ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Sistema de monitoreo y respuesta para seguridad vehicular en tiempo real con tecnologías IoT e inteligencia artificial

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Telemática

Presentado por:

Kléber Giovanny Nuñez Sánchez

Darío Humberto Plúas Vera

GUAYAQUIL - ECUADOR

Año: 2025

DEDICATORIA

ESTUDIANTE 1

El presente proyecto lo dedico a mis padres, quienes han sido parte fundamental de mi vida, sus valores inculcados, me han permitido enfrentar con determinación cada reto presentado en mi vida académica y personal. Además me gustaría expresar mi gratitud a aquellos profesores de ESPOL por los consejos y confianza depositada en mí desde un inicio. Sus enseñanzas han sido el reflejo de lo que he logrado.

Kléber Núñez S.

ESTUDIANTE 2

El presente proyecto lo dedico a mis padres que son mi inspiración y lo mas importante de mi vida, me han enseñado valores, me han brindado la fuerza para superar cualquier adversidad a la que me enfrente y a perseguir mis sueños También quisiera agradecer a todos los profesores de ESPOL que han sido parte de este camino y me han inculcado sus conocimientos, mejorándome y superándome día a día.

Dario Plúas Vera

AGRADECIMIENTOS

ESTUDIANTE 1

Mi más sincero agradecimiento al M.Sc. Danny Torres por su guía durante mi formación académica; sus conocimientos y orientación impartido han sido parte fundamental de este proceso, permitiéndome alcanzar mis objetivos, contribuyendo de manera significativa a mi crecimiento personal y profesional.

Kléber Núñez S.

ESTUDIANTE 2

Mi más sincero agradecimiento al M.Sc. Danny Torres por sus grandes enseñanzas a lo largo de esta etapa formativa. Su apoyo y conocimiento me han hecho crecer de manera profesional y personal. Gracias por su paciencia, dedicación y por compartir generosamente su experiencia, motivándome a superar los desafíos del aprendizaje.

Dario Plúas Vera.

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Kléber Giovanny Núñez Sánchez y Darío Humberto Plúas Vera y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Firmado electrónicamente por:
KLEBER GIOVANNY NUNEZ SANCHEZ
Validar únicamente con FirmaEC

Kléber Giovanny Núñez Sánchez

Firmado electrónicamente por:
DARIO HUMBERTO PLUAS VERA
Validar únicamente con FirmaEC

Darío Humberto Plúas Vera

EVALUADORES

Maria Isabel Mera Collantes
Time Stamping Security Data

Firmado electrónicamente por:
DANNY ALFREDO TORRES MORAN
Validar únicamente con FirmaEC

María Isabel Mera Collantes, Ph.D
PROFESOR DE LA MATERIA

M.Sc. Danny Alfredo Torres Morán
PROFESOR TUTOR

RESUMEN

La situación actual que atraviesa nuestro país en cuanto a inseguridad es uno de los problemas más críticos, especialmente en los servicios de transporte. Este proyecto tiene como objetivo implementar un sistema de seguridad vehicular autónomo, capaz de detectar situaciones de robos a mano armada o intimidación verbal, con el fin de generar y enviar alertas en tiempo real a las autoridades para su intervención oportuna. El desarrollo del proyecto incluyó el uso de la metodología CAPDA desde la captura de sus datos, hasta la generación de alertas acción final, esto basado en el entrenamiento de modelos de inteligencia artificial para su correcta detección y clasificación. Se integró el microcontrolador ESP32-CAM y ESP32 para el uso de la cámara incorporada en la captura de imágenes y la integración de los módulos de micrófono y GPS para la obtención de datos complementarios. Además, las pruebas de precisión y exactitud de los modelos fueron favorables en términos de recursos y tiempos de inferencia. Los resultados mostraron el correcto envío de datos procesados junto con mensajes automatizados al chat de Telegram, manteniendo una mínima cantidad de falsos positivos. Su diseño autónomo permite que las alertas se generen y envíen sin intervención manual, mejorando la capacidad de respuesta por parte de las autoridades.

Palabras Clave: Seguridad vehicular, sistema autónomo, inteligencia artificial, ESP32-CAM, ESP32, CAPDA.

