

UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

Ingeniería en Sistemas de Información y Ciencias de la Computación

Curso: Arquitectura de Sistemas II

Ciclo: Octavo

Sección: B

Sede Central, Plan sábado Virtual

EcoVision Belt AI

Diseño e implementación de una banda transportadora inteligente para la identificación visual de productos mediante YOLOv8 y alimentación con apoyo solar

Informe técnico del Proyecto Final Integrador

Presentado por

Giancarlo Enrique Morales Monzón

Carné 0900 23 11702

Selvin Geovani Pérez Aquino

Carné 0900 23 9654

Guatemala, 2026

Contenido

Resumen ejecutivo.....	3
1. Introducción y planteamiento del problema.....	4
1.1 Identificación del problema.....	4
1.2 Justificación.....	4
1.3 Alcance.....	5
2. Objetivos.....	5
Objetivo general.....	5
Objetivos específicos.....	5
3. Marco conceptual	6
Visión artificial	6
YOLOv8.....	6
Dashboard.....	6
Alimentación solar y almacenamiento de energía	6
4. Arquitectura de la solución	6
Flujo general de información.....	7
4.1 Componentes principales.....	7
4.2 Flujo de una detección	8
5. Diseño del dashboard	8
Elementos sugeridos de la interfaz	8
6. Alimentación y sostenibilidad energética.....	8
Registro energético pendiente de completar	9
7. Seguridad, estabilidad y mantenimiento	9
8. Pruebas y evaluación	9
Matriz de pruebas propuesta	10
8.1 Indicadores que deben medirse.....	10
9. Limitaciones y trabajo futuro	11
10. Diagramas técnicos	11
10.1 Diagrama de arquitectura general.....	12
10.2 Diagrama de flujo del procesamiento	13
11. Código de referencia.....	14
11.1 Dependencias sugeridas	14
11.2 Captura de imagen e inferencia con YOLOv8	14
11.3 Ejemplo de servicio para enviar el resultado al dashboard	15
11.4 Ejemplo de consulta desde una página web	16
11.5 Ejemplo de configuración de red local.....	17

11.6 Consideraciones para adaptar el código	17
12. video.....	18
https://drive.google.com/drive/folders/173mgEFavhiOa1xSLFM_mus43mrHw-K9d?usp=sharing	18
13. Conclusiones.....	18
14. Referencias.....	19

Resumen ejecutivo

EcoVision Belt AI es una propuesta de automatización orientada a identificar productos que pasan por una banda transportadora mediante visión artificial. Una cámara captura la imagen del objeto ubicado frente a ella y el sistema procesa esa información para reconocer elementos visibles, como el tipo de producto o su marca, siempre que la clase correspondiente esté contemplada en el modelo de inteligencia artificial.

La solución contempla una Raspberry Pi Model B+ V1.2 como unidad de control del prototipo, un modelo de visión artificial basado en YOLOv8 y un dashboard que presenta al usuario el resultado de cada detección. La alimentación eléctrica se plantea mediante una power bank recargable con apoyo de un panel solar, con el propósito de mejorar la autonomía y reducir la dependencia de una toma de corriente durante demostraciones controladas.

El proyecto se relaciona con el caso de uso de visión por computadora en el borde. Sin embargo, antes de afirmar que cumple todas las condiciones técnicas del curso, es necesario verificar dónde se ejecuta la inferencia y medir el rendimiento del hardware. La Raspberry Pi Model B+ V1.2 dispone de recursos limitados y no figura entre los modelos de Raspberry Pi autorizados en las especificaciones entregadas, que mencionan Raspberry Pi 3B+, 4B o 5. Por ello, este informe diferencia la arquitectura propuesta de los resultados que todavía deben comprobarse y recomienda validar el sistema con hardware admitido si la evaluación exige cumplimiento literal.

El documento presenta el problema, los objetivos, los fundamentos conceptuales, la arquitectura, el flujo de operación, los componentes, el diseño energético, las pruebas recomendadas, las limitaciones y las oportunidades de mejora. Los tiempos, porcentajes de precisión, autonomía y resultados experimentales se dejan pendientes de medición para evitar presentar como comprobados datos que aún no han sido proporcionados.

