

Tecnológico Nacional de México
Instituto Tecnológico de Mexicali



Ingeniería en Sistemas Computacionales
Inteligencia Artificial II

Reconocimiento de Patrones con Elegoo Smart Robot

Barba Navarro Luis Rodrigo - 20490687
Geraldo Armenta Angel Uriel - 20490699
Guadalupe Lualhati Angelica Mae - 20490702

Semestre: 9°
Grupo: AMD-2105
Profesor. Olvera Gonzalez Jaime

Mexicali, Baja California a jueves, 28 de noviembre de 2024

Índice

Introducción.....	1
Objetivos generales y específicos	3
Justificación, aplicación e importancia	5
Reflexión sobre el resultado del aprendizaje.....	7
Marco teórico	8
Fundamentos de Robótica.....	8
Componentes de Hardware utilizados	11
Software y Herramientas de Programación	14
Entorno de Desarrollo y Configuración	19
Desarrollo de la actividad	22
Materiales y componentes utilizados	22
Instalación de controladores para ESP32-S3-CAM y Arduino UNO	24
Configuración del entorno de programación en Arduino IDE para el módulo ESP32-S3-CAM.....	26
Modificación del código para el módulo ESP32-S3-CAM	28
Configuración del entorno de programación en Arduino IDE para Arduino UNO .	33
Modificaciones al código de Arduino Uno para el sensor MPU6050	34
Ensamblaje del Smart Robot con cámara y Arduino UNO	37
Instalación y configuración de Raspberry Pi OS en una Raspberry Pi 5, e instalación de Docker	39
Codificación del algoritmo de reconocimiento y evasión de obstáculos	43
Codificación del algoritmo de reconocimiento y seguimiento de esferas de colores con OpenCV	53
Creación de contenedores Docker para despliegue de las aplicaciones	57
Despliegue de contenedores en Raspberry Pi	60
Resultados	64
Funcionamiento del algoritmo de reconocimiento y evasión de obstáculos	64
Funcionamiento del algoritmo de reconocimiento y seguimiento de esferas de colores	68
Conclusiones.....	72
Referencias bibliográficas	74

Tabla de figuras

Ilustración 6. 1. 1 - Materiales de Hardware.....	22
Ilustración 6. 1. 2 - Materiales de Software	23
Ilustración 6. 2. 1 - ESP32-S3-CAM Elegoo.....	24
Ilustración 6. 2. 2 - Verificación de controlador de cámara.....	25
Ilustración 6. 3. 1 - Paquetería JSON de ESP32.....	26
Ilustración 6. 3. 2 - Configuración de ESP32-S3	27
Ilustración 6. 4. 1 - Configuración de seguridad de la cámara	29
Ilustración 6. 4. 2 - Ajuste de frecuencia	29
Ilustración 6. 4. 3 - Líneas comentadas de heartbeats.....	30
Ilustración 6. 4. 4 - Configuración para mejorar calidad de la cámara.....	31
Ilustración 6. 4. 5 - Ejemplo de la mejora de los fotogramas.....	32
Ilustración 6. 5. 1 - Selección de Arduino UNO	33
Ilustración 6. 6. 1 - Función de sensor MPU	34
Ilustración 6. 6. 2 - Importación de función.....	34
Ilustración 6. 6. 3 - Modificación de análisis de datos	35
Ilustración 6. 6. 4 - Implementación de nuevo switch case	35
Ilustración 6. 7. 1 - Materiales para el Smart Robot	37
Ilustración 6. 7. 2 - Producto final del Smart Robot	38
Ilustración 6. 8. 1 - Raspberry Pi Imager	39
Ilustración 6. 8. 2 - Primer vistazo de Raspberry OS.....	40
Ilustración 6. 9. 1 - Importación de módulos.....	44
Ilustración 6. 9. 2 - Función CMD.....	45
Ilustración 6. 9. 3 - Comunicación con Smart Robot	46

Ilustración 6. 9. 4 - Función evade_obstaculos	47
Ilustración 6. 9. 5 - Función ciclo principal.....	48
Ilustración 6. 9. 6 - Función de captura	49
Ilustración 6. 9. 7 - Aplicación Flask.....	50
Ilustración 6. 9. 8 - Web Socket para la vista	51
Ilustración 6. 9. 9 - Etiqueta para mostrar trasmisión de video.....	52
Ilustración 6. 10. 1 - Reconocimiento de contorno	54
Ilustración 6. 10. 2 - Cálculo de distancia y ángulo hacia la pelota	54
Ilustración 6. 10. 3 - Función find_ball.....	55
Ilustración 6. 10. 4 - Función track_ball.....	56
Ilustración 6. 11. 1 - Dockerfile.app1 y Dockerfile.app2	58
Ilustración 6. 11. 2 - Docker Compose	59
Ilustración 6. 12. 1 – Acceso a directorio de proyecto	60
Ilustración 6. 12. 2 - Acceso a directorio de Docker	60
Ilustración 6. 12. 3 - Generación de imágenes de Dockerfile	61
Ilustración 6. 12. 4 - Acceso a primera aplicación	62
Ilustración 6. 12. 5 - Acceso a segunda aplicación	62
Ilustración 6. 12. 6 - Acceso a la segunda aplicación.....	63
Ilustración 7. 1. 1 - Analizando entorno	64
Ilustración 7. 1. 2 - Gira cabezal a la izquierda	65
Ilustración 7. 1. 3 - Gira cabezal a la derecha	65
Ilustración 7. 1. 4 - Avanza hacia adelante.....	66
Ilustración 7. 1. 5 - Encuentra obstáculo, gira a la izquierda	66
Ilustración 7. 1. 6 - Encuentra obstáculo nuevamente.....	67
Ilustración 7. 2. 1 - Identifica el entorno.....	68
Ilustración 7. 2. 2 - Identifica la pelota roja	69
Ilustración 7. 2. 3 - Realiza el seguimiento de la pelota roja	69
Ilustración 7. 2. 4 - Llega hasta la distancia objetivo	70
Ilustración 7. 2. 5 - Encuentra pelota azul	70

Ilustración 7. 2. 6 - Verifica si es del color objetivo..... 71

Ilustración 7. 2. 7 - Ignora la pelota azul..... 71

Introducción

El avance de la inteligencia artificial y la robótica autónoma ha revolucionado la forma en que los sistemas interactúan con su entorno, abriendo nuevas posibilidades en áreas como la automatización, la exploración y la seguridad. Este proyecto tiene como finalidad desarrollar un sistema robótico capaz de reconocer, seguir y evadir objetos mediante técnicas avanzadas de visión por computadora, aplicando algoritmos diseñados para la toma de decisiones en tiempo real.

El sistema propuesto integra un Smart Robot equipado con un módulo ESP32-S3-CAM, sensores ultrasónicos y una Raspberry Pi configurada con Raspbian Bookworm 12. Este entorno es gestionado a través de tecnologías como Python 3.11, OpenCV, NumPy y Flask, optimizadas mediante contenedores en Docker para garantizar portabilidad y rendimiento. Además, se realizaron ajustes personalizados en el código fuente del módulo ESP32-S3-CAM para mejorar la calidad de imagen y la estabilidad de la conexión.

El proyecto combina dos funciones principales: **el reconocimiento y seguimiento de esferas de colores y la evasión de obstáculos**. Utilizando el espacio de color HSV, el robot identifica esferas según su tonalidad (por ejemplo, rojo) y las diferencia de otros objetos o colores que podrían considerarse obstáculos. El sistema calcula distancias y ángulos relativos mediante geometría aplicada, lo que permite al robot ajustar su trayectoria de forma autónoma para perseguir o evitar objetos según corresponda.

Por otra parte, la función de evasión de obstáculos garantiza la seguridad del desplazamiento del robot. Este analiza continuamente su entorno a través de sensores ultrasónicos, tomando decisiones rápidas para evitar colisiones y ajustar su dirección según las condiciones del espacio. Estas funcionalidades trabajan en conjunto para ofrecer un comportamiento autónomo y dinámico, simulando un nivel básico de inteligencia artificial.

Este proyecto refleja una integración práctica de los conceptos adquiridos en la carrera de Ingeniería en Sistemas Computacionales, destacando habilidades en

programación, diseño de sistemas embebidos, inteligencia artificial y visión por computadora. El desarrollo de este sistema demuestra no solo la viabilidad de estas tecnologías en aplicaciones del mundo real, sino también su potencial para ser adaptadas y mejoradas en proyectos futuros que requieran soluciones autónomas e inteligentes.

Objetivos generales y específicos

Objetivo General

Implementar un robot autónomo capaz de detectar, seguir esferas de colores específicos y evitar obstáculos, integrando visión por computadora, control dinámico e inteligencia artificial para un rendimiento eficiente en tiempo real.

Objetivos Específicos

1. Desarrollar un sistema de detección de colores utilizando OpenCV:

- Implementar un algoritmo de visión por computadora basado en OpenCV, utilizando el espacio de color HSV para identificar esferas de colores específicos (rojo, verde, azul) en las imágenes capturadas por la cámara ESP32-S3-CAM.

2. Implementar seguimiento dinámico de objetos:

- Usar OpenCV para calcular la trayectoria hacia la esfera detectada, ajustando la dirección y velocidad del robot en tiempo real para seguir el objeto en movimiento.

3. Desarrollar un sistema de evasión de obstáculos utilizando sensores ultrasónicos:

- Integrar sensores ultrasónicos para detectar obstáculos en el entorno del robot y desarrollar un algoritmo de evasión de obstáculos que permita al robot cambiar de dirección o retroceder en función de la proximidad de los objetos detectados.

4. Optimizar el procesamiento de imágenes con OpenCV:

- Mejorar la captura, segmentación y análisis de imágenes mediante el uso de funciones avanzadas de OpenCV, como el desenfoque, la conversión a HSV, y la detección de contornos, para garantizar una detección precisa y eficiente de los objetos.

5. Desarrollar una interfaz web de control y monitoreo:

- Crear una aplicación web utilizando tecnologías como Flask y WebSockets, para visualizar imágenes en tiempo real y controlar el robot de manera remota, integrando las imágenes procesadas por OpenCV.

6. Validar el sistema en escenarios reales:

- Realizar pruebas de funcionamiento utilizando OpenCV en condiciones reales para asegurar la precisión en la detección y seguimiento de objetos, así como la evasión de obstáculos en diferentes entornos.

7. Garantizar la robustez y estabilidad del sistema:

- Implementar mecanismos de manejo de errores y optimización de recursos para asegurar que el sistema, impulsado por OpenCV y otros componentes, funcione de manera continua y confiable.

Justificación, aplicación e importancia

Justificación

El desarrollo de un sistema autónomo que integre visión por computadora y sensores para la interacción con el entorno representa un avance significativo en la robótica aplicada. La capacidad de un robot para detectar y seguir esferas de colores específicos utilizando OpenCV, junto con la habilidad de evitar obstáculos gracias a sensores ultrasónicos, ofrece una solución eficiente en aplicaciones como la navegación autónoma, la recolección de objetos, y la interacción dinámica con el entorno. Este proyecto se justifica por su potencial para mejorar la autonomía y versatilidad de los robots, optimizando su desempeño en tareas de seguimiento y evasión en escenarios complejos, como los que pueden encontrarse en un hogar o en un entorno industrial.

Además, este tipo de tecnologías es cada vez más relevante en la industria 4.0, donde los robots autónomos pueden realizar tareas repetitivas o peligrosas sin intervención humana. La integración de estas capacidades en un sistema robusto y flexible proporciona una base para el desarrollo de soluciones más avanzadas en robótica.

Aplicación

El sistema desarrollado tiene aplicaciones diversas en múltiples campos:

1. **Robótica Autónoma:** En entornos de navegación donde el robot debe seguir un objeto específico, como en sistemas de recolección de productos en almacenes o en robots de servicio.
2. **Automatización Industrial:** Los robots pueden ser utilizados para realizar tareas repetitivas de seguimiento de objetos o manipulación en fábricas, mejorando la productividad y reduciendo el riesgo de accidentes.
3. **Educación en Robótica:** Este proyecto puede servir como una herramienta educativa para estudiantes de ingeniería en sistemas computacionales, electrónica y robótica, proporcionando una base para aprender sobre visión por computadora, control de robots y programación de sistemas autónomos.
4. **Interacción en Entornos Dinámicos:** El sistema es útil en aplicaciones como robots para el hogar, donde es necesario que el robot interactúe con objetos y

realice tareas como la limpieza, evitando obstáculos mientras sigue objetivos específicos.

Importancia

La importancia de este proyecto radica en la implementación de una solución accesible y práctica que combina visión por computadora y sensores para crear un sistema autónomo altamente eficiente. El uso de OpenCV para el procesamiento de imágenes, junto con los sensores ultrasónicos para la evasión de obstáculos, permite un desarrollo de robots más inteligentes y capaces de interactuar de manera autónoma con su entorno.

Este proyecto contribuye a la evolución de los sistemas autónomos al proporcionar una base sólida que puede ser extendida a aplicaciones más complejas, como la integración con inteligencia artificial para mejorar la toma de decisiones en tiempo real. Además, su capacidad de realizar tareas en entornos no estructurados incrementa la flexibilidad y la adaptabilidad de los robots, lo que puede impactar positivamente en sectores como la asistencia a personas con discapacidades, la logística de almacenes y la automatización en general.

Reflexión sobre el resultado del aprendizaje

A lo largo del desarrollo del proyecto, se logró aplicar de manera efectiva los conceptos teóricos y prácticos aprendidos en el curso de Inteligencia Artificial II. Este proyecto representó un desafío significativo al requerir la integración de conocimientos de robótica, programación y sistemas embebidos para el diseño y construcción del Elegoo Smart Robot Car V4.0.

En primer lugar, se evidenció un aprendizaje profundo en el manejo de hardware y software. La instalación y configuración de controladores, como los de ESP32-S3-CAM y Arduino UNO, no solo permitió una correcta conexión entre los dispositivos, sino que también fortaleció la comprensión sobre cómo estos controladores optimizan la comunicación entre el hardware y el software. Este aprendizaje es esencial para proyectos futuros, ya que fomenta habilidades técnicas aplicables a una variedad de entornos tecnológicos.

Asimismo, el proyecto permitió explorar herramientas avanzadas de programación, como Python 3.11 Slim y OpenCV, destacando su importancia en tareas como el procesamiento de imágenes y el reconocimiento de patrones. La implementación de algoritmos para mejorar la calidad de las imágenes capturadas por el módulo ESP32-S3-CAM fue un ejemplo práctico de cómo los sistemas de visión artificial pueden integrarse con hardware para resolver problemas complejos.

Por último, un aspecto especialmente destacado fue la capacidad de integrar diversos componentes de hardware y software en un sistema funcional único. Este proceso implicó poner en práctica habilidades avanzadas de resolución de problemas y fomentar una colaboración efectiva en equipo, particularmente en etapas clave como el ensamblaje del robot y la refinación del código para optimizar la interacción entre sensores y controladores.

Marco teórico

Fundamentos de Robótica

Robótica - es una disciplina multidisciplinaria que combina ingeniería mecánica, electrónica, informática y la inteligencia artificial para diseñar, construir y programar robots capaces de realizar tareas de manera autónoma o semiautónoma. Según Margaret Rouse, “la robótica es el campo interdisciplinario enfocado en el diseño, construcción y operación de robots, incluyendo avances en inteligencia artificial y análisis de datos, lo que permite a estas máquinas interactuar con su entorno de manera más inteligente y efectiva” (Rouse, 2024).

Por otro lado, Robotesfera describe que “la robótica estudia el diseño, construcción y aplicación de robots, elementos que tienen la finalidad de sustituir a los humanos o ayudarlos en tareas complejas o repetitivas, como en procesos industriales o en exploraciones espaciales” (Robotesfera, 2023). Este enfoque subraya la capacidad de la robótica para extender las habilidades humanas más allá de sus limitaciones físicas o cognitivas.

Sebastián Vidal, en Tecnobits, complementa al afirmar que “la robótica no solo se centra en la mecánica de los robots, sino también en su programación, su capacidad para interactuar con el entorno, y su integración en sistemas más amplios, como fábricas inteligentes o aplicaciones médicas avanzadas” (Vidal, 2023). Este enfoque resalta la evolución de la robótica hacia aplicaciones más complejas e innovadoras.

Inteligencia artificial en robótica - se refiere a la aplicación de técnicas y algoritmos de IA en sistemas robóticos para dotarlos de capacidades avanzadas de percepción, toma de decisiones y aprendizaje. Según IBM, "la IA permite a las máquinas aprender de la experiencia, adaptarse a nuevas entradas y realizar tareas similares a las humanas" (IBM, s.f.). Esto permite a los robots operar de manera autónoma en entornos complejos y dinámicos.

La Organización Internacional de Normalización (ISO) define la inteligencia artificial como "un campo técnico y científico dedicado al sistema de ingeniería que genera resultados como contenido, previsiones, recomendaciones o decisiones para un conjunto determinado de objetivos definidos por el ser humano" (ISO, s.f.). En robótica, esta integración mejora significativamente la capacidad de los robots para adaptarse y responder a diversas situaciones en tiempo real.

Por otro lado, el Instituto Andaluz de Tecnología señala que "uno de los objetivos más ambiciosos de la ciencia es conseguir una inteligencia artificial general lo más parecida posible a la humana. Aquí es donde también entra en juego la robótica, combinando hardware avanzado con algoritmos inteligentes" (Instituto Andaluz de Tecnología, s.f.). Esta combinación está transformando diversas industrias, como la manufactura, la atención médica y la logística, al automatizar tareas complejas y mejorar la interacción entre humanos y máquinas.

Reconocimiento de patrones - es una disciplina interdisciplinaria que combina herramientas matemáticas, algoritmos computacionales y metodologías de análisis para identificar, analizar y clasificar regularidades o estructuras en conjuntos de datos. Estos patrones pueden encontrarse en formas diversas como imágenes, señales acústicas, textos y datos biométricos. Según CEUPE, "el reconocimiento de patrones es una disciplina encargada de identificar figuras, reconocer formas o leer patrones, con la finalidad de recopilar información sobre el objeto que se está estudiando y asignarlo a un grupo o clase" (CEUPE, s.f.).

MathWorks define el reconocimiento de patrones como "un proceso que consiste en utilizar algoritmos informáticos para clasificar datos de entrada en objetos, clases o categorías, en base a sus características principales o elementos constantes" (MathWorks, s.f.). Este enfoque permite a los sistemas automatizar la clasificación y el análisis de datos, facilitando la toma de decisiones en diversos contextos tecnológicos.

Por otro lado, AcademiaLab explica que "el reconocimiento de patrones es la asignación de una etiqueta a un valor de entrada dado, permitiendo clasificar la información para extraer características clave y tomar decisiones fundamentadas"

(AcademiaLab, s.f.). Este proceso utiliza técnicas avanzadas como aprendizaje supervisado, aprendizaje no supervisado y redes neuronales para analizar grandes volúmenes de datos de forma eficiente.

Visión por computadora - es una disciplina que pertenece al ámbito de la inteligencia artificial, cuyo propósito principal es desarrollar sistemas capaces de interpretar y comprender el contenido de imágenes, videos y otros datos visuales provenientes del entorno. Según Manu Duque, “la visión por computadora permite extraer información visual útil y convertirla en datos procesables, posibilitando a las computadoras identificar objetos, analizar imágenes y realizar tareas complejas que requieren interpretación visual” (Duque, s.f.). Este campo emula parcialmente la percepción visual humana, aunque con capacidades ampliadas, como la rapidez y precisión en el análisis de grandes volúmenes de datos.

De manera complementaria, EDS Robotics explica que “la visión por computador abarca toda una disciplina de estudio dentro de la robótica e informática, cuyos avances han permitido el diseño de soluciones automatizadas inteligentes, dinámicas y versátiles” (EDS Robotics, 2022). Este enfoque combina técnicas avanzadas, como redes neuronales convolucionales y aprendizaje profundo, que facilitan la identificación de patrones y la toma de decisiones basadas en datos visuales. Además, su impacto es significativo en áreas donde la velocidad y la precisión son críticas.

Por otro lado, Daniel Nelson señala que “los sistemas de visión por computadora funcionan de manera muy similar al sistema visual humano, primero discerniendo los bordes de un objeto y luego uniendo estos bordes en la forma del objeto” (Nelson, 2023). Esta capacidad no solo permite la automatización de tareas visuales, sino que también asegura altos niveles de exactitud en procesos complejos, tales como la conducción autónoma o el análisis médico.

Procesamiento de imágenes - es una disciplina que se enfoca en la manipulación y análisis de imágenes digitales mediante técnicas computacionales, con el propósito de mejorar su calidad, extraer información relevante y facilitar su interpretación. En primer lugar, Margaret Rouse señala que “el procesamiento de

imágenes implica el uso de diversas técnicas para procesar o mejorar imágenes, utilizando operaciones matemáticas o algoritmos como herramientas para el procesamiento de una imagen" (Rouse, 2019). Esto destaca su importancia en la mejora visual y en la extracción de datos útiles.

Por otro lado, Aditya Kumar explica que "el procesamiento de imágenes es el proceso de transformar una imagen en una forma digital y realizar ciertas operaciones para obtener información útil de ella" (Kumar, 2023). Esta definición resalta la capacidad del procesamiento de imágenes para convertir datos visuales en elementos procesables mediante herramientas computacionales avanzadas.

Además, Ross Jukes agrega que "el procesamiento de imágenes cubre todo tipo de formas de modificar y mejorar imágenes, no solo para hacerlas más atractivas para compartir en línea, sino también para utilizar matemáticas y trucos informáticos especiales para cambiar la apariencia de las imágenes" (Jukes, 2024). De esta manera, esta tecnología no solo tiene aplicaciones estéticas, sino que también es esencial en sectores como la medicina, la astronomía y la seguridad.

Componentes de Hardware utilizados

Elegoo Smart Robot Car V4.0 - es un kit de robótica basado en la plataforma Arduino, diseñado para aprender programación y ensamblaje de hardware de manera práctica y divertida. Según Elegoo, "el kit incluye funciones avanzadas como seguimiento de líneas, evitación de obstáculos, control por infrarrojos y transmisión de video en tiempo real a través de Wi-Fi" (ELEGOO, s.f.). Este robot es ideal para principiantes y entusiastas de la robótica que desean explorar aplicaciones prácticas de programación.

Una de las características más destacadas es su cámara FPV (vista en primera persona), que permite transmitir imágenes en tiempo real, enriqueciendo la experiencia de usuario y fomentando el aprendizaje interactivo (Vaidehi Dalmia, 2023). Además, el kit es altamente personalizable, gracias a su compatibilidad con el entorno de desarrollo Arduino, que ofrece un enfoque de código abierto para la programación y ajustes específicos.

