UNIVERSIDAD PENÍNSULA DE SANTA ELENA

FACSISTEL

INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

Desarrollo de un sistema de supervisión sobre un robot móvil terrestre para búsqueda de personas con visión artificial

Roger Alexander Carbo Valenzuela

Dirigido por:
Ing. Junior Figueroa Olmedo, Mgtr.

La Libertad - 2025

DEDICATORIA

Dedico este proyecto de titulación a mi madre, gracias a ella, he aprendido a ser una persona fuerte, cualidades que me han formado para llegar hasta este punto, donde concluyo una etapa tan importante como lo son mis estudios universitarios, para dar paso a mi independencia y al inicio de mi carrera profesional. Mi mayor deseo es poder devolverte algún día todo el amor y la comprensión que me has brindado a lo largo de mi vida, sabiendo que seguirás acompañándome y apoyándome durante mucho tiempo.

A mis abuelos, quienes me guiaron por un buen camino, siempre con su alegría y amor incondicional. Su apoyo y orientación han sido esenciales para mi crecimiento personal y académico, y siempre estarán en mi corazón como los pilares que me ayudaron a construir el ser humano que soy hoy.

Roger Alexander Carbo Valenzuela

AGRADECIMIENTO

Agradezco en primer lugar a Dios, por darme la sabiduría necesaria para tomar decisiones correctas a lo largo de mi vida universitaria, lo que me ha permitido llegar hasta este día tan importante para mí y culminar exitosamente mis estudios y este proyecto de investigación.

A la Universidad Estatal Península de Santa Elena, por brindarme los conocimientos y la formación necesarios para alcanzar esta meta, dejándome con un profundo sentido de gratitud hacia tan distinguida institución.

Agradezco a mi madre, Mariuxi Carbo por su apoyo incondicional, por enseñarme principios y valores fundamentales en mi vida, y por cuidarme día a día, esforzándose siempre para brindarme un mejor futuro.

A mis abuelos, Odalia Valenzuela y Luis Carbo por cuidarme, brindarme su cariño y apoyo en todo momento, por su dedicación y por siempre estar a mi lado en cada etapa de mi vida dándome la fortaleza necesaria para seguir adelante.

A mi hermano, Dylan Reyes por estar presente para mí en los buenos y malos momentos y brindarme su apoyo incondicional en cada situación.

A mi tutor, Ing. Junior Figueroa, por su paciencia, dedicación y orientación a lo largo de todo mi proceso de titulación. Su apoyo y experiencia han sido fundamentales para mi crecimiento académico.

Mis más sinceros agradecimientos

Roger Alexander Carbo Valenzuela

APROBACIÓN DEL TUTOR

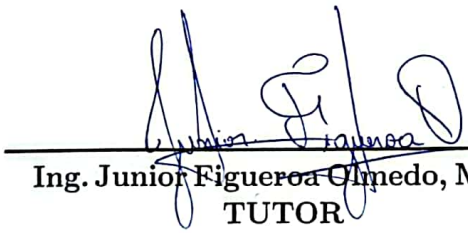
En mi calidad de tutor del trabajo de titulación denominado: **"Desarrollo de un sistema de supervisión sobre un robot móvil terrestre para búsqueda de personas con visión artificial"**, elaborado por el estudiante **Roger Alexander Carbo Valenzuela**, de la carrera de **Ingeniería en Electrónica y Automatización** de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

Ing. Junior Figueroa Olmedo, Mgtr.
Tutor

TRIBUNAL DE GRADO

Ing. Ronald Rovira Jurado, Ph.D.
DIRECTOR DE CARRERA

Ing. Luis Chuquimarca Jiménez, Mgtr.
DOCENTE GUÍA UIC
DOCENTE ESPECIALISTA

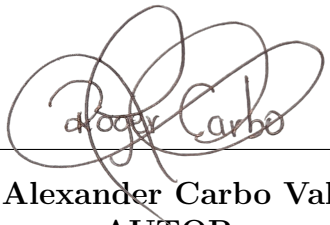


Ing. Junior Figueroa Olmedo, Mgtr.
TUTOR

Ing. Corina Gonzabay De la A, Mgtr.
SECRETARIA DEL TRIBUNAL

DECLARACIÓN

El contenido del presente trabajo de titulación es de mi entera responsabilidad, el patrimonio intelectual del mismo le pertenece a la Universidad Estatal Península de Santa Elena.



Roger Alexander Carbo Valenzuela
AUTOR

Resumen

El presente proyecto consiste en el desarrollo de un sistema de supervisión implementado sobre un robot móvil terrestre equipado con tecnología de visión artificial, diseñado para tareas de búsqueda y rescate en entornos complejos. El objetivo principal es asistir a los rescatistas en la identificación temprana de personas atrapadas o en peligro, reduciendo su exposición a zonas de alto riesgo y optimizando la toma de decisiones durante emergencias. El robot cuenta con una cámara de alta resolución y sensores que capturan imágenes del entorno en tiempo real, las cuales son procesadas mediante algoritmos de aprendizaje automático entrenados con redes neuronales convolucionales. El sistema permite clasificar entre personas y obstáculos, y opera de forma autónoma mediante navegación asistida por GPS y sensores ultrasónicos. Los resultados del proyecto muestran que el modelo Yolov5 alcanzó una precisión del 95 % en la detección de las distintas clases, con un tiempo promedio de respuesta de 2 segundos por imagen procesada. Estas métricas validan la eficacia del sistema en escenarios con escombros. La interfaz de usuario facilita el monitoreo remoto y el control del robot tanto en modo manual como automático. Este enfoque ofrece beneficios significativos como la detección oportuna de víctimas, el ahorro de recursos al priorizar áreas críticas y la reducción del riesgo humano. Este enfoque innovador busca optimizar recursos, permitir la detección temprana de víctimas y mejorar la eficacia de las operaciones, siendo escalable y adaptable a distintos tipos de emergencias.

Palabras clave: Robot móvil, visión artificial, personas, monitoreo, detección.

Abstract

The present project consists of the development of a monitoring system implemented on a land mobile robot equipped with artificial vision technology, designed for search and rescue tasks in complex environments. The main objective is to assist rescuers in the early identification of trapped or endangered people, reducing their exposure to high-risk areas and optimizing decision making during emergencies. The robot has a high-resolution camera and sensors that capture images of the environment in real time, which are processed by machine learning algorithms trained with convolutional neural networks. The system can classify between people and obstacles, and operates autonomously using GPS-assisted navigation and ultrasonic sensors. The project results show that the Yolov5 model achieved an accuracy of 95 % in detecting the different classes, with an average response time of 2 seconds per processed image. These metrics validate the effectiveness of the system in debris scenarios. The user interface facilitates remote monitoring and control of the robot in both manual and automatic modes. This approach offers significant benefits such as timely detection of victims, resource savings by prioritizing critical areas, and reduction of human risk. This innovative approach seeks to optimize resources, enable early detection of victims and improve the efficiency of operations, while being scalable and adaptable to different types of emergencies.

Keywords: Mobile robot, artificial vision, people, monitoring, detection.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En situaciones de emergencia, como desastres naturales u otros eventos que impliquen operaciones de rescate, el ámbito de búsqueda y rescate enfrenta constantes desafíos, para optimizar recursos y minimizar los riesgos asociados con la localización y recuperación de víctimas. La detección temprana y precisa de personas en peligro es crucial para tomar acciones inmediatas que puedan salvar vidas y mejorar la eficiencia de las operaciones. En este contexto, la visión artificial emerge como una solución eficaz para la captura y análisis de imágenes, permitiendo identificar víctimas y obstáculos con rapidez y precisión.

El robot móvil propuesto se desplazará de manera manual o automática por zonas afectadas, capturando imágenes en tiempo real del entorno. La implementación de algoritmos avanzados de visión artificial permitirá clasificar los elementos detectados, diferenciando entre personas, escombros y otros objetos relevantes. Esta capacidad de análisis inmediato no solo agilizará el proceso de búsqueda, sino que también facilitará la toma de decisiones informadas para priorizar las áreas de rescate más críticas. La precisión y la rapidez son factores esenciales que garantizan el éxito de las misiones de búsqueda y rescate. El uso de tecnologías avanzadas, como la visión artificial, desempeña un papel fundamental en la mejora de estos procesos, ofreciendo soluciones innovadoras para identificar y localizar personas en situaciones de emergencia.

Este proyecto se centra en el despliegue de un robot móvil autónomo equipado con tecnología de visión artificial, diseñado para operar eficazmente en entornos complejos y de difícil acceso.

1.1. Justificación

Hoy en día se han desarrollado dispositivos electrónicos que, cuando se integran en un dispositivo de cómputo, pueden lograr objetivos específicos y llegar a lugares a los que las personas no pueden llegar físicamente o tienen dificultades para hacerlo. En la actualidad, los robots de búsqueda y rescate están experimentando un gran crecimiento.

La robótica y la tecnología que los sustentan continúan avanzando, lo que permite el desarrollo de robots cada vez más avanzados y sofisticados que pueden ayudar a los equipos de rescate en situaciones de emergencia. Estos robots pueden ser utilizados para buscar y localizar personas desaparecidas,

transportar suministros y herramientas, realizar inspecciones y evaluar la seguridad de las estructuras, entre otras cosas. Además, se espera que la demanda de robots de búsqueda y rescate siga creciendo a medida que aumenta la frecuencia y la intensidad de las catástrofes naturales y otros eventos que requieren una respuesta rápida y efectiva.

Debido a los desastres naturales que ocurren a nivel nacional y mundial en los últimos tiempos, específicamente sucesos que han dejado una gran cantidad de perdidas como lo son los sismos de gran magnitud, la población en muchos casos, no se encuentra debidamente preparada para dichos eventos. A esto se le suma el poco o limitado apoyo correspondiente de parte del Estado hacia la población, debido a que no se cuenta con un respectivo plan de evacuación ante un sismo. En consecuencia, por la falta de información y capacitaciones, podría existir personas atrapadas como resultado ante la presencia de un desastre de gran magnitud, algunas de ellas con recursos limitados o con poco tiempo para la sobrevivencia frente a esta situación [1].

Las personas no se encuentran preparadas para sismos de alta intensidad, y esto crea pánico, lo que lleva a más desesperación y más pérdida de vidas, como el terremoto de magnitud 7.8 que azotó el noroeste de Ecuador frente a las costas de nuestro país el 16 de abril del 2016. Hasta el 21 de abril de ese año, un total de 587 personas murieron, 155 desaparecieron y 7015 resultaron heridas, 1.125 edificios fueron destruidos y más de 829 edificios resultaron dañados, incluidas 281 escuelas [2].

Como apoyo a situaciones de este tipo o similares se plantea el desarrollo de un sistema de supervisión sobre un robot móvil terrestre para detectar y localizar personas atrapadas en áreas de difícil acceso o en condiciones peligrosas, como edificios derrumbados, zonas de desastre o terrenos accidentados. Esto puede acelerar considerablemente el proceso de rescate y aumentar las posibilidades de supervivencia de las víctimas, así se puede acceder a áreas peligrosas o inaccesibles para los humanos. En situaciones de emergencia, puede haber lugares donde el acceso humano es extremadamente peligroso debido a condiciones adversas, como incendios, gases tóxicos o estructuras inestables y en donde un robot de este tipo puede dar soporte al grupo de rescate debido a su estructura mecánica y acondicionamiento electrónico que le permite movilizarse por estos entornos de difícil acceso.

1.2. Panorama actual

La humanidad ha sufrido innumerables desastres a lo largo de la historia. La Federación Internacional de Sociedades de la Cruz Roja define un desastre como un evento catastrófico repentino que cambia significativamente el funcionamiento de una comunidad o sociedad y causa pérdidas tales como humanas, materiales, económicas y/o ambientales en su totalidad. Los desastres generalmente se dividen en dos categorías principales: desastres naturales, tales como (terremotos, tsunamis, huracanes, deslizamientos de tierra, inundaciones, etc.) y desastres provocados como consecuencia de las malas acciones del hombre (explosiones, incendios, derrames químicos, descarrilamientos, etc.). Las zonas urbanas están más propensas a que sufran grandes pérdidas humanas, materiales, económicas o ambientales. La robótica aplicada en el ámbito de rescate o desastres es un nuevo campo de la robótica móvil que tiene como objetivo proporcionar herramientas tecnológicas que permitan a los rescatistas actuar rápidamente durante los desastres para responder y gestionar la fase de búsqueda y rescate ahorrando así el mayor tiempo posible sin exponer las vidas de los rescatistas [3].

Las tareas que realizarán los robots de rescate aún no están completamente definidas, por lo que, algunos expertos sugieren las siguientes labores: búsqueda, mapeo, reconocimiento, remoción de escombros, inspección estructural, evaluación médica e intervención en el sitio, expansión, cobertura de comunicación, apoyo logístico. Otro factor que determina las características de un robot móvil es el tipo de desastre que se está produciendo, por lo cual cada robot debe estar preparado con características especiales como su tipo y tamaño. Hay cuatro tipos de robots en la actualidad: vehículos terrestres no tripulados (UGV), vehículos aéreos no tripulados (UAV), vehículos submarinos no tripulados (UUV) y vehículos de superficie no tripulados (USV). A través de los despliegues de robots que se han realizado hasta el 2014, 34 fueron reportados y analizados en profundidad, de los cuales solo 29 fueron utilizados en incidentes y los 5 restantes no fueron desplegados. La información que se logró obtener de estos despliegues apunta a desafíos fundamentales que deben abordarse en la robótica de rescate, tales como: movilidad, sistemas de comunicación, control de robots, sistemas de sensores y fuentes de energía [4].

Uno de los eventos clave que impulsó el desarrollo de robots de búsqueda y rescate fue el terremoto de Northridge en 1994 en California. Este desastre natural dejó en evidencia la necesidad de contar con herramientas y tecnologías que pudieran acceder a áreas peligrosas y ayudar a localizar y rescatar

a personas atrapadas. A partir de este evento, se intensificó la investigación y el desarrollo de robots para aplicaciones de búsqueda y rescate [5].

La inteligencia artificial (IA), que ha sido estudiada desde los trabajos publicados en la década de 1940, aunque existen varios trabajos que no han tenido mucho reconocimiento en la comunidad científica, tiene su origen en un artículo publicado en 1950 por el británico Alan Turing llamado "Computo e inteligencia" donde describió los medios de detección para decidir ¿Cuándo es una maquina inteligente?, que llamo el "Tes de Turing", esta se basó en comparar máquinas y humanos, gracias a estos estudios y aportes al campo de la ciencia, Turing es considerado el padre de la inteligencia artificial. Los principales avances en inteligencia artificial ocurrieron entre 1990 y 2000, con avances significativos en el aprendizaje automático. Instrucción inteligente, programación de múltiples agentes, extracción de datos, compresión y traducción de lenguaje natural, visión, realidad virtual y otros temas importantes [6].

La tecnología de grabación de imágenes comenzó en el siglo XIX, con la invención de los daguerrotipos (Primera técnica usada para obtener imágenes) y los inventos de fotografía posteriores. Actualmente, la mayoría de las tecnologías de imágenes móviles (cámaras) funcionan con sensores electrónicos [7].

A lo largo de los años, se han logrado avances significativos en términos de capacidades de los robots, sensores avanzados, inteligencia artificial y colaboración entre diferentes actores, lo que ha mejorado la eficacia y la seguridad de las operaciones de búsqueda y rescate [8].

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un sistema de supervisión para un robot móvil especializado en labores de apoyo de búsqueda y rescate para entornos complejos que permita detectar víctimas, evaluar el entorno y realizar acciones eficientes durante su movilización.

1.3.2. Objetivos específicos

- Desarrollar un software de control inteligente que procese los datos de los sensores y permita al robot tomar decisiones autónomas para

realizar acciones pertinentes durante las operaciones de búsqueda y rescate.

- Establecer algoritmos de toma de decisiones para que el robot pueda adaptarse a situaciones cambiantes y realizar acciones eficientes durante las operaciones de apoyo en búsqueda y rescate.
- Diseñar una interfaz gráfica intuitiva que permita la visualización en tiempo real de las acciones y el estado operativo del robot móvil, facilitando la supervisión y el control durante las tareas de búsqueda y rescate.
- Evaluar el desempeño mediante pruebas exhaustivas para obtener la eficacia del robot en diferentes escenarios de búsqueda y rescate.

1.4. Alcance del Proyecto

Este proyecto se enfoca en el diseño de un sistema de supervisión destinado a un robot móvil terrestre, que tiene como prioridad brindar apoyo en la búsqueda, localización y rescate de personas atrapadas o en peligro, y así minimizar el riesgo a los que se exponen los socorristas en situaciones peligrosas. Para realizar este proyecto se va a adquirir un robot móvil ensamblado, al cual se le implementará un sistema de percepción del entorno basado en una cámara y varios sensores los cuales serán de gran apoyo para lograr cumplir el objetivo de este proyecto.

Además, el proyecto utilizará un sistema de inteligencia artificial para procesar datos en tiempo real y tomar decisiones autónomas en situaciones de presentarse algún obstáculo ante el robot móvil. Y así lograr evadirlo hasta llegar al objetivo. Mediante la demostración del sistema implementado, se llevarán a cabo pruebas adecuadas para asegurar su correcto funcionamiento, con el propósito de demostrar de manera efectiva el sistema de supervisión implantado en un robot móvil diseñado para brindar apoyo en las tareas de búsqueda y rescate de manera segura y eficiente.

Limitaciones: Debido a la estructura no tan rígida del robot no será posible realizar todos los posibles movimientos requeridos lo que conllevará al que el rendimiento operativo del robot disminuya, por ejemplo, al someter al robot en ambientes extremos como situaciones de inundaciones e incendios, este no será capaz de lograr su propósito ya que sus componentes no están diseñados para resistir el agua ni altas temperaturas.

Aunque el robot móvil terrestre pueda ser diseñado para moverse de manera autónoma, su movilidad puede verse limitada en comparación con los seres humanos. Por ejemplo, puede tener dificultades para subir escaleras, atravesar terrenos extremos, entre otros. Además, pueden presentarse obstáculos complicados por su gran tamaño, los cuales el robot móvil no podrá evadirlos y buscar una ruta alterna para continuar con la operación. El robot móvil tratará de evadirlos hasta donde sus capacidades lo permitan.

Otro factor importante a tener en cuenta es el tiempo de la operación en el entorno, debido a que la fuente de energía del robot no es continua y por lo tanto limitada. Esto dependerá del diseño y la capacidad de la batería con la que cuente, pueden tener limitaciones en cuanto a la duración de la operación ya que en algún momento va a necesitar recargarse. Además, en situaciones de desastre o áreas remotas, puede ser difícil el cambio o recargo de la fuente de energía para el robot.

El robot móvil contará de sensores y una cámara para detectar objetos y proyectar imágenes a los operadores, sin embargo, pueden tener dificultades para localizar y reconocer objetos en condiciones adversas y no controladas, como lo son los entornos oscuros, con humo o con obstáculos que ocultan la visibilidad.

En las operaciones de búsqueda y rescate, es crucial que el robot móvil terrestre pueda comunicarse con el personal de rescate. Sin embargo, las limitaciones en la infraestructura de comunicaciones o la interferencia pueden afectar la capacidad de los robots para transmitir información que detecten la cámara y los sensores, así como él envió de datos para controlar los actuadores del robot

1.5. Fundamentos teóricos

En los próximos apartados se tratarán temas esenciales como los fenómenos naturales, la robótica móvil, la visión artificial, las redes neuronales, entre otros aspectos importantes. El objetivo es ofrecer una visión completa de los avances en estas áreas tecnológicas.

A lo largo del documento se tratarán temas esenciales como los fenómenos naturales, la robótica móvil, la visión artificial, las redes neuronales, entre otros aspectos importantes. El objetivo es ofrecer una visión completa de los avances en estas áreas tecnológicas.

1.5.1. Fenómenos Naturales - Sismos (terremotos)

Un terremoto es un fenómeno geológico caracterizado por varias vibraciones en la superficie de la tierra, que es el resultado del movimiento de las capas internas en el suelo. La palabra "terremoto" proviene de la expresión griega "Sixmós", que significa "choques". Los terremotos ocurren regularmente y se asocian con el movimiento de las placas tectónicas. Estas placas pueden moverse, deslizarse, chocar o deformarse, crear energía y liberarla en vibración. Este tipo de terremoto se conoce como terremotos tectónicos [1].

Los sismos pueden ser generados no solo por el movimiento de las placas tectónicas, sino también por procesos volcánicos en los que la salida de magma hacia la superficie produce sacudidas en la tierra. Además, otros fenómenos como los deslizamientos de laderas o el colapso de cavidades cársticas pueden desencadenar sismos. El estudio de los sismos es realizado por la sismología, una rama de la geofísica. El lugar en el interior de la Tierra donde se origina el sismo se conoce como foco o hipocentro, mientras que su proyección en la superficie terrestre se denomina epicentro [1].

Los terremotos pueden tener diferentes consecuencias para la vida humana, como la fractura de la tierra, la destrucción de las estructuras, la pérdida de la vida humana, los incendios, los tsunamotes, el tsunami y los deslizamientos de tierra. Más de trescientos mil terremotos visibles ocurren en todo el mundo cada año, aunque la mayoría de ellos no causan daños significativos o pérdidas significativas. Es importante enfatizar que los terremotos no se pueden predecir para su ubicación, tamaño o momento exacto[1].

Los sismos pueden ser clasificados en diferentes categorías según el tipo de movimiento (véase la figura 1) que exhiban, como oscilatorios o trepidatorios.

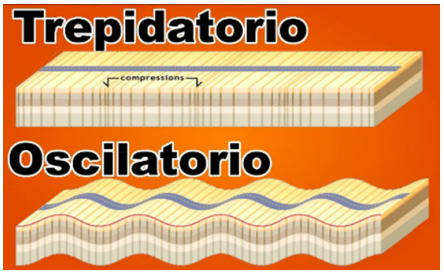


Figura 1: Tipos de sismos [52].

- **Sismos Oscilatorios:** Se refieren a un tipo de sismo en el cual el movimiento de los temblores ocurre de manera horizontal, generando una sensación de balanceo u oscilación, similar a la sensación de moverse de un lado a otro [1].
- **Sismos Trepidatorios:** Se tratan de un tipo de sismo en el cual el movimiento se caracteriza por sacudidas verticales, es decir, de arriba hacia abajo. Esta forma de movimiento puede provocar que los objetos sean lanzados hacia el aire [1].

Las dimensiones de un sismo se evalúan utilizando las escalas de magnitud e intensidad. La magnitud, también conocida como escala de Richter, está relacionada con la energía liberada en forma de ondas sísmicas que viajan a través de la tierra.

Los registros llamados sismogramas, brindan información sobre la amplitud máxima de las ondas sísmicas y la distancia entre las estaciones sísmicas y el epicentro. Estos valores se introducen en una fórmula específica para obtener la magnitud del sismo [1].

A continuación, en la Tabla 1 se describen las escalas de magnitudes de los sismos y sus efectos en la sociedad.

Tabla 1: Magnitud y efectos de un sismo

Magnitud escala de Richter	**Efectos de terremoto**
Menos de 3.5	Por lo general, no es perceptible, pero se detecta.
3.5 a 5.4	Es perceptible y causa daños menores en las cercanías del epicentro.
5.5 a 6.0	Provoca daños leves en construcciones deficientemente construidas y otras estructuras en un radio de 10 km.
6.1 a 6.9	Puede causar daños graves en áreas densamente pobladas.
7.0 a 7.9	Terremoto de gran magnitud, causa daños graves en comunidades en un radio de 100 km.
8.0 o mayor	Gran terremoto, destruye por completo las comunidades cercanas y causa estragos en un radio de más de 1.000 kilómetros

1.5.2. Robótica Móvil

La robótica se basa en la informática y se utiliza para desarrollar y operar equipos que actúan como alternativas a las personas en diversas áreas, especialmente en instalaciones industriales. Un robot es una máquina capaz de llevar a cabo acciones complejas, tomar decisiones y ejecutar tareas de manera manipulable. Además, puede ser reprogramado y equipado con diversos sensores que le permiten realizar múltiples acciones según las asignaciones específicas [9].

Un robot móvil es un equipo electromecánico diseñado para desplazarse de manera independiente, sin requerir una conexión física. Cuenta con sensores incorporados que le permiten monitorear continuamente su posición en relación con su punto de partida y su destino. Por lo general, su control se basa en un circuito cerrado. Para facilitar su movimiento, estos robots móviles están equipados con dispositivos de locomoción, como ruedas, patas, pistas, entre otros. Dependiendo del tipo de movimiento que utilizan, los robots móviles se clasifican en tres categorías principales: aéreos, acuáticos y terrestres. (véase la figura 2) [10].

Figura 2: Robots Móviles [10] .

1.5.3. Clasificación de los Robots Móviles

Los robots se pueden clasificar de acuerdo con su cronología, función o área de enfoque, estos robots tienen la capacidad de moverse en el entorno, adaptarse a circunstancias inestables y, en muchos casos, tomar decisiones reales para cumplir ciertas tareas. Hoy en día, pueden trabajar en muchas áreas, como la industria, la educación, la salud, las tareas de rescate e incluso las tareas locales; A continuación, se detalla su clasificación.

1.5.3.1. Robots Aéreos

La robótica aérea es una disciplina tecnológica enfocada en desarrollar, construir y mantener drones y robots aéreos no tripulados (véase la figura 3). Los vehículos aéreos son empleados en misiones de investigación, lo que les permite llevar a cabo tareas con un nivel de autonomía considerable. Su principal desafío radica en lograr que estos dispositivos sean capaces de realizar maniobras de manera autónoma y adaptarse a las variaciones del entorno [11].

Figura 3: Robot aéreo - Dron cuadricóptero [12].

1.5.3.2. Robots Acuáticos

El robot acuático ha sido diseñado con el propósito de llevar a cabo diversas tareas. Entre ellas se encuentran las labores de inspección, donde el robot se

orienta y no requiere de manipulación externa (véase la figura 4)). En otras palabras, es capaz de funcionar de manera autónoma para recopilar información de inspección, como la captura de imágenes mediante una o varias cámaras, además de contar con sensores de temperatura, acústicos y de calidad del agua. En cuanto a las tareas operativas, se refiere a la intervención robótica para realizar inspecciones en tiempo real del entorno operativo. Estas operaciones se conocen como operaciones remotas y abarcan actividades como la apertura y el cierre de válvulas, ensamblaje de componentes y muestreo de investigación. Por ejemplo, pueden apoyar el equipo de rescate para un acceso difícil.[11].

Figura 4: Robot acuático – Robot nadador [14].

1.5.3.3. Robot terrestres

Un robot terrestre es una máquina capaz de moverse o posicionarse automáticamente sobre la superficie de la tierra, estos robots cuentan con sistemas mecánicos diseñados para moverse y operar en el suelo, estos robots están equipados con tecnología de locomoción integrada como ruedas, orugas, patas articuladas o configuraciones híbridas. Estos robots pueden ser controlados a distancia, semiautomáticos o totalmente automáticos. A continuación, se hará referencia a unos cuantos tipos de estos robots.

Robots con Ruedas: Son los tipos de robots móviles más populares y se emplean con frecuencia para el transporte de materiales, mercancías y/o personas (véase la figura 5). Estos robots utilizan ruedas, lo que les confiere una gran movilidad y eficiencia en términos de distancia recorrida en relación con la energía empleada. Sin embargo, suelen estar limitados a entornos con superficies mayormente planas o ligeramente irregulares [13].

Figura 5: Robot con ruedas – RobotNik [15].

Robots Orugas con Cadena: Son simplemente robots con ruedas que giran de manera sincronizada. Su estructura particular los hace especialmente adecuados para desplazarse por terrenos irregulares con una eficiencia notable, aunque menor que la de los robots con ruedas convencionales (véase la Figura 6). Estos robots suelen ser capaces de recorrer largas distancias en línea recta, pero para girar necesitan deslizar sus ruedas, lo cual reduce su eficiencia [13].

Figura 6: Robot con orugas – Rover Track [16].

Robots con patas o zoomórficos: Son sistemas mecánicos diseñados para moverse utilizando extremidades que se asemejan a las patas de animales, presentan una estructura similar a la de los animales terrestres, como las arañas (véase la figura 7). A diferencia de los robots mencionados anteriormente, estos robots se mueven de manera menos eficaz, pero se ajustan de manera ideal a terrenos irregulares [13].

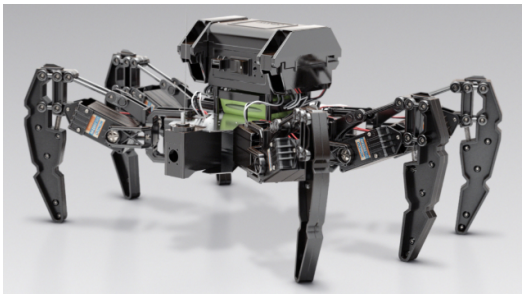


Figura 7: Robot con patas – Robot Kondo [17].

1.5.4. Robots para Entornos Peligrosos o Inaccesibles

La robótica se ha desarrollado notablemente en los últimos años, proporcionando soluciones novedosas para entornos extremos y situaciones de alto riesgo donde la intervención humana es imposible o peligrosa. Estos entornos incluyen áreas con condiciones extremas como calor, radiación, contaminación o lugares inaccesibles como bajo el mar, en el espacio o áreas afectadas por desastres naturales. Para abordar estos desafíos se han desarrollado varios tipos de robots especializados capaces de realizar tareas complejas de forma autónoma o por control remoto.

1.5.4.1. Robots Submarinos

Estos robots, también conocidos como vehículos submarinos no tripulados, se utilizan para la ivestigación y la exploración en aguas profundas, como en la industria petrolera o en la investigación oceanográfica (véase la figura 8). La utilización de robots submarinos ha revolucionado la forma en que se explora el fondo marino. Estos robots posibilitan llevar a cabo tareas en aguas profundas sin requerir la presencia de un vehículo tripulado. El futuro de esta tecnología es muy prometedor, con amplias posibilidades de avance y desarrollo [18].

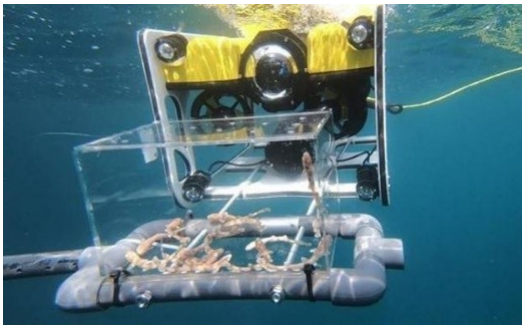


Figura 8: Robot submarino – Dron submarino [19].

1.5.4.2. Robots Espaciales

Los robots espaciales, como los rovers, son diseñados para operar en ambientes fuera de la tierra, como la luna o marte, donde las condiciones son extremadamente inhóspitas para los seres humanos (véase la figura 9). Estos robots realizan exploración, recopilación de datos y son muy usados en experimentos científicos. Los robots en el espacio ofrecen una serie de beneficios potenciales, ya sea al liberar a los astronautas de ciertas tareas, mejorar la calidad de las operaciones al colaborar con la tripulación, o llevar a cabo tareas que actualmente resultarían imposibles para los seres humanos [20].

Figura 9: Robot espacial – Robot curiosity [21].

1.5.4.3. Robots para Desactivación de Explosivos

Estos robots son utilizados por fuerzas militares y equipos de seguridad para desactivar explosivos de manera segura. Están equipados con brazos robóticos y cámaras de alta resolución para realizar tareas precisas en entornos peligrosos (véase la figura 10). Desde su presentación en la década de 1970, estos robots para desactivación de explosivos operados a distancia han sido responsables de salvar innumerables vidas. A lo largo de los años, han evolucionado desde su origen en la defensa civil hasta convertirse en una herramienta vital

en las operaciones militares en situaciones de guerra asimétrica. Los últimos avances y desarrollos han dado lugar a robots más rápidos, ágiles y bien equipados, capaces de rivalizar con la destreza de un técnico humano en la desactivación de artefactos explosivos [22].

Figura 10: Robot para desactivar bombas -Robot Vengador [23].

1.5.4.4. Robots de Rescate

Estos robots se utilizan en situaciones de desastres naturales o industriales, como terremotos, incendios o fugas químicas (véase la figura 11). Pueden ser robots de búsqueda y rescate, robots de extinción de incendios o robots de detección de sustancias tóxicas, entre otros. La robótica relacionada con el rescate constituye una novedosa área dentro de la robótica móvil, cuya finalidad es dotar a los equipos de rescate tecnología que les posibilite reaccionar y abordar con prontitud la fase de búsqueda y salvamento durante catástrofes naturales o causadas por humanos, con el objetivo primordial de preservar la mayor cantidad de vidas posible. Su enfoque radica en proteger a las víctimas sin exponer a los rescatistas a riesgos adicionales[24].