1. Introducción y planteamiento del problema

En los procesos de almacenamiento, clasificación y distribución, reconocer productos de forma manual puede requerir tiempo y depender de la atención constante del operador. Una solución basada en visión artificial puede apoyar esta tarea al analizar imágenes y presentar información visual de manera organizada.

EcoVision Belt AI propone integrar una banda transportadora, una cámara, una unidad de control y un modelo de inteligencia artificial. El objeto se desplaza hasta el campo de visión de la cámara, la imagen se procesa y el resultado se muestra en un dashboard. El prototipo incorpora además una fuente de alimentación portátil que puede recargarse mediante un panel solar.

El principal reto técnico consiste en coordinar la captura de imágenes, la inferencia del modelo y la actualización de la interfaz sin perder estabilidad. También es necesario considerar que detectar un objeto no equivale automáticamente a identificar su marca. Para reconocer marcas específicas, el modelo debe contar con clases entrenadas y ejemplos representativos de esas marcas.

1.1 Identificación del problema

La identificación manual de productos puede provocar demoras, errores de clasificación y falta de uniformidad al registrar los resultados. Cuando el proceso depende de una persona que observa cada objeto, resulta más difícil mantener un registro constante y consultar rápidamente lo detectado.

El proyecto busca demostrar cómo la visión artificial puede asistir la identificación de productos en una banda transportadora. El sistema debe presentar de forma clara qué detectó el modelo y, cuando sea posible, indicar la clase reconocida y el nivel de confianza de la predicción.

Existe una segunda dificultad relacionada con el hardware. La Raspberry Pi Model B+ V1.2 tiene una capacidad de cómputo y memoria considerablemente menor que las plataformas modernas de borde. La ejecución directa de YOLOv8 puede resultar inviable o demasiado lenta según el sistema operativo, las dependencias y el modelo elegido. Esta condición debe verificarse mediante pruebas reales y no darse por resuelta solo por conectar los componentes.

1.2 Justificación

El desarrollo de EcoVision Belt AI permite aplicar conocimientos de arquitectura de sistemas, integración de hardware, procesamiento de imágenes, inteligencia artificial e interfaces de usuario. El proyecto vincula un proceso físico con una aplicación informática que transforma imágenes en información útil para el operador.

La alimentación mediante una power bank con recarga solar aporta un componente de eficiencia energética y facilita las demostraciones en lugares donde no se dispone de una conexión eléctrica cercana. La autonomía real dependerá de la capacidad de la batería, la potencia del panel, la irradiación disponible y el consumo total de la banda y los dispositivos electrónicos.

1.3 Alcance

El alcance funcional comprende el movimiento de productos por una banda transportadora, la captura de imágenes con una cámara, el análisis de las imágenes mediante un modelo YOLOv8, la recepción de los resultados y su presentación en un dashboard. También contempla documentar la alimentación mediante una power bank y un panel solar.

El sistema se plantea como un prototipo académico. No se afirma que sea apto para operación industrial continua, ni que reconozca cualquier producto o marca. La calidad de detección depende de las clases disponibles en el modelo, las condiciones de iluminación, el ángulo de la cámara, la distancia al producto y la calidad de las imágenes.

La documentación final deberá incorporar fotografías del montaje, diagrama real de conexiones, versiones de software, configuración del modelo y resultados medidos. Las pruebas que aún no se hayan ejecutado deben permanecer identificadas como pendientes.

2. Objetivos

Objetivo general

Diseñar y documentar un prototipo de banda transportadora inteligente que utilice visión artificial para detectar productos y mostrar los resultados en un dashboard, integrando una unidad de control de borde y una fuente de alimentación portátil con recarga mediante panel solar.

Objetivos específicos

- Integrar una cámara con la unidad de control para capturar imágenes de los productos durante su desplazamiento.
- Incorporar un flujo de análisis basado en YOLOv8 para obtener las clases detectadas y sus niveles de confianza.
- Diseñar una interfaz de dashboard que presente los resultados de forma comprensible y permita distinguir una detección válida de un resultado no reconocido.
- Documentar el recorrido de los datos desde la captura de la imagen hasta la presentación del resultado.
- Evaluar la estabilidad del sistema mediante pruebas repetibles de detección, latencia y continuidad de funcionamiento.
- Describir la alimentación energética y establecer un método para medir la autonomía real de la power bank y el aporte del panel solar.
- Identificar las limitaciones del hardware utilizado y proponer una configuración compatible con las restricciones técnicas del curso.

