TomatAIT: Prototipo de Robot Estacionario con Cinta Transportadora para Detección de Madurez en Tomates

1st Alessandro Mateo Andrade Ortiz
Ingeniería de Tecnologías de Información y Sistemas
Universidad ESAN
Lima, Perú
23100124@esan.edu.pe

2nd Kheyla Sanheli Condor Garcia
Ingeniería de Tecnologías de Información y Sistemas
Universidad ESAN
Lima, Perú
23100185@esan.edu.pe

3rd Joe Jose Mantilla Huaman
Ingeniería de Tecnologías de Información y Sistemas
Universidad ESAN
Lima, Perú
23100113@esan.edu.pe

***Resumen*—Este documento detalla el diseño y construcción de TomatAIT, un sistema robótico automatizado y estacionario para la clasificación de tomates. El prototipo integra un chasis y una cinta transportadora diseñados en software CAD (Fusion 360 y CorelDRAW) y fabricados mediante corte láser de MDF. La arquitectura de procesamiento es híbrida, empleando una Raspberry Pi 5 para la inteligencia artificial (IA) y un microcontrolador Arduino para el control de la electrónica de bajo nivel (motores de la cinta y actuadores). Se implementó el modelo de visión por computador YOLOv8n, optimizado para Edge Computing, para la detección en tiempo real de cinco clases: *unripe* (verde), *half_ripe* (pintón), *ripe* (maduro), *mold* (moho) y *rotten* (podrido). Los resultados experimentales muestran un mAP@0.5 global de 0.901, destacando una precisión del 93.2 % en la detección de moho a pesar del desbalance del dataset. La gestión del despliegue del software se realiza mediante contenedores Podman, y la comunicación de los resultados de clasificación se gestiona a través del protocolo MQTT, validando la eficiencia del sistema para aplicaciones de Agricultura de Precisión en el borde.**

***Index Terms*—IoT, YOLOv8, Raspberry Pi 5, MQTT, Podman, Cinta Transportadora, Corte Láser, Edge Computing, Clasificación de Tomates, Agricultura de Precisión.**

I. Introducción

El sector agrícola mundial enfrenta desafíos crecientes relacionados con la eficiencia y la calidad en la post-cosecha [1]. La clasificación de frutos, en particular del tomate (*Solanum lycopersicum*), es una tarea crítica que impacta directamente en la cadena de suministro y en el precio final. Tradicionalmente manual, este proceso es lento, propenso a errores humanos y subjetivo, llevando a inconsistencias en la calidad del producto final [5].

El proyecto TomatAIT aborda esta problemática mediante la creación de un prototipo robótico que automatiza el proceso de transporte y clasificación visual. Este sistema integra tres pilares tecnológicos fundamentales: Fabricación Digital, Inteligencia Artificial Embarcada (Edge AI), e Internet de las Cosas (IoT). El prototipo sirve como una prueba de concepto de un clasificador inteligente de bajo costo y alta velocidad, utilizando la Raspberry Pi 5 como plataforma de cómputo principal [2].

Este trabajo fue realizado durante el curso de Robótica de la Universidad ESAN y el FABLAB.

II. Trabajos Relacionados

II-1. Algoritmos de Detección de Madurez en Tomates : El uso de redes neuronales convolucionales (CNN) ha superado a los métodos tradicionales de procesamiento de imágenes por color.

- **Yang y Ju (2024)** realizaron un estudio comparativo entre **YOLOv5 y YOLOv8** para la detección de madurez en tomates cherry. Sus resultados demostraron que YOLOv8 superó a su predecesor con una precisión media (mAP) de 0.757, mejorando la detección en condiciones de oclusión parcial [14].
- Por otro lado, **Zheng et al. (2025)** propusieron una mejora sobre la arquitectura YOLOv8 introduciendo mecanismos de atención para lidiar con la iluminación variable en invernaderos, logrando distinguir entre estados de madurez

complejos con una precisión superior al 0.95 [15]. Aunque estos modelos son potentes, suelen requerir hardware costoso (GPU dedicadas), lo que abre la oportunidad para soluciones optimizadas para bordes (Edge).

II-2. Implementaciones en Sistemas Embebidos (Edge Computing): La transición de servidores potentes a dispositivos de borde (como Raspberry Pi) es una tendencia clave.

- **Lopes et al. (2025)** desarrollaron un sistema de clasificación de frutas en tiempo real utilizando una **Raspberry Pi 4**. Aunque lograron una tasa de procesamiento aceptable (0.12 segundos por frame), notaron limitaciones térmicas y de FPS al usar modelos pesados [15].
- Un enfoque similar fue presentado en el sistema de clasificación de **Hassan et al.**, donde utilizaron una arquitectura híbrida enviando datos desde una Raspberry Pi a un **Arduino** para el control de actuadores de clasificación, logrando una eficiencia de 1350 tomates por hora [16].