ABSTRACT

The current security situation in our country is one of the most critical problems, especially in transportation services. This project aims to implement an autonomous vehicle security system capable of detecting armed robbery or verbal intimidation, in order to generate and send real-time alerts to authorities for timely intervention. Project development included the use of the CAPDA methodology from data capture to the generation of final action alerts, based on the training of artificial intelligence models for correct detection and classification. The ESP32-CAM and ESP32 microcontrollers were integrated to use the built-in camera for image capture, and microphone and GPS modules were integrated to obtain complementary data. Furthermore, precision and accuracy tests of the models were favorable in terms of resources and inference times. The results showed the correct sending of processed data along with automated messages to the Telegram chat, keeping false positives to a minimum. Its autonomous design allows alerts to be generated and sent without manual intervention, improving the response capacity of authorities.

Keywords: Vehicle safety, autonomous system, artificial intelligence, ESP32-CAM, ESP32, CAPDA.

Índice general

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
IoT	Internet de las Cosas
IA	Inteligencia Artificial
GPS	Sistema de Posicionamiento Global
NLP	Procesamiento de Lenguaje Natural
HTTPS	Protocolo Seguro de Transferencia de Hipertexto
JWT	JSON WebToken
TLS	Seguridad de la capa de transporte
API	Interfaz de Programación de Aplicaciones
YOLO	You Only Look Once
ECU 911	Servicio Integrado de Seguridad
UDP	Protocolo de datagramas de usuariol
NTLK	Natural Language Toolkit
GDPR	Reglamento General de Protección de Datos
MEMS	Sistemas Microelectromecánicos
I2S	Protocolo digital especializado en la transmisión de audio
UART	Protocolo de comunicación serie asíncrono para transmisión de datos
ONNX	Open Neutral Network Exchange
CPU	Central Processing Unit
GPU	Graphics Processing Unit
SVM	Maquinas de Soporte Vectorial
CAPDA	Captura – Análisis – Predicción – Decisión – Acción
CPS	Cyber-Physical Systems
AUC	Área bajo la curva
FIFO	First-In, First-Out
WER	Word Error Rate

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

Actualmente, Ecuador atraviesa una de las peores crisis de inseguridad de las últimas décadas, afectando a varios ciudadanos en sus actividades cotidianas. Según datos oficiales, en 2023 se registraron 7.878 muertes violentas, lo que corresponde a una tasa de 46,5 homicidios intencionales por cada 100.000 habitantes, consolidando al país como uno de los más inseguros del continente (Montalvo, 2023). En este contexto de inseguridad, se ha visto afectado directamente el transporte público y privado, evidenciando que las aplicaciones móviles, tales como, Uber, Didi, Indrive, entre otras, utilizadas para el servicio de transporte presentan vulnerabilidades que comprometen la seguridad de los usuarios durante su movilización. La Brigada Anticriminal (BAC) de la Policía Nacional registra 364 denuncias de robos a taxistas en 2023 en la Zona 8, entre ellos 334 conductores sin aplicación. Al menos 30 casos corresponden a taxistas informales y usuarios de aplicaciones de transporte. De ellos, 15 denuncias están relacionadas con conductores asaltados mediante la aplicación InDrive, 13 con Uber y 2 con Didi (Primicias, 2023).

En el contexto de seguridad vehicular, la implementación de un sistema antirrobo es considerado de suma importancia para poder disminuir incidentes delictivos, por lo tanto, se espera que, en caso de que durante la movilización del vehículo con pasajeros se presente una situación de riesgo, como un robo a mano armada, el sistema antirrobo emita una llamada de alerta informando la situación (Rojas et al., 2017). Con el objetivo de mejorar la seguridad de transporte, la Agencia Nacional de Tránsito del Ecuador (ANT) implementó el plan Transporte Seguro desde el año 2013 en todas las unidades de transporte público inscritas en su base de datos. La coordinación operativa quedó a cargo del ECU 911. En un inicio, los conductores de buses y taxis, afirmaron que lograron recuperar los niveles de confianza en la ciudadanía que utilizaba este servicio, lo que evidentemente redundó en el mejoramiento de sus ingresos económicos. Sin embargo, con el tiempo se evidenciaron falencias en el funcionamiento y atención de emergencias, asegurando que aunque los delitos fueron grabados, no se logró responder oportunamente (Corrales Barragán, 2020).