El ELEGOO Smart Robot Car V4.0 también incluye una batería recargable de 2000 mAh, que proporciona una autonomía considerable para sesiones prolongadas de aprendizaje (ELEGOO, s.f.). Según Vaidehi Dalmia, “este robot no solo es educativo, sino que también demuestra cómo los componentes electrónicos trabajan en conjunto para realizar funciones complejas como evitar obstáculos y seguir trayectorias específicas” (Dalmia, 2023).

Raspberry Pi 5 - es la más reciente iteración de la serie de microcomputadoras desarrolladas por la Fundación Raspberry Pi, diseñada para proporcionar un rendimiento superior y mayor capacidad de expansión. Según Javier Pastor, “la nueva Raspberry Pi 5 llega presumiendo de un potente chip propio y más capacidad de expansión que nunca” (Pastor, 2023). Esto la posiciona como una herramienta ideal para proyectos tecnológicos avanzados.

En cuanto a sus especificaciones técnicas, la Raspberry Pi 5 incluye un procesador Broadcom BCM2712 de cuatro núcleos ARM Cortex-A76 a 2,4 GHz, el cual “proporciona una mejora significativa en el rendimiento computacional en comparación con modelos anteriores” (Long, s.f.). Asimismo, está disponible en versiones de 4 GB y 8 GB de RAM LPDDR4X-4267, lo que amplía su capacidad para ejecutar aplicaciones más exigentes.

Una característica clave es la incorporación de un conector PCIe 2.0 x1, que según Element14 Community, “permite conectar periféricos de alta velocidad, como unidades de almacenamiento NVMe, ofreciendo a los usuarios una mayor flexibilidad para proyectos personalizados” (Element14 Community, 2023). Además, incluye soporte para doble salida HDMI 4K a 60 Hz y dos puertos USB 3.0, mejorando significativamente las opciones de conectividad y visualización.

En términos de consumo energético, la Raspberry Pi 5 requiere una fuente de alimentación de 5V/5A DC. Según la documentación oficial de Raspberry Pi, “este incremento en los requisitos de energía está relacionado con las mejoras en rendimiento y las nuevas capacidades del hardware” (Raspberry Pi, 2024).

ESP32-S3-CAM - es un módulo de desarrollo avanzado que combina un microcontrolador ESP32-S3 con conectividad Wi-Fi y Bluetooth, junto con una cámara de alta resolución. Según Espressif Systems, “el ESP32-S3 incluye un procesador Xtensa® LX7 de doble núcleo que opera a 240 MHz, lo que permite un rendimiento optimizado para aplicaciones intensivas en datos” (Espressif Systems, s.f.). Este módulo está diseñado específicamente para aplicaciones de visión artificial e Internet de las Cosas (IoT), ofreciendo una solución integral para desarrolladores.

Una de sus características clave es su cámara de 5 megapíxeles, capaz de capturar imágenes con una resolución de 2592×1944 píxeles, lo que lo hace ideal para tareas como reconocimiento de imágenes y códigos QR. Como describe ArduCam, “la cámara integrada, junto con el micrófono digital, ofrece capacidades de captura de datos visuales y sonoros de alta calidad en un módulo compacto” (ArduCam, s.f.).

Además, el ESP32-S3-CAM incluye 512 KB de SRAM y soporte para (Uno R3, n.d.) hasta 8 MB de PSRAM, lo que amplía su capacidad para gestionar tareas complejas de procesamiento de imágenes.

En términos de conectividad, el módulo ofrece 45 pines GPIO programables y soporta interfaces como SPI, I2C, I2S y UART, lo que facilita la integración con otros dispositivos. Esto lo convierte en una solución versátil para proyectos de vigilancia inalámbrica, dispositivos portátiles y aplicaciones de reconocimiento facial.

Arduino UNO - es una de las placas de desarrollo más populares en el ámbito de la electrónica y la programación, diseñada para facilitar la creación de prototipos y proyectos interactivos. Esta placa está basada en el microcontrolador ATmega328P, un componente que ofrece un equilibrio óptimo entre rendimiento y simplicidad. Según Arduino, “Arduino UNO es una placa de microcontrolador basada en el ATmega328P. Tiene 14 pines digitales de entrada/salida (de los cuales 6 pueden usarse como salidas PWM), 6 entradas analógicas, un resonador cerámico de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP y un botón de reinicio” (Arduino, s.f.). Estas características hacen que el Arduino UNO sea ideal para

principiantes y proyectos educativos, así como para aplicaciones avanzadas en el ámbito profesional.

En términos de energía, el Arduino UNO opera a un voltaje de 5V y puede ser alimentado de dos formas: a través de una conexión USB o mediante una fuente de alimentación externa con un rango de entre 7 y 12V. Esta flexibilidad de alimentación permite su uso en una amplia variedad de entornos y aplicaciones. Según Arduino, "la placa cuenta con una memoria flash de 32 KB, de los cuales 0.5 KB son utilizados por el gestor de arranque, 2 KB de SRAM y 1 KB de EEPROM" (Arduino, s.f.). Esto proporciona suficiente capacidad para almacenar y ejecutar programas de tamaño moderado, adecuados para proyectos como automatización básica, sistemas de sensores y dispositivos IoT.

Una de las principales ventajas del Arduino UNO es su accesibilidad y compatibilidad con una amplia gama de sensores, módulos y actuadores. Por ejemplo, los pines PWM permiten controlar dispositivos como servomotores, y sus entradas analógicas facilitan la lectura de señales de sensores ambientales. Además, la comunidad de Arduino ofrece una extensa documentación, recursos educativos y bibliotecas de software que simplifican el desarrollo de proyectos complejos.

Asimismo, el Arduino UNO es una plataforma de hardware libre, lo que significa que su diseño y esquemáticos están disponibles públicamente. Esto fomenta la innovación y permite personalizar el hardware según las necesidades específicas del usuario.

Software y Herramientas de Programación

Raspbian Bookworm 12 - es conocido también como Raspberry Pi OS Bookworm, es la versión más reciente del sistema operativo oficial para las computadoras Raspberry Pi, basado en Debian 12 "Bookworm". Este sistema operativo se caracteriza por introducir mejoras en rendimiento, seguridad y funcionalidad. Según la Fundación Raspberry Pi, "la nueva versión de Raspberry Pi OS se basa en Debian 12, lo que aporta una serie de actualizaciones de paquetes y mejoras en el sistema" (Raspberry Pi, 2023).

Una de las actualizaciones clave es la adopción de Wayland como servidor gráfico predeterminado, reemplazando a X11. Esto permite una mejor gestión de los recursos gráficos y una experiencia de usuario más fluida. Andrew Heinzman señala que “Raspberry Pi OS ahora utiliza Wayland por defecto, lo que mejora el rendimiento gráfico y la seguridad del sistema” (Heinzman, 2023).

Otra mejora significativa es la incorporación de PipeWire como servidor de audio, sustituyendo a PulseAudio. Este cambio proporciona una gestión de audio más eficiente y una menor latencia, beneficiando a aplicaciones multimedia. Según Michael Larabel, “la actualización a PipeWire proporciona una mejor experiencia de audio, especialmente en aplicaciones que requieren baja latencia” (Larabel, 2023).

Además, Raspbian Bookworm incluye una versión optimizada de Mozilla Firefox, que ofrece una alternativa al navegador Chromium. Esta versión de Firefox ha sido adaptada para funcionar eficientemente en el hardware de Raspberry Pi. Según la Fundación Raspberry Pi, “ahora ofrecemos una versión de Firefox optimizada para Raspberry Pi, proporcionando a los usuarios más opciones en navegadores web” (Raspberry Pi, 2023).

Python 3.11 Slim - es una versión optimizada del lenguaje de programación Python 3.11, diseñada específicamente para entornos de contenedores. Esta variante de la imagen oficial de Python se caracteriza por su reducido tamaño y la eliminación de paquetes adicionales que no son esenciales para la ejecución básica del lenguaje. Según Docker, “la imagen Slim elimina componentes no críticos del sistema base Debian, dejando solo los paquetes mínimos necesarios para que Python funcione correctamente” (Docker, s.f.). Esta característica la hace ideal para implementaciones en las que el tamaño de la imagen y el rendimiento son factores clave.

Una de las principales ventajas de Python 3.11 Slim es su capacidad para facilitar despliegues más rápidos y un uso eficiente del almacenamiento en aplicaciones basadas en contenedores. Sin embargo, esta optimización tiene ciertas limitaciones. Como indica Docker, “algunos paquetes Python que dependen de compilación desde código fuente pueden no instalarse correctamente debido a la

ausencia de bibliotecas comunes” (Docker, s.f.). Por lo tanto, esta versión es más adecuada para proyectos que utilizan paquetes disponibles en formato binario o que tienen requisitos mínimos de dependencia.

Python 3.11 Slim también incluye las mejoras propias de la versión 3.11 del lenguaje. Según la Python Software Foundation, “Python 3.11 introduce un aumento significativo en el rendimiento, logrando una velocidad hasta un 60% superior en comparación con versiones anteriores, gracias a la optimización del intérprete” (Python Software Foundation, 2022). Esto lo convierte en una opción no solo eficiente en términos de espacio, sino también poderosa en rendimiento.

Además, esta versión es ampliamente utilizada en entornos de producción donde la eficiencia de recursos y la seguridad son prioritarias. Debido a su reducido tamaño, es más fácil gestionar actualizaciones y aplicar parches de seguridad en sistemas sensibles.

Docker - es una plataforma de código abierto que permite a los desarrolladores crear, implementar y ejecutar aplicaciones en contenedores ligeros y portátiles. Estos contenedores encapsulan todo lo necesario para que una aplicación funcione, incluidas sus dependencias, bibliotecas, herramientas de sistema y código, asegurando consistencia entre los entornos de desarrollo, pruebas y producción. Según Red Hat, “Docker es la tecnología de organización en contenedores que posibilita la creación y el uso de los contenedores de Linux®” (Red Hat, 2023). Esta capacidad de empaquetar aplicaciones con todas sus dependencias facilita su despliegue en diferentes plataformas.

La arquitectura de Docker está basada en un modelo cliente-servidor. El cliente de Docker interactúa con el demonio Docker (`dockerd`), que se encarga de las tareas principales como construir, ejecutar y distribuir contenedores. Según Docker, “el demonio Docker escucha solicitudes API y administra objetos Docker, como imágenes, contenedores, redes y volúmenes” (Docker, s.f.). Esto permite gestionar contenedores de forma eficiente, utilizando una API REST para comunicarse entre componentes.

Una de las características clave de Docker es su portabilidad. Según Stephanie Susnjara e Ian Smalley, “Docker es una plataforma de código abierto que permite a los desarrolladores crear, desplegar, ejecutar, actualizar y administrar contenedores con facilidad” (Susnjara & Smalley, 2024). Esto lo convierte en una herramienta indispensable en entornos DevOps, donde la integración y entrega continua son esenciales para acelerar el ciclo de vida del desarrollo de software.

Además, Docker ha impulsado la adopción de arquitecturas basadas en microservicios. AWS señala que “Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución” (AWS, s.f.). Gracias a esta funcionalidad, los desarrolladores pueden dividir aplicaciones monolíticas en componentes más pequeños y escalables, simplificando su mantenimiento y despliegue.

Docker también permite una gestión eficiente de recursos en entornos de infraestructura compartida. Cada contenedor opera de forma aislada, pero comparte el mismo núcleo del sistema operativo, lo que minimiza el uso de recursos. Esto es especialmente útil en aplicaciones en la nube, donde la optimización del consumo de recursos es crítica.

OpenCV (Open Source Computer Vision Library) - es una biblioteca de software de código abierto diseñada para facilitar el desarrollo de aplicaciones en el ámbito de la visión por computadora y el aprendizaje automático. Fue creada inicialmente por Intel en 1999, con el propósito de proporcionar una infraestructura estándar para el desarrollo de aplicaciones de visión, fomentando su implementación en proyectos comerciales e investigativos. Según Python Geeks, “OpenCV es una biblioteca de visión por computadora que incluye una amplia gama de funciones para tareas como la detección de objetos, el reconocimiento facial y la mejora de imágenes” (Python Geeks, s.f.).

La biblioteca ofrece más de 2,500 algoritmos optimizados que permiten realizar tareas fundamentales como detección y reconocimiento de rostros, seguimiento de

movimientos, clasificación de acciones humanas en videos, análisis de imágenes médicas, generación de modelos 3D y realidad aumentada.

OpenCV es compatible con múltiples lenguajes de programación, incluyendo C++, Python, Java y MATLAB, y puede ejecutarse en plataformas como Windows, Linux, macOS, Android e iOS. Según la Fundación OpenCV, “la biblioteca está diseñada para ser altamente extensible y cuenta con una comunidad activa que continuamente contribuye con nuevas características y algoritmos” (OpenCV, s.f.). Esto asegura que la herramienta evolucione constantemente para satisfacer las demandas de la tecnología moderna.

Otra característica clave de OpenCV es su capacidad para integrarse con herramientas y frameworks de inteligencia artificial como TensorFlow, PyTorch y Keras. Esto permite a los desarrolladores combinar algoritmos clásicos de visión por computadora con modelos de aprendizaje profundo, logrando soluciones robustas y eficaces. Además, OpenCV incluye soporte para aceleración mediante GPU, lo que mejora el rendimiento en tareas intensivas como la segmentación de imágenes, el análisis de videos y la detección de objetos.

En el ámbito de la investigación, OpenCV ha sido ampliamente utilizado en áreas como la robótica, el análisis de imágenes biomédicas y la seguridad. Python Geeks destaca que “OpenCV ha revolucionado la visión por computadora, proporcionando herramientas accesibles tanto para principiantes como para desarrolladores avanzados” (Python Geeks, s.f.).

NumPy (Numerical Python) - es una biblioteca de código abierto fundamental para la computación científica en Python, diseñada para trabajar con grandes conjuntos de datos numéricos. Su principal objetivo es proporcionar soporte para la creación y manipulación de arreglos y matrices multidimensionales, junto con un conjunto integral de funciones matemáticas de alto nivel. Según la documentación oficial de NumPy, “es la biblioteca fundamental para la computación científica en Python, diseñada para ofrecer un rendimiento excepcional y facilidad de uso” (NumPy, s.f.).

Una de las principales características de NumPy es su capacidad para manejar estructuras de datos de gran tamaño y realizar cálculos complejos con rapidez. Como señala Erick Roch Moraguez, “NumPy es crucial para realizar operaciones matemáticas y lógicas sobre arreglos y matrices, incluyendo transformaciones de formas, transformadas de Fourier, álgebra lineal básica y simulaciones estadísticas” (Moraguez, s.f.). Estas capacidades la convierten en una herramienta indispensable para científicos de datos, ingenieros y desarrolladores en campos como la física, la ingeniería, las finanzas y el aprendizaje automático.

Además, NumPy ofrece compatibilidad con otras bibliotecas y frameworks de Python, como Pandas, Matplotlib, Scikit-learn y TensorFlow, lo que amplía significativamente sus aplicaciones en análisis de datos, visualización y aprendizaje profundo. Según Fernando Cardellino, “aprender NumPy es esencial para cualquier persona que desee incursionar en la ciencia de datos o la computación científica con Python, ya que muchas otras bibliotecas dependen de su estructura base” (Cardellino, 2023).

Otra ventaja clave de NumPy es su implementación en C, lo que permite realizar operaciones de forma eficiente, incluso en grandes conjuntos de datos. Su diseño optimizado asegura un uso eficaz de recursos computacionales, siendo ideal para sistemas que requieren procesamiento intensivo. Además, su capacidad para integrarse con otros lenguajes como C y Fortran lo hace altamente versátil para proyectos avanzados.

En términos de aplicaciones prácticas, NumPy se utiliza en análisis de datos, simulaciones numéricas, procesamiento de señales y modelos predictivos. Por ejemplo, es comúnmente empleado para realizar cálculos de álgebra lineal en algoritmos de aprendizaje automático o para realizar simulaciones estocásticas en la investigación financiera.

Entorno de Desarrollo y Configuración

Arduino IDE - es una aplicación multiplataforma de código abierto diseñada para facilitar la programación y carga de código en placas de microcontroladores

Arduino. Este entorno es una herramienta clave para desarrolladores que buscan una manera intuitiva de crear proyectos electrónicos. Según la documentación oficial de Arduino, “el Arduino IDE permite escribir y cargar código en las placas Arduino de manera simple y eficiente, utilizando un entorno gráfico accesible” (Arduino, s.f.).

Basado en el lenguaje de programación Processing y utilizando una versión simplificada de C/C++, el Arduino IDE permite a los usuarios concentrarse en la lógica de sus proyectos sin preocuparse por los detalles complejos de programación. Como señala Tuelectronica, “Arduino IDE simplifica el desarrollo al ofrecer un entorno basado en Processing, con funciones específicas para programar microcontroladores” (Tuelectronica, s.f.).

El entorno incluye características como resaltado de sintaxis, autoindentación, un compilador integrado y un cargador de arranque que permiten verificar y cargar el código directamente en las placas Arduino a través de conexiones USB. Byron Vargas resalta que “estas herramientas integradas hacen que el proceso de programación y depuración sea rápido y eficiente, especialmente para principiantes” (Vargas, 2024).

Controladores CH340 - son controladores de software necesarios para que las computadoras puedan comunicarse correctamente con dispositivos que integran el chip CH340. Este chip, desarrollado por WCH (WinChipHead), actúa como un convertidor de USB a serie, permitiendo la conexión de dispositivos como placas de desarrollo Arduino, impresoras 3D, módulos seriales y otros periféricos que utilizan comunicación serial a través de puertos USB. Según Akbari (Smart Robot Car Kit V4.0 (With Camera), s.f.), “el CH340 es un chip de conversión USB a serie que permite la comunicación entre una computadora y dispositivos seriales, proporcionando una interfaz accesible para tareas de programación y control” (Akbari s.f.).

La instalación de estos controladores es esencial, ya que sin ellos el sistema operativo no reconocerá correctamente los dispositivos basados en el chip CH340. Como explica SparkFun, “sin el controlador adecuado, su computadora no podrá comunicarse con el dispositivo CH340, lo que dificultará cualquier intento de programación o comunicación” (SparkFun, s.f.). Este proceso garantiza que las placas o periféricos sean detectados y funcionen de manera estable.

Una de las ventajas de los controladores CH340 es su compatibilidad multiplataforma. Estos drivers funcionan en sistemas operativos como Windows, macOS y Linux, aunque el proceso de instalación puede variar. Por ejemplo, en Windows, es necesario descargar los controladores desde el sitio web oficial de WCH, mientras que, en algunas distribuciones de Linux, el soporte para CH340 está integrado directamente en el kernel, lo que elimina la necesidad de instalación manual. HWLibre destaca que “en sistemas Linux modernos, simplemente conectar la placa al puerto USB es suficiente para que el sistema reconozca automáticamente el dispositivo” (HWLibre, s.f.).

Además de su versatilidad, estos controladores se utilizan ampliamente en proyectos de desarrollo y sistemas embebidos debido a su confiabilidad y simplicidad. Son esenciales para trabajar con placas de bajo costo que incluyen el chip CH340 en lugar de chips de interfaz más costosos. No obstante, se debe tener precaución al descargar estos controladores, ya que fuentes no confiables podrían comprometer la seguridad del sistema o generar problemas de compatibilidad.

Desarrollo de la actividad

Materiales y componentes utilizados

Hardware

- **Smart Robot Car V4.0** de la marca **Elegoo**: el robot principal utilizado para el desarrollo.
- **Raspberry Pi 5** con 4 GB de RAM y 64 GB de almacenamiento interno: para el procesamiento y control del robot.
- Fuente de alimentación para la Raspberry Pi: proporciona energía estable al dispositivo.
- Cable **micro-USB a HDMI**: permite conectar la Raspberry Pi a una pantalla para configuraciones iniciales y pruebas.
- Sensor y **ventilador de enfriamiento** (cooler) para la Raspberry Pi: ayuda a mantener temperaturas operativas óptimas.
- Tres pelotas de colores **azul, rojo y verde lima**: utilizadas como elementos de prueba para las funcionalidades programadas en el robot.



Smart Robot Car V4.0

Raspberry Pi 5

Fuente de alimentación



Cable Micro-USB a HDMI

Ventilador de enfriamiento

Pelotas

Ilustración 6. 1. 1 - Materiales de Hardware

Software

- **Raspbian Bookworm 12:** sistema operativo basado en Debian diseñado específicamente para Raspberry Pi.
- **Python 3.11 Slim:** versión optimizada de Python para implementar las funcionalidades del robot.
- **Docker:** herramienta para la virtualización de contenedores, usada para gestionar el entorno de desarrollo.
- **OpenCV:** biblioteca de visión artificial para el reconocimiento y procesamiento de imágenes.
- **Numpy:** biblioteca para la manipulación de datos y operaciones matemáticas, utilizada en el tratamiento de datos del robot.



Raspberry Pi



Python 3.11 Slim



Docker Official Repository



OpenCV Library



NumPy Library

Ilustración 6. 1. 2 - Materiales de Software

Instalación de controladores para ESP32-S3-CAM y Arduino UNO

Para configurar los microcontroladores **ESP32-S3-CAM** y **Arduino UNO**, es necesario instalar los controladores correspondientes, lo que permitirá que sean reconocidos como dispositivos seriales en el **IDE de Arduino**.

Instalación de controladores para el ESP32-S3-CAM

El modelo de cámara utilizado en este proyecto es el **ESP32-S3 WROOM-1** de la marca **Elegoo**, que requiere el controlador **CH340**. A continuación, se describen los pasos para realizar la instalación:



Ilustración 6. 2. 1 - *ESP32-S3-CAM Elegoo*

- 1. Descargar el controlador CH340:** En la documentación oficial se proporciona un enlace para descargar el controlador. Puedes acceder a él desde el siguiente vínculo: [Descargar CH340](#).
- 2. Extraer y ejecutar el instalador:** Una vez descargado, descomprime el archivo. En la carpeta extraída, encontrarás el archivo **CH341SER_windows.exe**. Ejecútalo como administrador.
- 3. Proceso de instalación:** Al iniciar el instalador, aparecerá una ventana similar a la siguiente. Haz clic en el botón **Install** para completar la instalación.
- 4. Verificación del controlador:** Una vez instalado, verifica que el controlador se haya configurado correctamente:

- Haz clic derecho en el icono de Windows y selecciona **Administrador de dispositivos**.
- En la sección **Puertos (COM y LPT)**, deberías encontrar un dispositivo identificado como **USB-SERIAL CH340** al conectar tu **ESP32-S3-CAM** mediante un cable USB.