II-3. Brecha Identificada y Diferenciación: A pesar de los avances mencionados, la mayoría de implementaciones actuales en la literatura se centran en la ejecución directa de scripts en el sistema operativo ("bare-metal"), lo que dificulta la replicabilidad y escalabilidad del software.

- **Diferenciador de TomatAIT:** A diferencia de los trabajos citados, este proyecto introduce el uso de **contenedores (Podman/Docker)** sobre una **Raspberry Pi 5**. Esto no solo estandariza el entorno de ejecución de la IA, sino que facilita las actualizaciones OTA (Over-the-Air) y el aislamiento de procesos, una práctica común en DevOps pero poco explorada en la robótica agrícola educativa de bajo costo. Además, se aborda específicamente la detección de la clase "Pintón", que suele ser la más propensa a errores en los estudios revisados.

III. Marco Teórico y Fundamentos

Esta sección establece el contexto general y los principios fundamentales que sustentan el desarrollo de TomatAIT, siguiendo la estructura amplia a específica del método embudo.

III-A. Agricultura de Precisión y la Cuarta Revolución Industrial

La Agricultura de Precisión (AP) busca optimizar el uso de recursos y aumentar la eficiencia de la producción mediante la adopción de tecnologías avanzadas, enmarcadas dentro de la Cuarta Revolución Industrial (Industria 4.0) [9]. La AP requiere sistemas automatizados y capaces de tomar decisiones autónomas, siendo la clasificación post-cosecha un candidato ideal para esta automatización.

III-B. Visión por Computador para la Clasificación Agrícola

La Visión por Computador (VC) ha demostrado ser la herramienta más efectiva y no destructiva para la inspección y clasificación de productos agrícolas [10].

III-B1. Modelos de Detección en Tiempo Real: Los modelos de detección de objetos en una sola etapa, como la familia YOLO (You Only Look Once), son preferidos en aplicaciones industriales por su balance óptimo entre velocidad y precisión. Se seleccionó la arquitectura YOLOv8n (nano) por ser la variante más ligera y adecuada para el despliegue en sistemas embebidos (Edge AI) [6]. YOLOv8 utiliza anclas (*anchor-free*) y un enfoque de predicción desacoplada, lo que lo hace más rápido que versiones anteriores sin sacrificar significativamente la precisión (mAP) [1].

III-C. Arquitecturas de Cómputo Distribuido

III-C1. Edge Computing: El Edge Computing consiste en procesar los datos cerca de la fuente de generación (el "borde"de la red), en este caso, en la Raspberry Pi 5 [7]. Esto es fundamental para TomatAIT, ya que reduce la latencia asociada al envío de imágenes a la nube, permitiendo decisiones de clasificación en tiempo real ($< 100\,\mathrm{ms}$) cruciales para una cinta transportadora rápida.

III-C2. IoT y MQTT: El protocolo MQTT (Message Queuing Telemetry Transport) es el estándar de mensajería ligero en el IoT [8]. Su arquitectura de publicación/suscripción es ideal para desacoplar el módulo de IA (el Publicador) del módulo de control y monitoreo (los Suscriptores), mejorando la robustez del sistema.

III-C3. Contenedorización con Podman: Para el despliegue de los servicios (especialmente el broker Mosquitto), se utilizó Podman, una herramienta de gestión de contenedores *daemonless* (sin demonio de fondo). Podman mejora la seguridad y simplifica la gestión de recursos en sistemas Linux embebidos como el Raspberry Pi OS, facilitando la portabilidad y la reproducibilidad del entorno de software [3].

IV. Objetivos

IV-A. Objetivo General

Diseñar y construir un prototipo robótico, denominado TomatAIT, con cinta transportadora capaz de detectar la madurez de un tomate frente a su cámara, utilizando IA embarcada (YOLOv8) e intercambio de información mediante IoT (MQTT), con un enfoque en la viabilidad de la fabricación digital de bajo costo.

IV-B. *Objetivos Específicos*

1. Diseñar el chasis modular y la cinta transportadora en software CAD (Fusion 360 y CorelDRAW) y fabricarlos mediante corte láser.
2. Seleccionar e integrar los componentes electrónicos: Raspberry Pi 5, Arduino, cámara CSI/USB, motoreductor de DC y drivers de potencia.
3. Configurar la infraestructura de comunicación IoT utilizando Podman para el despliegue del broker Mosquitto y el protocolo MQTT.
4. Entrenar e implementar un modelo YOLOv8 ligero para el reconocimiento visual de los estados *Verde*, *Pintón* y *Maduro*, así como defectos (*Moho* y *Podrido*).
5. Desarrollar el *script* de inferencia en Python que publique los resultados de detección (clase y confianza) en el tópico MQTT para su registro y actuación remota.