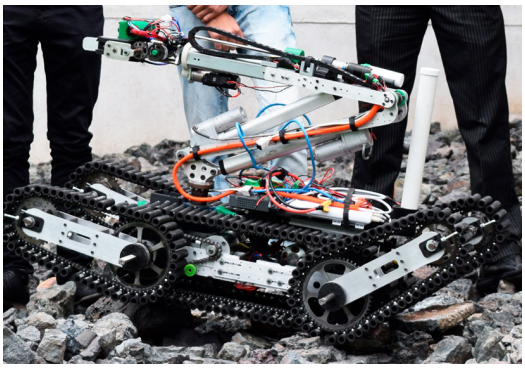


Figura 11: Robot de rescate – Robot Olinki [25].

1.5.5. Cinemática de los Robots Móviles

La cinemática es el aréa de la mecánica que analiza el desplazamiento de los cuerpos sin tener en cuenta sus masas o las fuerzas que lo generan. En robótica móvil, estudia la postura, velocidad y aceleración del robot, enfocándose en la geometría fija del robot respecto a un sistema de referencia global. Para obtener el modelo cinemático completo, se analiza cada rueda, que contribuye al movimiento del chasis y limita sus desplazamientos. Las fuerzas y restricciones de cada rueda se expresan en un sistema de referencia claro, esencial en robótica móvil para relacionar sistemas de referencia globales y locales y determinar la cinemática total del robot [26].

1.5.5.1. Cinemática Directa e Inversa del Robot

En la robótica móvil con ruedas, se usa la cinemática diferencial, que relaciona velocidades en lugar de posiciones u orientaciones, ya que las ruedas imponen limitaciones al desplazamiento del robot, muchas de estas limitaciones de velocidad son no integrables, lo que impide obtener relaciones directas de posición. Los análisis cinemáticos necesarios para un robot móvil se dividen en dos tipos: directo e inverso[26].

Cinemática diferencial directa: Conociendo las características de las ruedas (como el radio, la posición geométrica en el robot y la distancia entre ellas), un punto de referencia del robot (usualmente el centro geométrico) y las velocidades de giro de las ruedas, es posible calcular la velocidad del robot en relación con un sistema de referencia global. Esto permite determinar su velocidad cartesiana $(\dot{x}, \dot{y}, \dot{\theta})$, a partir de las velocidades de rotación de las ruedas $(\dot{\varphi}_i, \dot{\beta}_i)$, y sus orientaciones $(\dot{\beta}_i)$. Con esta información, se puede establecer la posición del robot en el plano global.

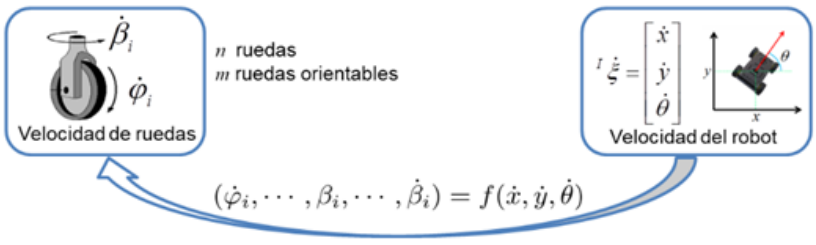


Figura 12: Cinemática diferencial directa [26].

$(\dot{\varphi}_i)$ = rapidez del giro de la rueda

$(\dot{\beta}_i)$ = variacion de orientacion de la rueda

Cinemática diferencial inversa: permite determinar las velocidades $(\dot{\varphi}_i, \dot{\beta}_i)$, y orientaciones individuales de las ruedas de un robot móvil terrestre para que este logre un movimiento específico en el espacio, dado un conjunto de velocidades deseadas en coordenadas cartesianas $(\dot{x}, \dot{y}, \dot{\theta})$. A través de este enfoque, se calculan las velocidades necesarias para cada rueda, tomando en cuenta las características del sistema de tracción, como la distancia entre las ruedas y sus radios, para que el robot se mueva de manera precisa en una trayectoria o dirección particular.

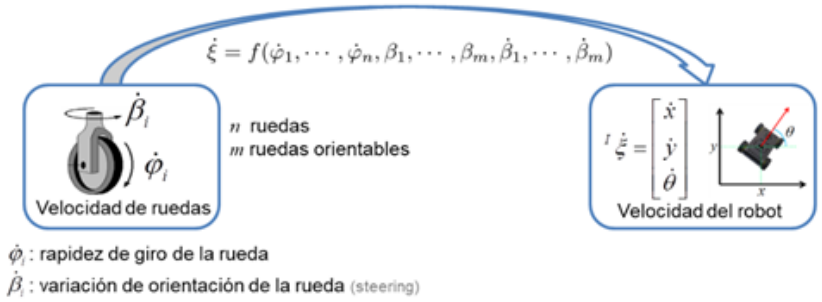


Figura 13: Cinemática diferencial inversa [26].

1.5.5.2. Locomoción de los Robots Móviles - Tipo Oruga

Como se observa en la figura 14, cuando ambas ruedas giran en la misma dirección y a la misma velocidad, el robot avanzará en línea recta, si se invierte el sentido de giro, pero se mantiene la velocidad, el robot se moverá en reversa.

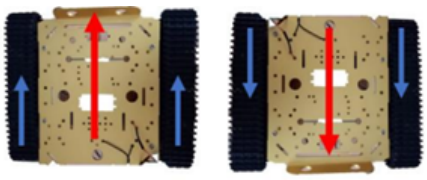


Figura 14: Control de movimiento de un robot móvil diferencial [27].

En la figura 15 se observa si ambas ruedas giran en la misma dirección, pero a velocidades distintas, el robot se desplazará hacia el lado opuesto a la rueda que gira más rápido. Por ejemplo, si la rueda derecha gira más rápido que la izquierda, el robot se dirigirá hacia la izquierda.

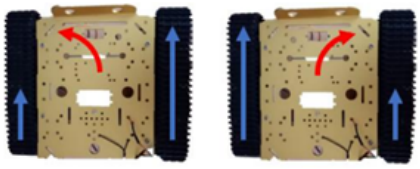


Figura 15: Control de rotación de un robot móvil diferencial [27].

Como se aprecia en la figura 16, si las dos ruedas giran a igual velocidad, pero en direcciones opuestas, el robot girará sobre su propio eje, ya sea en sentido horario o antihorario.

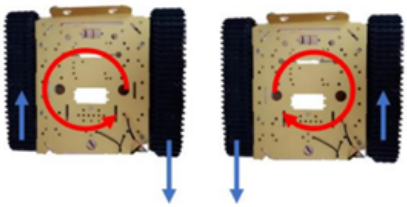


Figura 16: Rotación sobre el eje en un robot móvil diferencial [27].

Para determinar la localización del robot móvil tipo oruga en el plano cartesiano, se utiliza el modelo cinemático diferencial directo. Este modelo establece una relación entre las velocidades del punto de control h(x, y) y las velocidades de los actuadores, asumiendo que el vehículo se comporta como una masa puntual, sin tener en cuenta las fuerzas que actúan sobre él, como los momentos de inercia o los efectos del rozamiento.

En el caso de un robot móvil con direccionamiento diferencial, el objetivo principal del modelo cinemático es describir la velocidad del robot en función de las velocidades de sus ruedas y de sus parámetros geométricos. Este tipo de robot, diseñado para desplazarse mediante ruedas sobre una superficie, se analiza bajo ciertas hipótesis simplificadoras: el movimiento ocurre sobre una superficie plana y horizontal (con energía potencial constante), los ejes de referencia son perpendiculares al suelo, la estructura del robot y sus ruedas son rígidas, el contacto rueda-suelo se considera puntual, y no se produce deslizamiento. Bajo estas condiciones, el análisis se limita al plano bidimensional, ya que el movimiento del robot se restringe a un solo plano [27].

El análisis comienza estableciendo un sistema de referencia fijo $\sum_0(X_0, Y_0, Z_0)$ de manera conveniente como se observa en la figura 17. A continuación, se define un sistema de referencia móvil $\sum_r(X_r, Y_r, Z_r)$, el cual permite determinar los componentes de traslación y la orientación del robot móvil con respecto al sistema fijo $\sum_0$.

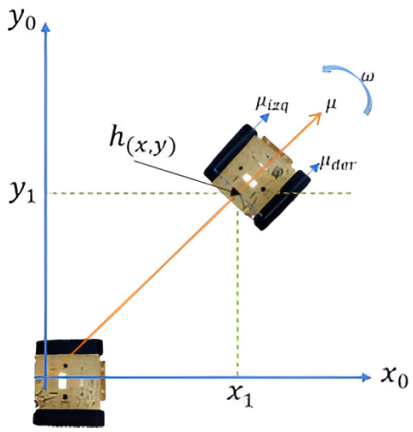


Figura 17: Geometría de un robot móvil tipo diferencial [27].

A partir de la información, se puede derivar el modelo cinemático para un vehículo diferencial, cuya posición está definida por el punto de control $h(x, y)$ y su orientación por el ángulo φ. Como resultado del análisis geométrico, la cinemática diferencial con el punto de control ubicado en el centro del eje que conecta las ruedas motrices se describe mediante las siguientes ecuaciones:

$$\begin{aligned} h_x &= x_1, \quad & \dot{h}_x &= \mu\cos(\varphi) \\ h_y &= y_1, \quad & \dot{h}_y &= \mu\sin(\varphi) \\ \dot{\varphi} &= \omega \end{aligned} \tag{1}$$

$$\begin{bmatrix} \dot{h}_x \\ \dot{h}_y \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu \\ \omega \end{bmatrix}$$
$$\uparrow \qquad\qquad \uparrow \qquad\quad \uparrow$$
$$\dot{\hat{h}} \qquad\qquad J \qquad\quad \dot{q}$$

Donde:

$$\begin{aligned} \mu &= \text{Velocidad lineal} \\ \omega &= \text{Velocidad angular} \\ J &= \text{Matriz Jacobiana} \end{aligned}$$

1.5.6. Navegación con Sensores

La navegación es una característica importante en la robótica móvil, ya que permite a los robots trasladarse sin la necesidad de intervención directa

de una persona, adaptándose de forma flexible al entorno a través de la integración de varios sensores. Este mecanismo conecta distintas tecnologías para la detección del entorno, localización y la toma de decisiones en tiempo real, lo que instruye al robot para evitar obstáculos y alcanzar su destino de manera eficiente y segura. La navegación representa un componente importante en la robótica móvil [28].

1.5.6.1. Tipos de Sensores

Sensores de distancia y proximidad: son dispositivos fundamentales en sistemas electrónicos y de automatización, ya que permiten detectar la presencia de objetos cercanos o medir la distancia entre el sensor y un obstáculo. Estos sensores se utilizan ampliamente en aplicaciones como vehículos autónomos, robótica, seguridad, automatización del hogar y sistemas de gestión industrial. Su capacidad para ofrecer datos precisos sin contacto físico los convierte en herramientas clave para la navegación, el mapeo y la toma de decisiones en tiempo real [28].

Existen diferentes tipos según su principio de funcionamiento:

- **Sensores ultrasónicos:** Crean sonidos a altas frecuencias y un tiempo de registro que requiere retorno de eco. Son muy frecuentes debido a su bajo precio y efectividad en la detección de obstáculos.
- **Sensor infrarrojo (IR):** Detecta objetos por la reflexión de luz infrarroja. Funciona mejor en interiores con adecuada iluminación, pero es menos confiable que el ultrasónico a largas distancias.
- **LiDAR (Light Detection and Ranging):** Emite impulsos de láser para generar un modelo en tres dimensiones del área circundante. Ofrece gran exactitud en la identificación de obstáculos y es perfecto para una navegación sofisticada y para el mapeo.

Sensores de posicionamiento: son dispositivos diseñados para determinar la ubicación, orientación o desplazamiento de un objeto en el espacio. Estos sensores son esenciales en aplicaciones como la robótica, los sistemas de navegación, la automatización industrial y los vehículos autónomos, ya que permiten conocer con precisión dónde se encuentra un elemento y cómo se mueve [28].

Existen diferentes tipos según su principio de funcionamiento:

- **Módulo GPS (Global Positioning System):** Permite obtener la ubicación global (latitud, longitud) del robot. Es fundamental en tareas de patrullaje en exteriores y navegación en terrenos abiertos.

- **UWB (Ultra-Wideband) o RTK-GPS:** Ofrecen mayor precisión que el GPS convencional, útiles en entornos donde se requiere localización milimétrica.

Sensores de orientación: Permiten detectar y calcular el movimiento, el ángulo y la dirección del objeto en el espacio. Son fundamentales en sistemas donde es necesario conocer la posición respecto a otro punto o el comportamiento dinámico de un cuerpo, como en la robótica, los drones, los vehículos autónomos o los dispositivos móviles. Estos sensores, entre los que se incluyen acelerómetros, giroscopios, magnetómetros y unidades de medición inercial (IMU), trabajan en conjunto para ofrecer datos precisos sobre la rotación, aceleración y dirección. Gracias a ellos, un robot puede mantener el equilibrio, seguir una trayectoria o corregir su rumbo con base en su orientación actual [28].

Existen diferentes tipos según su principio de funcionamiento:

- **MPU9250 (IMU):** Sensor de movimiento que incluye acelerómetro, giroscopio y magnetómetro. Permite medir la orientación, detectar giros y estabilizar el rumbo.
- **Brújula electrónica:** Detecta el campo magnético terrestre. Puede usarse para mantener una dirección constante durante la navegación.

1.5.7. Visión Artificial

La visión artificial, también conocida como visión por computadora, busca emular la capacidad que los seres vivos poseen para observar una escena (imagen), comprenderla y actuar conforme a la situación. El surgimiento de la visión artificial tiene como objetivo proporcionar un sistema visual que permita a las máquinas automatizar el proceso de percepción visual mediante el procesamiento digital de imágenes (véase la figura 18). En esencia, la visión por computadora se define como el proceso de extraer, mostrar e interpretar información de imágenes bidimensionales en un mundo tridimensional [29].

Figura 18: Visión artificial [29].

En la Tabla 2 se describen las etapas de la visión artificial.

Tabla 2: Etapas de la visión artificial

Etapas	**Nivel de la visión**	**Entrada**	**Salida**	**Área**
1. Captura	Bajo	Imagen	Imagen	Procesamiento de imágenes
2. Preprocesamiento	Bajo	Imagen	Grupos de píxeles	Análisis de imágenes
3. Segmentación	Medio	Imagen	Grupos de píxeles	Análisis de imágenes
4. Descripción	Medio	Objetos	Información cuantitativa	Análisis de imágenes
5. Reconocimiento	Medio	Información cuantitativa	Categorías	Análisis de imágenes
6.Interpretación	Alto	Categorías	Comprensión de la escena	Visión por computadora

Fuente: adaptado de Víctor Caranqui Sánchez (2015)

- **Captura:** El primer paso en cualquier sistema de visión artificial es capturar una imagen mediante una cámara o sensor digital. Esta etapa es muy importante, porque la calidad y resolución de las imágenes capturadas afectarán directamente la precisión del análisis posterior.
- **Preprocesamiento:** En esta fase la imagen capturada es sometida a diversos procesos para mejorar su calidad y eliminar cualquier ruido. Las técnicas de preprocesamiento incluyen corrección de luz, ajuste de contraste, eliminación de artefactos y nitidez para facilitar el procesamiento posterior.
- **Segmentación:** La segmentación implica dividir una imagen en diferentes áreas o segmentos en función de características como el color, la textura o los contornos. Este proceso permite aislar objetos o áreas de interés para su posterior análisis.

- **Descripción:** Cuando se segmenta una imagen, se capturan características relevantes como bordes, formas o patrones. Estos descriptores proporcionan información clave que se puede utilizar para identificar o distinguir objetos en una imagen.

- **Reconocimiento:** En esta etapa, se compara la descripción obtenida con un conjunto de patrones previamente almacenados o entrenados en el sistema. El objetivo es identificar y clasificar los objetos presentes en la imagen, asignándoles etiquetas específicas según su similitud con las referencias almacenadas.

- **Interpretación:** La interpretación es el proceso de traducir los resultados del reconocimiento en acciones o decisiones. El sistema analiza la información obtenida para tomar decisiones o generar respuestas adecuadas, que pueden incluir controlar acciones en sistemas automatizados o proporcionar resultados a los usuarios.

Los procesos de visión artificial se clasifican en tres niveles, según su complejidad e implementación:

Procesamiento de bajo nivel: Comprende operaciones básicas como la adquisición o captura de imágenes, así como el preprocesamiento para reducir el ruido y mejorar el contraste y la claridad de la imagen.

Procesamiento intermedio: Involucra operaciones como la segmentación y descripción de los diversos objetos presentes en la escena, con el propósito de reducirlos a una forma adecuada para su posterior procesamiento, reconocimiento y clasificación mediante sistemas computacionales.

Procesamiento de orden superior: Se relaciona con el reconocimiento de grupos de objetos en una imagen y la realización de funciones cognitivas que se asocian típicamente con la visión humana. A estos procesos se les denomina interpretativos [30].

1.5.7.1. Elementos de un Sistema de Visión Artificial

La integración de diferentes elementos en un sistema de visión artificial es crucial para su desempeño en tareas de análisis y detección. Desde la iluminación apropiada hasta el procesamiento de imágenes capturadas, cada componente es importante para asegurar que el sistema funcione de manera eficaz. Estos componentes trabajan juntos para proporcionar soluciones innovadoras en áreas como la automatización, la seguridad y la salud, permitiendo una mejor interacción entre las máquinas y su entorno visual. La integración de diferentes elementos en un sistema de visión artificial es fundamental para su desempeño en tareas de análisis y reconocimiento. Desde la

iluminación apropiada hasta el procesamiento de imágenes capturadas (véase la figura 19).

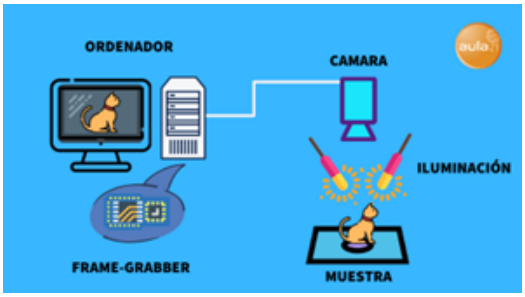


Figura 19: Elementos de un sistema de visión artificial [30].

- **Sistema de iluminación:** Tiene como objetivo principal mantener una intensidad y dirección de luz constante, así como optimizar el contraste para diferenciar objetos del fondo.
- **Sistema digital de cámaras:** Es capaz de capturar y almacenar imágenes digitalmente utilizando un sensor. La resolución de una cámara depende no solo del número y distribución de píxeles, sino también de factores como las características del sensor y del objetivo. La tarjeta de adquisición conecta la cámara a una computadora, digitaliza y almacena las imágenes en la memoria, además de realizar el procesamiento necesario. Entre las características clave de la tarjeta se encuentran el precio, la capacidad de controlar parámetros, la resolución, la velocidad de transferencia y el almacenamiento.
- **Módulo de procesamiento o Software:** Puede ser una computadora o un sistema integrado que recibe, almacena y procesa imágenes mediante algoritmos adecuados para obtener la información necesaria, tomando decisiones con base en la visión artificial. El software de procesamiento de imágenes es un conjunto de herramientas utilizadas para analizar imágenes y extraer información mediante los algoritmos en los que se basan estas herramientas, haciendo uso de la interpretación y análisis de píxeles [30].

Por ejemplo, en el ámbito de búsqueda y rescate, la visión artificial ha demostrado ser una herramienta valiosa y sus beneficios son evidentes, el uso de drones y robots autónomos en situaciones de emergencia permite localizar y rescatar a personas con mayor rapidez y precisión. La visión artificial aplicada

a tareas de búsqueda y rescate permite a los sistemas automatizados analizar rápidamente imágenes y videos para localizar personas o detectar señales de emergencia en áreas de difícil acceso. Utilizando drones y cámaras, junto con algoritmos de reconocimiento de patrones, estos sistemas pueden identificar movimientos, calor corporal o características específicas del entorno. Esto mejora la velocidad y precisión de las misiones de rescate, permitiendo una respuesta más rápida y eficiente en situaciones críticas.

1.5.7.2. Proceso de Detección con Visión Artificial

La visión artificial forma parte del campo de la inteligencia artificial que permite a las computadoras interpretar y evaluar información visual del mundo, emulando las capacidades del ojo humano. Esta tecnología ha revolucionado muchas áreas de la tecnología, permitiendo que las máquinas analicen, comprendan y extraigan información de imágenes y videos. En este campo se han desarrollado diversas técnicas para resolver problemas específicos relacionados con la interpretación visual, como la clasificación de imágenes (encargada de asignar etiquetas a las imágenes completadas) y la detección de objetos (enfocada a encontrar e identificar elementos específicos en una escena). Como se observa en la figura 20, la clasificación de imágenes permite que una computadora capte una imagen y la clasifique en la categoría exacta a la que pertenece, la visión por computadora puede comprender categorías y etiquetarlas, como árboles, aviones o edificios. Un ejemplo es una cámara que puede reconocer rostros en fotografías y enfocarlos [31].

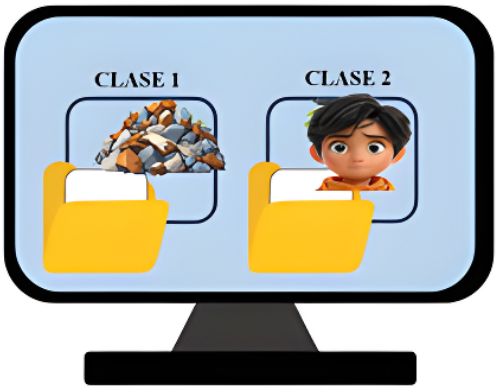


Figura 20: Clasificación de imágenes [Autoría Propia].

El procesamiento de imágenes puede ser un paso clave en tareas de clasificación, ya que permite optimizar y preparar las imágenes para que los algoritmos de clasificación puedan analizarlas de forma efectiva el procesamiento de imágenes incluye técnicas como la mejora de la calidad visual, la

reducción del ruido, el cambio de tamaño y la extracción de características específicas, estos ajustes facilitan que los modelos de clasificación identifiquen patrones o características relevantes en las imágenes, aumentando la precisión del proceso de clasificación [32].

Por otro lado, la detección de objetos a través de visión artificial nos permite la detección y localización de imágenes, está clasificación se usa para identificar, ordenar y organizar imágenes. La detección de objetos se utiliza en procesos industriales y de fabricación para controlar aplicaciones autónomas y monitorear líneas de producción [31].

La detección de objetos es una de las áreas más destacadas de la visión artificial utilizada para identificar y localizar elementos específicos dentro de imágenes o videos. Esta tecnología abarca diversas aplicaciones, desde el reconocimiento de personas hasta la identificación de obstáculos, permitiendo a los sistemas interpretar y reaccionar ante su entorno con mayor precisión. A través del entrenamiento de redes neuronales y el uso de algoritmos avanzados, se logran enfoques como la segmentación semántica y de instancias, que permiten clasificar píxeles y distinguir objetos individuales, respectivamente (véase la figura 21). La detección de objetos implica reconocer elementos como personas, animales u objetos y localizar su posición en una imagen o video. Para lograr esto, es común entrenar redes neuronales para identificar estos objetos, existen dos enfoques similares: la segmentación semántica, que asigna cada píxel a una clase específica, y la segmentación de instancias, que distingue objetos individuales dentro de una misma clase [33].

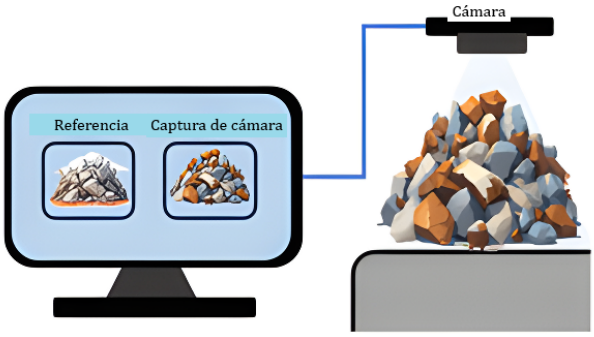


Figura 21: Identificación con visión artificial [Autoría Propia].

A continuación, se explora cómo estas técnicas se aplican en la detección de personas y obstáculos, dos áreas clave en el desarrollo de esta propuesta.

Detección de Personas: Identificar a una persona en un entorno vi-

sual significa el reconocimiento de un conjunto de características distintivas, como forma, tamaño, contorno y movimiento. La tecnología de visión artificial analiza los patrones faciales, las estructuras corporales y los cambios y los asocia con personas específicas en fotos y videos. El algoritmo de reconocimiento también puede considerar factores como actitudes, colores de ropa e interacciones con el medio ambiente. Además, las redes neuronales con imágenes grandes de imágenes se utilizan para mejorar con precisión en diferentes condiciones de iluminación, unidades y distancias. [34].

Detección de Obstáculos: La detección de obstáculos, por otro lado, se enfoca en identificar objetos que podrían representar un peligro o interferencia en el camino de una máquina o persona. El algoritmo analiza las propiedades físicas de los obstáculos, como la forma, el tamaño, y posición en el espacio. Dependiendo de la tecnología utilizada, pueden incluir sensores como cámaras térmicas, Lidar, o sensores ultrasónicos para obtener una representación precisa del entorno. Esta información es fundamental para la navegación autónoma, ya que permite a los sistemas evitar colisiones y realizar movimientos seguros dentro de su entorno [27].

1.5.8. Redes Neuronales

Las redes neuronales son una técnica de inteligencia artificial (IA) que enseña a las computadoras a procesar datos de manera similar al cerebro humano. Este es un proceso de aprendizaje automático (ML) llamado aprendizaje profundo que utiliza nodos o neuronas interconectados en una estructura jerárquica similar al cerebro humano (véase la figura 22), estas redes pueden ayudar a las computadoras a tomar decisiones precisas con la ayuda de personas limitadas. Esto se debe a que pueden aprender y modelar una relación compleja y no lineal entre los datos de entrada y salida [35].

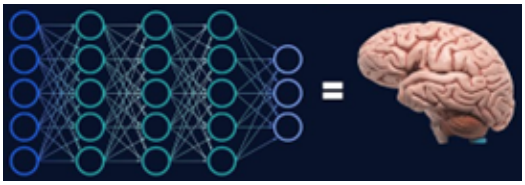


Figura 22: Redes Neuronales [35].

1.5.8.1. Arquitectura de una Red Neuronal

Como se puede observar en la figura 23, la arquitectura de una red neuronal es esencial para comprender cómo las máquinas pueden adquirir información

y tomar decisiones a partir de datos. Este tipo de red simula el proceso de aprendizaje humano al procesar información de manera jerárquica y secuencial, a través de algoritmos y estructuras matemáticas, las redes neuronales permiten a las computadoras identificar patrones, clasificar datos y realizar tareas complejas [35].

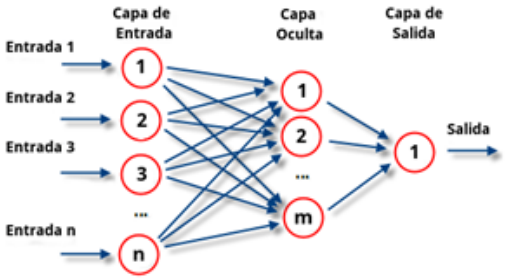


Figura 23: Arquitectura de una Red Neuronal [35].

La capa de entrada es el lugar donde la información es captada por la red neuronal. Los nodos en esta capa procesan, analizan o clasifican los datos y los envían a la siguiente capa para su posterior procesamiento.

La capa oculta recibe información de la capa de entrada u otras capas ocultas. Las redes neuronales artificiales pueden tener múltiples capas internas, cada capa oculta analiza la salida de la capa anterior, la analiza con mayor profundidad y la envía a la capa siguiente.

La capa de salida proporciona el resultado final de todo el procesamiento de datos realizado por la red neuronal artificial, adaptándose según el tipo de problema abordado. Puede tener uno o más nodos. Por ejemplo, si tenemos un problema de clasificación binaria (verdadero/falso), la capa de salida tendrá un nodo de salida con un valor entre 1 o 0.

1.5.8.2. Fases de Entrenamiento de una Red Neuronal

El proceso de entrenamiento de una red neuronal es fundamental para que el modelo aprenda a reconocer patrones y realizar predicciones precisas, los datos proporcionados para el entrenamiento están divididos en tres categorías, descritas a continuación.

Entrenamiento (training): Es el proceso mediante el cual la red ajusta sus parámetros internos para aprender a realizar una tarea, para que una red neuronal funcione, es necesario entrenarla previamente con datos (véase la

figura 24). Si tomamos el ejemplo de una red diseñada para identificar especies de plantas, los datos serían imágenes de hojas o flores y sus respectivas etiquetas. El conjunto de datos de entrenamiento proporcionado a la red por el entrenador se divide en lotes iguales de n elementos. Los cambios de peso sináptico de cada conexión (iteraciones) ocurren al final de cada lote, se considera que una era ha concluido cuando la red ha finalizado el análisis de todos los lotes, normalmente, el entrenamiento implica cientos de épocas en las que la red entrega todos los datos una y otra vez. Al final de cada época, estos datos se mezclan y se crea un nuevo lote, que se devuelve a la red [36].

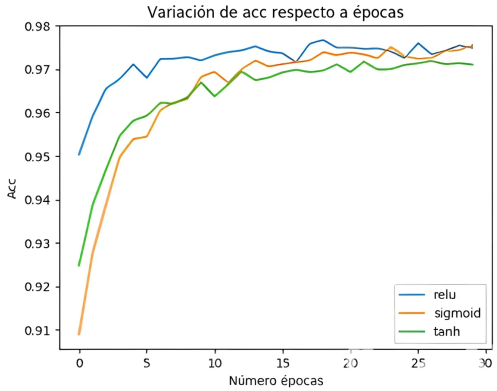


Figura 24: Entrenamiento de una Red Neuronal [37].

Validación (validation): Se realiza cada vez que se completa un análisis por lotes y se actualizan los pesos sinápticos, después de lo cual el valor de la función de pérdida se calcula utilizando datos que la red no vio durante el entrenamiento para monitorear el avance en la efectividad de la red (véase la figura 25), su uso ayuda a detectar el sobreajuste (overfitting) de la red, que ocurre cuando la red simplemente recuerda los datos que recibe durante el entrenamiento y, por lo tanto, no puede generalizar y clasificar datos que no se le han proporcionado previamente [36].

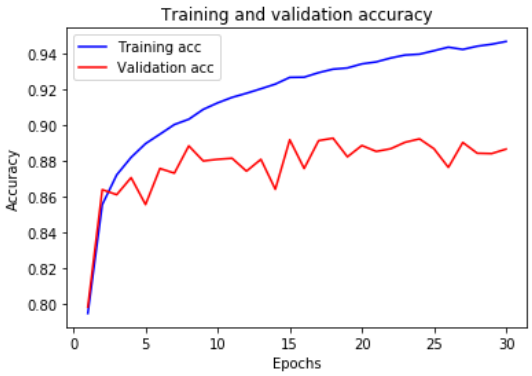


Figura 25: Validación de una Red Neuronal [38].

Comprobación (test): Es la etapa final del entrenamiento de una red neuronal, este paso evalúa la capacidad de generalización de la red, es decir, su capacidad para predecir con precisión nuevos datos que no forman parte del conjunto de entrenamiento o validación (véase la figura 26). Los datos de prueba se utilizan específicamente para medir el rendimiento de la red después del entrenamiento y el ajuste, y no afectan la actualización de los pesos sinápticos durante el entrenamiento [36].

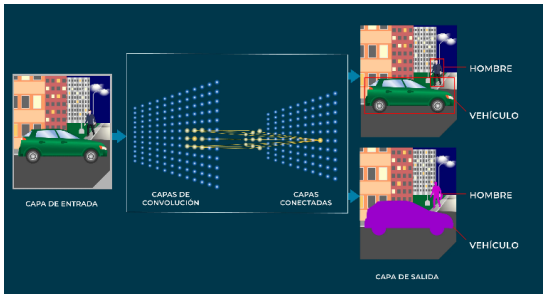


Figura 26: Comprobación de una Red Neuronal [39].

1.5.8.3. Clasificación de las Redes Neuronales

Las redes neuronales se pueden clasificar según la forma en que los datos fluyen desde los nodos de entrada a los nodos de salida, lo que determina su arquitectura y los tipos de tareas que pueden realizar. Existen muchos tipos de arquitecturas de redes neuronales, cada una con características específicas que las hacen más adecuadas para cierto tipo de problemas, como clasificación de imágenes, predicción de series temporales o procesamiento de lenguaje natural [35]. A continuación, se mencionan algunos ejemplos de

estas arquitecturas.

Redes Neuronales Feedforward (Prealimentada): Son diferentes de las redes neuronales recurrentes. Las redes neuronales feedforward emplean el modelo más básico de redes neuronales artificiales, estas redes procesan datos hacia un único sentido desde los nodos de entrada a los nodos de salida (véase la figura 27). Todos los nodos de una capa encuentran conectados con cada uno de los nodos de la capa siguiente capa, estas redes utilizan un proceso de retroalimentación para mejorar las predicciones a lo largo del tiempo [35].