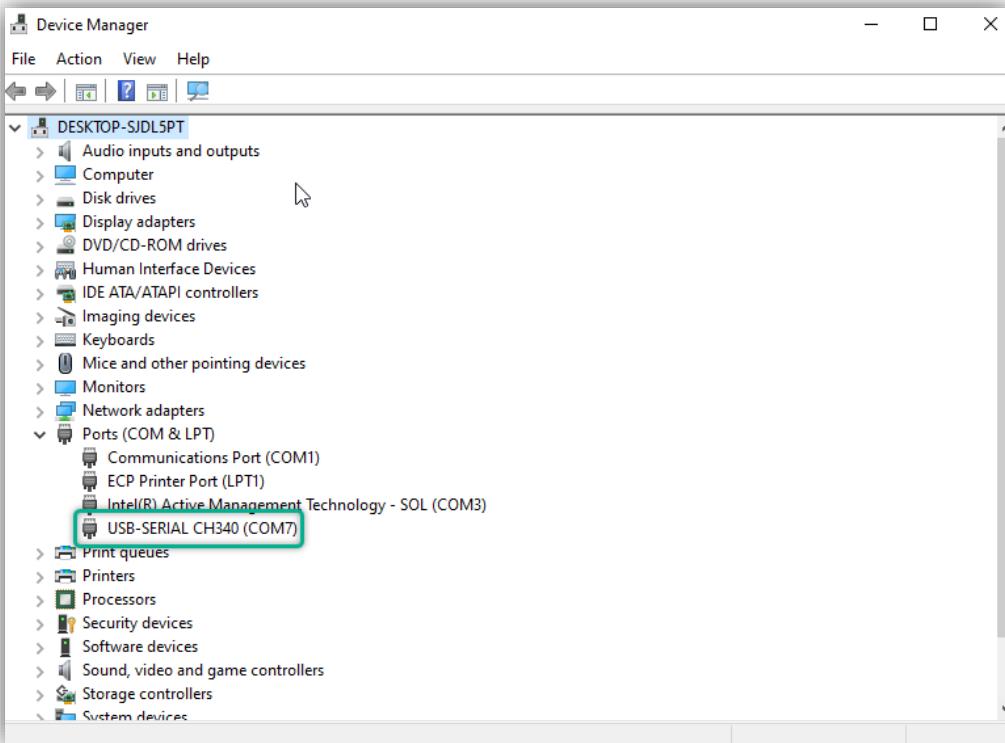


Ilustración 6. 2. 2 - Verificación de controlador de cámara

Si el dispositivo no aparece, podría haber problemas con el hardware, como un defecto en la cámara.

Instalación de controladores para Arduino UNO

El procedimiento para el **Arduino UNO** es similar, ya que también utiliza el controlador CH340. Sigue los mismos pasos descritos para instalar el controlador y verifica su correcto reconocimiento en el **Administrador de dispositivos**.

Configuración del entorno de programación en Arduino IDE para el módulo ESP32-S3-CAM

Para preparar el **Arduino IDE** y programar el módulo **ESP32-S3-CAM**, sigue los pasos detallados a continuación:

1. Conexión del módulo

- Conecta el módulo a tu computadora utilizando un cable USB tipo C.

2. Configuración inicial en Arduino IDE

1. Abre el **Arduino IDE**.
2. Ve al menú **Archivo > Preferencias**.
3. En la barra de "Additional Boards Manager URLs", copia y pega el siguiente enlace: <https://arduino.me/packages/esp32.json>
4. Haz clic en **OK** para guardar los cambios.

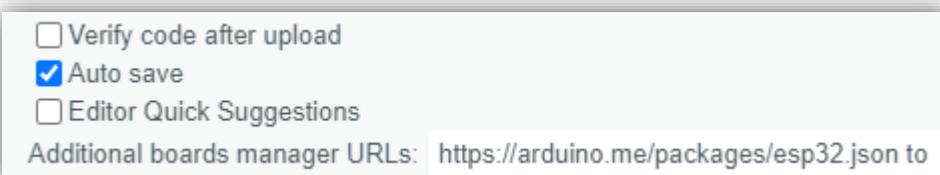


Ilustración 6. 3. 1 - Paquetería JSON de ESP32

3. Instalación de las placas ESP32

1. Dirígete al menú **Herramientas > Administrador de placas**.
2. En el buscador, escribe **esp32**.
3. Selecciona e instala la versión **1.0.4** del paquete ESP32.
4. Reinicia el **Arduino IDE** para aplicar los cambios.

4. Configuración de la placa ESP32-S3-CAM

1. En el menú **Herramientas > Placas**, selecciona **ESP32S3 Dev Module**.
2. Configura las siguientes opciones en el mismo menú:
 - **USB CDC on Boot**: Selecciona **Enable**.
 - **Flash Size**: Selecciona **8MB**.
 - **Partition Scheme**: Selecciona **8M with SPIFFS**.

- **PSRAM:** Selecciona **OPI PSRAM**.

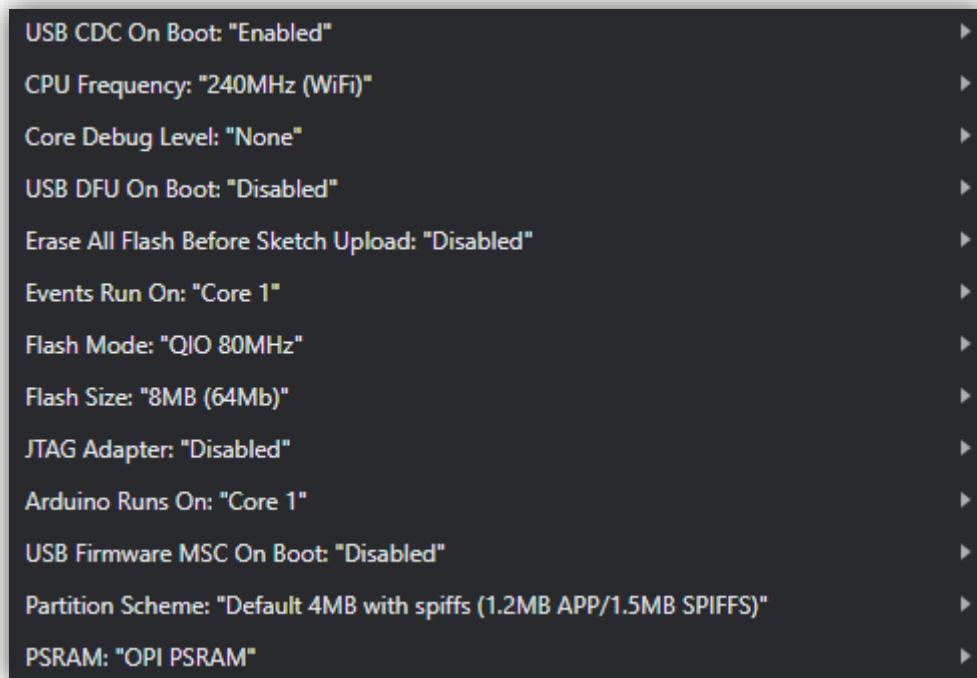


Ilustración 6. 3. 2 - Configuración de ESP32-S3

Modificación del código para el módulo ESP32-S3-CAM

Una vez configurado el entorno de programación, fue necesario realizar ajustes al código preinstalado en el módulo **ESP32-S3-CAM**. Según (Michael, 2021), es imprescindible habilitar y ajustar el uso de la API que proporciona el módulo para capturar métricas y controlar el movimiento del robot enviando peticiones.

1. Obtención del código fuente

El código fuente oficial fue descargado desde el siguiente enlace proporcionado por Elegoo: [Descargar código fuente](#).

Tras la descarga, se descomprimió el archivo y se seleccionó el directorio correspondiente al modelo de cámara utilizado, **ESP32-S3**. Dentro de este, se encontraron varios subdirectorios relacionados con diferentes versiones de firmware. Para este caso, se trabajó con el directorio '**Channel 9 with Password**'.

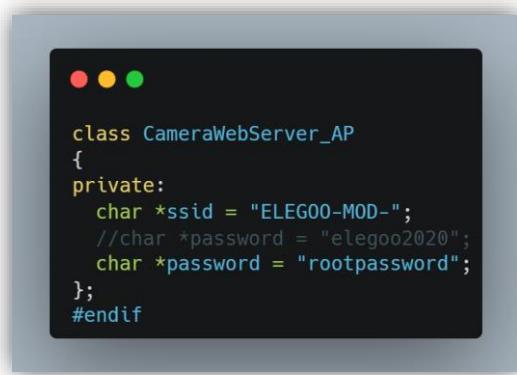
2. Configuración de seguridad

En el video oficial (Michael, 2021), se indica que es necesario añadir una contraseña para garantizar que el servidor web no sea accesible por usuarios no autorizados.

1. **Localización del archivo de configuración:** Navegamos al archivo **CameraWebServer_AP.h**, el cual contiene la configuración del servidor web.

2. Modificación de la clase principal

- Se agregó una contraseña personalizada definiendo el valor **rootpassword**. Además, se asignó un SSID con el prefijo **MOD** para facilitar la identificación de la red.



```
class CameraWebServer_AP
{
private:
    char *ssid = "ELEGOO-MOD-";
    //char *password = "elegoo2020";
    char *password = "rootpassword";
};

#endif
```

Ilustración 6. 4. 1 - Configuración de seguridad de la cámara

3. Ajuste de la frecuencia de la cámara

Para optimizar los tiempos de latencia de la cámara y mejorar la transferencia de fotogramas, es necesario ajustar la frecuencia del reloj (XCLK). Esto se logra modificando el archivo CameraWebServer_AP.cpp.

1. Localización de la línea de configuración

- Abrir el archivo **CameraWebServer_AP.cpp** en el editor de tu preferencia dentro del entorno configurado.
- Buscar la línea de código donde se define la frecuencia del reloj:

2. Modificación del valor de frecuencia

- Sustituir el valor actual por **20000000** para establecer la frecuencia a 20 MHz. La línea quedará de la siguiente manera:



```
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
```

Ilustración 6. 4. 2 - Ajuste de frecuencia

4. Modificación de los *heartbeats*

Se nos indicó que era necesario comentar las líneas relacionadas con los *heartbeats*, ya que esto ayuda a evitar desconexiones frecuentes durante el funcionamiento del módulo (Michael, 2021). Este cambio se realiza en el archivo principal **ESP32_CameraServer_AP_2023_V1.3**.

1. Ubicación de las líneas a modificar

- Abrir el archivo **ESP32_CameraServer_AP_2023_V1.3** en tu entorno de programación.
- Navegar hacia las líneas donde se implementan los *heartbeats*. Estas líneas se encuentran generalmente al final del código.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three colored window control buttons (red, yellow, green). Below them, the terminal displays the following code:

```
/*
1. Remove Heartbeat Status for Connection Establish Status.
*/

/*
if (true == Heartbeat_status)
{
Heartbeat_status = false;
Heartbeat_count = 0;
}
else if (false == Heartbeat_status)
{
Heartbeat_count += 1;
}
if (Heartbeat_count > 3)
{
Heartbeat_count = 0;
Heartbeat_status = false;
break;
}
*/
```

Ilustración 6. 4 - Líneas comentadas de heartbeats

5. Modificación personalizada: Ajustes de la cámara ESP32-S3-CAM

Durante las pruebas iniciales, se identificó que la imagen capturada por la cámara se veía demasiado oscura, con colores desleales a la realidad y una predominancia de tonos negros. Para mejorar la calidad de los fotogramas, se realizó un ajuste en el código, aplicando un filtro azul, disminuyendo el contraste y

aumentando la saturación. Este cambio permitió obtener imágenes más claras y con colores más precisos.

1. Ubicación de los ajustes

Los cambios se realizaron en el archivo **CameraWebServer_AP.cpp**, en las secciones encargadas de configurar los parámetros de la cámara.



Ilustración 6. 4. 4 - Configuración para mejorar calidad de la cámara

2. Explicación de los parámetros

- **set_vflip(sensor, 1)**: Rota la imagen verticalmente para corregir la orientación.
- **set_special_effect(sensor, 5)**: Aplica el filtro azul para equilibrar la tonalidad de la imagen.
- **set_brightness(sensor, 0)**: Establece el brillo en un nivel neutro.
- **set_contrast(sensor, -1)**: Reduce el contraste, suavizando las diferencias entre zonas claras y oscuras.
- **set_saturation(sensor, 2)**: Incrementa la saturación, resaltando los colores.

3. Resultado de los ajustes

Estos cambios lograron:

- Mejorar significativamente la iluminación de los fotogramas.
- Corregir los colores para que se vean más fieles a la realidad.
- Aumentar la calidad visual de los objetos capturados, facilitando el análisis de la imagen.

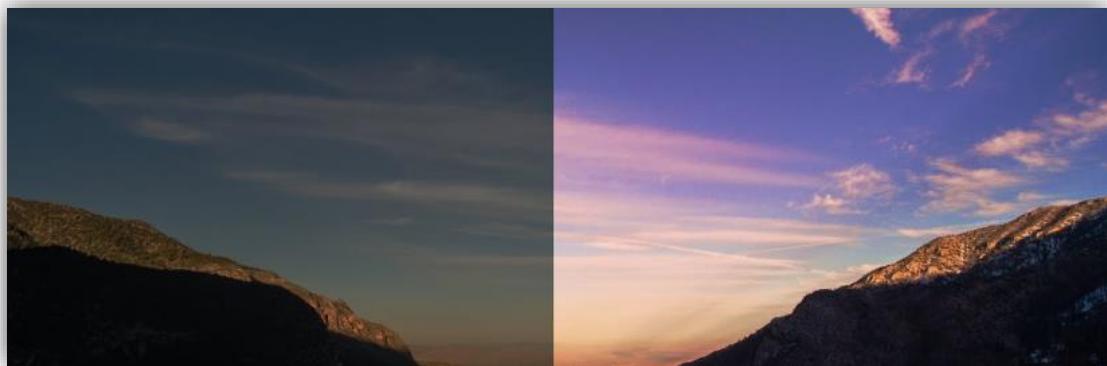


Ilustración 6. 4. 5 - Ejemplo de la mejora de los fotogramas

Finalmente, conectamos el módulo de la cámara ESP32-S3-CAM, verificamos el código en el **Arduino IDE**, lo compilamos y subimos al dispositivo. Luego, comprobamos el funcionamiento de la cámara accediendo al **localhost** en el puerto **40** para confirmar que transmite correctamente las imágenes.

Configuración del entorno de programación en Arduino IDE para Arduino UNO

Primeramente, accedimos al apartado de **Herramientas (Tools)** en el **Arduino IDE** para realizar la configuración inicial de la placa y el puerto serial. A continuación, realizamos los siguientes pasos:

1. **Selección de la placa:** En el menú desplegable de **Herramientas > Placa (Board)**, seleccionamos **Arduino UNO**, asegurándonos de que corresponde al modelo del dispositivo conectado.
2. **Selección del puerto serial:** Luego, en **Herramientas > Puerto (Port)**, seleccionamos el puerto COM correspondiente al dispositivo. Este puerto aparece automáticamente en la lista al conectar el Arduino a la computadora mediante el cable USB.
3. **Verificación de conexión:** Una vez configurada la placa y el puerto, confirmamos que el dispositivo estaba correctamente reconocido mediante la opción **Obtener información de la placa (Get Board Info)**, disponible en el mismo menú de **Herramientas**. Esto nos permitió validar que la comunicación entre el Arduino y el IDE estaba establecida.

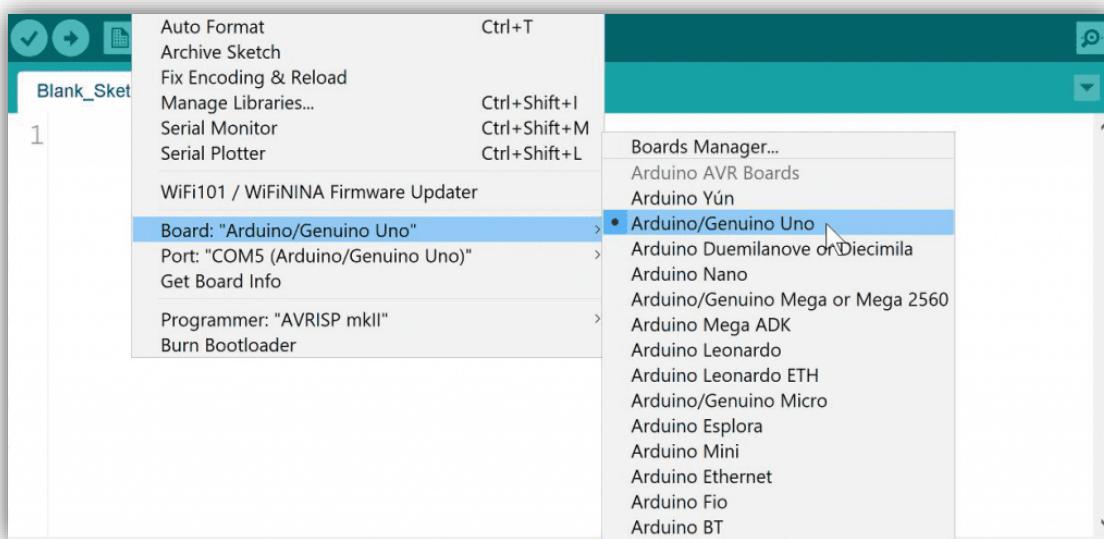


Ilustración 6. 5. 1 - Selección de Arduino UNO

Modificaciones al código de Arduino Uno para el sensor MPU6050

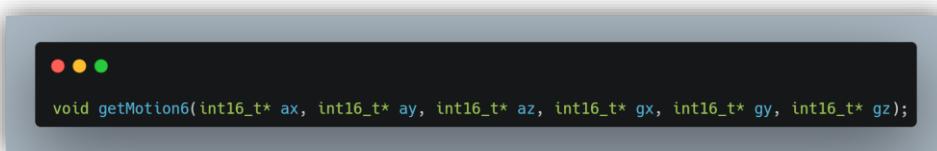
De acuerdo con (Michael, 2021), el primer paso consiste en descargar el código base desde la página oficial de Elegoo. Este será utilizado como punto de partida para realizar las modificaciones necesarias, compilarlo y cargarlo posteriormente en la placa. El enlace del proyecto es el siguiente: [Descargar proyecto oficial](#).

1. Activación del módulo de comunicación para el sensor MPU6050

Para obtener información del acelerómetro y del espacio en el que se encuentra el robot, es necesario activar el módulo de comunicación de datos del sensor **MPU6050**:

1. Edición del archivo **MPU6050.h**

- Abrir el archivo **MPU6050.h**.
- Localizar la clase **MPU6050**. Dentro de esta clase, busca la declaración de la función **getMotion6**.
- Copiar esta declaración, ya que será utilizada en el siguiente paso.

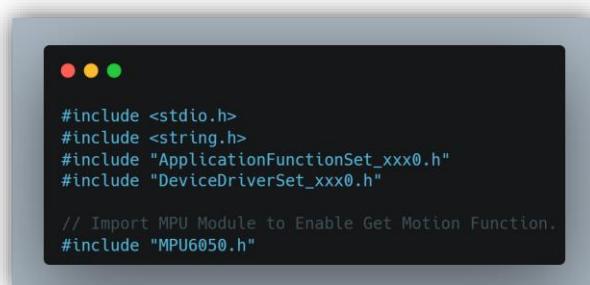


```
void getMotion6(int16_t* ax, int16_t* ay, int16_t* az, int16_t* gx, int16_t* gy, int16_t* gz);
```

Ilustración 6. 6. 1 - Función de sensor MPU

2. Importación de funciones en el archivo **ApplicationFunctionSet_xxx0.cpp**

1. Abrir el archivo **ApplicationFunctionSet_xxx0.cpp**.
2. Incluir la importación del encabezado que contiene la función **getMotion6** agregando lo siguiente al inicio del archivo:



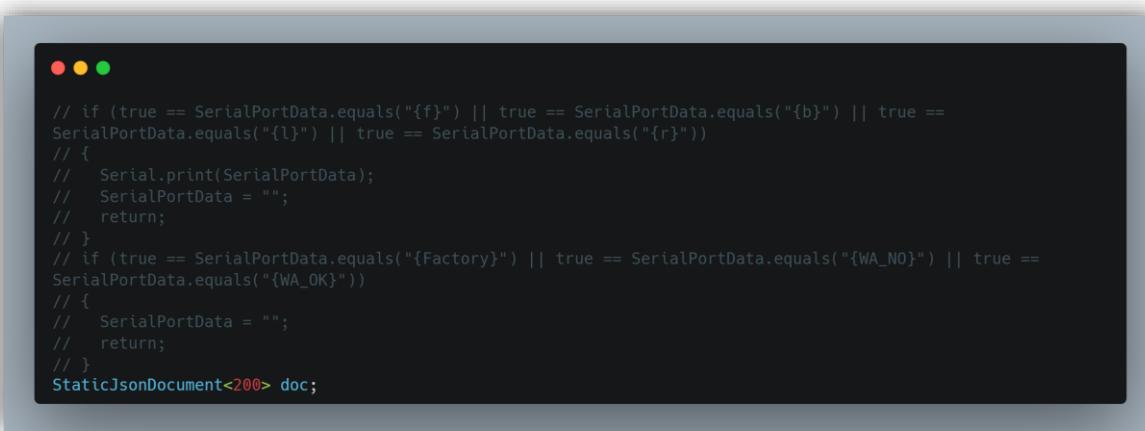
```
#include <stdio.h>
#include <string.h>
#include "ApplicationFunctionSet_xxx0.h"
#include "DeviceDriverSet_xxx0.h"

// Import MPU Module to Enable Get Motion Function.
#include "MPU6050.h"
```

Ilustración 6. 6. 2 - Importación de función

3. Modificación de la función de análisis de datos

1. Ubicar la función **SerialPortDataAnalysis**, que se encarga de analizar los datos enviados al puerto serial.
2. Comentar el bloque de código responsable de reportar errores en la decodificación de datos JSON. Esto evita interrupciones no deseadas al procesar comandos.



```
// if (true == SerialPortData.equals("{f}") || true == SerialPortData.equals("{b}") || true ==
SerialPortData.equals("{l}") || true == SerialPortData.equals("{r}") )
{
    Serial.print(SerialPortData);
    SerialPortData = "";
    return;
}
// if (true == SerialPortData.equals("{Factory}") || true == SerialPortData.equals("{WA_NO}") || true ==
SerialPortData.equals("{WA_OK}") )
{
    SerialPortData = "";
    return;
}
StaticJsonDocument<200> doc;
```

Ilustración 6. 6 - 3 - Modificación de análisis de datos

4. Implementación de un nuevo caso en el switch

1. Dentro de la función **SerialPortDataAnalysis**, agregar un nuevo caso en el **switch** para procesar los datos provenientes del sensor **MPU6050**.
2. Asegúrate de imprimir cada comando recibido en el puerto serial para facilitar la depuración y validación de datos.



```
case 6:
// Define Orientation Variables
int16_t ax, ay, az, gx, gy, gz;
// Get MPU Data
mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
#if _is_print
char toString[10];
sprintf(toString, "%d,%d,%d,%d,%d", ax, ay, az, gx, gy);
Serial.println('{' + CommandSerialNumber + toString);
// sprintf(toString, "%d}", round(Yaw));
// Serial.println('{' + CommandSerialNumber + toString);
```

Ilustración 6. 6 - 4 - Implementación de nuevo switch case

5. Compilación y carga del código en la placa

1. En el **Arduino UNO**, se movió el interruptor físico del modo **CAM** al modo **UPLOAD**.
2. Abrir el código en el **Arduino IDE**, seleccionar la placa **Arduino UNO** y el puerto serial correspondiente.
3. Compilar y subir el código a la placa utilizando el botón de **Subir** del IDE.