V. Metodología y Desarrollo Experimental

El desarrollo experimental se dividió en tres fases principales: Diseño y Fabricación, Integración de Hardware y Desarrollo de Software e IA.

Figura 1. Flujo de trabajo completo.

V-A. *Diseño Mecánico y Fabricación Digital*

El material elegido fue MDF de $3\,\mathrm{mm}$ por su bajo costo y la facilidad para ser procesado mediante corte láser.

V-A1. Diseño Paramétrico del Chasis: Se utilizó Autodesk Fusion 360 para el diseño 3D del chasis. Se empleó un diseño de uniones de tipo *finger joint* para lograr un ensamblaje robusto sin depender excesivamente de adhesivos. La estructura diseñada (Fig. **??**) alberga la Raspberry Pi 5 y mantiene la cámara rígidamente para minimizar vibraciones y mantener una distancia focal fija de $150\,\mathrm{mm}$ respecto a la cinta.

V-A2. Diseño y Optimización de la Cinta Transportadora: El diseño vectorial para el corte se realizó en CorelDRAW [4]. Se consideró un factor de tolerancia de $\pm 0{,}1\,\mathrm{mm}$ en los cortes internos para asegurar que los ejes y rodamientos se ajustaran por presión.

- **Transmisión:** La cinta mide $300\,\mathrm{mm}$ de largo y $80\,\mathrm{mm}$ de ancho. El eje de tracción se acopló al motoreductor DC de $12\,\mathrm{V}$ (relación de reducción 1:100), mientras que el eje pasivo se montó sobre rodamientos 608ZZ para minimizar la fricción. La banda de la cinta se fabricó con un material de goma antideslizante. Las piezas vectoriales se muestran en la Fig. 2.

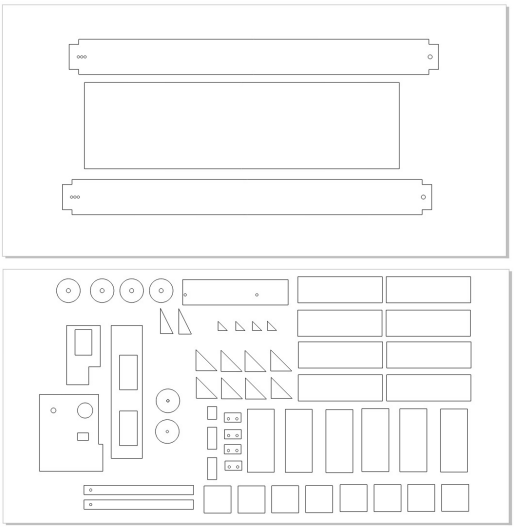


Figura 2. Diseño en CorelDRAW: (Sup) Rieles laterales; (Inf) Componentes mecánicos y optimización de corte (*nesting*).

V-B. *Integración de Hardware y Circuito de Control*

La arquitectura de control utiliza el principio de separación de tareas para optimizar el rendimiento.

V-B1. Diagrama de Conexiones y Esquemático: Para validar la lógica de control antes de la implementación física, se diseñó el circuito esquemático en la plataforma de simulación Tinkercad (Fig. 3). El sistema integra los siguientes subsistemas conectados al Arduino Mega 2560:

- **Detección de Presencia:** Un sensor ultrasónico HC-SR04 monitorea la entrada de la cinta. Al detectar un objeto a una distancia $d < 10\,\mathrm{cm}$, envía una señal de interrupción para activar la captura de imagen en la Raspberry Pi.
- **Sistema de Clasificación:** Dos servomotores SG90 actúan como mecanismo de desvío. Reciben señales PWM del Arduino para posicionarse en ángulos específicos $(0°, 90°)$ y dirigir el tomate a su contenedor correspondiente según la clasificación recibida vía Serial/MQTT.
- **Interfaz de Estado:** Se implementó un arreglo de indicadores LED para retroalimentación visual inmediata del estado de clasificación y diagnóstico de errores en tiempo real.

La placa verde en el esquema representa la interfaz lógica de comunicación (UART) con la Raspberry

Pi 5, la cual envía los comandos de actuación tras procesar la imagen con YOLOv8.

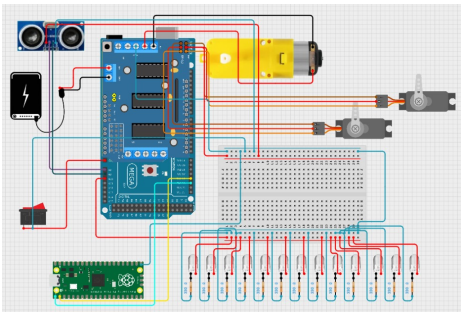


Figura 3. Diagrama esquemático de conexiones (Tinkercad). Se detalla la integración del Arduino Mega con los servomotores de clasificación, el sensor de presencia y la interfaz de comunicación con la unidad de IA.