3. Marco conceptual

La computación en el borde consiste en procesar datos cerca del lugar donde se generan, en lugar de depender de un servicio remoto. En este proyecto, la captura y el análisis deben realizarse en equipos locales para reducir la dependencia de internet. Para demostrar que la inferencia es local, se debe documentar qué dispositivo ejecuta realmente el modelo y comprobar que el sistema puede operar sin enviar imágenes a la nube.

Visión artificial

La visión artificial permite que un sistema informático analice imágenes para localizar objetos y asignarles clases. Su desempeño depende de la calidad de las imágenes, del conjunto de datos y del modelo utilizado. Los resultados son estimaciones del modelo, no una garantía absoluta de identidad.

YOLOv8

YOLOv8 es una familia de modelos de visión artificial desarrollada por Ultralytics. Puede utilizarse para detección de objetos, clasificación y otras tareas. Para localizar productos dentro de una imagen se utiliza una variante de detección. Si el objetivo es identificar una marca concreta, esa marca debe estar representada en las clases del modelo o debe emplearse un procedimiento adicional apropiado.

Inferencia

La inferencia es la etapa en la que un modelo ya entrenado recibe una imagen nueva y produce predicciones. En el prototipo, las predicciones deben transformarse en datos que el dashboard pueda presentar, como nombre de clase, confianza y hora de detección.

Dashboard

El dashboard es la interfaz de supervisión del sistema. Puede mostrar la imagen capturada el estado de la cámara y el estado del procesamiento. La interfaz debe indicar con claridad cuando no se detecta un objeto o cuando el modelo no puede identificarlo con suficiente confianza.

Alimentación solar y almacenamiento de energía

La power bank almacena energía y entrega alimentación a los dispositivos compatibles. El panel solar repone parte de esa energía cuando las condiciones de luz lo permiten y cuando el controlador de carga está diseñado para esa combinación. La potencia del panel no garantiza por sí sola autonomía continua. Se debe comprobar la compatibilidad eléctrica, la corriente requerida y la capacidad efectiva de la batería.

4. Arquitectura de la solución

La arquitectura funcional se organiza en cinco etapas. La primera corresponde al movimiento físico del producto por la banda. La segunda captura la imagen mediante la cámara. La tercera gestiona la

imagen y la comunicación desde la unidad de control. La cuarta ejecuta la inferencia YOLOv8 en el dispositivo local que se haya validado para ello. La quinta recibe la predicción y actualiza el dashboard.

Flujo general de información

Producto sobre la banda → cámara → captura de imagen → unidad de control → inferencia YOLOv8 en hardware local validado → resultado estructurado → dashboard → visualización de clase y confianza.

La Raspberry Pi Model B+ V1.2 puede participar en el control del prototipo y en la adquisición o transmisión de datos si las interfaces y los recursos disponibles lo permiten. No obstante, la ubicación de YOLOv8 debe declararse según la implementación real. Si el modelo se ejecuta en otro equipo local, dicho equipo debe figurar en el diagrama y en la lista de hardware. No debe describirse como inferencia ejecutada por la Raspberry Pi si esta solo envía imágenes a otro dispositivo.

La consigna del curso limita el hardware de borde a Raspberry Pi 3B+, 4B o 5, Orange Pi de los modelos indicados o mini PC integrada de bajo consumo. También solicita red privada, direccionamiento estático, switch Gigabit, tolerancia a fallos y redistribución de tareas. La configuración descrita inicialmente con una única Raspberry Pi Model B+ V1.2 no demuestra por sí sola el cumplimiento de esos requisitos. Si estas condiciones son obligatorias para el proyecto, será necesario ampliar o modificar la arquitectura.

4.1 Componentes principales

Banda transportadora: mecanismo físico que desplaza los productos hacia la zona de inspección. Debe mantener un movimiento estable y ubicar los objetos dentro del campo visual de la cámara.

Cámara: obtiene las imágenes que utilizará el sistema. La resolución, el enfoque, la iluminación y la posición deben ajustarse para reducir reflejos, sombras y desenfoque.