Ensamblaje del Smart Robot con cámara y Arduino UNO

Una vez configurados tanto el módulo de la cámara **ESP32-S3-CAM** como el **Arduino UNO**, procedimos a ensamblar las distintas piezas del robot. Para ello, utilizamos como referencia la guía oficial de ensamblaje incluida en la caja del producto.

1. Preparación de los materiales

- Separamos cuidadosamente todos los componentes necesarios, asegurándonos de contar con las piezas correctas según la lista de materiales provista en la guía. Entre los elementos principales se incluyen:

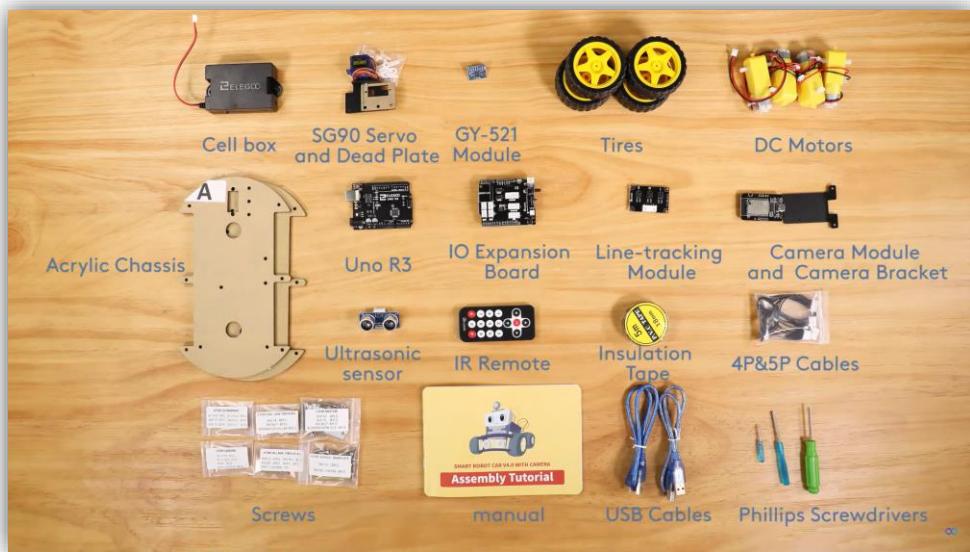


Ilustración 6. 7. 1 - Materiales para el Smart Robot

2. Proceso de ensamblaje

- Seguimos las instrucciones paso a paso de la guía oficial, ajustando cada componente en su lugar correspondiente. Esto incluyó:
 - La instalación de los servomotores y la fijación de las llantas.
 - La colocación del sensor ultrasónico y el módulo de la cámara en la parte frontal.
 - La integración del sensor de color en la base del robot.
 - La conexión del Arduino UNO y su correcta ubicación dentro de la estructura.

3. Resultado final

El ensamblaje concluyó con éxito, logrando un diseño funcional y estable. El producto final incluye:

- **Cuatro servos y llantas:** Para garantizar movilidad fluida y precisa.
- **Sensor ultrasónico con cámara integrada:** Ubicados en la parte frontal como encabezado, diseñados para la detección de obstáculos y captura de imágenes.
- **Arduino UNO:** Actuando como el cerebro principal del sistema.
- **Sensor de color:** Posicionado para detectar variaciones cromáticas en el entorno.



Ilustración 6. 7. 2 - Producto final del Smart Robot

Instalación y configuración de Raspberry Pi OS en una Raspberry Pi 5, e instalación de Docker

Primero, instalamos el **Raspberry Pi Imager** desde el sitio oficial de Raspberry Pi en nuestra computadora. Este programa nos permitirá flashear el sistema operativo en la memoria microSD. Una vez descargado e instalado, lo abrimos.

1. Flashear la memoria microSD

- Insertamos la tarjeta microSD en la computadora.
- En el Raspberry Pi Imager, seleccionamos "**Choose OS**" y escogemos **Raspberry Pi OS (64-bit)**, que es compatible con la Raspberry Pi 5.
- Luego, en "**Choose Storage**", seleccionamos la tarjeta microSD.
- Hacemos clic en el ícono de configuración (⚙️) para preconfigurar detalles como:
 - **Habilitar SSH** (seleccionando "Enable SSH" y definiendo una contraseña segura).
 - Configurar la red Wi-Fi con el nombre y la contraseña de nuestra red.
 - Establecer la zona horaria adecuada.
- Presionamos "**Write**" para iniciar el proceso de flasheo. Esto descargará e instalará Raspberry Pi OS en la tarjeta. Una vez finalizado, retiramos la tarjeta de la computadora.



Ilustración 6. 8. 1 - Raspberry Pi Imager

2. Iniciar la Raspberry Pi 5

- Insertamos la tarjeta microSD en la Raspberry Pi 5 y la conectamos a una fuente de alimentación.
- Conectamos un teclado, mouse y monitor a la Raspberry Pi para la configuración inicial.
- Al encenderla, seguimos el asistente de configuración para:
 - Configurar el idioma y la distribución del teclado.
 - Conectar la Raspberry Pi a la red Wi-Fi si no lo hicimos previamente.
 - Actualizar el sistema operativo a la última versión (el asistente ofrece esta opción automáticamente).

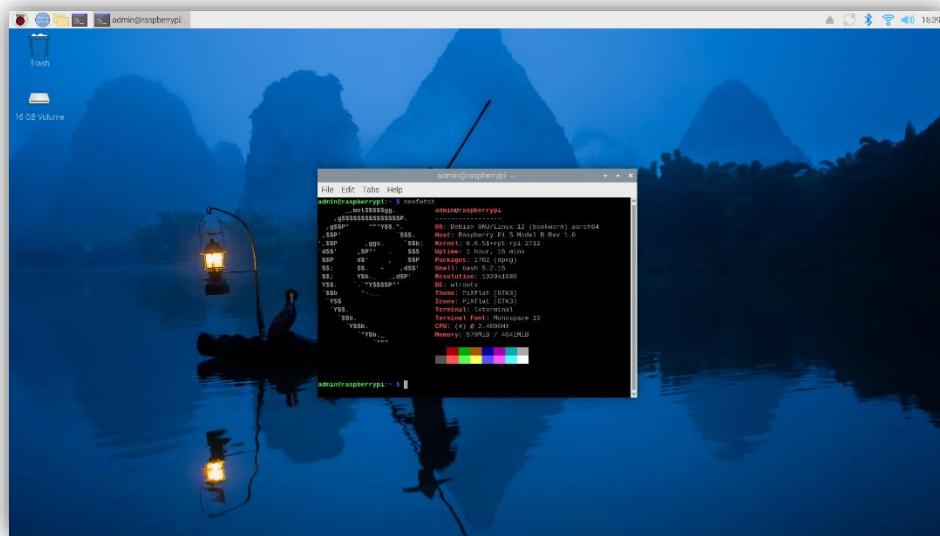
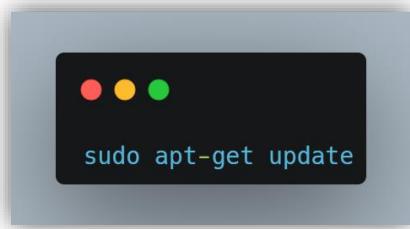


Ilustración 6. 8. 2 - Primer vistazo de Raspberry OS

3. Instalación de Docker desde la documentación oficial

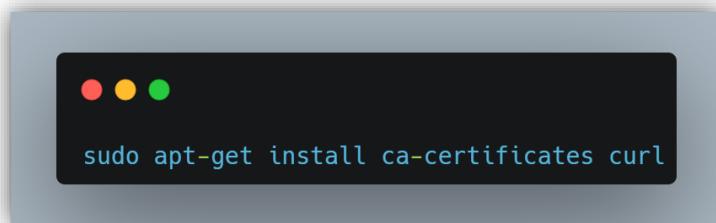
Docker será una herramienta de gestión y creación de contenedores que nos servirá para el seguimiento de nuestro proyecto y rápido despliegue, por lo que a continuación se detallan los pasos hechos en el Raspberry:

- Abrimos una terminal en la Raspberry Pi.
- Actualizar la lista de paquetes.



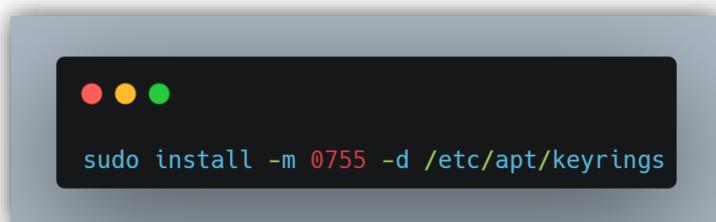
```
sudo apt-get update
```

- Instalar certificados y herramientas necesarias



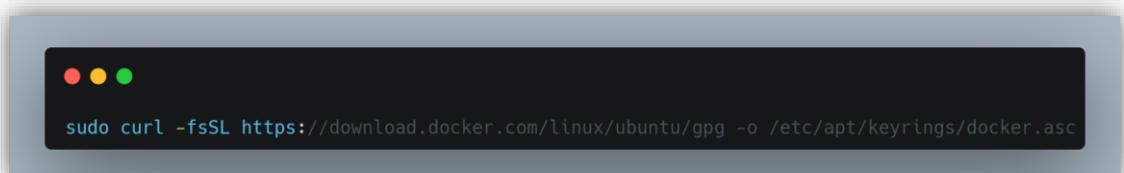
```
sudo apt-get install ca-certificates curl
```

- Crear un directorio para almacenar claves GPG



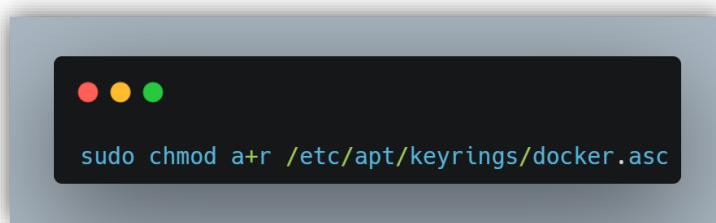
```
sudo install -m 0755 -d /etc/apt/keyrings
```

- Descargar la clave GPG de Docker



```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

- Establecer permisos de lectura para la clave



```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

- Configurar el repositorio de Docker

```
echo \  
  "deb [ arch=$(dpkg --print-architecture) signed-  
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \  
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- Actualizar la lista de paquetes nuevamente

```
sudo apt-get update
```

- Instalar los paquetes oficiales de Docker

```
sudo apt-get install docker-ce  
docker-ce-cli containerd.io docker-  
buildx-plugin docker-compose-plugin
```

- Verificar si la instalación se hizo de manera correcta

```
sudo docker run hello-world
```

Codificación del algoritmo de reconocimiento y evasión de obstáculos

El funcionamiento del **Smart Robot** se basó en el desarrollo de dos aplicaciones y algoritmos distintos, cada uno diseñado para ejecutar funciones específicas. El primero está enfocado en el reconocimiento y evasión de obstáculos, además de tomar decisiones en tiempo real sobre la dirección a seguir. Esto se logró mediante el uso de señales emitidas por el sensor ultrasónico del robot. Estas señales son enviadas desde el servidor web integrado en el ESP32-S3 hacia una Raspberry Pi o cualquier computadora conectada a la misma red de la tarjeta.

El servidor web transmite datos sobre la proximidad de los objetos al dispositivo anfitrión, el cual, utilizando un código desarrollado en Python, evalúa las distancias y determina los movimientos necesarios del robot. Posteriormente, se envía una respuesta al servidor web indicando la acción a realizar.

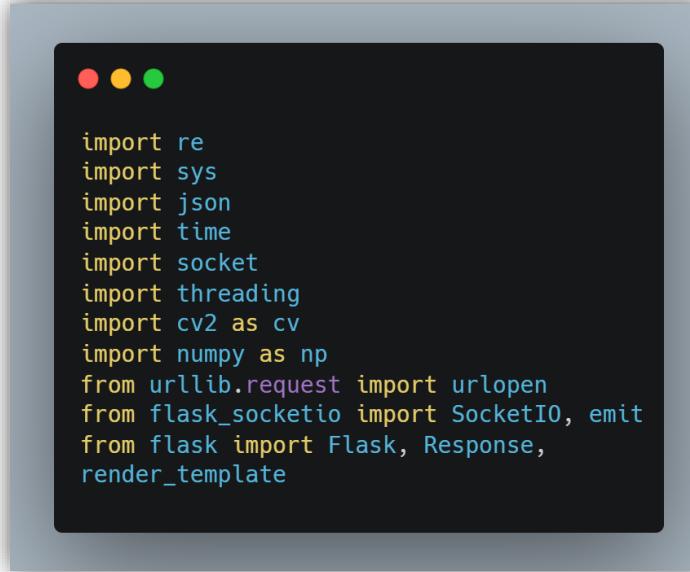
El código base utilizado para esta implementación pertenece a (Michael, 2021) y cuenta con las licencias necesarias para ser modificado. Se realizaron adaptaciones específicas para integrar nuevas funcionalidades, como el despliegue de resultados en una aplicación web. En esta plataforma, los usuarios pueden visualizar en tiempo real la transmisión de la cámara del robot, así como un historial de operaciones realizadas y las distancias registradas por el servidor web.

Toda la información, incluidas las imágenes en vivo y los datos históricos, se transmite a través de **Web Sockets**, garantizando actualizaciones dinámicas y en tiempo real en la interfaz de la aplicación.

Primero, se importaron diversas bibliotecas necesarias para implementar las funcionalidades del proyecto. Entre ellas se incluyeron:

- **Expresiones regulares** para procesar y analizar cadenas de texto de manera eficiente.
- **Operaciones de sistema** para la gestión de archivos, directorios y procesos en el entorno operativo.
- **Lectura de JSON** para manejar datos estructurados en formato JSON.

- **Manejo de tiempo** para funciones relacionadas con temporizadores y registros de eventos.
- **Socket** para la comunicación en red.
- **Multihilo** para optimizar la ejecución concurrente de tareas.
- **OpenCV** para la captura y procesamiento de imágenes.
- **NumPy** para realizar operaciones algebraicas y manipulaciones de datos numéricos.
- **Flask** para el despliegue de la aplicación web.
- **Flask-SocketIO** para habilitar comunicación en tiempo real mediante WebSockets.



```

import re
import sys
import json
import time
import socket
import threading
import cv2 as cv
import numpy as np
from urllib.request import urlopen
from flask_socketio import SocketIO, emit
from flask import Flask, Response,
render_template

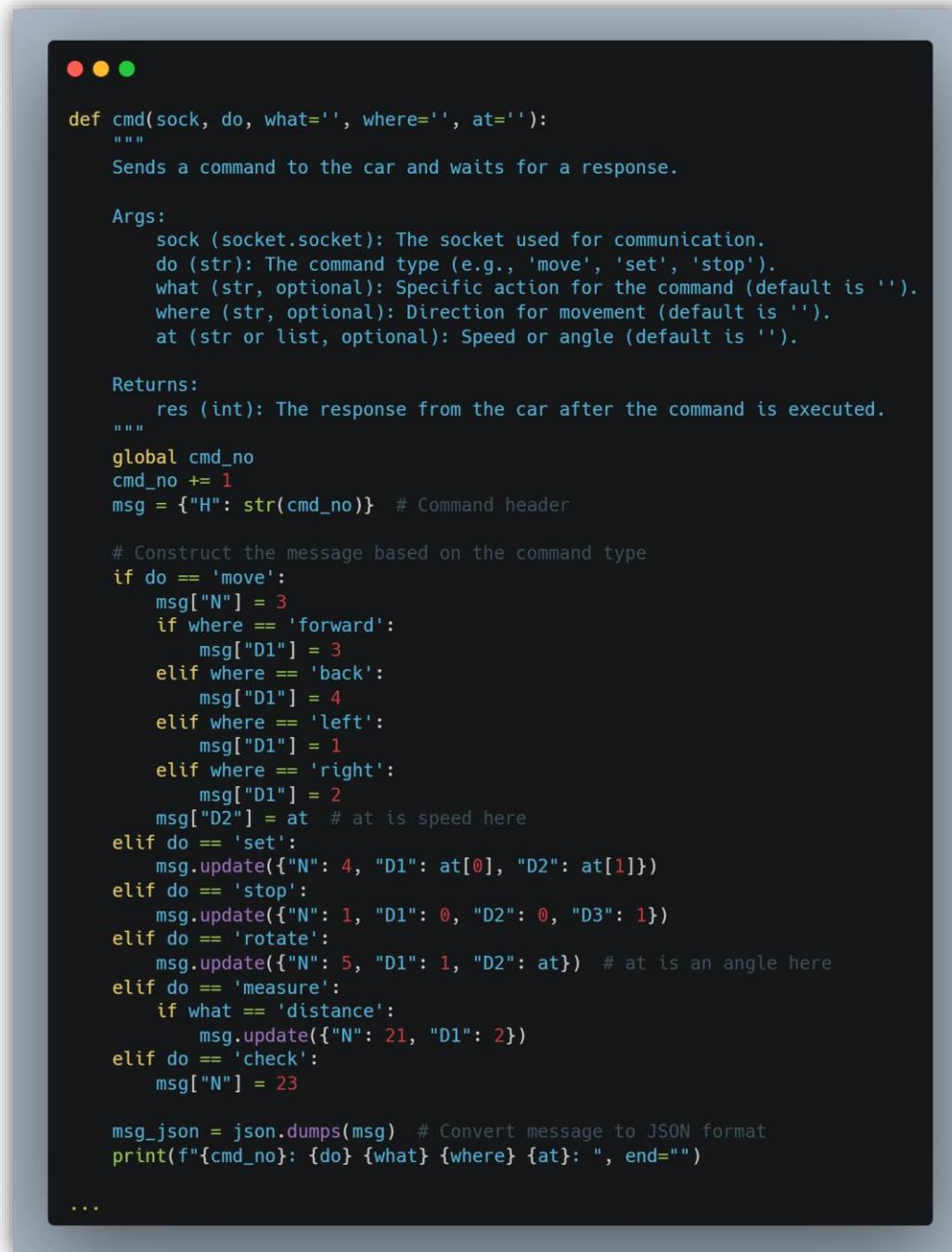
```

Ilustración 6. 9. 1 - Importación de módulos

Posteriormente, la función **CMD()** envía comandos al Smart Robot a través de un socket, y espera una respuesta. Recibe varios parámetros: **sock** (el socket de comunicación), **do** (el tipo de comando, como 'mover', 'detener', medir), **what** (una acción específica dentro del comando), **where** (la dirección de movimiento, como adelante o atrás), y **at** (la velocidad o ángulo, dependiendo del comando).

En función del tipo de comando, la función construye un mensaje en formato JSON con parámetros específicos, que luego es enviado al automóvil a través del socket. Después de enviar el mensaje, la función espera la respuesta del Smart

Robot, que es procesada y convertida en un valor numérico. Si se produce un error durante el envío, se captura y muestra un mensaje de error en la interfaz de usuario en tiempo real. Finalmente, la respuesta procesada es emitida a la interfaz de usuario, indicando el resultado del comando ejecutado.



```

def cmd(sock, do, what='', where='', at=''):
    """
    Sends a command to the car and waits for a response.

    Args:
        sock (socket.socket): The socket used for communication.
        do (str): The command type (e.g., 'move', 'set', 'stop').
        what (str, optional): Specific action for the command (default is '').
        where (str, optional): Direction for movement (default is '').
        at (str or list, optional): Speed or angle (default is '').

    Returns:
        res (int): The response from the car after the command is executed.
    """
    global cmd_no
    cmd_no += 1
    msg = {"H": str(cmd_no)} # Command header

    # Construct the message based on the command type
    if do == 'move':
        msg["N"] = 3
        if where == 'forward':
            msg["D1"] = 3
        elif where == 'back':
            msg["D1"] = 4
        elif where == 'left':
            msg["D1"] = 1
        elif where == 'right':
            msg["D1"] = 2
        msg["D2"] = at # at is speed here
    elif do == 'set':
        msg.update({"N": 4, "D1": at[0], "D2": at[1]})
    elif do == 'stop':
        msg.update({"N": 1, "D1": 0, "D2": 0, "D3": 1})
    elif do == 'rotate':
        msg.update({"N": 5, "D1": 1, "D2": at}) # at is an angle here
    elif do == 'measure':
        if what == 'distance':
            msg.update({"N": 21, "D1": 2})
    elif do == 'check':
        msg["N"] = 23

    msg_json = json.dumps(msg) # Convert message to JSON format
    print(f"{cmd_no}: {do} {what} {where} {at}: ", end="")
    ...

```

Ilustración 6. 9. 2 - Función CMD

No obstante, para lograr la comunicación primero se establece una conexión con el Wi-Fi del Smart Robot mediante un socket en la IP "192.168.4.1" y el puerto

"100". Si la conexión es exitosa, muestra un mensaje de confirmación en la interfaz de usuario. Luego, intenta recibir datos del Smart Robot a través del socket y mostrar el estado de la operación.

```
● ● ●

# Connect to car's WiFi
ip = "192.168.4.1"
port = 100
print(f"Connect to {ip}:{port}")
car = socket.socket()

# Try to establish a connection with the car
try:
    car.connect((ip, port))
except:
    socketio.emit(
        'console',
        {
            'type': 'action',
            'color': '#ff0000',
            'data': f"Error: {sys.exc_info()[0]}",
        }
    )
    sys.exit()

socketio.emit(
    'console',
    {
        'type': 'action',
        'color': '#a1ff0a',
        'data': f"Connected to {ip}:{port}",
    }
)

# Read first data from socket
socketio.emit(
    'console',
    {
        'type': 'action',
        'color': '#ff8700',
        'data': "Reading data from the socket...",
    }
)
try:
    data = car.recv(1024).decode() # Receive data from the car
except:
    socketio.emit(
        'console',
        {
            'type': 'action',
            'color': '#ff8700',
            'data': f"Error: {sys.exc_info()[0]}",
        }
)
sys.exit()
```

Ilustración 6. 9. 3 - Comunicación con Smart Robot

La función **evade_obstacle()** maneja la evasión de obstáculos del Smart Robot. Cuando detecta un obstáculo, detiene el robot y usa el sensor para medir las distancias en ambas direcciones (izquierda y derecha). Si ambos lados están despejados, avanza hacia adelante. Si solo un lado está libre, gira hacia ese lado y verifica si hay suficiente espacio para continuar; si no, retrocede. Si ambos lados están bloqueados, retrocede. Después de realizar la maniobra evasiva, hace una verificación final para asegurarse de que el camino esté libre. Si no lo está, retrocede más veces hasta que el camino se despeje o se alcance un número máximo de intentos.