V-C. Arquitectura y Diagrama de Flujo

La arquitectura lógica y física del sistema "Tomatait" integra subsistemas de control mecánico, visión artificial y comunicación inalámbrica. El proceso se divide en tres bloques funcionales: el Sistema Físico (Hardware), la Torre de Control (Red) y la Lógica de Actuación, tal como se detalla en la Figura 4.

El proceso automatizado sigue la siguiente secuencia lógica:

1. **Inicio y Detección Mecánica:** El ciclo inicia en el *Arduino Mega*, que mantiene la banda transportadora en movimiento continuo a una velocidad constante (PWM 70). El sistema monitorea el sensor ultrasónico hasta detectar un objeto a una distancia menor a 12 cm.
2. **Posicionamiento y Adquisición de Imagen:** Al confirmar la presencia del tomate, el Arduino ejecuta una rutina de posicionamiento, avanzando la banda durante 550 ms adicionales para centrar el producto bajo la cámara, deteniendo el motor momentáneamente. En este estado estático, la *Raspberry Pi 5* (ejecutando un contenedor Podman con YOLOv8) captura y analiza la imagen.
3. **Procesamiento de IA y Comunicación:** Si el modelo YOLOv8 detecta una clasificación válida (Maduro, Verde o Moho) con una confianza superior al 75 %, se genera un evento digital. La Raspberry Pi publica el resultado mediante el protocolo MQTT hacia el Broker (Laptop). Inmediatamente, el mensaje es retransmitido vía WiFi a la *Raspberry Pi Pico W*, que actúa como puente serial hacia el Arduino Mega.
4. **Sincronización y Actuación:** El Arduino Mega, tras recibir el carácter de clasificación ('R', 'U' o 'M'), reactiva la banda transportadora.

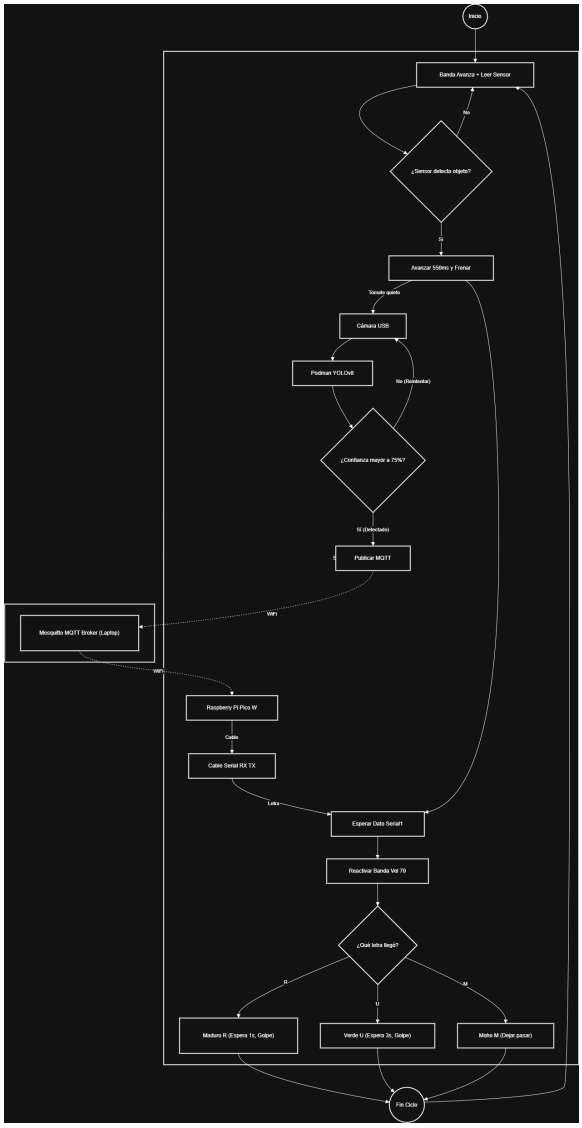


Figura 4. Diagrama de Flujo de la Arquitectura del Sistema Tomatait. Muestra la interacción entre los subsistemas de control (Arduino), visión (Raspberry Pi) y comunicación (MQTT).

Dependiendo de la clase recibida, el sistema ejecuta una espera de tiempo específica (sincronizada con la velocidad de la banda) antes de activar el servomotor correspondiente para clasificar el producto hacia la derecha o izquierda, o dejarlo pasar a la zona de descarte.

