

3.2. Análisis

En la industria alimentaria, uno de los factores que más influye en la calidad final del producto es el control preciso del proceso de cocción, especialmente en alimentos fritos como los snacks de papa.

3.2.1. Situación Actual

Diversos estudios han evidenciado que el error humano durante la supervisión visual del proceso de fritura puede conducir a una sobrecocción del producto, generando alteraciones en el color, textura y sabor, así como la formación de compuestos no deseados.

Según (Pedreschi et al., 2005) estos errores pueden provocar variaciones significativas en el contenido de humedad, la absorción de aceite y el grado de dorado, lo que afecta directamente la percepción del consumidor y la aceptabilidad comercial. Además, la falta de estandarización en la evaluación visual genera lotes inconsistentes, comprometiendo la calidad y la trazabilidad. Por ello, resulta necesario implementar sistemas automáticos de monitoreo que reduzcan la dependencia del criterio humano y permitan una detección objetiva del punto óptimo de fritura, lo cual es el propósito del presente proyecto.

3.2.2. Dimensionamiento de Stakeholders

Primeramente, se realiza el dimensionamiento de los stakeholders que se encuentran implicados en el proyecto y que se necesitan para el desarrollo del sistema, a continuación, se muestran en la Tabla 2.

Tabla 2

Stakeholders

Stakeholders	
Usuarios Directos	UTN
Usuarios Indirectos	Ing. Margarita Calderón
Administradores	Ing. Santiago Ruales
Director y Fiscalizadores	Msc. Edgar Maya
	Msc. Luis Suarez
Desarrollador	Sr. Pablo Muñoz

Fuente. Autoría propia

3.2.3. Fuentes y Métodos para la Generación del Dataset

La construcción de un sistema de alerta utilizando visión artificial requiere de un dataset representativo que refleje con precisión las condiciones reales del proceso de fritura. Según (Szeliski,2022) un dataset bien estructurado y obtenido bajo condiciones controladas es esencial para garantizar un rendimiento fiable del sistema en contextos prácticos. En este proyecto, se recurrió a diversas fuentes y métodos de recolección de datos visuales y contextuales con el fin de construir un dataset que represente adecuadamente las distintas etapas del proceso de fritura de papas.

Entre los métodos aplicados se incluyen:

- **Observación directa:** Es una técnica que permite extraer datos de forma metódica mediante la revisión y análisis de situaciones reales. A través de esta herramienta se puede identificar comportamientos, condiciones o patrones relevantes dentro del entorno de estudio, sin intervenir en el proceso observado.
- **Captura fotográfica y videográfica:** Este método permite documentar visualmente las características de los objetos o situaciones estudiadas. La información obtenida

mediante imágenes y videos es fundamental para alimentar sistemas de inteligencia artificial que requieren entrenamiento basado en reconocimiento visual.

- **Entrevistas al personal operativo:** Técnica cualitativa que consiste en una conversación planificada entre el investigador y los participantes del entorno, con el objetivo de obtener información profunda y detallada sobre experiencias, criterios o conocimientos relacionados con el fenómeno en estudio.
- **Encuestas al personal de producción:** Herramienta cuantitativa que recopila información estandarizada mediante un cuestionario aplicado a un grupo definido de personas.

En este proyecto se utilizaron las técnicas de observación directa en planta, la entrevista y encuesta al personal como técnicas principales, las cuales constituyen herramientas clave para recolectar información relevante que contribuya al desarrollo del sistema propuesto.

3.2.3.1. Observación directa en planta.

Se realizó un monitoreo estructurado del área de fritura dentro de la empresa "Delicias de mi Tierra", registrando los diferentes niveles de cocción de las papas fritas. Esta observación permitió identificar visualmente las etapas crítica del proceso (inicio, intermedia y sobrecocción), sirviendo como base para clasificar las imágenes que integran el dataset.

3.2.3.2. Entrevista y encuesta al personal operativo.

Se realizaron entrevistas y encuestas al personal de fritura para conocer los criterios utilizados al determinar el punto óptimo de cocción y detectar los problemas frecuentes de sobrecocción, información clave para definir los parámetros del sistema. Debido a ello se realizaron las siguientes preguntas:

- ¿Qué características visuales observa para determinar que una papa está bien cocida?
- ¿Cuáles son los errores más comunes que ocurren durante la fritura?
- ¿Cómo identifica visualmente una papa sobrecocida?

- ¿Utiliza algún criterio de color, textura o tiempo para evaluar el estado de cocción?
- ¿Cuánto influye la experiencia del operario en la calidad del producto final?
- ¿Considera que el proceso actual es propenso a errores humanos? ¿Por qué?
- ¿Qué consecuencias trae una sobrecocción en términos de calidad y desperdicio?
- ¿Con qué frecuencia se presentan errores en el punto de cocción durante una jornada de trabajo?

Tras la realización de la entrevista en la empresa "Delicias de mi Tierra", se presenta el resultado en el Anexo 1, correspondiente al Sr. Santiago Ruales, jefe de producción. Adicionalmente, se elaboró una encuesta dirigida al personal del área de fritura, con preguntas de opción múltiple, cuyo contenido se detalla en el Anexo 2 y cuyos resultados se encuentran en el Anexo 3.

3.3. Requerimientos

Para la definición de los requerimientos del sistema de alerta basado en visión artificial, se tomó como referencia el estándar IEEE 29148, el cual establece directrices claras para la especificación de necesidades funcionales y no funcionales. Dichos requerimientos contemplan las funcionalidades esenciales del sistema, así como sus restricciones operativas, tecnológicas y de rendimiento, considerando las limitaciones del dispositivo embebido utilizado (*ISO/IEC/IEEE 29148:2011*, 2011).

3.3.1. Nomenclatura de requerimientos

La Tabla 3 presenta las abreviaturas empleadas con el fin de facilitar la comprensión y organización de la información, en correspondencia con los requerimientos definidos para el sistema.

Tabla 3

Abreviatura de los requerimientos

Abreviaturas	Requerimiento
StSR	Requerimiento de Stakeholders
SySR	Requerimientos del Sistema
SRSH	Requerimientos de Arquitectura

3.3.2. Requerimientos de los Stakeholders

La Tabla 4 resume los requerimientos definidos por los stakeholders, los cuales sirven de base para el desarrollo del sistema.

Tabla 4

Requerimientos de Stakeholders

#	Requerimientos	Prioridad		
		Alta	**Media**	**Baja**
	Requerimientos Operacionales			
StSR1	Disponibilidad de un punto de energía eléctrica para alimentar el dispositivo ESP32-S3.	X		
StSR2	Acceso a red Wi-Fi estable			X
StSR3	Espacio físico adecuado para el montaje del sensor de cámara y toma de imágenes.	X		
	Requerimientos de Usuarios			
StSR4	Las alertas generadas deben ser comprensibles para el operario mediante mensajes o colores.	X		
StSR5	Visualización clara del estado de cocción mediante una alerta visual.		X	
StSR6	El sistema debe operar en tiempo real sin interrumpir el flujo normal de trabajo.	X		

3.3.3. Requerimientos del sistema

Los requerimientos del sistema definen las condiciones técnicas necesarias para el correcto funcionamiento del prototipo. Estos incluyen tanto especificaciones de hardware, como el uso de un microcontrolador con cámara integrada, como requerimientos de software,

relacionados con el procesamiento de imágenes, generación de alertas y comunicación en tiempo real como muestra la Tabla 5.

Tabla 5

Requerimientos del sistema

#	Requerimientos	Prioridad		
		Alta	**Media**	**Baja**
	Requerimientos de Uso			
SySR1	El dispositivo debe permanecer encendido durante todo el proceso de monitoreo.	X		
SySR2	El modelo de inteligencia artificial debe ejecutarse para iniciar el análisis de imágenes.	X		
SySR3	El sistema debe generar alertas automáticas ante casos de sobrecocción detectada.	X		
	Requerimientos de Interfaces			
SySR4	El sistema debe estar conectado a una cámara funcional para capturar las imágenes.	X		
SySR5	Debe contar con puertos de E/S para conexión de periféricos como relés o módulos Wi-Fi.	X		
	Requerimientos de Estados			
SySR6	El microcontrolador debe estar en estado operativo para iniciar el sistema.	X		
SySR7	La cámara debe estar encendida para permitir la detección del nivel de cocción.	X		
SySR8	El sistema debe mantenerse en ejecución continua durante el proceso de fritura.	X		
	Requerimientos de Físicos			
SySR9	El dispositivo debe ubicarse en un área con acceso a energía eléctrica para su funcionamiento.	X		
SySR10	Se debe contar con un espacio adecuado que permita capturar correctamente las imágenes del proceso.	X		
SySR11	El entorno de instalación debe contar con condiciones de iluminación adecuadas para asegurar la calidad de las imágenes capturadas.	X		

3.3.4. Requerimientos de arquitectura

Los requerimientos de arquitectura definen la estructura general del sistema, incluyendo la distribución de sus componentes físicos y lógicos. En este proyecto, se establece

una arquitectura distribuida, compuesta por un dispositivo embebido con cámara integrada para la captura y procesamiento de imágenes para el análisis de los datos recolectados.

Tabla 6

Requerimientos de arquitectura

#	Requerimientos	Prioridad		
		Alta	**Media**	**Baja**
	Requerimientos Lógicos			
SRSH1	El software debe ser compatible con el microcontrolador.	X		
SRSH2	El hardware debe permitir la integración fluida entre cámara, red y procesamiento.	X		
	Requerimientos de Hardware			
SRSH3	El sistema debe contar con una cámara	X		
SRSH4	El dispositivo debe tener conexión Wi-Fi integrada.		X	
SRSH5	Debe incluir puertos de E/S para conexión con periféricos.	X		
SRSH6	El procesador debe permitir la inferencia en tiempo real de modelos ligeros de visión artificial.	X		
	Requerimientos de Software			
SRSH7	El sistema debe ser compatible con plataformas	X		
SRSH8	El firmware debe estar optimizado para bajo consumo de RAM y CPU.	X		
SRSH9	Debe soportar bibliotecas ligeras para procesamiento de imágenes.	X		
SRSH10	Debe ser de libre uso.	X		
	Requerimientos Eléctricos			
SRSH11	El sistema debe alimentarse mediante fuente USB 5V compatible con el microcontrolador.	X		
SRSH12	La alimentación eléctrica debe ser estable para garantizar funcionamiento continuo.	X		

3.3.5. Propósito General y Aplicación del Sistema

Se propone el desarrollo de un sistema de visión artificial embebido, orientado a detectar el punto de cocción de papas fritas en una planta de producción alimenticia. El sistema se basa en el reconocimiento visual de distintas etapas del proceso de fritura (ebullición, deshidratación, cocción y sobrecocción), utilizando un microcontrolador ESP32-S3 Sense con

cámara integrada. A partir de un dataset construido con imágenes reales capturadas durante el proceso, se entrena un modelo de aprendizaje automático capaz de emitir alertas cuando se detecte el punto óptimo de cocción. El sistema está diseñado para funcionar en tiempo real en condiciones del área de fritura.

3.3.6. Descripción General del Sistema

El sistema propuesto consiste en un prototipo de visión artificial embebido que permite detectar el nivel de cocción de papas fritas en tiempo real. Está conformado por un microcontrolador ESP32-S3 Sense con cámara integrada, encargado de capturar imágenes durante el proceso de fritura. Estas imágenes son procesadas mediante un modelo previamente entrenado que clasifica las etapas del producto (ebullición, deshidratación, cocción y sobrecocción). En caso de detectar el punto óptimo de cocción o sobrecocción respectivamente, el sistema genera una alerta visual. El diseño está optimizado para operar en entornos industriales con recursos computacionales limitados y puede integrarse fácilmente a líneas de producción alimenticia.

3.4. Elección y Uso de Plataformas de Software

En esta etapa del proyecto se considera la evaluación de diferentes plataformas de software que permitan el desarrollo de un sistema de visión artificial eficiente, compatible con dispositivos de recursos limitados.

3.4.1. Desarrollo del Sistema mediante Plataforma Visual

Una vez definidos los requerimientos del sistema, se procede a analizar y seleccionar la plataforma de software más adecuada para su desarrollo. Se prioriza el uso de herramientas visuales que permitan crear modelos de visión artificial de forma intuitiva, y que sean compatibles con dispositivos de recursos limitados.

Las características de las plataformas consideradas se detallan en el Anexo 4, las cuales servirán como base para su análisis comparativo. Con el fin de seleccionar la opción de

software más adecuada para el proyecto, en la Tabla 7 se presenta una evaluación alineada con los requerimientos previamente establecidos.

Tabla 7

Comparación de plataformas de software acorde a los requerimientos

Requerimientos	Edge Impulse	Matlab
SRSH7	1	1
SRSH8	1	1
SRSH9	1	1
SRSH10	1	0
Valoración	4	3
Cumplimiento: 0 – No cumple 1 – Si cumple		

Fuente. Autoría propia

A partir del análisis comparativo presentado en la Tabla 7, se determina que la plataforma Edge Impulse es la opción más adecuada para el proyecto, al cumplir con la totalidad de los requerimientos establecidos (SRSH7 a SRSH10). Su compatibilidad con el sistema embebido, soporte para despliegue en tiempo real y entorno visual accesible la posicionan como la alternativa más eficiente frente a la plataformas de software Matlab que se utiliza en proyectos más complejos y que tiene un costo por su licencia.

3.4.2. Algoritmo

Una vez definida la plataforma de desarrollo, se procede a evaluar los algoritmos disponibles en Edge Impulse para la clasificación de imágenes. La elección se basa en criterios como precisión, eficiencia de inferencia y compatibilidad con dispositivos de recursos limitados. En la Tabla 8 se presenta un análisis comparativo de los modelos disponibles, donde se determina el más adecuado para el sistema propuesto acorde a las características mencionadas en el Anexo 5, que servirán para analizar los requerimientos.

Tabla 8

Comparación de algoritmos acorde a los requerimientos

Requerimientos	MobileNet SSD FPN-Lite	FOMO (MobileNetV2)	YOLO v5
SRSH7	1	1	1
SRSH8	0	1	0
SRSH9	1	1	1
SRSH10	1	1	1
Valoración	3	4	3
Cumplimiento: 0 – No cumple 1 – Si cumple			

Fuente. Autoría propia

A partir del análisis mostrado en la Tabla 8, se selecciona el algoritmo FOMO (MobileNetV2) por cumplir con la totalidad de los requerimientos establecidos. Su bajo consumo de memoria, alta velocidad de inferencia y compatibilidad con dispositivos con recursos computacionales limitados lo convierten en la opción más adecuada frente a MobileNet SSD FPN-Lite y YOLO v5.

3.5. Elección del Hardware

La selección del hardware se realiza considerando los requerimientos definidos por los stakeholders, así como las necesidades del sistema y su arquitectura. Para ello, se analizan diferentes opciones disponibles, evaluando sus características técnicas, compatibilidad con el modelo entrenado y viabilidad económica, con el fin de determinar el dispositivo más adecuado para implementar el prototipo.

3.5.1. Análisis Comparativo

Para llevar a cabo la comparación, se consideran las características técnicas de los componentes de hardware, las cuales se detallan en el Anexo 6.

3.5.1.1. Sistema Embebido.

A continuación, en la Tabla 9 se presenta una comparación entre los microcontroladores XIAO ESP32S3 Sense, Raspberry Pi 4 y Arduino Nicla Vision, considerando sus características técnicas y su grado de adaptación a los requerimientos del proyecto.

Tabla 9

Comparación de microcontroladores

Parámetros	XIAO ESP32S3 Sense	Raspberry Pi 4	Arduino Nicla Vision
Costo	$30.00	$89.90	$114.90
Procesador	Xtensa Dual-Core 240 MHz	Quad-Core Cortex-A72 1.5 GHz	Arm® Cortex-M7 480 MHz
Memoria RAM	512 KB SRAM + 8 MB PSRAM	1–8 GB (según modelo)	2 MB SRAM
Almacenamiento	Vía QSPI Flash (programable)	microSD o eMMC	16 MB Flash
Wi-Fi / Bluetooth	Wi-Fi 802.11 b/g/n + BLE 5.0	Wi-Fi (con adaptador) + BT 5.0	Wi-Fi y BLE integrados
Cámara integrada	Sí (OV2640 de 2 MP)	No (requiere módulo externo)	Sí (HM01B0 monocromo)
GPU / Aceleración gráfica	No	Sí (VideoCore VI)	No
Soporte para IA	Sí (Edge Impulse, TensorFlow Lite)	Sí (Python, OpenCV, TensorFlow, etc.)	Sí (OpenMV, TensorFlow Lite)
Consumo energético	Bajo	Alto	Bajo
Tamaño	(21 x 17.5 mm)	(85.6 x 56.5 mm)	(22.86 x 22.86 mm)

Fuente. Autoría propia

Tras analizar las opciones disponibles, se determina que el procesador Xtensa Dual-Core del XIAO ESP32S3 Sense es el más adecuado para el proyecto, ya que ofrece un equilibrio óptimo entre rendimiento, bajo consumo energético y compatibilidad con modelos

de inteligencia artificial ligeros. Su arquitectura está optimizada para sistemas embebidos, permite ejecutar inferencias en tiempo real mediante frameworks como Edge Impulse o TensorFlow Lite, y cuenta con 8 MB de PSRAM que facilitan el procesamiento de imágenes sin requerir hardware adicional. A diferencia del Raspberry Pi 4, que posee mayor potencia, pero consume más energía y espacio, o del Arduino Nicla Vision, que es más costoso y menos flexible, el ESP32-S3 resulta ideal por integrar una cámara, soportar procesamiento visual eficiente y cumplir con los requerimientos técnicos definidos para este sistema de visión artificial. Debido a ello se realiza la siguiente comparación de acuerdo con los requerimientos en la Tabla 10.

Tabla 10

Comparación de microcontroladores acorde a los requerimientos

Requerimientos	**XIAO ESP32S3 Sense**	**Raspberry Pi 4**	**Arduino Nicla Vision**
SRSH3	1	0	1
SRSH4	1	0	1
SRSH5	1	1	0
SRSH6	1	1	1
SySR7	1	1	1
Valoración	5	3	4
Cumplimiento: 0 – No cumple 1 – Si cumple			

Fuente. Autoría propia

Según los resultados presentados en la Tabla 10, el XIAO ESP32S3 Sense cumple con la totalidad de los requerimientos evaluados, obteniendo la mayor puntuación frente al Raspberry Pi 4 y al Arduino Nicla Vision. Su compatibilidad con cámara integrada, puertos de E/S, capacidad de procesamiento en tiempo real y eficiencia energética lo posicionan como la opción más adecuada para el desarrollo del sistema propuesto. A diferencia del Raspberry Pi 4, que presenta limitaciones en integración directa de periféricos embebidos, y del Nicla

Vision, que carece de puertos accesibles para expansión, el ESP32S3 ofrece un equilibrio entre funcionalidad, tamaño compacto y consumo optimizado, cumpliendo con los criterios definidos para la arquitectura del sistema.

3.6. Diseño del Sistema

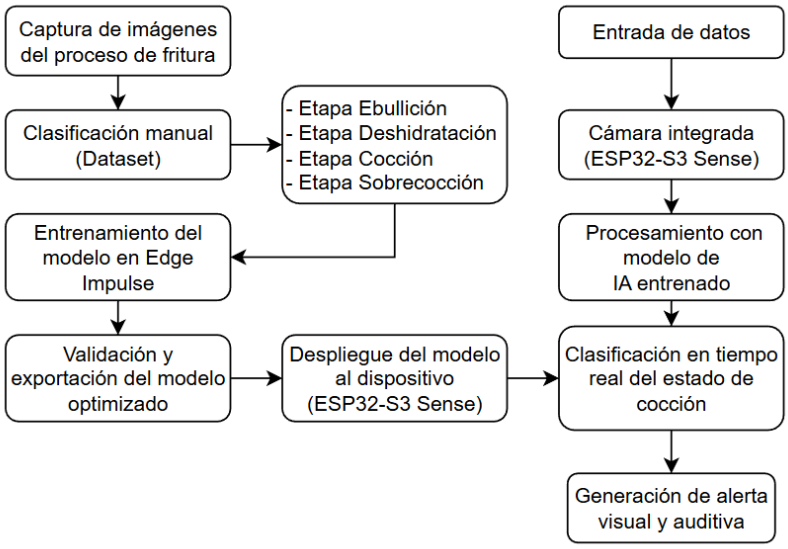
El diseño del sistema contempla la integración de los componentes seleccionados tanto a nivel de hardware como de software, con el objetivo de implementar un prototipo funcional que permita detectar visualmente el punto de cocción de las papas en tiempo real. Este diseño abarca la disposición del dispositivo embebido mediante diagramas, la estructura lógica del modelo de visión artificial y la generación de alertas en función del análisis realizado.

3.6.1. Diagrama de bloques del sistema

El diagrama de bloques del sistema inicia desde la captura de imágenes del proceso de fritura hasta la generación de alertas visuales y auditivas. Según la Figura 13 en la fase de entrenamiento, las imágenes son clasificadas manualmente según las etapas de cocción (ebullición, deshidratación, cocción y sobrecocción), lo que permite entrenar y validar un modelo en Edge Impulse mediante una arquitectura CNN. Este modelo es luego implementado en un ESP32-S3 Sense, donde clasifica en tiempo real el estado de cocción a partir de las imágenes capturadas por su cámara integrada, y genera una alerta visual y auditiva si se detecta el punto óptimo de cocción.