Este proyecto consiste en mejorar el nivel de respuesta automatizada ante un incidente delictivo,

integrando tecnologías IoT e inteligencia artificial para el reconocimiento de armas blancas y voz, asegurando el envío de datos automático a un bot de telegram, donde se recibirá los datos del vehículo previamente registrado en tiempo real, incluyendo imágenes donde se pueda observar al agresor y coordenadas GPS con actualizaciones constantes. De este modo, se pretende mejorar la seguridad vehicular al incorporar cámaras y micrófono con sistema de análisis inteligente asistido por computadora capaces de detectar comportamientos sospechosos sin necesidad de intervención manual, permitiendo operar de manera discreta y oportuna.

Como resultado, se espera que las autoridades puedan operar de manera eficiente gracias a las evidencias recolectadas durante el incidente, las cuales servirán como prueba para el inicio de las investigaciones pertinentes, facilitando la identificación y captura de los responsables. De esta manera, se busca garantizar la seguridad de los usuarios que utilizan medios de transporte convencionales o por aplicaciones móviles.

1.1. Descripción del problema

La inseguridad en Ecuador, es un problema emergente en la actualidad. Según datos proporcionados por el Ministerio de Transporte y compartidos por Napoleón Cabrera, presidente de la Confederación de Transporte Terrestre del Ecuador, existen 25 rutas viales catalogadas como zonas críticas en cuanto a seguridad. Estas vías, en un 90 % se dirigen hacia la costa del país, especialmente hacia Guayas, Manabí, Los Ríos, El Oro y Santo Domingo, donde se concentran los mayores niveles de criminalidad (Directo, 2025).

Ante esta situación, se han implementado iniciativas para fortalecer la seguridad en vehículos con el uso de cámaras y botones de pánico con grabación las 24 horas del día, con el fin de brindar seguridad a los usuarios y conductores. El proyecto Transporte Seguro se ejecutó entre el año 2016 y 2018 en la ciudad de Quito, el cual pretendía aumentar la confianza en la movilización, sin embargo, el funcionamiento a largo plazo no fue el esperado debido a que el sistema presentó errores en la comunicación y coordinación con la atención de emergencias (Corrales Barragán, 2020). Entre los factores que afectaron la deficiencia en el sistema de seguridad están, las fallas en la comunicación con los servicios de emergencia, limitaciones presupuestarias del Estado, falta de actualización e innovación tecnológica, altos costos de conectividad y mantenimiento (Corrales Barragán, 2020).

Muchos de estos sistemas de videovigilancia operan de forma aislada, sin conexión al ECU 911. Los dirigentes del transporte urbano se encuentran a la espera de formar parte del proceso Transporte Seguro, aunque algunas unidades cuentan con cámaras, incluyendo un taller de instalación y mantenimiento permanente, desde 2019 no reciben asistencia por parte de este centro de control (Diario Correo, 2022).

Por otro lado, la falta de soluciones para el monitoreo en tiempo real representa limitaciones significativas en la gestión de flotas, logística y seguridad vehicular. Las soluciones existentes en el mercado suelen ser costosas, difícil de implementar y no cuentan con fuente de energía sostenible. Además, la ausencia de un sistema que brinde geolocalización precisa mediante GPS ha generado una importante brecha en el seguimiento del incidente en tiempo real por parte de las autoridades (Laínez Apolinario & Navarrete Buste, 2024).

Basado en este contexto, nuestro trabajo se enfoca en la mejora de recopilación de datos necesarios durante un asalto a mano armada dentro del vehículo, automatizando el envío de evidencias mediante la implementación de tecnologías IoT e inteligencia artificial, buscando resolver los retos existentes en la intervención inmediata ante estas situaciones.

1.2. Justificación del problema

La seguridad en los vehículos no es un problema reciente, ya que anteriormente se han implementado sistemas de monitoreo mediante cámaras con un botón de pánico, el cual tuvo limitaciones importantes de comunicación con los servicios de emergencia, dificultando una respuesta oportuna (Corrales Barragán, 2020). Con el fin superar estas deficiencias, se plantea integrar nuevas tecnologías al sistema de monitoreo, permitiendo automatizar y precisar datos relevantes para el aumento de confianza en los usuarios.

