UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"ROBOT SEGUIDOR DE OBJETOS CON VISIÓN ARTIFICIAL EMBEBIDA"

Trabajo de grado previo a la obtención del título de Ingeniero en Mecatrónica

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:

Stiven Alexander Montero Chandi

DIRECTOR:

PhD. Carlos Xavier Rosero Chandi

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

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	
APELLIDOS Y NOMBRES:	Montero Chandi Stiven Alexander

DATOS DE LA OBRA	
TÍTULO:	"Robot seguidor de objetos con visión artificial embebida"
AUTOR:	Montero Chandi Stiven Alexander
FECHA:	13/05/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
DIRECTOR /ASESOR:	PhD. Carlos Xavier Rosero Chandi MsC. Cosme Damián Mejía Echeverria

2. 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 13 días del mes de Mayo de 2026

EL AUTOR:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
STIVEN ALEXANDER
MONTERO CHANDI

Nombre: Montero Chandi Stiven Alexander

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

Certificación del director del trabajo de grado

En mi calidad de director del trabajo de grado “Robot seguidor de objetos con vision artificial embebida”, presentado por el egresado Stiven Alexander Montero Chandi, que opta por el título de ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, 12 de mayo de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI

PhD. Carlos Xavier Rosero Chandi

Director de Tesis

Universidad Técnica del Norte

Facultad de Ingeniería en Ciencias Aplicadas

APROBACIÓN DEL COMITÉ CALIFICADOR

El Tribunal Examinador del trabajo de titulación, “Robot seguidor de objetos con vision artificial embebida”. Elaborado por Stiven Alexander Montero Chandi, previo a la obtención del título de Ingeniero en Mecatrónica, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte.

Ibarra, 12 de mayo de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI

PhD. Carlos Xavier Rosero Chandi

Director de Tesis

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
COSME DAMIAN MEJIA ECHEVERRIA

MsC. Cosme Damián Mejía Echeverria

Asesor de Tesis

Dedicatorias

"El éxito es la suma de pequeños esfuerzos repetidos día tras día."

Dedico este trabajo de titulación a mis padres, por su sacrificio, ejemplo y por siempre creer en mí, a mi pareja, Evelyn Ramos por su apoyo incondicional, comprensión y presencia constante en cada etapa de este camino; a mi hermano, Christian, por constituir un ejemplo de capacidad y conocimientos, cuya fortaleza ha sido un referente de superación en cada etapa de mi trayectoria universitaria. Finalmente, mi reconocimiento para todas aquellas personas que, de manera directa o indirecta, brindaron su respaldo y contribuyeron a la consecución de este objetivo profesional.

Agradecimientos

Quiero expresar un profundo reconocimiento al PhD. Xavier Rosero, tutor de este trabajo de titulación, por su guía experta, paciencia y por los conocimientos técnicos compartidos a lo largo de esta investigación. Su dirección académica ha sido fundamental para la consecución de los objetivos planteados y el rigor metodológico durante todo el proyecto. De igual manera, al MsC. Cosme Mejía, quien en su calidad de asesor aportó con su criterio profesional y observaciones críticas, contribuyendo significativamente al perfeccionamiento técnico de este trabajo.

Tabla de contenidos

Índice de figuras

Índice de tablas

Resumen

El crecimiento exponencial de la robótica industrial a nivel global y su creciente adopción en el Ecuador han generado una demanda de sistemas automatizados capaces de optimizar procesos mediante reconocimiento y seguimiento autónomo de objetos. No obstante, la oferta de prototipos económicos se limita mayormente a robots seguidores de línea básicos o sistemas industriales de alto costo y difícil mantenimiento local. El presente trabajo describe el desarrollo de un robot móvil capaz de realizar seguimiento autónomo mediante visión artificial embebida. La metodología se centra en proponer una arquitectura de hardware basada en un microcomputador Raspberry Pi 4B, una unidad de procesamiento de tensor Coral USB Accelerator y una plataforma de robot móvil. Mediante estos elementos se implementa algoritmos de procesamiento de imágenes bajo el modelo SSD MobileNet v2 y un sistema de control proporcional para el movimiento autónomo del robot basándose en los datos obtenidos por el sistema de visión artificial. A través de diversas pruebas experimentales, se valida la estabilidad del sistema y su capacidad de respuesta dinámica ante cambios en la trayectoria de los objetivos. El desarrollo no solo demuestra la viabilidad de implementar sistemas de visión y control sobre una plataforma, sino que también ofrece una alternativa económica y eficiente para el fortalecimiento de la educación y la industria tecnológica local.

Palabras clave: Robótica móvil, visión por computadora, sistemas embebidos, sistemas de control.

Abstract

The exponential growth of industrial robotics worldwide and its increasing adoption in Ecuador have created a demand for automated systems capable of optimizing pro- cesses through autonomous object recognition and tracking. However, the availability of affordable prototypes is largely limited to basic line-following robots or high-cost industrial systems that are difficult to maintain locally. This paper describes the development of a mobile robot capable of autonomous tracking using embedded computer vision. The methodology focuses on proposing a hardware architecture based on a Raspberry Pi 4B microcomputer, a Coral USB Accelerator tensor processing unit, and a mobile robot platform. Using these components, image processing algorithms are implemented under the SSD MobileNet v2 model, along with a proportional control system for the robot's autonomous movement based on data obtained by the computer vision system. Through various experimental tests, the system's stability and its dynamic response to changes in the targets' trajectories are validated. This development not only demonstrates the feasibility of implementing vision and control systems on a platform, but also offers an economical and efficient alternative for strengthening local education and the technology industry.

Keywords: Mobile robotics, computer vision, embedded systems, control systems.

Capítulo I

Introducción

1.1. Planteamiento del problema

Los robots se han abierto paso durante los últimos años de manera significativa en la industria. Una muestra de esto es que dentro del sector industrial, más específicamente en el campo de la automatización, tan solo en el año 2021 se instalaron más de 500 000 nuevos robots en la industria, según la Federación Internacional de Robótica (IFR) el stock operativo es de 3,5 millones de unidades, esto es tres veces más que hace una década [1]. Además, se proyecta un incremento en el tamaño del mercado de robótica, pasando de USD 114.67 mil millones en 2023 a USD 258.36 mil millones para el año 2028, con una tasa de crecimiento anual compuesta del 17.64 % [2].

El Ecuador, no ajeno a este crecimiento, pues muestra una tendencia al aumento global en el uso de robots en la industria [3]. Este aspecto es crucial, ya que ayuda a las empresas a optimizar procesos y costos. Para el pais, los sistemas robóticos industriales incluyen mejoras en la calidad de productos e incentivos a la ciencia y tecnología, así como un incremento en la producción. Todo esto permite a los empresarios analizar el costo-beneficio y el impacto en su producción y rentabilidad, aumentando así la demanda por la robótica industrial en las empresas [4].

El seguimiento de objetos es de gran relevancia en una variedad de campos como biología, estudios sociales, educación, seguridad, entre otros. Por ejemplo, en biología, es esencial seguir

y observar el comportamiento de organismos particulares para comprender mejor sus patrones y características. En la actualidad, se han llevado a cabo estudios relacionados con el seguimiento de objetos en áreas como vigilancia, navegación de vehículos, deportes, gestión de tráfico y seguridad.

Sin embargo, la capacidad de los seres humanos para realizar esta tarea no siempre es precisa o efectiva, ya que está limitada por las capacidades visuales individuales. Por tanto, la aplicación de estrategias basadas en visión por computadora puede automatizar estas tareas y facilitar el seguimiento de objetos mediante la identificación y representación de patrones. Esto es especialmente relevante en la industria de la robótica y otros campos donde el seguimiento preciso de objetos es esencial [5].

En el mercado actual, se encuentran pocos prototipos de robots con visión artificial embebida para el seguimiento de objetos que sean económicamente accesibles. En la mayoría de los casos se cuenta con robots diseñados para el seguimiento de líneas, que, aunque útiles en su propio contexto, emplean sensores distintos, como los ultrasónicos, para detectar la línea a seguir.

Adicionalmente, la mayoría de las empresas especializadas en la construcción de este tipo de robots están ubicadas en el extranjero. Esto conlleva a que el mantenimiento y la adquisición de piezas para estos robots resulten costosos y de gestión complicada.

Así, se propone realizar un robot seguidor de objetos con visión artificial embebida, que suplirá esta falta que existe en un mercado que se encuentra en auge principalmente en la industria que busca optimizar sus procesos. Además su desarrollo contribuirá con una nueva herramienta que supla necesidades académicas en los campos de robótica y visión artificial.

1.2. Objetivos

1.2.1. General

Desarrollar un robot seguidor de objetos equipado con un sistema de visión artificial para seguimiento autónomo.

1.2.2. Específicos

- Proponer una arquitectura basada en microcomputador y en unidad de procesamiento de tensor para el procesamiento de imágenes y el reconocimiento de objetos.
- Implementar algoritmos de procesamiento de imágenes y control que aprovechen las capacidades de cálculo del hardware para la identificación, clasificación de objetos y movimiento del robot.
- Validar los resultados garantizando la interacción fluida entre el control y una plataforma mecánica de robot móvil.

1.3. Justificación

En el ámbito industrial, la implementación de este tipo de robot con capacidades de reconocimiento y seguimiento autónomo representa un avance significativo en términos de eficiencia operativa y seguridad laboral. La ausencia de alternativas similares de bajo costo en el mercado supone una oportunidad valiosa para empresas que requieren soluciones específicas para tareas de manipulación de objetos en entornos desafiantes o peligrosos, como en la industria química. La capacidad de ejecutar trabajos de alto riesgo sin exponer a los trabajadores a situaciones potencialmente peligrosas es un componente crucial para la preservación de la integridad física y la salud ocupacional.

Además, la incorporación de robots seguidores de objetos con visión artificial embebida conlleva un incremento sustancial en la productividad industrial. Estos sistemas pueden operar de forma continua sin fatiga, lo que se traduce en una reducción significativa de los tiempos de inactividad y una optimización de los procesos de producción. La capacidad de reemplazar y mantener los robots de manera eficiente minimiza los periodos de interrupción en la cadena de producción, potenciando así la competitividad de las empresas.

Desde una perspectiva de investigación y desarrollo, este proyecto establece una base sólida para futuras expansiones y mejoras tecnológicas. La posibilidad de evolucionar hacia un robot completamente autónomo, con capacidades de procesamiento de datos más avanzadas y la integración de sensores ultrasónicos, abre un abanico de posibilidades para aplicaciones más complejas y especializadas en diversos sectores industriales y científicos.

En el ámbito educativo, este proyecto se presenta como una herramienta pedagógica invaluable para la formación de estudiantes en ingeniería. La exposición al funcionamiento y desarrollo de tecnologías robóticas enriquecerá la experiencia de aprendizaje, fomentando la adquisición de conocimientos en programación, electrónica y robótica. Esta interacción directa con un prototipo funcional permitirá a los estudiantes aplicar teorías y conceptos aprendidos en un entorno práctico y realista, preparándolos para enfrentar los desafíos tecnológicos del futuro.

1.4. Alcance

Se propone el desarrollo, construcción y pruebas de un prototipo de robot seguidor de objetos con visión artificial embebida, capaz de reconocer ciertos objetos y seguirlos de manera autónoma.

El sistema contará con una unidad de procesamiento de tensor, que integrará una cámara de alta resolución junto con un acelerador de procesamiento dedicado para un análisis eficiente de las imágenes capturadas. Esta combinación permitirá el reconocimiento y seguimiento de los objetos de interés de manera precisa y en tiempo real.

El microcontrolador (microprocesador) seleccionado actuará como el centro de control del robot, encargado de procesar la información proveniente de la unidad de procesamiento de tensor y coordinar las acciones del robot en función de los objetos detectados. Este componente será fundamental para la toma de decisiones autónomas y la ejecución de los movimientos de seguimiento.

En cuanto a la plataforma de robótica móvil, se empleará un módulo prefabricado que incluirá los elementos esenciales como baterías, controladores, motores, llantas y chasis. Esta elección se basa en la conveniencia de contar con una estructura sólida y funcional, permitiendo focalizar los esfuerzos en la integración de los componentes de visión artificial con la plataforma móvil.

Capítulo II

Análisis literario

2.1. Antecedentes

Diversos estudios precedentes demuestran cómo la integración de visión artificial y algoritmos de procesamiento de imágenes optimiza las estrategias de control de robots, permitiendo así a los sistemas robóticos interactuar de manera más autónoma y precisa con su entorno. En este sentido, destacan investigaciones como [6] donde se diseña y desarrolla un robot seguidor de línea mediante visión artificial, para esto se usa una cámara web como dispositivo para la adquisición de imágenes y se determina como mejor opción el uso del módulo de Raspberry Pi como unidad central de procesamiento. Asimismo, la arquitectura física se realiza mediante diseño asistido por computadora (CAD). El software usado para el modelamiento de la plataforma de robot móvil AutoCAD y se materializa a través de impresión 3D.

