UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE MECATRÓNICA

IMPLEMENTACIÓN DE UN ROBOT MÓVIL PARA EL CONTROL DE TRÁNSITO Y MOVILIDAD USANDO VISIÓN ARTIFICIAL

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTOR: Edgar Daniel Barba Martillo
TUTOR: Ing. Tomás Santiago Gavilánez Gamboa, MSc.

Guayaquil - Ecuador

2024

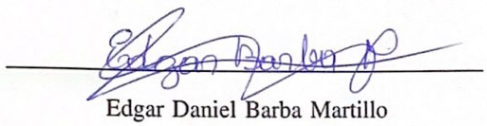
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, **Edgar Daniel Barba Martillo** con documento de identificación Nº **0959015496** manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 4 de septiembre del año 2024

Atentamente,



Edgar Daniel Barba Martillo
0959015496

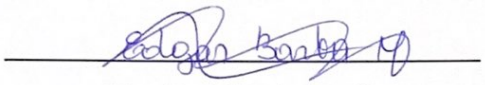
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, **Edgar Daniel Barba Martillo** con documento de identificación Nº **0959015496** , expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy el autor del **Dispositivo Tecnológico: IMPLEMENTACIÓN DE UN ROBOT MÓVIL PARA EL CONTROL DE TRÁNSITO Y MOVILIDAD USANDO VISIÓN ARTIFICIAL**, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 4 de septiembre del año 2024

Atentamente,



Edgar Daniel Barba Martillo
0959015496

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Tomás Santiago Gavilánez Gamboa**, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **IMPLEMENTACIÓN DE UN ROBOT MÓVIL PARA EL CONTROL DE TRÁNSITO Y MOVILIDAD USANDO VISIÓN ARTIFICIAL** , realizado por **Edgar Daniel Barba Martillo** con documento de identificación N° **0959015496** obteniendo como resultado final el trabajo de titulación bajo la opción **Dispositivo Tecnológico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 4 de septiembre del año 2024

Atentamente,

Ing. Tomás Santiago Gavilánez Gamboa, MSc.
1802792646

DEDICATORIA

El trabajo realizado va dedicado principalmente a mis padres, quienes durante todos estos años han sido mi pilar de fortaleza y mi apoyo incondicional. A mis hermanos mayores, por ser mis ejemplos y consejeros.

Finalmente agradezco a mi hermana Genesis Gabriela Barba Martillo por su confianza absoluta que me ha impulsado a seguir adelante.

Edgar Daniel Barba Martillo

AGRADECIMIENTO

Agradezco principalmente a Dios, a mi padre y madre por su amor incondicional y por inculcarme el valor de la perseverancia para afrontar los retos y dificultades de la vida.

Agradezco a todas esas amistades que me dio directamente como indirectamente la universidad, porque me han demostrado ser grandes seres humanos que siempre me han brindado de su apoyo aun cuando fue difícil avanzar, por acompañarme y hacer de este camino más agradable y enriquecedor..

Edgar Daniel Barba Martillo

RESUMEN

El proyecto de titulación tiene como objetivo general implementar un robot móvil para el control de tránsito y movilidad utilizando visión artificial. Se utilizará una Raspberry Pi como base del robot, junto con diversos materiales de robótica, para desarrollar un sistema de navegación radio controlada. Se diseñará un algoritmo avanzado de reconocimiento de vehículos y peatones mediante técnicas de visión artificial, permitiendo al robot identificar y clasificar objetos en su entorno. Además, se implementará una lógica difusa para la toma de decisiones, permitiendo al robot gestionar de manera autónoma el paso de vehículos y peatones, mejorando la fluidez y seguridad del tránsito. Con este enfoque, el proyecto pretende mejorar la funcionalidad, control y visualización del módulo interactivo, aprovechando una variedad de tecnologías para lograr una solución integral y eficiente en la gestión del tráfico urbano.

Palabras claves: RASPBERRY PI, ARTIFICIAL VISION, PYTHON, ROBOTICS.

ABSTRACT

The general objective of the degree project is to implement a mobile robot for traffic and mobility control using artificial vision. A Raspberry Pi will be used as the base of the robot, along with various robotics materials, to develop a radio controlled navigation system. An advanced vehicle and pedestrian recognition algorithm will be designed using computer vision techniques, allowing the robot to identify and classify objects in its environment. In addition, a fuzzy logic for decision making will be implemented, allowing the robot to autonomously manage the passage of vehicles and pedestrians, improving traffic flow and safety. With this approach, the project aims to improve the functionality, control and visualization of the interactive module, taking advantage of a variety of technologies to achieve a comprehensive and efficient solution for urban traffic management.

Keywords: RASPBERRY PI, ARTIFICIAL VISION, PYTHON, ROBOTICS.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. Introducción

En el ámbito de la gestión del tráfico y la movilidad urbana, la implementación de tecnologías avanzadas juega un papel crucial para mejorar la eficiencia y seguridad en las vías. Este proyecto se centra en la creación de un robot móvil controlado por radio que utiliza visión artificial para la supervisión y gestión del tránsito. Empleando un sistema basado en Raspberry Pi, el robot se integrará con diversos componentes electrónicos como sensores ultrasónicos, cámaras, servomotores y motores. La plataforma Raspberry Pi será el núcleo de procesamiento, ejecutando algoritmos de visión artificial desarrollados en Python y utilizando la biblioteca ViLib, TensorFlow para el reconocimiento de vehículos y peatones.

El objetivo principal es diseñar y construir un robot móvil capaz de navegar de manera semi asistida en entornos de tráfico, utilizando la visión artificial para identificar y clasificar objetos en su entorno. Este sistema no solo permitirá la detección precisa de peatones y señales de tránsito como señal PARE, sino que también incorporará lógica difusa para la toma de decisiones en tiempo real, facilitando el paso seguro de estos elementos según las condiciones detectadas.

Los objetivos específicos incluyen la implementación de sistemas mecatrónicos que permitan la navegación controlada del robot, el desarrollo de algoritmos robustos para el reconocimiento de objetos, y la integración de un sistema de toma de decisiones basado en lógica difusa. Con este enfoque, se espera crear una solución innovadora y efectiva para la gestión del tráfico, contribuyendo a un entorno urbano más seguro y eficiente.

Este proyecto combina la robótica, la visión y la inteligencia artificiales para abordar los desafíos contemporáneos de la movilidad urbana, demostrando el potencial de las tecnologías emergentes en la optimización de la gestión del tráfico.

II. Problema

La infraestructura del sistema vial constituye uno de los activos más significativos para cualquier nación, siendo su magnitud y calidad indicadores cruciales del nivel de desarrollo alcanzado por el país. A nivel global, los desafíos relacionados con la congestión vehicular han experimentado un aumento, producto del incremento en el crecimiento demográfico, la urbanización y las variaciones en la densidad de población. Esta congestión reduce la eficiencia de la infraestructura de transporte incrementando los tiempos de viaje, el consumo de combustible y la contaminación ambiental [1]. En la actualidad uno de los mayores problemas de los países latinoamericanos es el alto índice de siniestralidad por accidentes de tránsito, lo que ha afectado negativamente la calidad de vida de muchas personas [2].

En Ecuador se considera al vehículo liviano como el principal elemento de congestión dentro del tránsito vehicular, con una representación promedio del 64.83 %, seguido de las motocicletas (24.54 %), bicicletas (6.40 %), camiones (3.76 %) y buses (0.47 %) [3]. El control del tráfico se realiza mediante señales de tránsito complementariamente con agentes civiles de tránsito.

En el estudio realizado por Quiñónez y Mayer [4] se determinaron los riesgos laborales asociados con dirigir el tránsito peatonal y vehicular son: altas temperaturas, exposición a vapores y gases contaminantes, ruido, polvo y actividades repetitivas. Además, los funcionarios civiles encargados del control del tránsito se enfrentan diariamente a la posibilidad de sufrir diversas afecciones físicas, tales como golpes, caídas y atropellamientos.

Como resultado de estas circunstancias laborales, las enfermedades más comúnmente asociadas con la actividad de los agentes civiles de tránsito incluyen dolencias musculoesqueléticas, problemas lumbares, tendinitis y elevados niveles de estrés. A pesar de contar con una extensa red vial de 4,384 kilómetros en Guayaquil [5], surge una problemática significativa en la gestión del tráfico. La presencia de aproximadamente 900 agentes de tránsito [6] en las avenidas de la ciudad se enfrenta a desafíos para garantizar un flujo vehicular eficiente y 2 seguro.

Esta situación plantea interrogantes sobre la capacidad del sistema actual para mantener la ordenada circulación en la extensa red de carreteras, resaltando la necesidad de abordar posibles obstáculos en la movilidad urbana. A nivel mundial, diversas alternativas se han explorado para abordar la gestión del tráfico como las propuestas por Reyes & Sanchez [7] y Candia, Kopacz & Raggio [8].

La implementación de robots móviles para control de tráfico es un nuevo campo de estudio que promete ser una alternativa viable para afrontar esta problemática. Sin embargo, su desarollo implica poseer conocimientos en diversas áreas incluyendo sofisticados algoritmos para navegación y reconocimiento de patrones, los cuales resultan en un desafío para la ingeniería. En Ecuador, la congestión vehicular, especialmente causada por vehículos livianos, ha llevado a la exploración de varias alternativas, no obstante, no se abordan explícitamente la implementación exitosa de estas soluciones ni de los posibles desafíos que podrían surgir en el proceso.

El Proyecto de Titulación se desarrolló en la Universidad Politécnica Salesiana sede Guayaquil, Ecuador en el período 2024-2024.

III. JUSTIFICACIÓN

El flujo vehicular en una ubicación específica se genera a partir de una serie de decisiones, tanto para aquellos encargados del diseño de la infraestructura vial como para quienes gestionan el control y la optimización del tráfico [9].
La presencia de un Robot Móvil para control de Tránsito permitirá una supervisión continua y eficiente, reduciendo el riesgo de accidentes y mejorando la seguridad tanto para peatones como para conductores, desarrollar un prototipo permitirá sentar las bases para la escalabilidad de la solución, adaptándose a necesidades específicas y proporcionando una solución versátil y personalizable.

La visión artificial, según Sánchez [10], ha demostrado ser una herramienta poderosa y valiosa para detectar conflictos entre vehículos y peatones. La integración de tecnologías como la visión por computadora, el aprendizaje automático y la comunicación inalámbrica no solo respalda esta capacidad, sino que también impulsa la creación de un entorno moderno y a la vanguardia. El proyecto, al centrarse en la implementación de un Robot Móvil para el control de Tránsito y Movilidad, ofreciendo una valiosa oportunidad para la investigación y desarrollo en el ámbito de la mecatrónica y la robótica.

En particular, la incorporación de avanzadas tecnologías de detección no solo mejora la seguridad vial mediante la identificación precisa de vehículos, sino que también optimiza la interacción con peatones al detectar de manera efectiva la presencia de personas. Este enfoque demuestra un claro compromiso con la innovación y el progreso tecnológico en el ámbito de la detección y la movilidad urbana.

Por lo tanto la implementación de un robot móvil equipado con tecnología de visión artificial y control difuso, basado en una plataforma Raspberry Pi, permitirá mejorar significativamente la gestión del tráfico y la movilidad urbana. Este sistema será capaz de detectar con precisión vehículos y peatones, tomando decisiones semi asistidas en tiempo real que optimicen el flujo del tránsito y aumenten la seguridad vial.

IV. OBJETIVOS

IV-A. Objetivo general

Implementar un robot móvil para el control de tránsito y movilidad usando visión artificial.

IV-B. Objetivos específicos

- Implementar los sistemas mecatrónicos para la navegación radio controlada del robot móvil.
- Desarrollar un algoritmo de reconocimiento de vehículos y peatones utilizando visión artificial.
- Permitir el paso de vehículos o peatones implementando lógica difusa para la toma de decisiones.

V. Fundamentos Teóricos

V-A. Tránsito y Movilidad Urbana

Según la OPS (Organización Panamericana de Salud) [11], tránsito se refiere al movimiento de vehículos y peatones en las vías públicas. Incluye la regulación del tráfico, las señales de tránsito, y la infraestructura vial necesaria para facilitar el flujo de vehículos y personas. Por su parte la movilidad es un concepto más amplio que abarca no solo el tránsito de vehículos, sino también la capacidad de las personas para desplazarse de un lugar a otro de manera eficiente y segura. Esto incluye el uso de diferentes modos de transporte, como caminar, andar en bicicleta, usar transporte público, y conducir vehículos privados [12]. En conclusión la movilidad eficiente es crucial para la productividad económica, la calidad de vida y la sostenibilidad ambiental de las ciudades.

El tránsito en zonas urbanas está influenciada por una variedad de factores que impactan tanto al tráfico como a la seguridad vial. Entre los más importantes se encuentran el comportamiento de conductores y peatones, incluyendo el uso de teléfonos móviles y distracciones [13]; la calidad y el diseño de la infraestructura vial, como la señalización, los semáforos y las aceras [14]; el estado técnico de los vehículos, como los frenos, luces y neumáticos, que tienen un impacto directo en la seguridad vial; las condiciones ambientales, como la lluvia, la niebla y la iluminación, que afectan la visibilidad y la adherencia en la carretera [15]; y, finalmente, los factores sociales y económicos, como la densidad de población, la expansión urbana y las políticas de transporte público, que también influyen en el comportamiento del tránsito y movilidad en una ciudad.

V-A1. Incovenientes Relacionados al Tránsito: Uno de los principales problemas son los accidentes de tránsito que es una de las causas de mortalidad y discapacidad mas grande a nivel global. Se ha observado que los factores humanos, como el exceso de velocidad y la distracción al volante, son los principales contribuyentes a estos incidentes [16], tambien factores como la conducción bajo los efectos del alcohol, y la falta de mantenimiento adecuado de los vehículos [17]. Además, la infraestructura vial inadecuada y las condiciones meteorológicas adversas pueden aumentar significativamente el riesgo de accidentes.Las consecuencias de los accidentes de tránsito son devastadoras, tanto en términos de pérdida de vidas como de daños materiales ya que representan un alto costo para los sistemas de salud y generan importantes pérdidas económicas debido a la incapacidad laboral y los costos de reparación [18].

Otro problema es la congestión vehicular que es un fenómeno omnipresente en las ciudades modernas, derivado de la demanda excesiva sobre la capacidad de la infraestructura vial. Este problema afecta la red vial de un país y representa un desafío significativo para la población que debe circular por ella [19]. A su vez, genera una serie de problemas sociales y económicos. El trádico vehicular no solo afecta el tiempo de viaje de los conductores, sino que también incrementa las emisiones de gases contaminantes, lo que exacerba los problemas de salud pública y el cambio climático [20] los cuales han sido ampliamente documentados en la literatura científica.

Figura 1. Análisis del congestionamiento vehicular [19].

Las causas de la congestión vehicular son multifactoriales y suelen incluir la alta concentración de vehículos en áreas específicas, la insuficiencia de planificación urbana, y el predominio del uso de vehículos privados sobre el transporte público. Además, la falta de inversión en infraestructura y la expansión urbana descontrolada contribuyen significativamente a este problema [21].

El tráfico vehicular es una de las principales fuentes de contaminación del aire en las áreas urbanas. Los vehículos emiten una variedad de contaminantes, incluyendo dióxido de carbono (CO2) y óxidos de nitrógeno (NOx), que son perjudiciales para la salud humana y el medio ambiente [22]. Desde una perspectiva económica, el tráfico vehicular genera pérdidas significativas debido al tiempo improductivo y al aumento de los costos de transporte [23].

La principal causa de la contaminación vehicular es el uso de combustibles fósiles en los vehículos, lo que resulta en la emisión de gases de efecto invernadero y otros contaminantes atmosféricos [22]. La congestión vehicular también exacerba este problema, al aumentar el consumo de combustible y, por lo tanto, las emisiones[24]. La exposición prolongada a la contaminación del aire derivada del tráfico puede causar una serie de problemas de salud, incluyendo enfermedades respiratorias y cardiovasculares, así como contribuir al cambio climático [25]. Estos impactos son particularmente graves en áreas urbanas densamente pobladas.

V-A2. Soluciones Tecnológicas aplicadas al Tránsito: En los últimos años los avances en la robótica para tránsito ha permitido integrar tecnologías como inteligencia artificial (IA), la visión por computadora y los sistemas autónomos un ejemplo de esto serían los vehículos semiasistidos, que son automóviles equipados con sistemas avanzados de asistencia al conductor que automatizan ciertas funciones de conducción, mejorando la seguridad y la comodidad del conductor. Estos sistemas incluyen tecnologías como el control de crucero adaptativo, el asistente de mantenimiento de carril, la frenada automática de emergencia y la detección de puntos ciegos. A diferencia de los vehículos completamente autónomos, los vehículos semiasistidos requieren la supervisión activa del conductor y su intervención en situaciones complejas o de emergencia [26].

La robótica y visión artificial en la gestión del tráfico representa una solución innovadora a los problemas contemporáneos de congestión, seguridad y contaminación. A su vez implementar estas tecnologías junto con la lógica difusa, permite la toma de decisiones en tiempo real para mejorar el flujo vehicular y garantizar la seguridad de los peatones.

V-B. Robots Móviles para el Control de Tránsito

El avance de la tecnología en el ámbito del tránsito y la movilidad ha permitido la creación de sistemas automatizados que mejoran la eficiencia y seguridad en las vías urbanas, como por ejemplo el uso de robóts autónomos y otros sistemas que permitan el control del tránsito.

V-B1. Tipos de Robots Móviles para el Control de Tránsito: Los robots móviles para el control de tránsito están revolucionando la gestión del tráfico urbano. Estos dispositivos autónomos, equipados con sensores avanzados y sistemas de inteligencia artificial, pueden monitorear y dirigir el flujo vehicular de manera eficiente por ejemplo tenemos: Los vehículos semiasistidos que su propósito principal de es reducir la carga de trabajo del conductor y minimizar el riesgo de accidentes causados por errores humanos. Los sensores y las cámaras integrados en estos vehículos recopilan datos en tiempo real del entorno, permitiendo que el sistema tome decisiones informadas sobre aceleración, frenado y dirección. Esto no solo mejora la experiencia de conducción, sino que también contribuye a una mayor eficiencia energética y una reducción de las emisiones [27] como por ejemplo está la Figura 2.

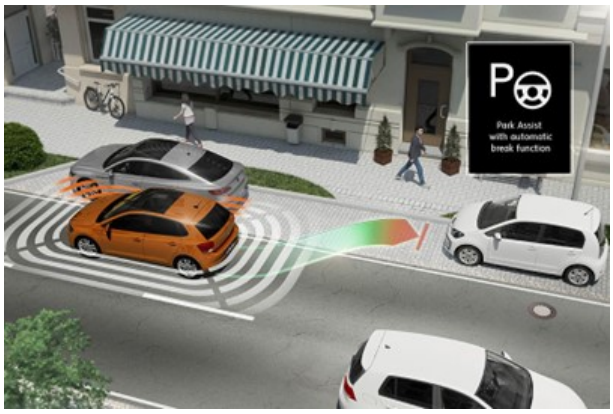


Figura 2. Vehículos Semiasistidos [28].