Raspberry Pi Model B+ V1.2: unidad de control prevista en el prototipo. Antes de asignarle procesamiento intensivo deben comprobarse compatibilidad del sistema operativo, memoria disponible, bibliotecas de cámara y rendimiento.

Modelo YOLOv8: componente de inteligencia artificial que genera las predicciones. Se debe registrar la variante exacta, el formato del archivo del modelo, las clases reconocidas y si se utilizó un modelo preentrenado o uno entrenado con imágenes propias.

Dashboard: interfaz que recibe los resultados y los presenta al operador. Puede mostrar nombre de clase, confianza, imagen de referencia y mensajes de estado.

Power bank y panel solar: conjunto de alimentación portátil. Se deben registrar sus especificaciones nominales, los dispositivos que alimenta y el método de recarga utilizado.

Equipo adicional de inferencia: solo aplica si YOLOv8 no se ejecuta en la Raspberry Pi principal. Si se incorpora, debe documentarse como nodo local y comprobarse que cumple las restricciones del curso.

4.2 Flujo de una detección

1. El producto se coloca sobre la banda transportadora.
2. La cámara captura una imagen cuando el producto entra en la zona de inspección.
3. La aplicación obtiene la imagen y la prepara para el modelo.
4. YOLOv8 procesa la imagen en el dispositivo local definido por la arquitectura.
5. El modelo devuelve las clases detectadas y sus valores de confianza.
6. La aplicación filtra o presenta los resultados conforme al umbral configurado.
7. El dashboard actualiza la información para que el usuario pueda consultar la detección.
8. El sistema queda listo para procesar el siguiente producto.

Si no existe una detección válida, el dashboard debe mostrar un mensaje como Producto no identificado en lugar de inventar una marca. El valor de confianza debe interpretarse como una salida del modelo y no como una garantía universal de exactitud.

5. Diseño del dashboard

La interfaz debe priorizar la lectura rápida de los resultados y mantener una distribución sencilla. un indicador del estado del sistema, un panel de imagen, un bloque con el producto detectado y una sección de eventos recientes.

Elementos sugeridos de la interfaz

- Vista de imagen: fotografía recibida y, si está disponible, imagen anotada con recuadros de detección.
- Resultado: clase detectada.
- Mensajes de error: indicaciones breves para fallos de cámara, ausencia de resultados o pérdida de comunicación.

El dashboard no debe afirmar que identifica una marca si el modelo solo reconoce una categoría genérica, como botella, caja o lata. Para obtener marcas, el conjunto de clases debe incluirlas y la precisión debe evaluarse con productos reales.

6. Alimentación y sostenibilidad energética

La solución utiliza una power bank como fuente de energía portátil y un panel solar para recargarla. Esta combinación puede facilitar el transporte del prototipo y reducir la necesidad de una toma eléctrica durante periodos de demostración. El funcionamiento continuo depende del balance entre la energía almacenada, la energía recuperada mediante el panel y el consumo de los componentes.

Para evaluar la autonomía se debe medir el tiempo de operación desde una carga conocida hasta el nivel de batería definido como límite seguro. También conviene registrar si la banda, la cámara y la unidad de procesamiento se alimentan desde la misma power bank o desde salidas separadas.

La conexión del panel debe respetar las especificaciones del fabricante de la power bank o utilizar un controlador de carga compatible. No se debe conectar un panel directamente a una entrada que no admita su voltaje y corriente. Las especificaciones exactas del panel y de la batería deben agregarse cuando se conozcan.

Registro energético pendiente de completar

Capacidad nominal de la power bank: pendiente de confirmar.

Potencia nominal del panel solar: pendiente de confirmar.

Consumo total medido del sistema: pendiente de medir.

Tiempo de funcionamiento por carga: pendiente de medir.

Tiempo de recarga en condiciones de prueba: pendiente de medir.

7. Seguridad, estabilidad y mantenimiento

La instalación debe mantener los cables sujetos y alejados de las partes móviles de la banda. La fuente de alimentación debe estar protegida frente a cortocircuitos y sobrecargas de acuerdo con las especificaciones de los fabricantes. Antes de operar, se debe comprobar que no existan conexiones expuestas, piezas sueltas ni interferencias con el desplazamiento de los productos.