Bajo esta misma línea de desarrollo, en [7] se profundiza en el tratamiento de la información obtenida por el sistema de visión, para esto se analizan las etapas de preprocesamiento esenciales antes de la ejecución de los algoritmos de control. En dicha investigación se destaca la importancia de transformar el espacio cromático de BGR a HSV para mitigar el impacto de las variaciones lumínicas, seguido de procesos de binarización y la obtención de bordes. El filtrado de imágenes permite depurar la imagen original, facilitando la extracción de características críticas de la información visual. Complementariamente, el desarrollo mecánico del prototipo se fundamenta en un diseño de chasis robusto utilizando el software SolidWorks, lo que garantiza la integridad física de la plataforma y la correcta distribución de los componentes electróni-

cos. En cuanto a la dinámica de movimiento del robot, se implementa un algoritmo de control Proporcional-Integral-Derivativo (PID) para gestionar la respuesta de los actuadores. Además la parte de procesamiento cuenta con dos microcontroladores de Arduino para los motores y una Raspberry como microcomputador que se encarga del procesamiento de las imágenes.

La transición hacia enfoques basados en aprendizaje profundo permite superar las limitaciones de los métodos de visión para detectar formas o elementos convencionales. Además del reconocimiento de objetos mediante la captura de imágenes con el uso de cámara, en [8] se detalla el entrenamiento de modelos de detección bajo el ecosistema de TensorFlow, este entorno permite la identificación de patrones complejos mediante redes neuronales convolucionales. Para el procesamoento y la puesta en marcha de estos algoritmos, la investigación destaca el uso de la Raspberry Pi como microcomputador central, esto demuestra que es una plataforma robusta para ejecutar tareas de inferencia en tiempo real, esta integración resulta fundamental para procesar grandes volúmenes de datos visuales sin comprometer la movilidad o el consumo energético del robot.

Bajo una perspectiva analítica, la eficacia de los sistemas de navegación autónoma reside en la interpretación de las imágenes como estructuras matriciales, ya que cada píxel de una imagen se le asigna un valor numérico, este valor representa información numérica procesable mediante operaciones algebraicas. En [9], se describe cómo el preprocesamiento de estas matrices como normalización y reescalado es indispensable para optimizar la entrada de información hacia arquitecturas de aprendizaje profundo. En el estudio se establece una comparativa técnica entre los modelos más prominentes de la industria: Faster R-CNN, SSD y YOLO. En particular, se resalta la arquitectura YOLO por su capacidad de predecir clases y cajas delimitadoras en un único paso convolucional.

No obstante, el procesamiento de imágenes bidimensionales no representa la única alternativa tecnológica en la percepción robótica. En [10] presenta el desarrollo de un robot móvil que utiliza una cámara RGB-D para la detección y el rastreo de objetos en tiempo real. Este dispositivo de adquisición de datos funciona mediante la captura simultánea de información cromática y datos de profundidad, por lo que, a cada píxel de la imagen se le asigna un valor de distancia métrica. El trabajo describe la implementación de una arquitectura que combina el algoritmo YOLOv5 para el reconocimiento visual y el método DeepSORT para el seguimiento persistente.

Extendiendo estas capacidades hacia entornos con presencia humana, en [11] se aborda de

manera integral la detección, seguimiento y monitoreo de personas en tiempo real. Esta investigación implementa el ecosistema de TensorFlow para el despliegue de la arquitectura YOLO, destacando su capacidad para realizar inferencias a alta velocidad sin sacrificar la precisión, este enfoque permite la identificación de individuos en la escena y facilita el seguimiento persistente de trayectorias en escenarios dinámicos, demostrando así, ser una solución robusta para sistemas de vigilancia y asistencia robótica.

En complemento a los avances en arquitecturas neuronales, en [12] se analiza la optimización del rendimiento mediante la integración de unidades de procesamiento tensorial (TPU) en sistemas embebidos. El estudio destaca cómo el uso de un acelerador de hardware como el TPU, específicamente la Coral USB Accelerator, reduce drásticamente la latencia de inferencia en comparación con el procesamiento basado exclusivamente en la unidad central (CPU) de la Raspberry Pi. Esta configuración permite alcanzar altas tasas de cuadros por segundo (FPS) en tareas de detección de objetos, lo que consolida a los dispositivos de borde como una solución eficiente que equilibra la velocidad de procesamiento y el bajo consumo energético.

En relación con el control de trayectorias en entornos dinámicos, en [13] se detalla la implementación de un controlador Proporcional-Integral-Derivativo (PID) para la corrección de trayectoria en tiempo real, donde el sistema procesa la información visual para calcular el error de desplazamiento respecto al eje central de la ruta, este desplazamiento se representa como el error en sistemas dinámicos. El algoritmo utiliza los datos obtenidos por el sistema de visión para ajustar de forma dinámica la velocidad angular y lineal de los motores. La investigación destaca que la arquitectura modular de ROS permite una comunicación eficiente entre el procesamiento de imágenes y la etapa de control PID asignada al movimiento del robot.

2.2. Bases teóricas

2.2.1. Visión artificial y procesamiento de imágenes

La visión artificial es la encargada de procesar imágenes haciendo uso de programación en un microprocesador o unidad que interprete el lenguaje de programación, estas técnicas de procesamiento de imágenes transforman los datos de una imagen o un fotograma de video para interpretarlos mediante píxeles con valores numéricos y así desarrollar operaciones para una acción específica [14]. En el ámbito de la robótica móvil, esta tecnología actúa como el

sensor principal para la percepción del entorno, permitiendo la identificación de trayectorias y la detección de objetivos en tiempo real.

El proceso de adquisición de datos visuales comienza cuando la energía lumínica reflejada por los objetos atraviesa el sistema óptico de la cámara. Esta luz impacta sobre un sensor fotosensible, usualmente de tecnología CMOS o CCD, el cual se compone de una matriz de fotoceldas que transforman los fotones en señales eléctricas, estos impulsos se transforman en una matriz de píxeles, donde cada valor numérico representa una propiedad del color o la luminosidad en un punto específico del espacio.

2.2.2. Adquisición de la imagen

Para la adquisición de imágenes es imprescindible tener en cuenta el dispositivo encargado de capturar la imagen o información del entorno. Este proceso constituye el primer eslabón en la cadena de percepción de un robot móvil, permitiendo la transición del mundo físico al dominio digital mediante diversos fenómenos físicos y ópticos.

2.2.3. Cámaras convencionales y digitalización

En una cámara digital estándar, la captura de información inicia cuando los fotones provenientes del entorno atraviesan el lente o sistema de lentes e impactan sobre un sensor de estado sólido, este sensor es denominado como tecnología CMOS (Complementary Metal-Oxide-Semiconductor). Luego cada celda fotosensible o píxel del sensor convierte la energía lumínica en una carga eléctrica mediante el efecto fotoeléctrico [15]. Una vez que se realiza este proceso la carga analógica se procesa a través de un convertidor analógico-digital (ADC), este convertidor es el encargado de cuantificar la señal para formar una matriz nimérica que relaciona a los píxeles con la intensidad de luz capturada en el sensor. Este proceso se observa detalladamente en Figura 2.1.

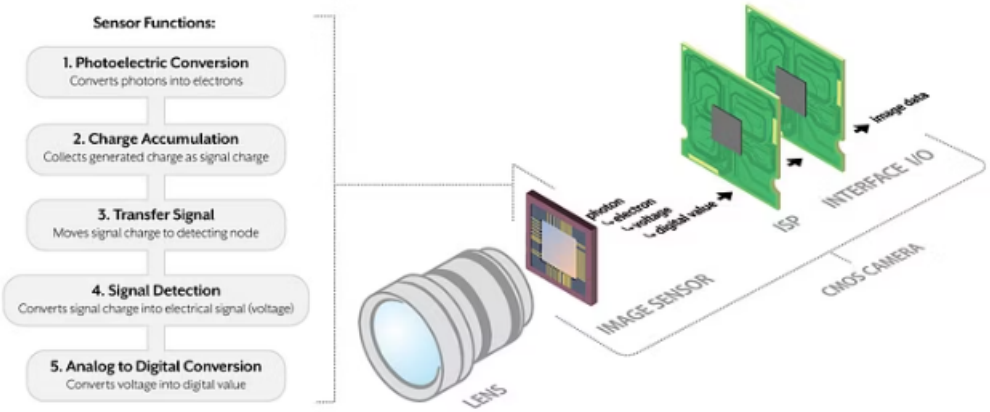


Figura 2.1: Arquitectura de una cámara digital [16].

2.2.4. Cámaras que captan profundidad

Este tipo de dispositivos se destacan por obtener una comprensión espacial del entorno, su característica principal es la de medir la distancia entre el sensor y los objetos, generando así un mapa de profundidad. Una metodología común de este tipo de sistemas es la visión estéreo, que utiliza dos sensores ópticos desplazados horizontalmente para capturar la escena desde ángulos ligeramente distintos, esto crea disparidad entre ambos puntos de vista y mediante técnicas de triangulación geométrica, el sistema determina la profundidad de cada punto en la imagen [15, 17]. Otra tecnología prevalente es el sensor de tiempo de vuelo (Time of Flight o ToF), el cual emite un haz de luz infrarroja y mide el desfase temporal o el tiempo exacto que tarda la señal en regresar al receptor. Este sistema se describe en la Figura 2.2 donde se muestra como el sensor envía y recibe la señal encargada de medir la profundidad.

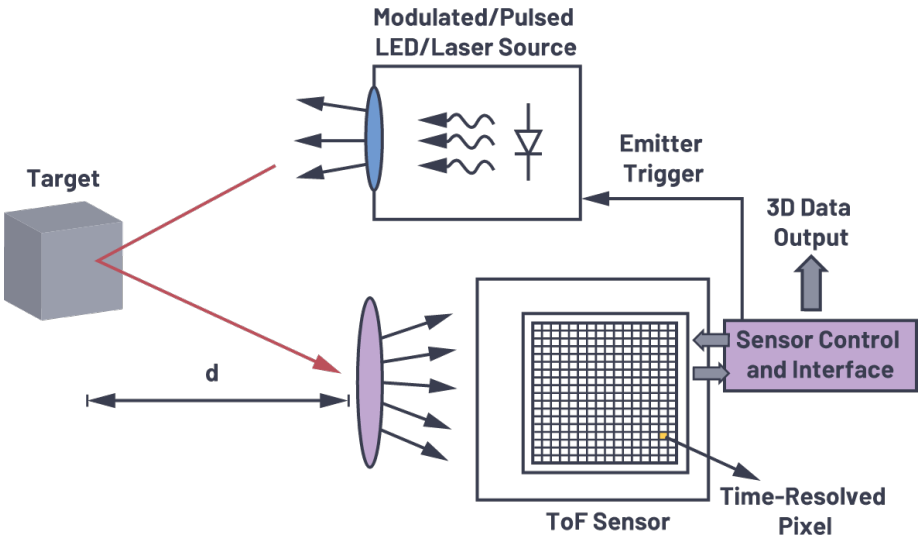


Figura 2.2: Funcionamiento de un sensor ToF [18].

2.2.5. Sistemas LiDAR

Existe también otros dispositivos capaces de reconocer el entorno y obtener datos como son dispositivos LIDAR (Light Detection and Ranging) que opera bajo el mismo principio que se muestra en la Figura 2.2, sin embargo este dispositivo crea un mapa de puntos tridimensional que representa la geometría del entorno, o sistemas de radiofrecuencia. Sin embargo, estos sistemas no se consideran dentro del ámbito de visión artificial general, ya que su funcionamiento se basa en la emisión y recepción de ondas electromagnéticas, y pertenecen a una rama avanzada de la percepción robótica que trasciende el uso de la visión pasiva convencional.

2.2.6. Preprocesamiento de imágenes

El procesamiento de imágenes constituye una etapa esencial para transformar los datos capturados por un sensor en una estructura matricial adecuada para la interpretación computacional, asegurando que la información sea procesable mediante operaciones algebraicas. En lo que respecta al redimensionamiento, el estudio en [19] describe esta técnica como una operación necesaria para ajustar las dimensiones de la imagen a los requisitos específicos de los algoritmos de procesamiento o de los dispositivos de visualización. Dicha tarea se fundamenta en procesos de interpolación ya que permite calcular nuevos valores de píxeles basándose en la información

de la matriz original lo que garantiza que la imagen conserve su integridad estructural y coherencia visual al cambiar de escala. Asimismo, el trabajo detalla la importancia de la conversión de espacios de color,destacando la transición de modelos RGB hacia escala de grises o hacia el modelo HSV. El autor explica que estas transformaciones facilitan la segmentación de objetos al permitir el análisis de componentes como el matiz y la saturación de manera independiente a la intensidad lumínica.