```

# Evasion of obstacles
speed = 100          # Car speed
ang = [90, 45, 135]  # Head rotation angles for sensor
dist = [0, 0, 0]      # Measured distances to obstacles
dist_min = 30         # Minimum distance to obstacle (cm)

# Evasion of obstacles
def evade_obstacle():
    """
    Handles obstacle evasion with smarter behavior to avoid getting stuck in
    corners or retrying unnecessary actions.
    """
    socketio.emit(
        'console',
        {
            'type': 'action',
            'color': '#147df5',
            'data': "Obstacle detected. Evading...",
        }
    )
    cmd(car, do='stop')  # Stop the car

    # Rotate the sensor to left and right to measure distances
    for i in [1, 2]:
        cmd(car, do='rotate', at=ang[i])
        dist[i] = cmd(car, do='measure', what='distance')
    cmd(car, do='rotate', at=90)  # Re-center the sensor

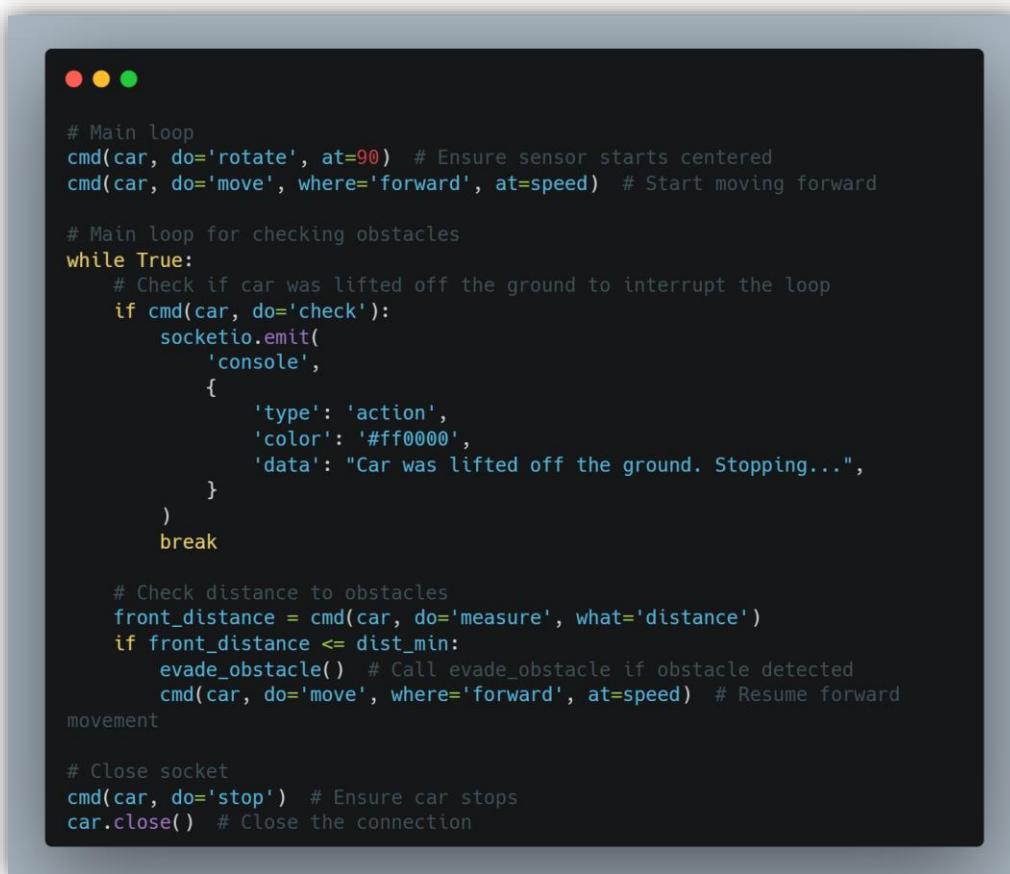
```

Ilustración 6. 9. 4 - Función `evade_obstacles`

Finalmente, se implementa un ciclo principal donde el Smart Robot se mueve hacia adelante y constantemente verifica si hay obstáculos en su camino. Primero, asegura que el sensor esté centrado y luego comienza a mover el Smart Robot hacia adelante a una velocidad establecida. En cada iteración del ciclo:

1. Verifica si el Smart Robot fue levantado del suelo, en cuyo caso detiene el ciclo y emite un mensaje de advertencia.

2. Mide la distancia frente al robot utilizando el sensor. Si la distancia es menor que el valor mínimo (dist_min), significa que hay un obstáculo cerca.
3. En caso de detectar un obstáculo, llama a la función evade_obstacle() para que el Smart Robot evite el obstáculo. Después de la maniobra evasiva, retoma su movimiento hacia adelante.
4. El ciclo sigue repitiéndose hasta que el Smart Robot es levantado del suelo o no detecta más obstáculos.



```

# Main loop
cmd(car, do='rotate', at=90) # Ensure sensor starts centered
cmd(car, do='move', where='forward', at=speed) # Start moving forward

# Main loop for checking obstacles
while True:
    # Check if car was lifted off the ground to interrupt the loop
    if cmd(car, do='check'):
        socketio.emit(
            'console',
            {
                'type': 'action',
                'color': '#ff0000',
                'data': "Car was lifted off the ground. Stopping...",
            }
        )
        break

    # Check distance to obstacles
    front_distance = cmd(car, do='measure', what='distance')
    if front_distance <= dist_min:
        evade_obstacle() # Call evade_obstacle if obstacle detected
        cmd(car, do='move', where='forward', at=speed) # Resume forward movement

    # Close socket
    cmd(car, do='stop') # Ensure car stops
    car.close() # Close the connection

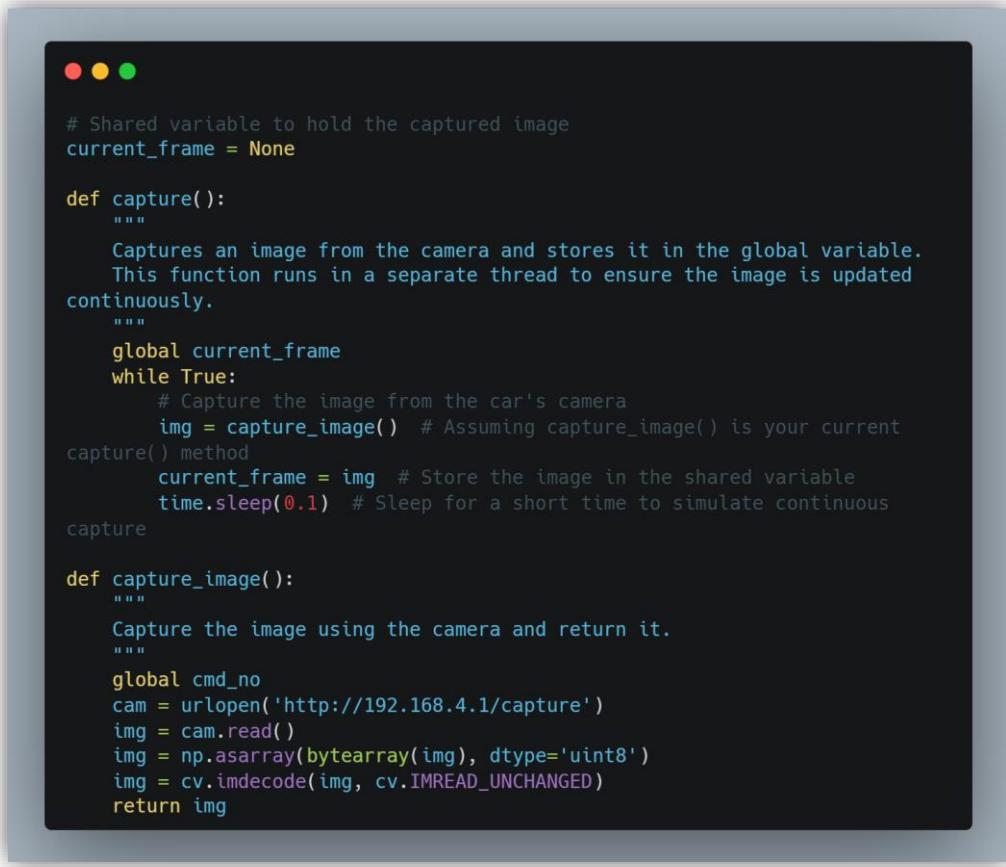
```

Ilustración 6. 9. 5 - Función ciclo principal

Asimismo, como se requería mostrar la trasmisión de video del Smart Robot a la aplicación web, se desarrolló este código para capturar imágenes de manera continua desde la cámara, almacenándolas en una variable global llamada current_frame.

La función **capture_image()** realiza la captura de la imagen a través de una solicitud HTTP a la cámara del ESP32-S3, descarga la imagen en formato de bytes, y luego la convierte en un objeto de imagen utilizando OpenCV. La función **capture()**

corre en un hilo separado, y en cada ciclo captura una nueva imagen utilizando capture_image(), la almacena en la variable current_frame y luego espera 0.1 segundos antes de capturar la siguiente. Esto asegura que la imagen en current_frame se actualice continuamente.



```
# Shared variable to hold the captured image
current_frame = None

def capture():
    """
    Captures an image from the camera and stores it in the global variable.
    This function runs in a separate thread to ensure the image is updated
    continuously.
    """
    global current_frame
    while True:
        # Capture the image from the car's camera
        img = capture_image() # Assuming capture_image() is your current
        capture() method
        current_frame = img # Store the image in the shared variable
        time.sleep(0.1) # Sleep for a short time to simulate continuous
        capture

def capture_image():
    """
    Capture the image using the camera and return it.
    """
    global cmd_no
    cam = urlopen('http://192.168.4.1/capture')
    img = cam.read()
    img = np.asarray(bytarray(img), dtype='uint8')
    img = cv.imdecode(img, cv.IMREAD_UNCHANGED)
    return img
```

Ilustración 6. 9. 6 - Función de captura

Luego para mostrar la aplicación, se establece un servidor web utilizando Flask para transmitir en tiempo real la imagen capturada por la cámara del robot. La ruta /video_feed permite que el navegador reciba un flujo continuo de imágenes JPEG, que se envían a través de la función generate(). La imagen más reciente se obtiene de la variable global current_frame. Además, la ruta principal / carga una página web donde se despliega tanto la transmisión como los datos recibidos.

Para manejar todo esto en paralelo, se crean dos hilos: uno ejecuta el servidor Flask y otra captura imágenes desde la cámara del robot. Ambos hilos se inician como hilos de fondo para que el programa principal pueda seguir ejecutándose sin esperar a que terminen.

```

@app.route('/video_feed')
def video_feed():
    """
    A Flask route to stream the current image to the browser.
    """
    def generate():
        global current_frame
        while True:
            if current_frame is not None:
                # Convert the image to JPEG for streaming
                ret, jpeg = cv.imencode('.jpg', current_frame)
                if ret:
                    # Return the image in the appropriate format for Flask streaming
                    yield (b'--frame\r\n'
                           b'Content-Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n\r\n')
                    time.sleep(0.1) # Sleep to reduce CPU usage
    return Response(generate(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/')
def console_log():
    return render_template('app_1.html')

# Start the Flask app in a separate thread
def start_flask():
    socketio.run(app, host='0.0.0.0', port=5050, allow_unsafe_werkzeug=True)

# Start Flask server in a new thread
flask_thread = threading.Thread(target=start_flask)
flask_thread.daemon = True # Daemonize the thread to allow the main program to exit
flask_thread.start()

# Start the camera capture in a separate thread

capture_thread = threading.Thread(target=capture)
capture_thread.daemon = True # Daemonize the thread to allow the main program to exit
capture_thread.start()

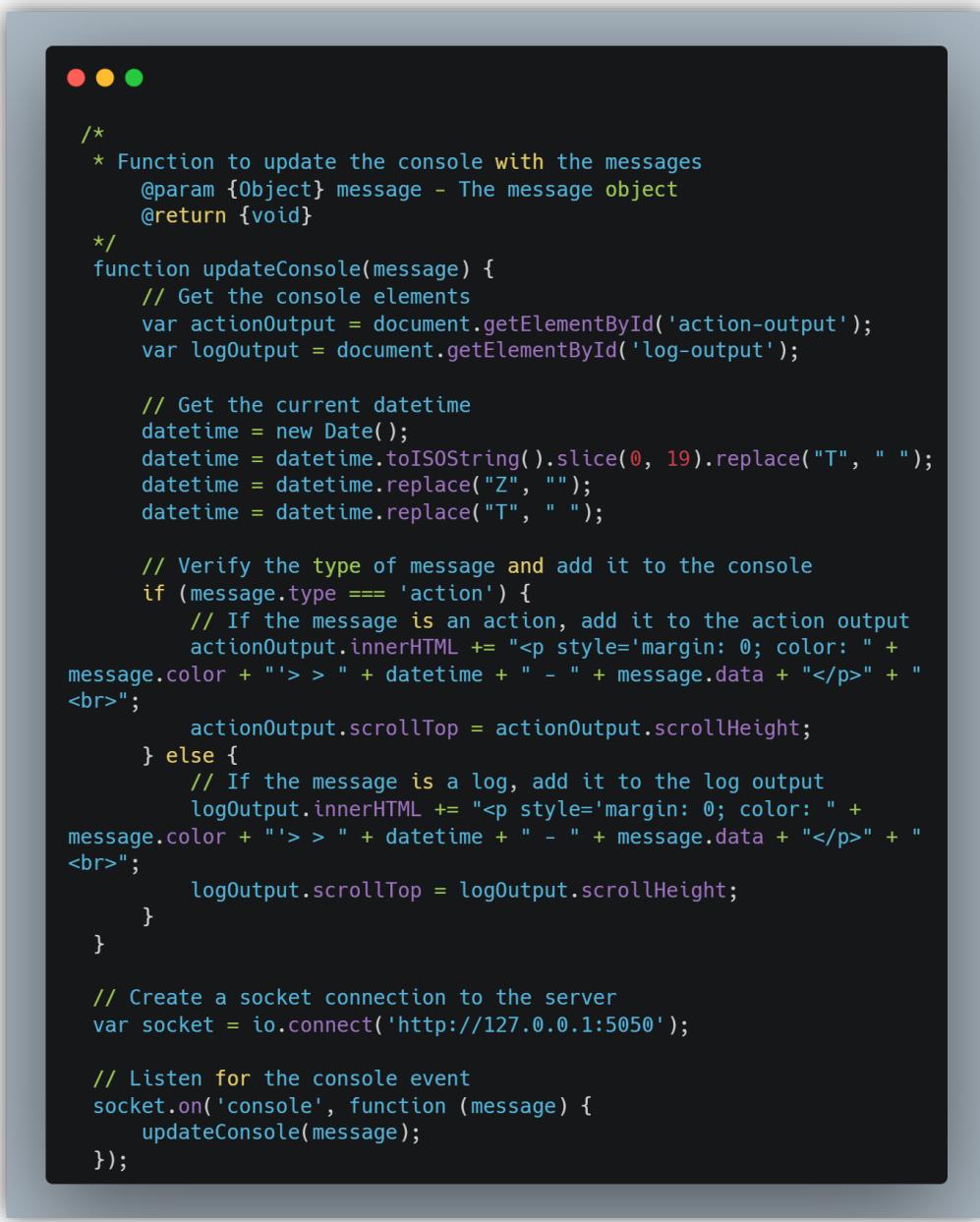
```

Ilustración 6. 9. 7 - Aplicación Flask

Posteriormente, del lado de la vista, se desarrolló un código en JavaScript que gestiona la actualización de los mensajes que son emitidos desde la aplicación de Python mediante el web socket en tiempo real. Se conecta a un servidor utilizando Socket.IO en `http://127.0.0.1:5050` siendo esta nuestra dirección de la aplicación Flask. Cuando el servidor emite un evento emit con etiqueta `console` y con un mensaje, el código recibe ese mensaje y lo agrega a la consola en el navegador.

Dependiendo del tipo de mensaje (`action` o `log`), se muestra en diferentes secciones de la página (`action-output` o `log-output`), siendo `action` para las acciones que con base al entorno realiza el Smart Robot, y `log` para las respuestas de las peticiones enviadas al Smart Robot. Además, se incluye la fecha y hora en que

se recibe el mensaje, y el contenido de la consola se desplaza hacia abajo automáticamente para mostrar el último mensaje.

A screenshot of a terminal window with a dark background. At the top, there are three colored window control buttons (red, yellow, green). Below them is the terminal command line. The code displayed is a Node.js script. It starts with a multi-line comment block. Inside, it defines a function `updateConsole` that takes a `message` object as a parameter. The function retrieves the `action-output` and `log-output` elements from the DOM. It then creates a `datetime` variable and formats it into ISO string format. The script then checks if the message type is 'action'. If so, it appends the message data to the `actionOutput` element's innerHTML, setting the margin and color based on the message color. It also sets the scroll top of the `actionOutput` to its scroll height. If the message type is not 'action', it appends the message data to the `logOutput` element's innerHTML, also setting the margin and color. Finally, it sets the scroll top of the `logOutput` to its scroll height. The script then creates a socket connection to 'http://127.0.0.1:5050'. It listens for a 'console' event on the socket, which triggers the `updateConsole` function with the received message.

```
/*
 * Function to update the console with the messages
 * @param {Object} message - The message object
 * @return {void}
 */
function updateConsole(message) {
    // Get the console elements
    var actionOutput = document.getElementById('action-output');
    var logOutput = document.getElementById('log-output');

    // Get the current datetime
    datetime = new Date();
    datetime = datetime.toISOString().slice(0, 19).replace("T", " ");
    datetime = datetime.replace("Z", "");
    datetime = datetime.replace("T", " ");

    // Verify the type of message and add it to the console
    if (message.type === 'action') {
        // If the message is an action, add it to the action output
        actionOutput.innerHTML += "<p style='margin: 0; color: " +
            message.color + "'> > " + datetime + " - " + message.data + "</p>" + "<br>";
        actionOutput.scrollTop = actionOutput.scrollHeight;
    } else {
        // If the message is a log, add it to the log output
        logOutput.innerHTML += "<p style='margin: 0; color: " +
            message.color + "'> > " + datetime + " - " + message.data + "</p>" + "<br>";
        logOutput.scrollTop = logOutput.scrollHeight;
    }
}

// Create a socket connection to the server
var socket = io.connect('http://127.0.0.1:5050');

// Listen for the console event
socket.on('console', function (message) {
    updateConsole(message);
});
```

Ilustración 6. 9. 8 - Web Socket para la vista

Por último, para mostrar la transmisión en tiempo real, se implementó una etiqueta de imagen cuyo atributo **src** apunta al recurso **video_feed**. Este recurso se encarga de enviar continuamente los fotogramas capturados por la cámara, permitiendo que la imagen se actualice en tiempo real en la interfaz web.



```
<div class="pannel-body">
  <div class="post" style="text-align: center;">
    
  </div>
</div>
```

Ilustración 6. 9. 9 - Etiqueta para mostrar trasmisión de video

Codificación del algoritmo de reconocimiento y seguimiento de esferas de colores con OpenCV

El siguiente código reutiliza gran parte de las funciones explicadas previamente, como la función CMD, la aplicación Flask, el WebSockets y el ciclo principal, adaptándolas para integrar un sistema de reconocimiento y seguimiento de esferas basado en la detección de colores específicos dentro de un rango definido. Además, incluye la capacidad de calcular la trayectoria hacia la esfera y seguirla, mientras implementa la lógica de evasión de obstáculos durante su desplazamiento. Esta ampliación permite que el robot no solo detecte y reconozca objetos, sino también interactúe con ellos de manera dinámica y autónoma. Asimismo, el código pertenece a (Michael, 2021), y se realizaron las modificaciones necesarias para el funcionamiento del Smart Robot.

Primeramente, se realizaron unas modificaciones a la función que captura la imagen; se selecciona un filtro de color utilizando la función `switch_color('red2')`, en este caso se definió cuatro colores: azul, verde, y dos tonos de rojo. Este filtro establece un rango en el espacio de color HSV que define los límites inferiores y superiores del color deseado.

El procesamiento continúa con el filtrado de la imagen para aislar los objetos que coinciden con el color especificado. Primero, se aplica un filtro de desenfoque mediano (`cv.medianBlur`) que reduce el ruido, eliminando detalles innecesarios que podrían interferir en el análisis. Luego, la imagen se convierte al espacio de color HSV (`cv.cvtColor`), ideal para segmentar colores ya que permite trabajar directamente con tonalidades. Usando `cv.inRange`, se genera una máscara binaria donde los píxeles que se encuentran dentro del rango definido en `lu_color_vision` aparecen en blanco, mientras que el resto queda en negro.

Finalmente, se extraen los contornos de los objetos detectados en la máscara. Con `cv.findContours`, se detectan los bordes de los objetos en la imagen binaria. Estos contornos serán utilizados posteriormente para identificar y calcular propiedades específicas para reconocer en este caso cuerpos esféricos, que, en nuestro caso, son pelotas que precisamente son de colores distintos.