La evolución de los vehículos semiasistidos está marcando el camino hacia la plena automatización, con investigaciones y desarrollos continuos en inteligencia artificial y aprendizaje automático que buscan perfeccionar estas tecnologías. No obstante, la implementación masiva de vehículos semiasistidos plantea desafíos regulatorios y éticos, especialmente en términos de responsabilidad en caso de fallos del sistema y la privacidad de los datos recopilados [29].

Otro caso de la implementación de la robótica para tránsito son los Sistemas Inteligentes de Transporte o ITS (Intelligent Transportation Systems) ya que estos comprenden aplicaciones informáticas y procedimientos tecnológicos establecidos en conjunto para mejorar los niveles de seguridad y eficiencia en la organización del transporte terrestre, facilitando así su gestión, control y seguimiento por parte de los organismos competentes haciendo uso de los elementos tecnológicos que se muestran en la Figura 3.

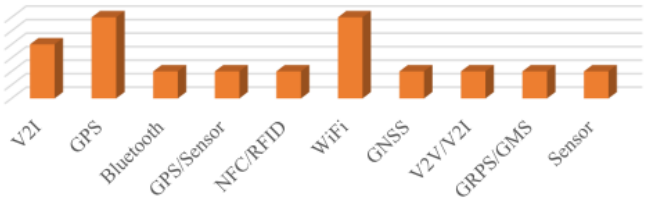


Figura 3. Tecnologias implementadas en los ITS [30].

Los robots policías del congo que se encuentran en las principales calles de Kinshasa, estos robots, de 2,5 metros de altura y 250 kilos de peso, tienen cuatro cámaras digitales de alta definición ubicadas en los ojos y los hombros. Estas cámaras registran toda la actividad en la carretera y transmiten el video en tiempo real a un centro de control policial, también cuentan con luces de señalización verde y roja en cada mano, lo que les permite regular el tráfico levantando las manos, y un pecho rotatorio que graba continuamente la actividad vial. También poseen un sistema integrado de grabación y transmisión de audio, lo que les permite emitir mensajes y sonidos mediante una voz sintética. Por ejemplo, pueden decir cosas como 'conductores, permitan el paso a los peatones'. Dado que la electricidad es un problema en el país, estos robots funcionan con energía solar, cuentan con una fuente de alimentación autónoma y están diseñados para resistir el clima cálido durante todo el año [31] como se ve en la Figura 4.

Figura 4. Robot Policía del Congo [31].

Como resultado de estos ejemplos mencionados se puede decir que la implementación de tecnólogias para control de tránsito como la visión artificial permite a los robots interpretar el entorno vial y tomar decisiones basadas en el reconocimiento de señales de tránsito, detección de peatones y vehículos, y otras variables críticas para la seguridad vial [32].

V-B2. Componentes de un robot Móvil: Los robots móviles son sistemas complejos que integran diversos componentes para realizar tareas autónomas o semiautónomas, entre los componentes principales se encuentra la plataforma móvil que es la estructura base del robot sobre la cual se montan todos los demás componentes, y puede tener ruedas, orugas o patas, dependiendo del diseño y el entorno en el que operará. Los actuadores, que incluyen motores eléctricos, servomotores y actuadores lineales, permiten el movimiento del robot al proporcionar la fuerza necesaria para mover las ruedas, las articulaciones o cualquier otro mecanismo móvil. Los sensores, como ultrasonidos, cámaras, sensores de infrarrojos, LIDAR e IMU, proveen al robot de la capacidad para percibir su entorno, midiendo distancias, realizando visión artificial, detectando proximidad, mapeando en 3D y midiendo aceleración y orientación [33].

Los controladores, que pueden ser microcontroladores como Arduino o computadoras más potentes como Raspberry Pi, interpretan los datos de los sensores y envían comandos a los actuadores. El sistema de navegación, que puede incluir GPS, sistemas de odometría y algoritmos de navegación como SLAM, permite al robot determinar su posición y planificar rutas. La fuente de energía, comúnmente baterías recargables de iones de litio o polímero de litio (LiPo), suministra la energía necesaria para el funcionamiento del robot [34]. La comunicación, que puede incluir módulos inalámbricos como Wi-Fi, Bluetooth o radios de frecuencia, permite la interacción del robot con otros dispositivos o sistemas. El chasis, generalmente fabricado en materiales como aluminio, plástico o fibra de carbono, es la estructura física que soporta y protege los componentes del robot, siendo robusta y ligera. Además, el software de control implementa los algoritmos necesarios para el funcionamiento del robot, incluyendo navegación, control de motores, procesamiento de datos de sensores y lógica de toma de decisiones. Finalmente, las interfaces de usuario permiten a los usuarios interactuar con el robot a través de pantallas, botones y aplicaciones móviles o de escritorio que facilitan el monitoreo y control remoto del robot [35].

Figura 5. Robot Móvil, por E. Barba

V-B3. Raspberry Pi: Características Técnicas y Aplicaciones: La Raspberry Pi es una serie de computadoras de placa única (SBC, por sus siglas en inglés) desarrollada por la Fundación Raspberry Pi en el Reino Unido. Originalmente diseñada para promover la enseñanza de ciencias de la computación básicas en las escuelas y países en desarrollo, la Raspberry Pi ha evolucionado para ser una herramienta versátil utilizada en una variedad de proyectos y aplicaciones tanto educativas como profesionales [36].

La mayoría de las Raspberry Pi utilizan procesadores ARM; por ejemplo, la Raspberry Pi 4 está equipada con un procesador Quad-Core ARM Cortex-A72 y ofrece opciones de memoria RAM que van desde 2 GB hasta 8 GB. En términos de conectividad, los modelos más avanzados, como la Raspberry Pi 4, incluyen puertos USB 3.0, Ethernet Gigabit, Wi-Fi, Bluetooth y múltiples puertos HDMI. El almacenamiento principal se realiza a través de tarjetas microSD, aunque los modelos recientes también permiten el arranque desde dispositivos USB. El sistema operativo oficial es Raspberry Pi OS (anteriormente conocido como Raspbian), una distribución basada en Debian optimizada para el hardware de Raspberry Pi, aunque también es compatible con otros sistemas operativos como Ubuntu, Windows 10 IoT Core y diversas distribuciones de Linux.

La Raspberry Pi ha revolucionado el mundo de la computación personal y embebida desde su lanzamiento, proporcionando un microordenador potente y accesible que ha inspirado a una amplia comunidad de usuarios. En el ámbito de los proyectos DIY (Do It Yourself), los makers y entusiastas encuentran en la Raspberry Pi una plataforma ideal para explorar la tecnología y desarrollar proyectos creativos. Estos proyectos pueden ir desde la creación de servidores domésticos para almacenar y gestionar datos, la implementación de sistemas de automatización que controlan luces y dispositivos en el hogar, hasta la construcción de estaciones meteorológicas que monitorean las condiciones climáticas en tiempo real. Además, la Raspberry Pi es una opción popular para revivir consolas de juegos retro, permitiendo a los usuarios jugar títulos clásicos en un formato compacto y moderno.

En aplicaciones industriales y comerciales, la Raspberry Pi no solo es una herramienta para aficionados, sino también un componente clave en soluciones de Internet de las Cosas (IoT), donde se utiliza para recopilar, procesar y transmitir datos en tiempo real en entornos variados, desde fábricas inteligentes hasta ciudades conectadas. Su versatilidad la convierte en una opción excelente para sistemas embebidos, donde la necesidad de un hardware robusto y personalizable es crucial. Además, su accesibilidad y capacidad de ser programada fácilmente la hacen ideal para dispositivos de prueba y desarrollo, permitiendo a los desarrolladores crear y testar nuevas ideas y prototipos con rapidez.

En el ámbito multimedia, la capacidad de la Raspberry Pi para reproducir video en alta definición ha llevado

a su adopción en sistemas de entretenimiento en el hogar. Con software como Kodi y Plex, los usuarios pueden transformar una Raspberry Pi en un completo centro multimedia, gestionando y reproduciendo películas, series, música y más, todo desde un dispositivo del tamaño de una tarjeta de crédito.

Esta combinación de accesibilidad, versatilidad y potencia ha consolidado a la Raspberry Pi como una herramienta indispensable en una amplia variedad de aplicaciones, desde la experimentación y el aprendizaje personal hasta soluciones profesionales en el mundo de la tecnología.

Figura 6. Raspberry Pi [37].

V-B4. Robot Hat SunFounder: La placa de expansión Robot HAT permite convertir una Raspberry Pi en un controlador de Robot, dentro de la placa existen circuitos integrados que permiten ampliar las entradas y salidas de una Raspberry Pi para el control de un robot, estas son: Salidas PWM, entradas ADC, Controlador de Motor, modulo Bluetooth, módulo de audio I2S, interfaz para Servomotore, y altavoz mono, además de otras entradas y salidas configurables en programación.

Figura 7. Robot Hat SunFounder [38].

V-B5. Software de programación Python.: Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Es conocido por su legibilidad y simplicidad, lo que lo hace accesible tanto para principiantes como para programadores experimentados. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python soporta múltiples paradigmas de programación, incluidos el orientado a objetos, el imperativo, el funcional y el procedimental.

La historia de Python comienza a finales de la década de 1980, cuando van Rossum comenzó a desarrollar lo que se convertiría en Python como un sucesor del lenguaje ABC. Python 1.0 fue lanzado en enero de 1994, introduciendo características fundamentales como el sistema de módulos. En octubre de 2000, se lanzó Python 2.0, que trajo mejoras significativas como un recolector de basura y soporte completo para Unicode. La serie 2.x continuó evolucionando hasta la versión 2.7, que fue la última de la serie y se mantuvo hasta 2020. Posteriormente,

Python 3.0 fue lanzado en diciembre de 2008 con el objetivo de corregir defectos fundamentales en el diseño del lenguaje y mejorar su consistencia. A diferencia de las versiones anteriores, Python 3.x no es completamente compatible hacia atrás con Python 2.x, lo que ha provocado una transición gradual en la comunidad.

Python se destaca por su simplicidad y legibilidad, con una sintaxis sencilla y fácil de entender que promueve la escritura de código limpio. Utiliza la indentación para definir bloques de código, lo que mejora la claridad y elimina la necesidad de llaves o palabras clave adicionales. Además, Python es un lenguaje de tipado dinámico, lo que significa que las variables no necesitan ser declaradas explícitamente con un tipo de dato y pueden cambiar de tipo en tiempo de ejecución. Como lenguaje interpretado, Python ejecuta el código línea por línea sin necesidad de compilación previa.

Otra característica clave de Python es su amplia biblioteca estándar, que proporciona módulos y paquetes para manejar una gran variedad de tareas, desde operaciones matemáticas hasta manipulación de archivos y redes. Python también es multiparadigma, soportando estilos de programación orientados a objetos, funcionales y procedimentales, lo que proporciona gran flexibilidad a los desarrolladores, por su portabilidad permite que el código Python se ejecute en múltiples plataformas sin cambios, incluyendo Windows, macOS, Linux, y otros sistemas operativos. Además, cuenta con una gran biblioteca estándar que ahorra tiempo y esfuerzo al proporcionar soluciones listas para usar. La comunidad activa de Python ofrece un amplio soporte, con abundante documentación y recursos disponibles para los desarrolladores, la principal desventaja de Python radica en que al ser un lenguaje interpretado, puede tener un rendimiento más lento en comparación con lenguajes compilados como C++ o Java. Además, la gestión automática de memoria en Python puede ser menos eficiente en ciertos contextos, lo que podría afectar el rendimiento en aplicaciones que requieren un uso intensivo de recursos.

Figura 8. Software de programación Python [39].

V-C. Lógica Difusa.

La lógica difusa es una extensión de la lógica clásica que ofrece una poderosa herramienta para manejar la incertidumbre y la imprecisión en la toma de decisiones y el análisis de sistemas complejos. A diferencia de la lógica binaria tradicional, que se basa en valores absolutos de verdadero (1) o falso (0), la lógica difusa permite un espectro continuo de valores entre 0 y 1. Este enfoque refleja de manera más precisa cómo los seres humanos perciben y razonan sobre el mundo, donde muchas situaciones no son completamente verdaderas ni completamente falsas, sino que caen en algún punto intermedio. Este enfoque más matizado es fundamental en la modelación de problemas en los que la ambigüedad y la imprecisión son intrínsecas, y ha encontrado aplicaciones en una amplia gama de campos que requieren decisiones basadas en información incompleta o vaga.

V-C1. Origen y evolución de la lógica difusa: La lógica difusa fue introducida por el matemático y profesor Lotfi A. Zadeh en 1965 a través de su trabajo seminal titulado Fuzzy Sets [40]. Zadeh propuso la teoría de conjuntos difusos como un marco para representar datos que no son precisos o que contienen algún grado de incertidumbre.

Esta innovación permitió abordar problemas que hasta entonces habían sido difíciles de tratar con los métodos de lógica tradicional. Desde su introducción, la lógica difusa ha sido desarrollada y aplicada en numerosos campos, demostrando su utilidad en áreas tan diversas como la ingeniería, la economía, la medicina y la inteligencia artificial. Su capacidad para modelar sistemas complejos y tomar decisiones en ambientes inciertos ha hecho de la lógica difusa una herramienta indispensable en muchos contextos donde la precisión absoluta no es alcanzable o necesaria.

V-C2. Fundamentos Teóricos de la lógica difusa:

- Conjuntos Difusos: Un conjunto difuso es una extensión de un conjunto clásico donde los elementos tienen grados de pertenencia que varían entre 0 y 1. Este grado de pertenencia indica en qué medida un elemento pertenece al conjunto.

- Funciones de Pertenencia: Una función de pertenencia asigna a cada elemento un valor entre 0 y 1. Estas funciones pueden tomar diversas formas, como trapezoidales, triangulares, gaussianas, entre otras.

- Unión Difusa: Similar a la unión en conjuntos clásicos, pero con la fórmula 1 que se muestra a continuación:

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{1}$$

- Intersección Difusa: Similar a la intersección en conjuntos clásicos, pero con la fórmula 2 que se muestra a continuación:

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{2}$$

- Complemento Difuso: Definido como la formula 3 que se muestra a continuación:

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \tag{3}$$

- Reglas Difusas: Las reglas difusas son expresiones de la forma ”Si A, entonces B”, donde A y B son expresiones que involucran conjuntos difusos. Estas reglas permiten modelar el razonamiento humano y son la base de los sistemas de inferencia difusa.

- Inferencia Difusa: El proceso de inferencia difusa implica aplicar reglas difusas para determinar una salida difusa a partir de entradas difusas. Se utilizan métodos como Mamdani, Sugeno y Tsukamoto para este propósito.

- Defuzzificación: La defuzzificación es el proceso de convertir una salida difusa en un valor crisp (preciso). Métodos comunes incluyen el centroide, el máximo de la media y el promedio ponderado.

V-C3. Aplicaciones Prácticas: La lógica difusa tiene múltiples aplicaciones prácticas en diversos campos. En control de sistemas, los controladores difusos se utilizan ampliamente en situaciones donde los modelos matemáticos precisos son difíciles de obtener, como en el control de temperatura, la regulación de motores y el control de procesos industriales. En toma de decisiones, la lógica difusa se aplica en sistemas de apoyo donde la evaluación de alternativas puede ser incierta o imprecisa, siendo útil en áreas como la planificación financiera, la gestión de riesgos y la selección de inversiones. También se emplea en procesamiento de imágenes, utilizando técnicas difusas para mejorar imágenes y manejar transiciones suaves e interpretaciones humanas. En inteligencia artificial y robótica,

la lógica difusa se integra para mejorar la toma de decisiones y el comportamiento adaptativo en robots y agentes autónomos. Finalmente, en medicina, se utiliza para el diagnóstico y tratamiento de enfermedades, permitiendo a los médicos manejar la incertidumbre en los síntomas y en las respuestas al tratamiento. La lógica difusa es particularmente útil en la toma de decisiones en situaciones donde las variables son inciertas o imprecisas. Por ejemplo, un robot semiasistido puede utilizar lógica difusa para determinar la velocidad adecuada al detectar un peatón cerca de un cruce [40].

V-C4. Ventajas y Desventajas: La lógica difusa ofrece varias ventajas significativas, como su capacidad para manejar la incertidumbre y la imprecisión, algo que es crucial en muchas aplicaciones del mundo real. Además, su capacidad para modelar el razonamiento humano de manera intuitiva la convierte en una herramienta poderosa para sistemas que requieren tomar decisiones en situaciones complejas. La flexibilidad de la lógica difusa en su implementación y adaptación a diferentes aplicaciones es otro punto a su favor, permitiendo que sea utilizada en una amplia gama de contextos.

Sin embargo, también tiene algunas desventajas como la definición de funciones de pertenencia y reglas difusas que puede ser compleja y requiere un profundo conocimiento del sistema en cuestión. Además, la necesidad de técnicas de defuzzificación para obtener resultados precisos añade un nivel adicional de complejidad al proceso. Por último, la lógica difusa a menudo requiere una considerable experimentación y ajuste para lograr un rendimiento óptimo, lo que puede ser un desafío en proyectos con limitaciones de tiempo o recursos.

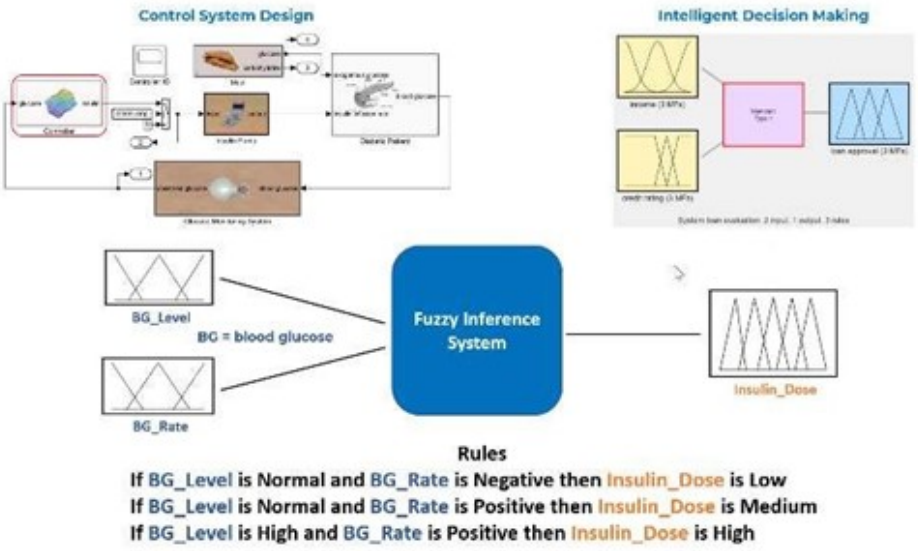


Figura 9. Fuzzy Logic [41].

V-D. Visión Artificial y OpenCV

La visión artificial, conforme a los criterios establecidos por la Asociación de Imágenes Automatizadas (AIA), abarca tanto aplicaciones industriales como no industriales, donde una combinación de hardware y software dirige diversos dispositivos mediante la captura y procesamiento de imágenes para ejecutar funciones específicas. Aunque la visión artificial industrial comparte muchos algoritmos y enfoques con aplicaciones académicas, educativas, gubernamentales y militares, existen diferencias significativas en las limitaciones y requisitos. La visión artificial (o visión por computadora) es un campo de la inteligencia artificial que permite a las computadoras y sistemas comprender e interpretar el contenido visual del mundo, de manera similar a como lo haría un ser humano. Esta

tecnología es fundamental para el desarrollo de sistemas inteligentes que requieren la interpretación de imágenes y videos para tomar decisiones o realizar acciones automáticas [42].