También se profundiza la aplicación de filtros espaciales para la reducción de ruido y el realce de bordes, esto depura la señal visual de artefactos no deseados generados por la naturaleza electrónica del sensor. Finalmente, el documento aborda la binarización mediante técnicas de umbralización, procedimiento que permite separar de manera clara los elementos de interés con respecto al fondo de la imagen.

En la Figura 2.3 se muestra el preprocesamiento de una imagen e ilustra la secuencia de transformaciones aplicadas a una señal visual. El proceso inicia con la conversión a escala de grises, ajuste de brillo y binarización de la imagen.

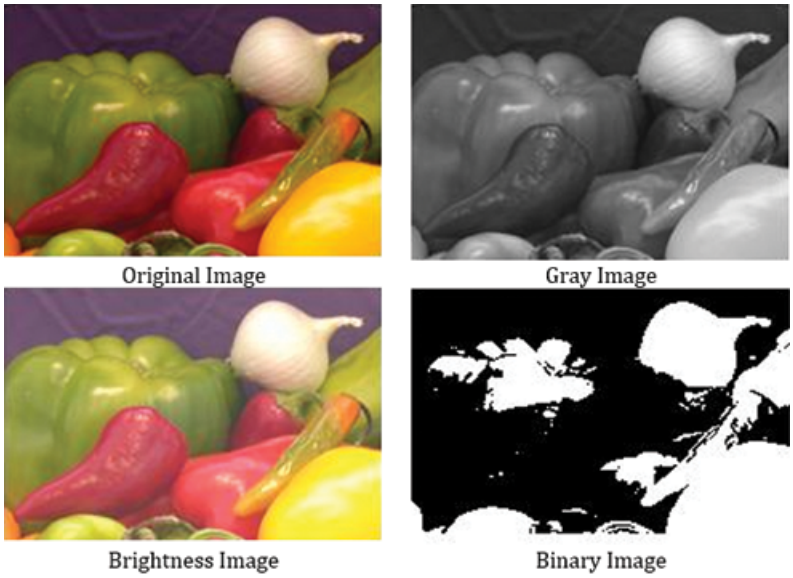


Figura 2.3: Técnicas de preprocesamiento de imágenes [20].

2.3. Tipos de robots móviles

Existen varios tipos de robots móviles, pero principalmente se pueden clasificar en robots móviles de ruedas como en la Figura 2.4- a, de tipo oruga como se representa en Figura 2.4- b y de patas como se muestra en Figura 2.4- c, tal como su nombre lo indica, su clasificación dependerá del método que use para moverse, esto dependerá mucho del tipo de trabajo que esté destinado a realizar el robot y principalmente por qué tipo de terrenos se va a desplazar. El sistema de locomoción es el que determina la interacción entre la plataforma y el entorno de operación.

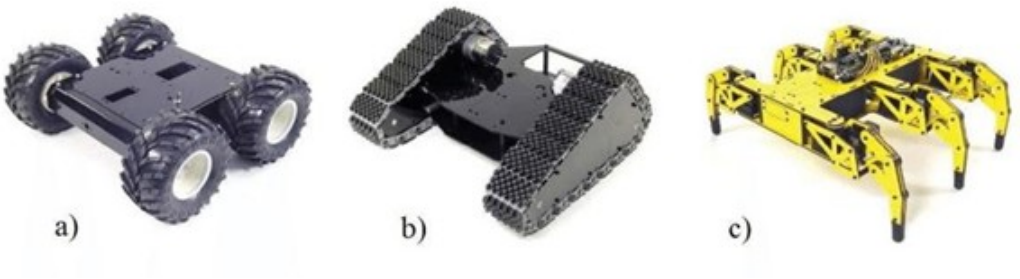


Figura 2.4: Tipos de robots móviles [21].

2.3.1. Locomoción mediante sistemas de ruedas

Mediante una implementación mecánica relativamente sencilla, los sistemas de locomoción basados en ruedas constituye el mecanismo de locomoción más extendido en la robótica móvil y en los vehículos creados por el ser humano debido a su capacidad para alcanzar niveles de eficiencia energética superiores. El diseño de estos robots se divide primordialmente en cuatro clases de ruedas: la rueda estándar, la rueda castor (orientable con eje desplazado), la rueda sueca (Swedish) y la rueda esférica, diferenciándose cada una por su grado de restricción cinemática y su impacto en la maniobrabilidad del chasis. Este tipo de locomoción a diferencia de otros sistemas, el equilibrio no es un problema central, ya que tres puntos de contacto son suficientes para garantizar la estabilidad estática del robot [15].

2.3.2. Locomoción tipo oruga

Este tipo de locomoción se caracteriza por reorientar el robot mediante la rotación de las bandas de tracción a diferentes velocidades o en direcciones opuestas, este tipo de robots mejora significativamente su maniobrabilidad y tracción en terrenos sueltos o irregulares en comparación con los diseños de ruedas convencionales. Sin embargo, el centro exacto de rotación resulta difícil de predecir y el consumo energético extremadamente ineficiente en superficies de alta fricción hacen un punto débil en este tipo de sistemas.

2.3.3. Locomoción de robots articulados

Este tipo de locomoción se caracteriza por establecer una secuencia de contactos puntuales entre la máquina y la superficie, lo cual otorga una adaptabilidad y maniobrabilidad superiores en terrenos accidentados. Sin embargo, para garantizar una marcha estática estable, un robot requiere un mínimo de seis patas, estas características junto con la complejidad mecánica y el elevado costo energético representan desafíos significativos que se deben tener en cuenta para la locomoción de un robot dinámico. En [15] se encuentra mas ejemplos de locomoción aplicados en industria y educación, así como también locomoción híbrida, que se refiere a la combinación de dos tipos de locomoción.

2.4. Microcomputador

El microcomputador se constituye como un sistema de procesamiento integral que utiliza una unidad central de procesamiento, módulos de memoria y puertos de entrada y salida en una configuración física compacta [22], tal como se ilustra en la Figura 2.5. Este dispositivo es capaz de ejecutar programas de propósito general y gestionar diversos periféricos de hardware de manera simultánea, operando mediante una arquitectura que interconecta sus componentes a través de un bus de datos común. El diseño del microcomputador permite la integración de bibliotecas especializadas y lenguajes de programación de alto nivel, facilitando la traducción de instrucciones algorítmicas en acciones de control deterministas o en el procesamiento de flujos de datos complejos [23].

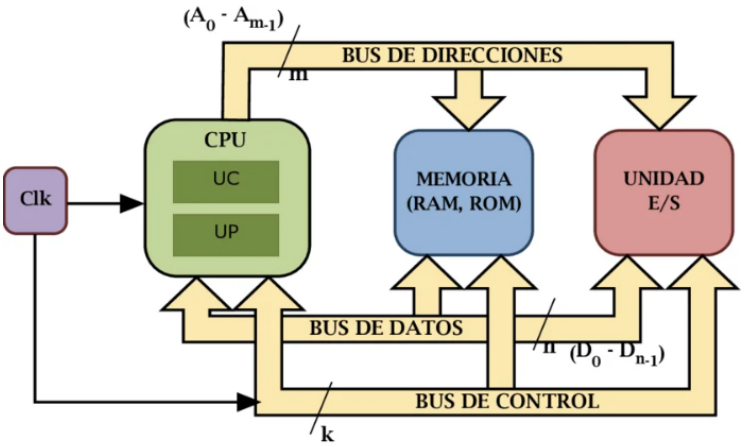


Figura 2.5: Estructura de un microcomputador [24].

2.4.1. Raspberry Pi 4

Se define a la Raspberry Pi 4 como un microcomputador de placa única que se caracteriza caracterizado por su arquitectura compacta y alta eficiencia, tal como se ilustra en la Figura 2.6. Este dispositivo consta de un solo circuito impreso un sistema en chip (SoC) que alberga un procesador basado en arquitectura ARM, un motor de procesamiento gráfico y memoria RAM de alta velocidad. Este dispositivo ofrece compatibilidad con una amplia gama de sistemas operativos, fundamentados mayoritariamente en el núcleo de Linux, lo que garantiza un entorno de ejecución versátil y robusto para la administración de recursos de hardware [25].

Figura 2.6: Raspberry Pi 4 [26].

2.4.2. BeagleBone Black

El BeagleBone Black es una plataforma de desarrollo de bajo costo respaldada por una comunidad global, orientada a la implementación de sistemas basados en el procesador ARM Cortex-A8. Este dispositivo facilita el arranque acelerado de sistemas operativos Linux y simplifica la etapa inicial del desarrollo al otorgar al usuario la facultad de diseñar circuitos personalizados. Asimismo, la placa integra múltiples terminales de propósito general que permiten la conexión y el control directo de diversos sensores y componentes electrónicos, tal como se ilustra en la Figura 2.7 [27].

Figura 2.7: Placa BeagleBone Black [27].

2.4.3. NVIDIA Jetson Nano

La NVIDIA Jetson Nano es un microcomputador de placa única, diseñado específicamente para la ejecución de aplicaciones de inteligencia artificial y visión computacional en sistemas embebidos como se muestra en la Figura 2.8. Este dispositivo integra una GPU basada en la arquitectura Maxwell con 128 núcleos CUDA [28], esto permite el procesamiento paralelo de múltiples redes neuronales simultáneamente. A diferencia de otros microcomputadores de propósito general, esta plataforma se optimiza para tareas de inferencia de alto rendimiento y operaciones computacionales demandantes.

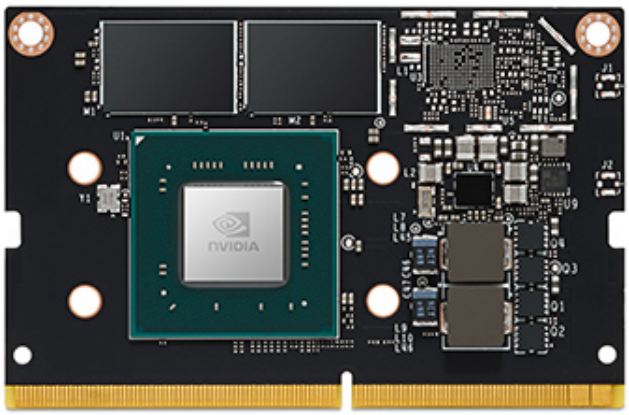


Figura 2.8: NVIDIA Jetson Nano [28].

2.5. TPU

La Unidad de Procesamiento de Tensores (TPU, por sus siglas en inglés) es un circuito integrado de aplicación específicamente para acelerar tareas relacionadas con el procesamiento de datos basado en tensores, siendo especialmente eficientes en operaciones utilizadas en redes neuronales y aprendizaje profundo [29].

En contraste con la Unidad Central de Procesamiento (CPU) y la Unidad de Procesamiento Gráfico (GPU), la TPU se configura específicamente para optimizar el rendimiento en operaciones con matrices de gran volumen. Esto es de gran ayuda para el procesamiento de video ya que se descompone el video en imágenes individuales y cada imagen se representa como una matriz de píxeles, lo que permite a la TPU realizar cálculos de manera eficiente.

Un tensor se establece como una generalización matemática de los escalares, vectores y matrices hacia un espacio de n dimensiones, en el álgebra lineal computacional, el rango de un tensor define su dimensionalidad. Un escalar es un tensor de rango 0, un vector, un tensor de rango 1, y una matriz se describe como un tensor de rango 2 [30]. Una imagen a color se representa como un tensor de rango 3, donde las dimensiones corresponden a la altura (h), el ancho (w) y los canales de color (c) y se denota matemáticamente como:

$$\mathcal{T} \in \mathbb{R}^{h \times w \times c} \tag{2.1}$$

2.5.1. Coral USB Acelerator TPU

El acelerador Coral USB TPU se presenta como un coprocesador de alto rendimiento de Google, diseñado para integrarse con los sistemas operativos de mayor difusión en el ámbito de la computación embebida y el desarrollo de inteligencia artificial. Este dispositivo permite ejecutar operaciones de inferencia de manera eficiente, optimizando el consumo energético y reduciendo la latencia en el procesamiento de modelos de aprendizaje profundo, el hardware alcanza una capacidad de procesamiento de 4 billones de operaciones por segundo (TOPS, por sus siglas en inglés) en un diseño de forma compacto, lo que facilita su implementación en plataformas de recursos limitados [31], tal como se ilustra en la Figura 2.9.

Figura 2.9: Google Coral USB Accelerator [31].