```

# Switch color filter before processing
lu_color_vision = switch_color('red2')

# Fetch image from the camera
cam = urlopen('http://192.168.4.1/capture')
img = cam.read()
img = np.asarray(bytearray(img), dtype='uint8')
img = cv.imdecode(img, cv.IMREAD_UNCHANGED)

# Filter image by color
mask = cv.medianBlur(img, 5)
img_hsv = cv.cvtColor(mask, cv.COLOR_BGR2HSV)
mask = cv.inRange(img_hsv, lu_color_vision[0], lu_color_vision[1])

# Detect contours
mask = cv.erode(mask, None, iterations=2)
mask = cv.dilate(mask, None, iterations=2)
cont = cv.findContours(mask.copy(), cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
cont = imutils.grab_contours(cont)

```

Ilustración 6. 10. 1 - Reconocimiento de contorno

Posteriormente, se calcula la distancia y el ángulo hacia la pelota detectada en la imagen mediante operaciones geométricas. La distancia en el eje y se determina proporcionalmente utilizando un factor de escala relacionado con la posición vertical del objeto en la imagen y la línea del horizonte. Para objetos con coordenadas x negativas, se aplica una corrección que reduce el valor, ajustando la perspectiva. La distancia en el eje x se calcula como una proporción directa del desplazamiento horizontal multiplicado por la distancia en y. La distancia total hacia el objeto se obtiene utilizando el teorema de Pitágoras. Finalmente, el ángulo hacia la pelota se calcula como la tangente inversa de la distancia en x sobre la distancia en y, convirtiendo este valor de radianes a grados para interpretarlo más fácilmente.

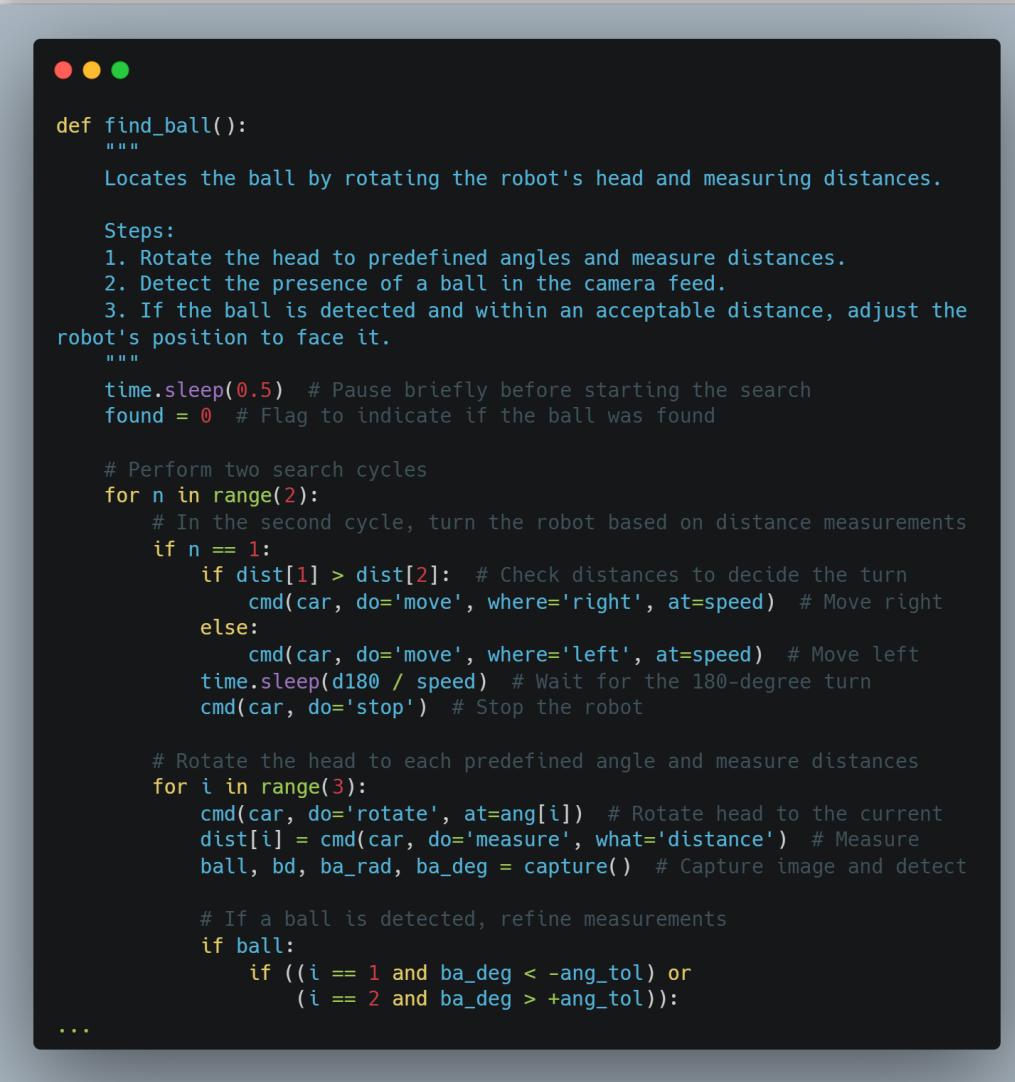
```

# Calculate distance and angle to the ball
if ball:
    cv.drawContours(img, cont, nc, (0, 0, 255), 1) # Highlight contour in red
    cv.circle(img, center, 1, (0, 0, 255), 2) # Mark the center of the ball
    cv.putText(img, '(' + str(xc) + ', ' + str(yc) + ')', center,
               cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv.LINE_AA)
    dy = 4.31 * (745.2 + yc) / (yh - yc) # Calculate distance along the y-axis
    if xc < 0: dy = dy * (1 - xc / 1848) # Apply correction for negative x-coord
    dx = 0.00252 * xc * dy # Calculate distance along the x-axis
    dist = np.sqrt(dx**2 + dy**2) # Calculate the total distance
    ang_rad = np.arctan(dx / dy) # Calculate the angle in radians
    ang_deg = round(ang_rad * 180 / np.pi) # Convert the angle to degrees

```

Ilustración 6. 10. 2 - Cálculo de distancia y ángulo hacia la pelota

Asimismo, la función **find_ball** realiza una búsqueda de la pelota utilizando la cámara y los sensores de distancia, combinando movimientos de su cabeza y cuerpo para localizarla y alinearse con ella. Primero, el Smart Robot realiza dos ciclos de búsqueda: en el primero, rota su cabeza hacia tres ángulos predefinidos, mide distancias con sensores y analiza la cámara en busca de la pelota; si se detecta, ajusta el ángulo y refina la posición midiendo nuevamente. En el segundo ciclo, si la pelota no fue encontrada, el robot gira sobre sí mismo 180° para buscar en otra dirección. Si encuentra la pelota, calcula la distancia y el ángulo de alineación necesarios, ajusta su dirección mediante movimientos hacia la izquierda o derecha y actualiza su orientación para enfrentar la pelota correctamente. Si la búsqueda falla, el robot simplemente centra su cabeza como posición inicial.



```

def find_ball():
    """
    Locates the ball by rotating the robot's head and measuring distances.

    Steps:
    1. Rotate the head to predefined angles and measure distances.
    2. Detect the presence of a ball in the camera feed.
    3. If the ball is detected and within an acceptable distance, adjust the
       robot's position to face it.
    """
    time.sleep(0.5) # Pause briefly before starting the search
    found = 0 # Flag to indicate if the ball was found

    # Perform two search cycles
    for n in range(2):
        # In the second cycle, turn the robot based on distance measurements
        if n == 1:
            if dist[1] > dist[2]: # Check distances to decide the turn
                cmd(car, do='move', where='right', at=speed) # Move right
            else:
                cmd(car, do='move', where='left', at=speed) # Move left
            time.sleep(d180 / speed) # Wait for the 180-degree turn
            cmd(car, do='stop') # Stop the robot

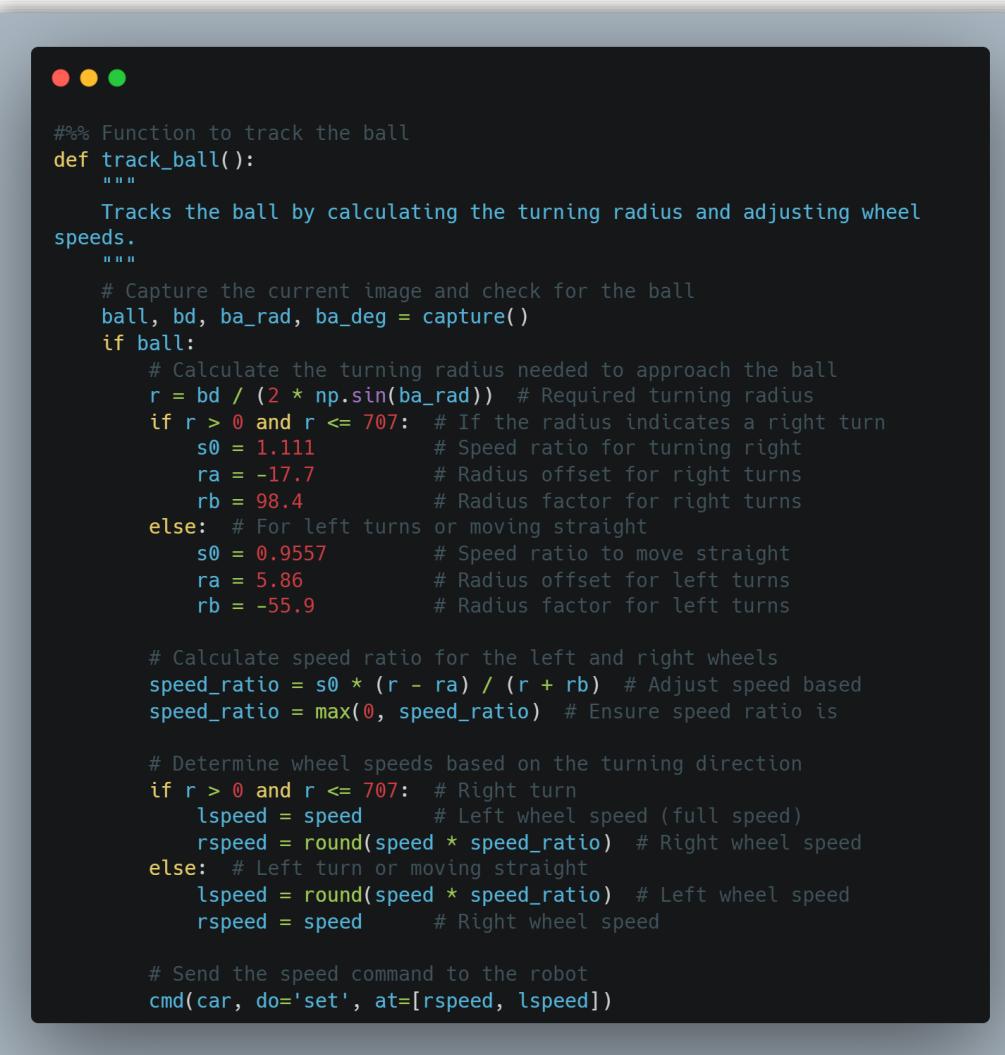
        # Rotate the head to each predefined angle and measure distances
        for i in range(3):
            cmd(car, do='rotate', at=ang[i]) # Rotate head to the current
            dist[i] = cmd(car, do='measure', what='distance') # Measure
            ball, bd, ba_rad, ba_deg = capture() # Capture image and detect

        # If a ball is detected, refine measurements
        if ball:
            if ((i == 1 and ba_deg < -ang_tol) or
                (i == 2 and ba_deg > +ang_tol)):
                ...

```

Ilustración 6. 10. 3 - Función *find_ball*

Finalmente, la función **track_ball** permite al Smart Robot seguir la pelota ajustando la velocidad de sus ruedas en función del radio de giro necesario para acercarse a ella. Primero, captura una imagen para determinar si hay una pelota visible y, en caso afirmativo, calcula el radio de giro requerido usando la distancia de la pelota y el ángulo en radianes. Dependiendo de este radio, la función decide si el giro debe ser a la derecha, a la izquierda, o si debe avanzar en línea recta. Para giros a la derecha, establece una relación de velocidad específica y ajusta las velocidades de las ruedas de manera que la izquierda mantenga una velocidad más alta.



```

#% Function to track the ball
def track_ball():
    """
    Tracks the ball by calculating the turning radius and adjusting wheel
    speeds.
    """

    # Capture the current image and check for the ball
    ball, bd, ba_rad, ba_deg = capture()
    if ball:
        # Calculate the turning radius needed to approach the ball
        r = bd / (2 * np.sin(ba_rad)) # Required turning radius
        if r > 0 and r <= 707: # If the radius indicates a right turn
            s0 = 1.111 # Speed ratio for turning right
            ra = -17.7 # Radius offset for right turns
            rb = 98.4 # Radius factor for right turns
        else: # For left turns or moving straight
            s0 = 0.9557 # Speed ratio to move straight
            ra = 5.86 # Radius offset for left turns
            rb = -55.9 # Radius factor for left turns

        # Calculate speed ratio for the left and right wheels
        speed_ratio = s0 * (r - ra) / (r + rb) # Adjust speed based
        speed_ratio = max(0, speed_ratio) # Ensure speed ratio is

        # Determine wheel speeds based on the turning direction
        if r > 0 and r <= 707: # Right turn
            lspeed = speed # Left wheel speed (full speed)
            rspeed = round(speed * speed_ratio) # Right wheel speed
        else: # Left turn or moving straight
            lspeed = round(speed * speed_ratio) # Left wheel speed
            rspeed = speed # Right wheel speed

        # Send the speed command to the robot
        cmd(car, do='set', at=[rspeed, lspeed])
    """
    
```

Ilustración 6. 10. 4 - Función track_ball

Creación de contenedores Docker para despliegue de las aplicaciones

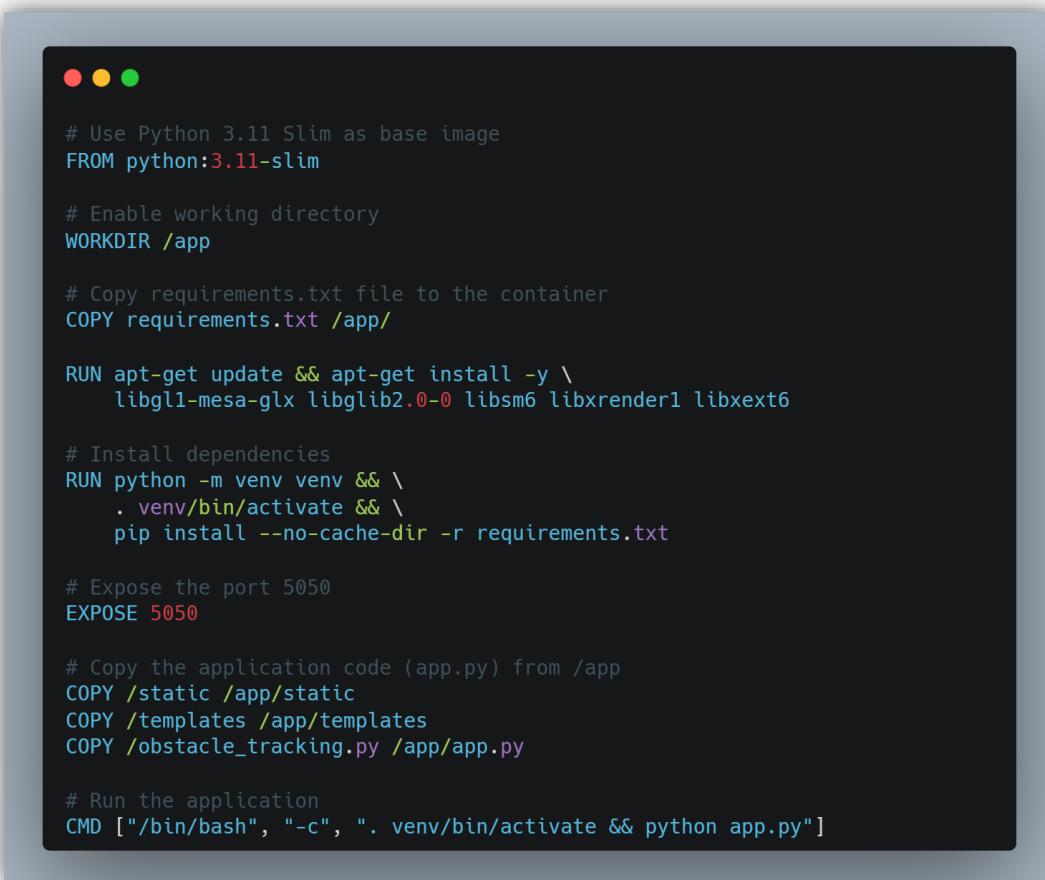
Para simplificar el despliegue de ambas aplicaciones en la Raspberry, se decidió utilizar **Docker**, una plataforma que permite empaquetar aplicaciones y sus dependencias en contenedores aislados. Esto facilita el despliegue, ya que elimina problemas de compatibilidad entre entornos y garantiza que la aplicación se ejecute de manera consistente, independientemente del sistema operativo o configuración del host.

Además, en entornos Linux, Docker es especialmente eficiente debido a su integración nativa con el kernel. Para implementar esta solución, primero se definieron las imágenes necesarias para cada aplicación y se diseñó un archivo de orquestación **docker-compose** que permite gestionar fácilmente los servicios asociados.

El primer archivo **Dockerfile.app1** dedicado a la aplicación de reconocimiento y evasión de obstáculos, configura una imagen Docker para ejecutar una aplicación en Python. Comienza con una base de Python 3.11 Slim, que es ligera y adecuada para entornos de contenedores. Establece un directorio de trabajo dentro del contenedor y copia el archivo requirements.txt para instalar las dependencias necesarias, junto con librerías del sistema requeridas para aplicaciones que utilizan OpenCV.

A continuación, crea y activa un entorno virtual de Python, instala las dependencias del archivo requirements.txt y expone el puerto 5050 para hacer accesible la aplicación desde fuera del contenedor. Finalmente, copia los archivos y directorios relevantes de la aplicación al contenedor y establece el comando para ejecutar la aplicación, asegurándose de activar el entorno virtual antes de iniciar el script principal.

El segundo archivo **Dockerfile.app2** está dedicado a la aplicación de reconocimiento y seguimiento de esferas de colores, es exactamente igual al archivo de Dockerfile.app2, a diferencia que copia y ejecuta el segundo script dedicado a lo dicho.



```
# Use Python 3.11 Slim as base image
FROM python:3.11-slim

# Enable working directory
WORKDIR /app

# Copy requirements.txt file to the container
COPY requirements.txt /app/

RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx libglib2.0-0 libsm6 libxrender1 libxext6

# Install dependencies
RUN python -m venv venv && \
    . venv/bin/activate && \
    pip install --no-cache-dir -r requirements.txt

# Expose the port 5050
EXPOSE 5050

# Copy the application code (app.py) from /app
COPY /static /app/static
COPY /templates /app/templates
COPY /obstacle_tracking.py /app/app.py

# Run the application
CMD ["/bin/bash", "-c", ". venv/bin/activate && python app.py"]
```

Ilustración 6. 11. 1 - Dockerfile.app1 y Dockerfile.app2

El archivo **docker-compose.yml** define la configuración para orquestar los dos contenedores de aplicaciones utilizando Docker. En primer lugar, define dos servicios, app-1 y app-2. Cada servicio tiene su propio contexto de construcción, donde se especifica la ruta al directorio donde se encuentran los archivos de la aplicación, junto con el Dockerfile correspondiente (docker/Dockerfile.app1 y docker/Dockerfile.app2).

Ambos servicios exponen el mismo puerto (5050) dentro del contenedor, pero el puerto en el host es diferente: app-1 mapea el puerto 5050 y app-2 mapea el puerto 5051. Los contenedores se conectan a una red compartida llamada flask-network mediante un controlador de red bridge, lo que les permite comunicarse entre sí.

```
version: '3.8'

services:
  app-1:
    build:
      context: ../ # Specify the context of the build
      dockerfile: docker/Dockerfile.app1 # Specify the Dockerfile to use
    ports:
      - "5050:5050" # Map port 5000 in the container to port 5050 on the
host
    networks:
      - flask-network # Connect the container to the flask-network

  app-2:
    build:
      context: ../ # Specify the context of the build
      dockerfile: docker/Dockerfile.app2 # Specify the Dockerfile to use
    ports:
      - "5050:5050" # Map port 5000 in the container to port 5051 on the
host
    networks:
      - flask-network # Connect the container to the flask-network