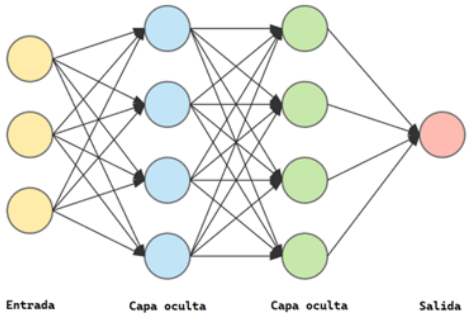


Figura 27: Red neuronal prealimentada [40].

Redes Neuronales Generativas (GAN): Se utilizan para generar datos sintéticos realistas como imágenes, texto o música, consta de dos redes neuronales profundas en competencia: un generador y un discriminador (véase la figura 28). Un generador crea nuevos datos basándose en una distribución de probabilidad aprendida, mientras que un discriminador diferencia entre datos reales y datos generados por el generador [35].

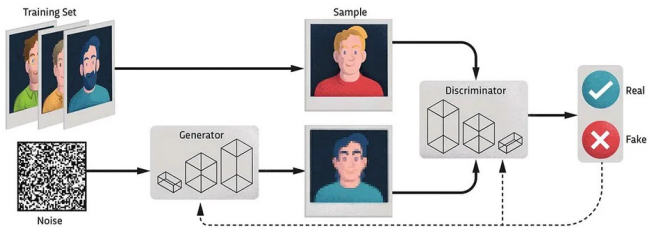


Figura 28: Red neuronal generativa [41].

Red Neural Convolucional: Es un tipo de red neuronal diseñada para procesar datos utilizando un sistema que imita el ojo humano, las capas ocultas en una red neuronal convolucional realizan determinadas funciones matemáticas, como síntesis o filtrado, llamadas convoluciones (véase la figura

29). Se utilizan principalmente para el reconocimiento de imágenes, se usan principalmente para el reconocimiento de imágenes, pero también se pueden usar para clasificar los datos en audio, señales o series de tiemporales. Estas redes consisten en varios píxeles colocados en la red, cada uno con un valor que representa el brillo y el color de cada píxel de la imagen [42].

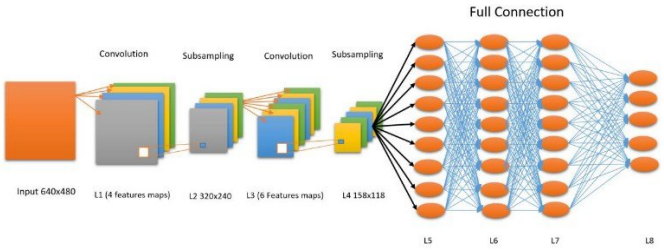


Figura 29: Redes neuronales convolucionales [43].

La siguiente fórmula representa la operación de convolución en imágenes, ampliamente utilizada en redes neuronales convolucionales

$$S(i,j) = (K * I)(i,j) = \sum_{m} \sum_{n} I(m,n)K(i-m, j-n) \tag{2}$$

Donde:

- $S(i,j)$ representa el valor del píxel en la posición (i,j) de la salida S.
- I es la entrada bidimensional en píxeles.
- K es el kernel (matriz).
- $I(m,n)$ representa la imagen en las coordenadas (m,n).

La fórmula describe la operación de convolución, donde un filtro K se desliza sobre una imagen I , realizando un producto punto en cada posición. El resultado S(i,j), es una nueva imagen que extrae características como bordes y texturas. Este proceso reduce parámetros y emula la percepción visual humana, mejorando la eficiencia del modelo.

Una arquitectura convolucional está compuesta por las siguientes capas:

- **Capa Input:** Contiene los valores de píxeles sin procesar de la imagen, en este caso una imagen de ancho b, alto h, con tres canales de color RGB.

- **Capa Conv:** calcula la salida de las neuronas conectadas a regiones locales en la entrada, donde cada neurona calcula el producto escalar entre su peso y la pequeña región a la que está conectada en el volumen de entrada.
- **Capa Relu:** Utiliza una función de activación de elementos como un umbral max(0,x) en cero. Por tanto, la magnitud del volumen sigue siendo la misma.
- **Capa Pool:** Realiza operaciones de muestreo a lo largo de las dimensiones espaciales (ancho, alto) para formar un volumen, donde (b, h) es menor que (B, H). En esta capa no se utilizan pesos que se puedan aprender.
- **Capa densa:** Calcula las puntuaciones de clase, lo que dará como resultado un volumen, donde cada número K corresponde a una puntuación de clase del conjunto de K clases.

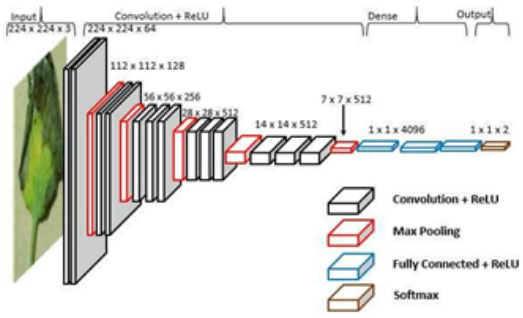


Figura 30: Capas de una red neuronal convolucional [44].

Las Redes Neuronales Convolucionales (CNN) destacan por su excelente capacidad para procesar entradas de señales de imagen, voz o audio, superando a otro tipo de redes neuronales en tareas relacionadas con el procesamiento de datos visuales y auditivos, se componen de tres capas principales: la capa convolucional, que extrae características locales mediante filtros; la capa de agrupación, que reduce la dimensionalidad y resalta características clave; y la capa totalmente conectada, que realiza la clasificación o predicción final. Esta estructura permite a las CNN sobresalir en tareas como el reconocimiento de imágenes y la clasificación de señales. Un detector de características, conocido como kernel o filtro, recorre los campos receptivos de una imagen para identificar características mediante un proceso llamado convolución (véase la figura 31). Este filtro, generalmente una matriz 2D de 3x3, calcula un producto escalar entre los píxeles de la imagen y sus pesos, a medida que el

filtro se desplaza por la imagen, genera un mapa de características, también llamado mapa de activación, que refleja las áreas donde las características están presentes [45].

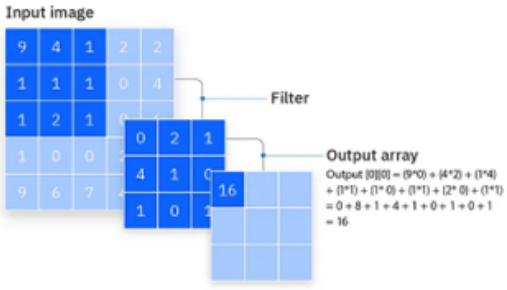


Figura 31: Modelo de una Red neuronal convolucional [45].

MobileNet: El modelo MobileNet es una red neuronal convolucional entrenada en un conjunto de datos de imágenes. Está diseñado para realizar tareas de clasificación de imágenes que tienen como objetivo otorgar etiquetas o categorías a las imágenes de entrada (véase la figura 32). El modelo hace esto aprendiendo un conjunto de características de una imagen de entrada y luego usando esas características para hacer predicciones, diseñadas específicamente para ser eficientes en dispositivos con recursos limitados, como teléfonos móviles, Raspberry pi y dispositivos IoT. Fueron desarrolladas por Google para ofrecer un buen rendimiento en tareas de visión artificial, mientras minimizan el consumo de memoria y capacidad de procesamiento [46].

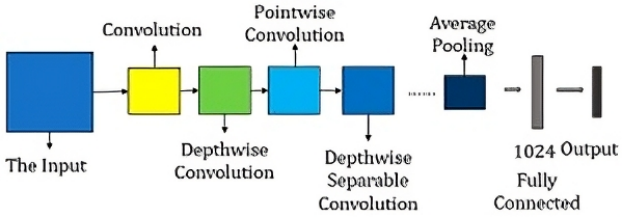


Figura 32: Arquitectura de una red neuronal mobileNet [47].

Redes Neuronales Recurrentes (RNN): Estas redes permiten el procesar secuencias de datos a medida que la información se retroalimenta a través de bucles en la red. Una red neuronal recurrente es una red neuronal que utiliza datos ordenados temporalmente (véase la figura 33). Al igual que las diferentes redes neuronales, las redes recurrentes emplean datos de entrenamiento para adquirir conocimiento. Se caracterizan por la "memoria"

en el sentido de que toman información de entradas pasadas para influir en las entradas y salidas actuales, son usadas comúnmente en la traducción de diferentes idiomas, subtítulos en las imágenes y detección de comandos de voz.[48].

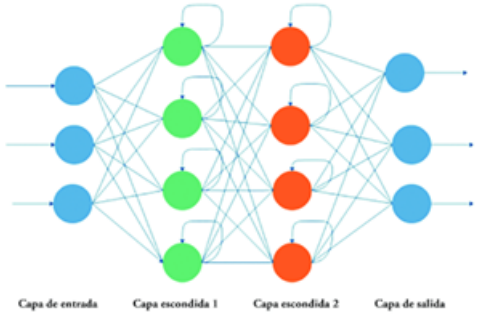


Figura 33: Redes neuronales recurrentes [48].

YOLO: Es un modelo de detección de objetos construido sobre redes neuronales convolucionales que permiten reconocer y localizar múltiples objetos en una imagen en tiempo real (véase la figura 34). A diferencia de otros métodos de detección de objetos que dividen la imagen en regiones y procesan cada una por separado, YOLO trata toda la imagen como una sola entrada a la red neuronal, lo que lo hace extremadamente rápido y eficiente [49].

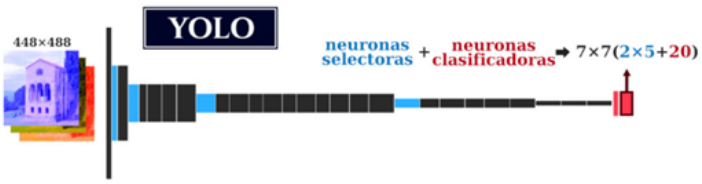


Figura 34: Yolo [49].

1.6. Marco Contextual

En situaciones de emergencia, como desastres naturales o colapsos estructurales, las tareas de búsqueda y rescate representan un desafío significativo debido a las condiciones peligrosas y de difícil acceso. Los terremotos, deslizamientos de tierra, inundaciones o explosiones pueden provocar daños severos en las infraestructuras, la formación de escombros inestables, lo que pone en riesgo a los equipos de rescate. La implementación de un sistema de supervisión para un robot móvil terrestre para tareas de apoyo en búsqueda y rescate permitirá conocer la ubicación del robot en tiempo real, lo cual es crucial para coordinar las operaciones en caso de desastres naturales, deslizamientos de tierra, colapsos de edificios y otras situaciones donde los seres humanos pueden tener dificultades para acceder o realizar tareas de rescate.

En este proyecto se diseñará un sistema de supervisión en un robot móvil terrestre, capaz de trasladarse de manera autónoma por terrenos escabrosos utilizando visión artificial y recolección de datos del entorno del área de búsqueda, trabajando en conjunto con el personal de rescate, proporcionando información de la ubicación de la víctima en tiempo real y aumentando la eficiencia de las operaciones de salvamento, al tiempo que reducen los riesgos para los rescatistas al operar en entornos peligrosos. El entorno de trabajo del robot estará delimitado en espacios donde solo se hayan producido colapsos de edificaciones que conllevan a la aparición de escombros que dificultan el acceso al personal de rescate.

Adicionalmente, se llevará a cabo una supervisión constante y minuciosa del uso del sistema con el propósito de asegurar que el robot mantenga su rendimiento óptimo en todo momento durante las labores de búsqueda y rescate. Esto garantizará que el robot funcione de manera segura y confiable, evitando situaciones peligrosas tanto para el robot como para las personas involucradas en la misión de rescate.

2. Métodos y diseño experimental

2.1. Métodos

Se necesita la metodología de investigación aplicada y exploratoria para la realización del proyecto:

- **Investigación Aplicada:** Este tipo de investigación se enfoca en abordar los desafíos específicos asociados con la utilización de robots en entornos de emergencia y en desarrollar tecnologías y estrategias que optimicen su efectividad y eficiencia en estas situaciones críticas, se pueden llevar a cabo estudios de viabilidad técnica y operativa para determinar cómo los robots pueden ser implementados de manera efectiva en operaciones de búsqueda y rescate.
- **Investigación Exploratoria:** Esto implicaría analizar estudios existentes, recopilar datos sobre las situaciones de emergencia y los escenarios de búsqueda y evaluar los distintos enfoques y tecnologías utilizadas en los robots de búsqueda.

Desde el punto de vista exploratorio, el proyecto investiga una problemática poco abordada en el contexto local, el uso efectivo de robots autónomos de bajo costo en misiones de rescate urbano en Ecuador. Actualmente, no existen soluciones estandarizadas ni protocolos tecnológicos definidos para estas aplicaciones, lo cual abre una línea de investigación sobre robótica de emergencia adaptada a escenarios reales de riesgo.

La construcción de un sistema de supervisión para un robot móvil de búsqueda y rescate, es fundamental integrar tecnologías que respondan de forma efectiva a las exigencias del entorno operativo, sensores confiables, localización precisa, procesamiento visual en tiempo real y capacidad de navegación autónoma. No obstante, la mayoría de las soluciones documentadas en la literatura internacional se han desarrollado en contextos con recursos tecnológicos avanzados, lo que plantea un desafío para su implementación en países como Ecuador, donde estas tecnologías aún no se han adoptado ampliamente en operaciones de rescate reales.

Esta situación evidencia un vacío que este proyecto busca explorar ¿cómo adaptar e implementar tecnologías emergentes en robótica de rescate a escenarios de bajos recursos, manteniendo eficacia operativa? La investigación, por tanto, no solo aborda el diseño técnico de un sistema funcional, sino que explora las condiciones mínimas viables para su adopción local. Además, se sientan las bases para futuras investigaciones enfocadas en la colaboración

entre robots terrestres, el uso de redes neuronales más robustas y la validación del sistema en situaciones de emergencia.

A continuación, se indican las fases de la metodología del proyecto:

- **Fase I Revisión bibliográfica:** Para llevar a cabo el diseño mencionado, se desarrollarán las investigaciones correspondientes en libros, revistas y artículos que traten sobre sistemas de supervisión, robots de búsqueda e inteligencia artificial.
- **Fase II Elección del modelo de robot móvil terrestre:** Para el desarrollo del proyecto se seleccionará un modelo de robot móvil terrestre que cumpla con los requisitos técnicos mínimos, considerando aspectos como movilidad, carga, autonomía, sensores, actuadores, tamaño y costo.
- **Fase III Selección de materiales eléctrico/electrónico:** Se procederá a seleccionar los dispositivos, sensores, cámara y demás componentes necesarios, para que el diseño del sistema sea factible de implementar asegurando de esta manera una mayor precisión en la ejecución de los diferentes movimientos del robot en el proceso de supervisión.
- **Fase IV Adquisición de datos:** A través del procedimiento de adquisición de imágenes, se mostrarán los datos pertinentes de la cámara y la ubicación del robot, los cuales serán transmitidos directamente del robot al usuario en tiempo real.
- **Fase V Sistema de control:** Una vez que los datos sean recibidos, el robot necesita un sistema de control para ejecutar los movimientos necesarios, esto implica controlar motores y actuadores del robot para que se desplace de acuerdo con la ruta determinada.
- **Fase VI Interfaz de usuario:** La interfaz de usuario debe proporcionar una representación visual del entorno en el cual se encuentra el robot, esta interfaz debe permitir a los operadores controlar el movimiento del robot de manera intuitiva, además de permitir visualizar diferentes parámetros de interés.
- **Fase VII Pruebas experimentales:** Es importante evaluar y mejorar continuamente el rendimiento del sistema. Esto implica recopilar datos durante las operaciones reales, analizar los resultados e identificar posibles áreas de mejora.

2.1.1. Descripción del proyecto

El proyecto implica el desarrollo de un sistema de supervisión para un robot móvil para realizar labores de apoyo en el área de búsqueda y rescate. La finalidad es desarrollar un sistema de supervisión para controlar un robót móvil capaz de realizar tareas de localización y supervisión en situaciones de emergencia, especialmente en zonas peligrosas o inaccesibles para el ser humano.

Para asegurar un funcionamiento óptimo y seguro del robot móvil, se desarrollará un sistema de supervisión y control basado en varios sensores para obtener la información del entorno. Este sistema permitirá monitorear en tiempo real el estado propio del robot móvil y su ubicación en su entorno, así como facilitar la comunicación bidireccional entre el robot y los operadores humanos. El robot móvil estará equipado con una amplia variedad de sensores, como cámaras, mpu,gps entre otros. Estos sensores permitirán al robot recopilar datos precisos y en tiempo real sobre el entorno y las condiciones que se encuentran en el lugar de la emergencia. Además, el robot contará con actuadores y mecanismos de movilidad que le permitirán desplazarse eficientemente por diferentes terrenos.

El núcleo que controlará el sistema será un software avanzado de control ejecutado sobre una Raspberry Pi, que procesará los datos recibidos de los sensores y permitirá enviar señales relevantes al robot para la toma decisiones en la ejecución de acciones en el proceso de su movilización. Este software estará basado en algoritmos inteligentes (redes neuronales) que le permitirán al robot móvil adaptarse y responder de manera autónoma a los diferentes desafíos y obstáculos que pueda encontrar durante la operación de búsqueda y rescate.

A continuación en la figura 35, se representa un bosquejo del robot móvil aplicado en tareas de búsqueda de personas.

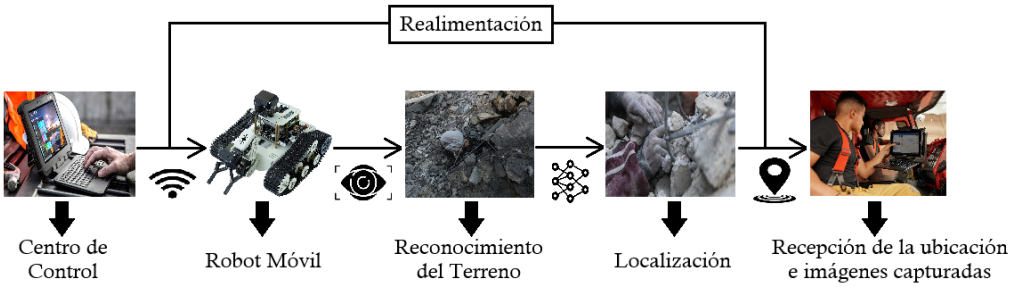


Figura 35: Bosquejo del robot móvil [Autoría Propia].

Los sensores y módulos incorporados sobre el robot móvil recopilarán la información necesaria del entorno, como imágenes y movimiento. Estos datos se enviarán a la unidad de procesamiento, donde se realizará el procesamiento y la toma de decisiones utilizando algoritmos de inteligencia artificial. La comunicación y el control son importantes para establecer enlaces de comunicación con los equipos de rescate humanos, permitiendo la coordinación y colaboración durante las operaciones conjuntas.

El enfoque principal del proyecto será mejorar la eficiencia y la seguridad de las operaciones de rescate, permitiendo que el robot móvil brinde apoyo a los equipos de rescate humano. Además, se buscará optimizar la capacidad de búsqueda y la precisión de la localización de víctimas en situaciones de emergencia, utilizando tecnología avanzada y algoritmos de procesamiento de datos.

Finalmente, el proyecto se centrará en el diseño y desarrollo de un sistema de supervisión para un robot móvil para realizar labores de apoyo en el área de búsqueda y rescate altamente especializado, que aprovechará la tecnología robótica para mejorar la capacidad de respuesta en situaciones de emergencia y brindar apoyo efectivo a los equipos de rescate.

2.2. Componentes de la Propuesta

En la siguiente sección se mencionan cada uno de los componentes físicos y lógicos, que se utilizaron para la implementación del presente proyecto, detallando las características técnicas importantes. Además, de una breve explicación sobre el principio de operación de cada componente, desde los elementos de hardware, como los actuadores y microcontroladores, hasta los componentes de software que permiten la programación y control del sistema.

2.2.1. Componentes Físicos

Los componentes físicos desempeñan un papel importante en la implementación práctica. A continuación, se encuentran detallados los componentes electrónicos utilizados para desarrollar esta propuesta de investigación.

2.2.1.1. Robot Móvil Transbot-SE

El Transbot-SE es un robot móvil diseñado con un sistema de locomoción basado en orugas como se observa en la figura 36, equipado con un brazo robótico 3-DOF y una cámara PTZ 2-DOF lo que le permite moverse con estabilidad y tracción sobre diversos terrenos, incluyendo superficies irregulares. Este tipo de configuración es ideal para situaciones en las que se requiere un alto nivel de maniobrabilidad y robustez. El Transbot-SE se utiliza en aplicaciones de investigación, educación, y puede ser adaptado para escenarios industriales o exploración en entornos difíciles.

Figura 36: Robot móvil Transbot-Se [58]

Este robot es compatible con la placa de desarrollo Raspberry Pi 5, lo que lo convierte en una plataforma excelente para investigar y desarrollar soluciones avanzadas en inteligencia artificial y aprendizaje automático. Los algoritmos integrados de procesamiento de imágenes OpenCV y aprendizaje automático permiten al Transbot-SE realizar funciones sofisticadas como reconocimiento de objetos, seguimiento y conducción autónoma. El Transbot-SE no solo ofrece una plataforma robusta para la experimentación y el desarrollo de nuevas tecnologías en robótica, sino que también sirve como una herramienta educativa para estudiantes e investigadores que desean explorar y aplicar conceptos avanzados en sus proyectos. Su configuración de alta calidad y sus funcionalidades avanzadas lo convierten en una opción ideal para aplicaciones prácticas en el campo de la robótica y la automatización. El Transbot SE incluye una Raspberry Pi 5, un ventilador, una tarjeta SD

con su respectivo adaptador, dos motores 520 con codificadores, una cámara PTZ de 2DOF, una batería recargable, un brazo robótico de 3DOF, una placa de expansión multifunción, un chasis de aleación de aluminio, un par de ruedas de oruga, cables, tornillos y engranajes para conectar los diferentes elementos.

A continuación, se presenta una tabla detallada con las características técnicas del Transbot-SE, destacando los componentes claves que conforman este avanzado robot móvil.

Tabla 3: Características técnicas del Transbot-SE

CARACTERÍSTICAS TÉCNICAS	
Control Principal	Raspberry Pi 5
Sistema operativo	Raspberry Pi Os
Lenguaje de programación	Python/Lenguaje C
Duración de la batería	3 horas
Esquema de energía	12.6V 4400mah
Comunicación	Wifi network (LAN/AP)
Control remoto	APP móvil, PC
Peso	2.1 kg

2.2.1.2. Raspberry Pi 5

La Raspberry Pi 5 es una computadora de placa única (SBC) económica diseñada para facilitar la enseñanza de informática y programación (véase la figura 37). Funciona como una computadora completa y se usa para navegar por internet, ver videos, escribir archivos, programar y jugar. Permite la programación a través de sus pines GPIO, que también incluyen comunicaciones serie, SPI e I2C. Estas propiedades significan que puede usarse en proyectos de electrónica y robótica que interactúan con sensores y actuadores, especialmente para aplicaciones en procesamiento de imágenes/vídeo, cámaras y cálculos matemáticos complejos [59].

Figura 37: Rasberry Pi 5 [59].

En la siguiente tabla se detallan las características técnicas de la Raspberry Pi 5, destacando sus componentes claves.

Tabla 4: Características técnicas del Raspberry Pi 5

CARACTERÍSTICAS TÉCNICAS	
Modelo	Raspberry Pi 5 - 4Gb
Procesador	Arm Cortex-A76 quad-core 64-bit
Memoria	4GB LPDDR4X-4267
Conectividad	Wi-Fi 802.11ac doble banda de 2.4Ghz y 5Ghz, Bluetooth 5.0 BLE
Video y sonido	2 puertos micro-HDMI
Puertos	2 puertos USB 3.0, 2 puertos USB 2.0
Alimentación	5.1V/5A (25.5 W) USB-C
Expansión	40 pines macho GPIO (27 E/S, UART, I2C, SPI)

2.2.1.3. Motores con Codificadores

Los motores con codificadores son motores de corriente continua (DC) que incluyen encoders o codificadores. Estos dispositivos se usan en el campo de robótica para dar retroalimentación sobre el lugar y la velocidad del motor, lo que permite un mejor control (véase la figura 38). Estos motores combinan buen torque y distintos rangos de velocidad con un codificador interno, que proporciona información sobre la velocidad y la posición del eje. Esta característica permite aplicar sistemas de control en lazo cerrado, cambiando dinámicamente la velocidad o sentido del motor según la información del encoder, lo que los hace ideales para robots móviles, brazos robóticos y otros trabajos que requieren alta precisión en el posicionamiento [60].

Figura 38: Motor 520 con codificador [60].

A continuación, se detallan las especificaciones técnicas de los motores con codificadores, destacando sus características principales.

Tabla 5: Características técnicas del motor con codificador

CARACTERÍSTICAS TÉCNICAS	
Torque	8.3 kgf.cm
Voltaje de operación	12 V
Velocidad de salida	205±10 RPM
Corriente	4A
Encoder	Hall encoder

2.2.1.4. Cámara Ptz

Una cámara PTZ 2DOF (Pan-Tilt-Zoom) tiene dos grados de libertad y puede girar de 0 a 180° en sentido horizontal y vertical. Está equipado con una cámara USB de alta definición sin unidad, con un ángulo de visión de 120°, píxeles de 120PFS y píxeles de 2MP (véase la figura 39). Su pequeño tamaño puede ahorrar espacio de instalación de manera efectiva, ofreciendo un campo de visión flexible para cubrir grandes áreas sin necesidad de reposicionar manualmente. Estas cámaras son ideales para aplicaciones de vigilancia o robótica, ya que pueden ajustar su ángulo de visión automáticamente para seguir objetos o personas en movimiento [61].

Figura 39: Cámara PTZ 2DOF [61].

A continuación, se presenta una tabla con las características técnicas de la cámara PTZ, destacando sus componentes fundamentales.

Tabla 6: Características técnicas de la cámara PTZ 2DOF

CARACTERÍSTICAS TÉCNICAS	
Modelo	2DOF Camera (Basic versión)
Grados de libertad	2 grados de libertad
Interfaz	USB 2.0
Angulo de visión	180°
Imagenes de pixeles	2 MP (1800)

2.2.1.5. Brazo Robótico

Un brazo robótico de 3DOF (tres grados de libertad) es una máquina que puede moverse en tres ejes distintos o direcciones, lo cual permite realizar movimientos básicos en el espacio 3D (ver la figura 40). Estos tres grados de libertad acostumbran consistir en rotación de la base (eje horizontal), mover hacia arriba o abajo el brazo principal (eje vertical) y rotar o inclinar más en la articulación del brazo o su extremo. Este tipo de brazo robot es comúnmente usado en tareas sencillas como manipular, ensamblar partes y mostrar conceptos básicos de robótica, pues ofrece una limitada variedad de movimientos, pero suficiente para realizar acciones como tomar y mover objetos en trayectorias simples [58].

Figura 40: Brazo robótico de 3DOF [58].

En la siguiente tabla se detallan las características técnicas del brazo robótico, enfatizando sus características más relevantes.

Tabla 7: Características técnicas del brazo robótico de 3DOF

CARACTERÍSTICAS TÉCNICAS	
Grados de libertad	3 grados
Motor	Servomotor
Controladores	Raspberry, Arduino
Velocidad	0.20seg/ 60°6v
Voltaje	4.8V-6V

2.2.1.6. Placa de Expansión Transbot

La placa de expansión de Transbot es un componente adicional que permite ampliar las capacidades del robot Transbot, proporcionando puertos y conexiones adicionales para integrar diferentes sensores, actuadores y módulos (véase la figura 41). Esta placa facilita la conexión y control de dispositivos externos, como cámaras, sensores ultrasónicos, motores adicionales, servos, y más, ofreciendo mayor flexibilidad para proyectos avanzados en robótica.

Incluye interfaces de comunicación como I2C, UART, SPI y GPIO, permitiendo la integración con plataformas como Raspberry Pi o Jetson Nano, además soporta control y programación a través de sistemas como ROS (Robot Operating System) para realizar funciones complejas, como navegación autónoma, manipulación de objetos o procesamiento de imágenes [62].

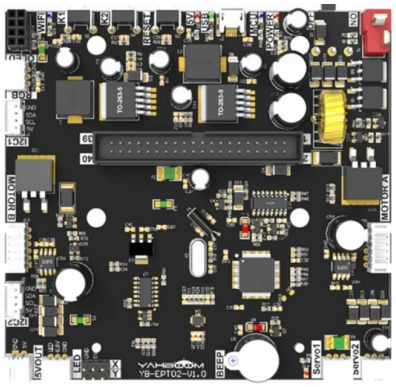


Figura 41: Placa de expansión Transbot [62].

Se presenta una tabla con las características técnicas de la placa de expansión, destacando sus características esenciales.

Tabla 8: Características técnicas de la place de expansión Transbot

CARACTERÍSTICAS TÉCNICAS	
Potencia de salida	5V DC
Chip de comunicación	Chip serie CH340C
Fuente de alimentación	12V DC
Interfaz de comunicación	Micro USB a USB

2.2.1.7. Batería de Litio

Las baterías de litio son baterías recargables ampliamente utilizadas en dispositivos electrónicos y robóticos debido a su alta densidad energética, bajo peso y larga vida útil (véase la figura 42). Estas baterías ofrecen varias ventajas, como una mayor capacidad de almacenamiento de energía en comparación con otros tipos de baterías (por ejemplo, de plomo-ácido o níquel-cadmio), lo que permite que los dispositivos funcionen durante más tiempo entre cargas. También tienen un bajo índice de autodescarga, lo que significa que retienen su carga durante largos períodos cuando no están en uso [58].

Figura 42: Batería de litio [58].

A continuación, se incluye una tabla con las características técnicas de batería, resaltando sus aspectos más importes.

Tabla 9: Características técnicas de la batería de litio

CARACTERÍSTICAS TÉCNICAS	
Capacidad	4400mAh
Peso	300g
Fuente de alimentación	12.6V 2A
Protección	Protección contra sobrecorriente

2.2.1.8. Módulo GPS UBLOX NEO 6M

El módulo GPS Unbox/u-blox NEO-6M es una solución avanzada de navegación satelital que proporciona una excelente combinación de precisión y flexibilidad para proyectos de control de vuelo y localización (véase la figura 43). Este módulo ha sido diseñado específicamente para ser compatible con APM2 y APM2.5, lo que lo convierte en una opción excelente para una variedad de aplicaciones aeroespaciales y de navegación. Su antena cerámica de alta calidad garantiza una recepción de señal optimizada, permitiendo obtener datos de ubicación precisos y confiables, incluso en condiciones difíciles [63].

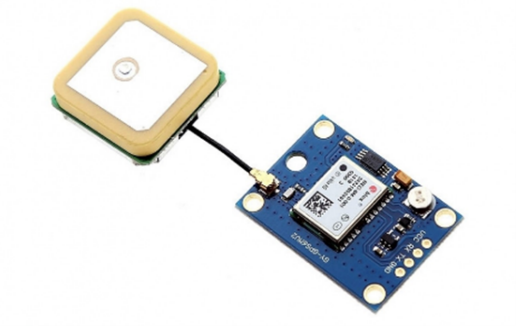


Figura 43: Modulo GPS [63].

A continuación, se presenta una tabla con las características técnicas del módulo GPS, destacando sus características principales.

Tabla 10: Características técnicas del módulo GPS

CARACTERÍSTICAS TÉCNICAS	
Interfaz	Serial UART 5V
Frecuencia de refresco	5 Hz
Fuente de alimentación	3-5 VDC
Antena	Cerámica

2.2.1.9. Sensor ultrasónico

Un sensor ultrasónico es un dispositivo electrónico que utiliza ondas sonoras de alta frecuencia para identificar objetos cercanos (véase la figura 44). Funciona enviando pulsos ultrasónicos y midiendo el tiempo que tarda el eco en regresar tras reflejarse en un objeto, lo que permite calcular la distancia [64].

Figura 44: Sensor ultrasónico [64].

A continuación se presenta una tabla con las características técnicas del sensor ultrasónico, destacando sus características esenciales.

Tabla 11: Características técnicas del sensor ultrasónico

CARACTERÍSTICAS TÉCNICAS	
Alimentación	5V Dc
Ángulo efectivo	15°
Distancia	2cm a 400cm
Consumo	15mA

2.2.2. Componentes Lógicos

A continuación, se mencionan los softwares utilizados para programar la parte física de esta propuesta de investigación. Estos componentes son esenciales para el desarrollo y el correcto funcionamiento de la implementación, ya que facilitan la coordinación y la gestión de las actividades realizadas.

2.2.2.1. Raspberry Pi Imager

Raspberry Pi Imager es una herramienta oficial desarrollada por la Fundación Raspberry Pi para simplificar el proceso de escritura de una imagen del sistema operativo en una tarjeta SD o unidad USB Raspberry Pi. El programa se encarga de descargar, formatear e instalar automáticamente el sistema operativo en la tarjeta SD, facilitando el proceso de configuración [65]. Con esta herramienta es posible instalar diversos sistemas operativos compatibles con Raspberry Pi 5. Para llevar a cabo este proyecto, instalaremos una versión que disponga de las librerías y requisitos imprescindibles para nuestro sistema, en este caso se utilizó el sistema operativo de Rasberry Pi Os (ver Anexo A).