Figura 13

Diagrama de Bloques del Sistema



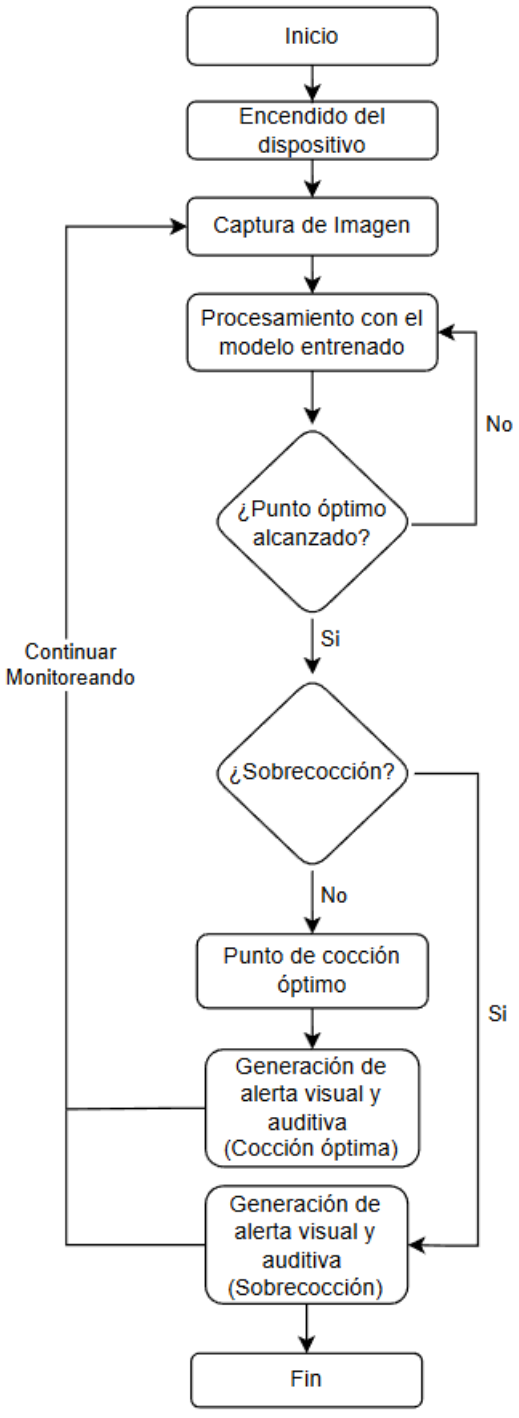
Fuente. Autoría propia

3.6.2. Diagrama de funcionamiento del sistema

El diagrama de la Figura 14 representa el flujo lógico del funcionamiento del sistema embebido encargado de detectar el punto óptimo de cocción en papas fritas utilizando visión artificial. El proceso inicia con el encendido del dispositivo, seguido por la captura de una imagen del snack papa frita en tiempo real. Esta imagen es procesada por el modelo de inteligencia artificial previamente entrenado, el cual evalúa si el estado visual del alimento corresponde al punto óptimo de cocción. Si el sistema determina que aún no se ha alcanzado esta etapa, el ciclo se repite, permitiendo un monitoreo continuo. En caso de que se detecte el punto óptimo de cocción o una sobrecocción, se genera una alerta visual y auditiva para advertir al operador, y el flujo finaliza. Este diseño asegura una supervisión constante y automática del proceso de fritura sin intervención humana directa.

Figura 14

Diagrama de funcionamiento del Sistema



Fuente. Autoría propia

3.6.3. Diagrama de Arquitectura del sistema

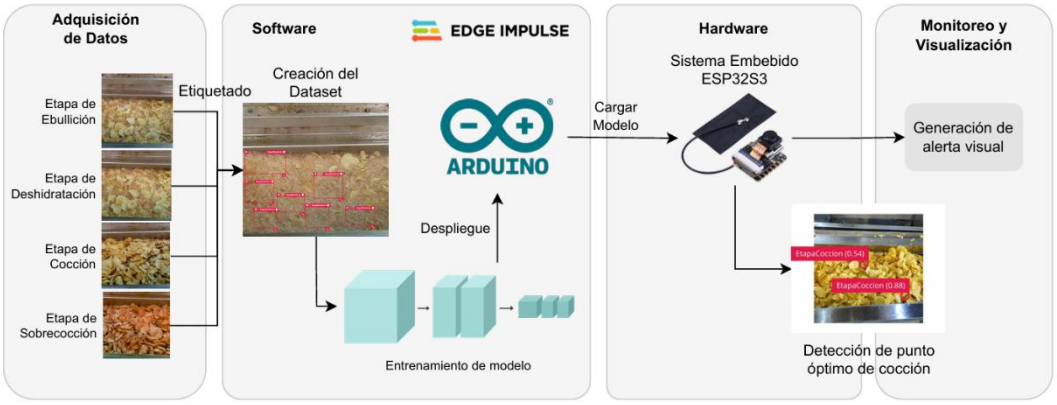
La arquitectura del sistema se divide en cuatro bloques principales como se observa en la Figura 15: adquisición de datos, software, hardware, monitoreo y visualización, permitiendo una implementación integrada y funcional de visión artificial para la detección del punto óptimo de cocción de papas fritas.

1. **Adquisición de Datos**: En esta etapa se recopilan imágenes reales del proceso de fritura, específicamente de las etapas de ebullición y cocción. Estas imágenes se utilizan como base para construir el dataset que alimentará el modelo de inteligencia artificial. La recolección es realizada directamente desde el entorno real de trabajo para asegurar que el sistema aprenda bajo condiciones reales.
2. **Software (Edge Impulse):** Las imágenes recopiladas son etiquetadas de acuerdo con su estado de cocción, creando un dataset organizado. Luego, este dataset es cargado en la plataforma Edge Impulse, donde se entrena un modelo de clasificación basado en una red neuronal convolucional (CNN). Durante el entrenamiento, el modelo aprende a identificar visualmente las distintas etapas de cocción. Una vez validado, el modelo optimizado se despliega mediante Arduino y se genera el archivo para ser cargado en el microcontrolador.
3. **Hardware (ESP32-S3 Sense)**: El modelo entrenado se implementa en un sistema embebido ESP32-S3 con cámara integrada, el cual es responsable de capturar imágenes en tiempo real del proceso de fritura. Estas imágenes son analizadas localmente por el modelo de IA, lo que permite detectar si el producto ha alcanzado el punto óptimo de cocción.
4. **Monitoreo y Visualización**: Cuando el sistema identifica que la imagen corresponde al punto óptimo de cocción, se genera una alerta visual y auditiva. Esta alerta puede visualizarse a través de un LED de color respectivo, indicando al operario que el

producto está listo para ser retirado, reduciendo así errores humanos y garantizando la calidad del producto final.

Figura 15

Diagrama de arquitectura del sistema



Fuente. Autoría propia

3.6.4. Entrenamiento y Despliegue del Modelo de Visión Artificial

A continuación, se lleva a cabo el entrenamiento del modelo de visión artificial utilizando la plataforma Edge Impulse, seleccionada conforme a los requerimientos del sistema. En primer lugar, se construyó un dataset con imágenes capturadas directamente en el entorno real de fritura. Estas fueron etiquetadas manualmente según su etapa de cocción y empleadas para entrenar un modelo ligero basado en la arquitectura FOMO. Una vez validado, el modelo optimizado fue exportado e implementado en el microcontrolador ESP32-S3 Sense, donde se realizaron pruebas iniciales que permitieron verificar su funcionamiento en tiempo real bajo condiciones reales de operación.

3.6.4.1. Edge Impulse.

Es una plataforma de desarrollo de software enfocada en la implementación de modelos de inteligencia artificial en dispositivos embebidos. Permite crear, entrenar y desplegar modelos de aprendizaje automático a partir de datos capturados directamente desde sensores, como cámaras, acelerómetros o micrófonos. Su interfaz gráfica facilita la construcción de

modelos sin necesidad de programar desde cero, lo que la hace ideal para proyectos con recursos computacionales limitados. Además, Edge Impulse ofrece soporte nativo para microcontroladores como el ESP32-S3, optimizando modelos mediante técnicas de cuantización y redes livianas como FOMO, lo que garantiza un rendimiento eficiente en tiempo real.

3.6.4.2. FOMO.

FOMO es un modelo de visión artificial diseñado por Edge Impulse para ejecutar detección de objetos en tiempo real sobre dispositivos embebidos de bajos recursos (como el ESP32-S3) (da Silva et al., 2023). A diferencia de los detectores tradicionales como YOLO o SSD, FOMO no utiliza bounding boxes, sino que realiza detección a través de un enfoque grid-based (por celdas), lo que permite reducir drásticamente el consumo de memoria y aumentar la velocidad de inferencia.

FOMO utiliza como backbone una versión reducida de MobileNetV2, encargada de extraer características relevantes de la imagen a distintos niveles de abstracción mediante convoluciones profundas. En lugar de predecir coordenadas de cajas delimitadoras como en los modelos tradicionales, FOMO divide la imagen en una cuadrícula de celdas (por ejemplo, 96x96 px → 12x12 celdas), donde cada celda actúa como un detector local. La red genera un mapa de activación por clase, en el que cada valor representa la probabilidad de que un objeto esté presente en esa celda. Si la probabilidad supera un umbral definido, se considera que el objeto ha sido detectado en esa región, permitiendo realizar detecciones múltiples con gran eficiencia sin necesidad de bounding boxes.

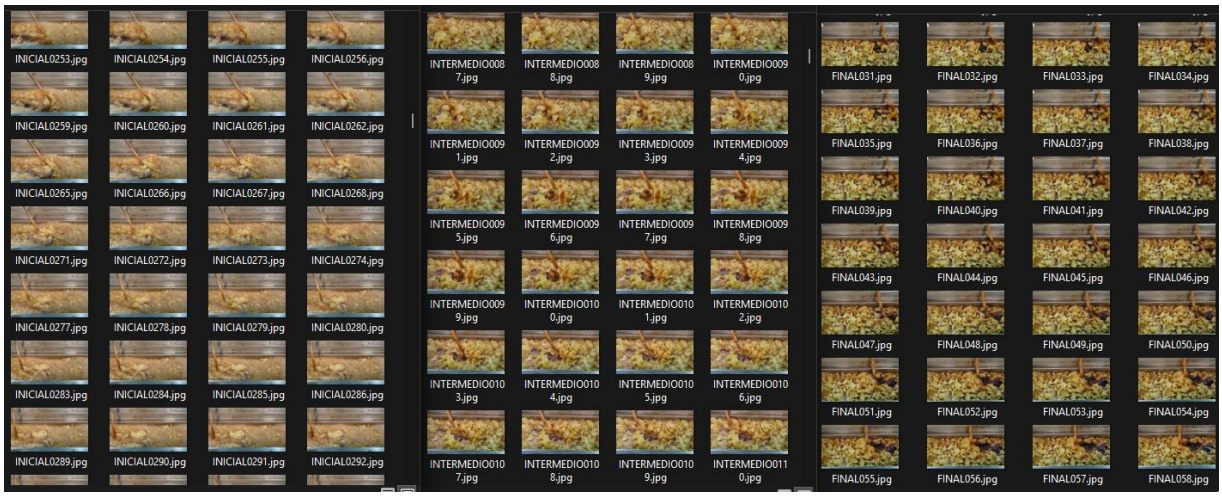
3.6.4.3. Adquisición de Imágenes.

El primer paso fundamental para el entrenamiento del modelo de visión artificial es la recolección de imágenes que conformarán el dataset. Para ello, se capturaron un total de 3000 imágenes reales del proceso de fritura, representando diferentes etapas de fritura. Sin embargo,

para el entrenamiento del modelo se usaron más de 1000 imágenes como se observa en la Figura 16. Estas imágenes fueron organizadas y etiquetadas manualmente según su clase correspondiente. Del total, el 20% fue destinado a la fase de testeo del modelo, mientras que el resto se utilizó para entrenamiento y validación. Como parte del preprocesamiento, todas las imágenes fueron reducidas a una resolución de 96x96 píxeles, en concordancia con los requisitos de entrada del modelo FOMO utilizado en Edge Impulse.

Figura 16

Adquisición de imágenes para Dataset



Fuente. Autoría propia

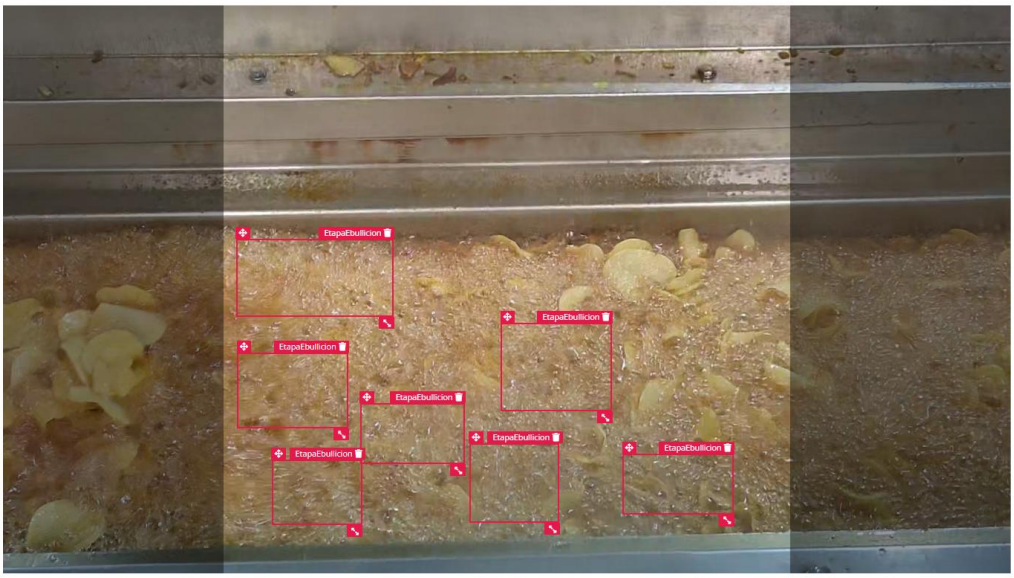
3.6.4.4. Etiquetado y armado de Dataset.

El proceso de etiquetado y armado del dataset se llevó a cabo directamente en la plataforma Edge Impulse, donde se definieron las clases correspondientes a cada etapa del proceso de fritura como muestra la Figura 17. En primer lugar, durante el proceso de etiquetado en la etapa de ebullición, se decidió marcar exclusivamente las burbujas presentes en la superficie del aceite, ya que estas representan el fenómeno visual más distintivo y constante de esta fase inicial de fritura. A nivel técnico, las burbujas generan un patrón visual de alto contraste y textura granular que puede ser identificado fácilmente por la red convolucional durante el entrenamiento. Etiquetar toda el área o incluir zonas sin burbujas podría introducir

ruido visual en el dataset, disminuyendo la precisión del modelo y dificultando la generalización de la clase. Por lo tanto, se optó por centrarse en las regiones donde el comportamiento visual del aceite (burbujeo activo) es más evidente, lo cual mejora la capacidad del modelo para distinguir con claridad esta etapa de otras fases más avanzadas, como la deshidratación, la cocción o sobrecocción.

Figura 17

Etiquetado en Etapa de Ebullición.

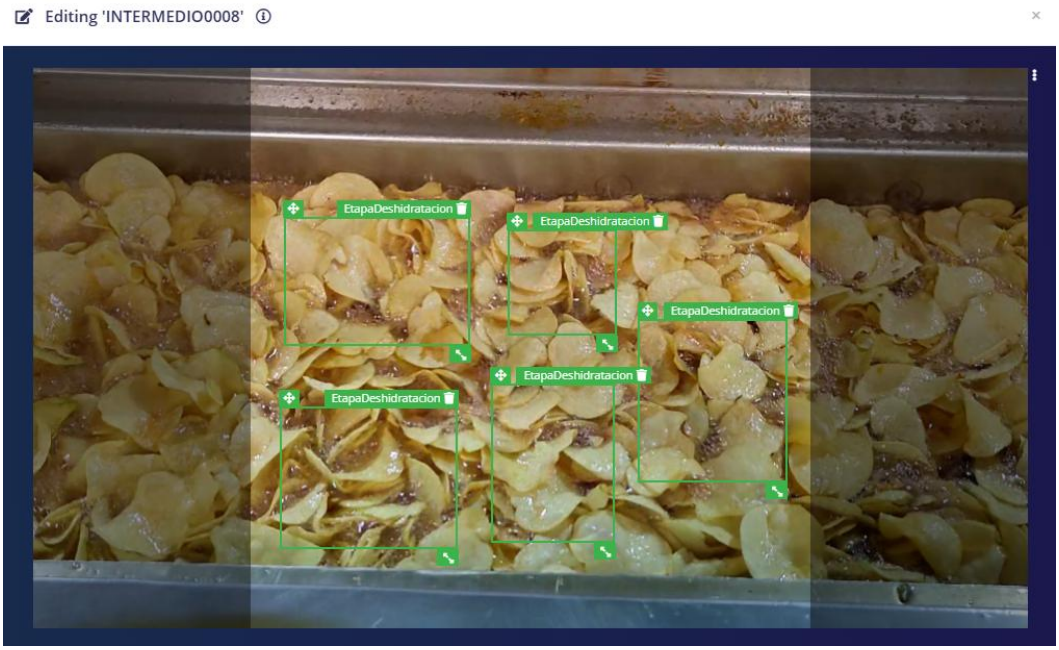


Fuente. Autoría propia

En la etapa de deshidratación se etiquetaron aquellas papas que, aunque ya han emergido a la superficie del aceite, aún presentan una tonalidad clara y homogénea, sin señales visuales de cocción completa como se observa en la Figura 18. Esta selección se basó en la intención de capturar el momento intermedio del proceso, donde las papas ya no están completamente sumergidas como en la ebullición, pero tampoco han adquirido aún la textura y coloración amarillenta característicos de la etapa de cocción.

Figura 18

Etiquetado en Etapa de deshidratación

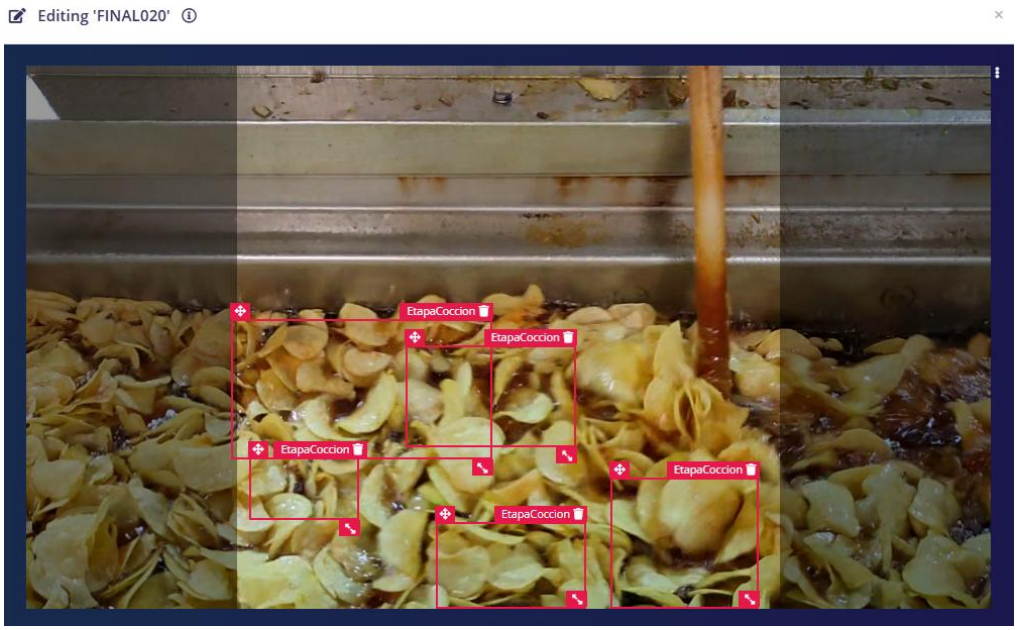


Fuente. Autoría propia

En el proceso de etiquetado correspondiente a la etapa de cocción, se incluyeron no solo las papas con tonalidad amarilla intensa y textura crujiente visible, sino también las áreas oscuras del aceite donde ya no se observan burbujas activas como lo evidencia la Figura 19. Esta decisión se tomó de manera intencional y fundamentada, ya que la ausencia de burbujeo en dichas zonas representa una señal visual clave que coincide con la finalización del proceso de fritura. A nivel técnico, estas regiones fueron consideradas parte del patrón de cocción óptima porque indican que la transferencia de humedad desde el interior del alimento hacia el aceite ha concluido, lo que concuerda con el estado de cocción deseado. Subjetivamente, al observar estas condiciones de forma repetida durante la toma de imágenes, se reconoció que la combinación entre el aspecto del producto y el comportamiento del aceite ofrecía una referencia confiable y coherente para el etiquetado. Esta estrategia permitió mejorar la capacidad del modelo para identificar de manera precisa el punto óptimo sin generar ambigüedad con las etapas anteriores.

Figura 19

Etiquetado en Etapa de Cocción óptima

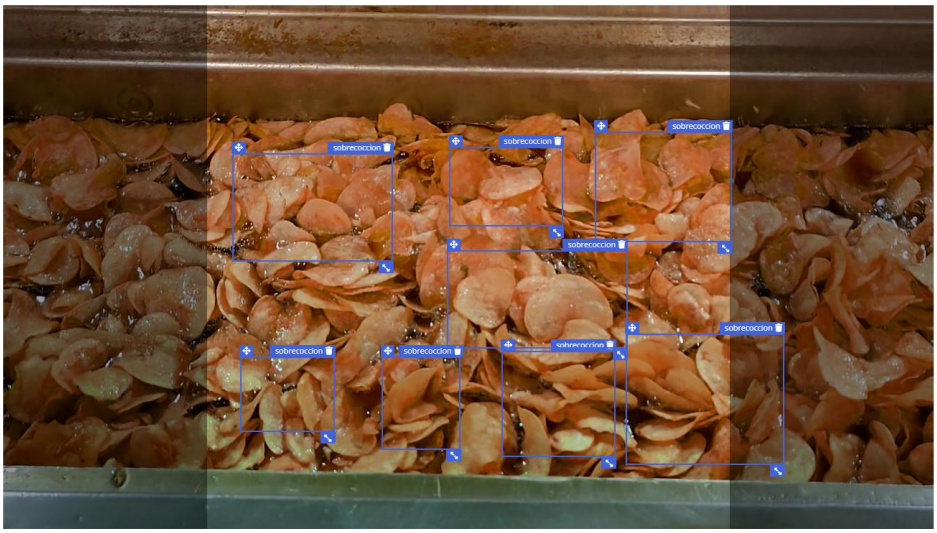


Fuente. Autoría propia

Luego se etiquetaron las papas en etapa de sobrecocción debido a las claras evidencias visuales de un estado avanzado de fritura. Se observa en la Figura 20 un color notablemente más oscuro en comparación con las fases anteriores del proceso, con tonos que van desde el dorado intenso hasta el marrón, lo que indica una exposición prolongada al calor. Además, la textura de las papas muestra signos de rigidez y arrugamiento excesivo, características propias de una pérdida de humedad más allá del punto óptimo de cocción. Estas condiciones permiten identificar de manera precisa las regiones sobrecosidas, justificando así la ubicación de los cuadros delimitadores sobre aquellas zonas donde predominan estas características visuales.