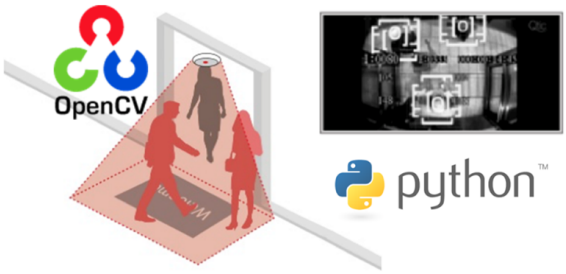


Figura 10. Vision Artificial [43].

V-D1. Historia y Evolución de OpenCV: OpenCV (Open Source Computer Vision Library) es una biblioteca de software libre que ha sido crucial para el desarrollo de aplicaciones de visión por computadora. El proyecto fue iniciado por Intel en 1999 y se lanzó al público en 2000 con el objetivo de acelerar el uso de la percepción por parte de aplicaciones comerciales y proporcionar una infraestructura común para las aplicaciones de visión por computadora. Desde su lanzamiento, OpenCV ha experimentado un crecimiento significativo, impulsado por una comunidad activa de desarrolladores y usuarios. La biblioteca ha sido optimizada para funcionar en una amplia variedad de plataformas, incluyendo Windows, Linux, macOS, iOS y Android, lo que ha facilitado su adopción en diversos entornos y dispositivos.

Las actualizaciones de OpenCV han introducido importantes mejoras y nuevas funcionalidades a lo largo del tiempo. La versión 2.x incluyó avances como el reconocimiento de patrones, mientras que las versiones 3.x y posteriores han mejorado el soporte para el aprendizaje automático y las redes neuronales profundas. Estas capacidades han expandido las aplicaciones de OpenCV, permitiendo a los desarrolladores implementar soluciones más complejas y eficientes en áreas que van desde la automatización industrial hasta la inteligencia artificial avanzada.

V-D2. Fundamentos de la Visión Artificial: La visión artificial se basa en una serie de conceptos fundamentales que permiten a los sistemas interpretar y procesar información visual. El primer paso es la captura de imágenes, que se realiza utilizando cámaras y sensores para obtener datos visuales en formato digital. Una vez capturadas, las imágenes suelen someterse a un preprocesamiento que incluye la conversión de color, reducción de ruido y normalización, lo que facilita el análisis posterior. La segmentación es otro paso crítico, donde la imagen se divide en partes significativas, como la detección de bordes o regiones de interés, para su análisis detallado.

El reconocimiento de patrones es una función central en la visión artificial, permitiendo la identificación de objetos, formas o características específicas dentro de una imagen. Esta capacidad es esencial para aplicaciones como la detección de rostros o la identificación de productos en una línea de ensamblaje. Además, el seguimiento de objetos a lo largo de una secuencia de imágenes permite monitorear el movimiento, lo que es crucial en aplicaciones de seguridad y vigilancia, así como en el análisis del comportamiento en videos.

V-D3. Funcionalidades de OpenCV: OpenCV ofrece un conjunto robusto de módulos que permiten el desarrollo de aplicaciones avanzadas de visión por computadora. El módulo Core proporciona funciones básicas para operaciones aritméticas, álgebra lineal y manipulación de matrices, que son fundamentales para casi todas las aplicaciones. Ïmgprocïncluye funciones específicas para el procesamiento de imágenes, como filtrado, transformaciones geométricas y segmentación, facilitando la manipulación y análisis de datos visuales. El módulo ”Video.ºfrece herramientas para la captura, procesamiento y análisis de video, mientras que Objdetect proporciona funciones especializadas para la detección de objetos como rostros y ojos.

Para la visualización de imágenes y videos, OpenCV incluye Highgui, una interfaz gráfica que simplifica la interacción con los datos visuales. Además, el módulo ML ofrece herramientas de aprendizaje automático que permiten entrenar y aplicar modelos predictivos en tareas de visión artificial. La biblioteca también se complementa con OpenCV contrib, un conjunto de módulos adicionales mantenidos por la comunidad, que introducen funcionalidades avanzadas y experimentales.

V-D4. Aplicaciones Prácticas: La visión artificial tiene un amplio rango de aplicaciones prácticas que abarcan múltiples industrias. En el ámbito de la seguridad y vigilancia, los sistemas inteligentes utilizan la detección de movimiento y el reconocimiento facial para monitorear áreas sensibles, identificando posibles amenazas o actividades inusuales. En el sector automotriz, la visión por computadora es fundamental para el desarrollo de vehículos autónomos, permitiendo la detección de peatones, el reconocimiento de señales de tráfico y la navegación segura a través de entornos complejos.

En el campo de la salud, la visión artificial se utiliza en el análisis de imágenes médicas para la detección temprana de enfermedades y condiciones médicas, como el análisis de radiografías y resonancias magnéticas. Esto mejora la precisión del diagnóstico y permite intervenciones más rápidas. En robótica, la visión artificial es clave para la navegación autónoma de robots, la manipulación de objetos y la ejecución de tareas de ensamblaje en entornos industriales. Estas aplicaciones demuestran cómo la visión artificial no solo amplía las capacidades de los sistemas tecnológicos, sino que también transforma industrias enteras, mejorando la eficiencia y seguridad en una variedad de contextos.

Figura 11. OpenCV [43].

V-E. MIT AppInventor

MIT App Inventor es una plataforma de desarrollo de aplicaciones móviles creada por el Massachusetts Institute of Technology (MIT). Esta herramienta está diseñada para ser accesible a personas sin experiencia en programación, permitiéndoles diseñar, desarrollar y publicar aplicaciones para dispositivos Android de manera sencilla e intuitiva. Utiliza una interfaz gráfica de usuario que emplea el método de arrastrar y soltar para construir aplicaciones, lo que facilita el proceso de creación a través de bloques de código visuales que representan diferentes componentes y funcionalidades de la aplicación.

La plataforma es especialmente valiosa en el ámbito educativo, ya que introduce conceptos de programación y desarrollo de software de una manera accesible para estudiantes y principiantes. Además, MIT App Inventor fomenta la creatividad y la resolución de problemas, proporcionando una base sólida para aquellos interesados en la programación y el desarrollo de aplicaciones móviles. A lo largo de los años, ha sido ampliamente utilizada en programas educativos y talleres alrededor del mundo para enseñar los fundamentos de la programación y la lógica computacional[44].

La comunidad de MIT App Inventor es activa y proporciona numerosos recursos, tutoriales y ejemplos que ayudan a los usuarios a aprender y mejorar sus habilidades de desarrollo. La plataforma no solo simplifica el

proceso de creación de aplicaciones, sino que también permite a los usuarios experimentar con diferentes ideas y conceptos, impulsando la innovación y el aprendizaje práctico [45].

Figura 12. APP INVENTOR [45].

VI. Marco Metodológico

VI-A. Descripción del robot Móvil

El robot desarrollado es un sistema móvil inteligente diseñado para la detección de señales de tráfico, reconocimiento de peatones y control automatizado en entornos urbanos. Equipado con una variedad de sensores y actuadores, el robot tiene como objetivo principal asistir en el control de tráfico y mejorar la seguridad peatonal esto a través de un flujo de procesos y funcionamiento que se muestran en la Figura 13.

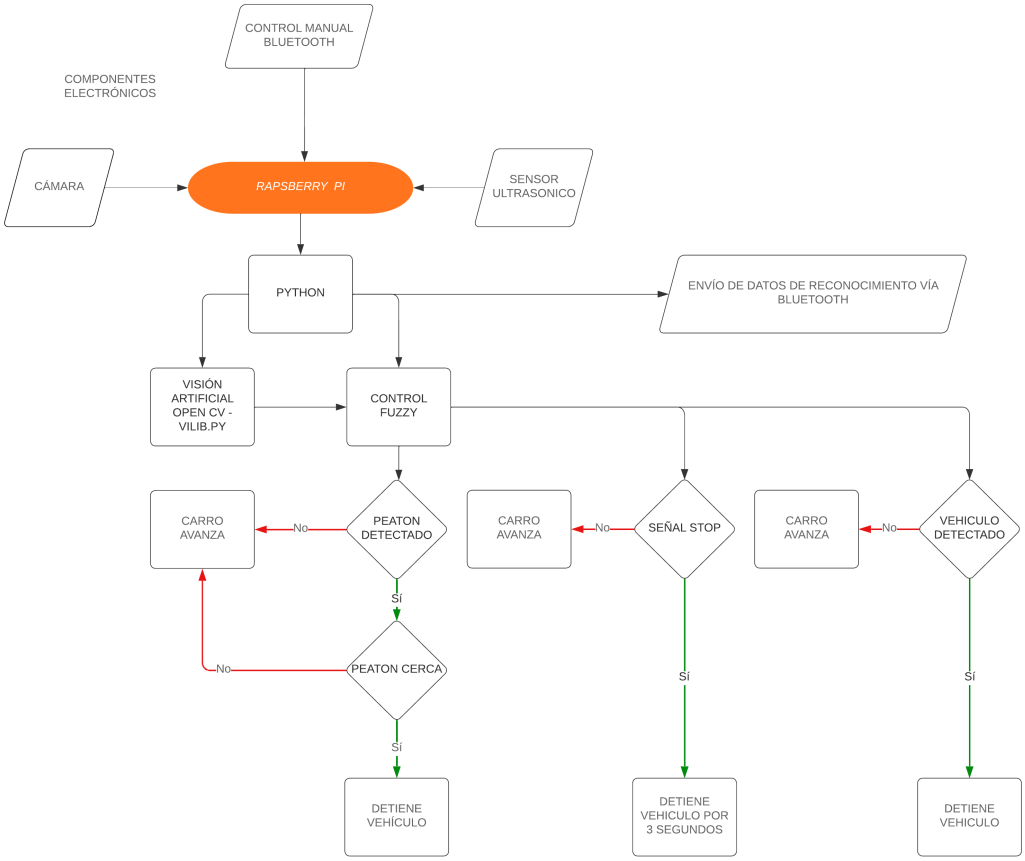


Figura 13. Diagrama de Flujo de Funcionamiento, por E. Barba

El flujo de funcionamiento del sistema comienza con la Visión Artificial, que procesa imágenes para detectar señales de tránsito, peatones y vehículos. Basado en estas detecciones, el Control Difuso toma decisiones: si detecta un peatón y está cerca, el robot se detiene; si detecta una señal de STOP, el robot se detiene por 3 segundos; si detecta un vehículo, el robot también se detiene. En caso de que no se detecten estos obstáculos, el robot continúa avanzando.

VI-A1. Características Principales:

1. Detección de Señales de Tráfico:
   - Utiliza una cámara y algoritmos de visión por computadora para identificar señales de tráfico, como la señal de ”STOP”. La detección de señales permite al robot tomar decisiones en tiempo real sobre su comportamiento.

2. Reconocimiento de Peatones:
   - Incorpora un sistema de detección de pose para identificar la presencia y ubicación de peatones en el entorno. Esta funcionalidad ayuda al robot a ajustar los mensajes para usuario y comportamiento en función de la proximidad de los peatones.

3. Control Basado en Fuzzy Logic:
   - Implementa un sistema de control difuso que ajusta un porcentaje de proximidad según la distancia y presencia de los peatones. El control fuzzy permite una adaptación suave y eficiente a diferentes condiciones del entorno.

4. Comunicación Bluetooth:
   - Incluye un módulo Bluetooth para recibir comandos remotos y enviar mensajes de estado. Esto permite el control manual del robot y la interacción con otros dispositivos.

5. Hardware:
   - Está construido sobre una plataforma Raspberry Pi con una serie de sensores como un sensor ultrasónico para medir distancias, servomotores para el control de la cámara y otros componentes electrónicos necesarios para la operación del robot.

6. Funcionalidad de Audio:
   - Equipado con un sistema de síntesis de voz y música para proporcionar retroalimentación auditiva, incluyendo alertas sobre la detección de señales de tráfico y la proximidad de peatones.

VI-B. Programación en Python

Se realiza realiza la programación en Python de Raspberry Pi usando librerías como CV2, Tensorflow, Vilib.py, picamera.

VI-B1. Explicación del código:

1. Descarga de Librerias: Las librerías necesarias para el robot se encuentran en el repositorio GIT de sunfounder picar-x, se realiza el comando git clone abriendo un terminal de Raspberry Pi como se muestra en la Figura 14.

Figura 14. Descarga de Librerías, por E. Barba, Terminal Raspberry Pi

2. Calibración de componentes de Robot: La librería picar-x tiene como funciones internas la calibración de servomotores, velocidades de motores, test de componentes para guardar los parámetros de uso colocando el siguiente comando en la terminal de Raspberry como se muestra en la figura a continuación en las Figuras 15 y 16.

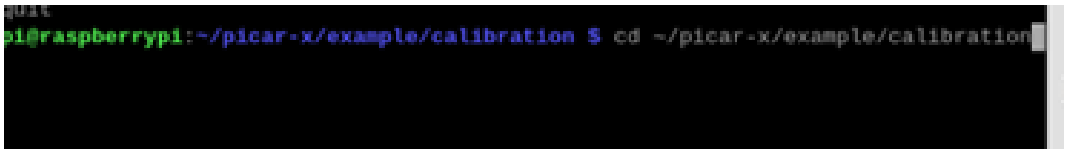


Figura 15. Descarga de Librerías, por E. Barba, Terminal Raspberry Pi

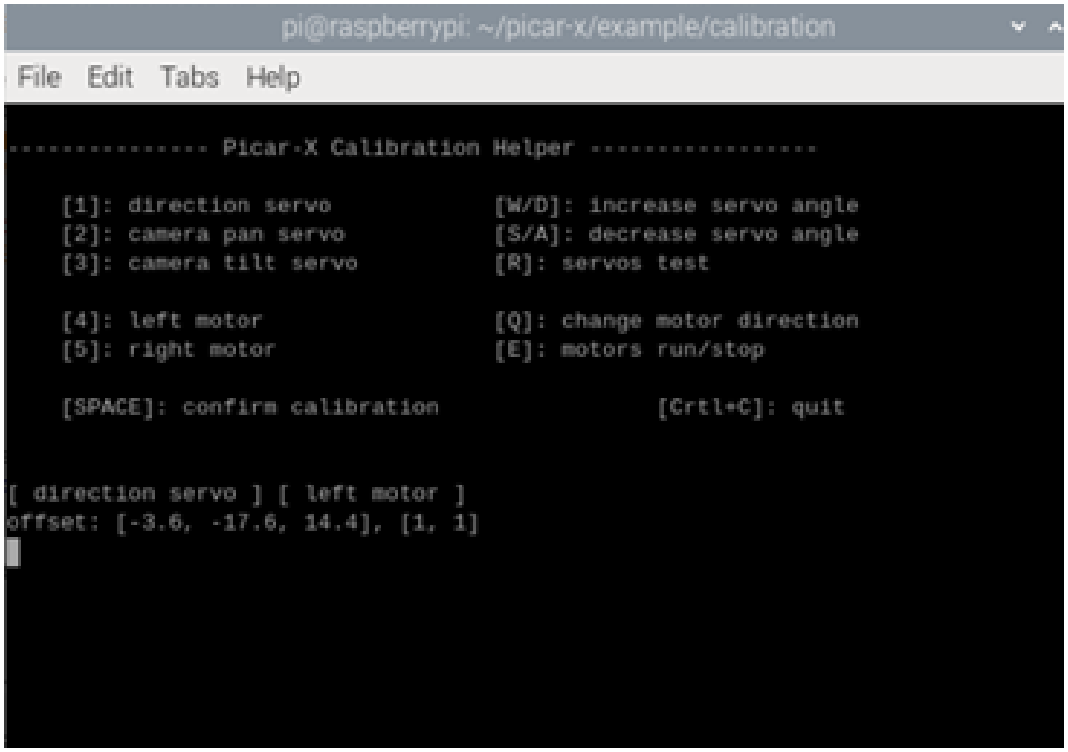


Figura 16. Descarga de Librerías, por E. Barba, Terminal Raspberry Pi

VI-C. Uso de librería ViLib.py

Vilib ofrece funcionalidades de reconocimiento de objetos y señales para trabajar con cámaras y sensores implementados en robots móviles y sistemas de visión artificial, las funciones necesarias para este proyecto son las siguientes:

VI-C1. Inicialización de cámara Raspberry Pi: Inicialización de cámara Raspberry Pi: La función camera () permite inicializar la cámara tanto localmente como en web usando CV2 y usando librería flask para inicialización de un servidor local de página web para mostrar la cámara mediante red como se muestra en la Figura 17.

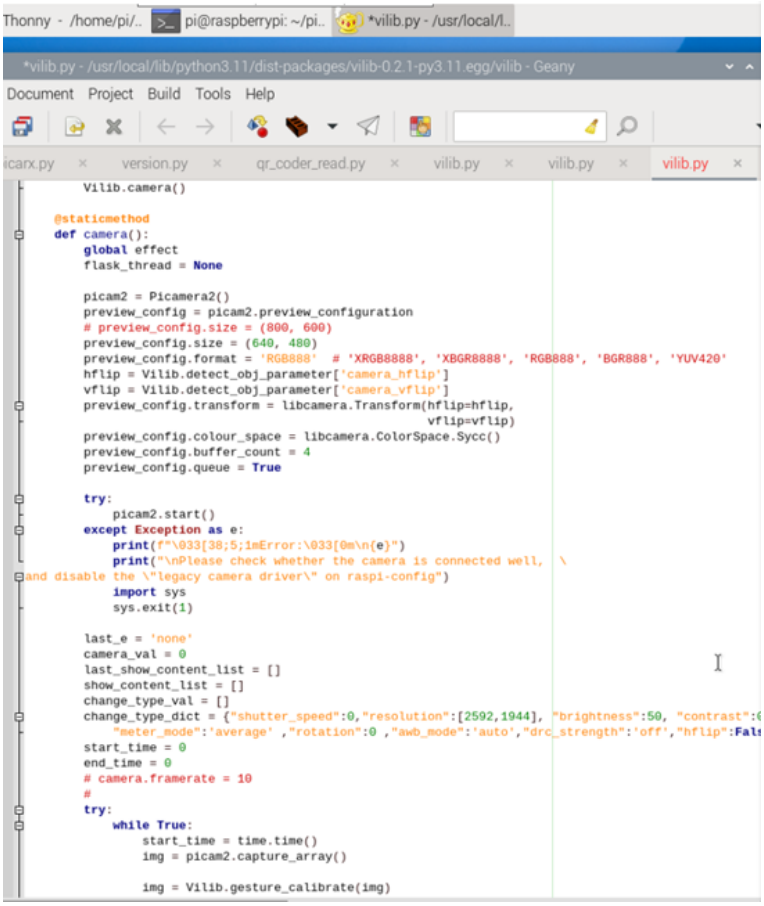


Figura 17. Blynk Dashboard, por E. Barba, ViLib.py

VI-C2. Función para detección de señales de tráfico: La función traffic_detect toma una imagen (img) y realiza varios pasos para identificar señales de tráfico en ella. Utiliza técnicas de procesamiento de imágenes y visión por computadora para encontrar y clasificar señales de tráfico, y luego actualiza parámetros de detección en Vilib.