Figura 45: Software Raspberry Pi Imanager [Autoría Propia].

2.2.2.2. Advanced Ip Scanner

Advanced IP Scanner es una herramienta potente y versátil para el escaneo

de redes, diseñada para mejorar la identificación y gestión de dispositivos. La herramienta proporciona a los usuarios una forma eficiente de analizar, monitorear y obtener detalles sobre la configuración de su red [66]. Este programa será esencial porque, al conectar nuestra Raspberry Pi 5 a la red, el módem le asignará una dirección IP única que será necesaria para configurar el control remoto, para utilizar este programa, simplemente accedemos al ícono de IP ubicado en la parte superior, el cual buscará y mostrará todos los dispositivos conectados a la red.

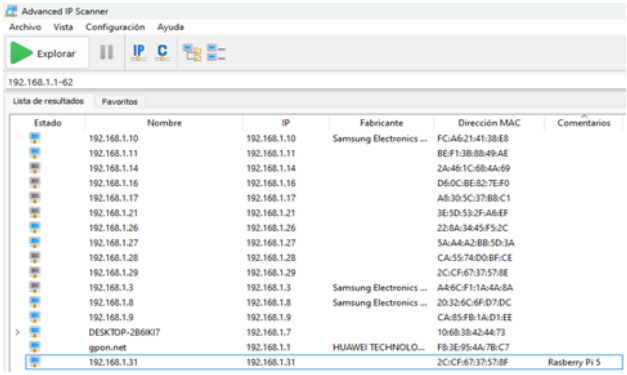


Figura 46: Software Advanced IP Scanner [Autoría Propia].

2.2.2.3. Vnc Viewer

VNC Viewer es una aplicación que permite ingresar y gestionar de manera remota otros dispositivos a través del protocolo VNC (Virtual Network Computing). Con VNC Viewer, puedes ver el escritorio y manejar el equipo remoto como si estuvieras frente a él, usando cualquier dispositivo con conexión a Internet [67]. En este caso, se obtiene acceso a la Raspberry Pi 5 mediante la dirección IP encontrada anteriormente con el programa de IP Scanner. Se ingresa la dirección IP en el navegador y al presionar “Enter”, se muestra automáticamente la pantalla inicial de la Raspberry Pi, para llevar a cabo las diversas configuraciones iniciales del dispositivo (Ver anexo B).

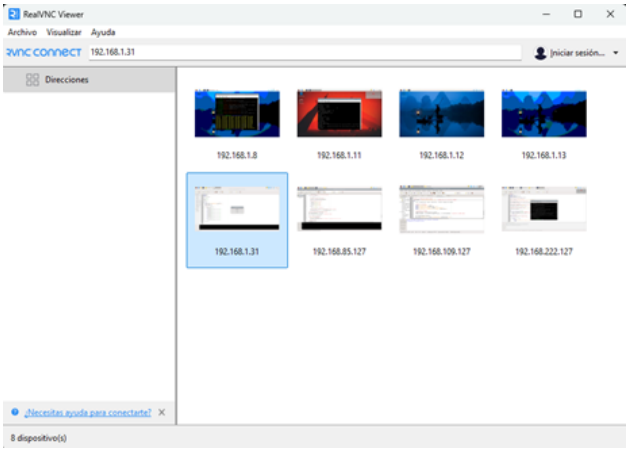


Figura 47: Software VNC Viewer [Autoría Propia].

2.2.2.4. Python

Python es un lenguaje de programación de nivel avanzado, de uso generalizado. Es conocido por su sintaxis clara y sencilla, lo que lo hace accesible tanto para principiantes como para desarrolladores experimentados, soporta múltiples paradigmas de programación, incluyendo programación orientada a objetos, imperativa y funcional [68]. Para utilizar Python, se utilizó el editor de texto Geany, que es una herramienta ligera y versátil diseñada para facilitar la edición del código.

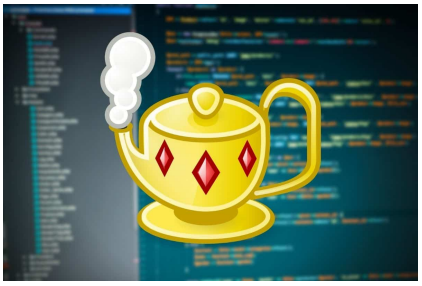


Figura 48: Lenguaje de programación Python en Geany [68].

A continuación se presentan las principales librerías empleadas en python para desarrollar la propuesta:

Tensorflow: Es una biblioteca de código abierto desarrollada por Google para el aprendizaje automático y la inteligencia artificial. Permite a los desarrolladores crear y entrenar modelos de aprendizaje profundo, que son particularmente útiles para tareas como el reconocimiento de imágenes.

Opencv: Es una biblioteca de código abierto para el procesamiento de imágenes y la visión por computadora, que permite a los desarrolladores realizar tareas como detección y reconocimiento de gestos, procesamiento de imágenes, seguimiento de video en movimiento, y reconocimiento facial.

Tkinter: es la herramienta estándar de Python, permite a los desarrolladores construir aplicaciones de escritorio con interfaces gráficas personalizadas. Esta librería facilita la organización y el diseño visual de los elementos de la interfaz, proporcionando una base sólida para proyectos de desarrollo de software.

Torch: Es una biblioteca de Python diseñada para el desarrollo de modelos de aprendizaje profundo, utilizada para el desarrollo de modelos de inteligencia artificial.

2.3. Diseño experimental

Para el diseño del sistema de supervisión, se seleccionó el robot móvil terrestre Transbot SE, este robot, especializado en visión artificial, destaca por su robustez y movilidad, lo que le permite desplazarse ágilmente en diversos entornos. La cámara integrada permite una captura precisa de imágenes, facilitando la detección de eventos y el análisis de datos visuales, proporcionando una monitorización detallada y en tiempo real del área de operación, para la detección personas y obstáculos se implementó una red neuronal convolucional diseñada para procesar las imágenes captadas por la cámara del robot. Este enfoque aprovecha las capacidades del Transbot SE, específicamente su sistema de procesamiento, para ejecutar modelos de aprendizaje profundo en tiempo real. Las redes neuronales podrán identificar objetos y personas en el entorno, distinguirlos y generar una alerta o una respuesta automatizada según sea necesario, en la figura 49 se muestra el flujo de procesamiento del sistema de visión artificial integrado en el robot móvil Transbot. Una cámara a bordo del robot captura imágenes del entorno, las cuales son alimentadas a la red neuronal convolucional. Esta red, especializada en tareas de visión por computador, segmenta la imagen y clasifica los objetos de interés. La información extraída de la red es transmitida al sistema de supervisión, donde se ejecuta un conjunto de algoritmos para la toma de decisiones en tiempo real, basándose en la información visual procesada, el robot puede detectar personas u obstáculos, evitar colisiones o explorar un área desconocida.

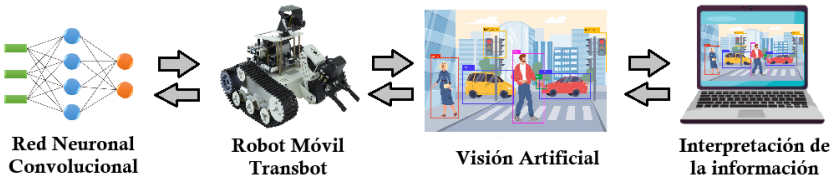


Figura 49: Flujo de trabajo del sistema de supervisión [Autoría Propia].

A continuación, se presentan las especificaciones del sistema, divididas en componentes de hardware y software para un análisis más detallado.

2.3.1. Diseño e Implementación del Hardware

El hardware juega un papel vital en el rendimiento y funcionamiento del sistema, ya que proporciona la base para las tareas de procesamiento, comunicación y control. Para comprender la estructura física del sistema propuesto, se presenta a continuación una descripción detallada de los componentes electrónicos que lo integran y su disposición en el sistema.

Como se puede observar en el diagrama esquemático mostrado en la figura 50, en la parte superior, se observa la Raspberry Pi, junto a un ventilador para disipar el calor generado durante su operación, sobre la Raspberry Pi se encuentra el módulo GPS, cuya ubicación elevada facilita la recepción óptima de las señales satelitales, indispensables para determinar la posición geográfica del dispositivo.

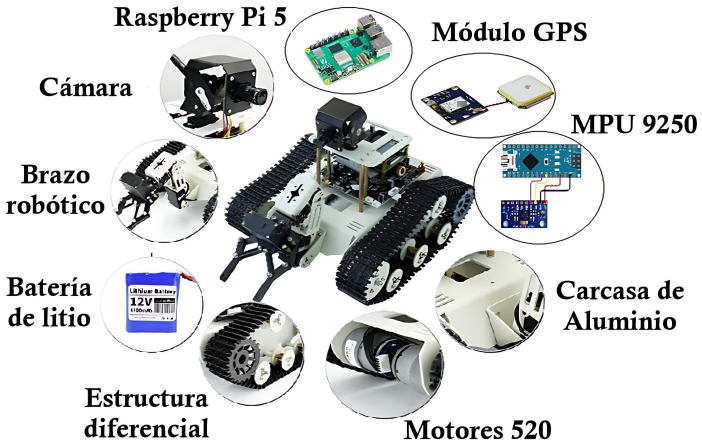


Figura 50: Esquema del robot Transbot [Autoría Propia].

En la parte frontal, se encuentra ubicada la cámara encargada de visualizar el entorno, proporcionando al robot una percepción precisa de su alre-

dedor. Adicionalmente, en esta misma sección se integra el brazo robótico, diseñado para realizar tareas de manipulación.También en la parte frontal, así como en los laterales, se encuentran sensores ultrasónicos, cuya función es detectar obstáculos y evitar colisiones durante el desplazamiento del robot. En la parte inferior, se ubican los motores que impulsan el movimiento del robot. Tambien se encuentra un arduino nano junto al mpu-9250 que se encargara de la orientación del robot, así como la batería que suministra la energía necesaria para su funcionamiento. En la parte superior se ubica el módulo GPS, acompañado de su convertidor, el cual permite captar las señales provenientes del satélite el cual nos permite conocer la ubicación actual del robot.

En la Figura 51 se muestran todos los componentes mencionados previamente ya ubicados en sus respectivas posiciones dentro del robot. Esta representación permite visualizar de forma clara la distribución física de los elementos, facilitando la comprensión de su organización y funcionamiento general.

Figura 51: Distribución del los elemementos en el robot [Autoría Propia].

2.3.1.1. Conexiones de la Placa de Expansión

La placa de expansión Transbot SE integra un microcontrolador de un solo chip, que es la principal responsable de controlar los controladores de los componentes de la placa, esto permite que la Raspberry Pi se comunique de manera eficiente con otros componentes periféricos. Esto incluye el brazo servomecánico de 3 grados de libertad, los motores, los servomotores para el movimiento de la cámara y la cámara PTZ, la placa de expansión incluye puertos y pines adicionales que permiten conectar una variedad de sensores y actuadores sin necesidad de adaptadores adicionales. En la figura 52 se muestra la distribución y características de esta placa.

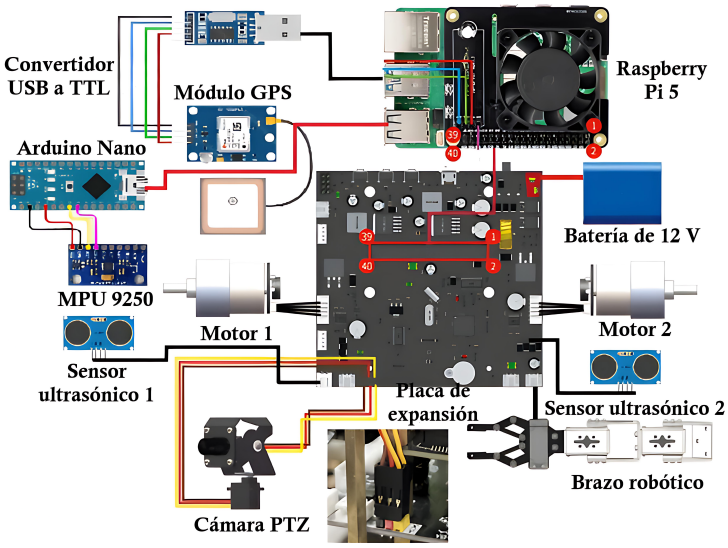


Figura 52: Conexión de los componentes con la placa de expansión [Autoría Propia].

A continuación, se presenta una tabla con las conexiones de los diferentes pines a la placa de expansión y Raspberry pi, destacando sus funciones principales.

Tabla 12: Dispositivos conectados a la placa de expansión

Componente	**Pines de conexión**	**Función**
Raspberry Pi	GPIO (cable plano)	Comunicación y energía con la placa de expansión
MPU-9250	USB	Medición de orientación
Motor izquierdo	Motor A	Dirección y velocidad
Motor derecho	Motor B	Dirección y velocidad
Cámara PTZ	USB	Transmisión de video
Batería	Fuente de alimentación	Energía para el sistema
Brazo robótico	Servo 1	Control de brazo y pinza
Módulo GPS	USB	Geolocalización
Arduino Nano	USB	Control de periféricos

La placa de expansión en el TRANSBOT-SE está diseñada para ampliar las capacidades de la placa de control principal a través de un cable plano, se asegura una conexión eficiente y directa entre la placa de expansión y la Raspberry Pi (véase la figura 53), facilitando tanto la comunicación de datos como el suministro de energía, el cable plano facilita la transferencia de señales y datos a altas velocidades, lo cual es esencial para el control en tiempo real del robot, especialmente en aplicaciones como el reconocimiento de objetos y la conducción autónoma.

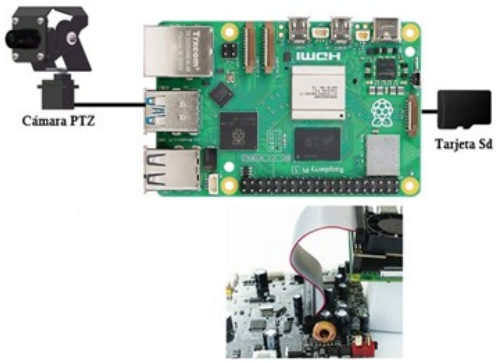


Figura 53: Conexión de la Raspberry con la placa de expansión [Autoría Propia].

A continuación, se detalla una tabla con las conexiones de los pines de la Raspberry Pi con la placa de expansión, que permiten la interacción entre los distintos módulos y componentes del sistema. La Raspberry Pi 5, como placa de control principal, cuenta con un conjunto de pines GPIO (General Purpose Input/Output) que permiten la conexión directa con la placa de expansión Transbot-SE. Estos pines se utilizan para controlar diversos dispositivos, como motores, servomotores y otros componentes electrónicos del robot.

Tabla 13: Pines conectados a la placa de expansión

GPIO	**Placa Expansión**	**Elemento**	**Función**
GPIO2 (SDA)	SDA (I2C)	Cámara PTZ, USB-TTL	I2C
GPIO3 (SCL)	SCL (I2C)	Cámara PTZ, USB-TTL	I2C
GPIO14 (TXD)	TX (UART)	Comunicación serial	UART
GPIO15 (RXD)	RX (UART)	Comunicación serial	UART
GPIO18	PWM	Motor izquierdo	Dirección/velocidad
GPIO19	PWM	Motor derecho	Dirección/velocidad
GPIO12	PWM	Servo brazo 1	Articulación 1
GPIO13	PWM	Servo brazo 2	Articulación 2
GPIO20	PWM	Servo brazo 3	Articulación 3
GPIO21	PWM	Servo cámara (Pan)	Giro horizontal
GPIO22	PWM	Servo cámara (Tilt)	Giro vertical
3.5/5V	Power	Sensores/módulos	Energía
GND	Tierra	Todos los elementos	Masa común

2.3.2. Diseño e Implementación del Software

El software es un componente esencial que da vida al hardware, permitiendo tareas e interacciones específicas entre diferentes módulos del sistema. El software utiliza algoritmos y programas diseñados para controlar el robot para gestionar el procesamiento de datos, la comunicación entre dispositivos y la toma de decisiones en tiempo real para garantizar un funcionamiento eficiente y preciso del sistema.

2.3.2.1. Librería Transbot

Para la instalación de esta librería, indispensable para el funcionamiento del robot, es necesario primero crear un entorno virtual, donde se almacenarán todas las librerías que utilizaremos. Una vez creado el entorno, descargamos el archivo directamente desde la página de yahboom, una vez descargado, lo descomprimimos y procederemos con su instalación. La librería Transbot está diseñada para controlar el hardware del Transbot-se.

La librería proporciona las herramientas necesarias para controlar el movimiento del robot, incluidos los motores, el brazo robótico y la cámara. Esto permite una integración fluida entre el software y el hardware del robot.

2.3.2.2. Entrenamiento y Clasificación de las Imágenes

El entrenamiento de imágenes se basa en el uso de una red neuronal convolucional para clasificar imágenes. En primer lugar, las imágenes se redimensionan y normalizan. Luego, se alimentan a la red, la cual ajusta sus parámetros a lo largo de varias rondas de aprendizaje (épocas). Durante este proceso, la red aprende a distinguir entre dos clases, como “personas” u “obstáculos”, mejorando progresivamente su capacidad para hacer predicciones precisas sobre nuevas imágenes.

En la Figura 54 se muestran ejemplos de las imágenes utilizadas durante el entrenamiento, correspondientes a las clases “personas” y “obstáculos”. Estas imágenes, ya almacenadas en el sistema de archivos internos de la Raspberry Pi, fueron previamente organizadas para facilitar su uso en el proceso de clasificación.

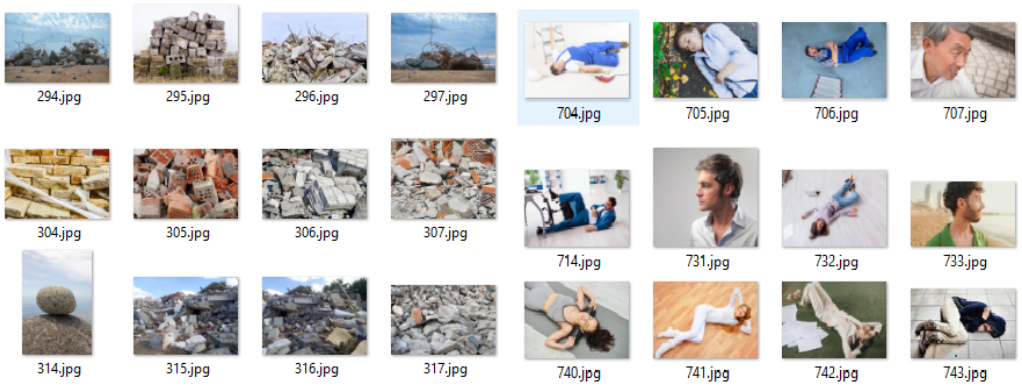


Figura 54: Imágenes usadas en el entranamiento [Autoría Propia].

Para el etiquetado de las clases se utilizo la plataforma makesense.ai como se observa en la figura 55, para realizar la etiquetación manual de imágenes. Esta herramienta en línea facilita la anotación precisa y eficiente de objetos dentro de imágenes, permitiendo generar datos etiquetados que serán fundamentales para entrenar el modelo.

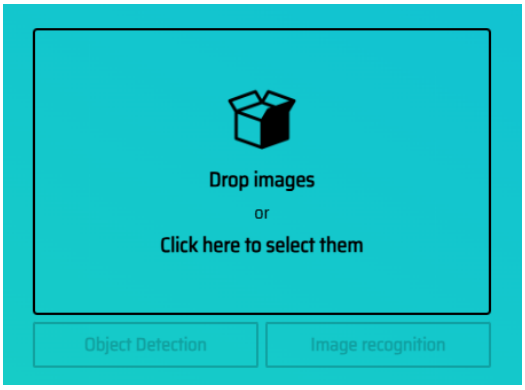


Figura 55: Makesense detección de objetos [Autoría Propia].

Como se observa en la imagen 56 se etiquetaron dos clases: personas y obstáculos. Estas categorías permitirán identificar y clasificar los elementos clave dentro de las imágenes para su posterior uso.

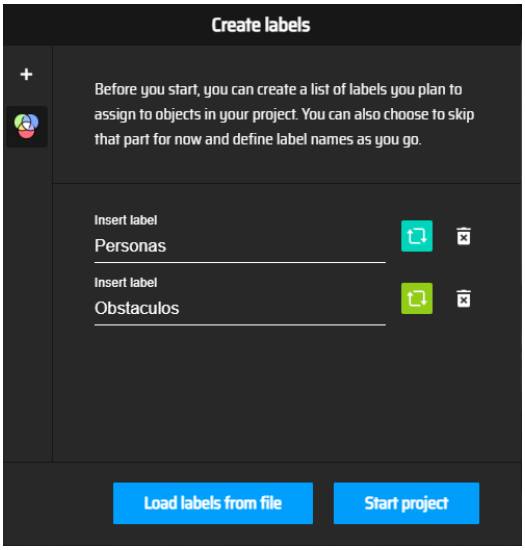


Figura 56: Clases de Objetos del Dataset [Autoría Propia].

Mediante la herramienta de rectángulo de makesense.ai, como se observa en la figura 57 se selecciono de manera manual las áreas de interés en cada imagen y asignándolas a una de las dos clases definidas: personas o obstáculos. Este proceso permite generar anotaciones precisas que servirán como base para el entrenamiento.

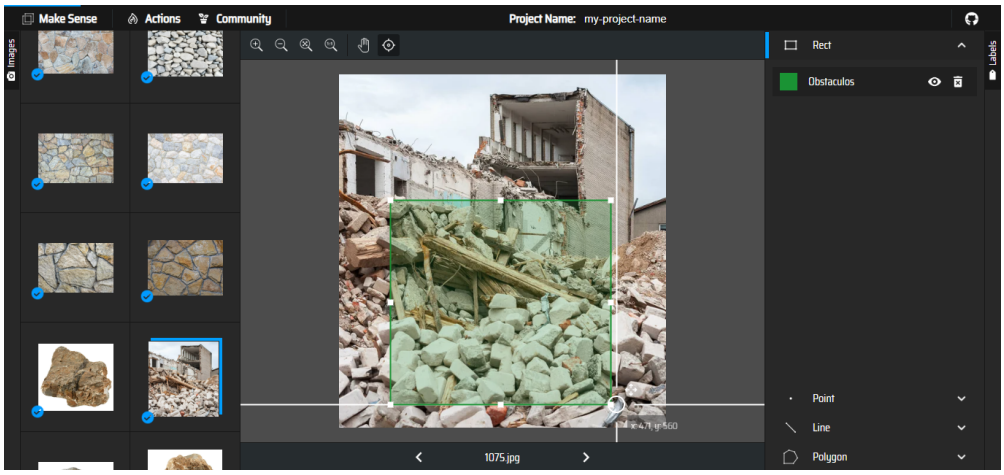


Figura 57: Etiquetado de objetos [Autoría Propia].

Se verifica que las imágenes están correctamente almacenadas al comprobar visualmente la presencia de los archivos correspondientes en cada una de las carpetas. Esto confirma que los datos se encuentran organizados de

manera adecuada para su posterior uso en el entrenamiento del modelo. Una vez finalizado el entrenamiento de la red neuronal, el modelo resultante se guarda en un archivo en formato Keras, lo cual permite su reutilización sin necesidad de volver a entrenarlo. Este modelo es capaz de distinguir entre personas y obstáculos, y fue utilizado para realizar predicciones en tiempo real, analizando las imágenes capturadas por la cámara del robot.

2.3.2.3. Análisis de Modelos de Redes Neuronales Convolucionales

En primer lugar, se instaló la librería de TensorFlow utilizando el siguiente comando "pip install tensorflow", una vez ingresado el comando, comenzará la descarga de todos los archivos y la creación de las dependencias necesarias. La red neuronal está compuesta por un total de n capas convolucionales seguidas de capas de pooling que ayudan a reducir la dimensionalidad y extraer características relevantes de las imágenes, el modelo se entrenó durante 200 épocas, lo que significa que el conjunto de datos de entrenamiento se utiliza para ajustar los pesos de la red 200 veces, este proceso permitió que el modelo aprenda a identificar patrones y características en los datos mejorando su capacidad para clasificar imágenes. Como se observa en la figura 58 la red neuronal procesa la imagen, identifica patrones relevantes y determina si contiene elementos de interés, como personas u obstáculos.

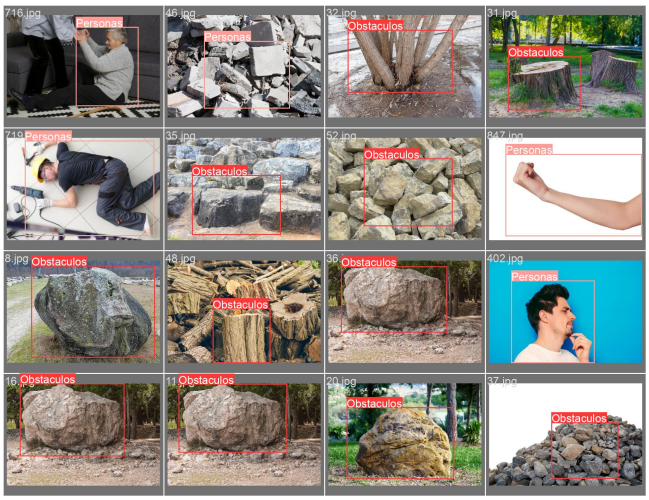


Figura 58: Reconocimiento de imágenes mediante la red neuronal [Autoría Propia].

Se llevaron a cabo pruebas utilizando tres redes neuronales diferentes, cada una con características específicas y técnicas de clasificación de imágenes

destinadas a detectar “personas” u “obstáculos”, cada una de estas redes utiliza arquitecturas optimizadas para diferentes aplicaciones y requisitos de hardware, lo que demuestra su complejidad y eficiencia en las tareas de procesamiento visual. Los modelos evaluados son descritos a continuación.

Red Neuronal Convolucional Estándar: Como se muestra en la figura 59, una red neuronal convolucional es una arquitectura de aprendizaje profundo diseñada específicamente para trabajar con datos visuales, inspirándose en la estructura del cerebro humano, las redes neuronales son capaces de extraer características de imágenes de manera jerárquica. Esto significa que comienzan identificando rasgos simples como bordes y texturas, y gradualmente aprenden a reconocer patrones más complejos como caras, objetos y escenas completas.

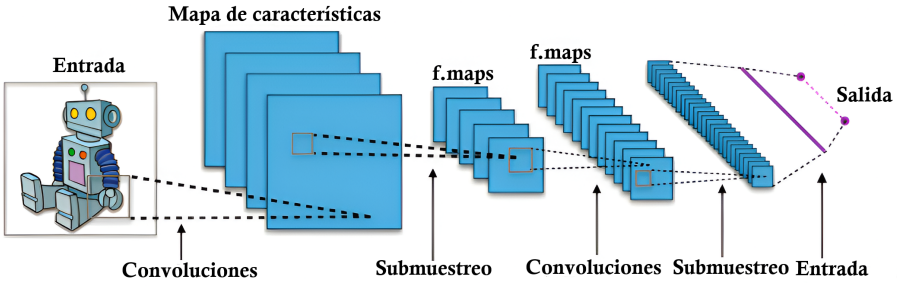


Figura 59: Arquitectura - Red neuronal [Autoría Propia].

La red neuronal convolucional estándar implementada en Python utiliza las bibliotecas OpenCV y TensorFlow/Keras, y su objetivo principal es clasificar imágenes capturadas por la cámara de la Raspberry Pi en dos categorías: “Personas” u “Obstáculos”. El sistema comienza con la captura de una imagen, la cual es preprocesada (redimensionada, normalizada y ajustada al formato RGB) para que sea compatible con la entrada del modelo. La arquitectura está compuesta por tres capas convolucionales con 32, 64 y 128 filtros respectivamente, cada una con kernels de 3x3 y función de activación ReLU, seguidas por capas de max-pooling de 2x2 que reducen la dimensionalidad. Posteriormente, la salida se aplana mediante una capa Flatten y se conecta a una capa densa de 128 neuronas con activación ReLU, finalizando con una capa de salida de una neurona con activación sigmoide, adecuada para clasificación binaria. El entrenamiento del modelo se realizó con un conjunto de 4200 imágenes etiquetadas, utilizando el optimizador Adam, función de pérdida binary crossentropy, una tasa de aprendizaje de 0.001, tamaño de batch de 32 y un total de 200 épocas. Finalizado el entrenamiento, una vez terminado su entramiento, el modelo se guarda para su uso futuro.

Red Neuronal Convolucional Basada en MobileNet: Como se muestra en la figura 60, una red neuronal basada en mobilenet es una arquitectura ligera y eficiente desarrollada para dispositivos con recursos limitados, utiliza Depthwise Separable Convolutions (convoluciones separables en profundidad), lo cual reduce significativamente el número de parámetros y cálculos al dividir la convolución en dos partes: una convolución de profundidad y una convolución de punto. Además, MobileNet emplea técnicas como la normalización por lotes (Batch Normalization) y una función de activación ReLU6, optimizando su rendimiento en dispositivos con baja capacidad de procesamiento.

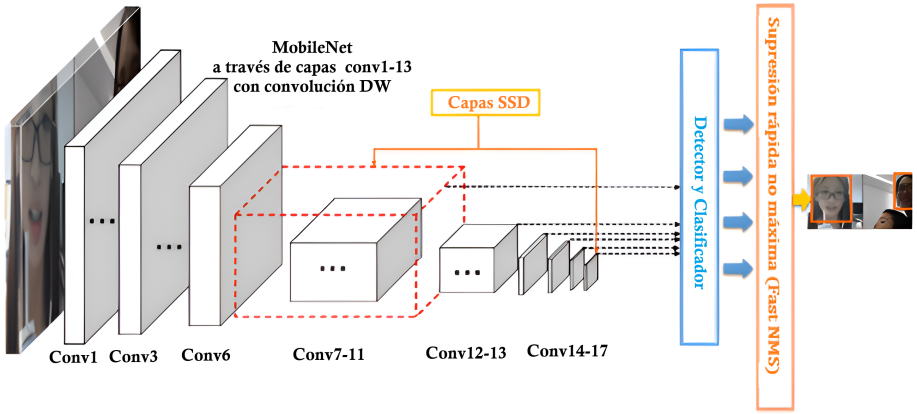


Figura 60: Arquitectura - red neuronal basada en mobilenet [Autoría Propia].

La red neuronal convolucional basada en MobileNet fue implementada en Python mediante el uso de TensorFlow/Keras y se emplea para realizar una clasificación binaria entre "Personas" u "Obstáculos", utilizando la técnica de aprendizaje por transferencia (transfer learning) con el modelo MobileNet preentrenado en ImageNet, lo que le permite reconocer una amplia variedad de patrones visuales sin necesidad de entrenar desde cero. Las imágenes capturadas por la cámara del robot son preprocesadas (redimensionadas, normalizadas y ajustadas al formato de entrada de MobileNet) y sometidas a técnicas de aumento de datos (data augmentation) como rotación, desplazamiento y escalado, con el objetivo de mejorar la capacidad de generalización del modelo. Durante el proceso de entrenamiento, se congelan las capas convolucionales de MobileNet para preservar los pesos aprendidos, y se agregan nuevas capas densas totalmente conectadas con función de activación ReLU y regularización mediante Dropout para reducir el riesgo de sobreajuste. La capa de salida es una sola neurona con activación sigmoide, adecuada para la

clasificación binaria. Se utiliza el optimizador Adam, una tasa de aprendizaje de 0.0001, función de pérdida binary crossentropy, tamaño de batch de 32 y un total de 200 épocas, junto con la técnica de EarlyStopping para detener automáticamente el entrenamiento si no se observa mejora en la precisión. Una vez finalizado, el modelo entrenado se guarda para su posterior uso.

Red Neuronal Convolucional Basada en YoloV5: Como se muestra en la figura 61, YOLOv5 es un modelo de detección de objetos en tiempo real que utiliza una arquitectura eficiente basada en CSPDarknet para extraer características visuales, junto con FPN y PANet para combinar información de diferentes niveles de la imagen. En este entramiento, se adaptó YOLOv5 a una tarea binaria de detección de "Personas" y "Obstáculos" mediante aprendizaje por transferencia, utilizando pesos preentrenados en el conjunto COCO y entrenando con un nuevo conjunto de imágenes etiquetadas. Se aplicaron técnicas de aumento de datos (rotación, escalado, cambios de brillo) para mejorar la capacidad de generalización del modelo. Una vez entrenado, el modelo fue exportado a un formato best.pt, lo que permite al robot analizar imágenes capturadas por la cámara en tiempo real, detectar personas u obstáculos con alta precisión y generar respuestas rápidas y eficientes.