Figura 20

Etiquetado en Etapa de Sobrecocción



Fuente. Autoría propia

3.6.4.5. Entrenamiento de modelo.

Una vez completado el proceso de etiquetado y armado del dataset, se procede al entrenamiento del modelo de visión artificial utilizando la plataforma Edge Impulse. El dataset, compuesto por imágenes clasificadas en cuatro etapas de cocción (ebullición, deshidratación, cocción y sobrecocción), fue cargado y dividido automáticamente por la plataforma en conjuntos de entrenamiento (aproximadamente 80%) y validación (20%). Para el modelo se seleccionó la arquitectura FOMO (Faster Objects, More Objects), ideal para aplicaciones en dispositivos embebidos debido a su bajo consumo de recursos y alta eficiencia en la detección por regiones. El modelo fue configurado con una resolución de entrada de 96x96 píxeles y un número definido de épocas de entrenamiento, utilizando funciones de pérdida y optimización propias del entorno Edge Impulse. Durante el entrenamiento, se monitorearon métricas como la precisión, la tasa de error y la confusión de acuerdo con la Figura 21, lo que permitió evaluar el rendimiento del modelo en la clasificación de cada época.

Figura 21

Entrenamiento de modelo

```
Splitting data into training and validation sets OK

Training model...
Training on 664 inputs, validating on 166 inputs
Training model...
Training on 664 inputs, validating on 166 inputs
Trained 14 batches.

Epoch   Train    Validation
        Loss     Loss    Precision Recall F1
   00   2.42592  1.11733 0.00      0.00   0.00
2.9% trained.

Epoch   Train    Validation
        Loss     Loss    Precision Recall F1

Epoch   Train    Validation
        Loss     Loss    Precision Recall F1
   01   0.77207  0.57256 0.50      0.00   0.01
```

Fuente. Autoría propia

3.6.4.6. Parámetros de configuración del entrenamiento.

Para la red convolucional del modelo a entrenar es necesario configurar ciertos parámetros que se detallan a continuación. Para el entrenamiento del modelo FOMO en la plataforma Edge Impulse, se utilizaron una serie de parámetros ajustados para equilibrar eficiencia, precisión y compatibilidad con dispositivos embebidos como el ESP32-S3 Sense. Uno de los principales parámetros es alpha=0.35, que controla la "anchura" del modelo MobileNetV2 utilizado como base. Este valor reduce el número de filtros en las capas convolucionales, disminuyendo el tamaño del modelo y su consumo de recursos, sin sacrificar significativamente su capacidad de aprendizaje.

En la Figura 22 el parámetro input_shape se fijó en 96x96 píxeles con 3 canales, ya que FOMO requiere entradas cuadradas y este tamaño representa un buen compromiso entre detalle visual y carga computacional. El entrenamiento se realizó con un total de 60 épocas (EPOCHS), lo que permitió al modelo ajustarse progresivamente a las características de cada clase sin caer en sobreajuste. Se empleó un learning_rate de 0.001, valor estándar para el

optimizador Adam, que posteriormente se explicará a más detalle, que garantiza una convergencia estable del modelo durante la actualización de pesos.

Figura 22

Parámetros para configurar del modelo

```
195
196 EPOCHS = args.epochs or 60
197 LEARNING_RATE = args.learning_rate or 0.001
198 BATCH_SIZE = args.batch_size or 32
199
200 model = train(num_classes=classes,
201                learning_rate=LEARNING_RATE,
202                num_epochs=EPOCHS,
203                alpha=0.35,
204                object_weight=100,
205                train_dataset=train_dataset,
206                validation_dataset=validation_dataset,
207                best_model_path=BEST_MODEL_PATH,
208                input_shape=MODEL_INPUT_SHAPE,
209                batch_size=BATCH_SIZE,
210                use_velo=False,
211                ensure_determinism=ensure_determinism)
212
```

Fuente. Autoría propia

El batch_size se configuró en 32, lo cual determina cuántas imágenes se procesan en cada iteración de entrenamiento. Esta cantidad es ideal para mantener eficiencia en dispositivos con memoria limitada. Además, se utilizó un object_weight de 100 en la función de pérdida, lo que significa que las regiones etiquetadas como objeto (por ejemplo, papas en cocción) tienen un peso mucho mayor que el fondo, obligando al modelo a priorizar la detección de objetos relevantes.

Como se observa en la Figura 23, para el proceso de entrenamiento del modelo se utilizó el optimizador Adam (Adaptive Moment Estimation), ampliamente reconocido por su eficiencia y estabilidad en redes neuronales convolucionales. Adam combina las ventajas de dos métodos clásicos de optimización: AdaGrad, que adapta el aprendizaje a cada parámetro, y RMSProp, que ajusta dinámicamente la tasa de aprendizaje en función de las medias móviles de los gradientes. Esta combinación permite que Adam realice actualizaciones más precisas y rápidas, especialmente en datasets ruidosos o problemas complejos como la detección de

objetos (Kingma & Ba, 2014). En este proyecto, se configuró una tasa de aprendizaje inicial de 0.001, valor que ofrece un equilibrio adecuado entre velocidad de convergencia y estabilidad durante el ajuste de los pesos del modelo.

Figura 23

Optimizador Adam

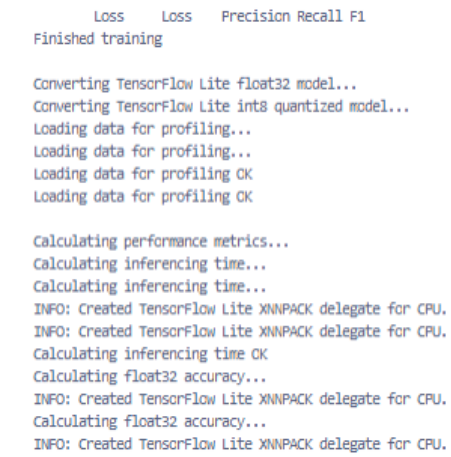
```
validation_dataset_for_callback = (validation_dataset
    .batch(batch_size, drop_remainder=False)
    .prefetch(prefetch_policy))

#! Initialise bias of final classifier based on training data prior.
util.set_classifier_biases_from_dataset(
    model, train_segmentation_dataset)

if not use_velo:
    model.compile(loss=weighted_xent,
                  optimizer=Adam(learning_rate=learning_rate))

#! Create callback that will do centroid scoring on end of epoch against
#! validation data. Include a callback to show % progress in slow cases.
callbacks = callbacks if callbacks else []
callbacks.append(metrics.CentroidScoring(validation_dataset_for_callback,
                                         output_width_height, num_classes_with_background))
callbacks.append(metrics.PrintPercentageTrained(num_epochs))

```

Fuente. Autoría propia

Al concluir el entrenamiento, Edge Impulse realiza una serie de procesos automáticos para optimizar y evaluar el rendimiento del modelo. Primero, como muestra la Figura 24 se generan dos versiones en formato TensorFlow Lite: una en precisión float32 y otra cuantizada en int8, esta última ideal para su ejecución en microcontroladores por su bajo consumo de memoria y mayor velocidad. Posteriormente, se cargan datos de prueba para perfilar el modelo y calcular métricas clave como precisión, recall y F1-score. También se mide el tiempo de inferencia, tanto en modelos estándar como cuantizados, para asegurar su viabilidad en entornos embebidos. Finalmente, se utiliza el delegado XNNPACK para ejecutar inferencias optimizadas en CPU. Según (TensorFlow, 2025) al usar XNNPACK permite que el modelo se ejecute más rápido en CPU, lo que resulta ideal para simular condiciones reales de funcionamiento en dispositivos donde no hay GPU ni aceleradores de IA, permitiendo estimar el comportamiento del modelo en condiciones reales de implementación.

Figura 24

Acciones finales del entrenamiento



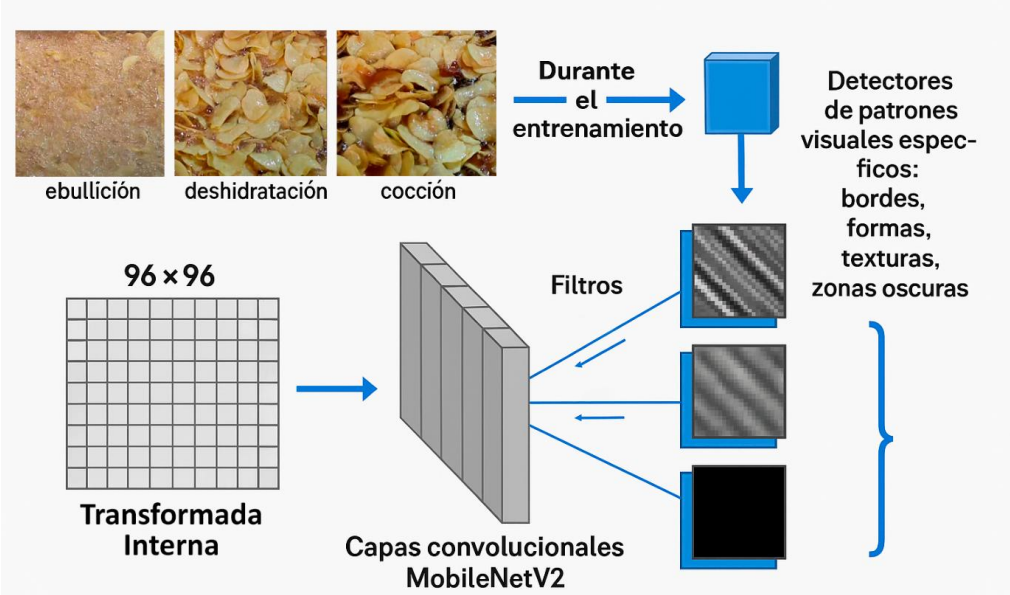
Fuente. Autoría propia

3.6.4.7. Procesos internos del entrenamiento del modelo.

Durante el entrenamiento, el modelo recibe como entrada las imágenes del dataset, cada una con una etiqueta que indica en qué etapa de cocción se encuentra (ebullición, deshidratación, cocción o sobrecocción). Según la Figura 25 cada imagen de tamaño 96x96 píxeles con 3 canales de color, es transformada internamente en una matriz de valores numéricos que representan la intensidad de color de cada píxel. Estas matrices se procesan a través de múltiples capas convolucionales de la red MobileNetV2, donde cada capa tiene cientos de filtros (neuronas convolucionales) que actúan como detectores de patrones visuales específicos, como bordes, formas, texturas o zonas oscuras.

Figura 25

Proceso de entrenamiento



Fuente. Autoría propia

A medida que una imagen atraviesa la red neuronal convolucional, esta es transformada en múltiples representaciones internas progresivamente más abstractas, conocidas como mapas de activación. En este caso, como se muestra en la Figura 26 se ha utilizado una versión ligera de MobileNetV2 configurada con alpha = 0.35, lo cual reduce considerablemente el número total de parámetros, situándolo entre aproximadamente 300,000 y 500,000 pesos entrenables, distribuidos en decenas de capas y miles de filtros. Al aplicar un include_top=True y truncar la red en el bloque block_6_expand_relu, se genera una reducción espacial de la imagen de entrada de 96×96 píxeles a una salida de 12×12, debido a una reducción por un factor de 8. Por tanto, el tamaño del mapa de activación resultante es 12×12×N, donde N representa el número de clases que el modelo puede predecir por celda.

El tamaño de la salida se obtiene según la Ecuación (3):

Ecuación (3)

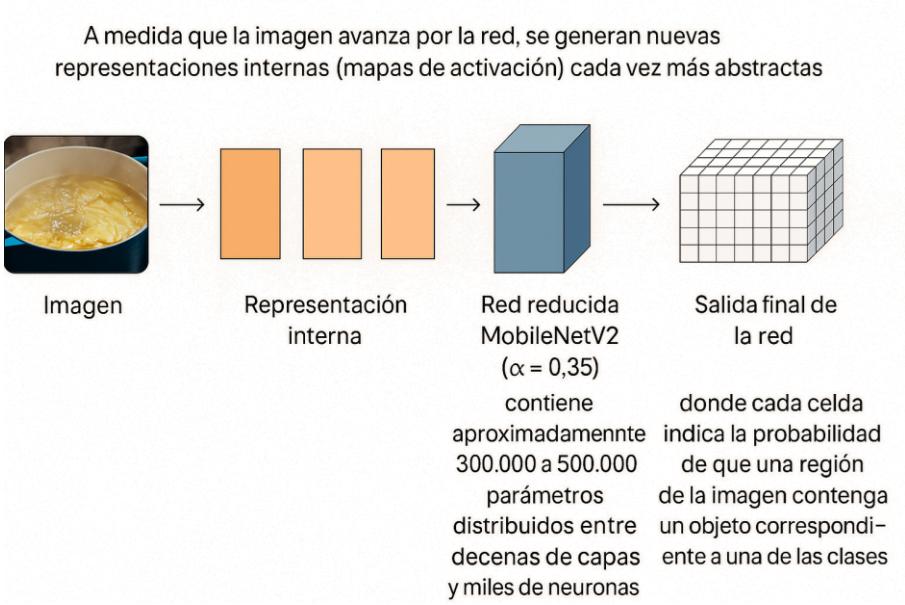
$$\frac{96}{8} \times \frac{96}{8} = 12 \times 12$$

Entonces se genera un mapa de activación de 12x12 celdas. Y complementando lo anterior 12x12x4, el "4" representa el número de clases que el modelo puede predecir por celda:

- Etapa de ebullición
- Etapa de deshidratación
- Etapa de cocción
- Etapa de sobrecocción

Figura 26

Representación esquemática de la salida final del modelo FOMO

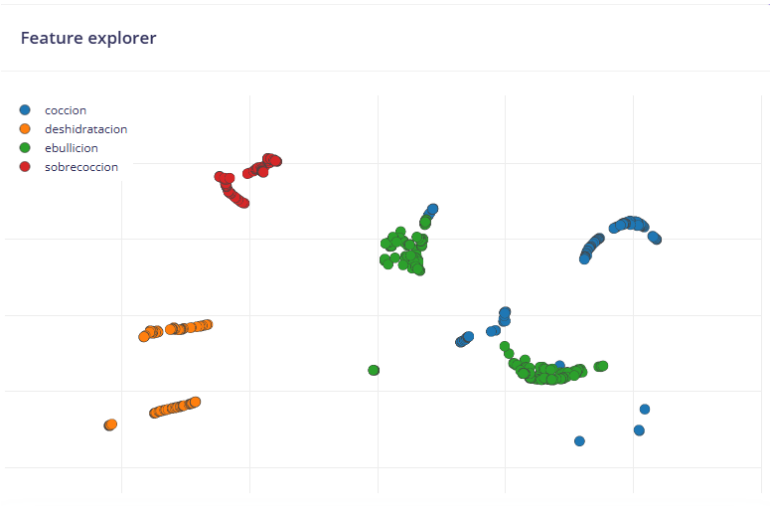


Fuente. Autoría propia

En la Figura 27 se muestra la gráfica presentada en el Feature Explorer que evidencia la distribución de las características extraídas mediante técnicas de reducción de dimensionalidad, representando visualmente cómo el modelo interpreta y agrupa los datos correspondientes a cada fase del proceso de fritura: ebullición, deshidratación, cocción y sobrecocción. En primer lugar, se observa una clara separación entre las muestras etiquetadas, lo cual indica que el modelo ha aprendido a distinguir patrones únicos asociados a estas etapas. Esta separación sugiere que las características visuales como color, textura o nivel de burbujeo difieren lo suficiente para permitir una clasificación robusta entre estas fases.

Figura 27

Exploración de características



Fuente. Autoría propia

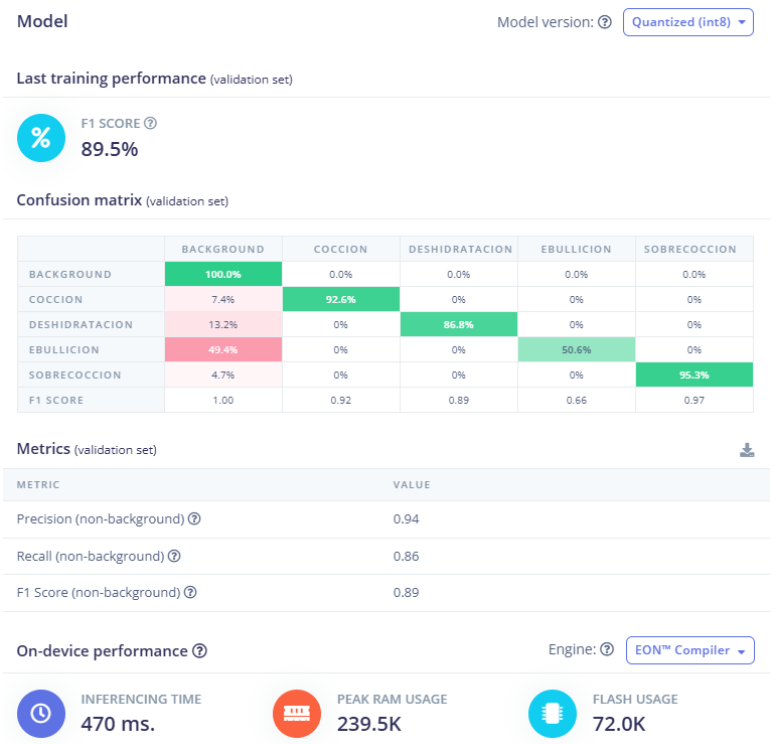
Durante el proceso de entrenamiento, el modelo compara lo que predice con la respuesta correcta (etiqueta real), utilizando una función matemática llamada entropía cruzada (cross-entropy) para medir qué tan lejos está de acertar. Luego, mediante un proceso llamado retropropagación, ajusta los valores internos de sus neuronas con el objetivo de reducir ese error. Este procedimiento se repite muchas veces, permitiendo que la red aprenda a identificar con precisión las diferencias visuales entre las distintas fases del proceso de fritura. Además, gracias al uso del optimizador Adam, que acelera y estabiliza el aprendizaje, y a la arquitectura ligera de FOMO, el modelo entrenado puede ejecutarse de forma eficiente en un microcontrolador de bajos recursos, reconociendo en tiempo real cuándo una papa ha alcanzado su punto de cocción ideal.

Como se observa en la Figura 28 muestra el resultado de entrenamiento que corresponde a la versión cuantizada (int8) del modelo, lo cual permite una ejecución más eficiente en dispositivos embebidos como el Xiao ESP32S3 Sense. En términos generales, se observa un F1 Score global del 89.5 %, lo cual refleja un desempeño altamente competitivo para un sistema de visión artificial ejecutado en microcontroladores de bajos recursos. Sin

embargo, al analizar la matriz de confusión y las métricas asociadas, se pueden identificar fortalezas y áreas que aún requieren mejora.

Figura 28

Proceso final de entrenamiento



Fuente. Autoría propia

En primer lugar, el modelo logra una identificación perfecta del fondo (background), con un 100 % de acierto y un F1 Score de 1.00. Esto es positivo, ya que evita falsos positivos en zonas donde no hay producto. Además, se evidencia una excelente clasificación en las etapas de cocción (92.6 % de acierto) y sobrecocción (95.3 %), ambas con F1 Scores superiores a 0.9. Esto sugiere que los patrones visuales de estas clases están bien definidos y el modelo ha logrado aprenderlos de forma consistente.

El análisis del desempeño en la fase de ebullición revela los desafíos intrínsecos del entorno operativo. El acierto del 50.6 % y un F1 Score de 0.66 se atribuyen principalmente a la alta densidad de vapores de aceite y la turbulencia del fluido, factores que generan una firma visual muy similar al fondo (background) de la freidora. Esta mimetización técnica dificulta la

extracción de características únicas por parte del modelo en los instantes iniciales; no obstante, el sistema logra mantener la detección de la fase, demostrando su capacidad de operación bajo condiciones de ruido visual extremo, lo cual es un hito considerando los limitados recursos computacionales del hardware utilizado.

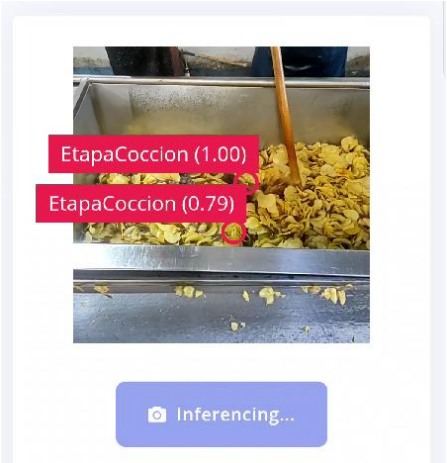
En cuanto al rendimiento en el dispositivo, se destaca un tiempo de inferencia reducido de 470 ms el cual en la práctica puede cambiar, con un uso de RAM de 239.5 KB y de memoria Flash de 72.0 KB, lo que confirma que el modelo ha sido correctamente optimizado para ejecutarse en tiempo real dentro de las restricciones de un microcontrolador. Esto ha sido posible gracias a la cuantización int8 y al uso del EON Compiler, que reduce considerablemente el peso del modelo sin comprometer demasiado su rendimiento.