V-C1. Arquitectura Híbrida de Control:

- **High-Level Control (RPi 5):** Manejo del sistema operativo, captura de video, ejecución del modelo YOLOv8, comunicación Wi-Fi y publicación MQTT.
- **Low-Level Control (Arduino Uno):** Recepción de comandos seriales o MQTT y generación de señales PWM para el driver del motor y el

control de los servomotores MG995.

V-C2. Circuito de Potencia: El sistema de potencia se alimenta de una batería de 12 V. Se utilizó el driver L298N para controlar la dirección y velocidad del motoreductor de la cinta, ya que este driver soporta corrientes de hasta 2 A por motor, suficiente para el torque requerido. La velocidad de la cinta se regula mediante una señal PWM generada por el Arduino.

V-D. Métricas de Evaluación del Modelo

Para evaluar objetivamente el desempeño del modelo YOLOv8n en la tarea de clasificación de madurez, se utilizaron las métricas estándar de la literatura de visión por computador. Se define la matriz de confusión basada en los siguientes cuatro estados posibles para cada predicción: Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TN) y Falsos Negativos (FN). A partir de estos valores, se calculan las siguientes métricas:

V-D1. Precisión (Precision):: Cuantifica la exactitud de las detecciones positivas. Es crítica en este sistema para evitar que un tomate verde sea clasificado erróneamente como maduro.

$$Precision = \frac{TP}{TP + FP}$$

V-D2. Sensibilidad (Recall):: Mide la capacidad del modelo para encontrar todos los objetos relevantes (tomates) en la escena.

$$Recall = \frac{TP}{TP + FN}$$

V-D3. Precisión Media (mAP@0.5):: Es la media de la precisión promedio (AP) para todas las clases, calculada con un umbral de Intersección sobre Unión (IoU) de 0.5. Esta métrica es el estándar para comparar arquitecturas de detección de objetos.

$$mAP@0{,}5 = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{1}$$

Donde N es el número de clases (Verde, Pintón, Maduro).

V-E. Despliegue de Software e Inteligencia Artificial

V-E1. Configuración IoT y Contenedorización: El broker Mosquitto se desplegó en la RPi 5 utilizando Podman para aislar el entorno.

```
podman run -d –name mosquitto
-p 1883:1883 eclipse-mosquitto
```

El *script* principal en Python utiliza la librería `paho-mqtt` para publicar los mensajes en formato JSON en el tópico `robot/deteccion/ia`.

V-E2. Análisis y Preprocesamiento del Dataset: Para garantizar la robustez del modelo, se recolectó y analizó un dataset de 1500 imágenes. La Fig. 5 muestra la distribución de las instancias. Las etiquetas originales en inglés se mapearon a las categorías del estudio: *unripe* (Verde), *half_ripe* (Pintón) y *ripe* (Maduro). Se observa un desbalance natural con mayor presencia de tomates verdes y maduros, y una escasez crítica en la clase *mold* (564 instancias).

Para mitigar el desbalance y mejorar la detección de objetos pequeños, se utilizó la técnica de *Mosaic Data Augmentation*, nativa de YOLOv8. Como se muestra en la Fig. 6, esta técnica combina 4 imágenes de entrenamiento en un solo cuadro, variando la escala y el recorte, lo que obliga al modelo a aprender características en contextos complejos.

V-E3. Entrenamiento del Modelo: El entrenamiento del modelo YOLOv8n se realizó utilizando la librería `ultralytics` [1] en un entorno GPU externo. La función de pérdida (L) fue minimizada durante 100 épocas, donde L es una combinación de la pérdida de clasificación (L_{cls}), la pérdida de localización (L_{box}) y la pérdida del objeto (L_{obj}):

$$L = w_{cls} \cdot L_{cls} + w_{box} \cdot L_{box} + w_{obj} \cdot L_{obj}$$

El modelo resultante (`weights.pt`) fue exportado y transferido a la RPi 5 para su inferencia local.

V-E4. Flujo de Inferencia en Edge: El script de Python en la RPi 5 realiza la captura del *frame* mediante `cv2`. Tras la inferencia, si la confianza Confianza $> 0{,}41$ (umbral determinado por la curva F1), la RPi 5 publica la detección. El mensaje JSON publicado contiene la etiqueta de clase y el valor de confianza.