- Verificación de Detección de Señal:

```
if Vilib.detect_obj_parameter['ts_flag'] == True:
```

  La función solo procede si ts_flag en los parámetros de detección de Vilib es True. Esto indica que se debe realizar la detección de señales de tráfico.

- Conversión de Color:

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

  Convierte la imagen de formato BGR (color estándar de OpenCV) a formato HSV (tono, saturación y valor), que es útil para segmentar colores específicos.

- Segmentación de Colores: Se crean máscaras para diferentes colores que podrían indicar señales de tráfico.

```
mask_red_1 = cv2.inRange(hsv, (157, 20, 20),
(180, 255, 255))
mask_red_2 = cv2.inRange(hsv, (0, 20, 20),
(10, 255, 255))
```

```
mask_blue = cv2.inRange(hsv, (92, 10, 10),
(125, 255, 255))
mask_all = cv2.bitwise_or(mask_red_1, mask_blue)
mask_all = cv2.bitwise_or(mask_red_2, mask_all)
```

- mask_red_1 y mask_red_2 segmentan el color rojo.
- mask_blue segmenta el color azul.
- mask_all combina estas máscaras para obtener todas las áreas rojas y azules en la imagen.

- Operaciones Morphológicas:

```
open_img = cv2.morphologyEx(mask_all, cv2.MORPH_OPEN,
Vilib.kernel_5, iterations=1)
```

Se aplica una operación de apertura para eliminar ruido y mejorar la detección de contornos. La apertura es una operación morfológica que elimina pequeños objetos del fondo.

- Detección de Contornos:

```
contours, hierarchy = findContours(open_img)
contours = sorted(contours, key=Vilib.cnt_area,
reverse=False)
```

Se encuentran los contornos en la imagen procesada y se ordenan por área. findContours es una función que detecta los bordes de los objetos en la imagen.

- Clasificación y Detección de Señales:

```
for i in contours:
    x, y, w, h = cv2.boundingRect(i)
acc_val, traffic_type = Vilib.traffic_predict(img, x, y, w, h)
```

Para cada contorno detectado:

- Se calcula el rectángulo delimitador.
- Se utiliza Vilib.traffic_predict para predecir el tipo de señal y la precisión de la detección.

- Dibujo y Etiquetado de Señales: Si la precisión (acc_val) es mayor o igual al 75 %:

```
if acc_val >= 75:
    if traffic_type == 1 or traffic_type == 2 or
    traffic_type == 3:
```

Dibuja y etiqueta la señal en la imagen, se dibuja un círculo alrededor de la señal detectada y se etiqueta con el tipo de señal y precisión.

- Actualización de Parámetros de Detección:

```
if traffic_sign_num > 0:
    Vilib.detect_obj_parameter['traffic_sign_x']
    = int(max_obj_x +
    max_obj_w / 2)
    Vilib.detect_obj_parameter['traffic_sign_y']
```

```
            = int(max_obj_y +
            max_obj_h / 2)
            Vilib.detect_obj_parameter['traffic_sign_w']
            = max_obj_w
            Vilib.detect_obj_parameter['traffic_sign_h']
            = max_obj_h
            Vilib.detect_obj_parameter['traffic_sign_t'] =
            traffic_dict[max_obj_t]
            Vilib.detect_obj_parameter['traffic_sign_acc']
            = max_obj_acc
    else:
            Vilib.detect_obj_parameter['traffic_sign_x'] = 320
            Vilib.detect_obj_parameter['traffic_sign_y'] = 240
            Vilib.detect_obj_parameter['traffic_sign_w'] = 0
            Vilib.detect_obj_parameter['traffic_sign_h'] = 0
            Vilib.detect_obj_parameter['traffic_sign_t'] = 'none'
            Vilib.detect_obj_parameter['traffic_sign_acc'] = 0
```

La función traffic_detect realiza una serie de pasos de procesamiento de imágenes para detectar señales de tráfico en una imagen. Utiliza técnicas de segmentación de colores, operaciones morfológicas, y detección de contornos para identificar señales específicas y actualizar parámetros de detección en Vilib. Luego, dibuja y etiqueta las señales detectadas en la imagen para su visualización siguiendo la Figura 18

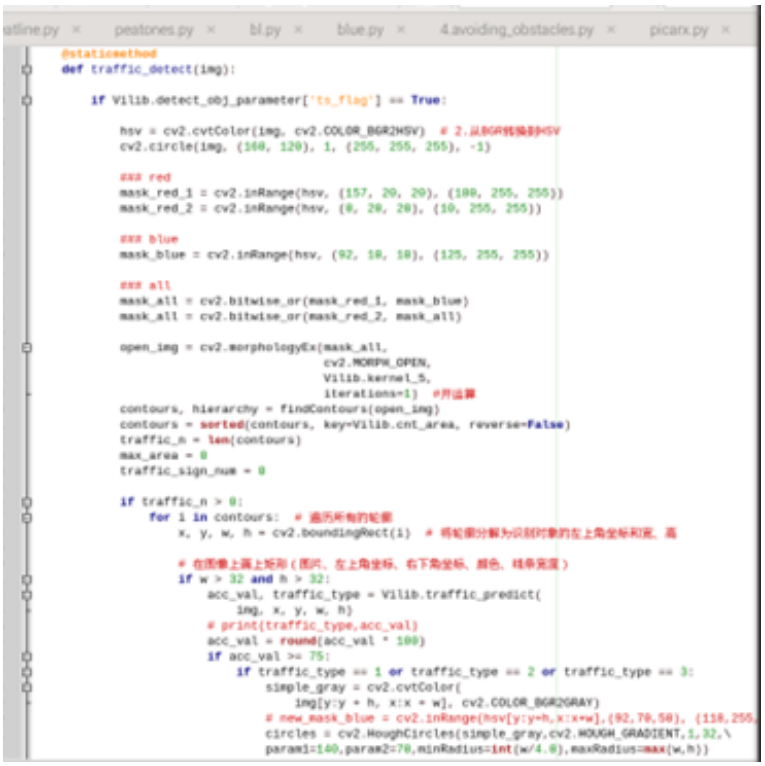


Figura 18. Detección de tráfico, por E. Barba, ViLib.py

VI-C3. Función para reconocimiento de peatones: La función human_detect_func () toma una imagen (img) y realiza los siguientes pasos para detectar y marcar rostros humanos en ella. Utiliza la detección de rostros mediante un clasificador de cascada y actualiza los parámetros de detección en Vilib.

Verificación de Detección de Humanos:

```
if Vilib.detect_obj_parameter['hdf_flag'] == True:
```

La función procede solo si hdf_flag en los parámetros de detección de Vilib es True. Esto indica que la detección de humanos está habilitada desde programa principal.

Redimensionamiento y Conversión a Escala de Grises:

```
resize_img = cv2.resize(img, (320, 240),
interpolation=cv2.INTER_LINEAR) gray =
cv2.cvtColor(resize_img, cv2.COLOR_BGR2GRAY)
```

- Redimensionamiento: La imagen se redimensiona a 320x240 píxeles para reducir el tamaño y mejorar la eficiencia de la detección.

- Conversión a Escala de Grises: La imagen redimensionada se convierte a escala de grises, ya que el clasificador de cascada de Haar trabaja con imágenes en escala de grises.

  Detección de Rostros:

  ```
  faces = Vilib.face_cascade.detectMultiScale(gray, 1.3, 2)
  ```

  Utiliza el clasificador de cascada (Vilib.face_cascade) para detectar rostros en la imagen en escala de grises. El método detectMultiScale devuelve una lista de rectángulos donde se detectan rostros.

  Actualización de Parámetros de Detección:

  ```
  Vilib.detect_obj_parameter['human_n'] = len(faces)
  ```

  Dibujar Rectángulos y Actualizar Parámetros de Detección:

  ```
  max_area = 0 if Vilib.detect_obj_parameter['human_n'] >
  0: for (x, y, w, h) in faces: x = x * 2 y = y * 2 w = w
  * 2 h = h * 2 cv2.rectangle(img, (x, y), (x + w, y +
  h), (255, 0, 0), 2) object_area = w * h if object_area
  > max_area: object_area = max_area
  Vilib.detect_obj_parameter['human_x'] = int(x + w / 2)
  Vilib.detect_obj_parameter['human_y'] = int(y + h / 2)
  Vilib.detect_obj_parameter['human_w'] = w
  Vilib.detect_obj_parameter['human_h'] = h
  ```

  La función human_detect_func () detecta rostros humanos en una imagen. Redimensiona la imagen, la convierte a escala de grises, y utiliza un clasificador de cascada de Haar para encontrar rostros. Luego, actualiza los parámetros de detección en Vilib y dibuja rectángulos alrededor de los rostros detectados en la imagen. Si no se detecta ningún rostro, establece valores predeterminados para los parámetros de detección.
- Dibujo de Rectángulos: Se dibuja un rectángulo rojo alrededor de cada rostro detectado en la imagen original.

- Ajuste de Coordenadas: Las coordenadas y dimensiones del rectángulo se duplican para ajustarlas a la imagen original (ya que la imagen fue redimensionada).

- Actualización de Parámetros: Se actualizan los parámetros de detección en Vilib con la posición y dimensiones del rostro detectado con el área más grande.

  Este proceso se muestra en la Figura 19

```python
@staticmethod
def human_detect_func(img):
    if Vilib.detect_obj_parameter['hdf_flag'] == True:
        resize_img = cv2.resize(
            img, (320, 240),
            interpolation=cv2.INTER_LINEAR)
        gray = cv2.cvtColor(resize_img, cv2.COLOR_BGR2GRAY)
        faces = Vilib.face_cascade.detectMultiScale(gray, 1.3, 2)
        # print(len(faces))
        Vilib.detect_obj_parameter['human_n'] = len(faces)
        max_area = 0
        if Vilib.detect_obj_parameter['human_n'] > 0:
            for (x, y, w, h) in faces:
                x = x * 2
                y = y * 2
                w = w * 2
                h = h * 2
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                object_area = w * h
                if object_area > max_area:
                    object_area = max_area
                    Vilib.detect_obj_parameter['human_x'] = int(x + w / 2)
                    Vilib.detect_obj_parameter['human_y'] = int(y + h / 2)
                    Vilib.detect_obj_parameter['human_w'] = w
                    Vilib.detect_obj_parameter['human_h'] = h
        else:
            Vilib.detect_obj_parameter['human_x'] = 320
            Vilib.detect_obj_parameter['human_y'] = 240
            Vilib.detect_obj_parameter['human_w'] = 0
            Vilib.detect_obj_parameter['human_h'] = 0
            Vilib.detect_obj_parameter['human_n'] = 0
        return img
    else:
        return img
```

Figura 19. Detección de cuerpo humano, por E. Barba, ViLib.py

VI-C4. Entrenamiento de modelo para reconocimiento de Vehículos: Para revelar un modelo cognitivo de visión artificial, se utilizarán modelos de entrenamiento para la identificación de imágenes que representan objetos, en este caso el tráfico vehicular. Teachable Machine es una herramienta que permite crear modelos de aprendizaje automático de manera intuitiva y accesible.

Fue desarrollada por Google con el objetivo de democratizar el acceso a la inteligencia artificial, permitiendo a cualquier persona, independientemente de su nivel técnico, entrenar modelos de IA para reconocimiento de imágenes, sonido y poses sin necesidad de escribir código, en la Figura 20 se muestra la página principal de la página web.

Figura 20. Entrenamiento Modelo detección Vehicular, por E. Barba, Teachable Machine

- Recopilación de Datos: El primer paso en Teachable Machine es la recopilación de datos. Se captan imágenes de vehículos con una cámara y se empieza muestrear. La cantidad de datos de muestreo indicará la precisión del modelo como se puede ver en la Figura 21

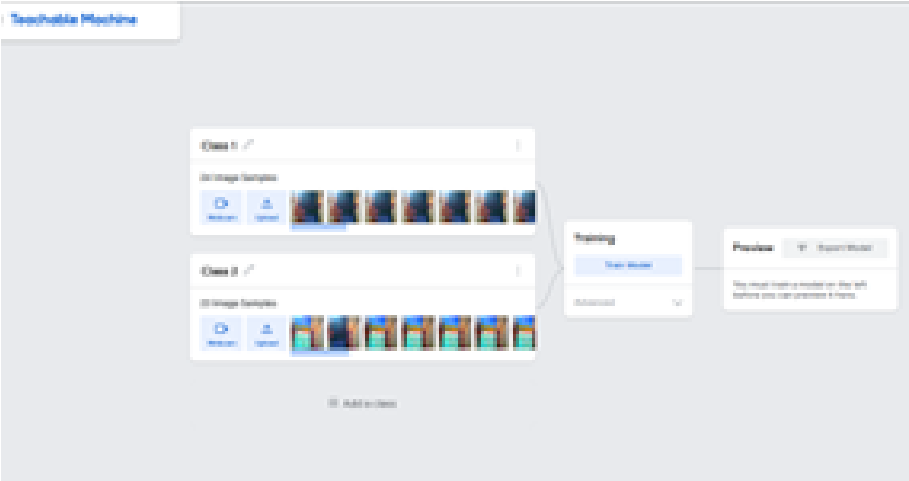


Figura 21. Recopilación de datos para detección vehicular, por E. Barba, Teachable Machine

- Entrenamiento del Modelo: Una vez recopilados los datos, se procede al entrenamiento del modelo. Teachable Machine utiliza algoritmos de aprendizaje automático para encontrar patrones en los datos proporcionados. El proceso de entrenamiento se realiza en la nube, y se puede ver en tiempo real cómo el modelo mejora su precisión a medida que se entrena con más datos.

- Evaluación y Exportación: Después del entrenamiento, el modelo puede ser evaluado utilizando datos de prueba. Teachable Machine proporciona una interfaz visual para probar el modelo y ver su desempeño. Una vez satisfechos con los resultados, se exporta el modelo para su uso en la programación del robot. En la Figura 22 se muestran los distintos modelos que se pueden exportar en formatos compatibles con TensorFlow, TensorFlow Lite y JavaScript, en este caso se utilizan los modelos de TensorFlow Lite con tipo de modelo Quantized.

Figura 22. Exportación de Modelo, por E. Barba, Teachable Machine

- Integración con programa de robot: Se modifica parte del código Vilib.py para obtener los resultados necesarios en la programación inicial colocando las direcciones de los archivos de modelo entrenado previamente.
  Los modelos entrenados para detectar una calle vacía y una calle con vehículos se reciben en la programación principal realizando una acción de reconocimiento mediante la cámara en tiempo real y presentando el resultado mediante un mensaje a la aplicación vía Bluetooth y un mensaje de sonido para alertar la presencia de vehículo, el código usado se encuentra en la Figura 23 y 24.

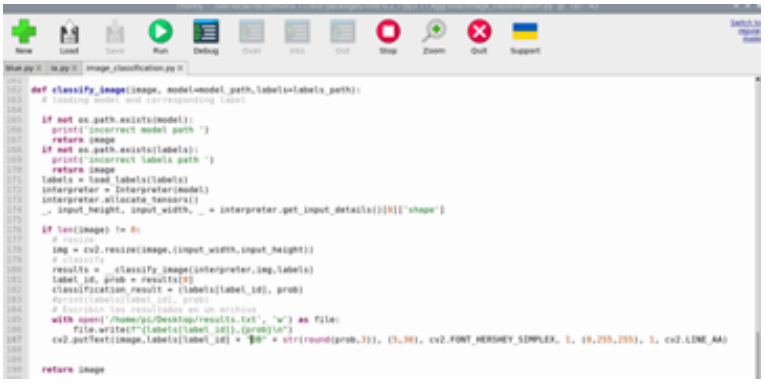


Figura 23. Funcion de modelo TensorFlow, por E. Barba

Figura 24. Detección de vehículos programa principal, por E. Barba

VI-D. Programa de Robot en Python

La programación principal de Robot se realiza en ambiente Python con IDE de programación Thony, se importan las bibliotecas necesarias de la manera que se muestra en la Figura 25, que incluyen Serial para comunicación Bluetooth, Threading para manejar múltiples tareas simultáneamente, Time para gestionar retrasos y tiempos de espera, Picarx para controlar el robot PicarX, Vilib para la detección de objetos y señales de tráfico mediante visión artificial, Robot_hat para reproducir música y convertir texto a voz (TTS), y Skfuzzy para lógica difusa.

```
import serial
import threading
import time
from picarx import Picarx
from time import sleep
from vilib import Vilib
from time import sleep
from robot_hat import Music,TTS
import readchar
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import numpy as np
```

Figura 25. Librerías programa principal , por E. Barba

VI-D1. Configuración de lógica Difusa: La lógica difusa es una forma de razonamiento que maneja la incertidumbre y la imprecisión, similar a cómo lo haría un humano. En lugar de valores binarios (verdadero/falso), la lógica difusa trabaja con grados de verdad, lo que permite una representación más flexible y realista de las situaciones.

1. Variables Difusas.
   Se definen tres variables difusas:

   - peaton_distancia: La distancia al peatón.
   - stop_senal: Si se ha detectado una señal de ”stop”.
   - seguimiento: La velocidad de seguimiento del robot.

2. Conjuntos Difusos.
   Para cada variable difusa, se definen conjuntos difusos que representan diferentes estados posibles.

3. peaton_distancia.

   - lejos: La distancia es grande.
   - cerca: La distancia es moderada.
   - muy_cerca: La distancia es pequeña.

4. stop_senal

   - no_detectada: No se ha detectado una señal de ”stop”.
   - detectada: Se ha detectado una señal de ”stop”.

5. seguimiento

   - lento: Velocidad baja.
   - moderado: Velocidad moderada.
   - rapido: Velocidad alta.

6. Reglas Difusas.
   Las reglas difusas definen cómo las entradas se combinan para producir una salida. En tu caso, se definen tres reglas:

   - Si la distancia al peatón es muy_cerca o se ha detectado una señal de ”stop”, el seguimiento es lento.
   - Si la distancia al peatón es cerca y no se ha detectado una señal de ”stop”, el seguimiento es moderado.
   - Si la distancia al peatón es lejos y no se ha detectado una señal de ”stop”, el seguimiento es rápido.

     En la Figura 26 se ve el comportamiento de estas reglas.

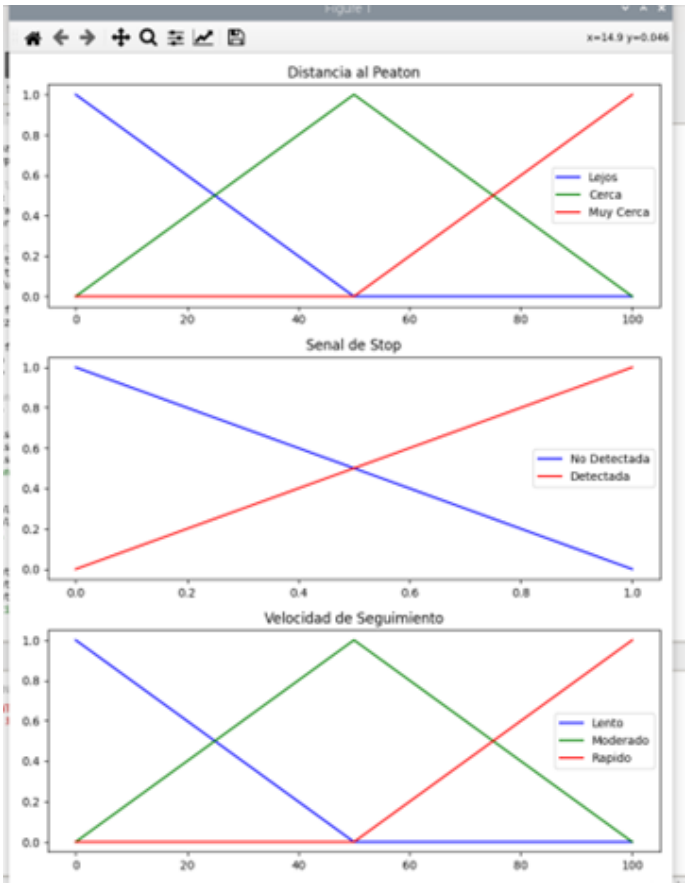


Figura 26. Grafico de las reglas difusas , por E. Barba

7. Inferencia Difusa.
   El proceso de inferencia difusa utiliza estas reglas para determinar la salida difusa basada en las entradas actuales.