3.6.4.8. Reconocimiento del punto de cocción.

Una vez completado el entrenamiento del modelo, se procedió a realizar pruebas preliminares para verificar su funcionamiento en la detección del punto óptimo de cocción de las papas fritas. La Figura 29 representa una prueba en tiempo real del modelo ya entrenado, y permite observar cómo se comporta la inferencia cuando se aplica a una imagen del entorno real, específicamente en la detección de papas fritas en etapa de cocción.

Figura 29

Punto de cocción reconocido por modelo entrenado



Fuente. Autoría propia

El modelo ha identificado dos regiones en la imagen donde ha clasificado visualmente papas como pertenecientes a la clase "EtapaCocción". Los valores entre paréntesis (0.1 y 0.79) indican la probabilidad de certeza con la que el modelo realiza dicha predicción. Un valor de 0.1, por ejemplo, indica un alto nivel de confianza de que esa región corresponde al punto óptimo de cocción.

Las etiquetas no están delimitadas por cuadros grandes (como en YOLO), sino por pequeñas marcas circulares. Esto es característico de FOMO, que identifica la clase presente en una celda específica de la imagen sin generar bounding boxes, lo que reduce el consumo de memoria y acelera el proceso.

3.6.4.9. Diseño físico y encapsulado del sistema de alerta.

Con el fin de validar el funcionamiento del sistema de alerta basado en visión artificial en un entorno real de operación, se desarrolló un prototipo físico que integra el módulo ESP32-S3 Sense junto con los elementos electrónicos de señalización y alerta como se observa en la Figura 30. El diseño del prototipo consideró de manera prioritaria las condiciones adversas presentes en el área de fritura, tales como la exposición a vapor de aceite, altas temperaturas y operación continua.

El dispositivo fue montado dentro de una carcasa protectora diseñada para proporcionar un alto grado de hermeticidad, minimizando el ingreso de partículas de aceite y humedad que pudieran afectar el desempeño del sistema embebido. Esta carcasa permite alojar de forma segura el módulo de procesamiento, el sistema de alimentación y los dispositivos periféricos, garantizando estabilidad mecánica y protección.

Figura 30

Encapsulado y protección del ESP32-S3 Sense

Fuente. Autoría propia.

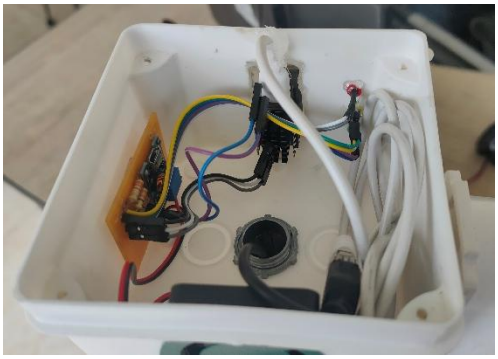
Para permitir la correcta adquisición de imágenes, se incorporó una ventana frontal transparente alineada con la cámara integrada del ESP32-S3 Sense. Esta ventana fue diseñada específicamente para no interferir con el campo de visión del sensor, permitiendo la observación directa del interior de la freidora y asegurando condiciones adecuadas para la ejecución del modelo de visión artificial durante las diferentes etapas del proceso de fritura.

Adicionalmente, el sistema integra un conjunto de indicadores visuales mediante diodos emisores de luz (LEDs), los cuales permiten al operario identificar de manera inmediata el estado del proceso. Un LED de color amarillo indica la detección de las etapas de ebullición y deshidratación; un LED verde señala que el sistema ha identificado el punto óptimo de cocción; mientras que un LED rojo advierte que el producto ha ingresado en una etapa de sobrecocción. Como complemento a la señalización visual, se implementó una sirena acústica ubicada en uno de los laterales del dispositivo, destinada a emitir una alerta sonora cuando se detecta una condición crítica. Este mecanismo de notificación auditiva permite alertar al operario incluso cuando no se mantiene contacto visual directo con el sistema, reforzando la seguridad y reduciendo la dependencia de la supervisión humana constante.

Debido a las condiciones térmicas del entorno, el prototipo incorpora un sistema de refrigeración activa mediante un ventilador, cuyo propósito es disipar el calor generado tanto por el dispositivo como por el ambiente circundante como se observa en la Figura 31. Este sistema contribuye a mantener la temperatura de operación dentro de rangos seguros, mejorando la estabilidad y la vida útil del equipo electrónico.

Figura 31

Sistema embebido usado para el Sistema de Alerta



Fuente. Autoría propia.

CAPÍTULO IV: EVALUACIÓN DE RESULTADOS

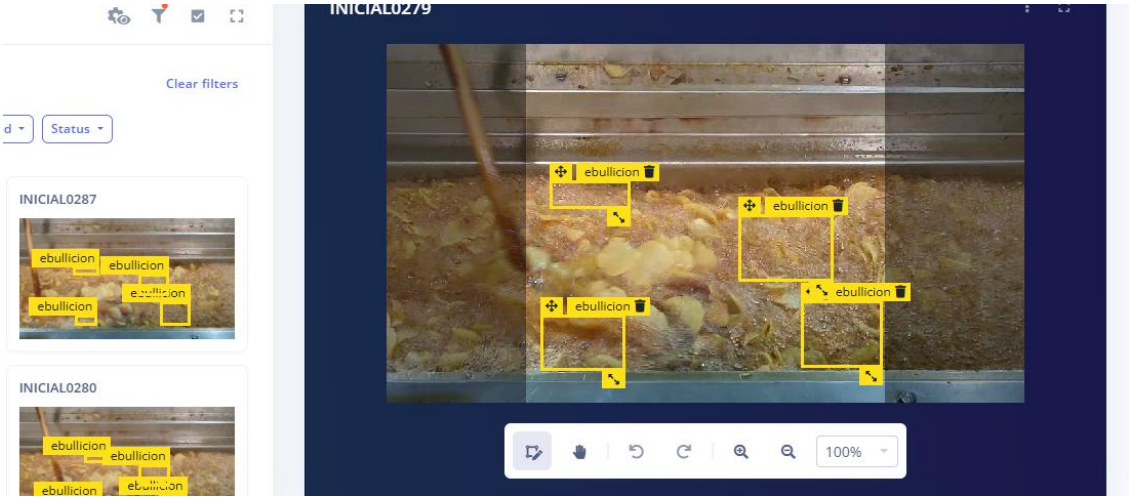
Este capítulo presenta el desarrollo e implementación del sistema propuesto, resultado de la investigación y análisis detallado realizado en los capítulos anteriores. A partir del planteamiento del problema y la fundamentación teórica, se diseñó una solución tecnológica basada en técnicas de visión artificial y aprendizaje profundo, adaptada a las limitaciones de hardware propias de dispositivos embebidos. En esta sección se describen los procesos de integración de los componentes, el entrenamiento y ajuste de los modelos de detección, así como las pruebas realizadas en entorno real. El objetivo principal es demostrar el rendimiento, precisión y eficiencia del sistema diseñado para cumplir con los requerimientos funcionales y operativos definidos al inicio del proyecto.

4.1. Adquisición de datos

Para el entrenamiento del modelo de visión artificial, se construyó un conjunto de datos compuesto por imágenes capturadas durante el proceso de fritura de papas en diferentes etapas de cocción. Como muestra la Figura 32, se registraron un total de más de 1000 imágenes, distribuidas en cuatro clases: etapa inicial, deshidratación, cocción y sobrecocción. Las imágenes fueron tomadas con la cámara de un celular, en condiciones de iluminación natural y controlada en el entorno real de producción. Las capturas se realizaron con resolución de 640x480 píxeles en formato JPEG, y posteriormente fueron etiquetadas manualmente en la plataforma de Edge Impulse, para garantizar la coherencia de las clases. Este conjunto de datos sirvió como base para el entrenamiento del modelo en la plataforma Edge Impulse, permitiendo evaluar el comportamiento del sistema con entradas visuales reales y representativas del proceso industrial. Para posteriormente una vez entrenado el modelo, optimizarlo y agregar demás funcionalidades, para desplegarlo mediante Arduino IDE en el microcontrolador como se describe en el Anexo 7.

Figura 32

Dataset en Edge Impulse



Fuente. Autoría propia.

4.1.1. Caracterización Empírica del Proceso de Fritura

Previo a la fase de validación del sistema, se llevó a cabo una caracterización empírica del proceso de fritura con el fin de establecer parámetros reales de referencia bajo condiciones operativas industriales. Esta etapa fue fundamental para definir los umbrales temporales y térmicos que posteriormente servirían como base para la segmentación de las etapas de cocción empleadas por el sistema de visión artificial.

El levantamiento de información se realizó mediante la observación directa de 20 muestras consecutivas de fritura, ejecutadas manualmente por el operario con mayor experiencia en el área de producción. En la Tabla 11 durante este procedimiento se registraron para cada muestra, el tiempo total de cocción, así como la temperatura inicial y final del aceite, variables que influyen de manera directa en el color, textura y grado de fritura del producto final.

Los datos obtenidos, presentados en la Tabla 11, evidencian que el proceso de fritura no se desarrolla bajo condiciones estrictamente constantes, sino que presenta variaciones inherentes al entorno industrial, tales como fluctuaciones térmicas del aceite y diferencias en

el tiempo de exposición. A pesar de estas variaciones, se identifica un rango temporal recurrente en el que el producto alcanza un estado de fritura considerado óptimo por criterios operativos y sensoriales del área de producción.

Tabla 11

Tiempos de cocción y variabilidad térmica observada

Nro Muestra	Tiempo Cocción (seg)	Temperatura Inicial (°C)	Temperatura Final (°C)
1	198.00	202	176
2	182.50	203	181
3	198.60	202	179
4	183.00	204	184
5	192.60	201	178
6	181.00	201	182
7	192.90	200	178
8	189.00	204	176
9	198.10	202	181
10	194.50	203	178
11	187.49	204	176
12	199.01	201	174
13	194.64	203	175
14	191.97	201	177
15	183.12	203	178
16	183.12	204	177
17	181.16	200	182
18	197.32	203	175
19	192.02	201	177
20	194.16	204	174

Fuente. Autoría propia.

El análisis de los tiempos registrados muestra que la mayoría de las muestras se concentran en un intervalo aproximado comprendido entre 180 s y 200 s, mientras que las temperaturas finales del aceite se mantienen dentro de un rango relativamente estable, con valores típicos entre 174 °C y 184 °C. Esta consistencia permite inferir que, bajo condiciones normales de operación, el punto óptimo de cocción se encuentra asociado a la combinación de estos rangos de tiempo y temperatura.

4.2. Pruebas del sistema

Una vez integrado el modelo entrenado en el microcontrolador XIAO ESP32-S3 Sense, se llevaron a cabo pruebas funcionales para evaluar su desempeño en condiciones reales del proceso de fritura en la Procesadora de Alimentos “Delicias de mi Tierra”.

4.2.1. Plan de Pruebas

Para validar el funcionamiento del sistema de visión artificial, en la Tabla 12 se realiza un plan de pruebas prácticas orientadas a verificar la capacidad del modelo para identificar correctamente las distintas etapas del proceso de fritura de papas. Estas pruebas incluyeron la captura y clasificación de imágenes en tiempo real correspondientes a las fases de ebullición, deshidratación, cocción y sobrecocción. Cada etapa fue registrada bajo condiciones reales en el área de fritura de la empresa Procesadora de Alimentos “Delicias de mi Tierra”, con el objetivo de evaluar su rendimiento, precisión y eficacia en la detección del punto de cocción adecuado.

Tabla 12

Plan de Pruebas del sistema

Etapa del proceso	Descripción	Resultado esperado
Ebullición	Inicio del proceso de fritura, se observan burbujas constantes debido a la evaporación del agua superficial.	El sistema detecta visualmente la presencia de burbujas y clasifica la imagen como fase de ebullición.
Deshidratación	Reducción progresiva de la humedad interna; las burbujas disminuyen y la superficie comienza a opacarse.	El sistema identifica la disminución de burbujas y textura intermedia, clasificando como fase de deshidratación.
Cocción	Color dorado homogéneo y textura crujiente, sin señales	El sistema reconoce características visuales

	de quemado o dureza extrema.	óptimas y clasifica la imagen como cocción ideal.
Sobrecocción	Color marrón oscuro o negro, textura muy rígida y señales de carbonización.	El sistema detecta cambios extremos de color y clasifica como sobrecocción.

Fuente. Autoría propia.

Durante la etapa experimental se realizaron múltiples ensayos con el objetivo de verificar el desempeño y la estabilidad del sistema en condiciones reales de operación. Para la presentación de resultados en este documento, se seleccionaron cinco pruebas mediante muestreo aleatorio simple, con el fin de mostrar datos representativos sin introducir sesgo en la selección.

La representatividad de estas cinco pruebas se respalda mediante el análisis de la variabilidad del conjunto total de ensayos, empleando el coeficiente de variación (CV%), definido como:

Ecuación (4)

$$CV(\%) = \frac{s}{\bar{x}} \times 100$$

Donde $\bar{x}$ es la media muestral y s la desviación estándar.

Para las 20 muestras obtenidas manualmente en el área de fritura a un operario de mayor experiencia, se calcularon los siguientes valores:

Tiempo de cocción

$$\bar{x} = 190.71\, s$$

$$s = 6.39\, s$$

$$CV = \frac{6.39}{190.71} \times 100 = 3.35\%$$

Temperatura inicial

$$\bar{x} = 202.30°C$$

$$s = 1.38°C$$

$$CV = \frac{1.38}{202.30} \times 100 = 0.68\%$$

Temperatura final

$$\bar{x} = 177.90°C$$

$$s = 2.83°C$$

$$CV = \frac{2.83}{177.90} \times 100 = 1.59\%$$

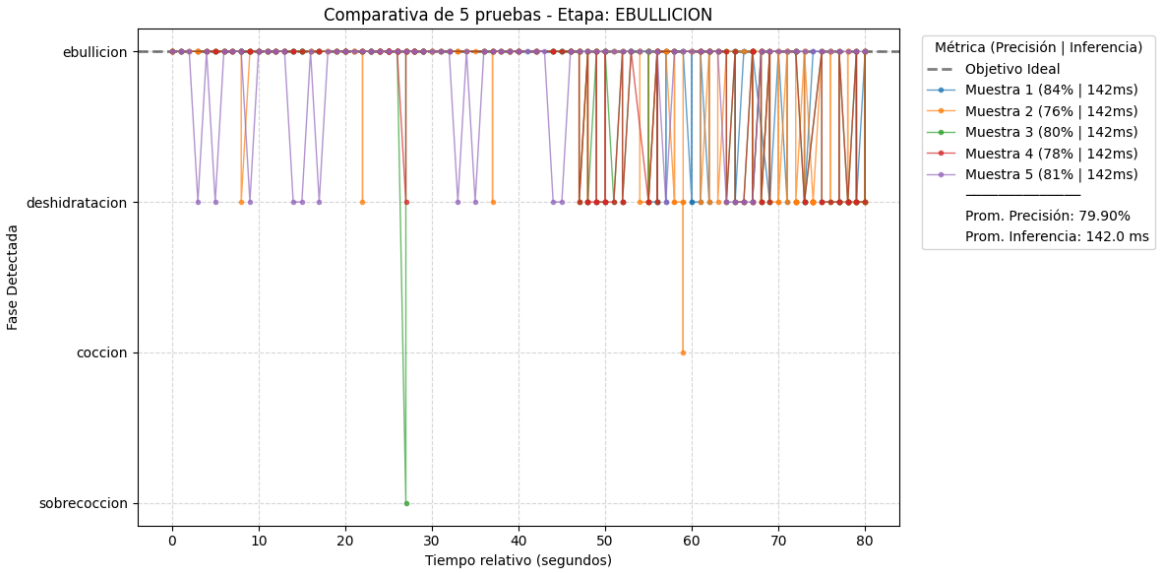
Los valores obtenidos son inferiores al 5%, lo que indica baja dispersión y alta estabilidad del proceso. En consecuencia, la presentación de cinco pruebas es técnicamente suficiente para ilustrar el desempeño del sistema, dado que el comportamiento global ya se encuentra caracterizado y no presenta variabilidad significativa. Las pruebas restantes se mantienen como respaldo experimental.

4.2.2. Pruebas de etapa de Ebullición

Para el análisis automatizado de los resultados, se empleó un script desarrollado en Python, descrito detalladamente en el Anexo 8. Este script extrae información desde los archivos de registro generados por el sistema durante la ejecución, detectando tanto las fases clasificadas cómo los tiempos de inferencia asociados a cada predicción. En la Figura 33 se presenta la comparativa de cinco pruebas correspondientes a la etapa de ebullición del proceso de fritura, evaluadas mediante el sistema de alerta basado en visión artificial. Esta gráfica muestra la fase detectada a lo largo del tiempo de los primeros 80 segundos, permitiendo observar el comportamiento del sistema bajo condiciones reales de operación en el área de fritura.

Figura 33

Resultado de las pruebas de etapa de ebullición



Fuente. Autoría propia.

De manera general, los resultados muestran que el sistema mantiene una tendencia predominante hacia la detección de la etapa de ebullición, alcanzando una precisión promedio del 79,90 %. Las muestras presentan un comportamiento consistente, con valores de precisión del superior al 80 % en su mayoría, lo que evidencia estabilidad en la clasificación bajo condiciones similares de operación.

Las desviaciones observadas se manifiestan principalmente como transiciones puntuales hacia las etapas de deshidratación y cocción, especialmente en los instantes intermedios y finales del intervalo analizado. Estas transiciones coinciden con momentos de mayor actividad en el aceite, donde el burbujeo y la liberación de vapor generan variaciones visuales que afectan la continuidad de la detección.

En cuanto al rendimiento computacional, el tiempo de inferencia se mantiene constante en 142 ms para todas las pruebas, lo cual indica que el sistema opera de forma estable y predecible, sin verse afectado por la variabilidad visual propia de la etapa de ebullición. Este

comportamiento es adecuado para aplicaciones en tiempo casi real dentro de un entorno industrial.

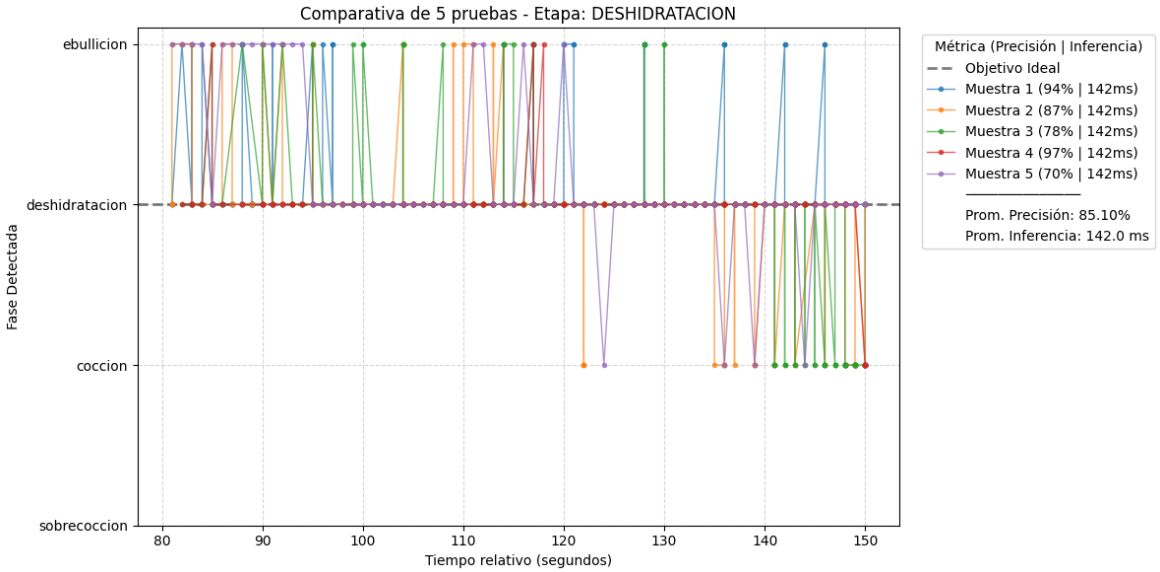
En conjunto, los resultados confirman que la etapa de ebullición representa una fase crítica pero controlable para el sistema de visión artificial. Si bien existen fluctuaciones en la clasificación debido a las condiciones físicas del proceso, el sistema logra identificar correctamente la etapa dominante y mantener un desempeño consistente tanto en precisión como en tiempo de respuesta.

4.2.3. Pruebas de etapa de Deshidratación

La Figura 34 muestra la comparativa de cinco pruebas correspondientes a la etapa de deshidratación, evaluadas en el intervalo aproximado entre 80 y 150 segundos. En términos generales, se observa que esta etapa presenta una mejor estabilidad de detección en comparación con la ebullición, lo cual se refleja en una precisión promedio del 85,10 %.

Figura 34

Resultado de las pruebas de etapa de deshidratación



Fuente. Autoría propia.

De manera general, la etapa de deshidratación muestra una precisión promedio del 85,10 %, lo que indica un desempeño aceptable y relativamente estable para la mayoría de las

pruebas. Las muestras 1,2 y 4 alcanzan valores de precisión elevados, superiores al 80 %, e incluso del 97 % en el caso de la muestra 4, lo cual refleja una adecuada identificación de esta fase cuando las condiciones visuales son favorables.

Las desviaciones detectadas se asocian principalmente a la pérdida progresiva de humedad superficial, que modifica gradualmente la textura y el color de las papas. A medida que la superficie se vuelve más seca, el sistema puede interpretar ciertos fotogramas como pertenecientes a etapas adyacentes, generando solapamientos en la clasificación.