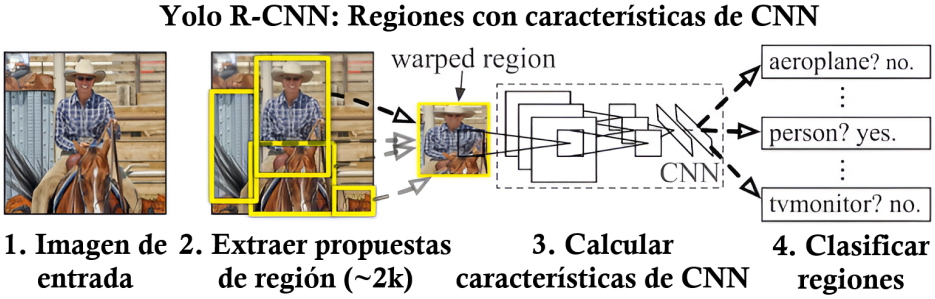


Figura 61: Arquitectura - red neuronal basada en Yolov5 [Autoría Propia].

Descripción del Modelo y Proceso de Entrenamiento: El modelo implementado para la detección de objetos corresponde a la arquitectura **YOLOv5s**, una red neuronal convolucional profunda optimizada para tareas de visión artificial en tiempo real. Este modelo fue entrenado específicamente para la detección de dos clases: *personas* y *obstáculos*, utilizando un conjunto de datos personalizado. El entrenamiento se llevó a cabo en Google Colab, empleando una GPU para acelerar el proceso. Se utilizó el modelo preentrenado `yolov5s.pt` como punto de partida, realizando una adaptación mediante *transfer learning*. El archivo `Tesis.yaml` definió las rutas al conjunto de datos, así como el número de clases y sus respectivas etiquetas.

El proceso de entrenamiento se realizó con las siguientes configuraciones:

Tabla 14: Parámetros de entrenamiento del modelo YOLOv5s

Parámetro	**Valor**
Modelo base	YOLOv5s
Épocas	200
Tamaño de imagen	640 × 640 píxeles
Tamaño del batch	16
Pesos iniciales	`yolov5s.pt`
Número de clases	2 (personas y obstáculos)
Aumento de datos	Activado (por defecto)
Función de activación	SiLU (no se usó ReLU)
Optimización	Adam / SGD (según configuración)

La arquitectura de YOLOv5s se divide en tres bloques principales: el **backbone**, el **neck** y la **cabeza de detección**. Esta versión no utiliza funciones de activación ReLU, sino que emplea mayoritariamente la función `SiLU (Sigmoid Linear Unit)`, la cual mejora la estabilidad durante el entrenamiento.

Tabla 15: Componentes de la arquitectura YOLOv5s

Módulo	**Función**	**Características**
Backbone	Extracción de características	CSPDarknet: Conv + BN + SiLU
Neck	Fusión multiescala	PANet: capas profundas y superficiales
Head	Predicción final	Detección en 3 escalas (P3, P4, P5)

Aunque YOLOv5s no está compuesto por capas densas tradicionales con neuronas explícitas, se puede describir la complejidad del modelo en términos de parámetros entrenables y capas convolucionales. A continuación, se describe esta información:

Tabla 16: Descripción técnica del modelo YOLOv5s

Aspecto	**Valor aproximado**
Total de parámetros entrenables	$\sim$ 7.2 millones
Capas convolucionales	Más de 220
Capas de activación SiLU	En casi todas las capas
Capas ReLU utilizadas	0
Salidas por celda de detección	7 (4 bbox + 1 conf + 2 clases)

2.3.3. Implementación del Sistema de Supervisión

La implementación del sistema de supervisión para un robot móvil terrestre consiste en desarrollar una plataforma que permite la monitorización y el

control en tiempo real del robot. Este sistema asegura que el robot opere de manera eficiente y segura, proporcionando una interfaz gráfica para visualizar su estado y progreso. La plataforma también facilita la toma de decisiones basadas en los datos del robot.

2.3.3.1. Diagrama de flujo del Sistema de Supervisión

El sistema de supervisión permite al robot capturar imágenes del entorno mediante la cámara, las cuales son procesadas por la red neuronal convolucional entrenada para identificar personas y reconocer obstáculos. El funcionamiento detallado de este proceso puede observarse en el diagrama de flujo mostrado en la figura 62.

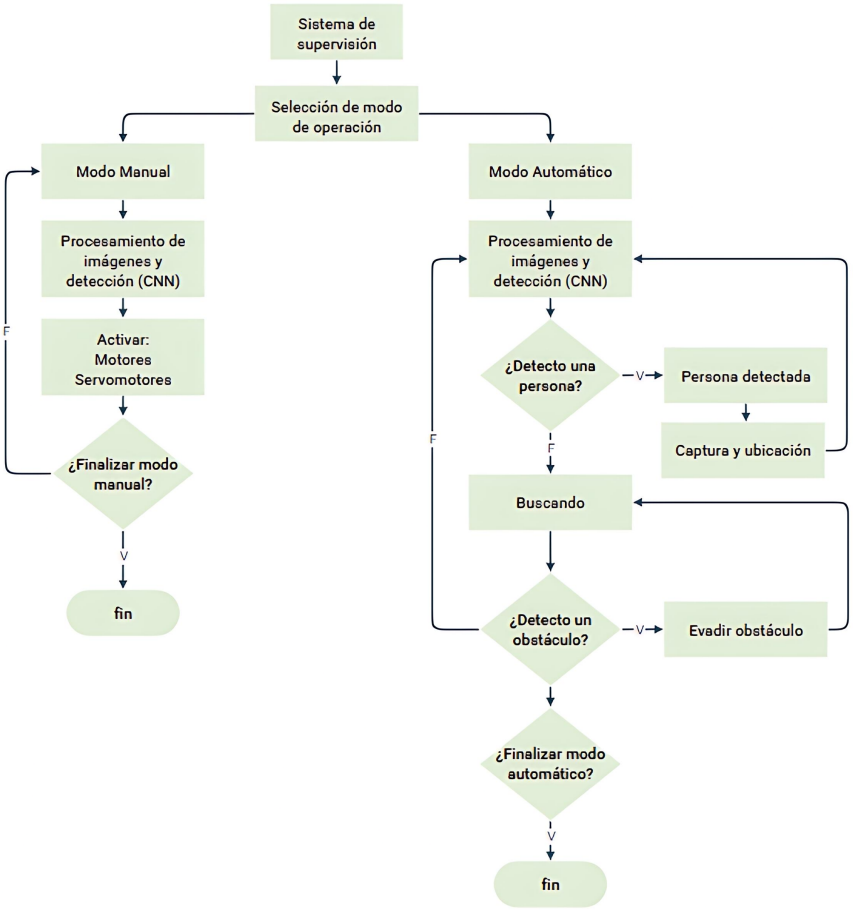


Figura 62: Diagrama de flujo de la propuesta [Autoría Propia].

El sistema inicia solicitando al usuario seleccionar entre los dos modos de operación: manual o automático. En el modo manual, el operador tiene

control total del robot, dirigiendo su movimiento y tomando decisiones en tiempo real. En este caso, el sistema funciona como una interfaz de control remoto, donde el usuario gestiona todos los aspectos del robot. En el modo automático, el robot inicia su búsqueda de manera autónoma, primero se selecciona la ubicación de destino especificando su latitud y longitud, lo que permite que el robot calcule una nueva ruta y ajuste automáticamente su dirección para desplazarse hacia ese punto, el sistema captura imágenes en tiempo real utilizando la cámara, estas imágenes son procesadas por la red neuronal, durante el proceso de búsqueda, si el robot detecta la presencia de una persona, toma una captura de la imagen y registra su ubicación geográfica. Estas imágenes, junto con sus coordenadas, se almacenan en una carpeta específica y pueden ser visualizadas por los usuarios a través de un servidor web, facilitando el acceso remoto a la información en tiempo real, en caso de encontrar un obstáculo, el robot activa un algoritmo de evasión que le permite modificar su trayectoria para evitar colisiones y continuar con su misión de búsqueda.

2.3.3.2. Plataforma de Interacción Usuario-Sistema

Para la implementación de la plataforma de interacción se utilizó la biblioteca tkinter, la cual podemos instalar mediante el siguiente comando "sudo apt-get install python3-tk", como se observa en la figura 63 esta nos permite la creación de la interfaz gráfica en el entorno de Python, esta librería nos facilita la creación de ventanas, botones, cuadro de texto, menús despegables, etiquetas entre otros elementos visuales.

Figura 63: Menú de opciones del sistema propuesto [Autoría Propia].

En el desarrollo de este proyecto, es fundamental contar con diferentes

modos de operación que se adapten a las necesidades del usuario y a las tareas específicas que se deben realizar. En el caso del robot Transbot SE, se ha implementado dos modos de operación: Modo Manual y Modo Automático. Estos modos permiten una interacción eficiente con el robot, brindando al usuario la opción de elegir cómo desea controlarlo.

Modo Manual: Está diseñado para permitir al usuario un control directo y preciso del robot, al seleccionar este modo, el usuario activa un programa que permite la manipulación del robot a través de comandos manuales donde podemos controlar el movimiento del robot, el brazo robótico, la posición de la cámara y la velocidad a la que se desplaza el robot. El funcionamiento de este modo se detalla en el diagrama de flujo mostrado en la figura 64).

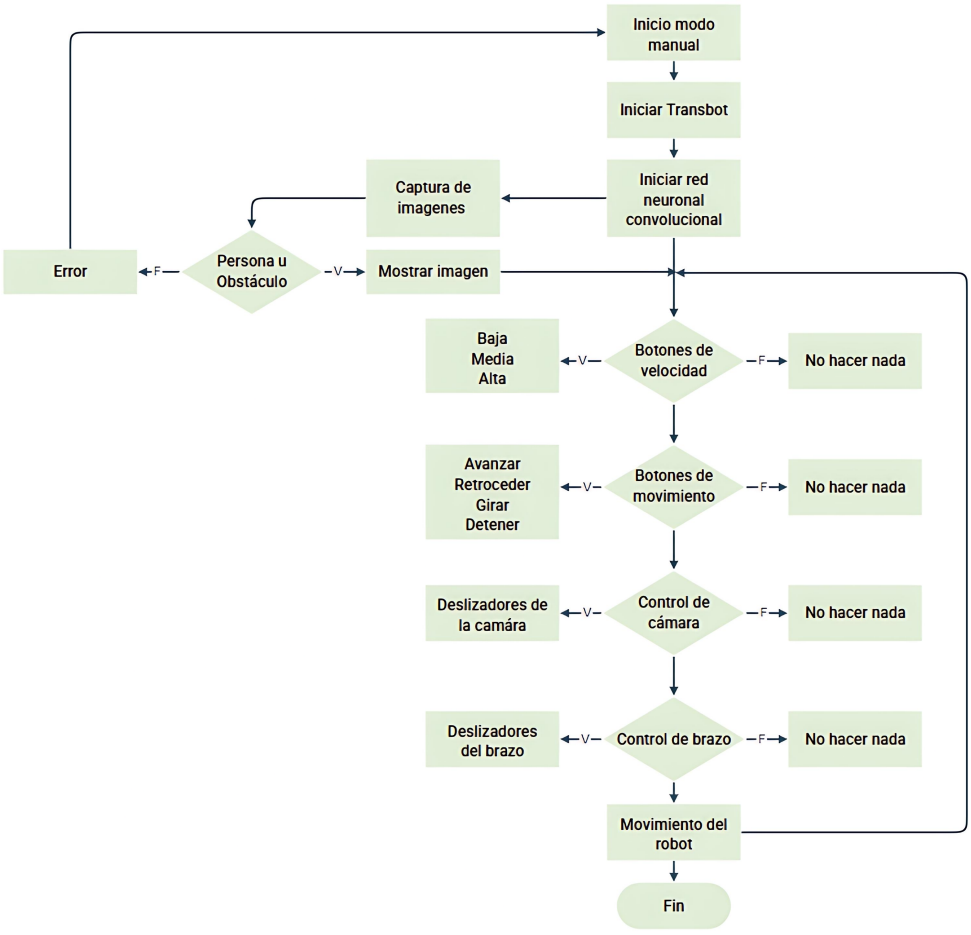


Figura 64: Diagrama de flujo del modo manual [Autoría Propia].

Se utilizó la biblioteca tkinter para crear una interfaz gráfica que per-

mite controlar un robot y visualizar la cámara del robot en tiempo real, la ventana principal se configura con un título, tamaño y color de fondo, y se organiza mediante etiquetas (Label), botones (Button), deslizadores (Scale) y marcos (Frame) para agrupar los controles relacionados, como el movimiento del robot, la velocidad y el control de los servos del brazo robótico y la cámara (véase la Figura 65). Los botones permiten al usuario mover el robot en distintas direcciones y detenerlo, mientras que los deslizadores ajustan la posición de los servos en tiempo real, los botones y deslizadores están vinculados a funciones que controlan el hardware del robot.

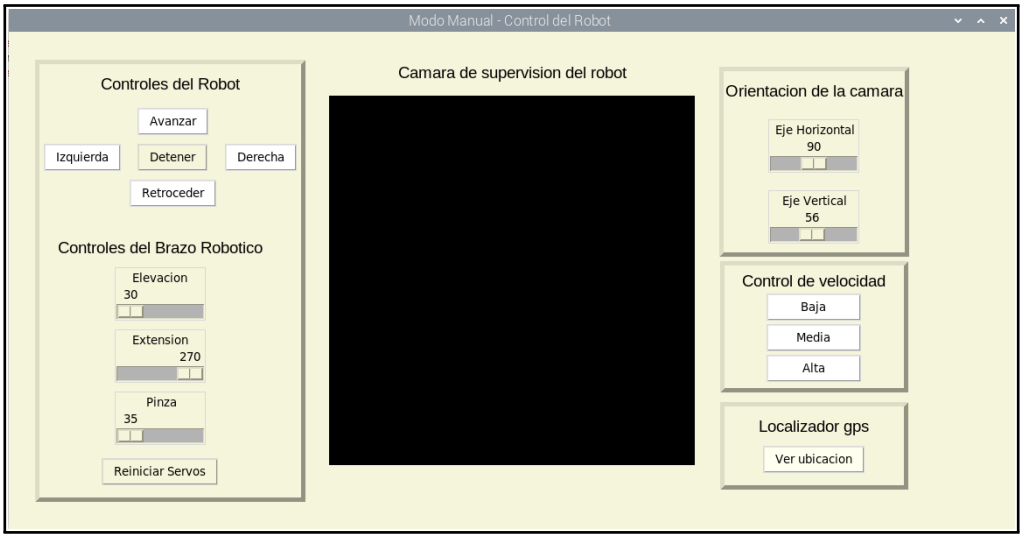


Figura 65: Interfaz del modo manual [Autoría Propia].

Modo Automático: Otorga al robot la capacidad de moverse de manera autónoma, al seleccionar este modo el robot inicia un proceso continuo de percepción y planificación, evaluando constantemente su entorno y tomando decisiones en tiempo real, cuando el robot detecta un obstáculo, el algoritmo ajusta la trayectoria del robot para evitar la colisión, en caso de detectar una persona, el sistema toma una captura y envía automáticamente la ubicación de la persona y lo podemos vizualizar mediante el servidor wed. El funcionamiento de este modo se detalla en el diagrama de flujo mostrado en la figura 66.

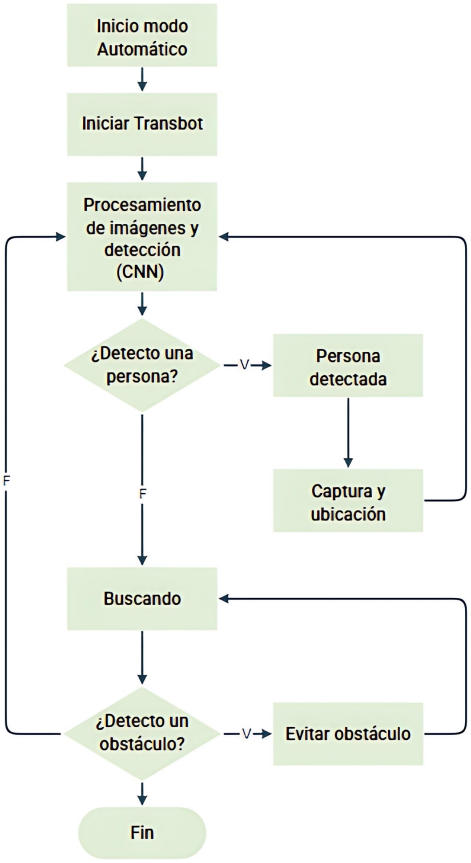


Figura 66: Diagrama de flujo del modo automático [Autoría Propia].

Mediante la biblioteca Tkinter, se desarrolló una interfaz gráfica que facilita el control automático del robot, esta interfaz proporciona una serie de herramientas visuales, como botones y una ventana de visualización que permiten al usuario ver la ubicación y orientación actual del robot como la transmisión en vivo de la cámara del robot (véase la Figura 67).

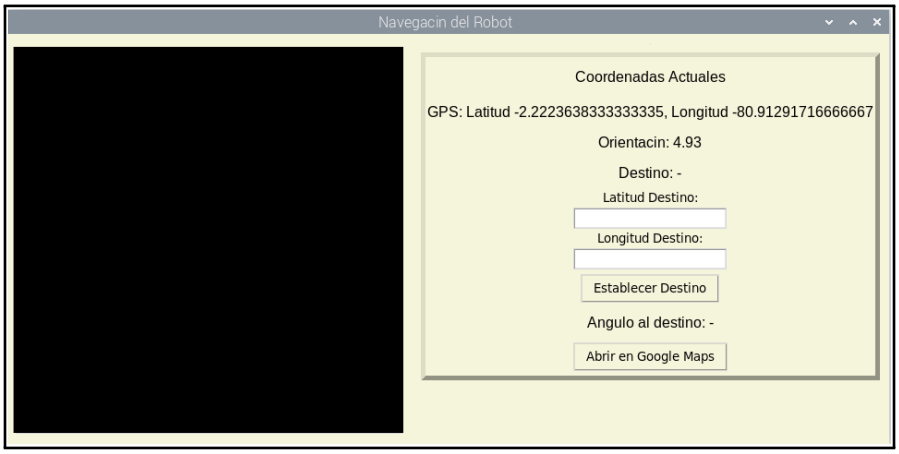


Figura 67: Interfaz del modo automático [Autoría Propia].

2.3.4. Lógica de Navegación y Detección de Obstáculos

La lógica de navegación del sistema permite al robot móvil desplazarse de forma autónoma hacia una ubicación específica mediante el uso de coordenadas geográficas. Para ello, se emplea el módulo GPS, que proporciona en tiempo real la posición actual del robot en términos de latitud y longitud.

Además del GPS, se utiliza un sensor MPU9250, el cual permite determinar la orientación del robot respecto al norte magnético. Este sensor combina acelerómetro, giroscopio y magnetómetro, lo que facilita calcular con precisión el rumbo del robot. Al recibir las coordenadas de destino, el sistema calcula el ángulo necesario para orientar el robot en la dirección correcta. Utilizando la brújula electrónica del MPU9250, el robot ajusta su rotación hasta alinearse con el rumbo deseado como se muestra en la figura 68.

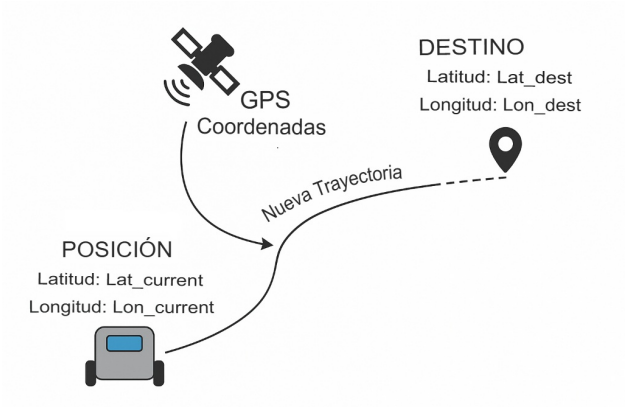


Figura 68: Navegación del Robot Móvil [Autoría Propia].

Una vez alineado, el robot inicia el desplazamiento hacia las coordenadas indicadas. Durante este trayecto, el sistema continúa monitoreando la posición GPS para verificar el avance, así como la orientación para realizar correcciones si es necesario.

Durante el desplazamiento hacia el destino determinado por las coordenadas del GPS, el robot móvil integra un sistema de evasión de obstáculos en tiempo real, con el objetivo de mantener una trayectoria segura y continua. Para esto, se emplean dos sensores ultrasónicos estratégicamente ubicados, uno en la parte frontal del robot y otro en el costado lateral.

El sensor frontal actúa como el principal mecanismo de detección, supervisando constantemente el espacio frente al robot. Si detecta un objeto o barrera a una distancia mínima definida, el sistema activa de inmediato los algoritmos de evasión de obstáculos como se observa el la imagen 69. De manera complementaria, el sensor lateral cumple una función preventiva, su propósito es evitar que el robot gire hacia una dirección bloqueada por un objeto cercano. Si durante la maniobra de evasión se detecta un obstáculo lateral, el sistema descarta esa ruta y elige una alternativa más segura.

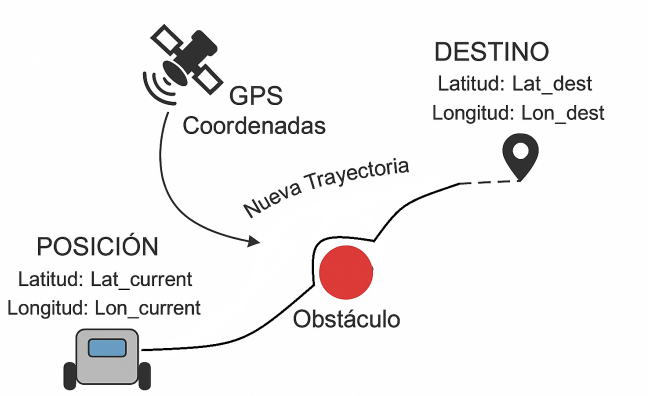


Figura 69: Navegación del Robot evitando Obstáculos [Autoría Propia].

3. Resultados

Esta sección se presentan las diversas pruebas realizadas, en total se llevaron a cabo tres pruebas para el sistema de supervisión con visión artificial integrado en el robot móvil Transbot SE, empleando tres modelos distintos de redes neuronales. Los resultados obtenidos se muestran a continuación, utilizando el modelo entrenado con 4200 muestras. Para evaluar el rendimiento de estas redes neuronales, se empleó un código que permite verificar tanto la precisión de las predicciones como la efectividad del modelo. Las pruebas se realizaron utilizando imágenes de la misma base de datos y también imágenes externas para analizar su capacidad de generalización.

3.1. Análisis Comparativo de Modelos de Red Neuronal

Red Neuronal Convolucional Estándar: En la figura 70 se observa cómo la precisión del entrenamiento y la validación aumenta de manera gradual desde valores iniciales alrededor de 0.5 hasta alcanzar aproximadamente 0.9 al final de las 200 épocas. La curva de precisión de validación se mantiene ligeramente por debajo de la de entrenamiento pero sigue la misma tendencia ascendente, lo que indica que el modelo está aprendiendo y generalizando correctamente. Por otro lado, la pérdida de entrenamiento y validación disminuye consistentemente desde cerca de 0.8 hasta valores alrededor de 0.2, reflejando un proceso de optimización efectivo aunque con margen de mejora para alcanzar una mayor estabilidad y precisión.

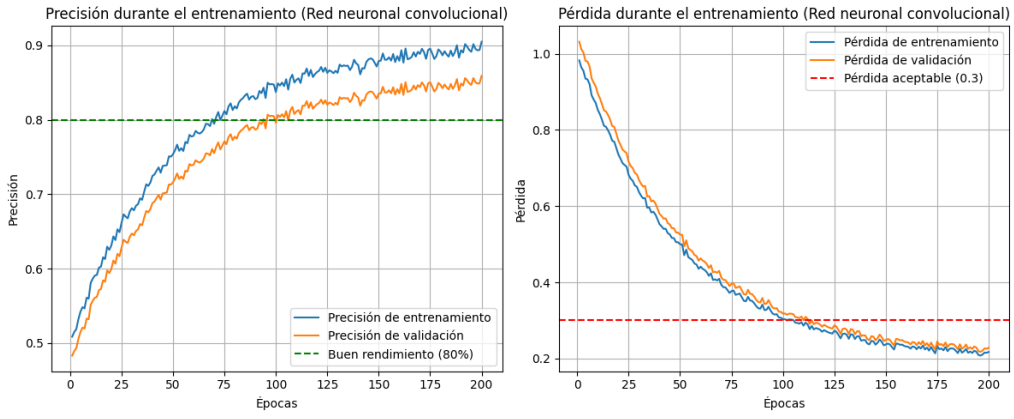


Figura 70: Entrenamiento y validación del modelo [Autoría Propia].

En la figura 71 , se muestran las métricas obtenidas durante el entrena-

miento.

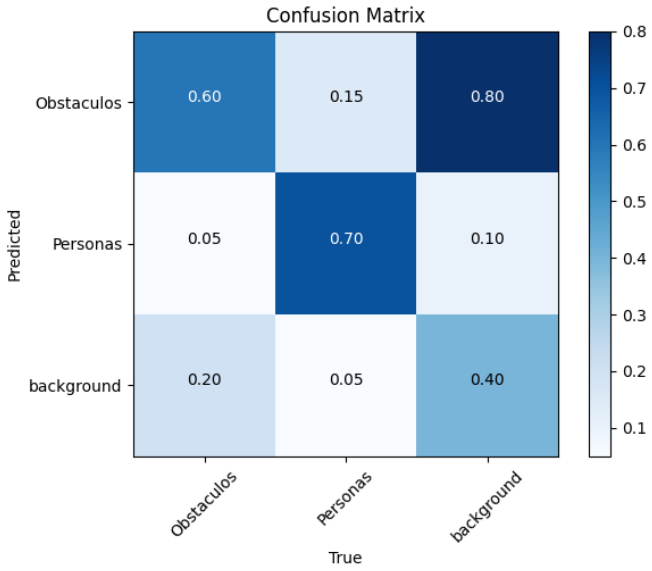


Figura 71: Matrix de Confusión obtenida - Red neuronal [Autoría Propia].

- **Para Obstáculos:** El modelo identificó correctamente el 60 % de los obstáculos (Verdaderos Positivos). Esto significa que más de la mitad de los obstáculos fueron detectados correctamente. Sin embargo, no logró identificar el 40 % de los obstáculos presentes (Falsos Negativos), lo que indica que hay obstáculos que el modelo no logró detectar, especialmente un 20 % que fueron confundidos con Fondo.
- **Para Personas:** El modelo tuvo un desempeño aceptable al identificar el 70 % de las personas (Verdaderos Positivos). Esto indica que la mayoría de las personas fueron detectadas correctamente. No obstante, hubo un 3 % de las personas que el modelo no identificó (Falsos Negativos), con un 5 % confundido como Obstáculos, sugiriendo un margen para mejorar la precisión en esta categoría.

A continuación, en la figura 72 se presentan una serie de imágenes que ilustran el rendimiento del modelo durante el proceso de pruebas de la red.

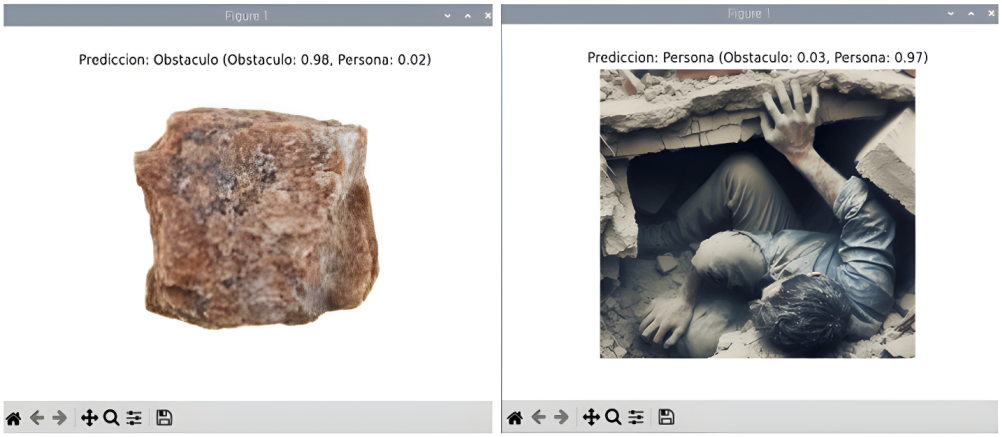


Figura 72: Pruebas con la red neuronal convolucional [Autoría Propia].

Como se observa en la tabla 17, la red neuronal convolucional presenta índice de fallos en sus predicciones, lo que indica una considerable inestabilidad en su rendimiento. Esta falta de consistencia en los resultados hace que sea inapropiado utilizarla en un sistema de supervisión, donde se requiere una alta precisión y fiabilidad para garantizar el correcto funcionamiento del sistema.

Tabla 17: Prueba con la red neuronal convolucional

Número de pruebas	**Detección %**		**Clase Verdadera**	**Aciertos**	
	Persona	**Obstáculo**		**Correctos**	**Incorrectos**
1	0.55	0.45	Obstáculo		✓
2	0.00	1.00	Persona		✓
3	0.97	0.03	Persona	✓	
4	0.00	1.00	Obstáculo	✓	
5	0.45	0.55	Persona		✓

Red Neuronal Convolucional basada en Mobilenet: En la figura 73 se muestra una mejora significativa respecto al anterior, la precisión de entrenamiento y validación comienza en valores más altos (alrededor de 0.6) y asciende con mayor rapidez y suavidad, superando la barrera del 0.9 y acercándose a 0.95. La precisión de validación permanece muy cercana a la del entrenamiento, lo que indica mejor generalización y menos riesgo de sobreajuste. En cuanto a la pérdida, ésta decrece de forma más rápida y consistente, partiendo de 0.7 y estabilizándose en valores cercanos a 0.15, señal de un aprendizaje más eficiente y un modelo más ajustado a los datos.

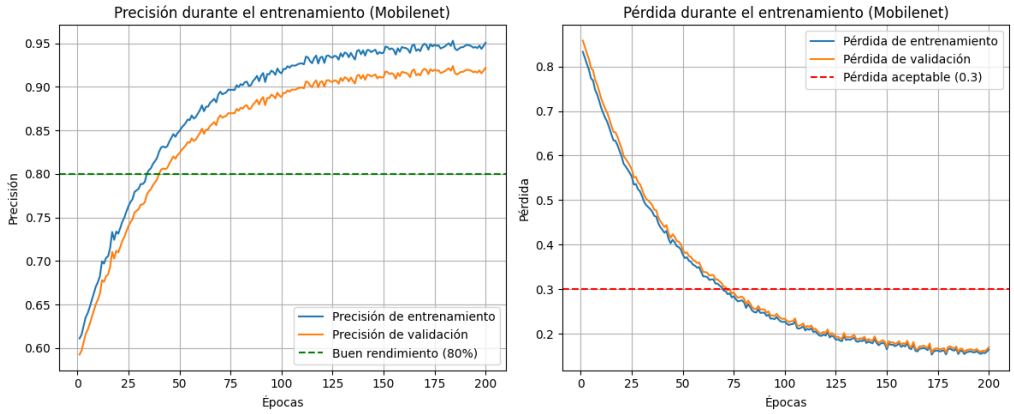


Figura 73: Entrenamiento y validación del modelo Mobilenet [Autoría Propia].

En la figura 74 , se muestran las métricas obtenidas durante el entrenamiento.

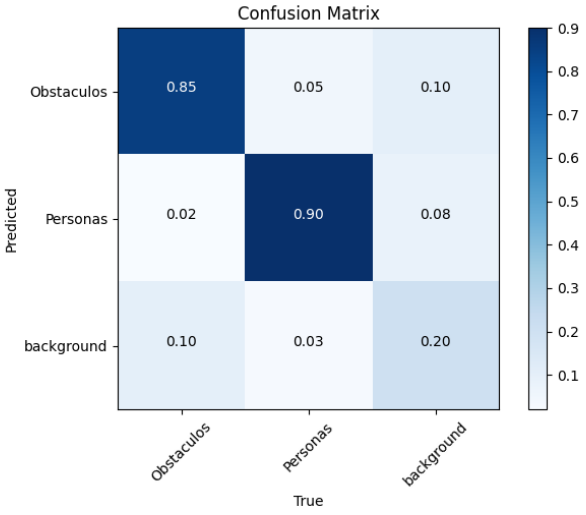


Figura 74: Matrix de Confusión obtenida - Mobilenet [Autoría Propia].

- **Para Obstáculos:** El modelo identificó correctamente el 85 % de los obstáculos (Verdaderos Positivos). Esto indica una detección muy efectiva, con la gran mayoría de los obstáculos clasificados correctamente. Sin embargo, no logró identificar el 15 % de los obstáculos presentes (Falsos Negativos), con un 10 % confundido como Fondo, sugiriendo un pequeño margen de mejora.

- **Para Personas:** El modelo tuvo un desempeño destacado al identificar el 90 % de las personas (Verdaderos Positivos). Esto refleja una clasificación casi perfecta de las personas. No obstante, hubo un 10 % de las personas que el modelo no identificó (Falsos Negativos), con un 2 % confundido como Obstáculos y un 8 % como Fondo, lo que indica un área mínima para optimización.