El incremento de asaltos a mano armada dentro de vehículos, crea la necesidad de implementar sistemas seguridad que permita alertar a las autoridades en tiempo real, con la recopilación de evidencia de forma automatizada. La integración de nuevas tecnologías, como la inteligencia artificial (IA) y el internet de las cosas (IoT) representa una evolución en la actualidad frente a las soluciones tradicionales. Este consiste en el uso de una cámara como medio de visualización para la detección de objetos peligrosos como armas blancas, y un micrófono para el reconocimiento de palabras o frases amenazantes, junto con la transmisión inmediata de evidencia visual y geolocalización del vechículo. Para garantizar un respuesta rápida y eficiente, nuestro proyecto emplea un modelo de aprendizaje automático previamente entrenado, el cual permite la clasificación de imágenes y audio en tiempo real con alta precisión, capaz de emitir alertas con menor tiempo de latencia.

Además de automatizar la recopilación de evidencia en tiempo real, esta solución busca mejorar la eficiencia en la toma de decisiones e intervención de las entidades responsables de la seguridad. Su implementación representa un impacto comercial de alto potencial debido a su avance tecnológico priorizando la seguridad emergente, siendo este sistema innovador en el mercado, llevando una ventaja significativa respecto a los sistemas tradicionales de seguridad vehicular en Ecuador. Con esta relación, las cooperativas podrán reforzar su reputación atrayendo más cliente y minimizando

el riesgo de robos, convirtiéndose en una inversión estratégica viable y sostenible.

1.3. Objetivos

Objetivo general

Implementar un sistema de seguridad vehicular inteligente capaz de detectar situaciones de riesgo en tiempo real mediante tecnología IoT e inteligencia artificial con el fin de generar alertas automáticas y facilitar la intervención inmediata de las autoridades.

Objetivos específicos

- Desarrollar un sistema de detección en tiempo real de armas blancas capaz de capturar y enviar imágenes a un servidor de procesamiento basado en IA y que ante la detección de un incidente, envíe automáticamente evidencia visual, datos y ubicación del vehículo a un bot de Telegram monitoreado por las autoridades.
- Desarrollar un sistema de detección de frases amenazantes en tiempo real, enviando los audios captados a un servidor de procesamiento basado en IA y que ante la detección de una amenaza, envíe automáticamente una alerta con los datos del vehículo, ubicación y evidencia visual a un bot de Telegram monitoreado por las autoridades.
- Diseñar un sistema basado en eventos que, ante un posible acto delictivo, envíe automáticamente información del vehículo, tales como ubicación en tiempo real (coordenadas GPS), placa, modelo, año, propietario y antecedentes, con el fin de facilitar una intervención oportuna e inmediata de las autoridades.

1.4. Alcance

El sistema de seguridad vehicular contempla una solución orientada principalmente a vehículos de transporte público o privado, como taxis o servicios por aplicación (Uber, InDriver), en el contexto de la creciente inseguridad en el territorio ecuatoriano.

El sistema está diseñado para operar de forma autónoma, dado que, en situaciones de peligro, el conductor no debe verse obligado a realizar acciones que puedan poner en riesgo su integridad. Por esta razón, no se requiere intervención humana en los procesos que ejecuta el sistema.

El servidor de procesamiento basado en IA analizará los datos recopilados por el microcontrolador y, en caso de detectar objetos peligrosos (como armas blancas) o frases asociadas a situaciones delictivas, se generará una alerta automática. Esta alerta incluirá datos relevantes del vehículo (propietario, modelo, marca y color), así como imágenes y audios de voz, que serán enviadas a la plataforma de Telegram para notificar a las autoridades correspondientes.

1.5. Limitaciones

Este proyecto presenta varias limitaciones tanto a nivel tecnológico como operativo. Una de las principales es la capacidad de procesamiento limitada de los microcontroladores como la ESP32-CAM, los cuales no cuentan con los recursos suficientes para ejecutar modelos de inteligencia artificial necesarios para el funcionamiento del sistema. Por esta razón, se optó por delegar el procesamiento de imágenes y audio a un servidor de procesamiento de imágenes y audios, que dispone de las librerías y la capacidad computacional requeridas para llevar a cabo dichas tareas.

Otra limitante es la dependencia directa de la conectividad a internet o diferentes métodos para comunicarse con el servidor , ya que la transmisión de datos desde el microcontrolador es fundamental para el análisis y la toma de decisiones. En zonas con baja cobertura o inestabilidad en la conexión, esto podría ocasionar retrasos en los procesos de detección y notificación.

Una de las funcionalidades del sistema se basa en detectar solo objetos que se asemejen a armas blancas, por lo que no es capaz de diferenciar con certeza entre armas reales o réplicas fabricadas con plástico o goma. Esto puede generar falsos positivos, haciendo que el sistema detecte objetos similares como utilería o juguetes.