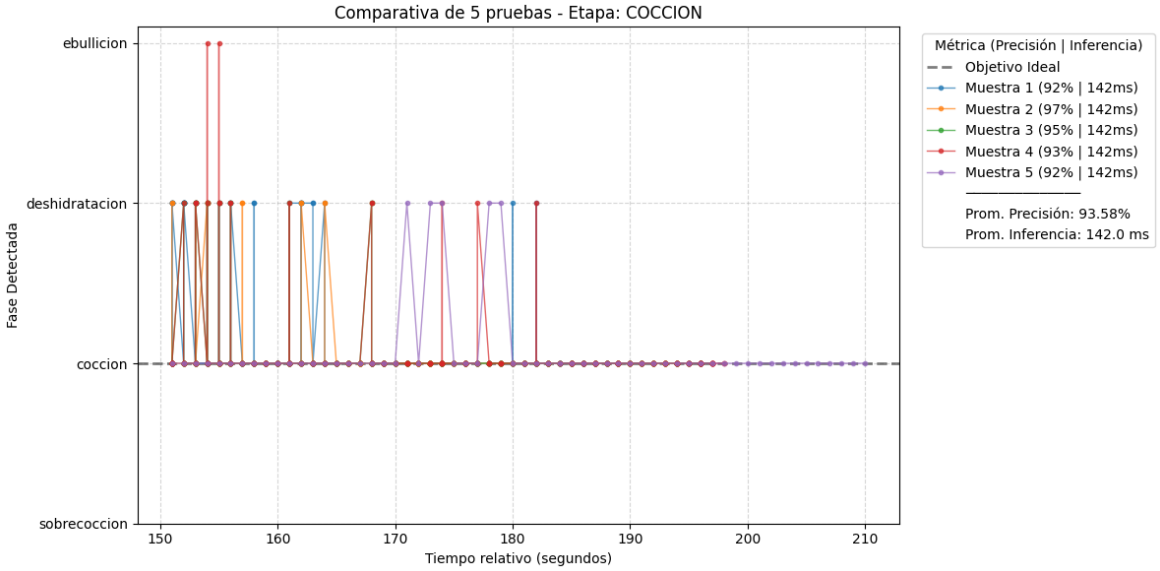
En términos de rendimiento computacional, el tiempo de inferencia se mantiene estable, con un promedio de 142 ms, lo que confirma la estabilidad del sistema en esta etapa de cocción.

4.2.4. Pruebas de etapa de Cocción

La Figura 35 muestra la comparativa de cinco pruebas correspondientes a la etapa de cocción, identificada como el punto óptimo del proceso de fritura, en el intervalo aproximado entre 150 y 210 segundos. En este tramo, el sistema de visión artificial presenta un alto nivel de consistencia en la clasificación.

Figura 35

Resultado de la prueba de etapa de cocción



Fuente. Autoría propia.

Los resultados evidencian una precisión promedio del 93,58 %, con valores individuales que alcanzan hasta el 97 %, lo que confirma una detección estable y confiable de esta etapa. La mayoría de las muestras mantiene la clasificación alineada con el objetivo ideal, observándose únicamente desviaciones puntuales hacia etapas anteriores, principalmente en los instantes cercanos al inicio del intervalo.

Estas transiciones aisladas se asocian a residuos visuales de etapas previas, como cambios graduales en textura y color, los cuales pueden generar confusión temporal en el sistema. No obstante, dichas variaciones no afectan la tendencia general ni la identificación del punto óptimo.

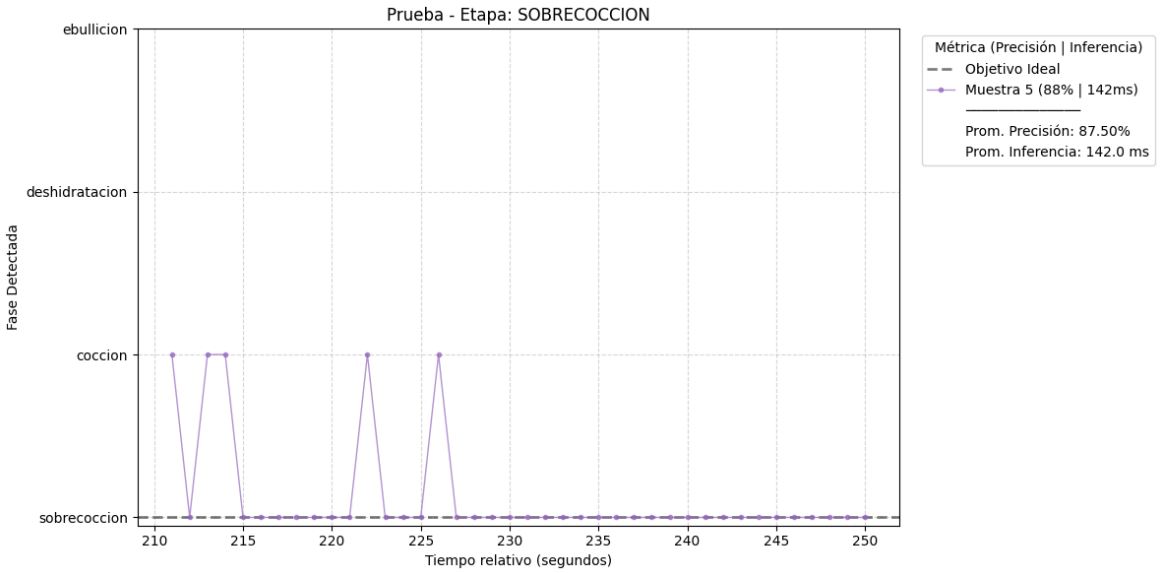
En términos de rendimiento computacional, el tiempo de inferencia se mantiene constante en 142 ms, lo que garantiza una respuesta rápida y adecuada para la generación de alertas en tiempo casi real.

4.2.5. Pruebas de etapa de Sobrecocción

La Figura 36 presenta los resultados correspondientes a la etapa de sobrecocción, evaluada en el intervalo aproximado entre 210 y 250 segundos. A diferencia de las etapas anteriores, para esta fase se realizó una sola muestra experimental, decisión tomada de manera intencional con el fin de evitar el desperdicio innecesario de materia prima, dado que el producto en esta condición ya no es apto para su comercialización.

Figura 36

Resultado de la prueba de etapa de sobrecocción



Fuente. Autoría propia.

Los resultados obtenidos muestran que el sistema de visión artificial logra identificar la etapa de sobrecocción con una precisión del 87,50 %, manteniendo un rendimiento con tiempos de inferencia constante de 142 ms. En la Figura 36 se observa que la detección de sobrecocción es predominante a lo largo del intervalo analizado, con apariciones puntuales de la etapa de cocción, principalmente en los instantes cercanos al inicio del tramo evaluado.

Estas transiciones aisladas hacia cocción se explican por la similitud visual entre el producto en punto óptimo y en estado inicial de sobrecocción, donde los cambios en color y textura aún no son completamente uniformes. No obstante, conforme avanza el tiempo, la clasificación se estabiliza en la etapa de sobrecocción, evidenciando que el sistema reconoce adecuadamente el deterioro progresivo del producto.

Si bien el número de muestras para esta etapa es limitado, los resultados permiten validar el comportamiento general del sistema ante condiciones de sobrecoccion. La decisión de restringir la cantidad de pruebas responde a criterios prácticos y productivos, priorizando la optimización de recursos de la empresa sin comprometer la evaluación funcional del sistema.

4.3. Metodología de evaluación

Para la evaluación del sistema se definió como una muestra válida cada inferencia correspondiente a un frame capturado por la cámara del dispositivo. Dado que el modelo de detección puede generar múltiples predicciones dentro de una misma imagen, se consideró únicamente la detección con mayor nivel de confianza asociada a cada timestamp. Este criterio garantiza una única decisión por frame y evita la sobre-representación de múltiples detecciones en una sola imagen. La etapa real del proceso de fritura fue determinada en función del tiempo transcurrido desde el inicio de la prueba, estableciendo umbrales temporales para cada fase del proceso: ebullición, deshidratación, cocción y sobrecocción. Esta segmentación temporal permitió comparar la etiqueta detectada por el sistema con la etapa real del proceso en cada instante de evaluación.

4.4. Análisis y Resultados de detección por etapa

La evaluación de la precisión del sistema se fundamenta en un criterio de concordancia binaria: una detección se valida como correcta si la etiqueta inferida por el modelo coincide estrictamente con la fase física del proceso, y como errónea en caso de discrepancia. A partir de esta métrica, se cuantificó el desempeño general mediante el tamaño de la muestra (N) la frecuencia de aciertos y errores, y el rendimiento porcentual desglosado por cada etapa de cocción.

4.4.1. Resultados de precisión por etapa a partir del análisis temporal

A partir del análisis de las gráficas temporales realizadas en las pruebas del sistema se obtuvieron los siguientes valores promedio de precisión dentro de cada etapa:

- **Ebullición:** 79,90 %
- **Deshidratación:** 85,10 %
- **Cocción:** 93,58 %
- **Sobrecocción:** 87,50 %

Estos resultados evidencian que el sistema presenta un mejor desempeño una vez que el proceso alcanza etapas visualmente más estables, como la cocción, mientras que las mayores tasas de error se concentran en las etapas iniciales y de transición.

4.4.2. Resultados globales de precisión por etapa

Con el fin de obtener una visión global del desempeño del sistema, se calculó la precisión por etapa considerando todas las muestras cuya etapa real era conocida. La Tabla 13 presenta los resultados obtenidos bajo este enfoque.

Tabla 13

Resultados de evaluación del sistema por etapa de fritura

Etapa real del proceso	N (muestras)	Aciertos	Errores	% de acierto	Observación técnica
Ebullición	342	281	61	82,16 %	Reflejos del aceite pueden generar confusión
Deshidratación	343	289	54	84,26 %	Cambios de burbujeo/vapor pueden saturar imagen
Cocción (óptima)	240	229	11	95,42 %	Color y textura más estables favorecen la inferencia
Sobrecocción	40	35	5	87,50 %	Cambios rápidos y oscurecimiento progresivo

Nota. Para la obtención de los resultados de esta tabla se utilizó un script el cual se describe en el Anexo 8.

Los resultados de la Tabla 13 muestran que la etapa de cocción presenta el mayor porcentaje de acierto, lo cual se asocia a la estabilidad visual del snack en esta fase. En contraste, la etapa de deshidratación evidencia una mayor tasa de error debido a las transiciones graduales entre fases y a la presencia de burbujeo y vapor, factores que introducen variabilidad

en la imagen capturada. La etapa de sobrecocción fue evaluada con un número reducido de muestras, ya que el proceso no se prolongó de forma intencional para evitar el desperdicio de materia prima, por lo que los resultados obtenidos en esta fase se consideran referenciales.

Se observa una diferencia entre los valores de precisión obtenidos a partir de las gráficas obtenidas en el plan de pruebas y los valores globales por etapa. Esta diferencia se debe a que las gráficas evalúan el desempeño del sistema únicamente dentro de la ventana temporal de cada etapa, donde se concentran los instantes más críticos de transición, mientras que la evaluación global considera todas las muestras correspondientes a una etapa, incluyendo intervalos de mayor estabilidad visual.

En consecuencia, el análisis temporal de las gráficas representa una evaluación más estricta del comportamiento del sistema, mientras que la evaluación global refleja su desempeño promedio por etapa.

4.4.3. Resultados del rendimiento del sistema

El rendimiento del sistema fue evaluado mediante el tiempo de inferencia promedio del modelo de visión artificial ejecutado en el dispositivo embebido. Durante las pruebas realizadas, el sistema presentó un tiempo promedio de inferencia de **142 ms**, considerando el tiempo de procesamiento de cada frame.

Este valor permite una operación en tiempo casi real, garantizando que las alertas generadas por el sistema se emitan de manera oportuna sin afectar el flujo normal del proceso de fritura. El resultado obtenido demuestra la viabilidad de implementar técnicas de visión artificial en dispositivos de recursos computacionales limitados.

4.4.4. Evaluación de la eficacia del sistema

La eficacia del sistema fue evaluada a partir de su capacidad para generar alertas visuales y auditivas durante el proceso de fritura. El prototipo cuenta con indicadores LEDs

que informan al operario sobre la etapa detectada como se muestra en la Figura 37, así como una sirena ubicada en la parte lateral del dispositivo para la emisión de alertas auditivas.

Figura 37

Sistema alertando visualmente a operario en área de fritura.

Fuente. Autoría Propia.

Durante las pruebas realizadas, el sistema permitió identificar de manera clara el punto óptimo de cocción y alertar oportunamente ante condiciones de sobrecocción, demostrando su utilidad práctica como herramienta de apoyo en la supervisión del proceso de fritura de la empresa “Delicias de mi Tierra”.

4.5. Video de funcionamiento del sistema

Para evidenciar el funcionamiento del sistema de alerta implementado durante el proceso de fritura de la empresa “Delicias de mi Tierra” se realiza el siguiente video:

- https://youtu.be/cgSOTDeLgEw

4.6. Mejoras futuras

Al evaluar el desempeño del sistema de visión artificial, resulta indispensable proyectar optimizaciones que potencien su eficiencia y adaptabilidad. Con este propósito, se realizó un análisis comparativo basado en múltiples iteraciones de reentrenamiento dentro de la

plataforma Edge Impulse. Este proceso permitió extraer métricas de rendimiento bajo distintas configuraciones de hardware y software, tal como se detalla en la Tabla 14 y Tabla 15. El estudio se centró en evaluar el impacto de las resoluciones de entrada (96x96 px y 160x160 px) sobre variables críticas como la latencia de inferencia, la precisión del modelo y el consumo de recursos (RAM y Flash).

Tabla 14

Tabla Comparativa de Desempeño por Dispositivo usando resolución 96x96px

Dispositivo	Latencia INT8 (ms)	Latencia Float32 (ms)	RAM INT8 (KB)	RAM Float32 (KB)	FLASH INT8 (KB)	FLASH Float32 (KB)	Precisión INT8 (%)	Precisión Float32 (%)
XIAO ESP32S3 Sense	142	1499	239.50	435.20	72	103.40	89.50	92.20
ESP-EYE	1092	2679	137.70	429.00	81.30	112.60	89.30	91.50
Arduino Nicla Vision	64	75	283.10	887.10	110.60	103.40	89.70	92.20
Raspberry Pi 5	4	2	293.40	1000.00	120.80	135.60	92.10	92.90
NVIDIA Jetson Nano	-	1	-	2000.00	-	223.30	-	93.50

Nota. Autoría propia, obtenido de sección *Deployment de Edge Impulse.*

A partir de la comparación de desempeño presentada en la Tabla 14, se identifican oportunidades claras de mejora para el sistema de visión artificial desarrollado, especialmente en lo referente a la plataforma de despliegue y la optimización del modelo. Los resultados evidencian que el dispositivo actualmente utilizado, XIAO ESP32-S3 Sense, ofrece un equilibrio adecuado entre consumo de recursos y precisión; sin embargo, también revela limitaciones en términos de latencia de inferencia cuando se compara con plataformas de mayor capacidad computacional.

Al trabajar con una resolución de 96x96 píxeles, se observa que los dispositivos de mayor capacidad computacional, como Raspberry Pi 5 y NVIDIA Jetson Nano, presentan latencias significativamente reducidas en comparación con los microcontroladores embebidos. No obstante, el objetivo del proyecto no es maximizar velocidad en hardware de alto rendimiento, sino garantizar operación eficiente en plataformas de recursos limitados. En este contexto, el XIAO ESP32S3 Sense demuestra un comportamiento equilibrado al utilizar

cuantización INT8, reduciendo drásticamente el tiempo de inferencia respecto al modo Float32 y manteniendo una precisión cercana al 90%. Esta diferencia confirma que la cuantización representa una estrategia viable para disminuir carga computacional sin comprometer de forma crítica la capacidad de clasificación.

Al incrementar la resolución a 160x160 píxeles como se muestra en la Tabla 15, se evidencia un aumento generalizado en el consumo de memoria y en la latencia de inferencia en todos los dispositivos evaluados. Sin embargo, también se registra una mejora en la precisión del modelo, alcanzando valores superiores al 95% en algunos casos. Este comportamiento refleja una relación directa entre la cantidad de información visual procesada y la capacidad discriminativa del modelo. No obstante, en dispositivos con memoria restringida, el incremento en RAM y FLASH puede representar un factor limitante para su implementación práctica.

Tabla 15

Tabla Comparativa de Desempeño por Dispositivo usando resolución 160x160px

Dispositivo	Latencia INT8 (ms)	Latencia Float32 (ms)	RAM INT8 (KB)	RAM Float32 (KB)	FLASH INT8 (KB)	FLASH Float32 (KB)	Precisión INT8 (%)	Precisión Float32 (%)
XIAO ESP32S3 Sense	1911	3813	250.90	688.00	81.40	112.80	93.00	95.40
ESP-EYE	2110	2679	202.10	642.60	82.36	196.10	91.00	92.30
Arduino Nicla Vision	82	98	283.10	887.10	152.06	172.30	93.20	95.30
Raspberry Pi 5	5	2	456.00	1830.20	120.80	265.51	94.10	95.80
NVIDIA Jetson Nano	-	1	-	4132.36	-	462.41	-	95.90

Nota. Autoría propia, obtenido de sección *Deployment de Edge Impulse.*

Los resultados obtenidos confirman que la cuantización INT8 constituye una estrategia clave para dispositivos de recursos limitados, al ofrecer una reducción considerable en tiempos de procesamiento y consumo de memoria con una pérdida mínima de precisión. Sin embargo, futuras investigaciones podrían analizar técnicas híbridas o cuantización selectiva por capas, con el propósito de mejorar aún más el balance entre eficiencia y exactitud.

.

Conclusiones y recomendaciones

Conclusiones

Se diseñó e implementó exitosamente un sistema de alerta basado en visión artificial capaz de identificar las diferentes etapas del proceso de fritura de papas, demostrando la viabilidad de aplicar técnicas de inteligencia artificial optimizadas en dispositivos embebidos de recursos computacionales limitados dentro de un entorno industrial real.

El modelo entrenado logró distinguir las etapas de ebullición, deshidratación, cocción y sobrecocción con altos niveles de precisión, destacando un mejor desempeño en la etapa de cocción, correspondiente al punto óptimo del proceso. Este resultado valida el cumplimiento del objetivo general planteado y confirma la capacidad del sistema para operar como herramienta de apoyo en la estandarización del proceso productivo.

Las pruebas experimentales realizadas en condiciones reales permitieron evaluar el desempeño integral del sistema, obteniéndose un tiempo promedio por ciclo de procesamiento de 142 ms, incluyendo captura, preprocesamiento e inferencia. El sistema opera bajo un esquema síncrono de captura–procesamiento–decisión, donde cada fotograma es procesado inmediatamente después de su adquisición, sin desfase inter-frame.

Dado que la dinámica del proceso de fritura evoluciona en escalas temporales del orden de segundos, la latencia obtenida es significativamente menor que la constante temporal del proceso, garantizando operación en tiempo real compatible con el entorno industrial sin interferir en el flujo productivo.

El análisis temporal de precisión reveló que los errores se concentran principalmente en las transiciones entre etapas adyacentes, fenómeno asociado a la naturaleza progresiva del proceso térmico y a la variabilidad visual generada por vapor y burbujeo. Esta observación confirma que las limitaciones detectadas no corresponden a fallas estructurales del modelo, sino a características inherentes del entorno físico evaluado.

El análisis comparativo de métricas obtenidas mediante reentrenamiento en Edge Impulse evidenció un compromiso técnico entre resolución de imagen y eficiencia computacional. Se determinó que la configuración de 96x96 px constituye el punto óptimo de operación, al ofrecer un equilibrio adecuado entre precisión y latencia, garantizando estabilidad térmica y determinismo en el microcontrolador seleccionado.

Desde una perspectiva aplicada, la incorporación de alertas visuales y auditivas permitió validar la funcionalidad práctica del sistema, reduciendo la dependencia exclusiva del criterio humano en la determinación del punto óptimo de fritura y contribuyendo potencialmente a la disminución de sobrecocción y desperdicio de materia prima.

Finalmente, aunque el sistema demostró viabilidad técnica y aplicabilidad en entornos de pequeña y mediana escala, se identifican como limitaciones la restricción de memoria del dispositivo embebido y la sensibilidad del modelo ante variaciones extremas de iluminación. No obstante, el presente trabajo sienta bases sólidas para futuras optimizaciones del modelo y su migración a plataformas de mayor capacidad computacional cuando la aplicación lo requiera.

Recomendaciones

Se recomienda ampliar el conjunto de datos utilizado en el entrenamiento del modelo incorporando variaciones controladas en tipo de materia prima, espesor de corte, carga de producto por lote y condiciones de iluminación artificial y natural. Esta ampliación permitiría incrementar la robustez del modelo ante escenarios operativos no controlados y mejorar su capacidad de generalización en diferentes líneas de producción de snacks derivados de papa.

Se sugiere realizar evaluaciones longitudinales del sistema bajo condiciones operativas extendidas, considerando la degradación progresiva del aceite de fritura y los cambios térmicos acumulativos del entorno. Este análisis permitiría cuantificar la estabilidad del modelo frente a

variaciones físico–químicas del proceso y determinar posibles ajustes dinámicos de calibración.

Se recomienda explorar técnicas avanzadas de optimización para modelos TinyML, tales como poda estructural, cuantización selectiva por capas, con el fin de reducir aún más la latencia de inferencia y el consumo de memoria, sin comprometer la precisión alcanzada. Estas estrategias podrían permitir la implementación de modelos más complejos dentro de las restricciones del microcontrolador.

Para aplicaciones que requieran determinismo más estricto o monitoreo simultáneo de múltiples bandejas de fritura, se aconseja evaluar plataformas embebidas con aceleradores dedicados para redes neuronales, seleccionando la arquitectura en función de métricas objetivas como latencia máxima permitida, consumo energético y capacidad de procesamiento paralelo.

Se recomienda incorporar un módulo de registro y almacenamiento de eventos que permita generar trazabilidad histórica del proceso, facilitando análisis estadísticos posteriores, identificación de patrones de error y soporte para decisiones de mejora continua dentro del entorno industrial.