8. Desfuzzificación.
   Finalmente, la desfuzzificación convierte la salida difusa en un valor crsip (preciso) que se puede usar para controlar el robot. este valor es la relación de porcentaje a la que el robot debe parar si detecta señales.

   En la Figura 27 se ve como está incluidas las reglas Difusas en la programación.

```
peaton_distancia = ctrl.Antecedent(np.arange(0, 101, 1), 'peaton_distancia')
stop_senal = ctrl.Antecedent(np.arange(0, 2, 1), 'stop_senal')
seguimiento = ctrl.Consequent(np.arange(0, 101, 1), 'seguimiento')

peaton_distancia['lejos'] = fuzz.trimf(peaton_distancia.universe, [0, 0, 50])
peaton_distancia['cerca'] = fuzz.trimf(peaton_distancia.universe, [0, 50, 100])
peaton_distancia['muy_cerca'] = fuzz.trimf(peaton_distancia.universe, [50, 100, 100])

stop_senal['no_detectada'] = fuzz.trimf(stop_senal.universe, [0, 0, 1])
stop_senal['detectada'] = fuzz.trimf(stop_senal.universe, [0, 1, 1])

seguimiento['lento'] = fuzz.trimf(seguimiento.universe, [0, 0, 50])
seguimiento['moderado'] = fuzz.trimf(seguimiento.universe, [0, 50, 100])
seguimiento['rapido'] = fuzz.trimf(seguimiento.universe, [50, 100, 100])

regla1 = ctrl.Rule(peaton_distancia['muy_cerca'] | stop_senal['detectada'], seguimiento['lento'])
regla2 = ctrl.Rule(peaton_distancia['cerca'] & stop_senal['no_detectada'], seguimiento['moderado'])
regla3 = ctrl.Rule(peaton_distancia['lejos'] & stop_senal['no_detectada'], seguimiento['rapido'])

control_seguimiento = ctrl.ControlSystem([regla1, regla2, regla3])
simulador_seguimiento = ctrl.ControlSystemSimulation(control_seguimiento)
```

Figura 27. Reglas Difusas , por E. Barba

VI-D2. Función de envío de mensajes vía bluetooth: La función de envío de mensajes vía Bluetooth permite la transferencia de mensajes de texto entre dispositivos sin necesidad de una conexión a Internet en la Figura 28 se puede ver como esta es integrada a la programación.

```
def enviar_mensaje(mensaje):
    bluetooth.write(mensaje.encode())
    print("Mensaje enviado:", mensaje)
```

Figura 28. Función envío mensajes bluetooth , por E. Barba

VI-D3. Función de detección de señales y peatones: Esta función corre en un hilo y continuamente como se ve en la Figura 29.

- Detecta señales de tráfico y peatones.
- Envía mensajes según la señal detectada vía bluetooth.
- Ajusta el promedio de detección del robot basado en la lógica difusa.
- Produce alertas de voz si un peatón está cerca.

```python
def detectar_senal():
    while True:
        signal = Vilib.detect_obj_parameter['traffic_sign_t']
        joints = Vilib.detect_obj_parameter['body_joints']

        distance = round(px.ultrasonic.read(), 2)
        print("distance: ",distance)

        peaton_detectado = isinstance(joints, list) and len(joints) == 33
        distancia_peaton = 100 if not peaton_detectado else distance
        if signal == "stop":
            enviar_mensaje("STOP")
            stop = 1
            bandera = True
            if bandera == True:
                pass
        else:
            bandera = False
            enviar_mensaje("SEGUIR")
            stop = 0

        simulador_seguimiento.input['peaton_distancia'] = distancia_peaton
        simulador_seguimiento.input['stop_senal'] = stop
        simulador_seguimiento.compute()
        porcentaje_seguimiento = simulador_seguimiento.output['seguimiento']
        print(f"Porcentaje de seguimiento: ")

        velocidad = (px_power * porcentaje_seguimiento) / 100
        print(velocidad)
        if porcentaje_seguimiento < 50:
            enviar_mensaje("SEGUIR")
        else:
            enviar_mensaje("PEATON CERCA % " + str(int(porcentaje_seguimiento)) )
            px.forward(0)
            words = "CUIDADO PEATON " + str(int(porcentaje_seguimiento) + " por ciento")
            tts.say(words)
            px.backward(0)

        time.sleep(0.1)

        if isinstance(joints,list) and len(joints) == 33:
            px.forward(0)
            px.backward(0)
            print("peaton detectado")
```

Figura 29. Función detección de señales , por E. Barba

VI-D4. Función de recepción de mensajes vía Bluetooth: Esta función realiza las siguientes acciones:

1. Detección de Señales de Tráfico: Verifica continuamente si se detecta una señal de ”stop”. Si es así, envía un mensaje de ”STOPz detiene el robot, además de reproducir un mensaje de voz.

2. Lectura de Datos Bluetooth: Lee un byte de datos recibidos a través de Bluetooth.

3. Procesamiento de Comandos: Decodifica y procesa los comandos recibidos, ajustando los ángulos de la cámara y los movimientos del robot según el comando.

   - Letras entre ’a’ y ’q’: Mapea la letra a un valor de ángulo para el servo.
   - Numeros ’2’, ’3’, ’z’, ’1’: Ajusta los ángulos de pan y tilt de la cámara.
   - Letras ’A’, ’B’, ’s’: Controlan el movimiento del robot hacia adelante, atrás, o detención.
   - Letra ’G’: Reproduce un mensaje de voz.

En la Figura 30 se observa como se integró la función.

```python
def recibir_mensajes():
    angulo_pan_actual = 0
    angulo_tild_actual = 0
    while True:
        signal = Vilib.detect_obj_parameter['traffic_sign_t']
        if signal == "stop":
            enviar_mensaje("STOP")
            if bandera2 == False:
                px.forward(0)
                px.backward(0)
                words = "senial pare detectada"
                tts.say(words)
                time.sleep(3)
                bandera2 = True
        else:
            bandera2 = False
        datos_recibidos = bluetooth.read(1)
        if datos_recibidos:
            mensaje_recibido = datos_recibidos.decode()
            if "a" <= mensaje_recibido <= "q":
                posicion = mapear_letra_a_valor(mensaje_recibido)
                print("Comando recibido:", mensaje_recibido, "Posicion del servo:", posicion)
                px.set_dir_servo_angle(posicion)
            if mensaje_recibido == "2":
                angulo_pan_actual += 5
                if angulo_pan_actual > 50:
                    angulo_pan_actual = 50
                px.set_cam_pan_angle(angulo_pan_actual)
            if mensaje_recibido == "3":
                angulo_pan_actual -= 5
                angulo_pan_actual = max(angulo_pan_actual, -50)
                px.set_cam_pan_angle(angulo_pan_actual)
            if mensaje_recibido == "z":
                angulo_tild_actual += 5
                angulo_tild_actual = min(angulo_tild_actual, 50)
                px.set_cam_tilt_angle(angulo_tild_actual)
            if mensaje_recibido == "1":
                angulo_tild_actual -= 5
                angulo_tild_actual = max(angulo_tild_actual, -50)
                px.set_cam_tilt_angle(angulo_tild_actual)
            if mensaje_recibido == "A":
                px.forward(px_power)
            if mensaje_recibido == "B":
                px.backward(px_power)
            if mensaje_recibido == "s":
                px.forward(0)
                px.backward(0)
            print("Mensaje recibido:", mensaje_recibido)
```

Figura 30. Función recepción de mensajes bluetooth , por E. Barba

VI-D5. Reconocimiento de vehículos: Los modelos obtenidos y descargados son colocados en las funciones de programa principal para su procesamiento de la siguiente manera:

1. Configuración del Modelo:

   - Vilib.image_classify_set_model(): Establece el modelo de clasificación de imágenes.
   - Vilib.image_classify_set_labels(): Establece las etiquetas para la clasificación.
   - Vilib.image_classify_switch(True): Activa la clasificación de imágenes.

2. Bucle Infinito:

   - Dentro de un bucle while True, el código lee continuamente el archivo results.txt configurado en la librería Vilib.py para obtener los resultados de la clasificación.
   - result = file.readline().strip(): Lee una línea del archivo y elimina espacios en blanco.
   - print(”ML RESULT: - result): Imprime el resultado de la clasificación.

3. Procesamiento del Resultado:

   - Si hay un resultado, se divide la cadena y se toma la primera parte (RESULTADO = result.split()[0]).
   - Si RESULTADO es ”1”, indica que se ha detectado un vehículo.
     - tts.say(”Vehiculo Detectado”): Utiliza el sistema de texto a voz para anunciar la detección de un vehículo.
     - enviar_mensaje(”Vehiculo detectado”): Envía un mensaje indicando que se ha detectado un vehículo.
     - bandera2 = True: Marca que se ha detectado un vehículo para evitar repetidos anuncios.
   - Si RESULTADO no es ”1”, bandera2 se reinicia a False.

   Esta integración se da como se muestra en la Figura 31.

```
def thread_main():

    Vilib.image_classify_set_model(path='/home/pi/vilib/model.tflite')
    Vilib.image_classify_set_labels(path='/home/pi/vilib/labels.txt')
    Vilib.image_classify_switch(True)

    while True:
        # Lee el archivo de resultados y muestra el contenido
        with open('/home/pi/Desktop/results.txt', 'r') as file:
            result = file.readline().strip()
            # label, prob = result.split(',')
            print("ML RESULT: "+ result)
            if result:
                RESULTADO = result.split()[0]

                if RESULTADO == "1":
                    if bandera2 == False:
                        tts.say("Vehiculo Detectado")
                        enviar_mensaje("Vehiculo detectado")
                        bandera2 = True
                else:
                    bandera2 = False
```

Figura 31. Detección de vehículos , por E. Barba

VI-D6. Inicio de hilos de programa: Los hilos son una técnica para ejecutar múltiples tareas en paralelo dentro de un solo proceso. Cada hilo representa una secuencia independiente de instrucciones a ejecutarse y ayudan a la velocidad de procesamiento del programa.
Los hilos realizan las siguientes acciones:

- Leer sensores (distancia al peatón, detección de señales).
- Procesar imágenes.
- Controlar motores y actuadores.
- Comunicar con otros dispositivos (vía Bluetooth).

Esto permitiría que el robot responda rápidamente a los cambios en su entorno sin demoras perceptibles debidas a la secuencialidad de las tareas.

En la Figura 32 se ve como se crean y se inician los hilos en la programación tales como:

- hilo_senal ejecuta la función detectar_senal, encargada de la detección de señales de tráfico y peatones.
- hilo_mensajes ejecuta la función recibir_mensajes, que maneja la recepción y el procesamiento de mensajes Bluetooth.
- main_thread.start(): Inicia la ejecución del hilo. La función thread_main() comenzará a ejecutarse en paralelo con el hilo principal del programa.

```
hilo_recepcion = threading.Thread(target=recibir_mensajes)
hilo_recepcion.daemon = True
hilo_recepcion.start()

hilo_senal = threading.Thread(target=detectar_senal)
hilo_senal.daemon = True
hilo_senal.start()

if __name__ == "__main__":

    main_thread = threading.Thread(target=thread_main)
    main_thread.start()
```

Figura 32. Hilos de programa robot , por E. Barba

VI-E. Programación App Móvil en APP INVENTOR

Se utiliza como medio de desarrollo la página web APP INVENTOR para desarrollo básico de aplicaciones Android debido a su facilidad de integración con dispositivos periféricos. Estos son los pasos a seguir para el desarrollo que dieron como resultado el de la Figura 33.

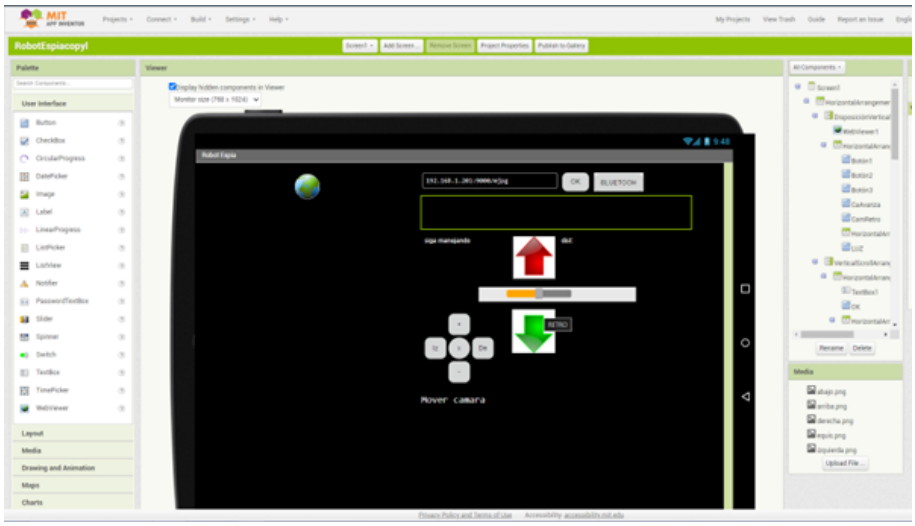


Figura 33. Interfaz App Inventor , por E. Barba

VI-E1. Diseño de la Interfaz en App Inventor: Se inicia un nuevo proyecto en App Inventor y accede al diseñador de la interfaz. En la Figura 34 se muestra como se añade el componente BluetoothClient desde la sección de Çonectividad"para habilitar la comunicación Bluetooth con el robot.

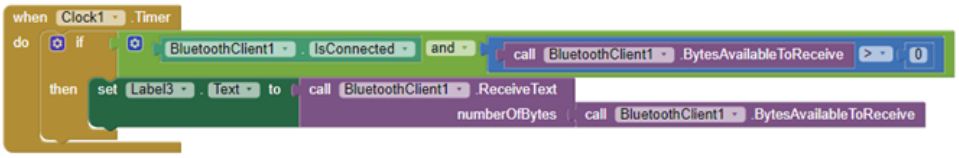


Figura 34. Conexión Bluetooth , por E. Barba

Se incorporan botones para los comandos del robot, etiquetados como .Avanzar", Retroceder", "Girar Izquierdaz "Girar Derecha". Estos botones envían comandos específicos al robot cuando se presionan, en la Figura 35 se muestra como están incorporados los comandos del robot en la App Inventor.

Figura 35. Bloques de envío Bluetooth App Inventor , por E. Barba

Se añade un componente WebViewer desde la sección de "Medios", que permite la visualización en tiempo real de la cámara web del robot.

Adicionalmente, se crean botones para maniobrar la cámara del robot, tales como "Mover Arriba", "Mover Abajo", "Mover Izquierdaz "Mover Derecha". Estos botones envían comandos específicos para ajustar la orientación de la cámara.

Se incluye un área para mostrar los mensajes recibidos del robot. Esto se hace mediante un componente Label o TextBox que se actualiza en tiempo real con la información enviada desde el robot.

VI-E2. Programación de los Comandos Bluetooth: En el bloque de programación, se utilizan bloques del componente BluetoothClient para establecer la conexión con el robot. El bloque Connect se configura con la dirección MAC del módulo Bluetooth del robot para permitir la comunicación.

Los botones de control del robot (Avanzar, Retroceder, etc.) se programan para enviar texto específico a través del bloque BluetoothClient.SendText cuando se presionan. Esto envía los comandos correspondientes al robot.

Se configuran bloques adicionales para manejar los eventos de conexión y desconexión de Bluetooth, informando al usuario sobre el estado de la conexión mediante mensajes en la interfaz de usuario.

VI-E3. Configuración y Visualización de la Cámara Web: El componente WebViewer se configura con la URL de la cámara web del robot. Es esencial que la cámara web esté transmitiendo en una dirección accesible para que el WebViewer pueda mostrar la transmisión en vivo.

El tamaño y la posición del WebViewer se ajustan para asegurar una visualización clara y accesible de la cámara web en la interfaz de usuario.

VI-E4. Control de la Cámara del Robot: Se añaden botones específicos para maniobrar la cámara, como ”Mover Arriba”, ”Mover Abajo”, ”Mover Izquierdaz ”Mover Derecha”. Estos botones envían comandos específicos al robot para ajustar la posición de la cámara según las necesidades del usuario.

Los comandos para el movimiento de la cámara se envían a través del componente BluetoothClient, utilizando bloques de programación que envían textos adecuados para controlar el ajuste de la cámara.

VI-E5. Visualización de Mensajes Recibidos: Para mostrar los mensajes enviados por el robot, se configura un componente Label o TextBox que se actualiza en tiempo real con los datos recibidos. Esto se logra mediante bloques que leen los datos recibidos del componente BluetoothClient y actualizan el componente de visualización en la interfaz de usuario.

VI-E6. Instalación en un Teléfono: Una vez finalizada la programación, el desarrollador selecciona ”Build.[en] el menú superior de App Inventor y elige .[A]pp (APK)”para generar el archivo APK de la aplicación.

El archivo APK se descarga en la computadora del desarrollador y se transfiere al teléfono mediante un cable USB o un servicio de transferencia de archivos.

En el teléfono, se navega a la ubicación del archivo APK y se abre para iniciar la instalación. Puede que el teléfono requiera permisos para instalar aplicaciones de fuentes desconocidas, los cuales deben ser concedidos para completar la instalación.

Tras la instalación, la aplicación estará disponible en el dispositivo, lista para controlar el robot y visualizar la cámara web, además de manejar la cámara del robot y recibir mensajes en tiempo real.

Este proceso permite a los usuarios controlar el robot mediante Bluetooth, visualizar la cámara web en vivo, maniobrar la cámara del robot y recibir mensajes en tiempo real desde la aplicación en su teléfono como se muestra en la Figura 36.

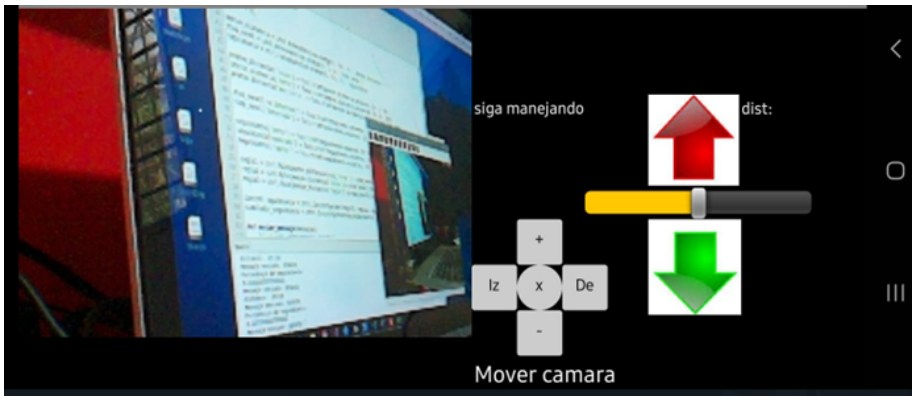


Figura 36. App Móvil para control de robot , por E. Barba

VII. Resultados

Los análisis y conclusiones de este proyecto son fundamentales para medir la mejora implementada en el Robot Móvil, así como para la validación y evaluación del mismo. Estos resultados también permiten identificar oportunidades para optimizaciones adicionales y facilitan la comunicación sobre el proyecto. A continuación, se presentan los hallazgos y el análisis correspondiente.