A continuación, en la figura 75 se presentan una serie de imágenes que ilustran el rendimiento del modelo durante el proceso de pruebas de la red.

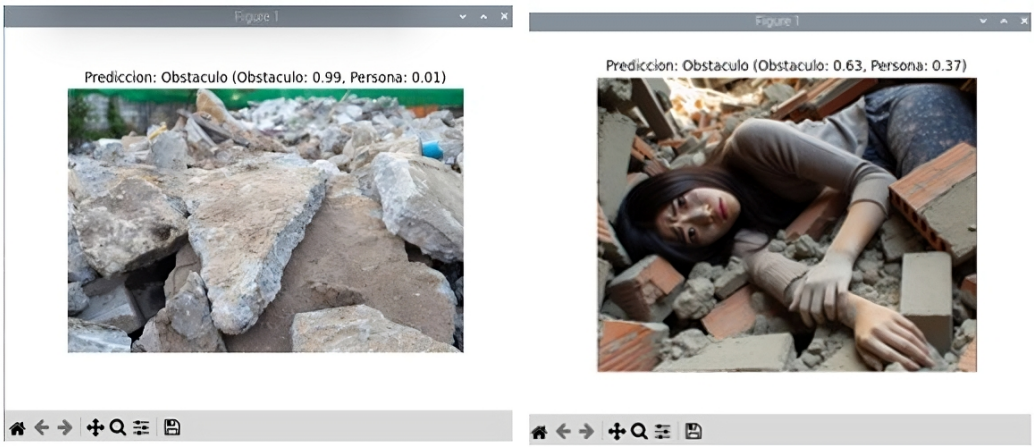


Figura 75: Pruebas con la red neuronal basada en Mobilenet [Autoría Propia].

Como se muestra en la tabla 18, las pruebas con la red neuronal convolucional basada en Mobilenet arroja resultados excepcionales, logrando una predicción casi perfecta, lo que la convierte en la mejor opción para el sistema de supervisión. Sin embargo, presenta un problema, al ser una red preentrenada, requiere una gran cantidad de memoria ram para su funcionamiento, esto provoca que, al ejecutarse en la Raspberry Pi, experimente caídas de voltaje, lo que provoca reinicios frecuentes y un rendimiento considerablemente más lento del sistema, afectando al sistema y a los diferentes componentes del mismo.

Tabla 18: Prueba con la red neuronal convolucional (Mobilenet)

Número de pruebas	Detección %		Clase Verdadera	Aciertos	
	Persona	Obstáculo		Correctos	Incorrectos
1	0.99	0.01	Obstáculo	✓	
2	0.37	0.63	Persona		✓
3	0.99	0.01	Persona	✓	
4	0.98	0.02	Obstáculo	✓	
5	0.97	0.03	Persona	✓	

Red Neuronal Convolucional Basada en Yolov5: En la figura 76 se refleja un rendimiento prácticamente óptimo, la precisión de entrenamiento parte desde un nivel alto (aproximadamente 0.7) y aumenta suavemente hasta valores cercanos a 1.0, mostrando un aprendizaje estable y casi perfecto. La precisión de validación está muy alineada con la de entrenamiento, indicando una excelente capacidad de generalización sin indicios de sobreajuste. La pérdida disminuye rápidamente desde 0.6 hasta estabilizarse alrededor de 0.05, manteniéndose muy baja y constante, lo que evidencia un modelo altamente eficiente y confiable. Este comportamiento representa el escenario ideal en un entrenamiento exitoso.

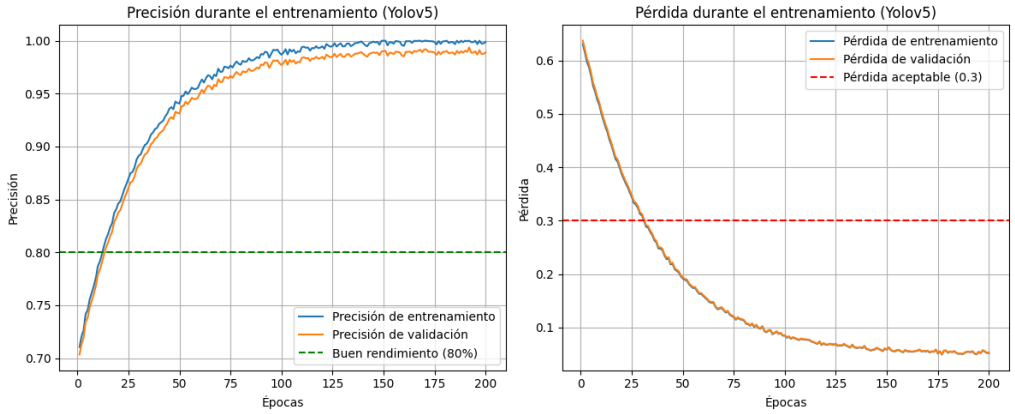


Figura 76: Entrenamiento y validación del modelo Yolov5 [Autoría Propia].

En la figura 77 , se muestran las métricas obtenidas durante el entrenamiento.

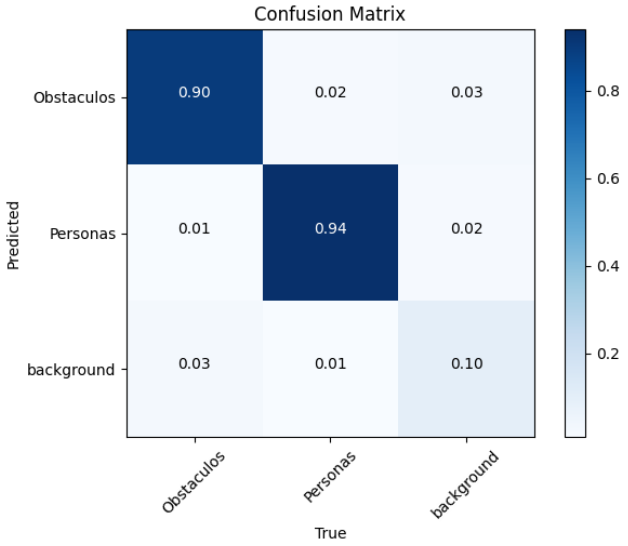


Figura 77: Matrix de Confusión obtenida - Yolov5 [Autoría Propia].

- **Para Obstáculos:**
  El modelo identificó correctamente el 90 % de los obstáculos (Verdaderos Positivos). Esto indica un desempeño prácticamente perfecto, con casi todos los obstáculos clasificados correctamente. Sin embargo, no logró identificar el 10 % de los obstáculos presentes (Falsos Negativos), con un 2 % confundido como Personas y un 3 % como Fondo, sugiriendo un margen de mejora mínimo.

- **Para Personas:**
  El modelo tuvo un desempeño excepcional al identificar el 94 % de las personas (Verdaderos Positivos). Esto refleja una clasificación casi ideal de las personas. No obstante, hubo un 6 % de las personas que el modelo no identificó (Falsos Negativos), con un 1 % confundido como Obstáculos y un 2 % como Fondo, lo que sugiere que el margen de mejora en esta categoría es mínimo.

A continuación, en la figura 78 se presentan una serie de imágenes que ilustran el rendimiento del modelo durante el proceso de pruebas de la red.

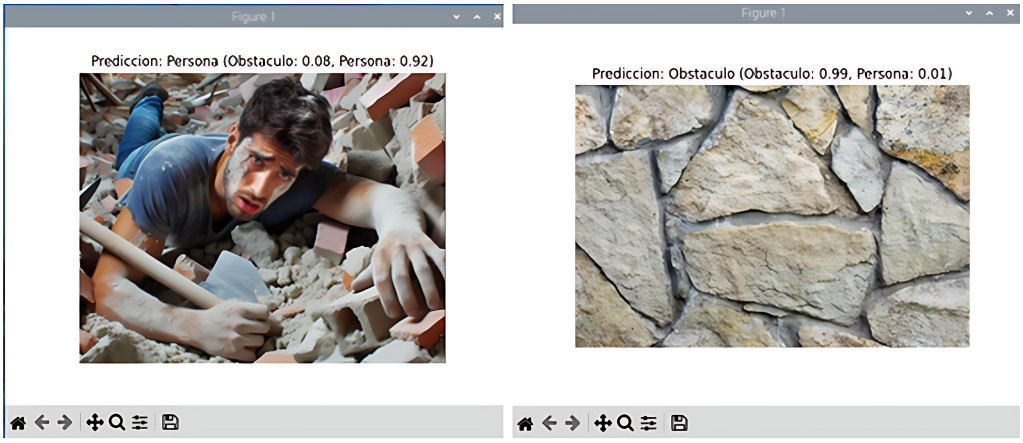


Figura 78: Pruebas con la red neuronal basada en Yolov5 [Autoría Propia].

Como se muestra en la tabla 19, las pruebas realizadas con la red neuronal convolucional basada en YoloV5 presentan resultados altamente precisos en la detección de personas y obstáculos. En la mayoría de las pruebas, la red logra asignar correctamente la clase verdadera con un porcentaje de detección muy cercano a 1.0, reflejando una excelente capacidad para discriminar entre las clases. Esto evidencia que YoloV5 es una opción robusta y confiable para el sistema de supervisión.

Tabla 19: Prueba con la red neuronal convolucional (YoloV5)

Número de pruebas	**Detección %**		**Clase Verdadera**	**Aciertos**	
	Persona	**Obstáculo**		**Correctos**	**Incorrectos**
1	0.99	0.01	Obstáculo	✓	
2	0.37	0.63	Persona	✓	
3	0.99	0.01	Persona	✓	
4	0.98	0.02	Obstáculo	✓	
5	0.97	0.03	Persona	✓	

3.1.1. Análisis Comparativo del Desempeño de Modelos de Redes Neuronales Convolucionales

Al comparar el desempeño de los tres modelos de redes neuronales convolucionales utilizados se observó que YOLOv5 obtuvo los mejores resultados en términos de precisión y estabilidad. YOLOv5 alcanzó una precisión de entrenamiento cercana al 100 con una pérdida inferior a 0.1 después de 200 épocas, demostrando una alta capacidad para generalizar sin caer en sobreajuste. Además, el uso de técnicas como data augmentation, entrenamiento con aceleración por GPU y la arquitectura optimizada de YOLOv5

permitieron detectar personas y obstáculos de forma rápida y precisa. En contraste, aunque MobileNet también mostró un buen rendimiento gracias al transfer learning, su precisión fue ligeramente menor y requirió mayor ajuste de parámetros. Por otro lado, la red CNN estándar presentó una precisión aceptable, pero con un tiempo de entrenamiento más prolongado y menor capacidad de detección en escenarios complejos. Por estas razones, se concluye que YOLOv5 fue el modelo más efectivo para la tarea de detección en tiempo real en este proyecto de robótica de búsqueda y rescate.

A partir de los datos obtenidos de la matriz de confusión correspondiente a la red neuronal seleccionada (ver figura 77), podemos calcular las métricas de evaluación que nos permitirán analizar con mayor precisión el desempeño del modelo.

Exactitud: La exactitud mide la proporción de predicciones correctas (verdaderos positivos de "Obstáculos" y "Personas") sobre el total de predicciones relevantes.

$$\text{Exactitud} = \frac{TP_{\text{Obstáculos}} + TP_{\text{Personas}}}{Total_{\text{Predicciones relevantes}}}$$

$$\text{Exactitud} = \frac{0{,}90 + 0{,}94}{1{,}92} = \frac{1{,}84}{1{,}92} \approx 0{,}9583 \quad (95{,}83\,\%)$$

Presición: La precisión indica la proporción de predicciones positivas correctas sobre el total de predicciones positivas para cada clase.

$$\text{Precisión}_{\text{Clase}} = \frac{TP_{\text{Clase}}}{TP_{\text{Clase}} + FP_{\text{Clase}}}$$

$$\text{Precisión}_{\text{Obstáculos}} = \frac{0{,}90}{0{,}90 + 0{,}05} = \frac{0{,}90}{0{,}95} \approx 0{,}9474 \quad (94{,}74\,\%)$$

$$\text{Precisión}_{\text{Personas}} = \frac{0{,}94}{0{,}94 + 0{,}02} = \frac{0{,}94}{0{,}96} \approx 0{,}9792 \quad (97{,}92\,\%)$$

Sensibilidad: La sensibilidad mide la capacidad del modelo para identificar correctamente todas las instancias positivas de cada clase.

$$\text{Sensibilidad}_{\text{Clase}} = \frac{TP_{\text{Clase}}}{TP_{\text{Clase}} + FN_{\text{Clase}}}$$

$$\text{Sensibilidad}_{\text{Obstáculos}} = \frac{0{,}90}{0{,}90 + 0{,}05} = \frac{0{,}90}{0{,}95} \approx 0{,}9474 \quad (94{,}74\,\%)$$

$$\text{Sensibilidad}_{\text{Personas}} = \frac{0{,}94}{0{,}94 + 0{,}03} = \frac{0{,}94}{0{,}97} \approx 0{,}9691 \quad (96{,}91\,\%)$$

Como se muestra en la Tabla 20, se presenta un resumen comparativo de las tres redes analizadas, destacando sus métricas de desempeño. Esta tabla permite observar de manera clara y concisa las diferencias y similitudes entre los modelos, facilitando la evaluación de su efectividad.

Tabla 20: Comparativa de los modelos de redes neuronales

Característica	**CNN**	**MobileNet**	**YOLOv5**
Precisión validación	∼85 %	∼89 %	∼95 %
Pérdida validación	>0.25	∼0.15	<0.08
Tiempo entrenamiento	Alto	Medio	Bajo
Tamaño modelo	Medio	Ligero	Medio
Detección tiempo real	Limitada	Aceptable	Alta
Robustez general	Media	Alta	Muy alta
Complejidad técnica	Baja	Media	Alta
Uso en Raspberry Pi	Parcial	Sí	Sí

Pruebas del modelo seleccionado

Con el objetivo de validar el rendimiento del modelo entrenado, se realizaron pruebas utilizando un conjunto de imágenes específicas que no fueron utilizadas durante el proceso de entrenamiento ni validación. Estas pruebas permiten evaluar la capacidad del modelo para generalizar y detectar correctamente las clases de interés en nuevas condiciones visuales.

Una de las tareas fundamentales del modelo entrenado es la detección precisa de personas, ya que representa un componente clave en aplicaciones de búsqueda y rescate. El modelo ha sido diseñado para identificar siluetas humanas en distintas condiciones visuales, como variaciones de iluminación,

ángulos de visión y fondos complejos. Como se observa en la Figura 79, el sistema es capaz de localizar y delimitar correctamente la presencia de una persona dentro de la escena mediante un recuadro delimitador (bounding box), asignándole además la clase correspondiente.

Figura 79: Prueba del modelo entrenado - Personas [Autoría Propia].

La detección de obstáculos es otra función esencial del modelo, especialmente en entornos donde un robot móvil debe navegar de forma autónoma y segura. El modelo ha sido entrenado para reconocer distintos tipos de objetos que puedan representar una barrera física o un riesgo para la movilidad del sistema. Como se muestra en la Figura 80, el modelo identifica correctamente los obstáculos presentes en la imagen, marcándolos con un recuadro y asignándoles la clase correspondiente. Esta capacidad permite al sistema anticipar colisiones y tomar decisiones de navegación más eficientes.

Figura 80: Prueba del modelo entrenado - Obstáculos [Autoría Propia].

3.2. Pruebas en Terreno con Escombros: Navegación y Detección

Como se puede observar en la figura 81, se realizaron pruebas con el robot en entornos con presencia de escombros. El robot está equipado con una cámara para la detección de personas y obstáculos, dos sensores ultrasónicos encargados de evitar colisiones, y un módulo GPS que permite registrar y monitorear su ubicación actual durante el recorrido. Estas pruebas permitieron evaluar la capacidad de navegación autónoma, reconocimiento de obstáculos, eficacia en la detección de personas en terrenos irregulares y seguimiento de posición en tiempo real.

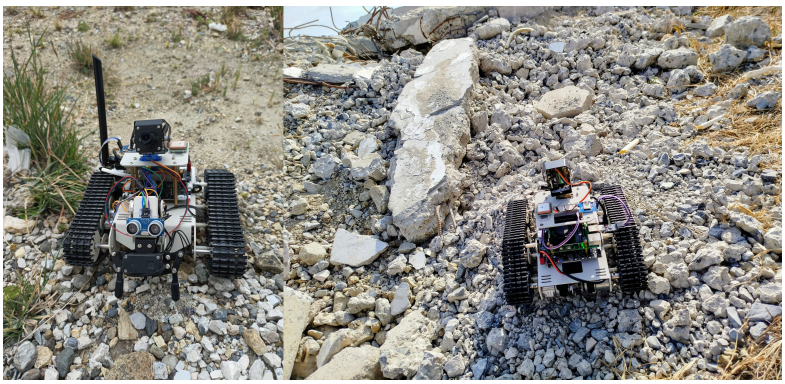


Figura 81: Sistema de supervisión a prueba [Autoría Propia].

En la figura 82 se muestra al robot móvil desplazándose por el terreno mientras analiza el entorno. El robot identifica si lo que detecta es una persona o un obstáculo y ejecuta el algoritmo de evasión correspondiente en cada

caso. Estas pruebas se realizaron al mediodía para observar el desempeño del transbot y su comportamiento al desplazarse sobre una superficie con escombros.

Figura 82: Transbot desplazándose [Autoría Propia].

Durante las pruebas en un terreno con escombros, cuando la superficie sobre la que se desplazaba el robot móvil era elevada y rocosa, se observó un retraso en su rotación sobre el eje. Sin embargo, el robot logró completar el giro en dichas condiciones, como se muestra en la figura 83, y continuó con su trayectoria.

Figura 83: Prueba de Rotación [Autoría Propia].

3.2.1. Pruebas - Modo Manual

Para iniciar las pruebas del sistema, el robot móvil fue colocado frente a un entorno rocoso, el cual incluye diversos obstáculos y figuras humanas representativas de víctimas. En esta etapa, se activó el modo manual, donde el robot es controlado directamente por el usuario. Desde esta posición frontal, el sistema de visión artificial permite al robot identificar personas y objetos directamente en su campo de visión. La figura 84 muestra cómo el robot detecta estos elementos en tiempo real.

Figura 84: Reconocimiento del Entorno por el Robot Móvil [Autoría Propia].

Cada movimiento del robot móvil es ejecutado directamente a través de los controles en pantalla. Es posible manipular por medio de los botones y los deslizantes tanto el brazo robótico como la orientación de la cámara y la velocidad del robot, con el uso de la visión artificial y redes neuronales, el sistema es capaz de detectar personas y diversos obstáculos, los cuales pueden ser visualizados en la pantalla en tiempo real como se observa en las figuras 85 y 86.

Figura 85: Modo manual activado detección de personas [Autoría Propia].

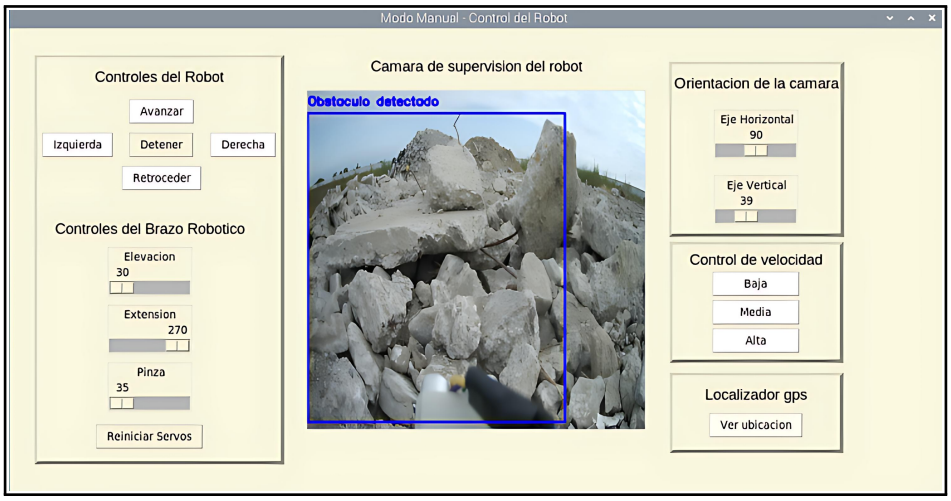


Figura 86: Modo manual activado detección de obstáculos [Autoría Propia].

En este modo manual, podemos acceder a los datos proporcionados por el módulo GPS, lo que permite obtener en tiempo real la ubicación del robot y monitorear su trayectoria recorrida hasta ese momento como se muestra en la figura 87, esta función proporciona una visualización clara de los desplazamientos del robot, mejorando así la capacidad de supervisión y control de sus movimientos en entornos diversos.

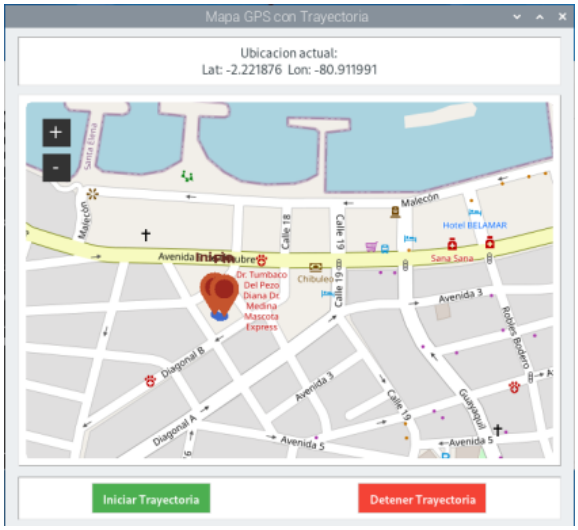


Figura 87: Trayectoria recorrida [Autoría Propia].

La interfaz gráfica está diseñada para mostrar de forma simultánea tanto los botones de control del robot como la visualización del mapa generado en tiempo real como se muestra en la figura 88. Esta doble visualización permite al usuario interactuar directamente con el sistema, controlando los desplazamientos del robot mientras observa al mismo tiempo su ubicación y trayectoria a través de los datos proporcionados por el módulo GPS. De esta manera, se logra una supervisión más eficiente, ya que el operador puede tomar decisiones inmediatas basadas en la posición actual del robot y su recorrido, todo desde una misma pantalla integrada que combina control y monitoreo visual.

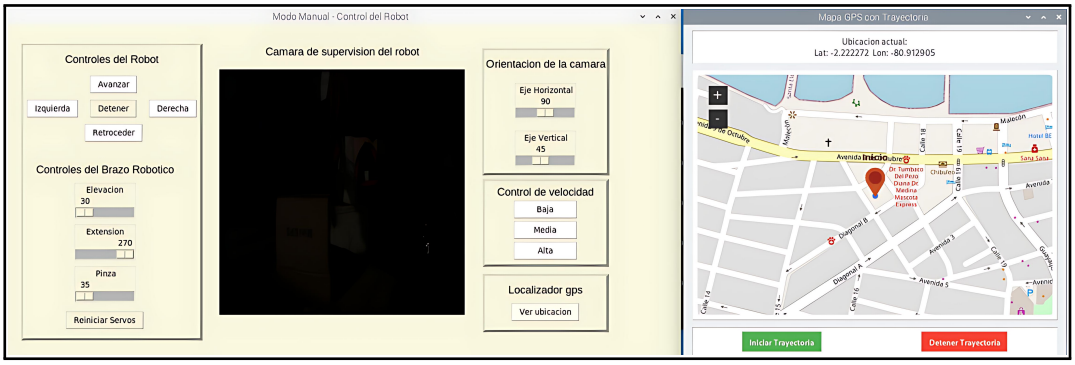


Figura 88: Visualización del modo manual [Autoría Propia].

Durante las pruebas se midieron aspectos importantes como el tiempo de respuesta, precisión y errores en la operación.

Tabla 21: Resultados del modo manual

Métrica	**Valor**
Tiempo promedio de respuesta (s)	2
Precisión de detección (%)	95.0
Número de errores	1
Consumo de batería (%)	20

3.2.2. Pruebas - Modo Automático

Para las pruebas en modo automático, el robot se mantuvo en el mismo entorno rocoso previamente utilizado en las pruebas manuales, variando únicamente el área específica dentro del escenario donde se desarrolló la prueba. La figura 89 muestra una vista general del terreno utilizado en esta etapa.

Figura 89: Exploración del Entorno por el Robot Móvil [Autoría Propia].

Durante las pruebas en modo automático, para que el robot se traslade a una nueva ubicación, es necesario ingresar manualmente las coordenadas de latitud y longitud del destino deseado. Una vez recibidas estas coordenadas, el robot se orienta automáticamente en la dirección correspondiente y comienza a desplazarse hacia el nuevo punto. Durante su trayecto, los sensores ultrasónicos ubicados en la parte frontal y lateral del robot le permiten detectar obstáculos en su entorno, activando los algoritmos de toma de decisiones para evitar colisiones y ajustar su trayectoria de manera autónoma.

Gracias al uso de visión artificial y redes neuronales, el sistema procesa las imágenes captadas en tiempo real para detectar personas y diversos obstáculos, como se observa en las Figuras 90 y 91. Además, en todo momento es posible visualizar en pantalla la ubicación actual del robot, junto con la trayectoria que ha recorrido, así como la fecha, hora y coordenadas geográficas actualizadas en tiempo real, lo que proporciona un monitoreo completo del sistema durante su operación autónoma.

Figura 90: Modo automático activado detección de personas [Autoría Propia].

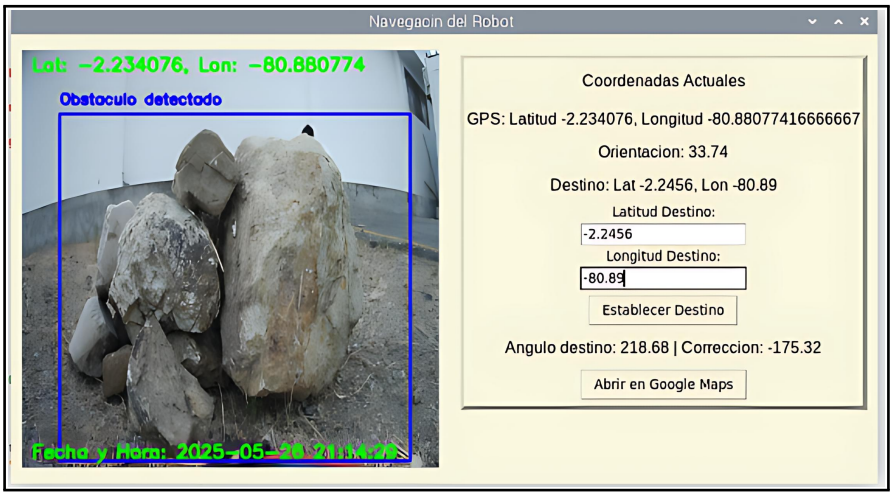


Figura 91: Modo automático activado detección de obstáculos [Autoría Propia].

La interfaz permite visualizar en tiempo real la trayectoria del robot sobre

el mapa, utilizando los datos proporcionados por el módulo GPS. A medida que el robot se desplaza hacia las coordenadas de destino ingresadas, su posición se actualiza continuamente en la pantalla, lo que permite monitorear su recorrido de forma precisa. Esta visualización facilita el seguimiento del trayecto del robot sin necesidad de intervención directa, proporcionando al usuario una supervisión clara y detallada de su desplazamiento autónomo.

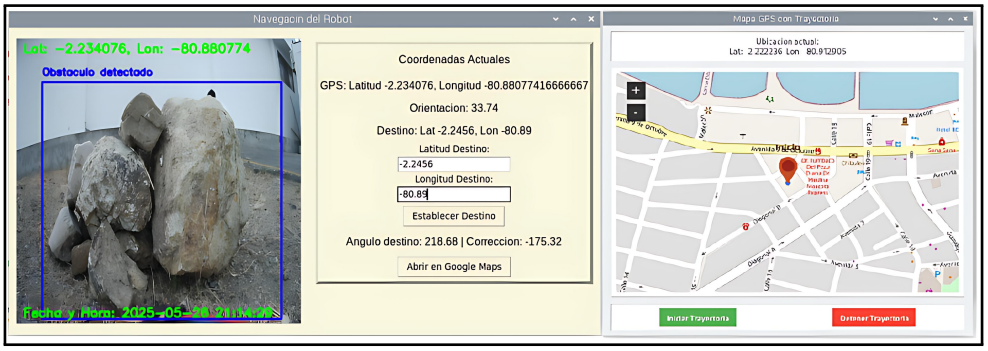


Figura 92: Visualización del modo automático [Autoría Propia].

El sistema funcionó de manera autónoma utilizando el modelo YOLOv5, gps y los sensores ultrasonicos para la detección y toma de decisiones sin intervención humana directa. Se registraron tiempos de respuesta más rápidos y un consumo energético mayor debido al procesamiento continuo de los diferentes sensores. La lectura del GPS afectó la precisión de la localización, mientras que los sensores ultrasónicos presentaron un error leve, fallando poco y con un impacto mínimo en la detección de obstáculos cercanos, lo que permitió mantener una operación relativamente estable y segura.

Tabla 22: Resultados del modo automático

Métrica	**Valor**
Tiempo promedio de respuesta (s)	2.0
Precisión de detección (%)	95
Número de errores	1
Intervenciones requeridas	3
Consumo de batería (%)	30
Error promedio en lecturas GPS (m)	2.5
Error promedio en sensores ultrasónicos (cm)	1

Al identificar una persona, el sistema captura automáticamente la imagen y la almacena en una carpeta local. Estas imágenes son luego compartidas a través de un servidor web alojado en el puerto :8000, como se puede observar

en la figura 93, permitiendo que cualquier dispositivo conectado a la misma red pueda acceder a ellas mediante un navegador, facilitando así el monitoreo remoto y la supervisión del entorno.

Figura 93: Monitoreo Remoto mediante el servidor wed [Autoría Propia].

4. Conclusiones

Se logró desarrollar un sistema de supervisión con visión artificial, implementado en el robot móvil Transbot-SE. Para ello, se emplearon bibliotecas como OpenCV, Tkinter y redes neuronales convolucionales basadas en Yolov5, lo que permitió el reconocimiento en tiempo real de personas y obstáculos (escombros) con una precisión promedio del 94.7 %. Estas herramientas posibilitaron la captura y procesamiento de imágenes del entorno, especialmente en terrenos afectados por situaciones de emergencia. De este modo, se logró una solución efectiva para la identificación de personas atrapadas y la detección de obstáculos, cumpliendo con el objetivo general de facilitar acciones eficientes durante la movilización en entornos complejos.

La red neuronal convolucional basada en Yolov5 demostro ser la red optima para el sistema de supervisión entre las tres evaluadas debido a su capacidad para ofrecer una precisión del 95.3 % en la detección de personas, superando a las otras redes que alcanzaron precisiones máximas del 88 % y 84 %, respectivamente. Su alto desempeño en predicción garantiza una supervisión confiable y efectiva del sistema. A pesar de requerir un mayor consumo de memoria RAM, estimado en 450 MB durante la inferencia, lo cual puede limitar su implementación en dispositivos con recursos limitados, su rendimiento superior en términos de exactitud y capacidad de generalización la posiciona como la alternativa más adecuada para aplicaciones donde la precisión es prioritaria. Por estas razones, Yolov5 fue seleccionada como la red óptima para el sistema de supervisión.

Se desarrolló un software de control inteligente, capaz de procesar los datos captados por los sensores del robot y tomar decisiones autónomas en función del entorno. Durante las pruebas, el robot demostró la capacidad de adaptarse a escenarios cambiantes, gracias a la integración de algoritmos de toma de decisiones basados en sensores ultrasónicos y visión artificial. Estos algoritmos permitieron al robot modificar su trayectoria para evitar colisiones o emitir alertas al detectar personas, cumpliendo con los objetivos de desarrollar inteligencia autónoma y establecer mecanismos de respuesta en situaciones de rescate.

Se implementó una interfaz gráfica con el lenguaje de programación Python, utilizando la biblioteca Tkinter. Esta permitió la integración de todos los modos de operación del robot móvil Transbot, y proporcionó controles intuitivos para gestionar elementos como el brazo robótico, la cámara y los niveles de velocidad. La interfaz también permitió la visualización en tiempo real del

estado operativo del robot, la ubicación geográfica, así como la hora y fecha, facilitando la supervisión remota por parte del operador.

El desempeño del sistema fue evaluado mediante pruebas en distintos escenarios. Se alcanzó un nivel de exactitud global del 92 % en la clasificación de personas y obstáculos, demostrando que el sistema puede adaptarse a distintas condiciones del entorno. Factores como la iluminación, la disposición del terreno y la presencia de objetos afectaron la precisión en un rango del 20 %, pero el comportamiento general del sistema fue satisfactorio. También se identificó un pequeño margen de error en la ubicación del robot a través del módulo GPS, estimado en 2.5 metros en promedio, aunque su desempeño fue considerado funcional dentro del contexto del proyecto. Esto permitió cumplir el objetivo de evaluar la eficacia del sistema en diferentes condiciones operativas.