En el software, se recomienda registrar los errores de captura, los fallos de comunicación y las predicciones vacías. La aplicación debe recuperarse de errores previsibles sin bloquear el dashboard. Si el sistema utiliza una red local, conviene restringir el acceso a los servicios y evitar exponer puertos innecesarios.

El mantenimiento debe incluir limpieza de la lente, revisión de la tensión y alineación de la banda, inspección de cables, verificación de la alimentación y revisión de los registros de la aplicación.

8. Pruebas y evaluación

La evaluación debe realizarse con productos conocidos, condiciones de iluminación controladas y una cantidad de repeticiones registrada. Cada ensayo debe comparar la salida del sistema con la identificación correcta definida previamente. No se deben presentar resultados estimados como si fueran mediciones reales.

Matriz de pruebas propuesta

Prueba de captura: colocar un producto en la zona de inspección y comprobar que la cámara obtiene una imagen utilizable. Resultado real: pendiente de completar.

Prueba de detección: presentar una clase incluida en el modelo y comprobar que aparece en el dashboard. Resultado real: pendiente de completar.

Prueba de producto no conocido: presentar un objeto que no corresponda a las clases disponibles y comprobar que el sistema no atribuye una marca sin fundamento. Resultado real: pendiente de completar.

Prueba de latencia: medir el tiempo entre la captura y la aparición del resultado. Realizar al menos veinte repeticiones y calcular promedio y máximo. Resultado real: pendiente de medir.

Prueba de estabilidad: ejecutar el sistema durante un periodo definido y registrar errores, reinicios o pérdida de comunicación. Resultado real: pendiente de completar.

Prueba energética: medir el tiempo de funcionamiento y el comportamiento de la power bank durante el uso real. Resultado real: pendiente de medir.

Prueba sin internet: desconectar el acceso a internet y verificar que la inferencia continúa si el modelo y todas las dependencias ya están instalados localmente. Resultado real: pendiente de completar.

Prueba de tolerancia a fallos: solo aplica si se implementa una arquitectura con más de un nodo. Se debe detener de manera controlada un nodo de procesamiento y comprobar si la tarea se redistribuye a otro nodo. La arquitectura inicial descrita no incluye evidencia de este mecanismo.

8.1 Indicadores que deben medirse

Exactitud de identificación: número de predicciones correctas dividido entre el total de casos evaluados, expresado en porcentaje.

Latencia de detección: tiempo desde la captura de la imagen hasta la visualización del resultado.

Tasa de detección: proporción de productos de prueba para los que el sistema produce una detección válida.

Falsos positivos: casos en los que el sistema identifica una clase que no corresponde al producto mostrado.

Autonomía energética: tiempo de operación medido bajo una carga de trabajo definida.

Disponibilidad: proporción del tiempo de prueba durante el cual el sistema permanece operativo.

Los indicadores deben calcularse con datos recogidos durante las pruebas. En el informe definitivo se recomienda incluir una tabla con el número de ensayos, aciertos, errores, latencia promedio, latencia máxima y autonomía medida.

9. Limitaciones y trabajo futuro

La primera limitación es la capacidad del hardware. La Raspberry Pi Model B+ V1.2 no corresponde a los modelos de Raspberry Pi indicados como válidos en las especificaciones del curso. Además, su memoria y capacidad de procesamiento pueden impedir la ejecución práctica de YOLOv8 con una experiencia aceptable. La viabilidad debe confirmarse en el dispositivo real.

La segunda limitación es el alcance del modelo. Un modelo general puede reconocer categorías de objetos sin conocer las marcas comerciales. Para reconocer marcas específicas se requiere un conjunto de datos adecuado, entrenamiento o ajuste del modelo y pruebas con diferentes presentaciones, tamaños y condiciones de iluminación.

La tercera limitación es energética. La autonomía depende de la carga conectada y de la energía solar disponible. Sin mediciones de consumo y recarga no es posible garantizar un número concreto de horas de funcionamiento.

Como mejoras futuras se propone utilizar una Raspberry Pi 4B o 5, o una mini PC de bajo consumo aprobada por la consigna, para la inferencia local; optimizar el modelo y la resolución de entrada; incorporar iluminación estable; construir un conjunto de imágenes propio de los productos; guardar un historial de detecciones; y, si se exige alta disponibilidad, implementar una red privada con direccionamiento estático, switch Gigabit y un nodo de respaldo con redistribución documentada de tareas.