networks:
  flask-network
  driver: bridge # Use the bridge network driver
```

Ilustración 6. 11. 2 - Docker Compose

Despliegue de contenedores en Raspberry Pi

Una vez verificado que los contenedores se crearon correctamente en nuestra computadora principal, el siguiente paso fue desplegarlos en la Raspberry Pi. Como Docker ya estaba instalado en este dispositivo, procedimos a ejecutar una serie de pasos. Primero, accedimos a la carpeta que contenía los archivos de Docker, luego generamos las imágenes de ambas aplicaciones asegurándonos de no utilizar la caché para evitar conflictos. Posteriormente, realizamos pruebas intercambiando entre las imágenes para verificar el correcto funcionamiento de ambas aplicaciones.

Accedemos al directorio del proyecto

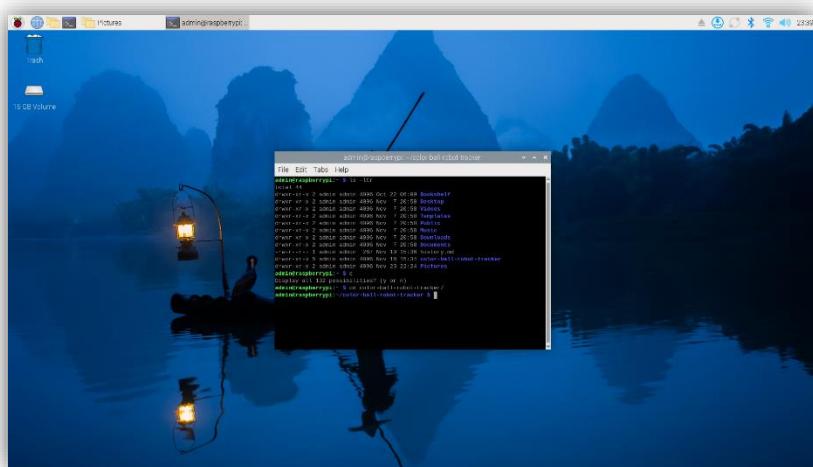


Ilustración 6. 12. 1 – Acceso a directorio de proyecto

Accedemos al directorio de Docker dentro del directorio de Applications

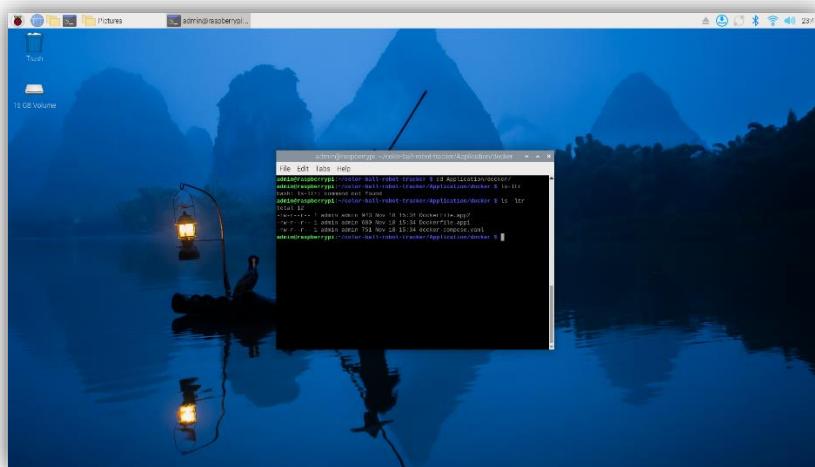


Ilustración 6. 12. 2 - Acceso a directorio de Docker

Generamos las dos imágenes sin cache de los dos Dockerfile

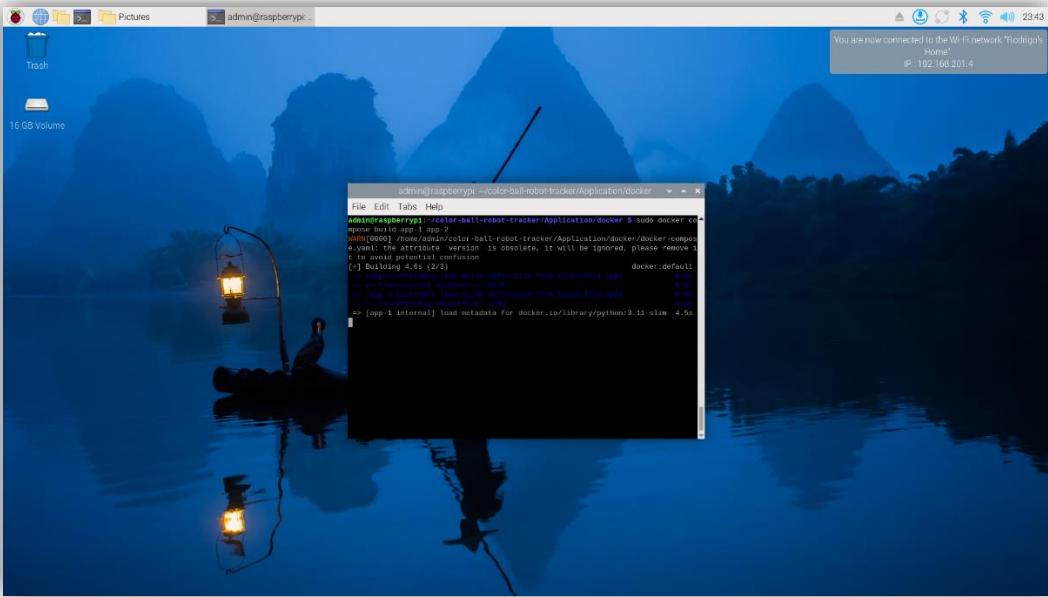
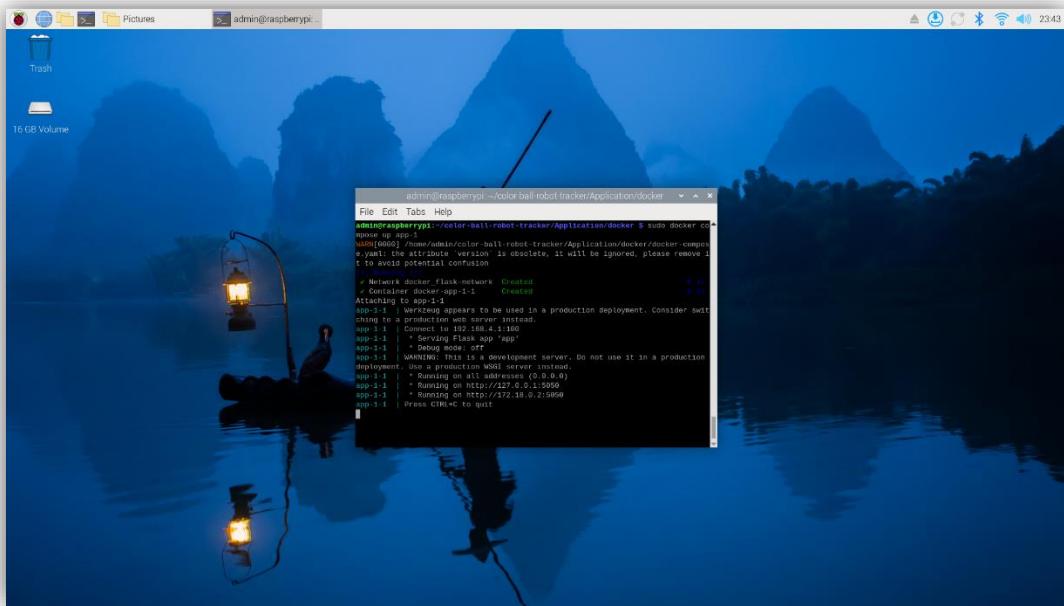


Ilustración 6. 12. 3 - Generación de imágenes de Dockerfile

Creamos los contenedores con base al docker-compose.yaml



Accedemos al localhost en el puerto 5050 para visualizar la primera aplicación

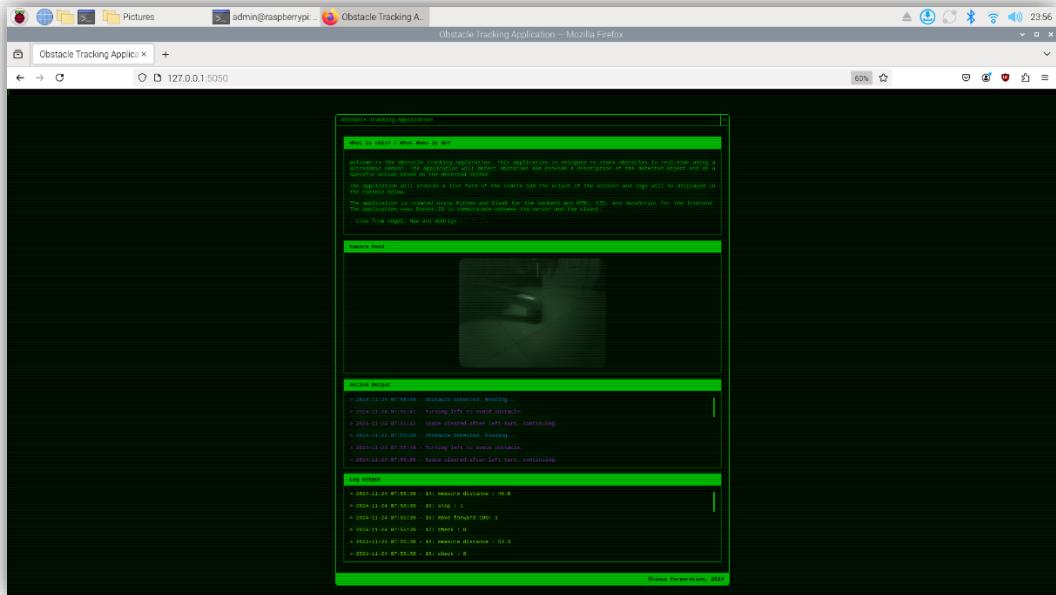


Ilustración 6. 12. 4 - Acceso a primera aplicación

Ejecutamos el segundo contenedor con la segunda aplicación

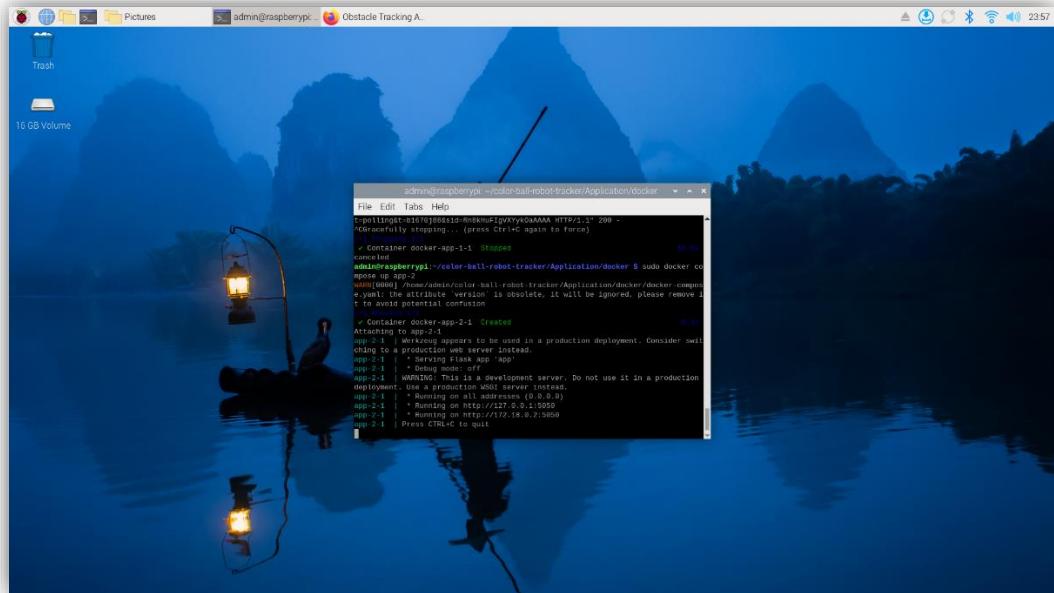


Ilustración 6. 12. 5 - Acceso a segunda aplicación

Accedemos nuevamente al localhost en el puerto 5050 para visualizar la segunda aplicación

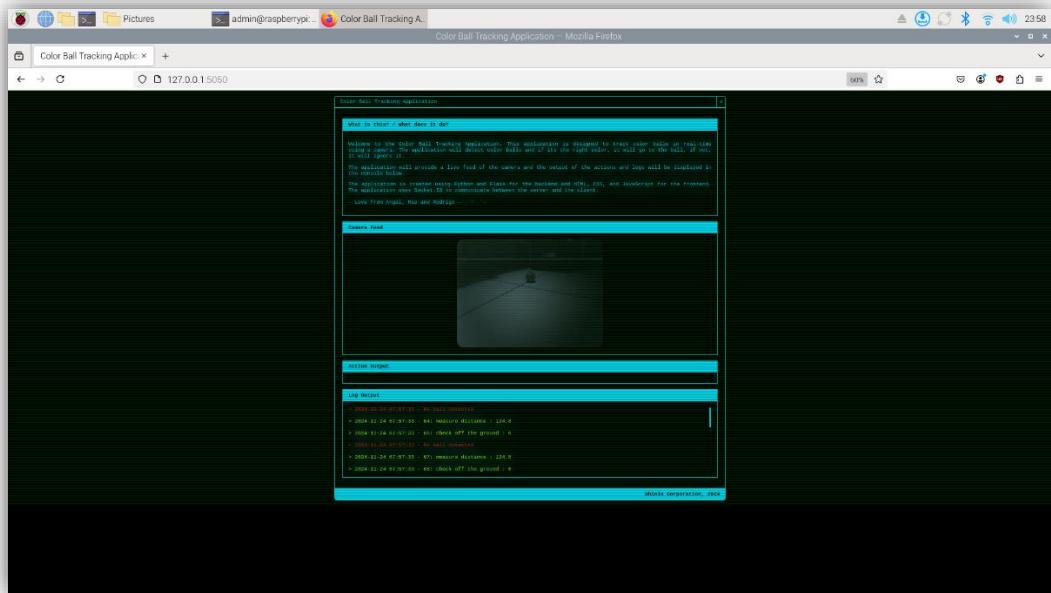


Ilustración 6. 12. 6 - Acceso a la segunda aplicación

Resultados

Funcionamiento del algoritmo de reconocimiento y evasión de obstáculos

El comportamiento del Smart Robot en este algoritmo es bastante neutral y se basa en la detección de obstáculos a una distancia máxima de 20 centímetros. Si no detecta obstáculos, avanza hacia adelante. En caso de encontrar uno, primero gira el cabezal para identificar obstáculos cercanos y determinar la dirección en la que debe girar. Si detecta un obstáculo a la derecha, gira hacia la izquierda y avanza; si ocurre lo contrario, gira hacia la derecha.

En situaciones más complejas, como al enfrentarse a una esquina o cuando encuentra obstáculos en ambas direcciones, el robot realiza maniobras hacia atrás y adelante, girando ligeramente hacia los lados para intentar identificar una posible ruta. Si no logra encontrar una solución después de tres intentos, realiza un giro de 180 grados y reanuda su trayectoria en línea recta.



Ilustración 7. 1. 1 - Analizando entorno

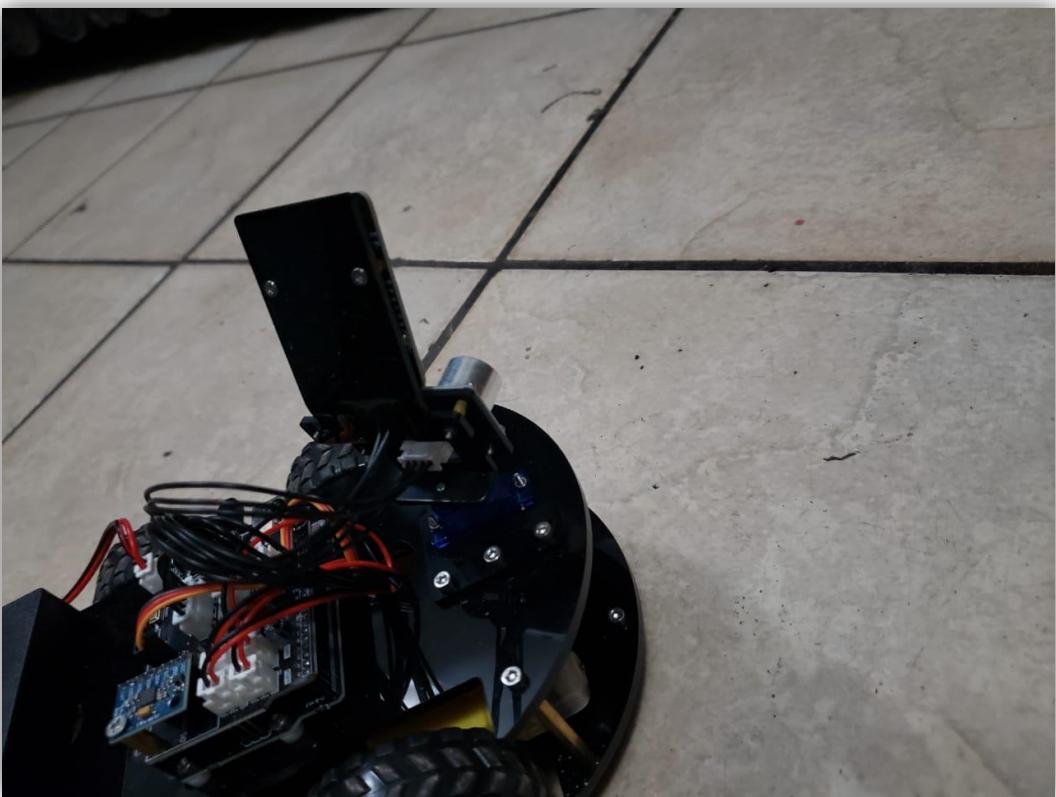


Ilustración 7. 1. 2 - Gira cabezal a la izquierda

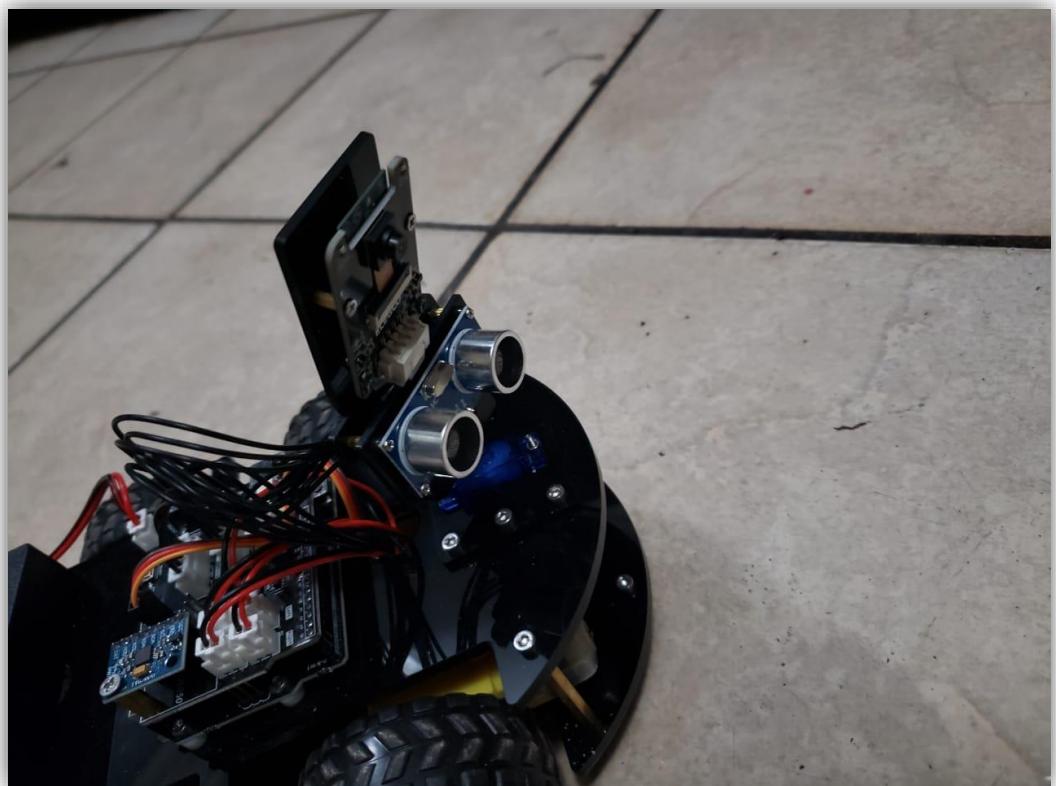


Ilustración 7. 1. 3 - Gira cabezal a la derecha



Ilustración 7. 1. 4 - Avanza hacia adelante



Ilustración 7. 1. 5 - Encuentra obstáculo, gira a la izquierda



Ilustración 7. 1. 6 - Encuentra obstáculo nuevamente



Funcionamiento del algoritmo de reconocimiento y seguimiento de esferas de colores

Con este algoritmo, el robot primero escanea su entorno moviendo el cabezal de izquierda a derecha en busca de una pelota. Si detecta una pelota del color seleccionado, en este caso rojo, calcula la trayectoria necesaria para seguirla y comienza a perseguirla. Por el contrario, si detecta una pelota de otro color, como azul, la considera un obstáculo y no intenta seguirla, mostrándose indiferente hacia su presencia.

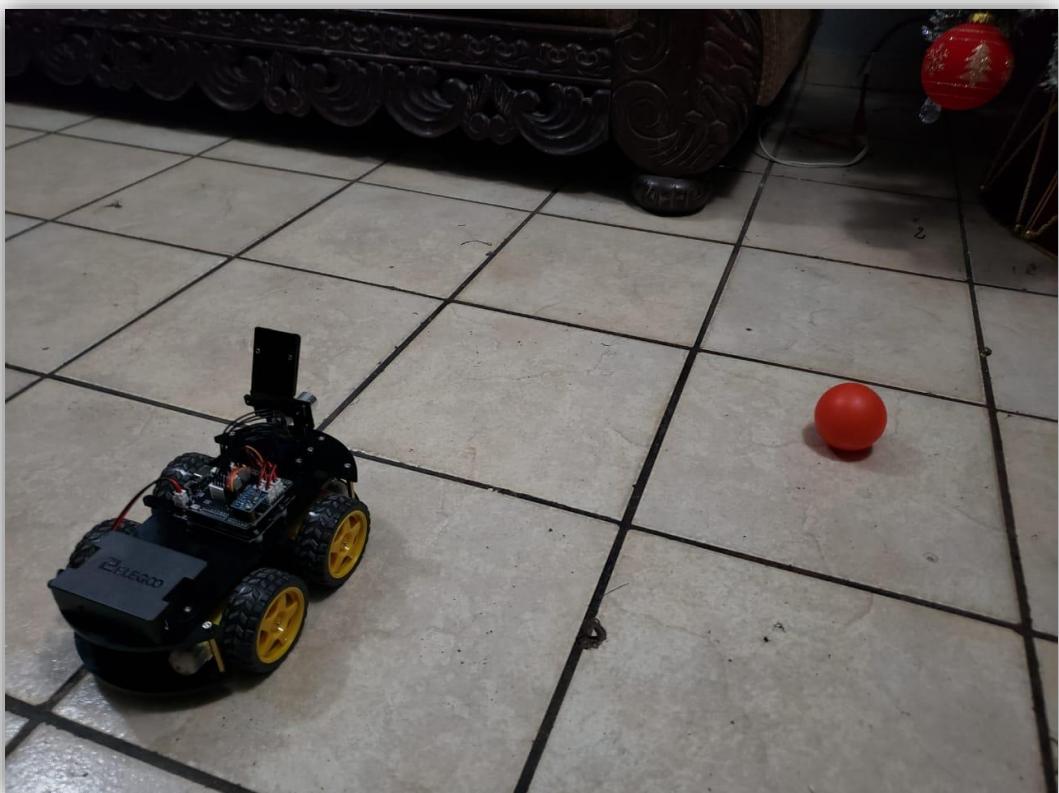


Ilustración 7.2. 1 - Identifica el entorno

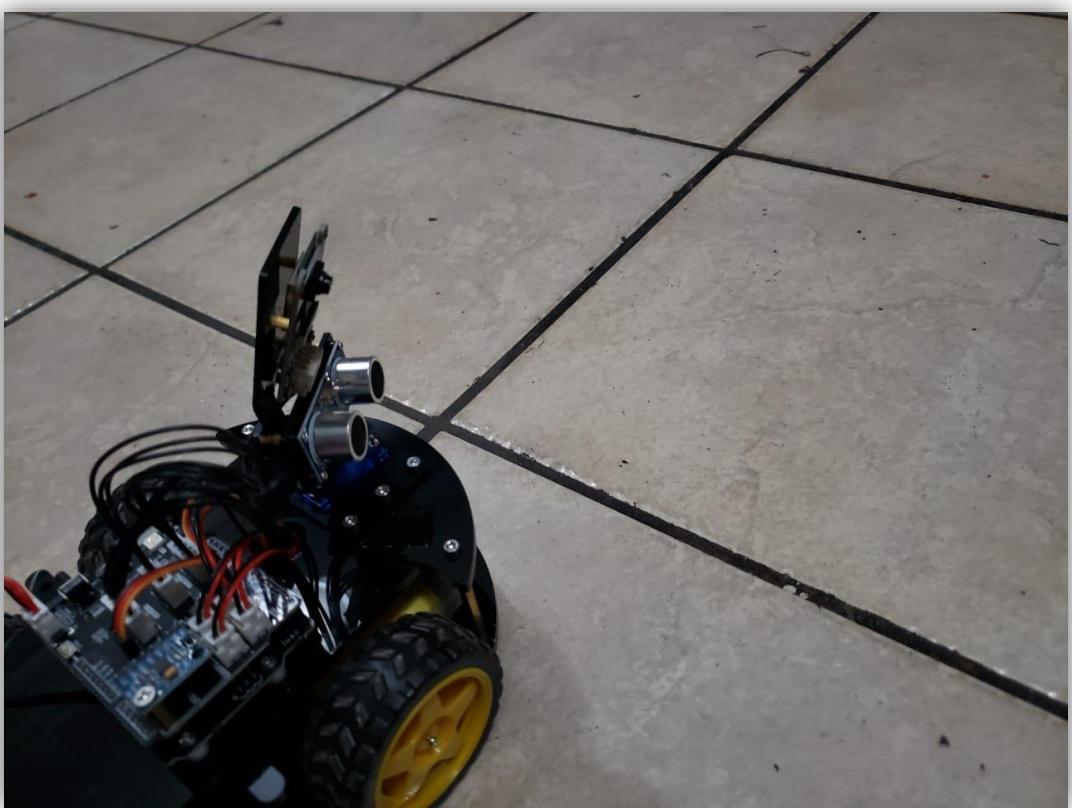


Ilustración 7. 2. 2 - Identifica la pelota roja



Ilustración 7. 2. 3 - Realiza el seguimiento de la pelota roja

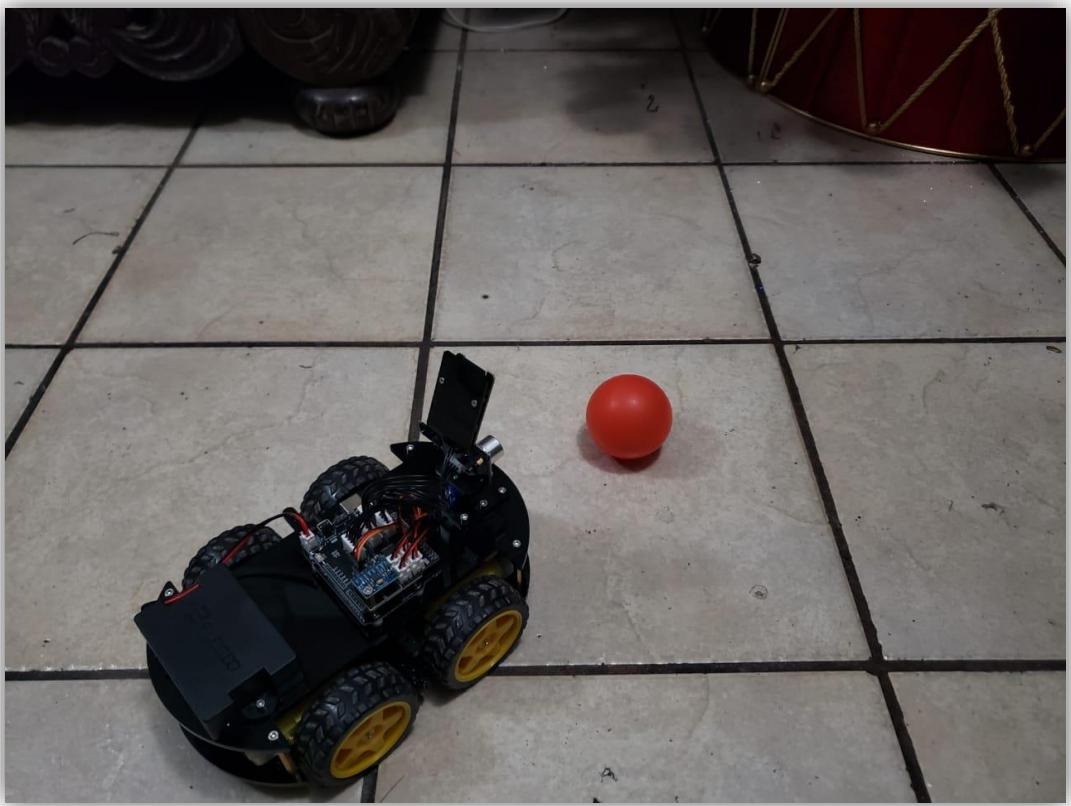


Ilustración 7.2. 4 - Llega hasta la distancia objetivo

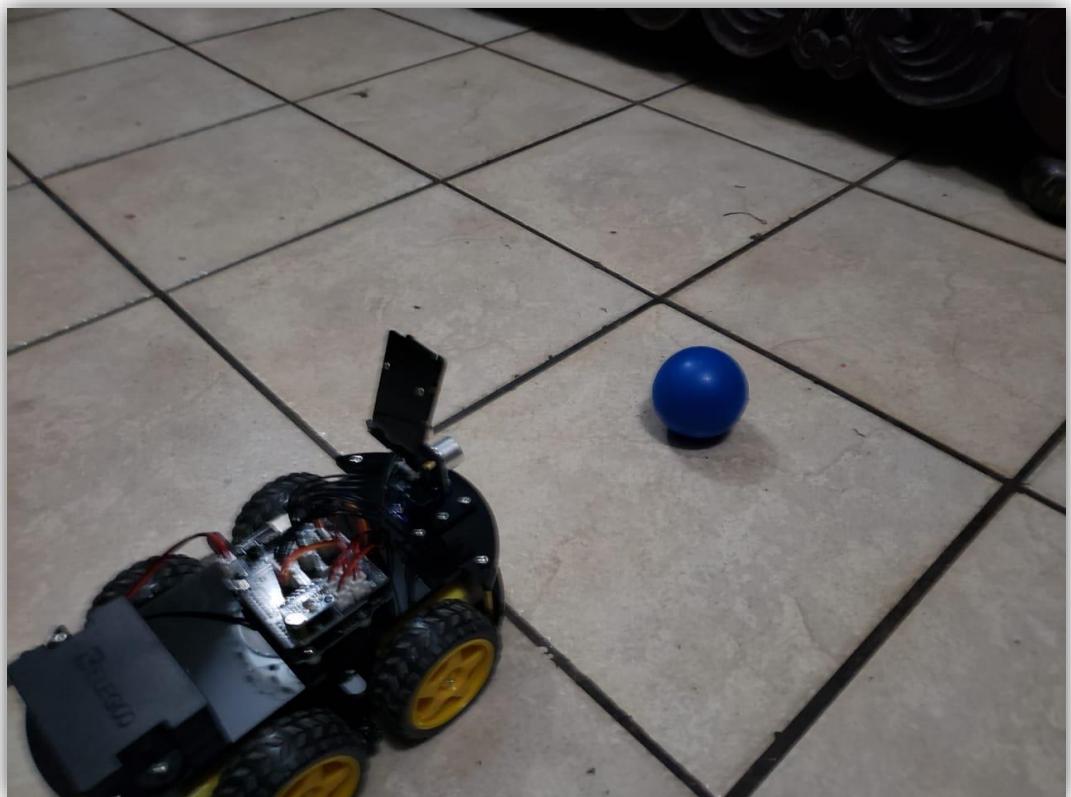


Ilustración 7.2. 5 - Encuentra pelota azul

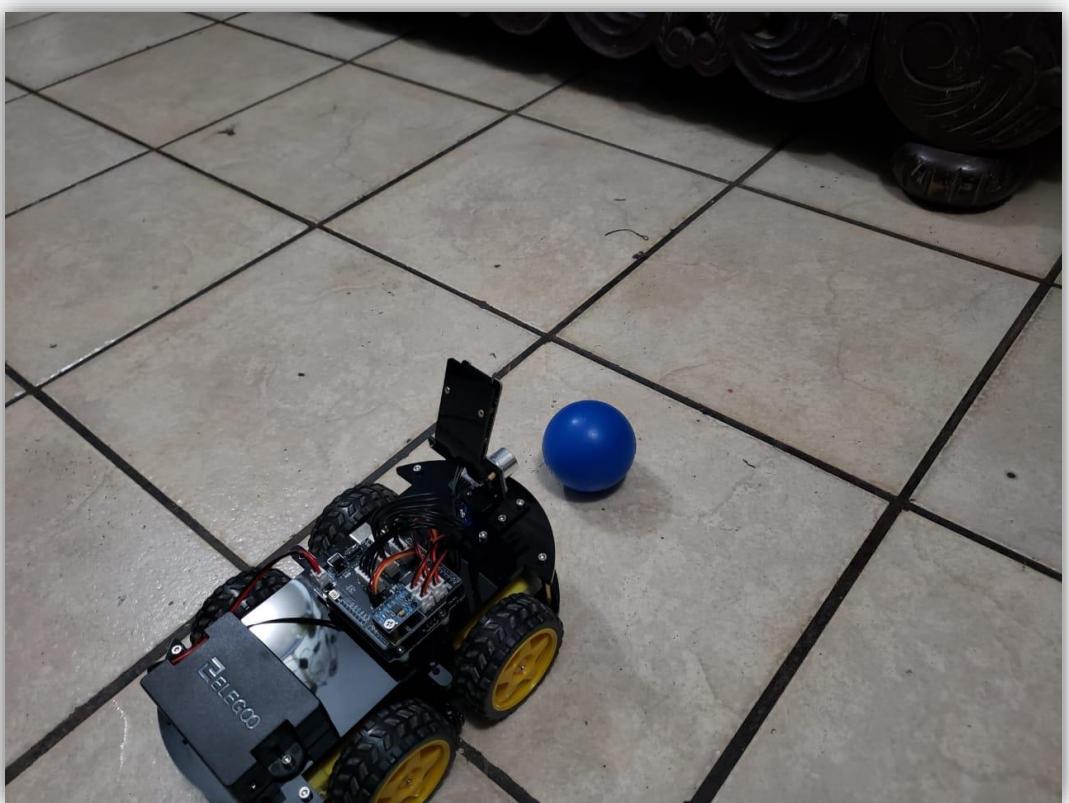


Ilustración 7. 2. 6 - Verifica si es del color objetivo

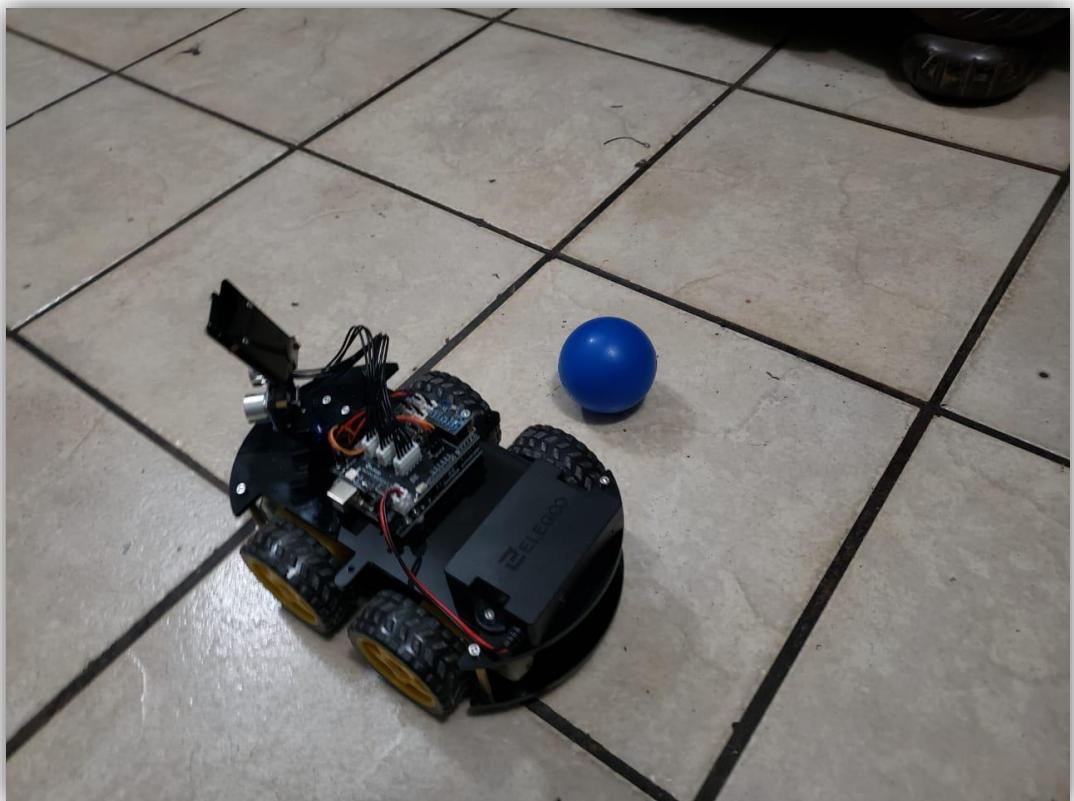


Ilustración 7. 2. 7 - Ignora la pelota azul

Conclusiones

El desarrollo del proyecto con el Elegoo Smart Robot Car V4.0 ofreció una oportunidad única para aplicar tecnologías avanzadas en los campos de la robótica y la inteligencia artificial. A lo largo del proceso, se integraron herramientas y componentes clave que permitieron construir un sistema funcional y eficiente, con la finalidad de lograr la automatización y el procesamiento de datos en tiempo real.

Durante su implementación, se utilizaron tecnologías como Arduino IDE y Python 3.11 Slim, que fueron esenciales para la programación del microcontrolador y el desarrollo de algoritmos avanzados. Esta combinación facilitó la optimización de la comunicación entre los distintos módulos del sistema, así como el procesamiento en tiempo real, lo que resultó crucial para la automatización de las tareas del robot.

Angelica Mae Guadalupe Lualhati

El desarrollo e implementación de sistemas basados en algoritmos de visión artificial y toma de decisiones en robots como el *Smart Robot Car* representa un enfoque práctico y eficiente para abordar desafíos complejos en automatización. Desde el análisis de su entorno hasta la interacción con objetos de interés, como pelotas de colores, el Smart Robot demostró capacidades de detección y seguimiento. La integración de cálculos de trayectoria basados en parámetros espaciales y el uso de estrategias claras para evitar obstáculos le permite operar de manera autónoma y adaptable frente a diversas situaciones del entorno.

El despliegue de aplicaciones complementarias en contenedores Docker, tanto en entornos locales como en un dispositivo Raspberry Pi, ejemplifica la importancia de utilizar tecnologías modernas para la orquestación y portabilidad de software. La implementación de archivos Dockerfile y *docker-compose* facilitó la creación, configuración y ejecución de servicios necesarios para el correcto funcionamiento del sistema.

En definitiva, la combinación de visión por computadora, algoritmos inteligentes y contenedores Docker permitió crear un sistema eficiente que resuelve tareas

específicas como el seguimiento de objetos y la evasión de obstáculos. Sin más que decir, Feliz Navidad.

Luis Rodrigo Barba Navarro

Este proyecto me permitió integrar diversos conceptos de ingeniería en sistemas computacionales, visión por computadora y robótica. A través del uso de OpenCV, logramos en equipo desarrollar un sistema de reconocimiento y seguimiento de esferas de colores, mientras que los sensores ultrasónicos fueron fundamentales para la evasión de obstáculos, permitiendo que el robot interactúe de forma autónoma con su entorno.

Durante el desarrollo, enfrenté varios obstáculos, como la calibración precisa de los sensores y la integración efectiva de los algoritmos de visión y control. Sin embargo, superar estos retos me permitió mejorar mis habilidades en programación, resolución de problemas y en el manejo de herramientas como Python y OpenCV. Además, la implementación del sistema de evasión de obstáculos me enseñó la importancia de considerar la interacción entre hardware y software para lograr un funcionamiento óptimo.