2.5.2. NVIDIA A100

La NVIDIA A100 Tensor Core GPU se posiciona como una unidad de procesamiento de alto rendimiento diseñada para la aceleración de inteligencia artificial y computación científica a gran escala. Este hardware integra una capacidad de memoria GPU de 80 GB [32], lo que facilita la gestión de volúmenes masivos de datos y permite ejecutar procesos de aprendizaje profundo con una velocidad hasta tres veces superior en comparación con arquitecturas previas del mismo fabricante. Debido a su compleja infraestructura interna y a la alta densidad de sus núcleos de procesamiento, el dispositivo presenta dimensiones físicas significativas, tal como se observa en

la Figura 2.10. No obstante, su diseño se optimiza específicamente para la ejecución intensiva de operaciones matriciales, resultando ideal para aplicaciones que demandan una potencia de cómputo excepcional en el procesamiento de tensores.

Figura 2.10: NVIDIA A100 Tensor Core GPU [32].

Aunque el mercado tecnológico ofrece diversas variantes de estos componentes, su funcionamiento converge en principios operativos similares, que se orientan principalmente en maximizar la eficiencia en el procesamiento de tensores y a reducir la carga computacional sobre la unidad central. De este modo, la selección de estos dispositivos es fundamental para garantizar la viabilidad técnica para el despliegue de modelos de inferencia complejos en entornos que demandan una respuesta en tiempo real.

Capítulo III

Marco metodológico

3.1. Requisitos del sistema

El sistema debe ser un sistema de tipo embebido ya que se opera de manera autónoma y está dedicado a una tarea en específico sin la intervención constante del usuario, en este caso, el reconocimiento de imágenes y movimiento autónomo. Este tipo de sistemas se caracteriza por la integración de hardware y software en una sola plataforma, esto permite mayor estabilidad en su funcionamiento. Asimismo, un sistema embebido busca reducir peso y tamaño del hardware, optimizando así el consumo energético.

Además, se requiere que el sistema sea lo suficientemente potente para procesar información en tiempo real, ya que debe recibir, analizar datos y ejecutar algoritmos de control para el movimiento de los motores que proporcionan desplazamiento a las ruedas del robot. También se debe contar con una potencia de cómputo adecuada que permita la toma de decisiones de control, para que estas se realicen con baja latencia garantizando así movimientos más precisos.

3.2. Arquitectura general

La Figura 3.1 presenta la arquitectura general del sistema, en esta figura se presentan los principales bloques de hardware que conforman al robot móvil. El diagrama permite abstraer la comunicación de los diferentes módulos facilitando la comprensión, y estableciendo una base

sólida para la selección e implementación de los componentes.

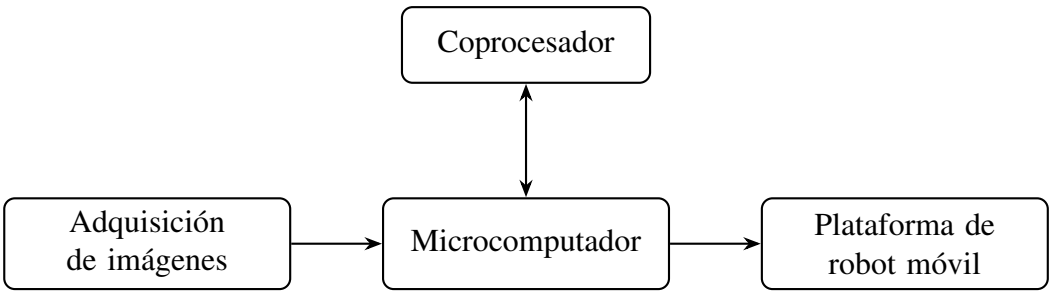


Figura 3.1: Arquitectura funcional conceptual del sistema.

3.3. Adquisición de imágenes

Para la selección del dispositivo encargado de la adquisición de datos se toma en cuenta la Tabla 3.1; luego de considerar los diferentes dispositivos diseñados para el reconocimiento de objetos, se opta por elegir la cámara de video con resolución de 1080p a 45 fps, sin embargo la resolución se puede adaptar en la programación para determinar entre una interacción más fluida o una imagen más nítida. Adicional a estas ventajas la cámara USB es más versátil y no necesita de la instalación de algún tipo de driver para su funcionamiento ya que en Windows y Linux los componentes que vienen por defecto en el sistema hace que este tipo de cámaras sean reconocidas de manera directa.

Tabla 3.1
Comparación de sistemas de visión artificial para robótica móvil

Sistema	**Modelo**	**Tipo de información**	**Costo (USD)**
Cámara USB	Fujikam H812	Imagen RGB	10–20
Cámara CSI	Raspberry Pi Camera Module v2	Imagen RGB	25–50
Cámara de profundidad	Intel RealSense D435	RGB + profundidad (3D)	150–300
Sistema estereoscópico	ZED Stereo Camera	Profundidad por visión dual	200–450
LiDAR	RPLIDAR A1	Distancia (nube de puntos)	100–200
Cámara industrial	Basler ace	Imagen de alta precisión	>300

Nota. Los valores de costo son aproximados y pueden variar según el proveedor y la configuración del sistema.

Estos componentes en conjunto forman un sistema balanceado entre flexibilidad, facilidad de integración, potencia de procesamiento para el reconocimiento de objetos mediante imágenes obtenidas por la cámara USB, sobre todo cumpliendo los requisitos de reconocimiento de objetos, movimiento autónomo y rango de presupuesto económico.

3.4. Microcomputador

Tomando en cuenta la Tabla 3.2 se elige el microcomputador Raspberry Pi 4B, esto debido a su amplia disponibilidad en el mercado local, su alta capacidad de procesamiento gracias a su CPU y memoria RAM, lo que permite ejecutar algoritmos de visión artificial y control en tiempo real. Además, esta plataforma ofrece gran flexibilidad, ya que permite ejecutar un sistema operativo con base en Linux, específicamente Raspberry Pi OS y con la aplicación integrada de Raspberry Pi Connect permite controlar y programar remotamente al dispositivo

desde cualquier lugar.

Tabla 3.2
Comparación de microcomputadores

Plataforma	CPU	RAM	Consumo	Costo (USD)
Raspberry Pi 4 Model B	Cortex-A72(1.5 GHz)	2–8 GB	$\sim$ 5–7 W	50–80
NVIDIA Jetson Nano	Cortex-A57+GPU	4 GB	$\sim$ 5–10 W	100–150
BeagleBone Black	Cortex-A8(1 GHz)	512 MB	$\sim$ 2–3 W	55–70
ODROID-XU4	Exynos 5422(Octa-core)	2 GB	$\sim$ 10 W	60–80

Nota. Los valores de costo y consumo energético son aproximados y pueden variar según el proveedor y la configuración del sistema.

Este microcomputador frente a otras opciones disponibles presenta una relación costo–beneficio favorable, ya que ofrece un rendimiento adecuado a un costo inferior.Asimismo, es compatible con múltiples versiones del lenguaje Python y con una amplia variedad de librerías, lo que la convierte en una plataforma idónea para el desarrollo de sistemas embebidos orientados a visión artificial y control para la plataforma de robot móvil.

Otro aspecto relevante para la selección de la Raspberry Pi 4 Model B es su bajo consumo energético frente a otras plataformas de mayor potencia, esta característica es fundamental en aplicaciones de robótica móvil y sistemas embebidos alimentados por baterías.

3.4.1. Instalación del sistema operativo y parámetros necesarios

Una vez instalado el sistema operativo Raspberry Pi OS se recomienda conectar la Raspberry mediante ¨Raspberry Pi Connect¨ para una mayor portabilidad, ya que esta herramienta permite controlar la Raspberry desde cualquier lugar incluso si la Raspberry y el computador

con el que se inicializan los códigos no se encuentran en la misma red.

La implementación del sistema comienza con la instalación del sistema operativo *Raspberry Pi OS* en la tarjeta microSD del microcomputador. Este sistema operativo, basado en Debian, proporciona compatibilidad nativa con los controladores de hardware, bibliotecas de visión artificial y herramientas de desarrollo necesarias para el reconocimiento de objetos y movimiento de la plataforma de robot móvil.

Una vez instalado el sistema operativo, se recomienda habilitar el acceso remoto mediante *Raspberry Pi Connect*. El uso de esta plataforma facilita la depuración, actualización de código y monitoreo del sistema sin necesidad de conexión física directa.

El uso de un entorno virtual es necesario para evitar conflictos entre versiones de bibliotecas como OpenCV, TensorFlow Lite y PyCoral, las cuales requieren versiones compatibles entre sí.

Adicionalmente, es indispensable instalar el software proporcionado por SunFounder para la plataforma móvil PiCar-X ya que este software incluye los controladores de bajo nivel necesarios para el manejo de los motores DC, el servomotor de dirección y el sensor ultrasónico. La instalación de estas bibliotecas permite abstraer la generación de señales PWM y simplifica la interacción con el hardware.

En la Tabla 3.3 se resumen las principales librerías empleadas en el desarrollo del sistema, indicando su versión y función dentro del flujo de percepción y control.

Tabla 3.3
Librerías utilizadas en el desarrollo del sistema embebido

Librería / Módulo	Versión	Función en el sistema
`opencv-python (cv2)`	4.10.0	Captura de video, conversión BGR–RGB, redimensionamiento y visualización de resultados.
`numpy`	1.26.0	Operaciones matriciales y procesamiento numérico para cálculo de variables de control y filtrado.
`pycoral`	2.0.0	Interfaz para ejecutar modelos optimizados en la Coral Edge TPU.
`tflite-runtime`	2.5.0	Ejecución del modelo TensorFlow Lite y gestión de tensores de entrada y salida.
`sunfounder-picarx`	2.0.4	Control de motores DC, servomotor de dirección y lectura del sensor ultrasónico.
`argparse`	Estándar	Gestión de parámetros de ejecución del script.
`pathlib`	Estándar	Gestión de rutas para carga de modelos y almacenamiento de datos.
`time`, `datetime`	Estándar	Control de temporización y registro temporal de eventos.

3.4.2. Arquitectura del software y flujo de procesamiento

La lógica de control y el procesamiento de visión artificial se implementa en lenguaje Python 3. El código fuente completo se encuentra disponible en el repositorio público de GitHub:

https://github.com/stivenalexandermontero-prog/Robot-Seguidor-de-Objetos-con-Vision-Artificial-Embebida.git

3.4.3. Preprocesamiento y adquisición de video con OpenCV

Este apartado es el encargado de la percepción del entorno y la detección de objetos en tiempo real. Su función principal es adquirir imágenes desde la cámara integrada al sistema

embebido, procesarlas adecuadamente y finalmente ejecutar el modelo de detección de objetos a través de la unidad de procesamiento acelerado o coprocesador.

Para la implementación de ssd_mobilenet_v2_coco_quant_postprocess_edgetpu.tflite es indispensable redimensionar las imágenes de entrada de una resolución de 640x480 píxeles a una resolución de 300 × 300, esto debido a que esta dimensión forma parte explícita de la arquitectura interna de la red neuronal convolucional con la que está entrenada este modelo para aligerar carga de procesamiento y optimiza el tiempo de inferencia.

También, es necesario considerar que la cámara y la biblioteca OpenCV capturan las imágenes en formato BGR (Blue-Green-Red) mientras que el modelo mencionado anteriormente está entrenado con imágenes en formato RGB (Red-Green-Blue), por esto, es necesario adecuar las imágenes antes de enviar los frames al coprocesador.

En la Figura 3.2 se ilustran las etapas del preprocesamiento que se aplica a la imagen de entrada. Se presenta inicialmente la imagen original capturada por la cámara en resolución 640×480 y en formato BGR, posteriormente se muestra la conversión al espacio de color RGB, y finalmente la imagen redimensionada a 300×300 píxeles. Esta imagen final se encuentra lista para ser transformada en tensor y enviada al modelo de detección.

Figura 3.2: Etapas de preprocesamiento de la imagen para el modelo de detección.

En la Figura 3.3 se muestra el flujo de datos específicos en el código, desde la adquisición de datos mediante OpenCV hasta la detección de objetos en el algoritmo una vez analizados los datos arrojados por el TPU.