10. Diagramas técnicos

Los diagramas siguientes representan la arquitectura funcional propuesta. Deben compararse con el montaje real antes de la entrega y actualizarse si el cableado o la distribución del procesamiento son diferentes.

10.1 Diagrama de arquitectura general

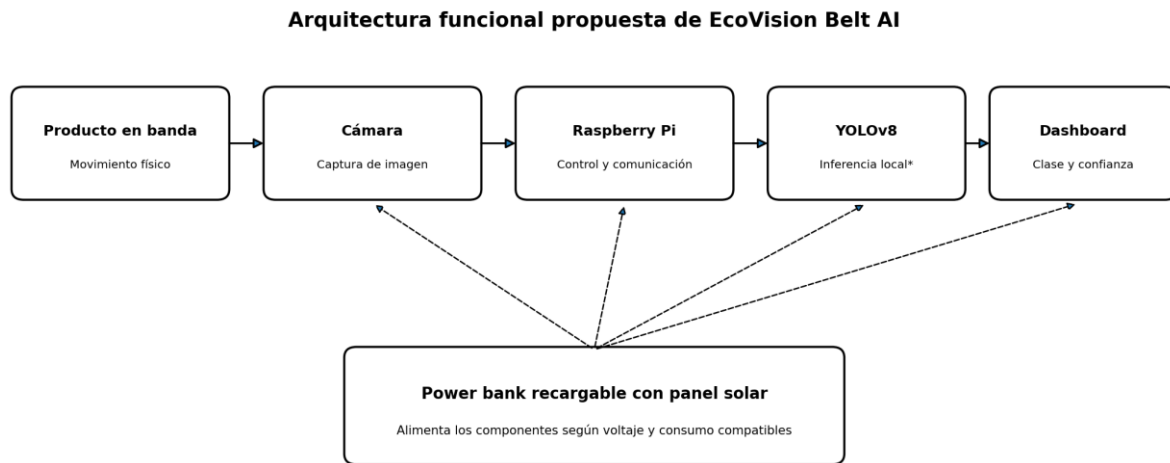


Figura 1. Flujo de datos y alimentación energética del prototipo. La ubicación de la inferencia YOLOv8 debe reflejar el equipo que realmente ejecuta el modelo.