Desde el punto de vista mecánico, se aconseja optimizar el encapsulado del sistema mediante materiales resistentes a altas temperaturas y protección IP adecuada, garantizando aislamiento frente a vapor y partículas de aceite, lo cual incrementaría la confiabilidad operativa y la vida útil del dispositivo en ambientes agresivos.

Finalmente, se recomienda integrar el sistema de manera permanente al flujo operativo de la empresa y realizar estudios comparativos antes y después de su implementación, con el propósito de medir cuantitativamente indicadores como reducción de sobrecocción, disminución de desperdicio de materia prima y mejora en la estandarización del producto final. Esta validación permitiría consolidar el impacto industrial del sistema propuesto.

.

Referencias Bibliográficas

Arciniegas, S., Rivero, D., Piñan, J., Diaz, E., & Rivas, F. (2025). IoT device for detecting abnormal vibrations in motors using TinyML. *Discover Internet of Things 2025 5:1*, *5*(1), 41-. https://doi.org/10.1007/s43926-025-00142-4

Arun, C. A., Karthik, M., Reddy, C. K. M., Sai, M. P., Thenappan, S., & Gadde, M. N. (2023). Efficient Detection of Missing and Misplaced Components in Electronic Boards through Edge Impulse's Automated Inspection. *2023 International Conference on Data Science, Agents and Artificial Intelligence, ICDSAAI 2023*. https://doi.org/10.1109/ICDSAAI59313.2023.10452438

Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., Huang, X., Hurtado, R., Kanter, D., Lokhmotov, A., Patterson, D., Pau, D., Seo, J., Sieracki, J., Thakker, U., Verhelst, M., & Yadav, P. (2020). *Benchmarking TinyML Systems: Challenges and Direction*.

Banzi -Arduino, M. (2021). *"Tiny machine learning with Arduino."*

Barrios-Chinchayhuara, Y. M., Valderrama-Armas, E. H., Cieza-Mostacero, S. E., Pacheco-Torres, J. P., & Alcántara-Moreno, O. R. (2022). Embedded System With Artificial Vision To Improve The Brick Production Process. *Iberian Conference on Information Systems and Technologies, CISTI*, *2022-June*. https://doi.org/10.23919/CISTI54924.2022.9820343

Bayar, D., Aridici, C., Metin, Y. S., Za, O., & Edizkan, R. (2023). Elderly Fall Detection System with ESP32 Module and Edge Impulse Studio. *ISAS 2023 - 7th International Symposium on Innovative Approaches in Smart Technologies, Proceedings*. https://doi.org/10.1109/ISAS60782.2023.10391675

Castillo, J. J., & Colorado, V. M. (2022). *Aplicación de la metodología Lean Manufacturing para mejorar la productividad en la línea de papas fritas, Snacks América Latina SRL Santa Anita 2022*. https://repositorio.ucv.edu.pe/handle/20.500.12692/104696

da Silva, J., Flores, T., Júnior, S., & Silva, I. (2023). TinyML-Based Pothole Detection: A Comparative Analysis of YOLO and FOMO Model Performance. *2023 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2023*. https://doi.org/10.1109/LA-CCI58595.2023.10409357

Deepak, G. D., & Bhat, S. K. (2025). Optimization of deep learning-based faster R-CNN network for vehicle detection. *Scientific Reports 2025 15:1*, *15*(1), 38937-. https://doi.org/10.1038/s41598-025-22828-z

Duque Quevedo, O. R. (2024). *Propuesta de mejoras de alertas de seguridad de dispositivos de IOT mediante inteligencia artificial.* http://repositorio.espe.edu.ec/jspui/handle/21000/37629

Edge Impulse. (2025). *¿Qué es Edge Impulse? | Documentación de Edge Impulse*. https://docs.edgeimpulse.com/docs/concepts/edge-ai-fundamentals/what-is-edge-impulse

Eki, R., Kawasaki, R., & Yanagisawa, E. (2023). Intelligent Vision Sensor and Edge Computing Envisage the Future. *Technical Digest - International Electron Devices Meeting, IEDM*. https://doi.org/10.1109/IEDM45741.2023.10413768

Escobar, A. (2019). *Visión artificial aplicada a la detección e identificación de personas en tiempo real*. http://bibdigital.epn.edu.ec/handle/15000/20098

Fiallos, M. (2023). *Calidad de un bocadito de maíz extruído para la empresa Simaa Cia. Ltda.* http://dspace.espoch.edu.ec/handle/123456789/21160

Gkillas, A., & Lalos, A. (2023). Resource Efficient Federated Learning for Deep Anomaly Detection in Industrial IoT applications. *International Conference on Digital Signal Processing, DSP*, *2023-June*. https://doi.org/10.1109/DSP58604.2023.10167873

Gonzáles, E. (2021). Aplicación de tecnología IoT para la mejora de procesos industriales. *Https://Investigacion.Teinco.Edu.Co/Wp-Content/Uploads/2024/07/Aplicacion-de-Tecnologia-IoT-Para-La-Mejora-de-Procesos-Industriales-.Pdf?Utm_source=chatgpt.Com*.

Greenpeace. (2017). *CLICKING CLEAN 2017 ¿QUIÉN ESTÁ GANANDO LA CARRERA PARA CONSTRUIR UN INTERNET VERDE?*

Guzman, T., Lorena, K., Rea, Y., & Chávez, A. G. (2024). *Obtención de snack alimenticio a partir de cuatro variedades de papa (Solanum tuberosum L), utilizando aceites esenciales de rizomas*. https://dspace.ueb.edu.ec/handle/123456789/6748

Hadida, S. (2020). *UNIVERSIDAD DEL CEMA Buenos Aires Argentina Serie DOCUMENTOS DE TRABAJO*. www.cema.edu.ar/publicaciones/doc_trabajo.html

ISO/IEC/IEEE 29148:2011. (2011). Https://Www.Iso.Org/Standard/45171.Html.

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://arxiv.org/pdf/1412.6980

Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L., Kawsar, F., & Labs, B. (2016). *DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including*

Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2/FIGURES/5

Lovato, C. (2020). *SISTEMA DE DETECCIÓN Y ALERTA DEL ESTADO DE SOMNOLENCIA DE CONDUCTORES MEDIANTE VISIÓN ARTIFICIAL*. http://repositorio.uisrael.edu.ec/handle/47000/2441

Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet V2: Practical guidelines for efficient cnn architecture design. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11218 LNCS*, 122–138. https://doi.org/10.1007/978-3-030-01264-9_8/TABLES/8

Marwedel, P. (2021). *Embedded System Design*. *Embedded Systems*. https://doi.org/10.1007/978-3-030-60910-8

Mathworks. (2025). *Descripción del producto MATLAB - MATLAB & Simulink*. https://la.mathworks.com/help/matlab/learn_matlab/product-description.html

Mellit, A., Blasuttigh, N., & Pavan, A. M. (2023). TinyML for fault diagnosis of Photovoltaic Modules using Edge Impulse Platform. *11th International Conference on Smart Grid, IcSmartGrid 2023*. https://doi.org/10.1109/ICSMARTGRID58556.2023.10171088

Montes O., N., Millar M., I., Provoste L., R., Martínez M., N., Fernández Z., D., Morales I., G., & Valenzuela B., R. (2016). Absorción de aceite en alimentos fritos. *Revista Chilena de Nutrición*, *43*(1), 87–91. https://doi.org/10.4067/S0717-75182016000100013

Morales Tamayo, Y. (2025). Aplicaciones del Internet de las cosas (IoT) en la Industria Electromecánica para la recolección, análisis y monitoreo en tiempo real. *Revista Ingenio Global*, *4*. https://doi.org/10.62943/rig.v4n1.2025.163

Pedreschi, F., Moyano, P., Kaack, K., & Granby, K. (2005). Color changes and acrylamide formation in fried potato slices. *Food Research International*, *38*(1), 1–9. https://doi.org/10.1016/j.foodres.2004.07.002

Posligua Bran, R., Ramos Yépez, L., Suárez Reyes, H., & Mendoza, O. D. (2009). *Proyecto de inversión para la elaboración y comercialización de un snack artesanal a base de papa china organica para el consumo en el mercado Guayaquileño.* http://www.dspace.espol.edu.ec/handle/123456789/5701

Prieto, M., Mouwen, J. M., López Puente, S., & Cerdeño Sánchez, A. (2008). Concepto de calidad en la industria Agroalimentaria. *Interciencia*, *33*(4), 258–264. http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0378-18442008000400006&lng=es&nrm=iso&tlng=es

Pumisacho, & Velasquez. (2009). *CHOLA – Inventario de Tecnologías e Información para el Cultivo de Papa en Ecuador*. https://cipotato.org/papaenecuador/2017/10/12/35-chola/

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352–2449. https://doi.org/10.1162/NECO_A_00990,

Redmon, J. S. D. R. G. A. F. (2016). (YOLO) You Only Look Once. *Cvpr*, *2016-December*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ruales, S. (2024). *Problema en área de fritura de la empresa Delicias de mi Tierra / Entrevistado por Pablo Muñoz*.

Seeed Studio. (2025). *Getting Started with Seeed Studio XIAO ESP32S3 Series | Seeed Studio Wiki*. Https://Wiki.Seeedstudio.Com/Xiao_esp32s3_getting_started/.

Sinergy Research Group. (2019). *SaaS Spending Hits $100 billion Annual Run Rate; Microsoft Extends its Leadership | Synergy Research Group*. https://www.srgresearch.com/articles/saas-spending-hits-100-billion-annual-run-rate-microsoft-extends-its-leadership

Sosa-Savedra, J. C., Trejo-Estrella, M. A., García-García, A. L., Barceina-Sánchez, J. D. O., Velázquez-González, R. S., & Hernández-Tovar, R. (2023). Implementación de la metodología V como eje de desarrollo de un tribómetro de perno en disco. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías Del ICBI*, *11*, 21–29. https://doi.org/10.29057/icbi.v11iespecial4.11358

Sougey Medialdea, M. (2023). *Impacto del IoT en el desempeño económico de las empresas.* https://repositorio.comillas.edu/xmlui/handle/11531/69177

Szeliski, R. (2022). *Computer Vision*. Springer International Publishing. https://doi.org/10.1007/978-3-030-34372-9

TensorFlow. (2025). *Faster Dynamically Quantized Inference with XNNPack — The TensorFlow Blog*. https://blog.tensorflow.org/2024/04/faster-dynamically-quantized-inference-with-xnnpack.html

UNDP. (2024). *Objetivo 9: Industria, innovación e infraestructura | Programa De Las Naciones Unidas Para El Desarrollo*. https://www.undp.org/es/sustainable-development-goals/industria-innovacion-infraestructura

Vera, S., Chuquimarca, L., Galdea, W., Véliz, B., & Saldaña, C. (2024). *Automated Quality Control System for Canned Tuna Production using Artificial Vision*. https://doi.org/10.1109/AIIoT58432.2024.10574669

Wu, J. J. (2023). *Application of Systems Engineering Process in Building ML-Enabled Systems*.

Yalli, J. S., Hasan, M. H., & Badawi, A. (2024). Internet Of Things (IOT): Origin, Embedded Technologies, Smart Applications and its Growth in the Last Decade. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3418995

Yandrapati, A. B., Singam, V. K. P. R., Namburu, B., Paladugu, V., & Karakula, T. K. (2023). Design of a Smart Embedded Vision System Based on FPGA for Medical Applications. *ICSPC 2023 - 4th International Conference on Signal Processing and Communication*, 1–5. https://doi.org/10.1109/ICSPC57692.2023.10125679

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 2921–2929. https://doi.org/10.1109/CVPR.2016.319

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, *107*(8), 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951

Anexos

Anexo 1: Entrevista

Ing. Santiago Ruales

Jefe de producción de la Procesadora de Alimentos “Delicias de mi Tierra”

"En el área de fritura hemos tenido un problema constante: los tiempos de cocción de las papas no siempre son consistentes. A veces se nos pasan de cocción y otras veces quedan semi crudas, lo que afecta mucho la calidad del producto y genera desperdicio de materia prima. Hemos capacitado al personal varias veces, pero al final seguimos dependiendo de la vista y la experiencia de cada operario, y eso da lugar a errores. La verdad, necesitamos un sistema que nos avise cuándo las papas están en su punto exacto, que nos ayude a mantener una cocción uniforme y reducir esas pérdidas. Eso nos facilitaría mucho el trabajo y garantizaría un mejor producto al cliente."

Anexo 2: Encuesta

Preguntas de la Encuesta

Pregunta 1

¿Qué criterio utiliza principalmente para identificar que una papa está bien cocida?

1. El color dorado de la superficie
2. El tiempo de cocción estimado
3. La textura crujiente al tacto
4. Combinación de color, textura y tiempo

Pregunta 2

¿Qué errores se presentan con mayor frecuencia durante el proceso de fritura? (puede marcar más de una opción)

1. Papas quemadas o sobrecocidas
2. Papas poco cocidas
3. Tamaño desigual de los trozos
4. Falta de atención del operario

Pregunta 3

¿Qué señales visuales le indican que una papa está sobrecocida?

1. Color marrón oscuro o negro
2. Textura excesivamente dura o seca
3. Olor a quemado

Pregunta 4

¿Qué factores toma en cuenta al evaluar el punto de cocción?

1. Solo el color
2. Solo la textura
3. Solo el tiempo

4. Todas las anteriores

Pregunta 5

¿Qué tan importante considera la experiencia del operario en la calidad del producto final?

1. Muy importante
2. Moderadamente importante
3. Poco importante
4. No influye

Pregunta 6

¿Con qué frecuencia se cometen errores en el punto de cocción durante un turno de trabajo?

1. Nunca
2. 1 a 2 veces
3. 3 a 5 veces
4. Más de 5 veces

Pregunta 7

¿Cuáles considera que son las consecuencias de una sobrecocción en el producto final?

(puede marcar más de una)

1. Pérdida de sabor
2. Cambios negativos en la textura
3. Apariencia poco atractiva
4. Pérdida de materia prima

Pregunta 8

¿El proceso de fritura actual es vulnerable a errores humanos?

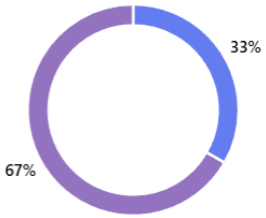
1. Sí, constantemente

2. Algunas veces
3. Rara vez
4. No
5. No estoy seguro/a

Anexo 3: Respuestas de Encuesta

1. **¿Qué criterio utiliza principalmente para identificar que una papa está bien cocida?**

El color dorado de la superficie	1
El tiempo de cocción estimado	0
La textura crujiente al tacto	0
Combinación de color, textura y tiempo	2

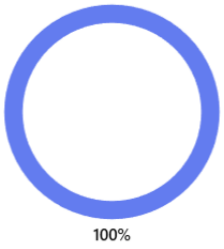


2. **¿Qué errores se presentan con mayor frecuencia durante el proceso de fritura?** (puede marcar más de una opción)

Papas quemadas o sobrecocidas	3
Papas poco cocidas	0
Tamaño desigual de los trozos	0
Falta de atención del operario	3

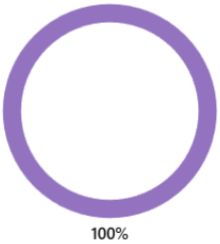
3. **¿Qué señales visuales le indican que una papa está sobrecocida?**

Color marrón oscuro o negro	3
Textura excesivamente dura o seca	0
Olor a quemado	0



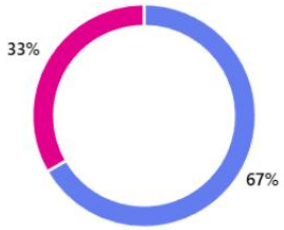
4. **¿Qué factores toma en cuenta al evaluar el punto de cocción?**

Solo el color	0
Solo la textura	0
Solo el tiempo	0
Todas las anteriores	3



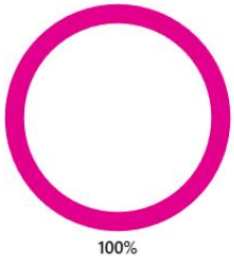
5. **¿Qué tan importante considera la experiencia del operario en la calidad del producto final?**

Muy importante	2
Moderadamente importante	1
Poco importante	0
No influye	0



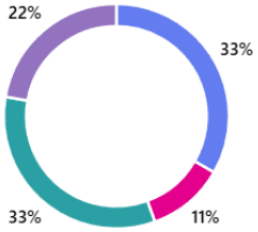
6. **¿Con qué frecuencia se cometen errores en el punto de cocción durante un turno de trabajo?**

Nunca	0
1 a 2 veces	3
3 a 5 veces	0
Más de 5 veces	0



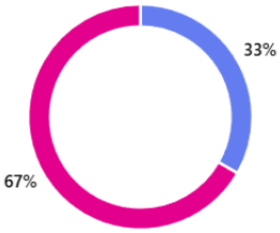
7. **¿Cuáles considera que son las consecuencias de una sobrecocción en el producto final?** *(puede marcar más de una)*

Pérdida de sabor	3
Cambios negativos en la textura	1
Apariencia poco atractiva	3
Pérdida de materia prima	2



8. **¿El proceso de fritura actual es vulnerable a errores humanos?**

Sí, constantemente	1
Algunas veces	2
Rara vez	0
No	0
No estoy seguro/a	0



Anexo 4: Plataformas de software

Para el desarrollo e implementación del sistema de visión artificial embebida propuesto en este proyecto, se emplearon dos plataformas complementarias: Edge Impulse y MATLAB. Cada una de estas herramientas ofrece funcionalidades específicas que han facilitado distintas fases del desarrollo, desde el entrenamiento del modelo hasta la visualización y validación de resultados y comparados en la Tabla 16.

Edge Impulse

Según (Mellit et al., 2023) es una plataforma de desarrollo de aprendizaje automático orientada específicamente a sistemas embebidos y dispositivos de bajo consumo. Fue diseñada con el propósito de facilitar la creación de modelos de inteligencia artificial que puedan ejecutarse directamente en microcontroladores, sistemas en chip (SoC), y dispositivos de borde (edge devices). Además (Edge Impulse, 2025)recalca que una de sus principales fortalezas radica en su interfaz gráfica basada en la web, que permite a los usuarios crear flujos completos de adquisición, entrenamiento, optimización y despliegue de modelos sin necesidad de programar desde cero.

Características:

- **Orientado a dispositivos embebidos**: Diseñado específicamente para el desarrollo de modelos de inteligencia artificial que se ejecuten en microcontroladores o dispositivos de borde con recursos limitados.
- **Interfaz web intuitiva**: Permite trabajar desde un navegador sin necesidad de instalaciones complejas, lo que facilita el flujo de trabajo incluso para usuarios sin experiencia avanzada en programación.
- **Adquisición y etiquetado de datos**: Permite conectar sensores o cámaras para capturar datos directamente desde el dispositivo. También ofrece herramientas para etiquetar y clasificar los datos dentro del mismo entorno.

- **Automatización del entrenamiento**: Ofrece modelos base y permite configurar procesos de entrenamiento con parámetros ajustables y validación automática.
- **Compatibilidad con modelos optimizados**: Genera modelos listos para desplegarse en plataformas como Arduino, ESP32, STM32 y otros microcontroladores, incluyendo formatos como TensorFlow Lite y C++.
- **Compilación eficiente con EON Compiler**: Reduce el tamaño del modelo y optimiza el uso de memoria RAM y almacenamiento Flash, lo que mejora el rendimiento en hardware embebido.
- **Visualización de métricas**: Muestra precisión, F1 score, matriz de confusión y tiempo de inferencia, permitiendo evaluar fácilmente el rendimiento del modelo.
- **Despliegue directo en dispositivos**: Ofrece flujos de trabajo automatizados para cargar el modelo entrenado directamente en el microcontrolador sin pasos intermedios complejos.

MATLAB

Según (Mathworks, 2025) es un entorno de programación de alto nivel y una plataforma numérica ampliamente utilizada en ingeniería, ciencias aplicadas y análisis de datos. A diferencia de Edge Impulse, que se enfoca en la implementación de modelos ligeros, MATLAB se distingue por su capacidad para realizar simulaciones complejas, análisis numéricos avanzados y procesamiento profundo de datos, lo cual lo convierte en una herramienta poderosa para la validación técnica de sistemas de visión artificial.

Características:

- **Plataforma de análisis técnico**: Diseñada para realizar cálculos matemáticos avanzados, procesamiento de señales, análisis estadístico y visualización de datos a nivel profesional.
- **Procesamiento de imágenes**: Ofrece un toolbox especializado para trabajar con imágenes, lo cual es ideal para analizar visualmente patrones.
- **Visualización personalizada de datos**: Permite representar gráficamente resultados, comparar clases visuales, analizar histogramas, texturas, intensidad y bordes de las imágenes procesadas.
- **Simulación y prototipado**: Permite simular el comportamiento del sistema antes de desplegarlo, útil para validar la lógica de funcionamiento del modelo IA fuera del entorno embebido.
- **Alta compatibilidad y extensibilidad**: Se integra con otros lenguajes como Python, C++, Java y plataformas como Simulink, lo cual permite ampliar su funcionalidad en proyectos complejos.
- **Manejo eficiente de datos**: Ideal para organizar, manipular y procesar grandes cantidades de información en forma de matrices, estructuras o archivos masivos.
- **Programación personalizada**: Aunque ofrece herramientas visuales, su verdadero potencial se alcanza mediante el uso de scripts y funciones programadas a medida.
- **Enfoque académico y científico**: Ampliamente usado en investigación por su precisión en cálculos, trazabilidad de resultados y herramientas para replicar experimentos.