VII-A. Resultados Obtenidos

El robot, al implementar el sistema de reconocimiento de peatones basado en lógica difusa y Python, ha mostrado avances significativos en la identificación y seguimiento de peatones y vehículos en diferentes entornos.La integración de la lógica difusa ha permitido al robot lograr una detección precisa al ajustar sus parámetros de detección en tiempo real, mejorando la precisión del reconocimiento de peatones. Además, el sistema ha demostrado adaptación a diferentes condiciones, mostrando capacidad para ajustarse a variadas condiciones de iluminación y ángulos de visión. Gracias a la lógica difusa, el robot ha mejorado su capacidad de seguimiento efectivo de peatones, manteniéndose enfocado en su objetivo mientras se mueve en el campo de visión conjunto con el sensor ultrasónico, lo cual ha sido crucial para aplicaciones en tráfico y movilidad, donde el seguimiento preciso de peatones es esencial. La optimización del desempeño se ha logrado al reducir el tiempo de procesamiento y la carga computacional, resultando en una respuesta más rápida y eficiente durante la operación del robot.

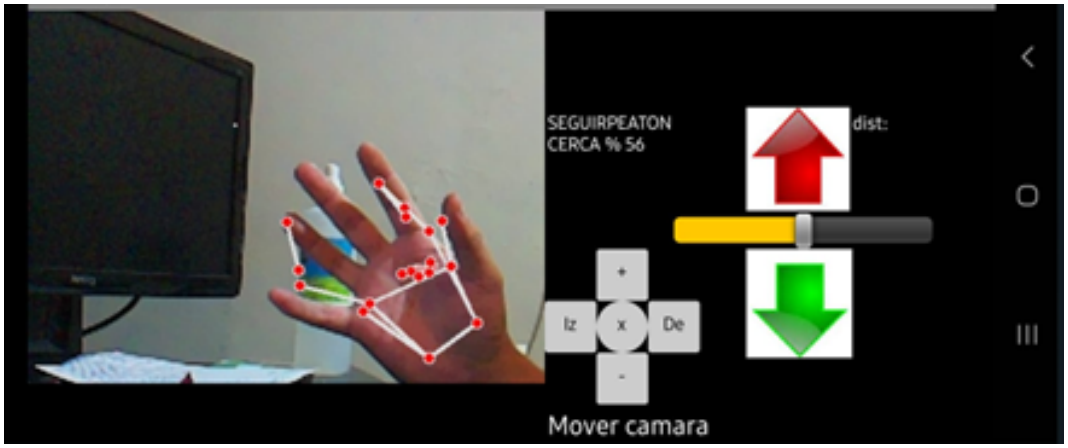


Figura 37. Identificación de peatones con porcentaje de aproximación (Antes) , por E. Barba

En la Figura 37 y la Figura 38 se ha observado una mejora en la interacción humano-robot, con el robot mostrando una mejor capacidad para interpretar las acciones y movimientos de los peatones, facilitando una interacción más fluida y segura en entornos compartidos, junto con la salida de lógica difusa sobre el porcentaje de aproximación de peatones para alertar a los usuarios del vehículo sobre los obstáculos peatonales en la vía. Finalmente, las pruebas de campo en diferentes escenarios del mundo real han confirmado que el sistema de lógica difusa proporciona resultados consistentes y confiables, permitiendo al robot identificar y seguir peatones con alta precisión en situaciones dinámicas y variadas, a su vez el uso de la lógica difusa en la programación del robot ha optimizado significativamente su capacidad de reconocimiento y seguimiento de peatones, haciendo el sistema más robusto y adaptable a diferentes condiciones de operación.

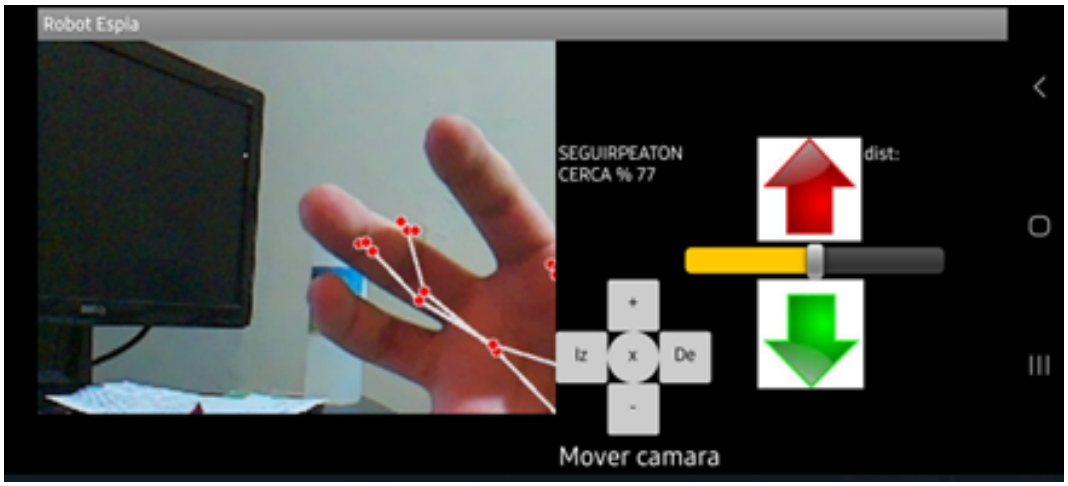


Figura 38. Identificación de peatones con porcentaje de aproximación (Despues) , por E. Barba

En la Tabla I se puede ver como actúa este sistema a través de la distancia tomando en cuenta que en el análisis realizado se tuvo las mismas condiciones externas.

Tabla I
RECONOCIMIENTO DE PEATONES

Detección de Peatón	
Distancia (cm)	Porcentaje de detección
30	56%
70	60%
120	66%
150	83%
200	94%

El sistema de reconocimiento de señales de tránsito, específicamente la señal de ”STOP”, ha sido implementado utilizando lógica difusa y Python, mostrando resultados destacables en la precisión y confiabilidad del proceso. El sistema ha demostrado alta precisión en la detección de señales STOP, con un algoritmo que ajusta los parámetros de detección para distinguir esta señal de otros signos y objetos, minimizando los errores de clasificación y permitiendo detener el vehículo por 3 segundos al detectar una señal. La adaptabilidad a diferentes entornos se ha mantenido consistente en diversas condiciones de iluminación, con la lógica difusa jugando un papel crucial en la identificación precisa de la señal STOP. Además, el sistema ha logrado una detección rápida y fiable de la señal STOP, con tiempos de respuesta eficientes que permiten al robot tomar decisiones en tiempo real, lo cual es esencial para garantizar la seguridad y la adecuada toma de decisiones. La mejora en la interacción con el entorno se ha logrado a través de una interpretación más efectiva de las señales STOP, permitiendo al robot tomar acciones apropiadas como detenerse o ajustar su velocidad. El sistema también ha mostrado robustez en condiciones adversas, manejando eficazmente situaciones como obstrucciones parciales, reflejos o señales dañadas, gracias a la lógica difusa que mantiene un alto nivel de precisión. Finalmente, los resultados consistentes en pruebas de campo han validado la eficacia del sistema, con el robot identificando correctamente la señal STOP en diversas configuraciones y entornos, y permitiendo la notificación en tiempo real de la señal escaneada.

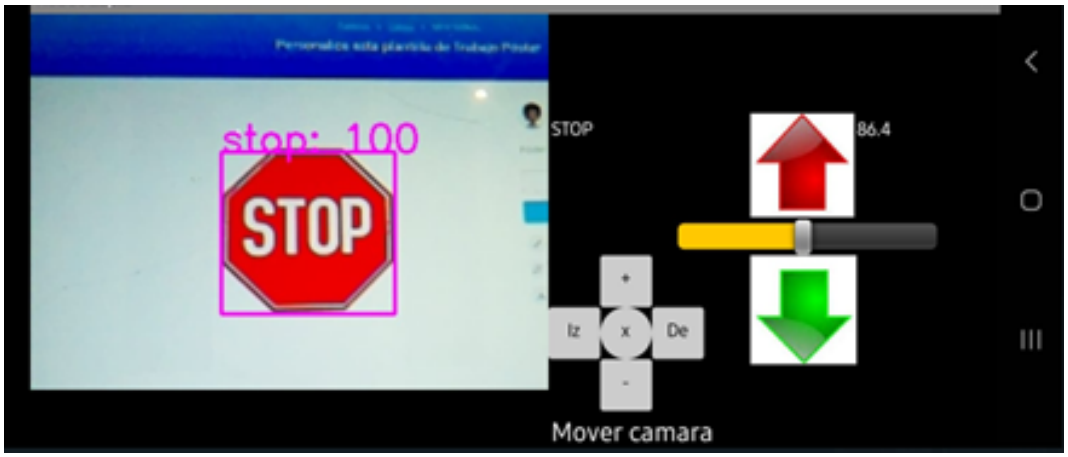


Figura 39. Visualización de señal stop , por E. Barba

A través de un análisis que se ve en la Tabla II se obtuvo los siguientes resultados basados en la distancia en la que se encuentran la señal.

Tabla II
RECONOCIMIENTO DE SEÑAL STOP

Detección de STOP	
Distancia (cm)	Porcentaje de detección
30	100%
70	94%
120	96%
150	92%
200	89%

VIII. CRONOGRAMA

A continuación se muestra el cronograma de trabajo en la figura III.

Tabla III
CRONOGRAMA

Sede:	Guayaquil	Estudiante - Autor	Edgar Daniel Barba Martillo																
Campus:	Centenario																		
Carrera:	Mecatrónica	Universidad Politécnica Salesiana																	
Tema de trabajo de titulación:		Implementación de un robot móvil para el control de tránsito y movilidad usando visión artificial.																	
Tutor del trabajo de titulación:		Ing. Tomás Santiago Gavilánez Gamboa, MSc																	
N°	Actividades	Estado	Meses/Semanas																
			may-23				jun-23				jul-23				ago-23				
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
	Planificación																		
1	Elaboración de un plan inicial de investigación.	Completado																	
2	Planteamiento del tema y método a utilizar	Completado																	
	Inicio del articulo académico																		
3	Construcción del robot	Completado																	
4	Armar el Circuito electrónico del robot	Completado																	
5	Establecer antecedentes del problema.	Completado																	
6	Implementar posibles soluciones de visión artificial	Completado																	
7	Entrenamiento de Visión artificial para detección de peatones, señales y autos.	Completado																	
8	Programar el funcionamiento mecánico del Robot	Completado																	
9	Realizar Pruebas	Completado																	
10	Realizar correcciones en base a los resultados de las pruebas	Completado																	
11	Elaborar el Documento de Tesis	Completado																	
12	Realizar correcciones del documento, por observaciones realizadas por el revisor	Completado																	
13	Presentación y exposición ante el tribunal de grado	Incompleto																	

IX. PRESUPUESTO

Tabla IV
PRESUPUESTO

Gastos del trabajo de investigación	**Valores ($)**
Carrocería del robot móvil	$ 250
Raspberry Pi	$ 80
Sensores	$ 40
Cámaras	$ 50
Baterías	$ 30
Asesoramiento externo	$ 100
Impresión de Documentación	$ 15
Servicio de Internet	$55
TOTAL	**$ 620**

X. Conclusiones

En este proyecto, se ha desarrollado un sistema avanzado de control y detección para un robot móvil. El objetivo principal fue implementar una solución que permitiera al robot reconocer señales de tráfico, vehículos y peatones, y ajustar su comportamiento en consecuencia. El robot está equipado con sensores como cámaras y un sistema de control por Bluetooth, así como una interfaz para interactuar con el usuario.

Se ha utilizado la biblioteca Vilib.py basada en OpenCV para la detección de señales de tráfico y la identificación de peatones. La detección de señales se realiza mediante el procesamiento de imágenes en tiempo real, identificando diferentes colores y formas de señales, mientras que la detección de peatones se basa en la identificación de áreas corporales en las imágenes capturadas, como se puede ver en la Tabla I y la II de cinco pruebas en distintas distancias se obtiene una efectividad entre en 56 % y el 94 % en el caso de los peatones y del 89 % al 100 % para el caso de la señal STOP.

Para el reconocimiento de vehículos, se empleó un modelo de aprendizaje automático entrenado mediante la herramienta. Teachable Machine Este proceso incluyó la carga de imágenes representativas de vehículos en Teachable Machine y la creación de un modelo entrenado que pudiera clasificar estos vehículos con precisión. El entrenamiento del modelo en Teachable Machine es accesible a través de su interfaz web, facilitando la integración del modelo en el proyecto y usando en Vilib.py.

Finalmente, se ha implementado un sistema de control basado en lógica difusa para notificar el porcentaje de acercamiento del robot según la proximidad de peatones y la detección de señales de tráfico. Esta solución combina hardware y software para lograr una interacción eficiente y segura en un entorno urbano.

XI. Recomendaciones

- Optimizar el Preprocesamiento de Imágenes:

  Es recomendable que el equipo mejore el preprocesamiento de imágenes para asegurar que los datos alimentados al modelo sean de alta calidad. Esto puede incluir técnicas de ajuste de iluminación, reducción de ruido y mejora del contraste. Optimizar estos aspectos ayudará a mejorar la precisión y la eficiencia del sistema de reconocimiento de señales y vehículos.

- Implementar Pruebas en Diferentes Condiciones Ambientales:

  Se sugiere realizar pruebas exhaustivas del sistema en diversas condiciones ambientales y de iluminación para evaluar su robustez. Esto incluye situaciones de baja visibilidad, variaciones en la luz solar y diferentes condiciones meteorológicas. La capacidad del sistema para funcionar de manera confiable en estos escenarios mejorará su efectividad general.

- Actualizar el Modelo con Nuevos Datos Regularmente:

  El equipo debe considerar la actualización regular del modelo de reconocimiento con nuevos datos para mantener la precisión del sistema. A medida que el entorno de tráfico y las señales cambian, el modelo debe adaptarse a estas variaciones. La incorporación de datos recientes y relevantes ayudará a que el sistema se mantenga actualizado y preciso.

- Desarrollar una Interfaz de Usuario Intuitiva:

  Es importante que se enfoque en desarrollar una interfaz de usuario intuitiva para facilitar la interacción con el sistema. La interfaz debe ser fácil de usar y proporcionar retroalimentación clara sobre las señales detectadas y las acciones realizadas. Una interfaz bien diseñada mejorará la experiencia del usuario y la eficiencia operativa del sistema.

- Monitorear el Rendimiento del Sistema y Ajustar Parámetros:

  Se recomienda monitorear continuamente el rendimiento del sistema y ajustar los parámetros según sea necesario. Esto incluye la evaluación de la precisión del reconocimiento, la velocidad de procesamiento y la estabilidad general del sistema. Ajustar los parámetros de acuerdo con el rendimiento observado ayudará a mantener la eficacia y la confiabilidad del sistema a lo largo del tiempo.

Referencias

[1] R. Cal, M. R. Spíndola y J. C. Grisales, *Ingeniería de Tránsito Fundamentos y Aplicaciones*. México: Alfaomega Grupo Editor, S.A., 2018.

[2] A. C. Rebelo y J. A. O. Toro, «Publicidad social y su influencia en la percepción de las campañas sociales de prevención de accidentes de tránsito en Ecuador,» *RETOS: Revista de Ciencias de la Administración y Economía*, 2020.

[3] K. Abata, F. Artega y D. Delgado, «Análisis del Congestionamiento Vehicular en Diferentes Intersecciones en la Ciudad de Portoviejo, Ecuador,» *RIEMAT*, 2022.

[4] Q. Álava y M. Santiago, «Análisis de riesgos laborales a los Agentes Civiles de la Agencia Municipal de Transito de la Ciudad de Esmeraldas,» Tesis de mtría., Pontificia Universidad Católica del Ecuador, 2020.

[5] «Guayaquil tiene vías más anchas que Quito y menos tráfico.,» *El Comercio*, 2022.

[6] M. E. P. Espinoza., «Análisis de riesgo ergonómicos derivados de la postura forzada en agentes de tránsito y seguridad vial de una empresa pública de control de tránsito y propuesta de medidas de intervención en la ciudad de Guayaquil, periodo 2018-2020,» Tesis de mtría., Escuela Superior Politécnica del Litoral (ESPOL)., 2021.

[7] D. J. R. Jarrín y J. K. S. Sánchez, «Diseño e implementación de un prototipo de semáforo controlado inalámbricamente y con sistema de energía emergente para el control del tránsito vehicular en la Ciudad de Guayaquil,» Tesis de mtría., Universidad de Guayaquil, 2019.

[8] C. de Candia, E. Kopacz y N. Raggio, «USO DE TECNOLOGÍA DE DRONES PARA EL RELEVAMIENTO DE INFORMACIÓN DEL TRÁNSITO,» en *2018: XXXII Jornadas de Investigación y XIV Encuentro Regional SI + Campos*, 2019.

[9] Y. A. R. Perdomo y M. S. R. Beltrán, «Visión artificial para el reconocimiento del tráfico vehicular,» Universidad Piloto de Colombia, Facultad de Ingenierías, Ingeniería Mecatrónica, inf. téc., 2020.

[10] M. A. S. Sánchez, «Visión artificial aplicada en escenarios reales. Una aproximación práctica,» Tesis doct., Universidad Rey Juan Carlos, 2021.

[11] O. P. de la Salud (OPS), *La seguridad vial se refiere a las medidas adoptadas para reducir el riesgo de lesiones y muertes causadas por el tránsito*. OPS, sin fecha.

[12] C. B. de Desarrollo de América Latina, *Movilidad sostenible y seguridad vial, un desafío para todos en el Ecuador*. CAF, sin fecha.

[13] M. Ángel Pérez, *Diseño de una carretera versus el comportamiento de los conductores. Adelantamiento, velocidad y distancia de visibilidad*. UPCommons, 2003.

[14] A. B. C. M. V. C. H. J. M. F. G. A. B. G. Guillermo Sierra Gros Pilar López Castillo, *Epidemiología y prevención de los accidentes de tráfico*. RSI, 2022.

[15] F. M. V. L. Ángel David Rivera Tigre, *Factores de riesgos sociales que intervienen en la ocurrencia de accidentes de tránsito con vehículos livianos*. sin información, 2021.

[16] X. Zhao et al., «Human factors in road traffic accidents: An analysis of crash data,» *Accident Analysis Prevention*, vol. 174, pág. 106 735, 2022.

[17] K. Rumar, «The role of perceptual and cognitive filters in observed behavior,» en *Human Behavior and Traffic Safety*, 1985, págs. 151-165.

[18] W. H. Organization, *Global status report on road safety 2018*. World Health Organization, 2018.