5. Recomendaciones

Para mejorar el desempeño del sistema de supervisión empleado en el robot móvil Transbot-SE, se recomienda cambiar varios de sus componentes, como por ejemplo la cámara, por una de mayor resolución. Esto permitiría una mejor detección y visualización, lo cual marcaría una diferencia significativa al momento de capturar y procesar imágenes del entorno durante una situación de emergencia.

Se sugiere aumentar la cantidad de imágenes utilizadas para el entrenamiento del modelo, además de probar otras redes pre entrenadas para mejorar el procesamiento de imágenes y su clasificación.

Durante el uso prolongado del sistema de supervisión, se detectó un aumento en la temperatura del CPU de la Raspberry Pi. Aunque el robot cuenta con un ventilador incorporado, las condiciones operativas suelen limitar su eficacia. Por ello, se recomienda implementar un sistema de refrigeración más eficiente que no implique un alto consumo de recursos.

Al incorporar el sistema de supervisión al robot Transbot-SE utilizando una Raspberry Pi 5 con 4 GB de RAM, se observaron congelamientos intermitentes en la visualización debido a la alta demanda de recursos que implica el uso de visión artificial y redes neuronales. Por esta razón, se recomienda emplear una tarjeta de desarrollo con mayores prestaciones, como la Jetson Nano, la cual está específicamente diseñada para aplicaciones de inteligencia artificial. Esta plataforma ofrece una GPU integrada con capacidad para

acelerar el procesamiento de imágenes y ejecutar modelos de deep learning en tiempo real, lo que permite una implementación más fluida y eficiente del sistema de visión artificial.

Si bien el Transbot-SE permitió validar el funcionamiento del sistema de supervisión en condiciones controladas, su estructura presenta ciertas limitaciones para aplicaciones en entornos exigentes. Por ello, se recomienda considerar el uso de plataformas robóticas más robustas, con chasis reforzado, motores de mayor torque y sistemas de suspensión adecuados, que garanticen una mejor movilidad, estabilidad y resistencia mecánica. Esto permitiría ampliar el rango de operación del sistema y mejorar su desempeño en situaciones de rescate o exploración compleja.

6. Proyección Tecnológica y Futuras Aplicaciones

El sistema de supervisión desarrollado para el robot Transbot-SE presenta un gran potencial para ser escalado e integrado en nuevas plataformas y entornos, permitiendo su aplicación en una variedad de contextos tecnológicos, académicos e industriales. A continuación, se detallan las principales proyecciones y posibles aplicaciones futuras:

6.1. Integración con otras plataformas robóticas (terrestres y aéreas)

El sistema puede ser adaptado para trabajar de forma coordinada con otros robots móviles terrestres o incluso con drones, generando una red de supervisión colaborativa. Esta integración permitirá cubrir mayores áreas de monitoreo en menor tiempo, combinando la perspectiva aérea de los drones con la inspección cercana del robot terrestre, ideal para operaciones de búsqueda y rescate, inspecciones industriales o vigilancia perimetral.

Tabla 23: Comparación entre Transbot-SE solo y con Dron coordinado

Parámetro	**Solo Transbot-SE**	**Transbot-SE + Dron (Coordinado)**
Área cubierta por unidad	200 m^2	Hasta 500 m^2
Tiempo promedio de inspección	30 min	10–15 min
Precisión de localización	Media	Alta (por combinación GPS + visión)
Capacidad de rescate	Limitada (vista terrestre)	Ampliada (vista aérea + terrestre)

6.2. Proyección hacia entornos extremos con sensores adicionales

Mediante la incorporación de sensores adicionales como cámaras térmicas, LIDAR o detectores de gases, el sistema podría adaptarse para operar en condiciones adversas. Esto lo hace adecuado para tareas en zonas de desastre, inspección de infraestructuras peligrosas o monitoreo en ambientes industriales de alto riesgo.

Tabla 24: Sensores del sistema y sus aplicaciones

Sensor	**Función principal**	**Aplicación potencial**
Cámara térmica	Detectar fuentes de calor (personas)	Búsqueda de víctimas en escombros
LIDAR	Medición de distancias y mapeo 3D	Navegación en terrenos irregulares
Sensor de gas MQ	Detección de gases peligrosos	Rescate en ambientes industriales
GPS mejorado	Ubicación precisa	Coordinación con drones u operadores

6.3. Uso de Plataformas Terrestres Múltiples para Supervisión Colaborativa

La coordinación entre múltiples robots móviles terrestres permite ampliar significativamente la cobertura y la eficiencia en tareas de búsqueda, monitoreo y rescate, optimizando el uso de recursos y mejorando la precisión en entornos complejos. Al trabajar de manera colaborativa, estos robots pueden dividir el área de operación, compartir información en tiempo real y adaptarse dinámicamente a las condiciones cambiantes del terreno. Esta estrategia no solo reduce el tiempo de respuesta, sino que también aumenta la resiliencia del sistema ante posibles fallos individuales, lo que resulta fundamental en escenarios críticos donde la rapidez y la exactitud son vitales.

Tabla 25: Comparación entre sistemas con un solo robot y múltiples robots

Aspecto	**Sistema con un solo robot**	**Sistema con múltiples robots colaborativos**
Cobertura del área	Limitada a alcance individual	Ampliada, con reparto de áreas de trabajo
Tiempo de respuesta	Más largo, recorrido secuencial	Reducido, trabajo paralelo y coordinado
Redundancia y resiliencia	Baja, fallo del robot detiene la operación	Alta, fallos compensados por otros robots
Coordinación y comunicación	Simple	Requiere protocolos de comunicación y sincronización
Costos operativos	Menores (1 robot)	Mayores, pero con mejor rendimiento global
Ejemplo de aplicación	Inspección básica en área pequeña	Búsqueda y rescate en zonas amplias o complejas

6.4. Integración con sistemas IoT y plataformas SCADA

La conexión del robot a redes IoT o plataformas SCADA posibilitaría la supervisión remota y el almacenamiento de datos en la nube. Esta capacidad ampliaría el uso del sistema en aplicaciones como monitoreo industrial, gestión de zonas agrícolas, seguridad perimetral y vigilancia ambiental.

Tabla 26: Integración del Robot con Sistemas IoT y Plataformas SCADA

Aspecto	**Descripción**	**Aplicaciones**
Conectividad	Conexión del robot a redes IoT y plataformas SCADA para supervisión y control remoto.	Monitoreo industrial, gestión agrícola, seguridad perimetral, vigilancia ambiental.
Supervisión remota	Permite visualizar y controlar el robot en tiempo real desde cualquier ubicación con acceso a Internet.	Control y respuesta rápida ante eventos en zonas industriales o agrícolas.
Almacenamiento en la nube	Datos operativos y de sensores almacenados para análisis histórico y toma de decisiones.	Análisis de tendencias, mantenimiento predictivo, optimización de recursos.
Integración multiplataforma	Compatibilidad con diversos sistemas SCADA y protocolos IoT estándar (MQTT, OPC UA, etc.).	Facilita la interoperabilidad con sistemas existentes y futuros.
Escalabilidad	Posibilidad de conectar múltiples robots y dispositivos para cobertura amplia y coordinada.	Supervisión de grandes extensiones agrícolas o industriales.

Bibliografía

Referencias

[1] H. R. Choquehuanca Huaraccallo y L. M. Llaiqui Lobon, "Robot móvil para detección de personas en lugares inaccesibles frente a desastres naturales en la Ciudad Arequipa: Caso Sismos," Repositorio Institucional, 2020. [En línea]. Disponible en: https://repositorio.ucsm.edu.pe/items/fd39cc16-2a9f-4ded-9dac-b3b2af434054.

[2] C. Sandoval, "16 De abril de 2016," *El Comercio*, 2019. [En línea]. Disponible en: https://www.elcomercio.com/cartas/aniversario-terremoto-opinion-cartas-direccion.html.

[3] R. A. G. García, "Prototipo virtual de un robot móvil multi-terreno para aplicaciones de búsqueda y rescate," *Academia.edu*, 2018. [En línea]. Disponible en: https://www.academia.edu/37411886/Prototipo_virtual_de_un_robot_m%C3%B3vil_multi_terreno_para_aplicaciones_de_b%C3%BAsqueda_y_rescate.

[4] G. Bermúdez, K. S. Novoa y W. Infante, "La robótica en actividades de búsqueda y rescate urbano. Origen, actualidad y perspectivas," *Tecnura*, [s.f.]. [En línea]. Disponible en: https://revistas.udistrital.edu.co/index.php/Tecnura/article/view/6165.

[5] P. X. de Sandoval, "25 años del Terremoto de Northridge, El aperitivo del 'Big One' en Los Ángeles," *El País*, 2019. [En línea]. Disponible en: https://elpais.com/elpais/2019/01/18/album/1547770866_044938.html#foto_gal_1.

[6] D. A. Acuña, "Visión artificial aplicada a la detección e identificación de personas en tiempo real," *Escuela Politécnica Nacional*, 2018. [En línea]. Disponible en: https://bibdigital.epn.edu.ec/bitstream/15000/20098/1/CD-9539.pdf.

[7] I. García y V. Caranqui, "La visión artificial y los campos de aplicación," *Tierra Infinita*, vol. 1, pp. 98–108, ene.–dic. 2015. [En línea]. Disponible en: https://doi.org/10.32645/26028131.76

[8] J. J. Bryson, "La última década y el futuro del impacto de la IA en la sociedad," *BBVA OpenMind*, 2014. [En línea]. Disponible en: https://es.scribd.com/document/436101507/BBVA-OpenMind\protect\penalty\z@-Joanna-J-Bryson\protect\penalty\z@-La-Ultima-Decada\protect\penalty\z@-y-El-Futuro

[9] M. R. Tapia, *Robótica Móvil*. Escudo UGTO, 2017. [En línea]. Disponible en: http://repositorio.ugto.mx/handle/20.500.12059/4898.

[10] V. R. Barrientos y J. R. García, "Robots móviles: Evolución y estado del arte," *scispace*, 2007. [En línea]. Disponible en: https://scispace.com/pdf/robots-moviles-evolucion-y-estado-del-arte-3nrav3u8w8.pdf.

[11] D. Y. Forero Quintero y M. A. Meza Calderon, "Diseño y construcción de un robot acuático," *Tesis de pregrado, Universidad Piloto de Colombia*, 2015. [En línea]. Disponible en: http://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/1063/00002151.pdf?sequence=1&isAllowed=y.

[12] P. Á. Patiño, "elCOLOMBIANO," 10 de junio de 2025. [En línea]. Disponible en: https://www.elcolombiano.com/tecnologia/drones-mas-que-juguetes-y-sin-reglas-DE1699764.

[13] R. C. Huaraccallo y L. M. L. Lobon, "Robot Móvil para Detección de Personas en Lugares Inaccesibles Frente a Desastres Naturales en la Ciudad Arequipa: Caso Sismos," 2020. [En línea]. Disponible en: https://alicia.concytec.gob.pe/vufind/Record/UCSM_7435656e3a34652ff1c1529b5cfef4e2.

[14] E. J. Carletti, "Robot nadador comprueba teorías sobre la locomoción de animales actuales y extintos - Axxón - Garrafex News," 17 de julio de 2006. [En línea]. Disponible en: https://robots-argentina.com.ar/not/06/1640120.php.

[15] "Robotnik RB-WATCHER reviews, price, use-cases, compare mobile robots...," *QVIRO*. [En línea]. Disponible en: https://qviro.com/product/robotnik/rb-watcher.

[16] Zhengzhou Defy Mechanical & Electrical Equipment Co., Ltd., "600kg Heavy Load Warehouse Robot UGV Delivery Robot Chassis ROS2." [En línea]. Disponible en: https://spanish.alibaba.com/product-detail/600kg-heavy-load-warehouse-robot-UGV-1601223916816.html.

[17] iRhine Studio, "Robot japonés Hexapod (robot de seis patas, Kondo KMR-M6), modelo 3D," [En línea]. Disponible en: https://www.turbosquid.com/es/3d-models/3d-kmr-6-hexapod-robot-kondo/949144.

[18] H. Moreno Ávalos, R. J. Saltarén Pazmiño, L. J. Puglisi, I. G. Carrera Calderón, P. F. Cárdenas Herrera y C. Álvarez, "Robótica submarina: conceptos, elementos, modelado y control," *Revista Iberoamericana de Automática e Informática Industrial*, 2014. [En línea]. Disponible en: https://oa.upm.es/38737/.

[19] L. Verdad, "La Fundación Oceanogràfic pide ayuda a una empresa murciana para liberar tiburones pintarroja," *La Verdad*, 28 de octubre de 2019. [En línea]. Disponible en: https://www.laverdad.es/ababol/ciencia/fundacion-oceanografic-pide-20191026010653-ntvo.html.

[20] A. Barrientos, L. F. Peñín, C. Balaguer y R. Alacid, *Fundamentos de Robótica* (2ª ed.). Universidad Politécnica de Madrid, [En línea]. Disponible en: https://www.casadellibro.com/libro-fundamentos-de-robotica-2-ed/9788448156367/1132459.

[21] K. Boykins, "Él construye robots espaciales para ganarse la vida," *National Geographic*, 13 de octubre de 2021. [En línea]. Disponible en: https://www.nationalgeographicla.com/espacio/el-construye-robots-espaciales-para-ganarse-la-vida.

[22] E. S. Estrella Estrella, "Diseño y construcción de un robot para la manipulación de explosivos," 2013. [En línea]. Disponible en: https://repositorio.pucesa.edu.ec/handle/123456789/918.

[23] I. González, "Así es el Vengador, el robot del Ejército que desactiva explosivos con un mando de PlayStation," *El Español*, 9 de abril de 2022. [En línea]. Disponible en: https://www.elespanol.com/omicrono/tecnologia/20220409/vengador-nuevo-robot-ejercito-desactiva-explosivos-playstation/663184011_0.html.

[24] R. E. F. Alberto y B. A. J. Manuel, "Tendencia tecnológica de robots para atención en desastres naturales," 27 de junio de 2020. [En línea]. Disponible en: http://repositorio.uts.edu.co:8080/xmlui/handle/123456789/3155.

[25] El Informador, "Presentan robot que rescató a víctimas del sismo," *El Informador: Noticias de Jalisco, México, Deportes y Entretenimiento*, 17 de octubre de 2017. [En línea]. Disponible en: https://www.informador.mx/jalisco/Presentan-robot-que-rescato-a-victimas-del-sismo-20171016-0167.html.

[26] J. R. Figueroa Olmedo, W. M. Montalvo López y M. M. Bayas Sampedro, *Cinemática y Dinámica de Robots Móviles con Ruedas*, 1ª ed., ed. Antonio Poveda G., Ediciones CILADI, marzo de 2023. [En línea]. Disponible en: https://ciladi.org/wp-content/uploads/Libro-Robots-VF3.pdf.

[27] L. D. Gómez Silva, "Implementación de un sistema de visión artificial de detección y evasión de obstáculos para un robot móvil tipo oruga," Tesis de pregrado, Universidad Autónoma de Bucaramanga, 2023. [En línea]. Disponible en: https://repository.unab.edu.co/bitstream/handle/20.500.12749/23082/Libro-ProyectoGrado%281%29%20%281%29.pdf?sequence=1&isAllowed=y.

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

[29] Y. Méndez, "La visión artificial," MV PERCEPTION, 8 de febrero de 2023. [En línea]. Disponible en: https://mvperception.com/2023/02/08/la-vision-artificial/.

[30] G. I. Viera Maza, "Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao," Tesis de pregrado, Universidad de Piura, 2017. [En línea]. Disponible en: https://pirhua.udep.edu.pe/bitstream/handle/11042/2916/IME_218.pdf?sequence=1&isAllowed=y.

[31] Amazon Web Services, Inc., "¿Qué es la visión artificial? – Explicación de la IA y el aprendizaje automático de imágenes – AWS," [En línea]. Disponible en: https://aws.amazon.com/es/what-is/computer-vision/.

[32] Azion Technologies, "¿Qué es el procesamiento de imágenes?," Azion, 26 de febrero de 2025. [En línea]. Disponible en: https://www.azion.com/es/learning/performance/que-es-el-procesamiento-de-imagenes/.

[33] H. López, "Visión artificial aplicada a la detección de personas," TFM, Máster en Inteligencia Artificial Avanzada y Aplicada, Universitat de

València, 2022. [En línea]. Disponible en: https://www.uv.es/lapeva/Thesis/TFM_2022_Hector_Lopez_compressed.pdf.

[34] R. Benítez Iglesias, G. Escudero Bakx, S. Kanaan Izquierdo y D. Masip Rodó, *Inteligencia artificial avanzada*, Manuales (e-Libro), Editorial UOC, 1ª ed., 30 de septiembre de 2013. [En línea]. Disponible en: https://openaccess.uoc.edu/bitstream/10609/140427/8/Inteligencia%20artificial%20avanzada_M%C3%B3dulo%201_Inteligencia%20artificial%20avanzada.pdf.

[35] Amazon Web Services, Inc., "¿Qué es una red neuronal?", [En línea]. Disponible en: https://aws.amazon.com/es/what-is/neural-network/.

[36] "La Máquina Oráculo," "Aprendizaje de redes neuronales," 2023. [En línea]. Disponible en: https://lamaquinaoraculo.com/deep-learning/aprendizaje-de-redes-neuronales/.

[37] Jmv, "GitHub - jmv74211/Redes_neuronales: Repositorio creado para construir una red neuronal usando el conjunto de datos MNIST. Realizado para la asignatura de inteligencia computacional del máster de ingeniería informática en Granada," *GitHub*. [En línea]. Disponible en: https://github.com/jmv74211/Redes_neuronales.

[38] «Clasificación multiclase con deep learning y Keras», 6 de octubre de 2019. [En línea]. Disponible en: https://www.ellaberintodefalken.com/2019/10/clasificacion-deep-learning-keras.html.

[39] «Todo lo que necesitas saber sobre Deep Learning: La tecnología que imita al cerebro humano». [En línea]. Disponible en: https://www.algotive.ai/es-mx/blog/todo-lo-que-necesitas-saber-sobre-deep-learning-la-tecnologia-que-imita-al-c

[40] F. Guijarro, "3. RNA. Parte I," RPubs [En línea]. Disponible en: https://rpubs.com/fraguima/lonja3.

[41] P. Bentley, «How do machine learning GANs work?», *BBC Science Focus Magazine*, 7 de octubre de 2024. [En línea]. Disponible en: https://www.sciencefocus.com/future-technology/how-do-machine-learning-gans-work

[42] E. De Redacción de la Universidad Internacional de la Rioja, «¿Qué son las redes convolucionales y para qué se usan?», *UNIR*, 4 de marzo de 2025. [En línea]. Disponible en: https://www.unir.net/revista/ingenieria/redes-neuronales-convolucionales/

[43] S. Contreras y F. De la Rosa, "Aplicación de Deep Learning en Robótica Móvil para Exploración y Reconocimiento de Objetos basados en Imágenes," memoria de conferencia, septiembre de 2016. Figura 7: Red neuronal convolucional. [En línea]. Disponible en: https://www.researchgate.net/figure/Red-neuronal-convolucional-4_fig7_308783857.

[44] Autor desconocido, "Modified VGG16 Network Structure" (Figura 2), [En línea]. Disponible en: https://www.researchgate.net/figure/Modified-VGG16-Network-Structure_fig2_357222208.

[45] IBM, "What are Convolutional Neural Networks?", IBM Think, [En línea]. Disponible en: https://www.ibm.com/think/topics/convolutional-neural-networks.

[46] C. Brokate, "Object detection using a Raspberry Pi with Yolo and SSD Mobilenet," RPubs, 6 de marzo de 2019. [En línea]. Disponible en: https://cristianpb.github.io/blog/ssd-yolo.

[47] B. Han, "The structure of the MobileNet-V1 network" (Figura 1), [En línea]. Disponible en: https://www.researchgate.net/figure/The-structure-of-the-MobileNet-V1-network_fig1_364485711.

[48] Lean Componentes, "Tipos de redes neuronales y sus aplicaciones en la empresa," Lean Componentes, [En línea]. Disponible en: https://leancomponentes.com/tipos-de-redes-neuronales/.

[49] J. Canales Luna, "YOLO Object Detection Explained: A Beginner's Guide," DataCamp, 28 de septiembre de 2024. [En línea]. Disponible en: https://www.datacamp.com/es/blog/yolo-object-detection-explained.

[50] Angel, "La importancia de una buena comunicación," Lerner Comunicación, [En línea]. Disponible en: https://lernercomunicacion.blogspot.com/2011/10/la-importancia-de-una-buena.html.

[51] Y.Fernández, "Arduino y Raspberry Pi: qué son y cuáles son sus diferencias," *Xataka Basics*, 28 de agosto de 2018. [En línea]. Disponible en: https://www.xataka.com/basics/arduino-raspberry-pi-que-cuales-sus-diferencias

[52] Sparkleo, "Arduino vs Raspberry Pi," *Medium*, 31 de marzo de 2024. [En línea]. Disponible en: https://medium.com/@sparkleo/arduino-vs-raspberry-pi-9df428eb3f4b.

[53] I. Challenger Pérez, R. Yanet Díaz y R. A. B. García, El lenguaje de programación Python / The programming language Python, 2014. [En línea].

[54] J. Moleiro, "Qué lenguaje de programación conviene aprender hoy," *LinkedIn*, [En línea]. Disponible en: https://es.linkedin.com/pulse/que-lenguaje-de-programaci%C3%B3n-conviene-aprender-hoy-jonatan-moleiro.

[55] E. A. León Zárate, Lenguaje de programación C++ aplicado a las ecuaciones no lineales, 2018. [En línea]. Disponible en: http://repositorio.unac.edu.pe/handle/20.500.12952/4435..

[56] F. D. Bianchi, Introducción a MATLAB, 2001. [En línea]. Disponible en: https://catedra.ing.unlp.edu.ar/electrotecnia/senysis/files/apuntes/ResumenMatlab2.pdf

[57] L. G. M. Beltrán, JAVA como lenguaje universal de programación, 2016. [En línea]. Disponible en: https://repository.uaeh.edu.mx/revistas/index.php/xikua/article/view/332/4434

[58] Shopify, «Yahboom,» 2024. [En línea]. Disponible en: https://category.yahboom.net/collections/robotics/products/transbot-se?variant=45278072144188

[59] N. Mechatronics, Naylamp Mechatronics SAC, 2023. [En línea]. Disponible en: https://naylampmechatronics.com/tarjetas-raspberrypi/1255-raspberry-pi-5-4gb.html#:~:text=Raspberry%20Pi%205%20es%20un,redactar%20documentos%2C%20programar%20y%20jugar

[60] Shopify, «YAHBOOM,» 2024. [En línea]. Disponible en: https://category.yahboom.net/products/md520

[61] YAHBOOM, Yahboom Electric Camera Platform, [En línea]. Disponible en: https://category.yahboom.net/products/camera-ptz

[62] Yahboom, Transbot expansion board, 2024. [En línea]. Disponible en: https://category.yahboom.net/products/transbot-expansion-board

[63] Triacs, Módulo GPS NEO-6M, [En línea]. Disponible en: https://triacs.cl/modulos/228-modulo-gps-neo-6m-.html.

[64] Ingeniería Mecafenix, ¿Qué es un sensor ultrasónico y cómo funciona?, 28 de noviembre de 2023. [En línea]. Disponible en: https://www.ingmecafenix.com/automatizacion/sensores/ultrasonico/.

[65] L. Llamas, Luis Llamas Ingeniero, desarrollador y divulgador, 2024. [En línea]. Disponible en: https://www.luisllamas.es/que-es-raspberry-pi-imager/.

[66] P. Warren, Imypass, 19 Enero 2024. [En línea]. Disponible en: https://www.imypass.com/es/gps-location/review-advanced-ip-scanner/.

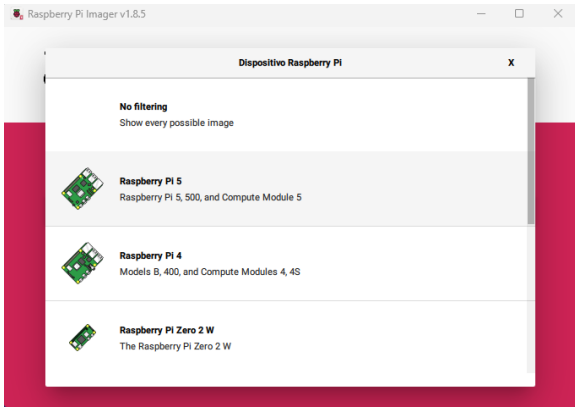
[67] R. Limited, Google Play, 2024. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.realvnc.viewer.android&hl=es_419.

[68] Pantia, «Blog Pantia,» 19 abril 2013. [En línea]. Disponible en: https://www.pontia.tech/que-es-python-para-que-srive-y-como-se-usa/.

Anexos

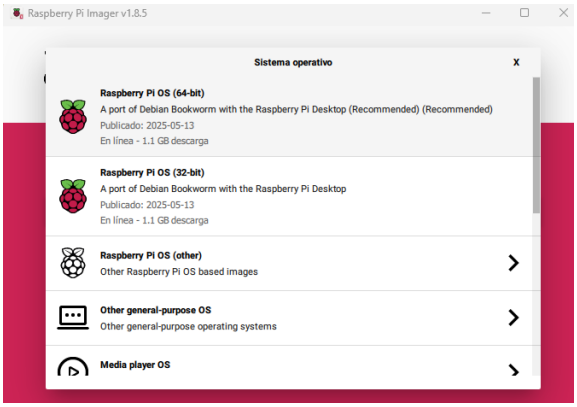
Anexo A: Instalación del sistema operativo a la Raspberry pi 5

Para instalar el sistema operativo en la placa Raspberry Pi 5, primero tenemos que instalar la herramienta Raspberry Pi Imager en tu computadora. Una vez que se ejecuta como administrador, se selecciona el dispositivo en el que se realizara la instalación.



Selección del dispositivo Raspberry PI 5

Después de elegir el dispositivo, se continua con la selección del sistema operativo. El fabricante recomienda instalar un sistema operativo de 64 bits en la Raspberry Pi 5, aunque esta elección puede variar dependiendo de los programas que se requiera utilizar y de la capacidad de almacenamiento de la tarjeta microSD que se está utilizando en el proyecto. En este caso, se eligió el sistema operativo recomendado por el fabricante.



Selección del sistema operativo Raspberry PI Os

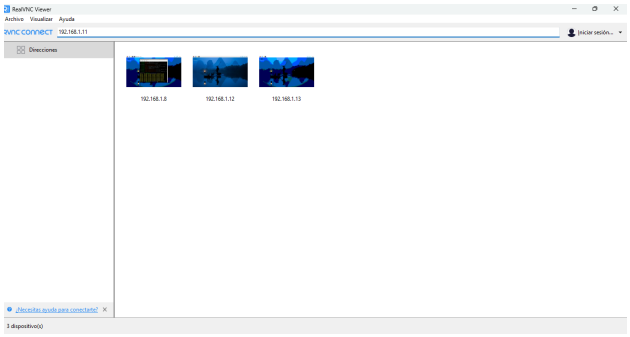
Posteriormente se inserte la tarjeta SD mediante su adaptador en la computadora para que la herramienta Raspberry Pi Imager la detecte y la muestre en su interfaz. Este paso verifica la capacidad de memoria de la tarjeta SD, que debe estar entre 16 GB y 64 GB.

Luego de realizar los pasos anteriores, se carga el sistema operativo en la tarjeta microsd. durante este proceso, se mostrará una barra de progreso con el mensaje “Escribiendo” indicando el progreso de la escritura, una vez finalizado, se retira la tarjeta del computador para insértela en la ranura microsd de Raspberry pi 5.

Instalación del sistema operativo Raspberry PI Os

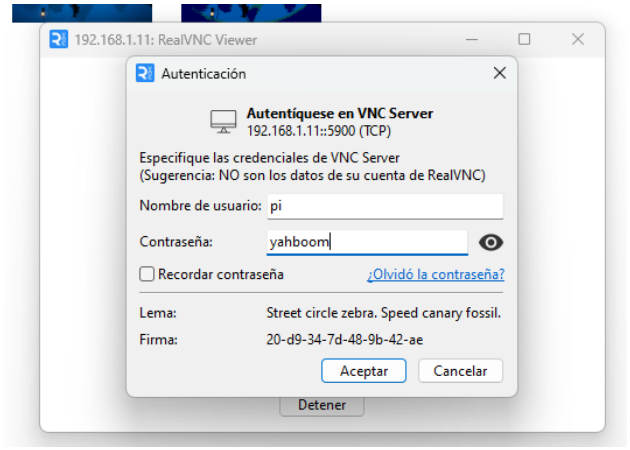
Anexo B: Visualización de la pantalla de la Raspberry mediante RealVnc

Para acceder a la Raspberry Pi 5 se utiliza la dirección IP obtenida previamente mediante el programa IP Scanner. El software escanea la red local e identifica los dispositivos conectados, incluida la Raspberry Pi. Una vez encontrada su dirección IP, se establece una conexión remota utilizando protocolos VNC Viewer Esto permite controlar y administrar fácilmente su Raspberry Pi sin conectar dispositivos externos directamente al dispositivo.



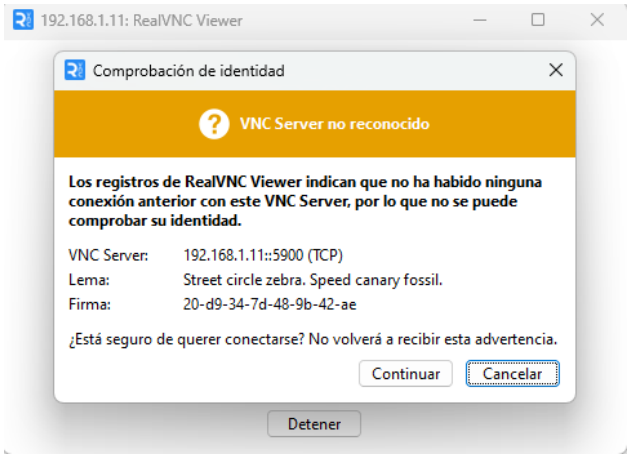
Conexión entre RealVnc y la Raspberry Pi

Antes de acceder a la Raspberry Pi a través del programa RealVNC, es necesario ingresar el nombre de usuario y la contraseña previamente configurados en el dispositivo.



Acceso remoto a la Raspberry Pi

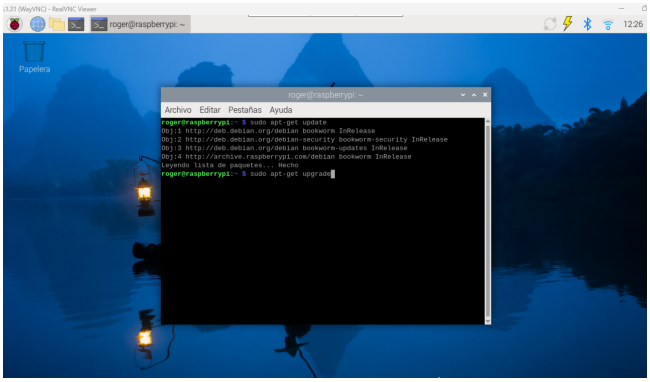
Una vez que conectado la Rasberry con el programa RealVnc, nos va a salir una ventana de comprobación, aquí simplemente aceptamos para poder acceder a la pantalla principal del dispositivo.



Comprobación de identidad para poder acceder a la Raspberry Pi

Luego de realizar los pasos anteriores, ya podremos ingresar al entorno inicial y configurar la Raspberry Pi para su posterior uso en este proyecto de investigación. Es recomendable que, al iniciar cualquier nueva instalación o configuración de software en la Raspberry Pi, se actualicen los paquetes del sistema ejecutando los siguientes comandos en la terminal:

- `sudo apt update`
- `sudo apt upgrade`



Pantalla inicial de la Raspberry Pi

Esto garantiza que el sistema operativo cuente con las últimas actualizaciones de seguridad y funcionalidad, evitando posibles errores o incompatibilidades durante el desarrollo del proyecto.

Anexo C: Instalación de las diferentes librerías

Para instalar las diversas librerías utilizadas en este proyecto de investigación, primero con el siguiente comando crearemos un entorno virtual, este entorno nos permite gestionar las diferentes dependencias, librerías y configuraciones específicas de nuestro proyecto.

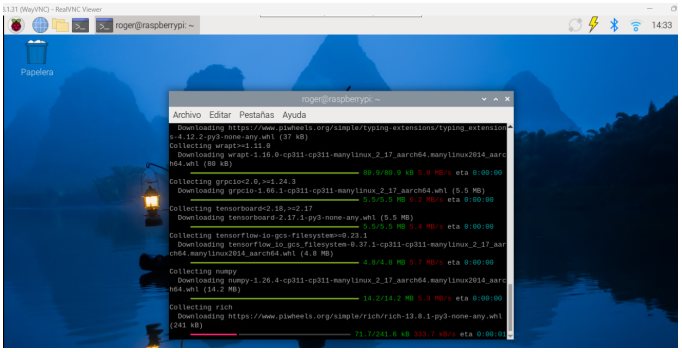
`-python -m venv Robot_tesis`

Una vez creado el entorno, para poder acceder al mismo, usaremos el siguiente comando:

`-source Robot_tesis/bin/activate`

Para instalar las diversas librerías implementadas en el proyecto lo vamos a hacer mediante *pip*, este instalador de paquetes nos permite instalar diferentes bibliotecas en Python. Para instalar una biblioteca lo haremos utilizando el siguiente comando.