Tabla 16

Tabla de comparación entre Edge Impulse y Matlab

Criterio de comparación	Edge Impulse	MATLAB
Enfoque principal	Modelado y despliegue de IA en sistemas embebidos	Análisis numérico, simulación y procesamiento de datos e imágenes
Tipo de interfaz	Plataforma web gráfica con flujos automáticos	Entorno de programación y scripts, con GUI para tareas específicas
Curva de aprendizaje	Baja (interfaz intuitiva para usuarios sin experiencia en código)	Media a alta (requiere conocimientos en programación y matemáticas)
Soporte para dispositivos embebidos	Compatible con ESP32, Arduino, STM32, etc.	No se ejecuta directamente en microcontroladores
Optimización de recursos	Modelos cuantizados (int8), bajo uso de RAM y Flash	Alta demanda de recursos, orientado a PC
Procesamiento de imágenes	Limitado pero funcional para clasificación de imágenes en el borde	Avanzado, permite segmentación, filtrado, análisis morfológico, etc.
Entrenamiento de modelos IA	Integrado en la nube con arquitecturas optimizadas	Se puede hacer, pero requiere librerías externas o toolboxes
Visualización de datos y resultados	Intermedia (matriz de confusión, precisión, etc.)	Avanzada (gráficos personalizados, simulación, análisis dinámico)
Costo de uso	Gratuito para usuarios individuales	Licencia comercial

Despliegue de modelos IA	Directo a dispositivos con botón "Deploy to device"	No orientado a despliegue embebido
Compatibilidad multiplataforma	Web (funciona en Windows, Mac, Linux, incluso desde móviles)	Principalmente para escritorio (Windows, Mac, Linux)

Fuente. Autoría Propia, basada en (Edge Impulse, 2025) y (Mathworks, 2025)

Anexo 5: Algoritmos de detección de objetos

En el desarrollo de sistemas de visión artificial enfocados en detección de objetos o clasificación visual, la elección del algoritmo adecuado es fundamental. Dependiendo del contexto, como la disponibilidad de recursos de hardware, la precisión esperada y el entorno de despliegue, se opta por modelos más ligeros o robustos. A continuación, se describen tres de los algoritmos más utilizados, los cuales fueron considerados en el marco de este proyecto por su aplicabilidad en entornos embebidos y su soporte dentro de plataformas como Edge Impulse y comparados en la Tabla 17.

MobileNet SSD FPN-Lite

MobileNet SSD FPN-Lite es un modelo de detección de objetos que combina la arquitectura MobileNetV2 como backbone, con la estructura Single Shot Multibox Detector (SSD) y una capa adicional llamada Feature Pyramid Network (FPN-Lite). Según el autor (Liu et al., 2016) esta combinación permite realizar detección de objetos de múltiples tamaños, manteniendo una estructura ligera y eficiente para dispositivos con limitaciones computacionales. SSD realiza predicciones de bounding boxes en una sola pasada de red, lo cual lo hace rápido, mientras que FPN mejora la detección de objetos pequeños mediante la reutilización de características extraídas a diferentes escalas. Este modelo es ideal para tareas en tiempo real, como seguimiento de objetos o detección de elementos en cámaras móviles.

FOMO (Fast Object Detection for Mobile Devices)

FOMO es una arquitectura de detección de objetos optimizada específicamente para dispositivos de muy bajo consumo, como microcontroladores. Utiliza como base una versión compacta de MobileNetV2, el autor (da Silva et al., 2023) dice que en lugar de generar bounding boxes tradicionales, divide la imagen en una malla de celdas y predice la presencia de clases directamente en cada celda. Este enfoque reduce drásticamente el

uso de memoria y procesamiento, permitiendo que la detección se ejecute en dispositivos con menos de 1 MB de RAM, como el ESP32-S3. Es altamente eficiente para contar objetos, detectar su presencia o estimar ubicación relativa. Es ideal para tareas donde la precisión de la caja delimitadora no es crítica, como detección de defectos, clasificación rápida o monitoreo de estados visuales.

YOLOv5

YOLOv5 es una de las versiones más populares y eficientes de la familia YOLO de detección de objetos. Está desarrollada en PyTorch y ofrece una excelente relación entre velocidad y precisión. En la comparativa que realiza (da Silva et al., 2023) a diferencia de FOMO, YOLOv5 sí genera cajas delimitadoras precisas y predice tanto la clase como la ubicación del objeto en una sola etapa, lo que lo hace adecuado para tareas donde se requiere alta precisión espacial. Aunque YOLOv5 es más pesado en comparación con los modelos TinyML como FOMO. Debido a su diseño modular, YOLOv5 es ampliamente usado en tareas industriales, vigilancia, vehículos autónomos y otras aplicaciones avanzadas de visión por computadora.

Tabla 17

Tabla comparativa de Algoritmos de Detección de Objetos

Criterio de comparación	MobileNet SSD FPN-Lite	FOMO (MobileNetV2)	YOLOv5
Tipo de arquitectura	Backbone MobileNetV2 + SSD + FPN-Lite	Grid-based detection con MobileNetV2	Convolucional en una sola etapa (end-to-end)
Tamaño del modelo (aprox.)	< 5 MB	< 1 MB	Entre 7 y 40 MB
Salida de detección	Bounding boxes por objeto	Detección por celdas (sin cajas)	Bounding boxes con clase y coordenadas

Precisión general	Moderada-alta	Moderada	Alta (según versión y dataset)
Velocidad de inferencia	Alta (tiempo real)	Muy alta (óptimo para microcontroladores)	Alta (requiere GPU para tiempo real)
Requisitos de hardware	Procesador intermedio o SoC optimizado	Microcontroladores (ESP32, STM32)	GPU o CPU potente (Jetson, Raspberry Pi)
Aplicaciones típicas	Seguimiento de objetos, detección en cámaras móviles	Detección ligera en tiempo real, conteo, estados simples	Detección precisa, vigilancia, vehículos autónomos
Compatibilidad con Edge Impulse	Sí (parcial en modelos personalizados)	Sí (nativamente soportado)	No directamente, requiere integración externa

Fuente. Autoría propia.

Anexo 6: Dispositivos de recursos limitados

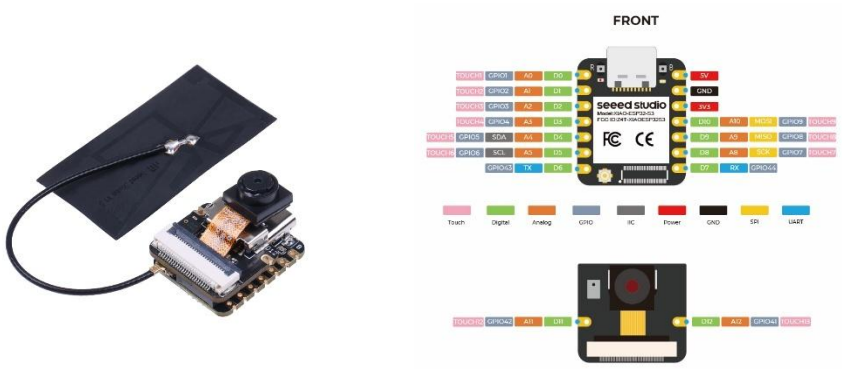
Con el objetivo de seleccionar el hardware más adecuado para la implementación del sistema de visión artificial embebido, se han analizado las características técnicas de tres plataformas ampliamente utilizadas en aplicaciones de visión artificial en el borde: XIAO ESP32S3 Sense, Raspberry Pi 4 Model B y Arduino Nicla Vision. A continuación, se describen sus principales especificaciones y ventajas comparativas.

XIAO ESP32S3 Sense

El XIAO ESP32S3 Sense, desarrollado por Seeed Studio, es un microcontrolador compacto y eficiente diseñado para proyectos de inteligencia artificial y visión computacional en tiempo real. Está basado en el chip ESP32-S3, que incluye un procesador Xtensa® dual-core LX7 a 240 MHz, con soporte nativo para operaciones vectoriales y aceleradores para tareas de inferencia como muestra la Figura 38 y sus características en la Tabla 18.

Figura 38

Microcontrolador XIAO ESP32S3 Sense



Nota. Recuperado de https://wiki.seeedstudio.com/es/xiao_esp32s3_getting_started/

Tabla 18

Características técnicas del XIAO ESP32S3 Sense

Especificación	Detalle

Procesador	Xtensa® dual-core LX7 a 240 MHz
Memoria RAM	512 KB interna
Memoria Flash	Hasta 8 MB externa
Cámara integrada	OV2640 (2MP)
Conectividad	Wi-Fi 802.11n, Bluetooth 5.0 LE
Dimensiones	21 mm x 17.5 mm
Compatibilidad IA	Modelos int8 / TensorFlow Lite / Edge Impulse
Consumo energético	Muy bajo

Fuente. Autoría Propia.

Raspberry Pi 4 Model B

En la Figura 39 se observa la Raspberry Pi 4 Model B es una computadora de placa única (SBC) de alto rendimiento que incorpora un procesador Broadcom BCM2711 con arquitectura quad-core Cortex-A72 a 1.5 GHz como se observa en la Tabla 19. Se encuentra disponible en variantes con 2 GB, 4 GB y 8 GB de RAM LPDDR4, lo que le permite ejecutar sistemas operativos completos como Raspberry Pi OS o distribuciones Linux para tareas de procesamiento intensivo.

Figura 39

Raspberry Pi 4 Model B

Nota. Recuperado de https://www.mouser.ec/ProductDetail/Raspberry-Pi/RPI4-MODBP-8GB-BULK?qs=sPbYRqrBIVlrjhMid19vUA%3D%3D

Aunque su consumo energético es mayor que el de un microcontrolador, su potencia de procesamiento lo convierte en una excelente opción para prototipos de visión artificial que requieran alta precisión y procesamiento local intensivo.

Tabla 19

Características del Raspberry Pi 4 Model B

Especificación	Detalle
Procesador	Broadcom BCM2711, quad-core Cortex-A72 a 1.5 GHz
Memoria RAM	2 GB, 4 GB o 8 GB LPDDR4
Almacenamiento	MicroSD / USB 3.0
Conectividad	Wi-Fi, Bluetooth 5.0, Ethernet, HDMI dual, USB 3.0
Dimensiones	85.6 mm × 56.5 mm
Compatibilidad IA	TensorFlow, OpenCV, PyTorch, etc.
Sistema Operativo	Linux, Raspberry Pi OS
Consumo energético	Moderado a alto

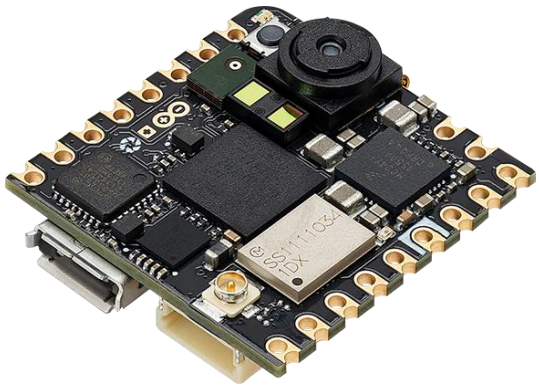
Fuente. Autoría propia.

Arduino Nicla Vision

En la Figura 40 se muestra el Arduino Nicla Vision es un módulo embebido altamente integrado, diseñado específicamente para proyectos de visión artificial en el borde. Cuenta con un microcontrolador STM32H747AII6, que incluye un núcleo Cortex-M7 a 480 MHz y un núcleo Cortex-M4 a 240 MHz y demás características especificadas en la Tabla 20, permitiendo ejecutar tareas paralelas y modelos de inferencia.

Figura 40

Arduino Nicla Vision



Nota. Recuperado de https://store.arduino.cc/products/nicla-vision

Tabla 20

Características técnicas del Arduino Nicla Vision

Especificación	Detalle
Procesador	STM32H747AII6 (Cortex-M7 480 MHz + Cortex-M4 240 MHz)
Memoria RAM	1.5 MB
Memoria Flash	2 MB
Cámara integrada	2MP
Sensores adicionales	IMU, micrófono, barómetro
Conectividad	Bluetooth Low Energy (BLE)
Dimensiones	22.86 mm x 22.86 mm
Compatibilidad IA	Edge Impulse, Arduino IDE

Fuente. Autoría propia.

Anexo 7: Script para despliegue de sistema

El presente anexo describe el funcionamiento del código desarrollado en el entorno Arduino IDE, el cual implementa el sistema de alerta para la detección del punto óptimo de cocción mediante visión artificial. El código integra la captura de imágenes, la inferencia del modelo entrenado en Edge Impulse y la generación de alertas visuales y auditivas en función del estado del proceso de fritura.

Para mitigar la variabilidad estocástica inherente a las predicciones de la red neuronal (fluctuaciones entre cuadros) y eliminar la dependencia rígida de temporizadores fijos, se diseñó e implementó un algoritmo de control basado en Acumulación de Evidencia (Evidence Accumulation). Este algoritmo actúa como un filtro temporal que pondera la consistencia de las detecciones antes de activar los cambios de estado en el sistema.

```
#include <SystemAlert_v3_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"
#include "esp_camera.h"

// ------------------------------------------------------------------
// DEFINICIÓN DE PINES
// ------------------------------------------------------------------
#define LED_PIN_ROJO      7
#define LED_PIN_VERDE     8
#define LED_PIN_AMARILLO  9
#define PIN_BUZZER        4

// ------------------------------------------------------------------
// CALIBRACIÓN
// ------------------------------------------------------------------
const int UMBRAL_ALERTA_VERDE = 100;
const int UMBRAL_ALERTA_ROJA = 30;

const unsigned long TIEMPO_PARA_RESET_MS = 20000;
const unsigned long ZONA_MUERTA_INICIAL_MS = 150000;

// ------------------------------------------------------------------
// VARIABLES GLOBALES
// ------------------------------------------------------------------
#define CAMERA_MODEL_XIAO_ESP32S3
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS           320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS           240
#define EI_CAMERA_FRAME_BYTE_SIZE                 3

static bool debug_nn = false;
static bool is_initialised = false;
uint8_t *snapshot_buf;

static camera_config_t camera_config = {
```

```
            .pin_pwdn = -1, .pin_reset = -1, .pin_xclk = 10, .pin_sscb_sda = 40,
.pin_sscb_scl = 39,
           .pin_d7 = 48, .pin_d6 = 11, .pin_d5 = 12, .pin_d4 = 14, .pin_d3 = 16, .pin_d2
= 18, .pin_d1 = 17, .pin_d0 = 15,
           .pin_vsync = 38, .pin_href = 47, .pin_pclk = 13,
            .xclk_freq_hz = 20000000, .ledc_timer = LEDC_TIMER_0, .ledc_channel =
LEDC_CHANNEL_0,
           .pixel_format = PIXFORMAT_JPEG, .frame_size = FRAMESIZE_QVGA, .jpeg_quality
= 12,
             .fb_count = 1, .fb_location = CAMERA_FB_IN_PSRAM, .grab_mode =
CAMERA_GRAB_WHEN_EMPTY,
       };
```

Debido a la naturaleza ruidosa del entorno industrial (cambios de luz, reflejos en el aceite), el sistema debe diferenciar entre un "falso positivo" momentáneo y el inicio real del proceso de fritura. Para ello, se implementó un filtro de consistencia temporal estricto. El sistema permanece en un estado de ESPERA (consumo mínimo, actuadores apagados) hasta cumplir la siguiente condición lógica

```
       // --- VARIABLES DE CONTROL ---
       unsigned long inicioFritura = 0;
       unsigned long ultimoTiempoVistoAlgo = 0;
       bool procesoActivo = false;

       int puntosCoccion = 0;
       int puntosSobrecoccion = 0;
       int contadorArranque = 0;

       enum EstadoFritura { ESPERA, EN_PROCESO, OPTIMO, SOBRECOCCION };
       EstadoFritura estadoActual = ESPERA;

       enum EstadoAlarma { ALARMA_OFF, ALARMA_VERDE, ALARMA_ROJA };
       EstadoAlarma tipoAlarma = ALARMA_OFF;
       unsigned long buzzerLastToggle = 0;
       int pulsosRealizados = 0;
       bool buzzerEncendido = false;
       bool secuenciaTerminada = false;

       // Prototipos
       bool ei_camera_init(void);
       void ei_camera_deinit(void);
       bool  ei_camera_capture(uint32_t  img_width,  uint32_t  img_height,  uint8_t
*out_buf);
       static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr);
       void setLeds(int rojo, int verde, int amarillo);
       void gestionarSirena(unsigned long tiempoActual);

       void setup() {
           Serial.begin(115200);
           // Configurar pines
                pinMode(LED_PIN_ROJO,   OUTPUT);   pinMode(LED_PIN_VERDE,   OUTPUT);
pinMode(LED_PIN_AMARILLO, OUTPUT); pinMode(PIN_BUZZER, OUTPUT);

           // Test inicial
            setLeds(HIGH, HIGH, HIGH); digitalWrite(PIN_BUZZER, HIGH); delay(200);
digitalWrite(PIN_BUZZER, LOW); setLeds(LOW, LOW, LOW);

           if (ei_camera_init() == false) {
```

```
            ei_printf("Error camara\r\n");
        }
        ei_sleep(2000);
    }

    void loop() {
        if (ei_sleep(5) != EI_IMPULSE_OK) return;

            snapshot_buf   =   (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS   *
EI_CAMERA_RAW_FRAME_BUFFER_ROWS * EI_CAMERA_FRAME_BYTE_SIZE);
        if(snapshot_buf == nullptr) return;

        ei::signal_t signal;
                  signal.total_length      =      EI_CLASSIFIER_INPUT_WIDTH      *
EI_CLASSIFIER_INPUT_HEIGHT;
        signal.get_data = &ei_camera_get_data;

                     if       (ei_camera_capture((size_t)EI_CLASSIFIER_INPUT_WIDTH,
(size_t)EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {
            free(snapshot_buf); return;
        }

        ei_impulse_result_t result = { 0 };
        EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);
        if (err != EI_IMPULSE_OK) {
            free(snapshot_buf); return;
        }

        // --- 1. CONTEO Y VISUALIZACIÓN DE DETECCIONES (SIN TIEMPO) ---
        int boxesEbullicion = 0;
        int boxesDeshidratacion = 0;
        int boxesCoccion = 0;
        int boxesSobrecoccion = 0;

        Serial.println("-------------------------------------------------");

    #if EI_CLASSIFIER_OBJECT_DETECTION == 1
        for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {
            ei_impulse_result_bounding_box_t bb = result.bounding_boxes[i];
            if (bb.value < 0.50) continue;

            // IMPRIMIR DETALLE DE CADA CAJA (QUÉ VE)
             Serial.printf("Deteccion: %s (Conf: %.2f) [x:%u, y:%u]\n", bb.label,
bb.value, bb.x, bb.y);

            if (strcmp(bb.label, "ebullicion") == 0) boxesEbullicion++;
           else if (strcmp(bb.label, "deshidratacion") == 0) boxesDeshidratacion++;
            else if (strcmp(bb.label, "coccion") == 0) boxesCoccion++;
            else if (strcmp(bb.label, "sobrecoccion") == 0) boxesSobrecoccion++;
        }
    #endif

         int totalObjetos = boxesEbullicion + boxesDeshidratacion + boxesCoccion +
boxesSobrecoccion;
        if (totalObjetos > 0) ultimoTiempoVistoAlgo = millis();

        free(snapshot_buf);

        // --- 2. LÓGICA DE CONTROL ---
        unsigned long ahora = millis();

        // A. ARRANQUE
        if (!procesoActivo) {
            // MODIFICACIÓN: Solo cuenta si detecta ebullición
            if (boxesEbullicion > 0) {
                contadorArranque++;
```

```
                        Serial.printf(">>  Estado:  Detectando  Arranque  (%d/5)\n",
contadorArranque);
                    // El LED amarillo aún NO se enciende aquí, estamos en ESPERA
                    if (contadorArranque >= 5) {
                        procesoActivo = true;
                        inicioFritura = ahora;
                         estadoActual = EN_PROCESO; // Aquí pasamos a EN_PROCESO, que
encenderá el amarillo
                           puntosCoccion = 0; puntosSobrecoccion = 0; tipoAlarma =
ALARMA_OFF; digitalWrite(PIN_BUZZER, LOW);
                        Serial.println(">>> ARRANQUE CONFIRMADO <<<");
                    }
                } else {
                    // Si ve otra cosa o nada, reinicia el contador.
                    // Necesita 5 SEGUIDAS de ebullición.
                    if (contadorArranque > 0) {
                           Serial.println(">> Arranque interrumpido (No es ebullición
seguida)");
                         contadorArranque = 0;
                    }
                }
            }

            // B. RESET
            if (procesoActivo && (ahora - ultimoTiempoVistoAlgo > TIEMPO_PARA_RESET_MS))
{
                procesoActivo = false; estadoActual = ESPERA; tipoAlarma = ALARMA_OFF;
digitalWrite(PIN_BUZZER, LOW);
                contadorArranque = 0; // Reiniciar contador de arranque
                Serial.println(">>> RESET (FREIDORA VACIA) <<<");
            }
```

Una vez iniciado el proceso, el sistema abandona el control lineal por tiempo y adopta un enfoque de densidad de detección. Se define una variable de estado acumulativa $P_{cocción}$ (Puntos de Cocción) que funciona bajo el principio de *Leaky Bucket* (cubeta con fugas):

La actualización del puntaje en cada ciclo de inferencia (t) se define por:

Ecuación (5)

$$P_{coccion}(t) = P_{coccion}(t-1) + \Delta_{ganancia} - \Delta_{perdida}$$

- Ganancia ($\Delta_{ganancia}$): Es proporcional al número de objetos validados como "Cocción" en el cuadro actual. Si la red detecta 3 papas listas, se suman 3 puntos. Esto permite que el sistema acelere la decisión si la cantidad de producto listo es masiva.

- Pérdida ($\Delta_{perdida}$): Es un factor de enfriamiento. Si el sistema deja de detectar "Cocción" (por ejemplo, debido a una oclusión por vapor), el puntaje disminuye gradualmente, evitando que un falso positivo aislado dispare la alarma.