[19] F. F. C. M. O. B. R. M. Tarek Ziad Ashhad Verdezoto, *Análisis del congestionamiento vehicular para el mejoramiento de vía principal en Guayaquil-Ecuador*. Universidad Centroccidental Lisandro Alvarado, 2020.

[20] H. Wu et al., «Traffic congestion, atmospheric particulate matter, and public health: Evidence from Chinese cities,» *Journal of Cleaner Production*, vol. 253, pág. 119 925, 2020.

[21] X. Cao et al., «Urban planning and traffic congestion: A review of the evidence,» *Transportation Research Part D: Transport and Environment*, vol. 90, pág. 102 654, 2021.

[22] L. Zhang et al., «Vehicle emissions and traffic congestion: Evidence from China,» *Environmental Science Technology*, vol. 53, n.º 23, págs. 13 982-13 991, 2019.

[23] A. Amin et al., «The economic impact of traffic congestion: A case study of London,» *Journal of Urban Economics*, vol. 45, n.º 2, págs. 231-249, 2019.
[24] Q. Wang et al., «Urban traffic congestion and ambient air pollution: Evidence from Chinese cities,» *Journal of Environmental Economics and Management*, vol. 100, pág. 102 270, 2020.
[25] J. Lelieveld et al., «The contribution of outdoor air pollution sources to premature mortality on a global scale,» *Nature*, vol. 525, n.º 7569, págs. 367-371, 2015.
[26] S. Thrun, «Toward Robotic Cars,» *Communications of the ACM*, 2010.
[27] T. Litman, *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning*. Victoria Transport Policy Institute, 2020.
[28] Autocasion.com, «¿Qué tipos de asistentes de aparcamiento hay?» *Autocasion*, 2024.
[29] S. International, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE International, 2018.
[30] L. C. G. Lady Viviana Guzmán Palma, *Estudio bibliográfico de sistemas de transporte inteligente orientado a los buses urbanos de la ciudad Portoviejo*. sin información, 2023.
[31] B. Koigi, *El robot policía del Congo resuelve problemas de tráfico*. Fair Planet, 2018.
[32] J. Redmon et al., «YOLO9000: Better, Faster, Stronger,» en *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[33] G. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Contro*. MIT Press, 2005.
[34] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT Press, 2011.
[35] D. G y J. M, *Computational Principles of Mobile Robotics*. Cambridge University Press, 2010.
[36] F. G. Caballero-Julián, M. Morales-Hernández, E. M. Silva-Cruz y D. G. Caballero-Cantarell, «Raspberry Pi, conectividad y programación mediante puertos GPIO,» *Revista de Ingeniería Innovativa*, 2020.
[37] A. Alex. «Notebookcheck.» (2024), dirección: https://www.notebookcheck.org/Todos-los-SBC-de-Raspberry-Pi-pueden-optar-a-una-funcionalidad-mejorada-gracias-a-una-nueva-actualizacion.854619.0.html.
[38] SunFounder. «SunFounder Robot HAT.» (2021), dirección: https://docs.sunfounder.com/projects/robot-hat/en/latest/.
[39] F. Cristancho. «talently.» (2022), dirección: https://talently.tech/blog/python-ventajas-y-desventajas/.
[40] L. A. Zadeh, «Fuzzy sets,» *Information and Control*, vol. 8, n.º 3, págs. 338-353, 1965.
[41] «What Is Fuzzy Logic Toolbox?» (2023), dirección: https://la.mathworks.com/videos/what-is-fuzzy-logic-toolbox-1678346486056.html.
[42] C. Corporation, «INTRODUCCIÓN A LA VISIÓN ARTIFICIAL - Una guía para la automatización de procesos y mejorar la calidad,» *Cognex*, 2018.
[43] «Raspberry Pi.» (2022), dirección: https://www.raspberrypi.com/.
[44] D. Wolber, H. Abelson, E. Spertus y L. Looney, *App Inventor 2: Create Your Own Android Apps*. O'Reilly Media, 2014.
[45] P. T y B. D, *Mobile Apps for Everyone: Using App Inventor for Android (2nd ed.)* Jones Bartlett Learning, 2016.
[46] «SunFounder PiCar-X.» (2024), dirección: https://picar-x-v20.rtfd.io/.

ANEXO A
MATERIALES

Materiales de construccion del robot:

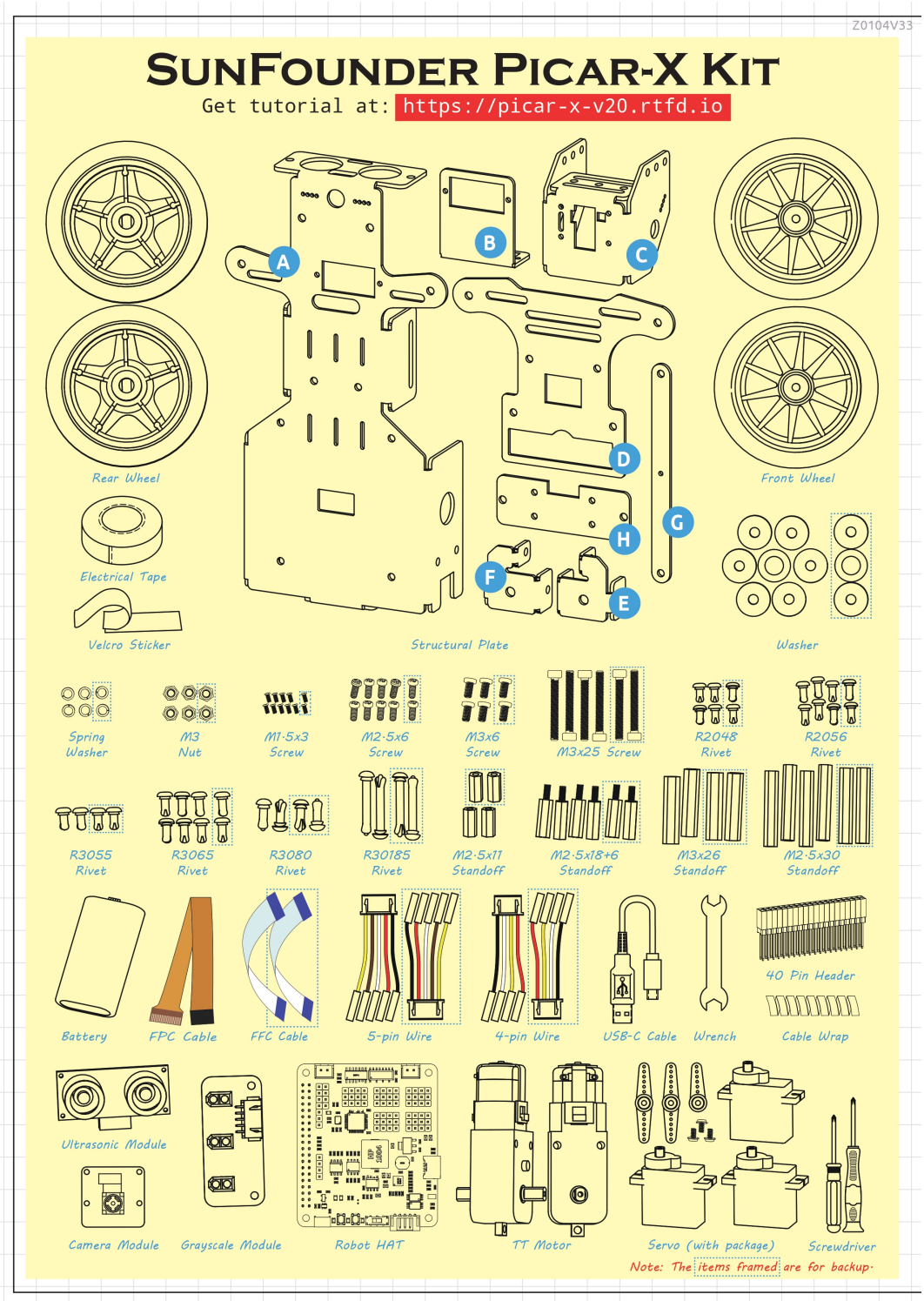


Figura 40. SunFounder Picar-X Kit [46].

ANEXO B
PASOS DE ENSAMBLE DEL ROBOT

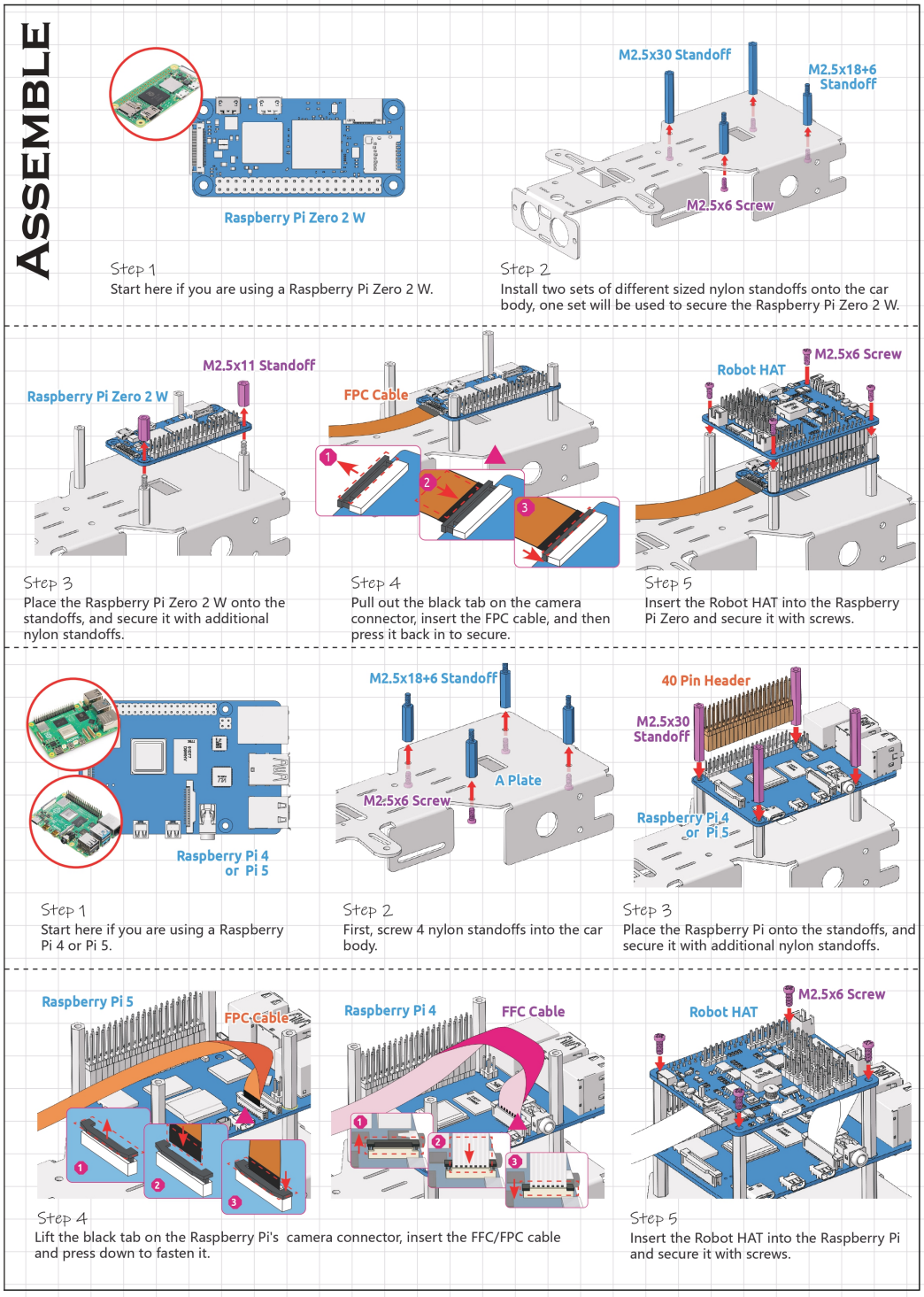


Figura 41. SunFounder Picar-X Kit, Paso 1 al 5 [46].

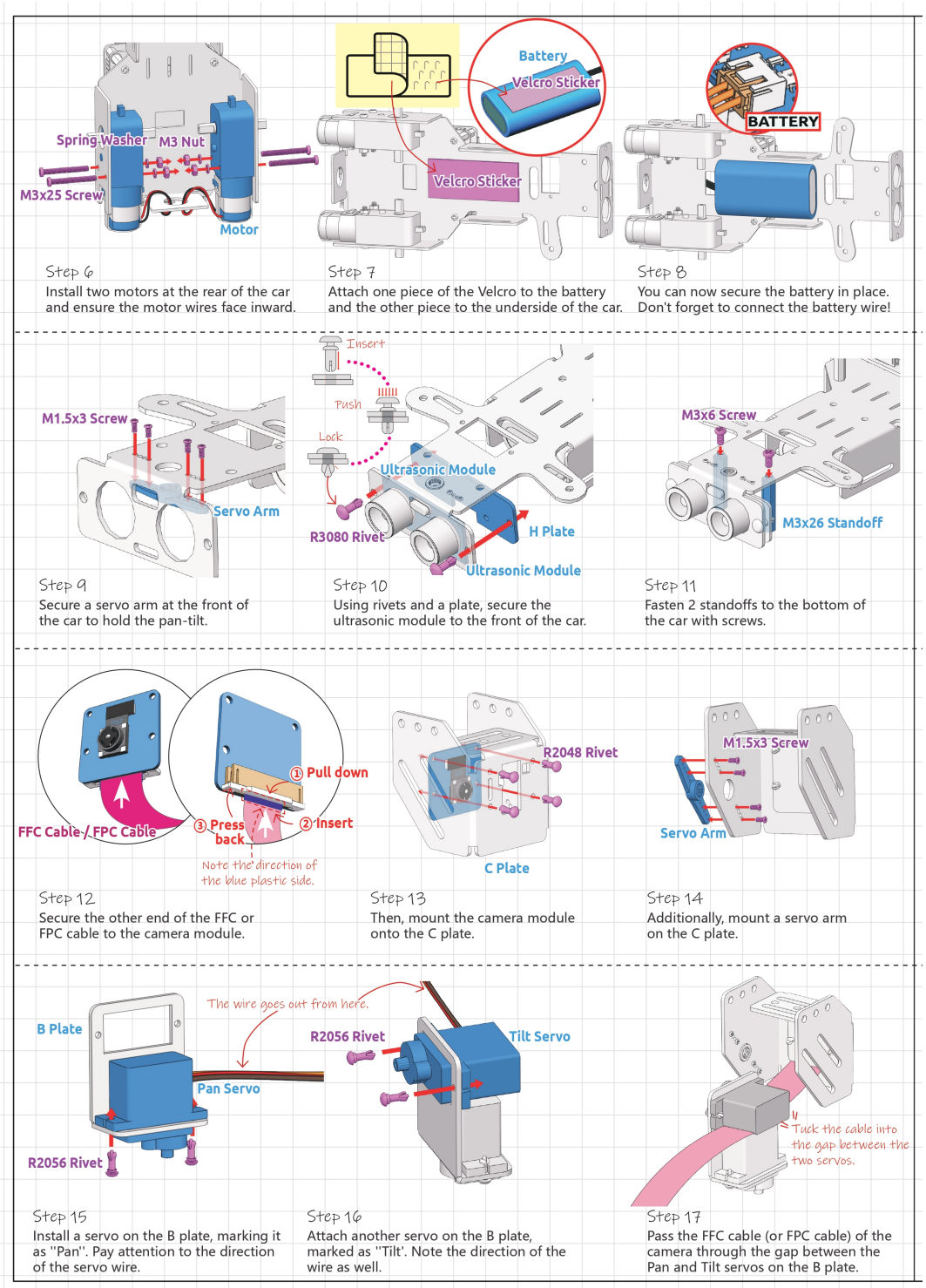


Figura 42. Ensamblaje SunFounder Picar-X Kit, Paso 6 al 17 [46].

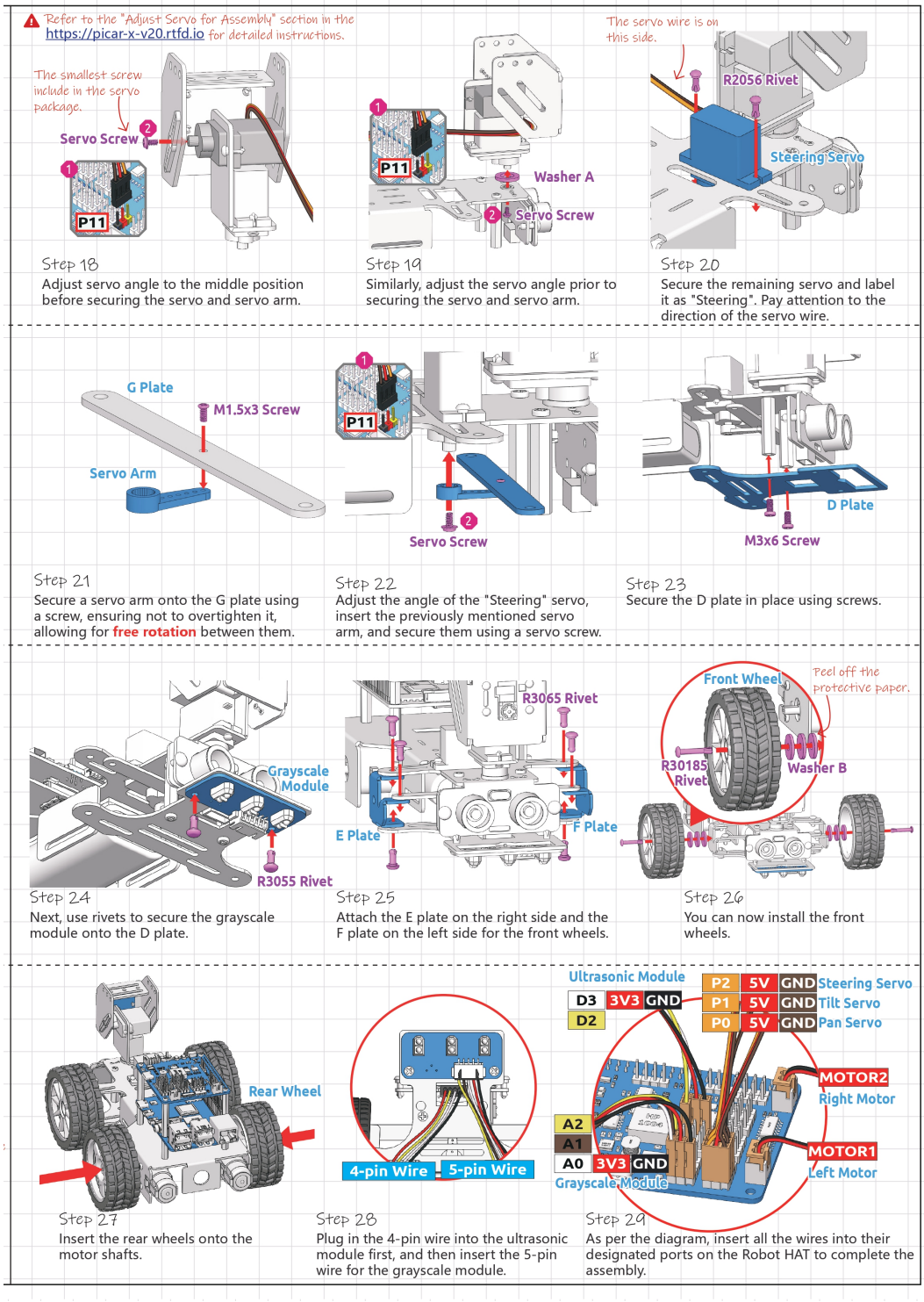


Figura 43. Ensamblaje SunFounder Picar-X Kit, Paso 18 al 29 [46].