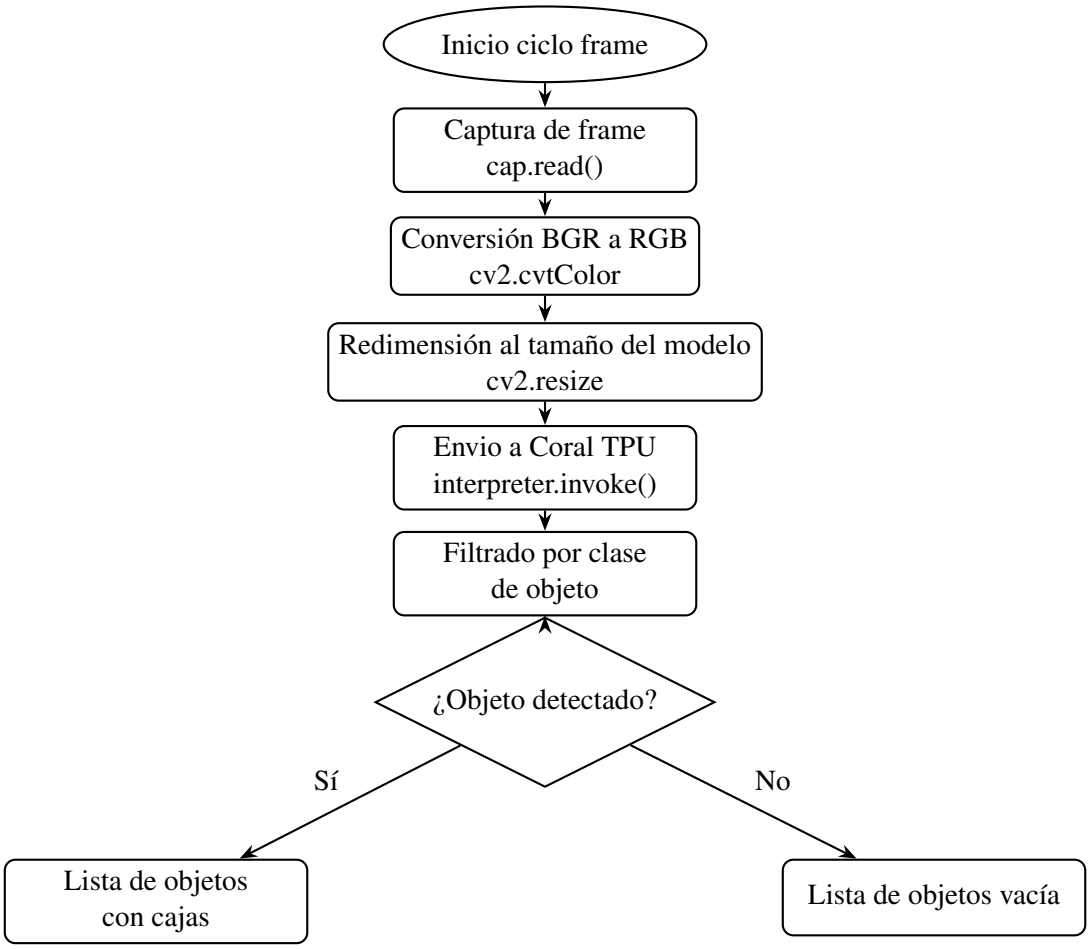


Figura 3.3: Diagrama de flujo de la fase de vision artificial.

3.4.4. Implementación del algoritmo de navegación y control

El diagrama de la Figura 3.4 se detalla el bucle de control de movimiento que se ejecuta al obtener los datos del coprocesador para el seguimiento de objetos.

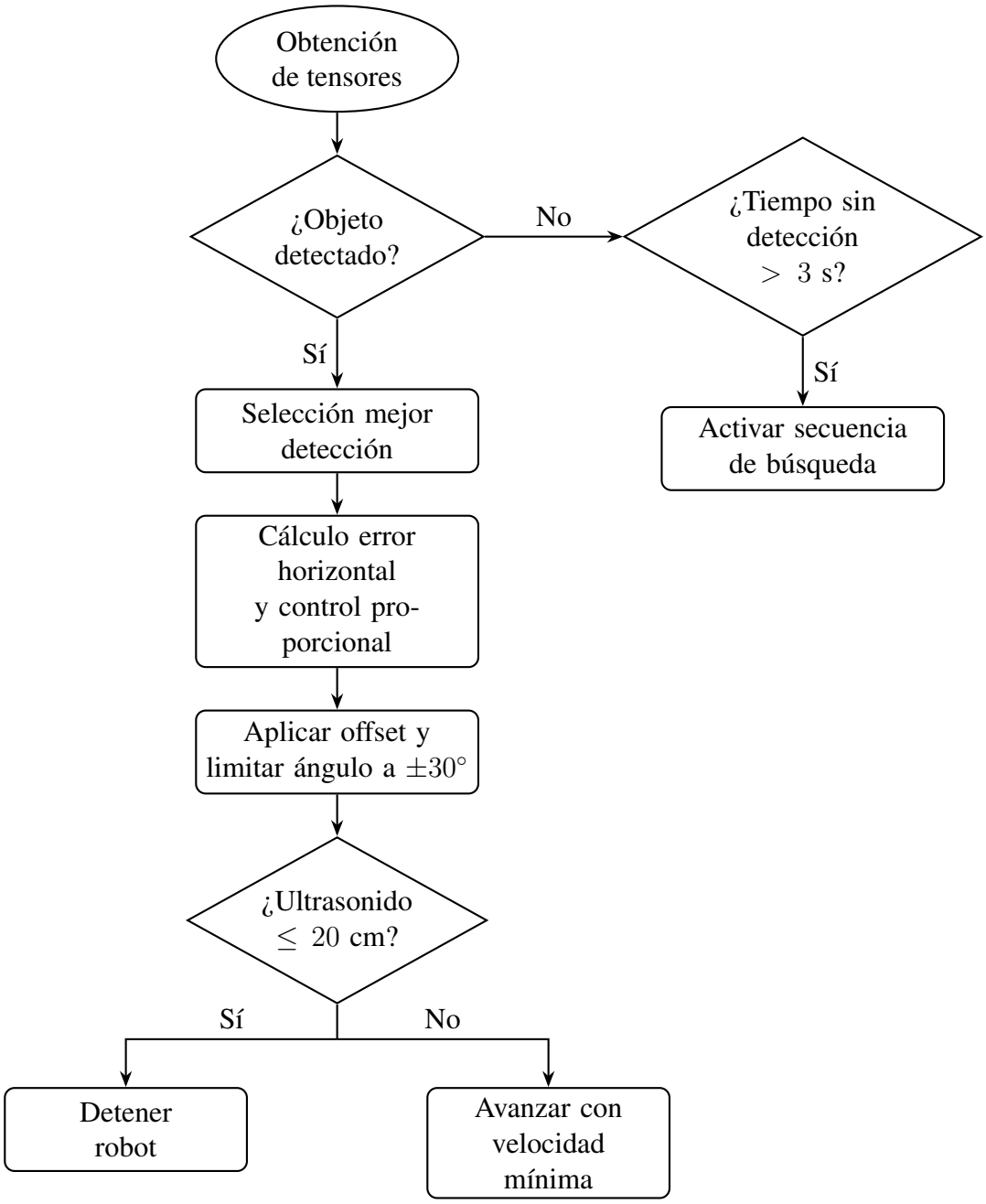


Figura 3.4: Diagrama de flujo para la fase de control del sistema autónomo.

En la Figura 3.5 se ilustra el proceso que realiza el sistema de control previo al movimiento del robot. El diagrama muestra cómo los datos obtenidos a partir del tensor de salida son transformados progresivamente en información útil para la toma de decisiones. El objetivo principal del sistema de control es calcular la posición relativa del objeto en la imagen y generar una señal de control que permite ajustar la orientación del robot.

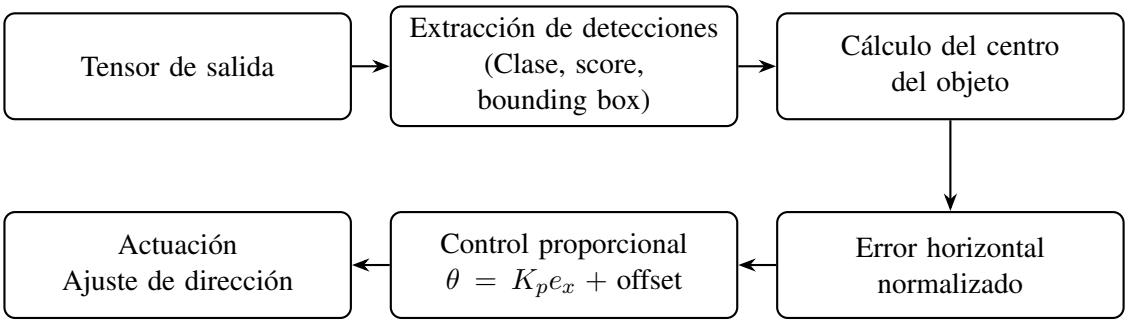


Figura 3.5: Flujo conceptual de conversión de datos desde el tensor de salida hasta la generación de la señal de control.

3.4.5. Conversión de datos en el microcomputador

Una vez que el coprocesador infiere los frames enviados por el sistema de control devuelve datos en forma de tensor que muestran si se detecta el objeto y en qué posición de la imagen se encuentra.

Estos tensores de salida contienen información estructurada del nivel de confianza, el tipo de objeto y la posición del objeto, dicha información es extraída mediante el intérprete de TensorFlow Lite, específicamente a través de la función `detect.get_objects()`.

Las coordenadas son reescaladas a las dimensiones originales del frame capturado por la cámara con el fin de trabajar en el mismo sistema de referencia espacial utilizado por el módulo de control.

A partir de estas coordenadas se calcula el centro horizontal del objeto detectado:

$$c_x = \frac{x_{\text{min}} + x_{\text{max}}}{2} \tag{3.1}$$

Este valor es comparado con el centro de la imagen para obtener el error horizontal normalizado, este error constituye una variable principal de entrada para el controlador proporcional del sistema.

$$e_x = \frac{c_x - w/2}{w/2} \tag{3.2}$$

Donde W corresponde al ancho de los píxeles de la imagen reescalada y e_x corresponde al error

3.4.6. Control proporcional

La implementación de una estrategia de control puramente proporcional (P) se seleccionó debido a la demanda de recursos computacionales y su capacidad para ofrecer una respuesta dinámica inmediata ante las detecciones del sistema de detección de objetos. A diferencia de los esquemas PI o PID, que pueden introducir sobreoscilaciones o requerir un ajuste de parámetros significativamente más complejo, el control P garantiza la agilidad necesaria para un sistema de seguimiento en tiempo real sobre una plataforma embebida. Asimismo, se evitó el uso de la acción derivativa (PD) para eludir la amplificación de ruido en el cálculo del error horizontal. En consecuencia, la robustez y simplicidad de la acción proporcional, configurada con una ganancia $K_p = 28{,}0$ en el código fuente, resulta ser la opción más eficiente para mantener la estabilidad del robot durante la navegación autónoma.

El control proporcional implementado en el sistema permite transformar el error de posición del objeto detectado en una señal angular encargada de ajustar la dirección del robot mediante un servomotor. Este esquema pertenece a un lazo de control cerrado en el cual el sistema de visión actúa como sistema de retroalimentación como se observa en la Figura 3.6.

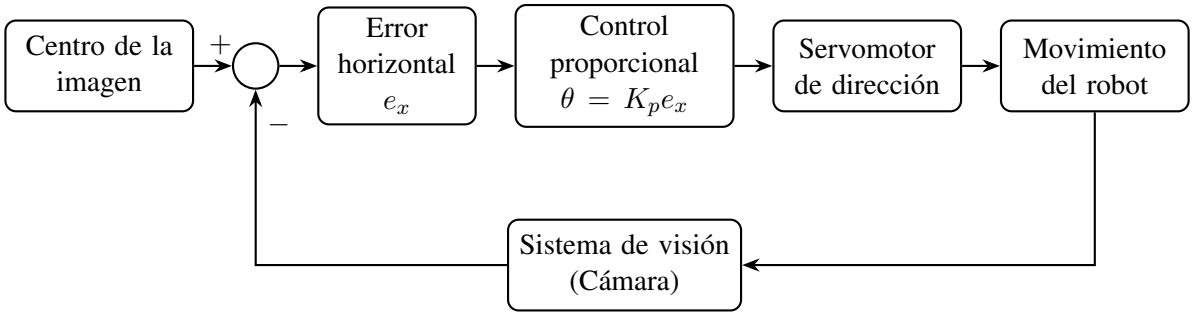


Figura 3.6: Representación del lazo de control proporcional implementado en el robot autónomo.

Matemáticamente, el controlador proporcional puede expresarse como:

$$u(t) = K_p \cdot e(t) \tag{3.3}$$

donde $u(t)$ representa la señal de control generada, K_p es la ganancia proporcional y $e(t)$ es el error del sistema en un tiempo t.