VI. Resultados y Análisis de Rendimiento

VI-A. Rendimiento Mecánico y de Flujo

La velocidad de la cinta se fijó en $v = 5\,\mathrm{cm/s}$. Dada la longitud de la zona de detección ($L_{det} = 20\,\mathrm{cm}$), el tiempo de permanencia de cada tomate bajo la cámara fue de:

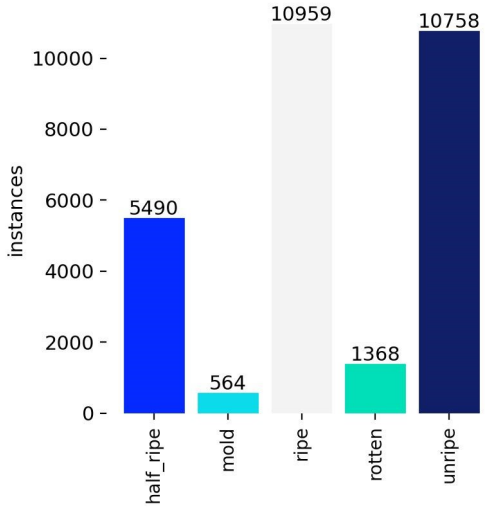
$$T_{permanencia} = \frac{L_{det}}{v} = \frac{0{,}20 \text{ m}}{0{,}05 \text{ m/s}} = 4\,\mathrm{s}$$

Este tiempo garantiza una ventana amplia de detección para la IA.

VI-B. Convergencia del Modelo e IA

El modelo YOLOv8n demostró una convergencia estable tras 50 épocas. La Fig. 8 muestra la disminución consistente de la pérdida, sin signos de sobreajuste significativo.

La Tabla I detalla el rendimiento exacto por clase obtenido en la validación (basado en curvas PR).



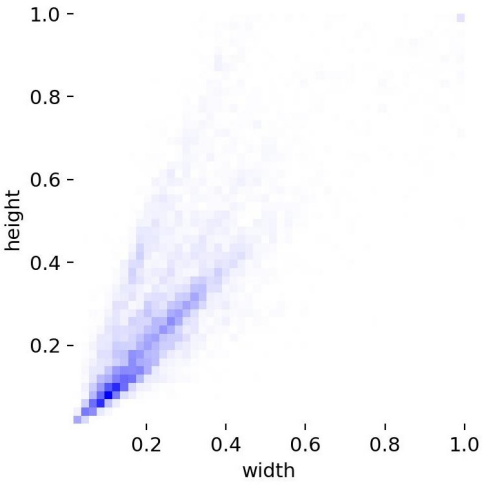
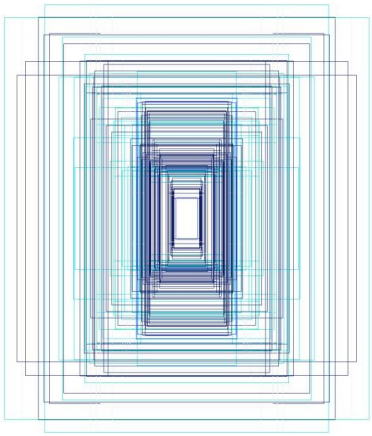


Figura 5. Estadísticas del Dataset: (A) Distribución de instancias por clase. (B) Mapa de calor de ubicación espacial (x,y) de los tomates en las imágenes.



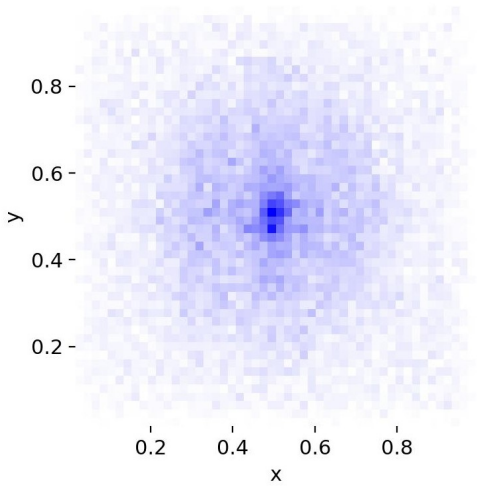


Figura 6. Mosaico de entrenamiento (Batch 0): Visualización del aumento de datos para mejorar la invarianza a la escala durante el entrenamiento.

Destaca el alto rendimiento en la detección de moho (0.932 mAP) a pesar de la escasez de datos, lo cual valida la efectividad del aumento de datos.

VI-C. Análisis de Confusión

La matriz de confusión normalizada (Fig. 9) revela los puntos críticos. La clase ”Pintón”(*half_ripe*) presenta la precisión más baja (74 %), confundiéndose frecuentemente con el fondo o con tomates maduros debido a la ambigüedad visual en la transición. Por el contrario, la clase *Unripe* alcanza un 95 % de precisión.