10.2 Diagrama de flujo del procesamiento

Diagrama de flujo de una detección

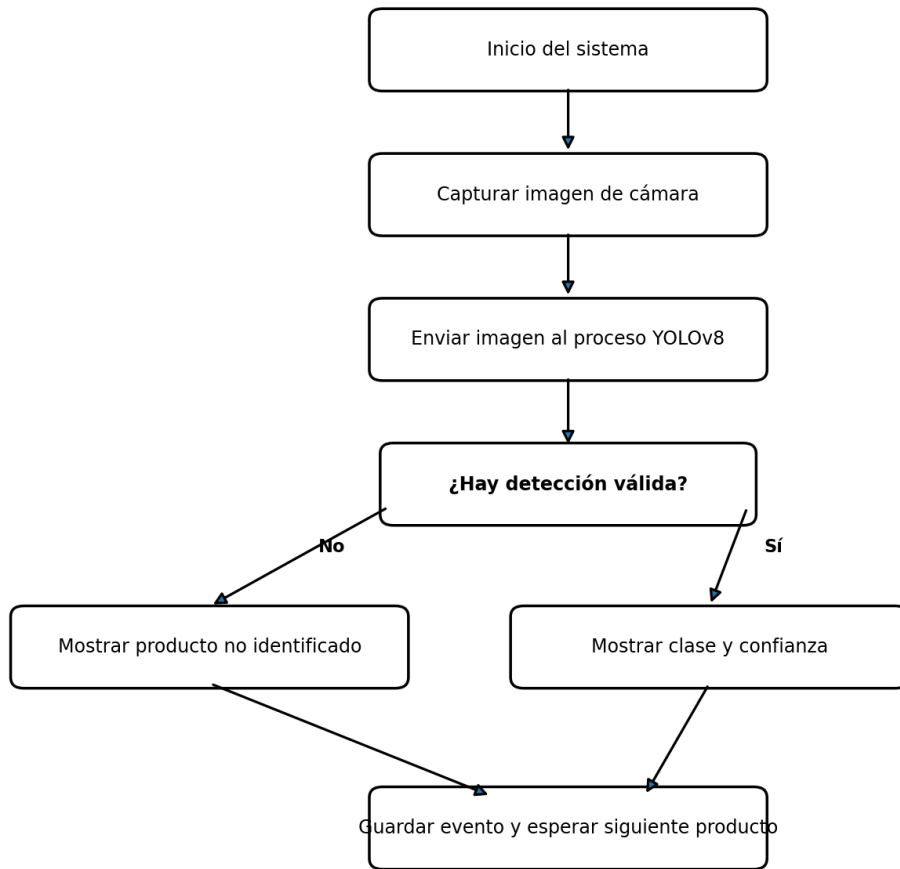


Figura 2. Lógica prevista desde la captura hasta la presentación del resultado en el dashboard.

11. Código de referencia

Los siguientes fragmentos son una base de desarrollo, no una afirmación de que el programa ya se encuentre instalado o validado en el prototipo. Deben ajustarse a la cámara, al sistema operativo, al formato del modelo y al dispositivo que ejecutará YOLOv8. En particular, la Raspberry Pi Model B+ V1.2 puede no soportar de forma práctica las dependencias y el rendimiento requeridos por YOLOv8.

11.1 Dependencias sugeridas

En un equipo local compatible con Python y el modelo seleccionado, se puede preparar un entorno virtual e instalar las bibliotecas necesarias. La instalación debe verificarse según la arquitectura del procesador y la versión del sistema operativo.

```
python3 -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
pip install ultralytics opencv-python flask
```

11.2 Captura de imagen e inferencia con YOLOv8

Ejemplo de referencia para una cámara accesible por OpenCV y un archivo de pesos compatible. El índice de cámara y la ruta del modelo deben ajustarse. La ejecución real dependerá del hardware y de las bibliotecas instaladas.

```
import cv2
from ultralytics import YOLO

MODEL_PATH = "best.pt" # Pesos entrenados o modelo compatible
CAMERA_INDEX = 0

model = YOLO(MODEL_PATH)
camera = cv2.VideoCapture(CAMERA_INDEX)

if not camera.isOpened():
    raise RuntimeError("No se pudo abrir la cámara")

try:
    while True:
        ok, frame = camera.read()
        if not ok:
            print("No se pudo capturar la imagen")
            break

        results = model.predict(
```

```

        source=frame,
        conf=0.50,
        imgsz=320,
        verbose=False
    )

    for result in results:
        for box in result.bboxes:
            class_id = int(box.cls[0])
            confidence = float(box.conf[0])
            label = model.names[class_id]
            print(f"Producto: {label} | Confianza: {confidence:.2f}")

    cv2.imshow("EcoVision Belt AI", results[0].plot())
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
finally:
    camera.release()
    cv2.destroyAllWindows()

```

11.3 Ejemplo de servicio para enviar el resultado al dashboard

Este ejemplo ilustra cómo publicar una detección como JSON mediante una API local. Para usarlo en el proyecto, la función de inferencia debe conectarse con el proceso que captura y analiza las imágenes. No debe ejecutarse como un sistema separado sin integrar esa lógica.

```

from flask import Flask, jsonify
from datetime import datetime

app = Flask(__name__)

# Sustituir estos valores por la salida real del modelo.
last_detection = {
    "product": "Sin detección",
    "confidence": 0.0,
    "timestamp": None,
    "status": "waiting"
}

@app.get("/api/detection")
def get_detection():
    return jsonify(last_detection)

```

```
def update_detection(product, confidence):
    last_detection.update({
        "product": product,
        "confidence": round(float(confidence), 3),
        "timestamp": datetime.now().isoformat(timespec="seconds"),
        "status": "detected" if product else "not_detected"
    })

if __name__ == "__main__":
    # Usar solo en una red de laboratorio controlada.
    app.run(host="0.0.0.0", port=5000, debug=False)
```