Este proyecto no solo reforzó mi interés en la robótica y la automatización, sino que también me brindó una comprensión más profunda sobre cómo crear soluciones autónomas en el campo de la inteligencia artificial. Como resultado, he obtenido valiosos conocimientos y experiencia que me motivan a seguir explorando nuevas aplicaciones en la robótica y el procesamiento de imágenes.

Angel Uriel Geraldo Armenta

Referencias bibliográficas

¿Qué es Arduino IDE? (s.f.). Obtenido de Tuelectronica: <https://tuelectronica.es/que-es-arduino-ide/>

¿Qué es Docker y cómo funciona? (20 de Enero de 2023). Obtenido de Red Hat: <https://www.redhat.com/es/topics/containers/what-is-docker>

¿Qué es Docker? (s.f.). Obtenido de Amazon Web Services: <https://aws.amazon.com/es/docker/>

¿Qué es la inteligencia artificial (IA)? (s.f.). Obtenido de IBM: <https://www.ibm.com/mx-es/topics/artificial-intelligence>

¿Qué es la inteligencia artificial (IA)? (s.f.). Obtenido de ISO: <https://www.iso.org/es/inteligencia-artificial/que-es-ia>

¿Qué es la Robótica? (14 de Marzo de 2023). Obtenido de Robotesfera: <https://robotesfera.com/robotica>

About OpenCV. (s.f.). Obtenido de OpenCV: <https://opencv.org/about/>

Akbari, M. (s.f.). How to Install CH340 Driver on Windows. Obtenido de ElectroPeak: <https://electropeak.com/learn/how-to-install-ch340-driver/>

Arducam 5MP ESP32 S3 Camera Module, Wi-Fi Bluetooth ESP32-S3 WROOM Development Board. (s.f.). Obtenido de ArduCam: <https://www.arducam.com/product/arducam-5mp-esp32-s3-camera-module-wi-fi-bluetooth-esp32-s3-wroom-development-board/>

Cardellino, F. (20 de Marzo de 2021). La guía definitiva del paquete NumPy para computación científica en Python. Obtenido de freeCodeCamp: <https://www.freecodecamp.org/espanol/news/la-guia-definitiva-del-paquete-numpy-para-computacion-cientifica-en-python/>

Cómo instalar el driver CH340 en diferentes sistemas operativos. (s.f.). Obtenido de HWLibre: https://www.hwlible.com/como-instalar-el-driver-ch340-en-diferentes-sistemas-operativos/#google_vignette

Dalmia, V. (24 de Septiembre de 2024). *Exploring Robotics with the Elegoo Smart Robot Car 4.0: A Beginner's Journey*. Obtenido de Vaidehi Dalmia: <https://vaidehidalmia.github.io/posts/elegoo/>

Duque, M. (s.f.). *Visión por Computadora (Computer Vision)*. Obtenido de manuduque: <https://www.manuduque.com/diccionario-inteligencia-artificial/vision-por-computadora/>

ESP32-S3 Series Datasheet. (s.f.). Obtenido de Espressif Systems: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

Heinzman, A. (11 de Octubre de 2023). *Raspberry Pi OS Gets the Debian 12 "Bookworm" Upgrade*. Obtenido de How-To Geek: <https://www.howtogeek.com/raspberry-pi-os-gets-the-debian-12-bookworm-upgrade>

How to Install CH340 Drivers. (s.f.). Obtenido de SparkFun: <https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all>

Inteligencia artificial y robótica: el binomio del futuro. (s.f.). Obtenido de Instituto Andaluz de Tecnología: <https://iat.es/tecnologias/inteligencia-artificial/robotica/>

Introducción al reconocimiento de patrones. (s.f.). Obtenido de MathWorks: <https://la.mathworks.com/discovery/pattern-recognition.html>

Jukes, R. (27 de Mayo de 2024). *Image Processing*. Obtenido de XPS: <https://www.xps.net/definition/image-processing/>

Kumar, A. (17 de Octubre de 2024). *What Is Image Processing : Overview, Applications, Benefits, and More*. Obtenido de Simplilearn: <https://www.simplilearn.com/image-processing-article>

Larabel, M. (11 de Octubre de 2023). *Raspberry Pi OS Now Based On Debian 12 "Bookworm" + Wayland*. Obtenido de Phoronix: <https://www.phoronix.com/news/Raspberry-Pi-OS-Bookworm>

Long, S. (11 de Octubre de 2023). *Bookworm — the new version of Raspberry Pi OS*. Obtenido de Raspberry Pi: <https://www.raspberrypi.com/news/bookworm-the-new-version-of-raspberry-pi-os/>

Michael, B. (Marzo de 2021). *Programming Elegoo Smart Robot*. Obtenido de You Tube: <https://www.youtube.com/watch?v=j7p44vxyvdM>

Moraguez, E. R. (s.f.). *¿Qué es NumPy: cómo funciona y para qué sirve?* Obtenido de LovTechnology: <https://lovtechnology.com/que-es-numpy-como-funciona-y-para-que-sirve/>

Nelson, D. (Julio de 21 de 2023). *¿Qué es la Visión por Computadora?* Obtenido de Unite.AI: <https://unite.ai/es/que-es-la-vision-por-computador/>

Pastor, J. (28 de Septiembre de 2023). *Llega la nueva Raspberry Pi 5 y lo hace presumiendo de un potente chip propio y más capacidad de expansión que nunca.* Obtenido de Xataka: <https://www.xataka.com/ordenadores/raspberry-pi-5-caracteristicas-precio-ficha-tecnica>

Python 3.11.0. (24 de Octubre de 2022). Obtenido de Python Software Foundation: <https://www.python.org/downloads/release/python-3110/>

Raspberry Pi 5. (s.f.). Obtenido de Raspberry Pi: <https://www.raspberrypi.com/products/raspberry-pi-5/>

Raspberry Pi 5 Technical Specifications and Mechanical Drawings. (28 de Septiembre de 2023). Obtenido de Element14 Community: <https://community.element14.com/products/raspberry-pi/b/blog/posts/raspberry-pi-5-technical-specifications-and-mechanical-drawings>

Raspberry Pi Hardware. (s.f.). Obtenido de Raspberry Pi: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>

Reconocimiento de patrones. (s.f.). Obtenido de Academia Lab: <https://academia-lab.com/enciclopedia/reconocimiento-de-patrones/>

Reconocimiento de patrones: qué es y cómo funciona. (s.f.). Obtenido de CEUPE - European Business School: <https://www.ceupe.com/blog/reconocimiento-de-patrones.html>

Rouse, M. (6 de Febrero de 2019). *Image Processing*. Obtenido de Techopedia: <https://www.techopedia.com/definition/2009/image-processing>

Rouse, M. (11 de Septiembre de 2024). *Robótica*. Obtenido de Techopedia: <https://www.techopedia.com/es/definicion/robotica>

Smart Robot Car Kit V4.0 (With Camera). (s.f.). Obtenido de ELEGOO: https://us.elegoo.com/products/elegoo-smart-robot-car-kit-v-4-0?utm_source=officiallisting&utm_medium=referral&utm_id=usstore

Susnjara, S., & Smalley, I. (6 de Junio de 2024). *¿Qué es Docker?* Obtenido de IBM: <https://www.ibm.com/mx-es/topics/docker>

Uno R3. (s.f.). Obtenido de Arduino: <https://www.arduino.cc/en/Main/ArduinoBoardUno>

Vargas, B. (2 de Mayo de 2024). *Análisis completo: Ventajas y desventajas de Arduino*. Obtenido de Byron Vargas: <https://www.byronvargas.com/web/cuales-son-las-ventajas-y-desventajas-de-arduino/>

Vidal, S. (21 de Septiembre de 2023). *What is robotics and how does it work?* Obtenido de Tecnobits: https://tecnobits.com/en/que-es-la-robotica-y-como-funciona/#google_vignette

Visión por computador: qué es, objetivos y aplicaciones. (31 de Enero de 2022). Obtenido de EDS Robotics: <https://www.edsrobotics.com/blog/vision-computador-que-es/>

What is Arduino IDE? (s.f.). Obtenido de Arduino: <https://docs.arduino.cc/software/ide/>

What is Docker? (s.f.). Obtenido de Docker Docs: <https://docs.docker.com/get-started/docker-overview/>

What is NumPy? (s.f.). Obtenido de NumPy:

<https://numpy.org/doc/stable/user/whatisnumpy.html>

What is OpenCV? – An Introduction Guide. (s.f.). Obtenido de Python Geeks:

<https://pythongeeks.org/what-is-opencv/>