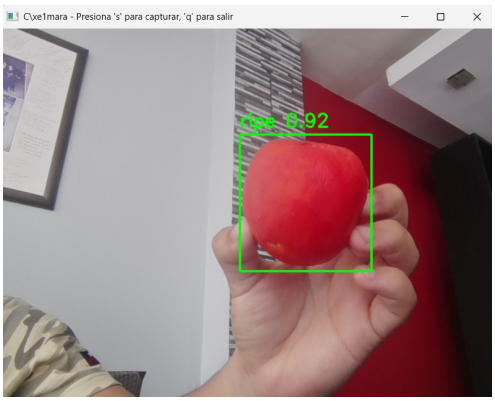


Figura 7. Detección de madurez en tiempo real mediante YOLOv8n. Se muestra la clase (*Verde*, *Pintón*, *Maduro*) y el porcentaje de confianza.

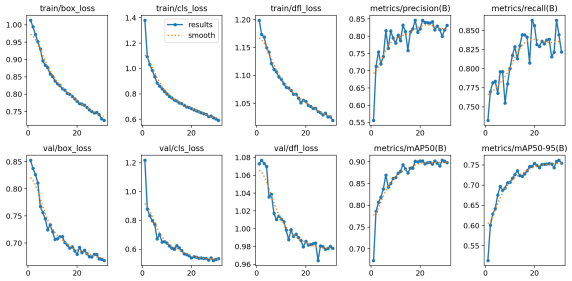


Figura 8. Curvas de entrenamiento: Pérdida (Loss) y Métricas a lo largo de 50 épocas. La convergencia es clara y estable.

Cuadro I
MÉTRICAS DE RENDIMIENTO POR CLASE (MAP@0.5)

Clase	mAP@0.5	Análisis
Unripe (Verde)	0.926	Excelente distinción visual.
Ripe (Maduro)	0.918	Clase mayoritaria.
Mold (Moho)	**0.932**	**Alta precisión por textura única.**
Rotten (Podrido)	0.896	Buena detección de deformaciones.
Half_ripe (Pintón)	0.834	Confusión en transición de color.
Global	**0.901**	**Promedio General**

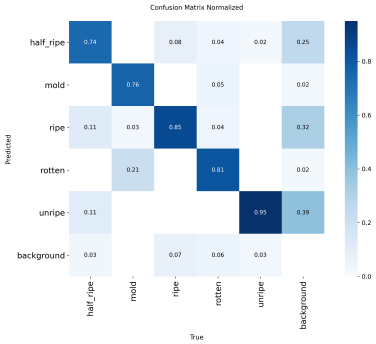


Figura 9. Matriz de Confusión Normalizada. Muestra el porcentaje de acierto por clase real vs predicha.

VI-D. Rendimiento en Edge Computing

El rendimiento se evaluó en términos de la tasa de cuadros que la RPi 5 puede procesar por segundo [2].

- **Tasa de Inferencia Pura:** 18 FPS.
- **Tasa Operacional (Incluyendo Captura y MQTT):** 12 FPS.

El tiempo de procesamiento por tomate es de $T_{proceso} = 1/12\,\text{FPS} \approx 83\,\text{ms}$. Para maximizar el balance entre precisión y recall en el borde, se determinó un **umbral de confianza óptimo de 0.41** basado en la curva F1, obteniendo un puntaje F1 máximo de 0.83.

VI-E. Estabilidad y Latencia de IoT

La comunicación MQTT demostró una latencia promedio inferior a 50 ms para la publicación y recepción de resultados de clasificación en la red local. Esto es crítico para garantizar que el sistema de actuación (futuro) reciba el comando antes de que el tomate salga de la zona de influencia.

VII. DISCUSIÓN

TomatAIT valida la premisa de que soluciones avanzadas de Agricultura de Precisión pueden ser implementadas de manera económica y efectiva utilizando plataformas de código abierto.

- **Viabilidad Económica:** El uso de MDF de 3 mm (corte láser) y la RPi 5 (bajo consumo) mantienen el costo de la unidad prototipo muy por debajo de los sistemas industriales, haciéndolo accesible para pequeñas y medianas empresas agrícolas [11].
- **Separación de Tareas:** La arquitectura híbrida es superior a la integración monolítica en un solo microcontrolador. Permite que la RPi 5 se centre exclusivamente en la tarea de IA (alto consumo de recursos), dejando el control de los motores (baja latencia) al Arduino.
- **Capacidad de Procesamiento:** La tasa operacional de 12 FPS es altamente competitiva para un dispositivo Edge y supera significativamente a las alternativas de VC que dependen de la nube.

Una limitación es la dependencia de condiciones de luz estables, ya que el color es el principal descriptor para el modelo de IA. El rendimiento podría degradarse bajo iluminación solar directa o variable.

VII-A. Limitaciones del Sistema

- **Capacidad de Procesamiento:** Menciona que el sistema actual depende de una iluminación controlada (interior) y que la luz solar directa podría afectar la precisión de la cámara.