3.5. Coprocesador

Para mejorar el rendimiento del proyecto se optó por usar el TPU de Google llamado Coral USB Accelerator luego de revisar las opciones que se encuentran en la Tabla 3.4, se seleccionó este dispositivo ya que se puede conectar directamente a los puertos USB 3.0 del Raspberry garantizando así que el intercambio de datos entre el TPU y el microcomputador suceda de manera fluida, logrando así que la detección sea más rápida en cuanto a la detección y reconocimiento de imágenes. Este coprocesador responde a la necesidad de integrar capacidades de computación perimetral (*Edge Computing*) en el sistema.Al delegar el procesamiento tensorial a este hardware dedicado, se reduce significativamente la carga computacional sobre los núcleos de la CPU de la Raspberry Pi 4B, permitiendo que el microcomputador gestione de manera concurrente los algoritmos de control de los actuadores y la lectura de los sensores. El coprocesador es capaz de correr modelos optimizados de SSD MobileNet v2 a través del compilador de TensorFlow Lite, que además de ocupar poco espacio en la memoria su eficiencia energética es valiosa en este tipo de sistemas embebidos.

Tabla 3.4
Comparación de coprocesadores para aplicaciones de visión artificial

Plataforma	**Tipo**	**Rendimiento**	**Consumo**	**Costo (USD)**
Google Coral TPU (USB)	TPU (Edge AI)	Hasta 4 TOPS	$\sim$ 2–4 W	60–75
Intel Movidius NCS2	VPU	$\sim$ 1 TOPS	$\sim$ 1–2 W	80–100
NVIDIA Jetson Nano	GPU integrada	$\sim$ 0.5 TOPS	$\sim$ 5–10 W	100–150
CPU (sin aceleración)	Procesamiento general	Bajo (dependiente del CPU)	$\sim$ 5–15 W	–

Nota. Los valores de rendimiento, consumo y costo son aproximados y pueden variar según la configuración del sistema y el proveedor.

El modelo usado para la detección de imágenes en el coprocesador es ssdmobilenetv2 en la variante cocoquantpostprocessedgetpu, se elige la arquitectura SSD MobileNet v2 frente a otros sistemas de detección, como las variantes de YOLO (You Only Look Once) o Faster R-CNN, se fundamenta en el equilibrio óptimo entre precisión y eficiencia computacional para aplicaciones en tiempo real ya que otras arquitecturas requieren una capacidad de procesamiento superior que podría comprometer la latencia en sistemas embebido. Al integrarse con el método de detección

de disparo único (SSD), el algoritmo es capaz de localizar y clasificar objetos en un solo paso, lo que resulta crítico para que el robot móvil mantenga un seguimiento fluido del objetivo sin desfases temporales en el lazo de control.

El sistema implementado utiliza modelos basados en el conjunto de datos COCO (Common Objects in Context), este es un modelo pre-entrenado que proporciona una base de conocimiento robusta para la detección de 80 categorías de objetos comunes. Una ventaja técnica determinante es que estas versiones de SSD MobileNet v2 ya se encuentran sometidas a un proceso de cuantización de enteros de 8 bits (INT8) y han sido compiladas específicamente para la arquitectura de la Edge TPU.

Una vez que el microcomputador ha normalizado la imagen, el flujo de datos en el coprocesador representado en la Figura 3.7 se limita estrictamente a la gestión de tensores y la ejecución del motor de inferencia. Este proceso inicia con la carga del tensor de entrada y culmina con el filtrado lógico de los objetos detectados por el modelo SSD MobileNet v2.

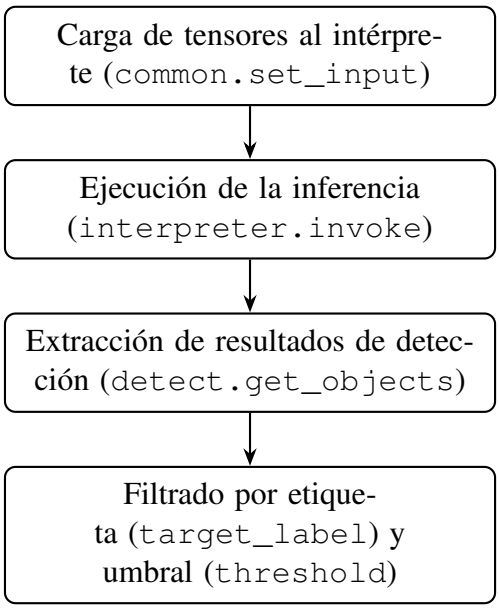


Figura 3.7: Diagrama de flujo lógico para la detección acelerada por hardware.

3.6. Plataforma de robot móvil

Luego de analizar la Tabla 3.5 se selecciona la plataforma de SunFounder PiCar-X Kit ya que ofrece componentes integrados fundamentales para el proyecto como drivers de control,

ruedas, baterías, motores para el movimiento del robot y un sistema de dirección mediante servomotor, también es fundamental señalar la integración con la placa de Raspberry en la base metálica ya que esta diseñado para trabajar directamente con esta placa. En el apartado de software cuenta con calibración de los servomotores, pruebas y flexibilidad en las opciones para el control de sus componentes mediante el microcomputador.

Tabla 3.5
Comparación de plataformas de robot móvil.

Plataforma	**Grado de integración**	**Flexibilidad**	**Costo (USD)**
SunFounder PiCar-X Kit	Alto	Media	90–150
TurtleBot3 (Robotis)	Medio	Media	500–800
Clearpath Husky UGV	Bajo	Baja	$>$10000
Chasis genérico DIY	Bajo	Muy alta	70–150

Nota. La flexibilidad se refiere a la capacidad de modificación e integración de hardware y software.

3.6.1. Construcción de la plataforma móvil

Para el ensamblaje de la plataforma de robot móvil de SunFounder, el fabricante incluye una guía de instalación dentro del kit, la cual describe paso a paso el proceso de montaje mecánico y conexión de los componentes electrónicos, esta guía se muestra en Figura 3.8.

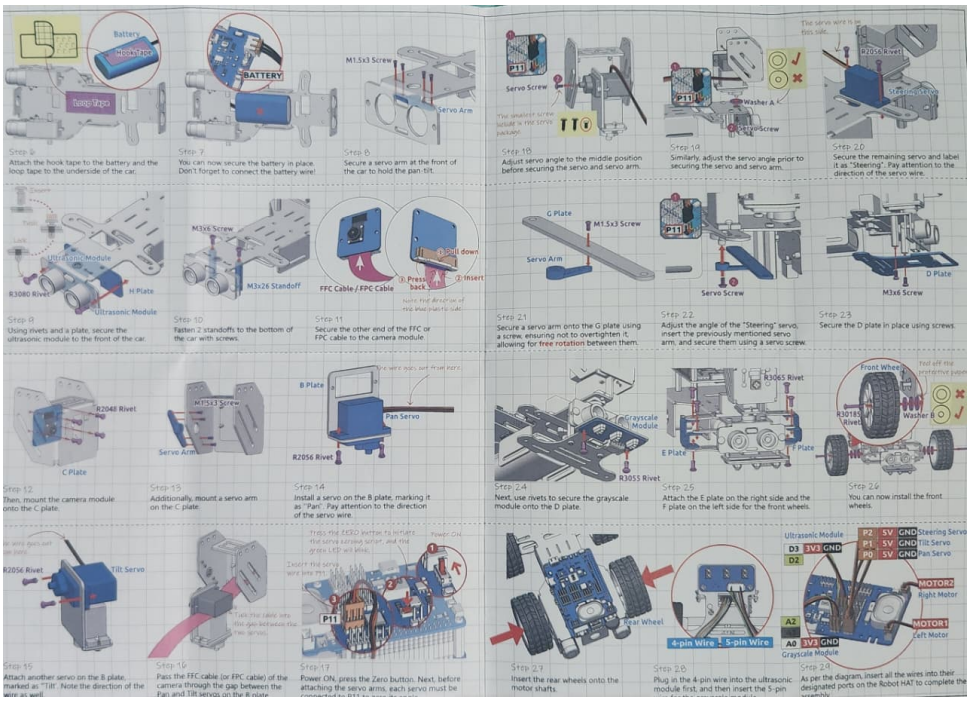


Figura 3.8: Hoja detallada del ensamblaje de PiCar-X [33].

En caso de requerir mayor nivel de detalle, el fabricante proporciona material audiovisual complementario disponible en su sitio web oficial, donde se presenta un video explicativo del proceso de ensamblaje de la plataforma móvil [34].

Una vez construida la plataforma de robot móvil se agrega la cámara y el coprocesador para integrar el sistema de visión al robot como se muestra en la Figura 3.9

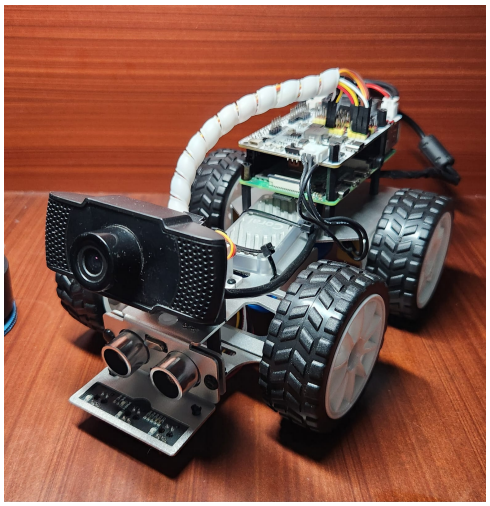


Figura 3.9: Plataforma integrada con visión artificial [33].

3.6.2. Movimiento del robot

Después de que el controlador proporcional genera el ángulo de corrección θ, este ángulo se envía directamente al servomotor de dirección mediante una señal de modulación por ancho de pulso (PWM), la cual permite posicionar el eje del servo dentro del rango mecánico establecido previamente en la configuración del servomotor cuyo rango físico es $\pm 30°$, valores superiores a este rango hacen que la rueda no gire correctamente por lo que no puede seguir el objeto de manera óptima tal como se muestra en la Figura 3.10.

Figura 3.10: Ruedas con rango máximo.

Durante el seguimiento del objeto, se aplica a los motores una velocidad mínima constante previamente programada que garantiza estabilidad en el desplazamiento, en caso de detectarse un obstáculo mediante el sensor ultrasónico, el sistema interrumpe inmediatamente la señal de avance, deteniendo el robot como medida de seguridad; no obstante, se mantiene detectando el objeto programado en todo momento por si se puede reanudar el seguimiento.

En la Figura 3.11 se muestra el diagrama de flujo para el movimiento del robot, desde el algoritmo de control procesado en la Raspberry hasta el movimiento del robot y la implicación de los actuadores en el mismo.

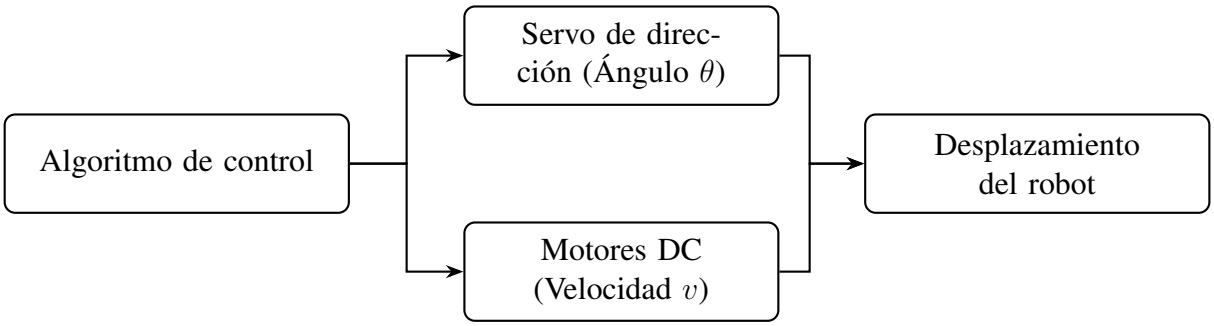


Figura 3.11: Flujo de ejecución de movimiento del robot.

Capítulo IV

Resultados obtenidos

En este apartado se presentan los resultados de la experimentación del sistema autónomo de seguimiento de objetos basado en visión artificial usando Coral TPU para la detección de objetos. Para esto, se describen las pruebas experimentales realizadas para evaluar el desempeño del sistema en la detección de objetos, el procesamiento en tiempo real y la generación de señales de control para el movimiento del robot móvil. También se evalúan métricas cuantitativas relacionadas tanto con la etapa de visión como con la etapa de control, entre estas métricas se encuentra el tiempo de inferencia en el TPU y el error de seguimiento que corresponde a la discrepancia espacial entre la posición actual del objetivo detectado y la posición deseada, estas métricas permiten evaluar la estabilidad y eficiencia del controlador proporcional implementado para el control del movimiento de la dirección.

4.0.1. Pruebas experimentales y visualización de la detección de objetos

En los experimentos realizados se modifican diversas variables del entorno con el objetivo de analizar la robustez del algoritmo de detección y la respuesta del controlador, entre las variables consideradas están las condiciones de iluminación, el tipo de escenario y el tipo de objetos a detectar permitiendo comparar el comportamiento del sistema frente a distintas situaciones y estímulos visuales.