11.4 Ejemplo de consulta desde una página web

El dashboard puede consultar periódicamente la API local y mostrar el nombre del producto, la confianza y la hora de la última detección. Este fragmento requiere que el navegador tenga acceso a la dirección IP del equipo que ejecuta Flask.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>EcoVision Belt AI</title>
</head>
<body>
  <h1>EcoVision Belt AI</h1>
  <p>Producto: <strong id="product">Esperando...</strong></p>
  <p>Confianza: <strong id="confidence">--</strong></p>
  <p>Última actualización: <span id="time">--</span></p>

  <script>
    async function refreshDetection() {
      try {
        const response = await fetch("/api/detection");
        if (!response.ok) throw new Error("API no disponible");
        const data = await response.json();
        document.getElementById("product").textContent =
          data.product || "Producto no identificado";
        document.getElementById("confidence").textContent =
          `${Math.round((data.confidence || 0) * 100)}%`;
        document.getElementById("time").textContent =
          data.timestamp || "Sin datos";
```

```

    } catch (error) {
        document.getElementById("product").textContent =
            "Error de conexión con el sistema";
    }
}
refreshDetection();
setInterval(refreshDetection, 1000);
</script>
</body>
</html>

```

11.5 Ejemplo de configuración de red local

Solo si la arquitectura final incluye más de un equipo conectado por Ethernet, se debe asignar una dirección IP fija a cada nodo y documentar la subred utilizada. Los siguientes valores son ilustrativos y no deben copiarse sin comprobar la red real.

Nodo de control: 192.168.100.10/24
 Nodo de inferencia: 192.168.100.20/24
 Equipo del dashboard: 192.168.100.30/24
 Máscara de subred: 255.255.255.0
 Gateway: según la red local utilizada

11.6 Consideraciones para adaptar el código

- Confirmar el sistema operativo y la compatibilidad de Python, OpenCV y Ultralytics con el hardware real.
- Sustituir best.pt por la ruta real del modelo y verificar las clases que reconoce.
- Si se requiere identificar marcas, entrenar o ajustar el modelo con imágenes representativas de esas marcas.
- Integrar update_detection con la salida real de YOLOv8 antes de conectar el dashboard.
- Si la inferencia se ejecuta en otro equipo local, documentar ese equipo en la arquitectura y en la defensa.
- No exponer el servicio Flask a internet ni usar el servidor de desarrollo como despliegue industrial.
- No afirmar tolerancia a fallos ni balanceo de carga hasta implementar y probar esos mecanismos.

12. video

https://drive.google.com/drive/folders/173mgEFavhiOa1xSLFM_mus43mrHw-K9d?usp=sharing

13. Conclusiones

EcoVision Belt AI propone integrar transporte físico, captura de imágenes, visión artificial y visualización de resultados en un solo flujo de trabajo. Su valor académico consiste en demostrar cómo una imagen puede convertirse en información presentada al usuario mediante una arquitectura de hardware y software.

YOLOv8 puede utilizarse para la detección o clasificación de objetos, pero el resultado depende de las clases disponibles y del entrenamiento del modelo. La identificación de marcas debe respaldarse con clases específicas y pruebas, por lo que no puede asumirse únicamente por utilizar inteligencia artificial.

La alimentación mediante power bank y panel solar añade portabilidad y una oportunidad para estudiar el consumo energético. La autonomía real debe calcularse a partir de mediciones del prototipo.

Para cerrar el proyecto con rigor académico, es indispensable verificar el equipo donde se ejecuta la inferencia, registrar las versiones de software y documentar pruebas reales. También se debe revisar la compatibilidad del hardware con las restricciones del curso, especialmente si se requiere una arquitectura de varios nodos con red privada y tolerancia a fallos.

14. Referencias

Ultralytics. Documentación oficial de YOLOv8. <https://docs.ultralytics.com/models/yolov8/>

Ultralytics. Documentación de predicción e inferencia. <https://docs.ultralytics.com/modes/predict/>

Raspberry Pi Ltd. Documentación oficial de hardware Raspberry Pi. <https://www.raspberrypi.com/documentation/>

Universidad Mariano Gálvez de Guatemala. Especificaciones generales del Proyecto Final Integrador proporcionadas para el curso.

Nota de integridad académica: las referencias deben revisarse y complementarse con las fuentes efectivamente consultadas durante el desarrollo. La configuración de hardware, las pruebas y los resultados deben corresponder al prototipo real.