- **Velocidad de la Cinta:**Explica que la velocidad de la cinta transportadora debe estar sincronizada con el tiempo de inferencia de la IA (aprox 30-50ms) para que el actuador golpee el tomate en el momento exacto.

VIII. Conclusiones

El proyecto TomatAIT ha logrado integrar exitosamente la fabricación digital de precisión con tecnologías avanzadas de IA y comunicación IoT.

1. Se diseñó y construyó un prototipo físico robusto y modular utilizando técnicas de fabricación digital.
2. Se validó que la Raspberry Pi 5 es una plataforma con capacidad suficiente para ejecutar modelos complejos como YOLOv8 en el borde con una tasa de cuadros adecuada para la clasificación automatizada.
3. La implementación del protocolo MQTT a través de contenedores Podman garantizó una comunicación IoT estable y desacoplada, esencial para el monitoreo y la futura integración con sistemas de gestión agrícola [3].
4. El sistema en su conjunto ofrece una solución eficiente, de bajo costo y alta velocidad para la clasificación de tomates, demostrando la madurez de la tecnología Edge AI en el sector agrícola.

IX. Trabajo Futuro

Para llevar a TomatAIT a una fase pre-comercial, se proponen las siguientes líneas de trabajo:

- **Implementación de la Actuación:** Integrar el mecanismo físico de desvío (servomotores o solenoides) controlado por el Arduino en respuesta a los comandos MQTT de la RPi 5.
- **Optimización por Cuantización:** Explorar la cuantización del modelo (de punto flotante a INT8) para aumentar el FPS y reducir el consumo de recursos de la RPi 5, posiblemente utilizando la Unidad de Procesamiento Neuronal (NPU) de la placa [12].
- **Análisis Hiperespectral/RGB-D:** Integrar cámaras multispectrales o sensores de profundidad para mejorar la robustez de la clasificación, permitiendo la detección de defectos internos y no solo superficiales [13].
- **Monitoreo en Tiempo Real:** Desarrollar un dashboard web (utilizando Node-RED o Grafana) suscrito al broker MQTT para la visualización remota de las métricas de clasificación, el estado de la batería y la temperatura del sistema.

Referencias

[1] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics. https://github.com/ultralytics/ultralytics.

[2] Raspberry Pi Foundation. (2024). Raspberry Pi 5 Documentation and Performance Benchmarks.

[3] Docker Inc. (2024). Podman & Containers Architecture: A Daemonless Approach.

[4] Corel Corporation. (2024). CorelDRAW User Guide and Technical Drawing Best Practices.

[5] R. A. F. G. A. L. N. Z. Z. S. C. C. M. L. W. L. M. G. L. A. J. L. P. B. F. S. L. T. C. M. H. X. S. P. (2021). Intelligent Classification and Grading of Tomatoes Based on Computer Vision and Machine Learning: A Review. *Journal of Food Engineering*, 306, 110688.

[6] W. L. L. Y. (2023). YOLOv8: Revolutionizing Object Detection for Real-time Applications. *ArXiv Preprint*.

[7] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5), 637-646.

[8] H. F. D. W. (2010). MQTT-S - A Protocol for Sensor Networks. *EAI Endorsed Transactions on Smart Cities*.

[9] S. J. C. J. B. (2019). Industry 4.0: A comprehensive review of the core technologies, applications and challenges. *IEEE Access*, 7, 42095-42112.

[10] M. K. K. A. H. (2020). A review of computer vision techniques for fruit recognition and classification. *Applied Soft Computing*, 86, 105672.

[11] K. C. M. T. L. C. (2022). Low-cost smart farming system using Raspberry Pi and IoT. *IEEE International Conference on Smart Information, Communication and Technology (SmartICT)*.

[12] C. M. Z. P. L. J. H. (2024). Towards efficient AI on micro-edge devices: A study on Raspberry Pi 5 NPU usage. *Journal of Embedded Systems*.

[13] J. G. D. L. B. (2018). Hyperspectral imaging for food safety and quality: a review. *Applied Spectroscopy Reviews*, 53(2), 163-181.

[14] Y. Yang and H. Ju, "Performance Comparison of Cherry Tomato Ripeness Detection Using Multiple YOLO Models,"*Semantic Scholar*, 2024.

[15] Z. Zheng, et al., "Tomato ripeness detection method based on improved YOLOv11 lightweight model,"*Frontiers of Agricultural Science and Engineering*, 2025.

[16] M. T. Okano, W. A. C. Lopes, et al., ."Enhancing Fruit Quality Assessment: A Real-Time Grading System Based on YOLO and Image Processing,"*Algorithms*, vol. 18, no. 8, 2025.

[17] A. Hassan et al., "Tomato Sorting System Based on Machine Vision and Raspberry Pi,"*Journal of Robotics and Control*, 2024.