Durante las pruebas se registran datos asociados a la detección por frame, el tipo de objeto, el tiempo de procesamiento, las coordenadas del objeto detectado y el ángulo de las ruedas

del robot. Esta información permite evaluar su desempeño frente a perturbaciones externas y analizar la precisión del reconocimiento, la estabilidad del seguimiento y la capacidad del robot para mantener el objeto dentro del campo visual.

En la Figura 4.1 se puede observar el reconocimiento del objeto una vez detectado por el TPU y postprocesado por el microcomputador para mostrar la imagen. En esta etapa, los resultados de inferencia son reescalados y representados gráficamente sobre el frame original.

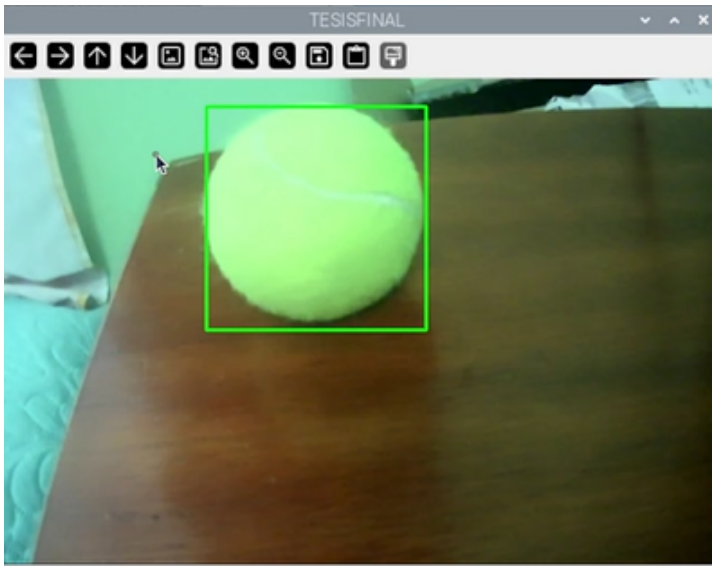


Figura 4.1: Visualización del reconocimiento de objetos.

4.0.2. Detección por tipo de objeto

Con el objetivo de determinar la eficiencia del modelo de visión, las evaluaciones experimentales se centran en la capacidad de reconocimiento de las clases ¨persona¨ y ¨balón deportivo¨. Para validar la precisión del sistema se utilizan objetos que sí corresponden a la clase como elementos distractores de morfología similar, esto con el objetivo de analizar la capacidad del modelo para discriminar correctamente entre instancias válidas y posibles falsos positivos.

En la Figura 4.2 se muestra una línea de tiempo que organiza las detecciones por frames para la categoría de ¨balón deportivo¨, la gráfica permite visualizar las instancias donde se detecta el objeto y a que tipo de objeto pertenece.

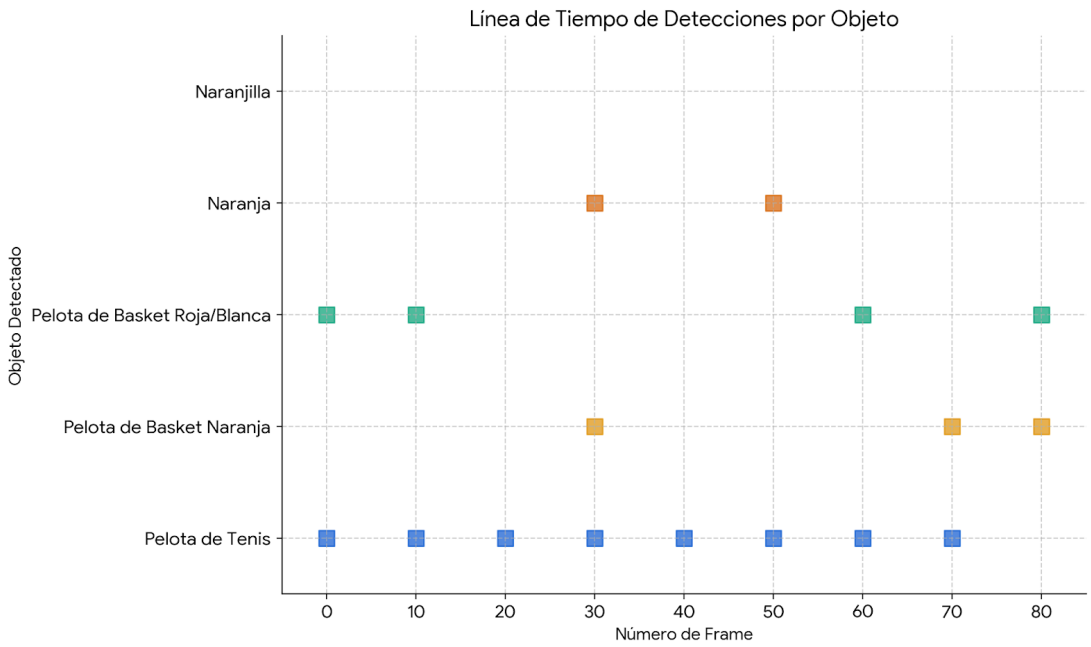
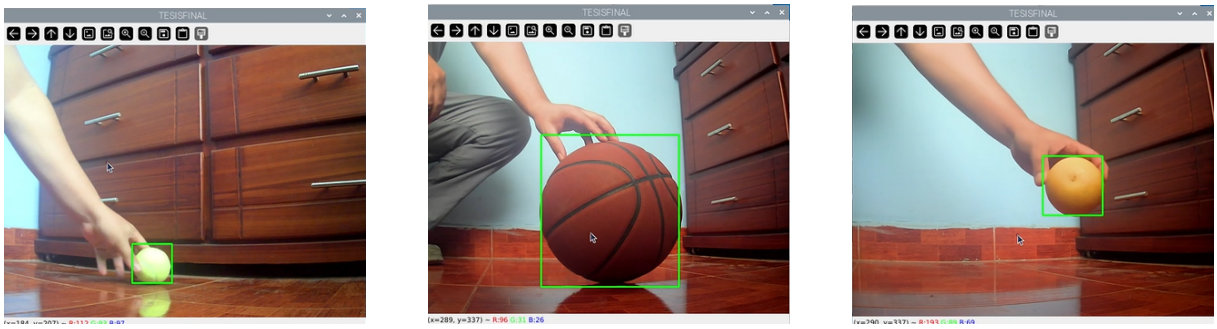


Figura 4.2: Resultado de las pruebas realizadas con diferentes objetos en la clase ¨balón deportivo¨.

En la Figura 4.3 se exponen los escenarios más representativos derivados del análisis previo. En la izquierda se observa la pelota de tenis, objeto con el cual el sistema demuestra mayor robustez respecto a la detección. En la sección central se presenta el balón naranja de básquetbol, este objeto es categorizado como falso negativo; esto se atribuye a que, pese a ser una instancia válida, las condiciones de iluminación deficiente y el mimetismo con el fondo impiden una detección correcta. Finalmente, en el lado derecho se encuentra una naranja, esta fruta se considera falso positivo intermitente; debido a su morfología esférica y color, el modelo la identifica erróneamente como balón deportivo en determinados frames, similar al balón de baloncesto.



(a) Pelota de tenis. (b) Balón de baloncesto. (c) Naranja (fruta).

Figura 4.3: Objetos relevantes en la detección de balón deportivo.

Como se ilustra en la Figura 4.4 se analiza el desempeño del algoritmo para la categoría persona en un escenario exterior. Las pruebas integran variaciones en la postura de la persona, en reposo y movimiento. En ambos escenarios, el modelo muestra un comportamiento robusto, detectando de manera acertada la mayor parte de frames a la persona.

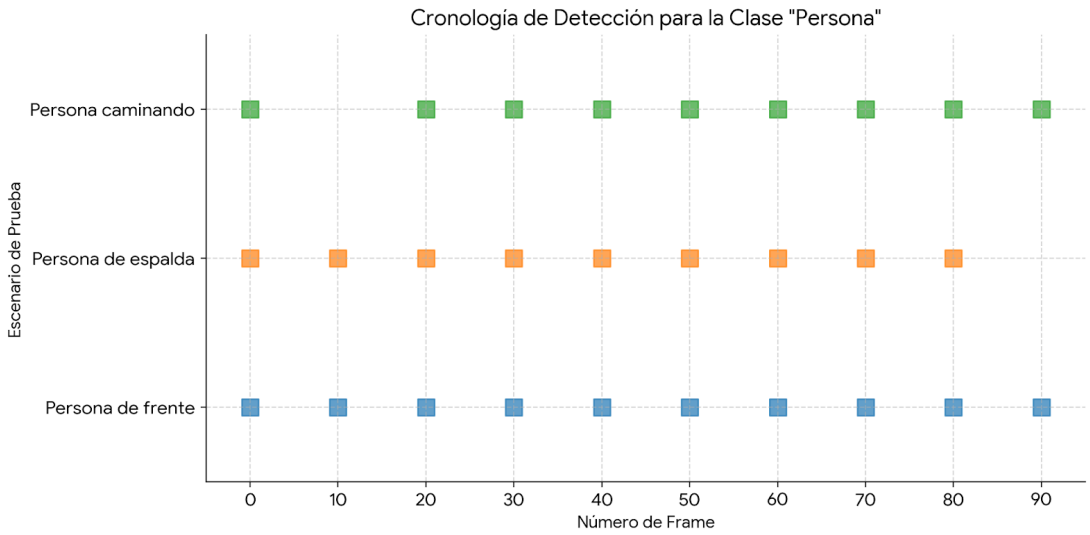


Figura 4.4: Clase persona con diferentes posturas y movimiento.

4.0.3. Influencia de las condiciones de iluminación

En la Tabla 4.1 se detallan los resultados experimentales obtenidos tras someter el modelo de visión a distintos escenarios lumínicos.

Tabla 4.1
Desempeño del modelo de visión bajo distintas condiciones lumínicas.

Condición de luz	**Tasa de detección (%)**	**Confianza $\bar{x}$**
Luz natural (exterior)	90 %	0.74
Luz artificial (interior)	50.0 %	0.88
Baja luminosidad	10 %	0.6

En el escenario de luz natural exterior se evidencia una mayor tasa de detección del objeto pero una menor confianza, este resultado puede ser causado por la cantidad de luz que puede provocar sombras y pérdida de textura, lo que puede provocar incertidumbre en el sistema de reconocimiento.

Por otro lado, en el experimento realizado con luz artificial la tasa de detección es menor, pero la confianza del modelo es mayor. Esta discrepancia sugiere que, bajo luz artificial, el modelo es más selectivo; aunque pierde el rastro del objetivo con mayor frecuencia debido a una posible falta de intensidad o alcance del sensor.

Los resultados de baja luminosidad confirman que el rendimiento se degrada de forma crítica. Esta pérdida de eficiencia es consecuencia directa del ruido digital generado en el sensor de la cámara por la falta de luz, resultando casi imposible detectar el objeto en esas condiciones.

Finalmente, es pertinente destacar que durante la experimentación se nota una variación considerable en el rango operativo del sistema bajo distintas fuentes lumínicas. El alcance efectivo de detección en exteriores bajo luz natural es aproximadamente tres veces superior al registrado en entornos de luz artificial. Este incremento se puede adjudicar a la uniformidad en la luz que se presenta en exteriores y la ventaja del sensor ante mayor cantidad de luz para detectar objetos con mayor claridad a pesar de la distancia.

4.0.4. Rendimiento computacional y eficiencia del control

La Tabla 4.2 se muestra la experimentación de la clase persona con movimiento, en sus primeros frames para un análisis enfocado en la inferencia y el error normalizado.

Tabla 4.2
Resultados de rendimiento computacional y respuesta del sistema de control (Frames 0-50).

Frame ID	Tiempo Inferencia (ms)	Error Normalizado (e_x)	Ángulo Dirección (°)
0	43.42	0.0219	0.00
10	20.78	0.0000	0.00
20	27.89	-0.0781	0.00
30	20.78	-0.3109	-3.71
40	22.28	-0.3578	-5.02
50	26.75	-0.2953	-3.27

El análisis de los datos presentados en la Tabla 4.2 permite evaluar la sinergia entre el procesamiento del sistema de visión artificial y la respuesta mecánica del sistema de control mediante la dirección de las llantas del robot. A continuación, se detallan los hallazgos más relevantes.

El tiempo de inferencia superior en el primer frame es característico de un proceso de carga del modelo desde el microcomputador al TPU, una vez inicializado el modelo los tiempos de inferencia se encuentran entre 20.78 ms y 27.89 ms. Esta velocidad de ejecución garantiza una baja latencia, asegurando así, que la señal de control enviada a los servomotores se base en la posición actual del objetivo y no en una coordenada desplazada por el retraso en el procesamiento de datos.