  El cambio de estado a ÓPTIMO (Alerta Verde) se dispara cuando:

Ecuación (6)

$$P_{coccion} \geq Umbral_{verde}$$

Este método permite que el sistema se adapte dinámicamente: si la temperatura del aceite es más alta y el producto se cocina más rápido, la densidad de detecciones aumentará velozmente, activando la alarma antes de lo previsto por un temporizador tradicional.

```
        // C. ACUMULADORES
            unsigned long tiempoTranscurrido = 0;

            if (procesoActivo) {
                tiempoTranscurrido = ahora - inicioFritura;

                if (tiempoTranscurrido < ZONA_MUERTA_INICIAL_MS) {
                   puntosCoccion = 0; puntosSobrecoccion = 0; estadoActual = EN_PROCESO;
                }
                else {
                    // Acumulador Coccion
                    if (boxesCoccion > 0) puntosCoccion += boxesCoccion;
                    else {
                       if (boxesEbullicion > 0 || boxesDeshidratacion > 0) puntosCoccion
-= 2;
                        else puntosCoccion -= 1;
                    }
                    if (puntosCoccion < 0) puntosCoccion = 0;
                      if (puntosCoccion > UMBRAL_ALERTA_VERDE + 20) puntosCoccion =
UMBRAL_ALERTA_VERDE + 20;

                    // Acumulador Sobrecoccion
                    if (boxesSobrecoccion > 0) puntosSobrecoccion += (boxesSobrecoccion
* 2);
                    else puntosSobrecoccion -= 1;
                    if (puntosSobrecoccion < 0) puntosSobrecoccion = 0;
                   if (puntosSobrecoccion > UMBRAL_ALERTA_ROJA + 20) puntosSobrecoccion
= UMBRAL_ALERTA_ROJA + 20;

                    // Decision Estado
                        if (puntosSobrecoccion >= UMBRAL_ALERTA_ROJA) estadoActual =
SOBRECOCCION;
                    else if (puntosCoccion >= UMBRAL_ALERTA_VERDE) {
                        if (estadoActual != SOBRECOCCION) estadoActual = OPTIMO;
                    }
                    else {
                         if (estadoActual != OPTIMO && estadoActual != SOBRECOCCION)
estadoActual = EN_PROCESO;
                    }
                }
```

```
            // Alarmas Auditivas
            if (estadoActual == SOBRECOCCION) {
                if (tipoAlarma != ALARMA_ROJA) {
                  tipoAlarma = ALARMA_ROJA; pulsosRealizados = 0; secuenciaTerminada
= false; buzzerEncendido = true; buzzerLastToggle = ahora; digitalWrite(PIN_BUZZER,
HIGH);
                }
            }
            else if (estadoActual == OPTIMO) {
                 if (tipoAlarma != ALARMA_VERDE && tipoAlarma != ALARMA_ROJA) {
                            tipoAlarma = ALARMA_VERDE; pulsosRealizados = 0;
secuenciaTerminada = false; buzzerEncendido = true; buzzerLastToggle = ahora;
digitalWrite(PIN_BUZZER, HIGH);
                 }
            }

            gestionarSirena(ahora);
        } else {
            digitalWrite(PIN_BUZZER, LOW);
        }

        // Salidas LED
        switch (estadoActual) {
            case ESPERA:       setLeds(LOW, LOW, LOW); break; // Todo apagado hasta
confirmar arranque
            case EN_PROCESO:   setLeds(LOW, LOW, HIGH); break; // Amarillo
            case OPTIMO:       setLeds(LOW, HIGH, LOW); break; // Verde
            case SOBRECOCCION: setLeds(HIGH, LOW, LOW); break; // Rojo
        }

        // IMPRESION DE RESUMEN
        Serial.printf("PtsVerde: %d/%d | PtsRojo: %d/%d | Estado: %d | T: %lu s\n",
                  puntosCoccion, UMBRAL_ALERTA_VERDE,
                  puntosSobrecoccion, UMBRAL_ALERTA_ROJA,
                  estadoActual, tiempoTranscurrido/1000);
    }

    // ------------------------------------------------------------------
    // GESTIÓN SIRENA Y AUXILIARES
    // ------------------------------------------------------------------
    void gestionarSirena(unsigned long tiempoActual) {
             if    (secuenciaTerminada    ||    tipoAlarma    ==    ALARMA_OFF)    {
digitalWrite(PIN_BUZZER, LOW); return; }
        unsigned long delta = tiempoActual - buzzerLastToggle;

        if (tipoAlarma == ALARMA_VERDE) {
                 if  (pulsosRealizados  >=  3)  {  secuenciaTerminada  =  true;
digitalWrite(PIN_BUZZER, LOW); return; }
              if (buzzerEncendido) { if (delta >= 500) { digitalWrite(PIN_BUZZER,
LOW); buzzerEncendido = false; buzzerLastToggle = tiempoActual; pulsosRealizados++; }
}
                  else  {  if  (delta  >=  3000)  {  if  (pulsosRealizados  <  3)  {
digitalWrite(PIN_BUZZER,   HIGH);   buzzerEncendido   =   true;   buzzerLastToggle   =
tiempoActual; } } }
        }
        else if (tipoAlarma == ALARMA_ROJA) {
                 if  (pulsosRealizados  >=  5)  {  secuenciaTerminada  =  true;
digitalWrite(PIN_BUZZER, LOW); return; }
              if (buzzerEncendido) { if (delta >= 1000) { digitalWrite(PIN_BUZZER,
LOW); buzzerEncendido = false; buzzerLastToggle = tiempoActual; pulsosRealizados++; }
}
                  else  {  if  (delta  >=  1000)  {  if  (pulsosRealizados  <  5)  {
digitalWrite(PIN_BUZZER,   HIGH);   buzzerEncendido   =   true;   buzzerLastToggle   =
tiempoActual; } } }
        }
```

```
       }

       void setLeds(int rojo, int verde, int amarillo) { digitalWrite(LED_PIN_ROJO,
rojo); digitalWrite(LED_PIN_VERDE, verde); digitalWrite(LED_PIN_AMARILLO, amarillo); }

       bool ei_camera_init(void) {
           if (is_initialised) return true;
           #if defined(CAMERA_MODEL_ESP_EYE)
           pinMode(13, INPUT_PULLUP); pinMode(14, INPUT_PULLUP);
           #endif
           esp_err_t err = esp_camera_init(&camera_config);
           if (err != ESP_OK) return false;
           sensor_t * s = esp_camera_sensor_get();
           if (s->id.PID == OV3660_PID) { s->set_vflip(s, 1); s->set_brightness(s, 1);
s->set_saturation(s, 0); }
           #if defined(CAMERA_MODEL_M5STACK_WIDE)
           s->set_vflip(s, 1); s->set_hmirror(s, 1);
           #elif defined(CAMERA_MODEL_ESP_EYE)
           s->set_vflip(s, 1); s->set_hmirror(s, 1); s->set_awb_gain(s, 1);
           #endif
           is_initialised = true; return true;
       }
       void ei_camera_deinit(void) { esp_camera_deinit(); is_initialised = false; }
       bool   ei_camera_capture(uint32_t   img_width,   uint32_t   img_height,   uint8_t
*out_buf) {
           bool do_resize = false; if (!is_initialised) return false;
           camera_fb_t *fb = esp_camera_fb_get(); if (!fb) return false;
           bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);
           esp_camera_fb_return(fb); if(!converted) return false;
             if  ((img_width  !=  EI_CAMERA_RAW_FRAME_BUFFER_COLS)  ||  (img_height  !=
EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) do_resize = true;
                                                  if                     (do_resize)
ei::image::processing::crop_and_interpolate_rgb888(snapshot_buf,
EI_CAMERA_RAW_FRAME_BUFFER_COLS, EI_CAMERA_RAW_FRAME_BUFFER_ROWS, out_buf, img_width,
img_height);
           return true;
       }
       static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr) {
           size_t pixel_ix = offset * 3; size_t pixels_left = length; size_t out_ptr_ix
= 0;
           while (pixels_left != 0) {
                   out_ptr[out_ptr_ix]  =  (snapshot_buf[pixel_ix  +  2]  <<  16)  +
(snapshot_buf[pixel_ix + 1] << 8) + snapshot_buf[pixel_ix];
               out_ptr_ix++; pixel_ix+=3; pixels_left--;
           } return 0;
       }
```

Anexo 8: Scripts

Script para leer por serial el sistema embebido

El siguiente script de pyhton establece una comunicación serial con el dispositivo embebido a través del puerto configurado, recibe en tiempo real los mensajes generados por el sistema durante el proceso de fritura y los almacena automáticamente en un archivo de texto sin sobrescribir registros anteriores. Cada línea recibida es registrada junto con una marca temporal de fecha y hora, lo que permite generar un historial cronológico de eventos e inferencias del sistema, facilitando posteriormente el análisis, la validación de resultados y la trazabilidad de las pruebas experimentales realizadas.

```
import serial
import time
import os
from datetime import datetime

# CONFIGURAR PUERTO Y BAUD RATE
puerto = 'COM3'  # ← Asegúrate de colocar aquí el puerto correcto
baud_rate = 115200

# FUNCION PARA EVITAR SOBRESCRITURA
def generar_nombre_archivo(base):
    contador = 0
    nombre = base
    while os.path.exists(nombre):
        contador += 1
        nombre = f"{base.rsplit('.', 1)[0]}_{contador}.txt"
    return nombre

# GENERA NOMBRE DE ARCHIVO ÚNICO
nombre_base = 'Test_Fritura_PapaClasica.txt'
archivo_salida = generar_nombre_archivo(nombre_base)

# INICIA CONEXIÓN SERIAL
ser = serial.Serial(puerto, baud_rate)
print(f"Conectado a {puerto} a {baud_rate} bps")
time.sleep(2)  # Espera a que el Arduino reinicie

# ABRE ARCHIVO Y GUARDA DATOS CON TIMESTAMP
with open(archivo_salida, 'w', encoding='utf-8') as file:
     print(f"Guardando datos en '{archivo_salida}' con fecha y hora.
Presiona Ctrl+C para detener.")
    try:
        while True:
            if ser.in_waiting:
                               linea   =   ser.readline().decode('utf-8',
errors='replace').strip()
```

```
                            timestamp  =  datetime.now().strftime("[%Y-%m-%d
%H:%M:%S]")
                    linea_con_tiempo = f"{timestamp} {linea}"
                    print(linea_con_tiempo)
                    file.write(linea_con_tiempo + '\n')
            except KeyboardInterrupt:
                print("\nLectura finalizada por el usuario.")

        # CIERRA PUERTO SERIAL
        ser.close()
        print("Puerto serie cerrado.")
```

Script para generar resultados en gráficas temporales por etapa

El siguiente script procesa cinco archivos de registro de pruebas de fritura de manera aleatoria, extrae por cada línea la clase detectada y el tiempo de inferencia, calcula el tiempo relativo desde el inicio de cada prueba y asigna una fase teórica según umbrales temporales; luego filtra únicamente las muestras cuya fase real corresponde a la etapa objetivo, calcula la precisión de detección de esa etapa por archivo (coincidencias/total) y el promedio de latencia de inferencia (ms), grafica la evolución temporal de las detecciones (mapeando las clases a valores numéricos) junto a una línea de referencia del "objetivo ideal", y finalmente reporta en consola y en la leyenda del gráfico las métricas por prueba y los promedios globales de precisión e inferencia.

```
        import re
        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import glob
        import random
        from datetime import datetime

        # --- CONFIGURACIÓN DE SELECCIÓN ALEATORIA ---
        # Busca todos los archivos .txt en la carpeta /content/ que empiecen con
"Test_Fritura"
        # Puedes ajustar el patrón si tus archivos tienen otro nombre
        patron = '/content/Test_Fritura_PapaClasica_*.txt'
        todos_los_disponibles = glob.glob(patron)

        # Seleccionamos 5 al azar
        cantidad_muestras = min(5, len(todos_los_disponibles))

        if cantidad_muestras == 0:
            print("Error: No se encontraron archivos en /content/ que coincidan con el
patrón.")
            archivos = []
        else:
            archivos = random.sample(todos_los_disponibles, cantidad_muestras)
```

```
            print(f"Se han seleccionado {cantidad_muestras} archivos aleatorios de un
total de {len(todos_los_disponibles)}:")
            for a in archivos: print(f" - {a}")

        ETAPA_OBJETIVO = "sobrecoccion"
        fase_valor   =   {"ebullicion":   3,   "deshidratacion":   2,   "coccion":   1,
"sobrecoccion": 0}

        # --- FUNCIÓN DE PROCESAMIENTO ---
        def procesar_archivo(ruta):
            # Regex para fase y para tiempo (ms)
                                                    pattern_fase                        =
re.compile(r"\[(.*?)\]\s+(ebullicion|deshidratacion|coccion|sobrecoccion)\s+\(([\d.]+
)\)")
            pattern_tiempo = re.compile(r"Classification:\s+(\d+)\s+ms") # <--- NUEVO:
Regex para inferencia

            data = []
            tiempos = [] # Lista para guardar los ms de este archivo

            try:
                with open(ruta, 'r', encoding='utf-8') as file:
                    for line in file:
                        # Buscar Fase
                        match = pattern_fase.search(line)
                        if match:
                            timestamp_str, clase, confianza = match.groups()
                                timestamp = datetime.strptime(timestamp_str, "%Y-%m-%d
%H:%M:%S")
                            data.append((timestamp, clase))

                        # Buscar Tiempo Inferencia
                        match_t = pattern_tiempo.search(line)
                        if match_t:
                            tiempos.append(int(match_t.group(1)))

            except FileNotFoundError:
                print(f"Advertencia: No se encontró el archivo {ruta}")
                return pd.DataFrame(), 0

            if not data:
                return pd.DataFrame(), 0

            df = pd.DataFrame(data, columns=["timestamp", "clase_detectada"])
                        df["tiempo_relativo"]       =       (df["timestamp"]       -
df["timestamp"].min()).dt.total_seconds()

            # Calcular promedio de inferencia de este archivo
            inf_promedio = np.mean(tiempos) if tiempos else 0

            # Calcular fase teórica
            def fase_teorica(t):
                if t <= 80: return "ebullicion"
                elif t <= 150: return "deshidratacion"
                elif t <= 210: return "coccion"
                else: return "sobrecoccion"

            df["fase_teorica"] = df["tiempo_relativo"].apply(fase_teorica)

            # FILTRAR: Nos quedamos solo con la etapa objetivo
            df_filtrado = df[df["fase_teorica"] == ETAPA_OBJETIVO].copy()

            if not df_filtrado.empty:
                                                df_filtrado["valor_detectado"]        =
df_filtrado["clase_detectada"].map(fase_valor)
```

```
            return df_filtrado, inf_promedio # <--- Devolvemos también la inferencia

        # --- GRAFICAR ---
        plt.figure(figsize=(12, 6))
        colores = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']

        # Línea de referencia (Objetivo ideal)
        plt.axhline(y=fase_valor[ETAPA_OBJETIVO],    color='black',    linestyle='--',
linewidth=2, label='Objetivo Ideal', alpha=0.5)

        metricas_globales = []
        valores_precision = []
        valores_inferencia = [] #

        for i, archivo in enumerate(archivos):
            # Desempaquetamos los dos valores que devuelve la función
            df_run, inferencia = procesar_archivo(archivo)

            if not df_run.empty:
                # Calcular precisión de este archivo
                coincidencias = (df_run["clase_detectada"] == ETAPA_OBJETIVO).sum()
                total = len(df_run)
                precision = (coincidencias / total) * 100

                valores_precision.append(precision)
                valores_inferencia.append(inferencia) # Guardamos para promedio global

                # Texto para consola
                       metricas_globales.append(f"Run   {i+1}:   P:{precision:.1f}%   |
I:{inferencia:.1f}ms")

                # Jitter para visualización
                jitter = 0

                # Graficamos agregando la inferencia al label
                plt.plot(df_run["tiempo_relativo"],
                         df_run["valor_detectado"] + jitter,
                         marker='.',
                         linestyle='-',
                         linewidth=1,
                         color=colores[i % len(colores)],
                        label=f"Muestra {i+1} ({precision:.0f}% | {inferencia:.0f}ms)",
# <--- AQUI SE MUESTRA EN LA GRAFICA
                         alpha=0.7)

        # --- CÁLCULO Y VISUALIZACIÓN DE PROMEDIOS ---
        if valores_precision:
            promedio_prec = sum(valores_precision) / len(valores_precision)
             promedio_inf  =  sum(valores_inferencia)  /  len(valores_inferencia)  # <---
Promedio global de inferencia

            # Truco: Agregamos líneas "invisibles" para la leyenda
            plt.plot([], [], ' ', label="————————————")
            plt.plot([], [], ' ', label=f"Prom. Precisión: {promedio_prec:.2f}%")
             plt.plot([], [], ' ', label=f"Prom. Inferencia: {promedio_inf:.1f} ms") #
<--- Dato nuevo en leyenda

        # Detalles del gráfico
        plt.yticks(list(fase_valor.values()), list(fase_valor.keys()))
        plt.xlabel("Tiempo relativo (segundos)")
        plt.ylabel("Fase Detectada")
        plt.title(f"Comparativa     de     {len(archivos)}     pruebas     -     Etapa:
{ETAPA_OBJETIVO.upper()}")

        # La leyenda ahora incluirá los promedios al final
```

```
        plt.legend(bbox_to_anchor=(1.02,    1),    loc='upper    left',    title="Métrica
(Precisión | Inferencia)")

        plt.grid(True, which='both', linestyle='--', alpha=0.5)
        # Mostrar resumen en consola
        print("Resumen por Archivo:")
        for m in metricas_globales:
            print(m)
        if valores_precision:
            print("-" * 30)
               print(f"GLOBAL   ->   Precisión:   {promedio_prec:.2f}%   |   Inferencia:
{promedio_inf:.2f} ms")

        plt.tight_layout()
        plt.show()
```

Script para generar metricas globales

Este código procesa múltiples archivos de registro generados durante las pruebas del sistema de visión artificial, extrae las clases detectadas junto con su nivel de confianza y las asocia a una fase teórica del proceso de fritura según el tiempo transcurrido desde el inicio de cada prueba; posteriormente consolida todas las detecciones, selecciona una única predicción por instante (la de mayor confianza) y calcula de forma global, para cada etapa real del proceso, el número total de muestras, aciertos, errores y el porcentaje de acierto, generando así métricas cuantitativas que permiten evaluar el desempeño general del sistema, las cuales son finalmente presentadas en consola y almacenadas en un archivo CSV para su análisis y documentación.

```
        import re
        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import glob
        import random
        from datetime import datetime

        # --- CONFIGURACIÓN DE SELECCIÓN ALEATORIA ---
        # Busca todos los archivos .txt en la carpeta /content/ que empiecen con
"Test_Fritura"
        # Puedes ajustar el patrón si tus archivos tienen otro nombre
        patron = '/content/Test_Fritura_PapaClasica_*.txt'
        todos_los_disponibles = glob.glob(patron)

        # Seleccionamos 5 al azar
        cantidad_muestras = min(5, len(todos_los_disponibles))

        # --- FUNCIÓN: fase teórica ---
        def fase_teorica(t):
            if t <= 80: return "ebullicion"
            elif t <= 150: return "deshidratacion"
            elif t <= 210: return "coccion"
```

```
            else: return "sobrecoccion"

        # --- FUNCIÓN DE PROCESAMIENTO ---
        def procesar_archivo(ruta, modo_por_timestamp=True):
            pattern_fase = re.compile(
                 r"\[(.*?)\]\s+(ebullicion|deshidratacion|coccion|sobrecoccion)\s+\(([\
d.]+)\)"
            )

            detecciones = []

            try:
                with open(ruta, 'r', encoding='utf-8') as file:
                    for line in file:
                        m = pattern_fase.search(line)
                        if m:
                            timestamp_str, clase, conf = m.groups()
                                timestamp = datetime.strptime(timestamp_str, "%Y-%m-%d
%H:%M:%S")
                            detecciones.append((timestamp, clase, float(conf)))

            except FileNotFoundError:
                print(f"Advertencia: No se encontró el archivo {ruta}")
                return pd.DataFrame()

            if not detecciones:
                return pd.DataFrame()

             df = pd.DataFrame(detecciones, columns=["timestamp", "clase_detectada",
"confianza"])
                         df["tiempo_relativo"]       =       (df["timestamp"]       -
df["timestamp"].min()).dt.total_seconds()
            df["fase_teorica"] = df["tiempo_relativo"].apply(fase_teorica)

            # UNA MUESTRA = UN TIMESTAMP (se queda con la mayor confianza)
            if modo_por_timestamp:
                df = (df.sort_values("confianza", ascending=False)
                        .drop_duplicates(subset=["timestamp"], keep="first")
                        .copy())

            return df

        # --- PROCESAR TODOS LOS ARCHIVOS ---
        todos = []
        for archivo in archivos:
            df_run = procesar_archivo(archivo, modo_por_timestamp=True)
            if not df_run.empty:
                todos.append(df_run)

        if not todos:
            print("No se encontraron detecciones en los archivos.")
        else:
            df_total = pd.concat(todos, ignore_index=True)

            # --- MÉTRICAS GLOBALES POR ETAPA REAL ---
            resumen = (
                          df_total.assign(acierto=(df_total["clase_detectada"]    ==
df_total["fase_teorica"]))
                .groupby("fase_teorica")
                .agg(
                    N=("acierto", "size"),
                    Aciertos=("acierto", "sum")
                )
                .reset_index()
            )
```

```
            resumen["Errores"] = resumen["N"] - resumen["Aciertos"]
            resumen["%Acierto"] = (resumen["Aciertos"] / resumen["N"] * 100).round(2)

            orden = ["ebullicion", "deshidratacion", "coccion", "sobrecoccion"]
                 resumen["fase_teorica"]    =    pd.Categorical(resumen["fase_teorica"],
categories=orden, ordered=True)
            resumen = resumen.sort_values("fase_teorica").reset_index(drop=True)

            print("\n=== MÉTRICAS GLOBALES POR ETAPA (fase real) ===")
            print(resumen.to_string(index=False))

            resumen.to_csv("/content/resumen_metricas_por_etapa.csv", index=False)
            print("\nArchivo guardado:")
            print("- /content/resumen_metricas_por_etapa.csv")
```