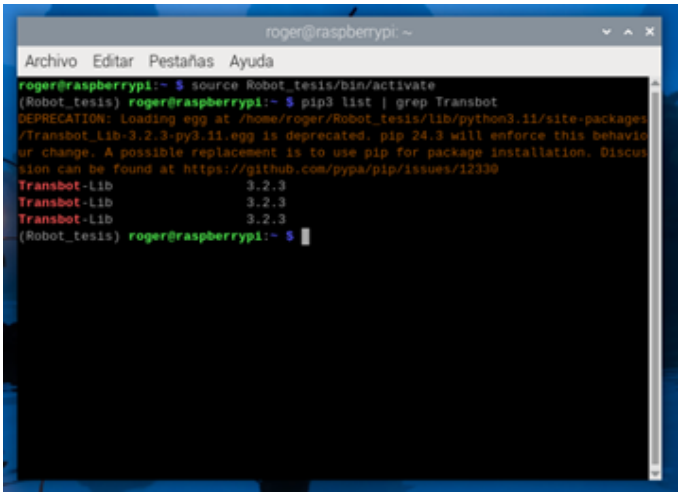
`-pip install nombre_del_paquete`



Instalación de las diferentes librerías

Anexo C.1 Instalación de la Libreria Transbot

Para la instalación de esta librería, para el funcionamiento del robot, descargamos el archivo directamente desde la página de yahboom, una vez descargado, lo descomprimimos y procederemos con su instalación.



Libreria Transbot Se

Anexo C.2 Librerías utilizadas en el proyecto

Número	librerías utilizadas
1	import numpy as np
2	import cv2
3	import tensorflow import load_model
4	import os
5	from Transbot_Lib import Transbot
6	import tkinter as tk
7	from PIL import Image, ImageTk
8	import subprocess
9	import torch
10	import numpy as np
11	import threading
12	import serial
13	import time
14	from math import atan2, degrees

Anexo D: Código de programación en python

Código Modo Manual:

```
import cv2
import numpy as np
from Transbot_Lib import Transbot
```

```
from tensorflow.keras.models import load_model
import tkinter as tk
from tkinter import Label
from PIL import Image, ImageTk  # Para manejar imagenes
   en tkinter
import subprocess
import torch
import threading

import warnings
warnings.filterwarnings("ignore", category=UserWarning,
   module="torch")

bot = Transbot()
velocidades = {
    "baja": 45,
    "media": 65,
    "alta": 100
}
velocidad_actual = velocidades["media"]

def change_speed(nueva_velocidad):
    global velocidad_actual
    velocidad_actual = velocidades[nueva_velocidad]
    print(f"Velocidad cambiada a {nueva_velocidad}: {
       velocidad_actual}")

def move_forward():
    bot.set_motor(1, velocidad_actual)
    bot.set_motor(2, velocidad_actual)
    print("Avanzando")

def move_backward():
    bot.set_motor(1, -velocidad_actual)
    bot.set_motor(2, -velocidad_actual)
    print("Retrocediendo")

def turn_left():
    bot.set_motor(1, -velocidad_actual)
    bot.set_motor(2, velocidad_actual)
    print("Girando a la izquierda")

def turn_right():
```

```
    bot.set_motor(1, velocidad_actual)
    bot.set_motor(2, -velocidad_actual)
    print("Girando a la derecha")

def stop():
    bot.set_motor(1, 0)
    bot.set_motor(2, 0)
    print("Detenido")

def move_servos(servo7_angle, servo8_angle, servo9_angle
    ):
    pulse_servo7 = int(96 + (servo7_angle / 270.0) *
        (4000 - 96))
    pulse_servo8 = int(96 + (servo8_angle / 270.0) *
        (4000 - 96))
    pulse_servo9 = int(96 + (servo9_angle / 180.0) *
        (4000 - 96))
    bot.set_uart_servo(7, pulse_servo7)
    bot.set_uart_servo(8, pulse_servo8)
    bot.set_uart_servo(9, pulse_servo9)

def update_arm_servos():
    servo7_angle = servo7_scale.get()
    servo8_angle = servo8_scale.get()
    servo9_angle = servo9_scale.get()
    move_servos(servo7_angle, servo8_angle, servo9_angle
        )

def update_camera_servos():
    servo1_position = servo1_slider.get()
    servo2_position = servo2_slider.get()
    bot.set_pwm_servo(1, servo1_position)
    bot.set_pwm_servo(2, servo2_position)

def reset_servos():
    servo7_scale.set(30)   # Reiniciar Servo 7
    servo8_scale.set(270)  # Reiniciar Servo 8
    servo9_scale.set(32)   # Reiniciar Servo 9
    update_arm_servos()    # Llama a la funcin para
        actualizar los servos

# Crea la ventana principal
root = tk.Tk()
```

```
root.title("Modo Manual - Control del Robot")

# Configura el tamao de la ventana (ancho x alto)
root.geometry("1150x570")  # Ajusta el tamao de la
    ventana

# Cambia el color de fondo
root.configure(bg="#F5F5DC")

# Crear una etiqueta de texto
label = tk.Label(root, text="Camara de supervision del
    robot", bg="#F5F5DC", font=("Arial", 14))

# Colocar la etiqueta en una posicin especfica
label.place(relx=0.50, rely=0.08, anchor='center')  #
    Centrado en la ventana

# Crea un marco para los botones del robot
frame_robot = tk.Frame(root, bg="#F5F5DC", bd=5, relief=
    "raised")
frame_robot.pack(side=tk.LEFT, padx=30, pady=30)
# Crea la etiqueta para colocarla encima de los botones
movement_label = tk.Label(frame_robot, text="Controles
    del Robot", font=("Arial", 14), bg="#F5F5DC")
movement_label.grid(row=0, column=0, columnspan=3, padx
    =5, pady=10)

# Botones de movimiento del robot
forward_button = tk.Button(frame_robot, text="Avanzar",
    command=move_forward, bg="#FFFFFF", fg="black")
backward_button = tk.Button(frame_robot, text="
    Retroceder", command=move_backward, bg="#FFFFFF", fg=
    "black")
left_button = tk.Button(frame_robot, text="Izquierda",
    command=turn_left, bg="#FFFFFF", fg="black")
right_button = tk.Button(frame_robot, text="Derecha",
    command=turn_right, bg="#FFFFFF", fg="black")
stop_button = tk.Button(frame_robot, text="Detener",
    command=stop, bg="#F5F5DC", fg="black")
vi_button = tk.Button(frame_robot, text="", bg="#f0f0f0"
    , fg="#f0f0f0", bd=0, highlightthickness=0)

# Organiza los botones en un diseo similar a una consola
```

```
forward_button.grid(row=2, column=1, padx=5, pady=5)
left_button.grid(row=3, column=0, padx=5, pady=5)
stop_button.grid(row=3, column=1, padx=5, pady=5)
right_button.grid(row=3, column=2, padx=5, pady=5)
backward_button.grid(row=4, column=1, padx=5, pady=5)
vi_button.grid(row=5, column=1, padx=5, pady=150)

# Crea el marco para los botones de velocidad
speed_frame = tk.Frame(root, bg="#F5F5DC", bd=5, relief=
    "raised")

# Establecer un tamao especfico para el frame
speed_frame.config(width=215, height=150)  # Ajusta el
    ancho y alto segn tus necesidades

# Desactivar la propagacin de tamao automtica para
    mantener el tamao fijo del frame
speed_frame.pack_propagate(False)

# Coloca el frame en la ventana
speed_frame.place(relx=0.8, rely=0.592, anchor='center')

# Crear la etiqueta dentro del Frame
label = tk.Label(speed_frame, text="Control de velocidad
    ", bg="#F5F5DC", font=("Arial", 14))
label.pack(side="top", pady=(5, 0))  # Coloca la
    etiqueta en la parte superior con un poco de espacio

# Crear los botones con el mismo ancho
baja_button = tk.Button(speed_frame, text="Baja",
    command=lambda: change_speed("baja"), bg="#FFFFFF",
    fg="black", width=10)
media_button = tk.Button(speed_frame, text="Media",
    command=lambda: change_speed("media"), bg="#FFFFFF",
    fg="black", width=10)
alta_button = tk.Button(speed_frame, text="Alta",
    command=lambda: change_speed("alta"), bg="#FFFFFF",
    fg="black", width=10)

# Coloca los botones dentro del frame
baja_button.pack(pady=2)
media_button.pack(pady=2)
alta_button.pack(pady=2)
```

```
# Crea un marco para los controles de la cmara
frame_camera = tk.Frame(root, bg="#F5F5DC", bd=5, relief
    ="raised")
frame_camera.place(relx=0.8, rely=0.26, anchor='center')

# Agrega una etiqueta en el marco de la cmara
camera_label = tk.Label(frame_camera, text="Orientacion
    de la camara", bg="#F5F5DC", font=("Arial", 14))
camera_label.pack(pady=10)
# Deslizadores para controlar los servos de la cmara
servo1_slider = tk.Scale(frame_camera, from_=0, to=180,
    orient=tk.HORIZONTAL, label="Eje Horizontal", command
    =lambda x: update_camera_servos(), bg="#F5F5DC")
servo1_slider.set(90)
servo2_slider = tk.Scale(frame_camera, from_=0, to=120,
    orient=tk.HORIZONTAL, label="  Eje Vertical", command
    =lambda x: update_camera_servos(), bg="#F5F5DC")
servo2_slider.set(45)

# Coloca los deslizadores en el marco
servo1_slider.pack(pady=10)
servo2_slider.pack(pady=10)

# Aadimos un marco adicional para los deslizadores del
    brazo robtico
frame_arm = tk.Frame(root, bg="#F5F5DC")
frame_arm.place(relx=0.15, rely=0.65, anchor='center')

# Etiqueta encima de los deslizadores
arm_label = tk.Label(frame_arm, text="Controles del
    Brazo Robotico", font=("Arial", 14), bg="#F5F5DC")
arm_label.pack(pady=(20, 5))  # Espaciado vertical

# Deslizadores para controlar los servos del brazo
    robtico
servo7_scale = tk.Scale(frame_arm, from_=30, to=225,
    orient=tk.HORIZONTAL, label="   Elevacion", command=
    lambda x: update_arm_servos(), bg="#F5F5DC")
servo7_scale.set(30)  # Posicin inicial para Servo 7
servo8_scale = tk.Scale(frame_arm, from_=147, to=270,
    orient=tk.HORIZONTAL, label="   Extension", command=
    lambda x: update_arm_servos(), bg="#F5F5DC")
```

```
servo8_scale.set(270)  # Posicin inicial para Servo 8
servo9_scale = tk.Scale(frame_arm, from_=35, to=140,
   orient=tk.HORIZONTAL, label="       Pinza", command=
   lambda x: update_arm_servos(), bg="#F5F5DC")
servo9_scale.set(35)  # Posicin inicial para Servo 9

servo7_scale.pack(pady=5)
servo8_scale.pack(pady=5)
servo9_scale.pack(pady=5)

# Botn para reiniciar los servos
reset_button = tk.Button(frame_arm, text="Reiniciar
   Servos", command=reset_servos, bg="#F5F5DC", fg="
   black")
reset_button.pack(pady=10)  # Agrega un espacio entre el
    botn y los deslizadores

def run_other_script():
    subprocess.Popen(["python", "gps3.py"])  # Asegrate
       de que el archivo est en el mismo directorio o
       proporciona la ruta completa
# Crear el marco para el boton
frame_button = tk.Frame(root, bg="#F5F5DC", bd=5, relief
   ="raised")

# Desactivar la propagacin de tamao para el frame
frame_button.pack_propagate(False)

# Establecer el tamao del frame (ancho y alto)
frame_button.config(width=215, height=100)  # Ajusta
   estos valores segn sea necesario

frame_button.place(relx=0.8, rely=0.83, anchor='center')

# Agrega una etiqueta en el marco del gps
gps_label = tk.Label(frame_button, text="Localizador gps
   ", bg="#F5F5DC", font=("Arial", 14))
gps_label.pack(pady=10)

# Crear el botn dentro del marco
run_button = tk.Button(frame_button, text="Ver ubicacion
   ", command=run_other_script, bg="#FFFFF0", fg="black"
   )
```

```
run_button.pack()


model = torch.hub.load('ultralytics/yolov5', 'custom',
    path='Redbest.pt')
model.eval()  # Modo evaluacin para mejor rendimiento

# Configuracin de la cmara
cap = cv2.VideoCapture(0)

# Configuracin de resolucin y frame rate
resize_width = 416
resize_height = 416
frame_rate = 5  # Procesar cada 5 fotogramas

# Colores y etiquetas para las clases
class_colors = {'Personas': (0, 0, 255), 'Obstaculos':
    (255, 0, 0)}
custom_labels = {'Personas': "Persona detectada", '
    Obstaculos': "Obstaculo detectado"}

frame_count = 0

# Funcin para mostrar la cmara y procesar la deteccin en
    la interfaz grfica
def show_camera():
    global frame_count
    while True:
        ret, frame = cap.read()
        if not ret:
            print("No se pudo capturar la imagen de la
                cmara.")
            break

        frame_count += 1
        # Solo procesar cada 'frame_rate' fotogramas
        if frame_count % frame_rate == 0:
            # Redimensionar la imagen para mejorar el
                rendimiento
            frame_resized = cv2.resize(frame, (
                resize_width, resize_height))

            # Realizar la deteccin
```

```
            results = model(frame_resized)

            # Obtener las predicciones
            pred = results.pred[0]
            class_names = results.names

            # Dibujar cuadros de deteccin y etiquetas
            for det in pred:
                x1, y1, x2, y2, conf, cls = det.tolist()
                class_name = class_names[int(cls)]  #
                   Actualizar class_name con la clase
                   detectada
                color = class_colors.get(class_name, (0,
                    255, 0))  # Asignar color
                label = custom_labels.get(class_name,
                   class_name)  # Asignar etiqueta

                # Dibujar el cuadro de la deteccin
                cv2.rectangle(frame_resized, (int(x1),
                   int(y1)), (int(x2), int(y2)), color,
                   2)
                cv2.putText(frame_resized, label, (int(
                   x1), int(y1) - 10), cv2.
                   FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

            # Convertir la imagen a formato compatible
               con Tkinter
            img = cv2.cvtColor(frame_resized, cv2.
               COLOR_BGR2RGB)
            img = Image.fromarray(img)
            img_tk = ImageTk.PhotoImage(img)

            # Actualizar la interfaz grfica de Tkinter
            camera_label.config(image=img_tk)
            camera_label.image = img_tk  # Mantener la
               referencia para evitar recoleccin de
               basura

        # Actualizar Tkinter para mostrar la imagen
        root.update_idletasks()
        root.update()

# Crear la etiqueta para la cmara
```

```
camera_label = tk.Label(root)
camera_label.place(relx=0.5, rely=0.5, anchor='center')
    # Centrar la etiqueta en la ventana

# Llamar a la funcin para mostrar la cmara
show_camera_thread = threading.Thread(target=show_camera
   )
show_camera_thread.start()

# Iniciar la interfaz de Tkinter
root.mainloop()

# Cerrar la cmara al finalizar
cap.release()

# Llamar a la funcin para mostrar la cmara
show_camera_thread = threading.Thread(target=show_camera
   )
show_camera_thread.start()

# Iniciar la interfaz de Tkinter
root.mainloop()

# Cerrar la cmara al finalizar
cap.release()
```

Código Modo Automático:

```
import serial
import pynmea2
import tkinter as tk
import cv2
import torch
from math import atan2, degrees
import webbrowser
import time
from Transbot_Lib import Transbot
import threading
from PIL import Image, ImageTk
from datetime import datetime
import os
import subprocess

# Cargar el modelo YOLOv5
```

```
model = torch.hub.load('ultralytics/yolov5', 'custom',
    path='Redbest.pt')
model.eval()  # Modo evaluacin para mejor rendimiento

# Configuracin de la cmara
cap = cv2.VideoCapture(0)

# Configuracin de resolucin y frame rate
resize_width = 416
resize_height = 416
frame_rate = 5  # Procesar cada 5 fotogramas

# Colores y etiquetas para las clases
class_colors = {'Personas': (0, 0, 255), 'Obstaculos':
    (255, 0, 0)}  # Rojo para Personas, Azul para
    Obstculos
custom_labels = {'Personas': "Persona detectada", '
    Obstaculos': "Obstaculo detectado"}

frame_count = 0
class_name = ''  # Mantener la variable class_name
    global
# Crear la carpeta si no existe
folder_path = "/home/roger/Archivos/Fotos_Detecciones"
if not os.path.exists(folder_path):
    os.makedirs(folder_path)

# Funcin para guardar la foto con la fecha, hora y
    coordenadas actuales en la imagen
def save_photo_with_timestamp(frame):
    global current_lat, current_lon  # Asegurarse de que
        estas variables estn accesibles

    # Obtener la fecha y hora actual
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M
        :%S')

    # Crear el nombre del archivo basado en la fecha y
        hora
    filename = f"persona_detectada_{timestamp.replace
        (':', '-')}.jpg"

    # Crear la ruta completa para guardar la imagen en
```

```
        la carpeta deseada
    filepath = os.path.join(folder_path, filename)

    # Aadir la fecha y hora en la parte inferior
    height, width, _ = frame.shape
    cv2.putText(frame, f"Fecha y Hora: {timestamp}",
        (10, height - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255,
                    0), 1)

    # Aadir coordenadas GPS en la parte superior
    if current_lat and current_lon:
        gps_text = f"Lat: {current_lat:.6f}, Lon: {
            current_lon:.6f}"
    else:
        gps_text = "GPS: Sin senal"
    cv2.putText(frame, gps_text, (10, 20), cv2.
        FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    # Guardar la imagen
    cv2.imwrite(filepath, frame)
    print(f"Foto guardada en: {filepath}")



class_name = ''
# Funcin para mostrar la cmara y procesar la deteccin en
    la interfaz grfica
def show_camera():
    global frame_count
    global class_name  # Hacemos que class_name sea
        global
    while True:
        ret, frame = cap.read()
        if not ret:
            print("No se pudo capturar la imagen de la
                cmara.")
            break

        frame_count += 1
        # Solo procesar cada 'frame_rate' fotogramas
        if frame_count % frame_rate == 0:
            # Redimensionar la imagen para mejorar el
```

```
            rendimiento
        frame_resized = cv2.resize(frame, (
            resize_width, resize_height))

        # Realizar la deteccin
        results = model(frame_resized)

        # Obtener las predicciones
        pred = results.pred[0]
        class_names = results.names

        # Dibujar cuadros de deteccin y etiquetas
        for det in pred:
            x1, y1, x2, y2, conf, cls = det.tolist()
            class_name = class_names[int(cls)]  #
                Actualizamos class_name globalmente

            # Determinar el color del cuadro y el
                texto basado en la clase
            if class_name == 'Personas':
                color = class_colors['Personas']  #
                    Rojo para persona
                label = custom_labels['Personas']  #
                     Etiqueta personalizada para
                    persona
                # Guardar foto con la fecha y hora
                    en la parte inferior
                save_photo_with_timestamp(
                    frame_resized)  # Llamar a la
                    funcin para guardar la foto
            elif class_name == 'Obstaculos':
                color = class_colors['Obstaculos']
                    # Azul para obstculo
                label = custom_labels['Obstaculos']
                     # Etiqueta personalizada para
                    obstculo
            else:
                color = (0, 255, 0)  # Verde por
                    defecto
                label = class_name  # Etiqueta por
                    defecto para otras clases

            # Dibujar el cuadro de la deteccin
```

```
                cv2.rectangle(frame_resized, (int(x1),
                    int(y1)), (int(x2), int(y2)), color,
                    2)
                cv2.putText(frame_resized, label, (int(
                    x1), int(y1) - 10), cv2.
                    FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

            # Poner la fecha y hora en la parte inferior
                de la imagen con texto ms pequeo
            height, width, _ = frame_resized.shape
            timestamp = datetime.now().strftime('%Y-%m-%
                d %H:%M:%S')
            cv2.putText(frame_resized, f"Fecha y Hora: {
                timestamp}", (10, height - 10), cv2.
                FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0),
                2)
            # Mostrar las coordenadas GPS en la parte
                superior
            if current_lat and current_lon:
              gps_text = f"Lat: {current_lat:.6f}, Lon:
                {current_lon:.6f}"
            else:
              gps_text = "GPS: Sin senal"
            cv2.putText(frame_resized, gps_text, (10,
                20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0),
                2)

            # Convertir la imagen a formato compatible
                con Tkinter
            img = cv2.cvtColor(frame_resized, cv2.
                COLOR_BGR2RGB)
            img = Image.fromarray(img)
            img_tk = ImageTk.PhotoImage(img)

            # Actualizar la interfaz grfica de Tkinter
            camera_label.config(image=img_tk)
            camera_label.image = img_tk  # Mantener la
                referencia para evitar recoleccin de
                basura

        # Actualizar Tkinter para mostrar la imagen
        root.update_idletasks()
```

```
        root.update()

# Configuracin de puertos
gps_port = '/dev/ttyUSB1'
mpu_port = '/dev/ttyUSB0'
baud_rate = 9600
baud_rate_mpu = 115200

# Inicializa conexin con Transbot
bot = Transbot()
velocidades = {"baja": 0, "media": 70, "alta": 100}
velocidad_actual = velocidades["baja"]

try:
    gps_serial = serial.Serial(gps_port, baud_rate,
       timeout=1)
    print("Conexin establecida con el mdulo GPS.")
except serial.SerialException as e:
    print(f"No se pudo conectar al GPS: {e}")
    gps_serial = None

try:
    mpu_serial = serial.Serial(mpu_port, baud_rate_mpu,
       timeout=1)
    print("Conexin establecida con el Arduino MPU.")
except serial.SerialException as e:
    print(f"No se pudo conectar al MPU: {e}")
    mpu_serial = None

# Variables globales
current_lat = 0.0
current_lon = 0.0
destination_lat = 0.0
destination_lon = 0.0
robot_heading = 0.0
obstacle_distance_front = 999.0
obstacle_distance_right = 999.0

# Funciones de movimiento del robot
def turn_left():
    bot.set_motor(1, velocidad_actual * 0.5)
    bot.set_motor(2, velocidad_actual)
    time.sleep(0.1)  # Ajusta para una correccin ms
```

```
        precisa
    stop()

def turn_right():
    bot.set_motor(1, velocidad_actual)
    bot.set_motor(2, velocidad_actual * 0.5)
    time.sleep(0.1)
    stop()

def move_forward():
    bot.set_motor(1, velocidad_actual)
    bot.set_motor(2, velocidad_actual)
    time.sleep(0.5)  # Mueve por un tiempo corto
    stop()

def move_forward_I():
    start_time = time.time()
    while time.time() - start_time < 2:
        if obstacle_distance_front < 25:
            print("Obstaculo inesperado durante avance")
            stop()
            return
        bot.set_motor(1, velocidad_actual)
        bot.set_motor(2, velocidad_actual)
        time.sleep(0.1)
    stop()

def turn_right_I():
    start_time = time.time()
    while time.time() - start_time < 2:
        bot.set_motor(1, velocidad_actual)
        bot.set_motor(2, -velocidad_actual)
        time.sleep(0.1)
    stop()

def turn_left_I():
    start_time = time.time()
    while time.time() - start_time < 2:
        bot.set_motor(1, -velocidad_actual)
        bot.set_motor(2, velocidad_actual)
        time.sleep(0.1)
    stop()
```

```
def move_backward_I():
    start_time = time.time()
    while time.time() - start_time < 1.5:
        bot.set_motor(1, -velocidad_actual)
        bot.set_motor(2, -velocidad_actual)
        time.sleep(0.1)
    stop()

def stop():
    bot.set_motor(1, 0)
    bot.set_motor(2, 0)

def update_camera_servos():
    servo1_position = 90     # Posicin fija deseada
    servo2_position = 50     # Posicin fija deseada
    bot.set_pwm_servo(1, servo1_position)
    bot.set_pwm_servo(2, servo2_position)

def move_servos(servo7_angle, servo8_angle, servo9_angle
   ):
    pulse_servo7 = int(96 + (servo7_angle / 270.0) *
       (4000 - 96))
    pulse_servo8 = int(96 + (servo8_angle / 270.0) *
       (4000 - 96))
    pulse_servo9 = int(96 + (servo9_angle / 180.0) *
       (4000 - 96))
    bot.set_uart_servo(7, pulse_servo7)
    bot.set_uart_servo(8, pulse_servo8)
    bot.set_uart_servo(9, pulse_servo9)

def update_arm_servos():
    # Posiciones fijas predeterminadas
    servo7_angle = 42     # Elevacion
    servo8_angle = 270    # Extension
    servo9_angle = 35     # Pinza
    move_servos(servo7_angle, servo8_angle, servo9_angle
       )

# Funcin para actualizar los datos del GPS
def update_gps():
    global current_lat, current_lon
    if gps_serial:
        gps_data = gps_serial.readline().decode('ascii',
```

```
            errors='replace').strip()
        if gps_data.startswith('$GPGGA') or gps_data.
           startswith('$GPRMC'):
            try:
                msg = pynmea2.parse(gps_data)
                current_lat = msg.latitude
                current_lon = msg.longitude
                gps_label.config(text=f"GPS: Latitud {
                   current_lat}, Longitud {current_lon}"
                   )
            except pynmea2.ParseError:
                print("Error al analizar datos NMEA.")
    root.after(1000, update_gps)

def calculate_direction_async():
    """Ejecuta calculate_direction en un hilo separado.
       """
    thread = threading.Thread(target=calculate_direction
       )
    thread.daemon = True  # Permite que el hilo se
       cierre cuando el programa termine
    thread.start()

def update_mpu():
    global robot_heading, obstacle_distance_front,
       obstacle_distance_right
    if mpu_serial and mpu_serial.in_waiting > 0:
        try:
            data = mpu_serial.readline().decode('utf-8')
               .strip()
            if data.startswith("H:"):
                value = float(data[2:])
                if abs(value - robot_heading) > 1:
                    robot_heading = value
                    mpu_label.config(text=f"Orientacion:
                        {robot_heading:.2f}")
                    calculate_direction_async()
            elif data.startswith("D1:"):
                obstacle_distance_front = float(data
                   [3:])
                # Aqu podras actualizar una etiqueta
                   visual
            elif data.startswith("D2:"):
```

```
                obstacle_distance_right = float(data
                    [3:])
            except ValueError:
                print(f"Error de conversion: {data}")
    root.after(5, update_mpu)

# Funcin para definir destino
def set_destination():
    global destination_lat, destination_lon
    try:
        destination_lat = float(lat_entry.get())
        destination_lon = float(lon_entry.get())
        destination_label.config(text=f"Destino: Lat {
            destination_lat}, Lon {destination_lon}")
        calculate_direction()
    except ValueError:
        destination_label.config(text="Error: Ingresa
            valores vlidos.")# Funcin para calcular y
            corregir la direccin del robot

from math import radians, sin, cos, sqrt, atan2, degrees

# Funcin para calcular la distancia entre dos
    coordenadas GPS en kilmetros
def haversine_distance(lat1, lon1, lat2, lon2):
    R = 6371  # Radio de la Tierra en kilmetros
    dlat = radians(lat2 - lat1)
    dlon = radians(lon2 - lon1)
    a = sin(dlat/2)**2 + cos(radians(lat1)) * cos(
        radians(lat2)) * sin(dlon/2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance_km = R * c
    return distance_km  # en kilmetros

# Funcin principal de clculo de direccin con control de
    llegada
def calculate_direction():
    global robot_heading

    if set_destination and obstacle_distance_front < 25:
        avoid_obstacle()
        return
```

```
    if current_lat and current_lon and destination_lat
       and destination_lon:
        dx = destination_lon - current_lon
        dy = destination_lat - current_lat
        target_angle = (degrees(atan2(dx, dy)) + 360) %
           360
        angle_difference = (target_angle - robot_heading
            + 360) % 360
        if angle_difference > 180:
            angle_difference -= 360

        # Calcular distancia restante
        distance_km = haversine_distance(current_lat,
           current_lon, destination_lat, destination_lon
           )
        distance_m = distance_km * 1000  # convertir a
           metros

        # Mostrar distancia en etiqueta (si quieres
           puedes usar otra)
        direction_label.config(text=f"Angulo destino: {
           target_angle:.2f}| Correccion: {
           angle_difference:.2f}\nDistancia restante: {
           distance_m:.2f} m")

        # Verificar si ya lleg al destino
        if distance_m < 1.0:  # Menos de 1 metro
            stop()
            direction_label.config(text="Destino
               alcanzado!")
            return

        # Movimiento segn correccin
        correction_threshold = 5
        if angle_difference > correction_threshold:
            turn_right()
        elif angle_difference < -correction_threshold:
            turn_left()
        else:
            move_forward()

ultimo_obstaculo = None
```

```
def avoid_obstacle():
    global ultimo_obstaculo
    print("Obstaculo detectado al frente. Iniciando
       maniobra de evasion.")
    stop()
    tiempo_inicio = time.time()

    # Esperar unos segundos para ver si el obstaculo
       desaparece
    while obstacle_distance_front < 25:
        print("Esperando que el obstaculo desaparezca...
           ")
        time.sleep(0.2)
        if time.time() - tiempo_inicio > 2:
            print("Obstaculo persistente. Retrocediendo
               ...")
            move_backward_I()
            break

    # Tomar decisin basada en el sensor derecho
    if obstacle_distance_right < 30:
        print("Lado derecho bloqueado. Girando a la
           izquierda.")
        turn_left_I()
    else:
        print("Lado derecho libre. Girando a la derecha.
           ")
        turn_right_I()

    print("Avanzando tras evasion.")
    move_forward_I()

    ultimo_obstaculo = time.time()
    calculate_direction()

# Funcin para abrir Google Maps con la ubicacin actual
def open_maps():  # Ahora ejecuta un script en vez de
   abrir un enlace
    try:
        subprocess.Popen(["python", "gps3.py"])
    except Exception as e:
        print(f"Error al ejecutar gps3.py: {e}")
```

```
# Interfaz grfica con Tkinter
root = tk.Tk()
# Cambia el color de fondo
root.configure(bg="#F5F5DC")
root.title("Modo Automatico - Control del Robot")
root.geometry("960x570")  # Ajusta el tamao de la
    ventana
# Marco para la cmara (izquierda)
left_frame = tk.Frame(root,bg="#F5F5DC")
left_frame.grid(row=0, column=0, padx=10, pady=10)

camera_label = tk.Label(left_frame, text="Camara", font
    =("Arial", 14))
camera_label.pack(pady=5)

# Marco para los dems elementos (derecha)
right_frame = tk.Frame(root)
right_frame.grid(row=0, column=1, padx=10, pady=10,
    sticky="n")

# Marco para los dems elementos (derecha)
right_frame = tk.Frame(root, bg="#F5F5DC", bd=5, relief=
    "raised")
right_frame.grid(row=0, column=1, padx=20, pady=80,
    sticky="n")  # Aumenta pady para bajarlo

# Agregar espacio en la parte superior para empujar todo
    hacia abajo
tk.Label(right_frame, text="Coordenadas Actuales", bg="#
    F5F5DC", font=("Arial", 12)).pack(pady=10)  #
    Espaciado extra

gps_label = tk.Label(right_frame, text="GPS: Latitud -,
    Longitud -",bg="#F5F5DC", font=("Arial", 12))
gps_label.pack(pady=5)

mpu_label = tk.Label(right_frame, text="Orientacion: -",
    bg="#F5F5DC", font=("Arial", 12))
mpu_label.pack(pady=5)

destination_label = tk.Label(right_frame, text="Destino:
    -",bg="#F5F5DC", font=("Arial", 12))
destination_label.pack(pady=5)
```

```
tk.Label(right_frame, text="Latitud Destino:",bg="#
    F5F5DC").pack()
lat_entry = tk.Entry(right_frame)
lat_entry.pack()

tk.Label(right_frame, text="Longitud Destino:",bg="#
    F5F5DC").pack()
lon_entry = tk.Entry(right_frame)
lon_entry.pack()

set_button = tk.Button(right_frame, text="Establecer
    Destino",bg="#F5F5DC", command=set_destination)
set_button.pack(pady=5)

direction_label = tk.Label(right_frame, text="Angulo al
    destino: -",bg="#F5F5DC", font=("Arial", 12))
direction_label.pack(pady=5)

maps_button = tk.Button(right_frame, text="Ver ubicacion
    ",bg="#F5F5DC", command=open_maps)
maps_button.pack(pady=5)

# Iniciar actualizacin de datos
update_gps()
update_mpu()
update_camera_servos()
update_arm_servos()

th_camera = threading.Thread(target=show_camera, daemon=
    True)
th_camera.start()
root.mainloop()
```