En los datos obtenidos también se observa una zona muerta en el ángulo de dirección, ya que en el frame 20 a pesar de tener un error normalizado de -0.0781, el ángulo de la dirección de las ruedas no cambia sino hasta que el error normalizado se vuelve mayor como en el frame siguiente. Esta configuración es intencional y demuestra la eficiencia del algoritmo para ignorar errores mínimos o ruido visual, evitando vibraciones innecesarias. Luego de esto, en los siguientes frames cuando el error normalizado es mayor, el sistema de control aumenta el ángulo de dirección para centrar el objeto con la cámara del robot, y así dirigirse a la misma dirección que el objeto. Sin embargo, cuando los objetos salen del campo visual del robot o se desplazan de manera rápida de un lado a otro, el robot es incapaz de corregir rapidamente el ángulo de las ruedas para seguir al objeto por la limitación del rango de ángulo seguro en la dirección.

4.1. Análisis de resultados

La implementación de la arquitectura de visión computacional y control de movimiento en el robot ha permitido consolidar un sistema de seguimiento dinámico capaz de reaccionar a estímulos visuales con una latencia baja, gracias a la implementación del acelerador Coral TPU para la detección de imagenes mediante la inferencia de modelos de aprendizaje profundo (SSD MobileNet v2), y al microcomputador Raspberry para el pre-procesamiento de imágenes, junto con el manejo de datos recibidos por la plataforma móvil para el control de la dirección de las ruedas del robot.

La evaluación de desempeño ha determinado que la frecuencia de actualización de los datos es altamente estable gracias a la descarga del procesamiento de datos hacia la TPU. Por este motivo, los tiempos de inferencia promedio oscilaron en los 25 ms.

Respecto al análisis del sistema de control, se evidencia una estructura de respuesta proporcional eficiente, donde el error horizontal normalizado se vincula directamente con el ángulo de dirección de los servomotores encargados de dar dirección al robot.

La validación del sistema mediante pruebas en entornos interiores y exteriores demuestra que el robot es capaz de operar bajo distintas condiciones lumínicas con un nivel adecuado de precisión, sin embargo se ve limitado al sensor de la cámara con la que opera que en entornos de poca luz es deficiente en la detección de objetos. El alcance efectivo de detección bajo luz natural supera en una proporción de 3:1 al alcance obtenido bajo luz artificial, lo que habilita la posibilidad de extender el sistema hacia aplicaciones en entornos exteriores principalmente dirigido a tareas de seguimiento autónomo y toma de decisiones en tiempo real. Aunque la luz solar reduce ligeramente el score de confianza debido a los contrastes marcados, la mayor profundidad de campo obtenida valida el uso de este prototipo para aplicaciones en diferentes ambientes lumínicos.

Capítulo V

Conclusiones, recomendaciones y trabajo a futuro

5.1. Conclusiones

Se concluye que el desarrollo del robot móvil, mediante la integración de visión artificial embebida, permite obtener un prototipo funcional capaz de realizar el seguimiento autónomo de objetos con altos niveles de estabilidad y precisión. La integración del sistema de control permite traducir las coordenadas visuales obtenidas en tiempo real en comandos de actuación mecánicos, validando la capacidad del robot para responder dinámicamente ante cambios en la trayectoria de los objetivos seleccionados.

En cuanto a la propuesta técnica, se valida que la arquitectura basada en la sinergia entre el microcomputador Raspberry Pi 4B y el acelerador Coral USB Edge TPU constituye una plataforma de alto rendimiento para el procesamiento tensorial. Esta combinación de hardware permite ejecutar modelos de aprendizaje profundo con una latencia mínima, descargando la carga computacional de la CPU central y garantizando la viabilidad de realizar inferencias complejas.

Respecto a la fase de software, se concluye que la implementación del modelo SSD MobileNet v2, junto con un algoritmo de control proporcional, asegura un equilibrio óptimo entre precisión y eficiencia computacional para la identificación y clasificación de categorías de objetos. El uso de este esquema de control permitió transformar el error de posición detectado en

la imagen en una señal angular inmediata para el servomotor de dirección, logrando un seguimiento fluido y evitando vibraciones innecesarias en el desplazamiento del robot.

Finalmente, las pruebas experimentales han demostrado una interacción fluida entre el sistema de control y la plataforma mecánica PiCar-X, confirmando la estabilidad del lazo de retroalimentación visual. Se determinó que factores ambientales como la iluminación son críticos para el desempeño, concluyendo que el alcance efectivo de detección bajo luz natural exterior es tres veces superior al obtenido bajo luz artificial interior.

5.2. Recomendaciones

Para asegurar el éxito del seguimiento autónomo y la estabilidad global del prototipo, se recomienda establecer un entorno de operación donde los objetos de interés posean un contraste cromático significativo respecto al fondo de la escena. Esta diferenciación de colores es fundamental para maximizar la precisión del sistema de visión artificial, permitiendo que el robot mantenga el rastro de manera constante y reduciendo la probabilidad de errores en la identificación de objetivos durante la navegación.

Respecto a la arquitectura técnica conformada por el microcomputador Raspberry Pi 4B y el acelerador Coral TPU, se recomienda integrar sistemas de ventilación activa en escenarios de funcionamiento ininterrumpido. Esta medida resulta fundamental para mitigar el sobrecalentamiento de los módulos, lo cual previene el estrangulamiento térmico y evita retardos en la adquisición o procesamiento de datos, garantizando así la estabilidad operativa

Para optimizar la fase de software y ampliar las capacidades de clasificación más allá de las categorías estándar, se sugiere el entrenamiento de modelos de detección personalizados cuando se requiera identificar objetos específicos no contemplados en el modelo SSD MobileNet v2 original. Es imperativo que dichos modelos sean convertidos al formato TensorFlow Lite para asegurar la total compatibilidad con el hardware de procesamiento tensorial, logrando una ejecución ligera y eficiente que no comprometa la latencia del algoritmo de control.

Se recomienda establecer un protocolo de calibración periódica para la alineación del eje óptico de la cámara respecto al centro geométrico del sistema de dirección. Esta medida permite corregir pequeñas desviaciones mecánicas o desajustes provocados por las vibraciones durante el desplazamiento, asegurando que el cálculo del error horizontal normalizado mantenga una

correspondencia exacta con la trayectoria física del robot y evitando derivas innecesarias en el seguimiento del objetivo.

5.3. Trabajo a futuro

A partir de los resultados obtenidos, se identifican diversas líneas de mejora que podrían potenciar significativamente el desempeño operativo del prototipo en futuras investigaciones.

Primero, la implementación de un sistema de iluminación activa sujeta al chasis del robot para que no tenga problemas de detección de objetos en condiciones de iluminación deficientes, esto permitiría estandarizar la entrada visual del sensor, garantizando una detección óptima. De igual manera, se propone la evolución hacia una plataforma móvil con mayores grados de libertad, sustituyendo la dirección actual por una dirección que permita eliminar las restricciones en el radio de giro y de esta manera evitar que el objeto se pierda o salga del campo de visión del robot, permitiendo así un seguimiento más fluido del objeto. Otra implementación adicional, sería la adopción de un procesador de mayor capacidad computacional, junto con otro sensor para visión artificial, el procesador ayudará con el manejo de datos de ambos sensores; el nuevo sensor permitirá detectar objetos con mayor confiabilidad en diversos tipos de entornos y trabajar en conjunto para llegar a un mejor resultado en la detección de objetos.

Bibliografía

[1] M. Mónica, "El número de robots industriales se ha triplicado en la última década," *statista*, 8 2022.

[2] M. Intelligence, "Industria robótica - análisis de tamaño y participación - tendencias y pronósticos de crecimiento (2023 - 2028)," 2023.

[3] G. Lizarzaburo, "La robótica en la industria está en franco crecimiento: cinco consejos para ecuador," *Expreso*, 8 2020.

[4] F. GROUP, "Soluciones de ingenierÍa, automatizaciÓn industrial seguridad de mÁquinas y robÓtica en ecuador," 6 2021.

[5] D. Santos, L. Dallos, and P. A. Gaona-García, "Algoritmos de rastreo de movimiento utilizando técnicas de inteligencia artificial y machine learning," *Información tecnológica*, vol. 31, pp. 23–38, 6 2020.

[6] B. Tom and M. Dustin, "http://dspace.ups.edu.ec/handle/123456789/24933," 2023.

[7] C. Jorge, "Control de un robot seguidor de línea tipo diferencial mediante visión artificial," 8 2018.

[8] M. M. Miranda-Ramos, J. O. Hallon, and J. D. Suriaga, "Sistema de cámaras para la detección de mascarillas con python y raspberry pi con comunicación inalámbrica a través del sistema global de comunicación móvil (gsm)," *Información tecnológica*, vol. 33, pp. 23–30, 6 2022.

[9] S. Jorge, "Evaluación de algoritmos de detección de objetos basados en deep learning para detección de incidencias en carreteras," 9 2020.

[10] A. Saha, B. C. Dhara, S. Umer, K. Yurii, J. M. Alanazi, and A. A. AlZubi, "Efficient obstacle detection and tracking using rgb-d sensor data in dynamic environments for robotic applications," *Sensors*, vol. 22, no. 17, 2022.

[11] C. Bryan, "Seguimiento y evaluaciÓn de personas en ambientes cerrados / abiertos," 2021.

[12] B. L. Coelho, P. R. Bodmann, N. Cavagnero, C. Frost, and P. Rech, "Vision transformer reliability evaluation on the coral edge tpu," *IEEE Transactions on Nuclear Science*, vol. 72, no. 4, pp. 1443–1452, 2025.

[13] C. Vargas-Torres and J. Santiago-Paz, "Robot seguidor de línea basado en visión artificial con ros y opencv," *Revista de Aplicaciones de la Ingeniería*, vol. 6, pp. 1–10, Marzo 2019.

[14] V. Alvear-Puertas, P. Rosero-Montalvo, D. Peluffo-Ordóñez, J. Pijal-Rojas, V. Alvear-Puertas, P. Rosero-Montalvo, D. Peluffo-Ordóñez, and J. Pijal-Rojas, "Internet de las cosas y visión artificial, funcionamiento y aplicaciones: Revisión de literatura," *Enfoque UTE*, vol. 8, pp. 244–256, 2 2017.

[15] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2nd ed., 2011.

[16] LUCID Vision Labs, "Understanding digital image sensors," 2024.

[17] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB*. Springer, 2nd ed., 2017.

[18] T.-Y. Wu, "Time-of-flight system design—part 2: Optical design for time-of-flight depth sensing cameras," *Analog Dialogue*, vol. 55, July 2021.

[19] A. S. Mathavan, "A comprehensive study on image preprocessing techniques for enhanced image analysis," 11 2025.

[20] W. Duckerman, "Procesamiento de imágenes digitales con MATLAB," ene 2022. Figura: Representación matricial de una imagen digital.

[21] M. Arias-Montiel, *Prototipo virtual de un robot móvil multi-terreno para aplicaciones de búsqueda y rescate*, ch. 29, pp. 337–351. Asociación Mexicana de Mecatrónica A.C., 2016.

[22] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*. Waltham, MA: Morgan Kaufmann, 5th ed., 2021.

[23] J. W. Valvano, *Embedded Systems: Real-Time Interfacing to Arm Cortex M Microcontrollers*. CreateSpace Independent Publishing Platform, 5th ed., 2019.

[24] J. Osio, W. Aróztegui, and J. Rapallini, *Sistemas digitales basados en microcontroladores: Descripción, funcionalidades y aplicaciones de los microcontroladores basadas en el CPU HCS08*. 01 2020.

[25] G. E. Rodriguez, "Diseño y desarrollo de un prototipo de riego automático controlado con raspberry pi y arduino," 2 2015.

[26] Raspberry Pi, "Buy a raspberry pi 4 model b."

[27] BeagleBoard.org, "Beaglebone® black - beagleboard."

[28] NVIDIA Corporation, "Desarrollo de productos de Jetson Nano," 2026. [Accedido: 3 de mayo de 2026].

[29] Y. Sun and A. M. Kist, "Deep learning on edge tpus: A survey," *arXiv preprint arXiv:2108.13732*, oct 2022.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[31] Coral, "Usb accelerator | coral."

[32] NVIDIA, "Nvidia a100 | nvidia."

[33] SunFounder, "Documentación oficial picar-x v2.0." http://picar-x-v20.rtfd.io/, 2024.

[34] SunFounder, "Documentación oficial picar-x v2.0." https://docs.sunfounder.com/projects/picar-x-v20/es/latest/, 2024.

Anexos