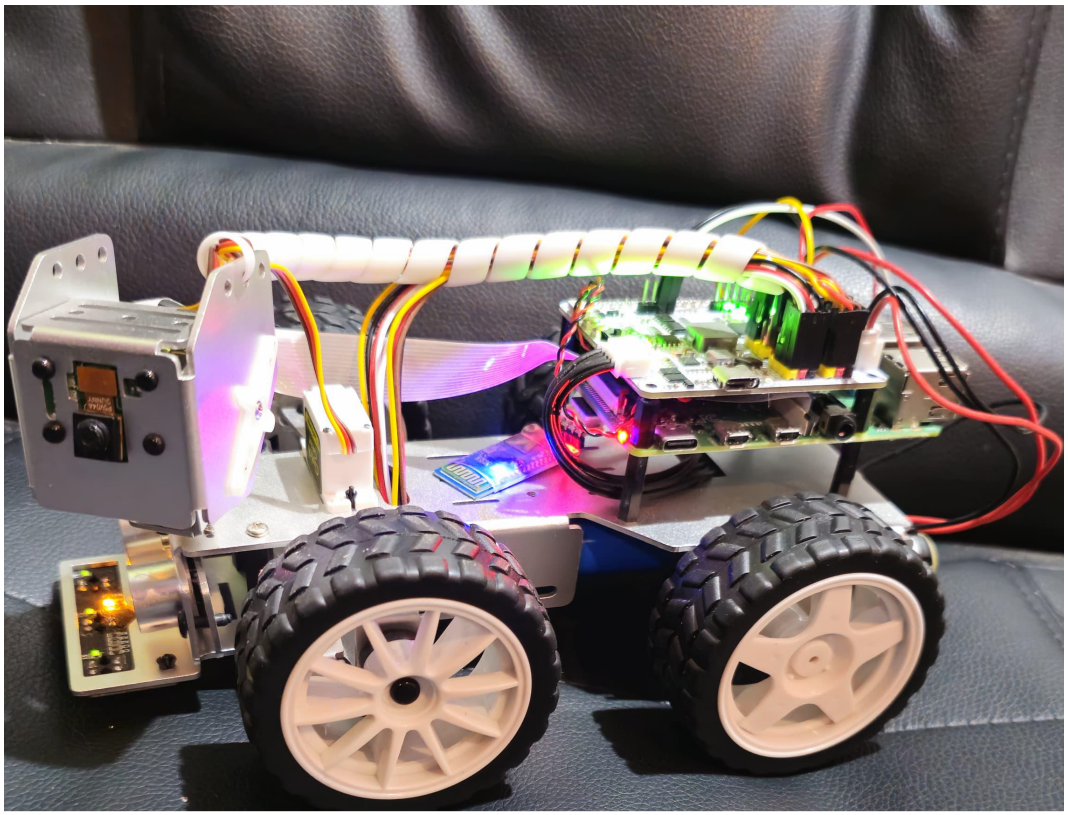


Figura 44. Picar-X Kit ensamblado, por E. Barba

ANEXO C
BLOQUES DE APP INVENTOR

Figura 45. Diagrama de bloques completo por, E. Barba, App Inventor

ANEXO D
DIAGRAMA DE FLUJO FUNCIONAMIENTO

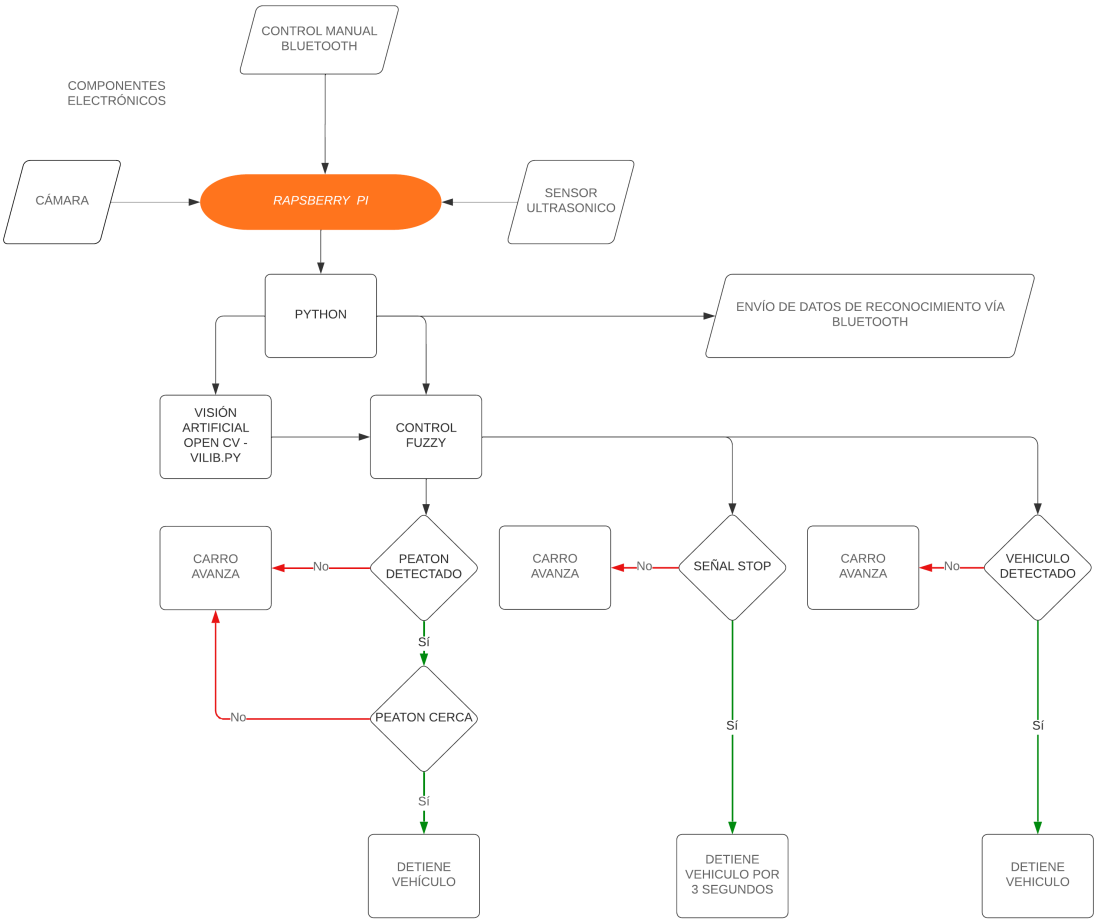


Figura 46. Diagrama de Flujo de Funcionamiento, por E. Barba

ANEXO E
ROBOT EN FUNCIONAMIENTO

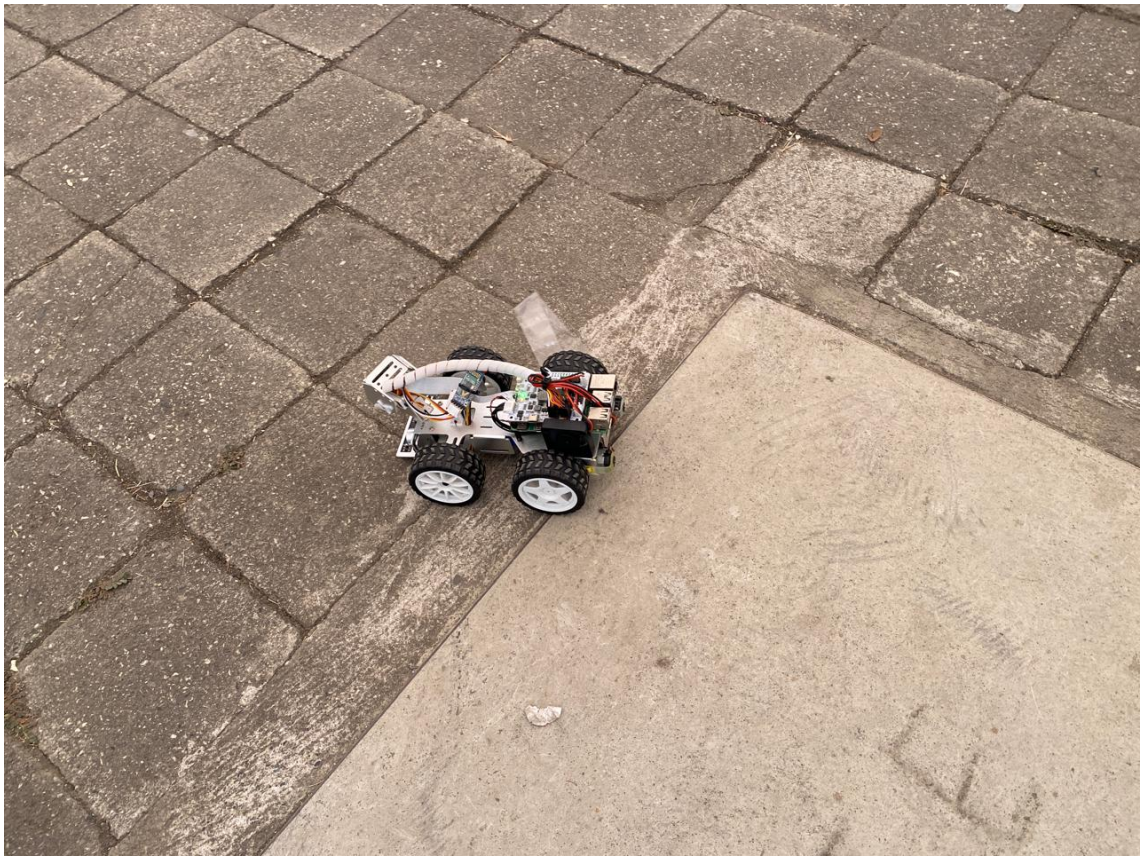


Figura 47. Inicio prueba de funcionamiento del robot, por E. Barba

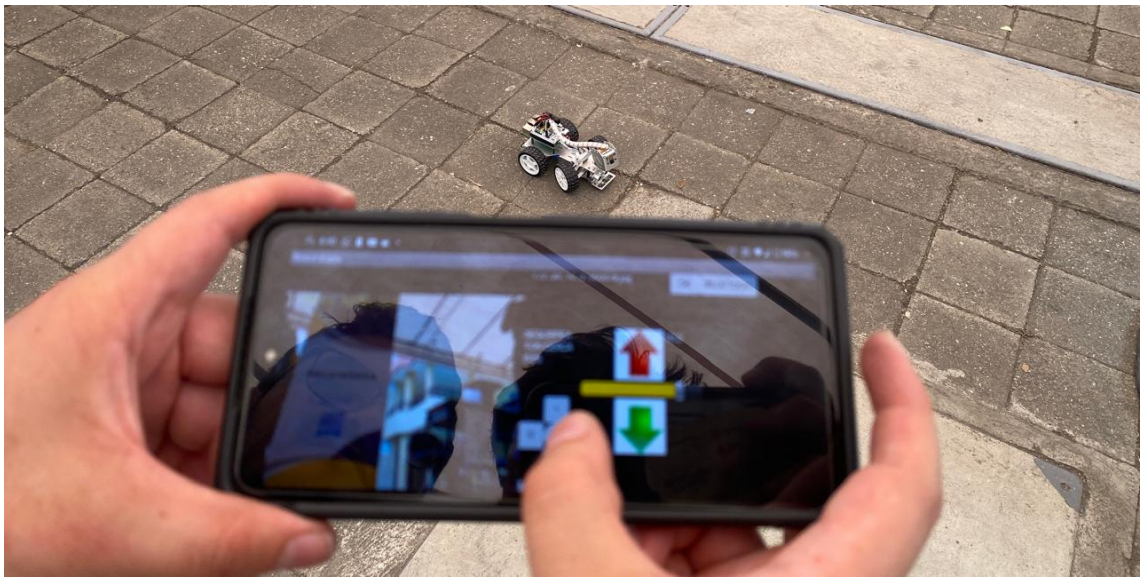


Figura 48. Prueba de Manejo desde APP, por E. Barba

Figura 49. Funcionamiento del Robot, por E. Barba

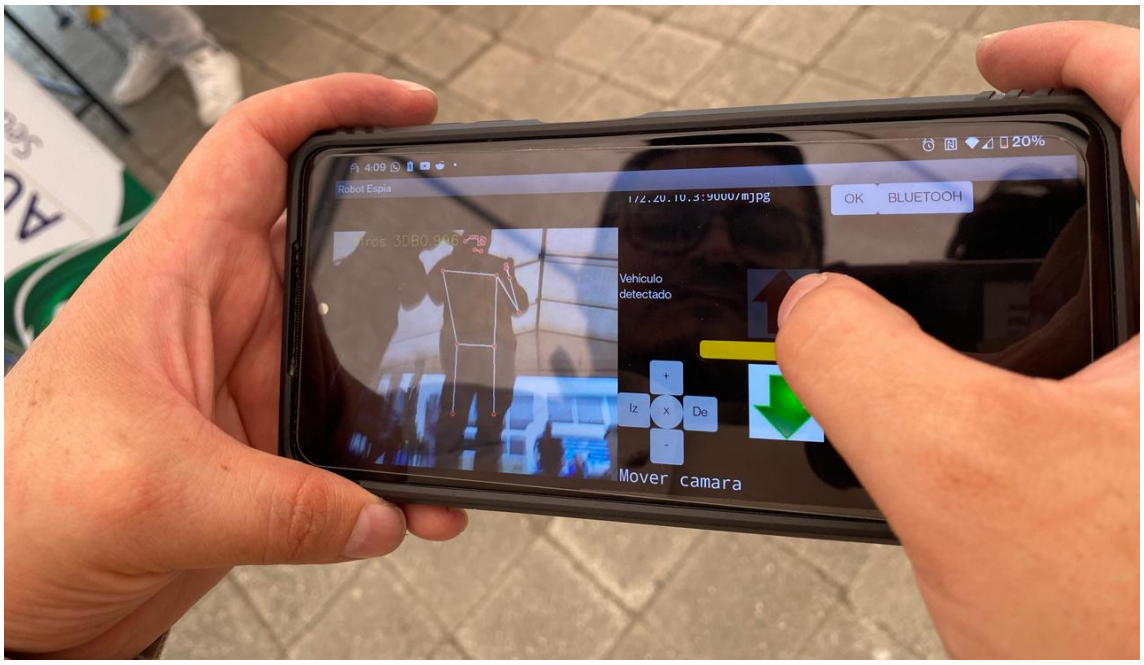


Figura 50. Mensaje en pantalla de APP, por E. Barba

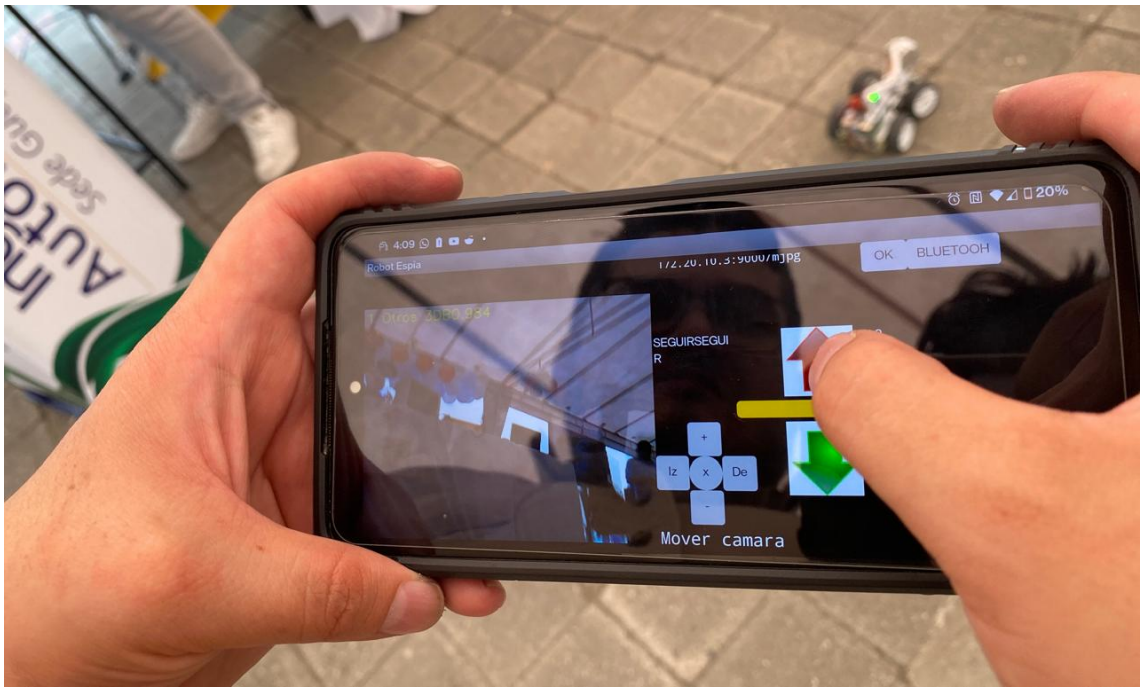


Figura 51. Mensaje de seguir adelante, por E. Barba

Figura 52. Mensaje de Peatón detectado, por E. Barba

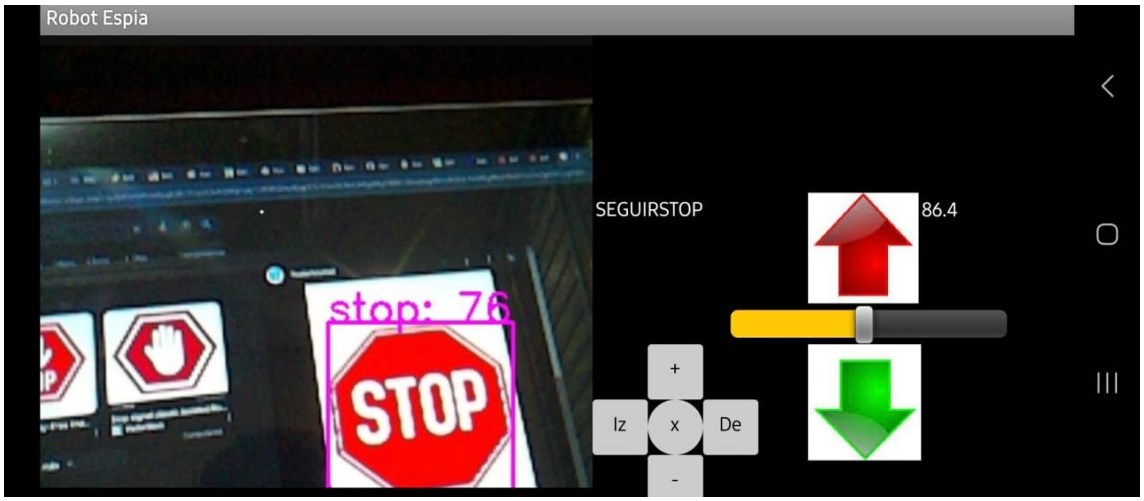


Figura 53. Detección de Señal STOP, por E. Barba