La detección de lenguaje ofensivo se ve limitada por la jerga y modismo del Ecuador. Muchas expresiones pueden sonar agresivas, pero no lo son para diversos grupos sociales. El uso irónico o ambiguo del lenguaje puede generar errores en el modelo entrenado para su detección y análisis.

Finalmente, aunque el sistema está diseñado para enviar alertas automáticas de forma inmediata, la respuesta efectiva ante dichas alertas dependerá de actores externos, como las autoridades competentes. Es decir, el sistema no tiene control sobre la actuación posterior de las instituciones receptoras de la información.

1.6. Estado de Arte

En los últimos años, el desarrollo de sistemas de seguridad vehicular han evolucionado integrando tecnologías inteligentes capaces de anticipar, detectar y reaccionar ante catástrofes.Esta transformación es posible gracias al avance del IoT e IA. (Goyal et al., 2022).

Los sistemas de seguridad actuales cuya función en seguridad para transporte publico y privado como localización GPS, sensores, botones de pánico son grandes avances de la tecnología (Crisgar et al., 2021). No obstante, muchos de estos mecanismos requieren intervención humana para activar funciones críticas o enviar alertas, lo cual representa una limitación importante en situaciones en las que el conductor se encuentra bajo amenaza directa. Debido a esta restricción, el desarrollo de sistemas autónomos, que no dependan de la acción humana para operar, se vuelve esencial en contextos donde la seguridad está comprometida. Dentro de la detección de objetos existen

diversos modelos o herramientas web que han demostrado gran eficacia en la identificación en tiempo real, permitiendo entrenar de forma personalizada el reconocimiento de imágenes de manera sencilla, (Chitravanshi et al., 2024). Por otro lado, el reconocimiento de voz y el procesamiento de lenguaje natural (NLP) ha sido aplicado para la detección de agresiones verbales y amenazas, (García Torres et al., 2020). Modelos como Whisper, desarrollados por OpenAI, permiten transcribir audio a texto con alta precisión, lo que posibilita aplicar análisis semánticos sobre el contenido utilizando herramientas como NLTK o Transformers. Estas tecnologías pueden identificar frases o palabras clave relacionadas con violencia, insultos o lenguaje agresivo, (Jemai et al., 2021).

En cuanto a la seguridad con IoT en los vehículos, se han desarrollado microcontroladores capaces de interactuar con cámaras, micrófono o sensores para recopilar información relevante desde el vehículo.Estos dispositivos, debido a su bajo consumo energético y reducido costo, son ideales para este tipo de sistemas. Sin embargo, su capacidad limitada de procesamiento ha llevado al uso de un servidor externo de procesamiento para ejecutar tareas más exigentes, como el análisis de imágenes, la transcripción de voz y la clasificación semántica de frases. Esta arquitectura distribuida permite mantener la funcionalidad del sistema sin comprometer su autonomía ni su velocidad de respuesta, (Husni & Hasibuan, 2018).

1.7. Marco teórico

Internet de las Cosas (IoT)

El Internet de las Cosas ha sido definido como la conexión global de dispositivos electrónicos que están embebidos con diversos tipos de sensores que permiten recopilar y transmitir datos a través de redes. (Salama et al., 2023).

En el ámbito de la seguridad vehicular, el IoT permite el desarrollo de sistemas automatizados capaces de monitorear el entorno, generando alertas con poca o nula intervención humana, (Goyal et al., 2022).

Inteligencia artificial (IA)

La inteligencia artificial es un campo centrado en la creación de máquinas capaces de imitar comportamientos humanos como actuar, razonar y tomar decisiones, (Kumar, 2013a).

Dentro de la IA, se encuentra el aprendizaje automático (machine learning) que permite que un sistema pueda aprender a través de la experiencia, (Alpaydin, 2021). Es decir, algoritmos capaces de identificar patrones en grandes volúmenes de datos. En este proyecto, esta tecnología se utiliza para distinguir entre frases agresivas y no agresivas, ademas para la clasificación de imágenes.

Vision por computadora y Deteccion de objetos

La visión por computadora y la detección de objetos permiten que un sistema interprete y comprenda imágenes o secuencias de video, simulando la percepción visual humana, (Zhao et al., 2019). Existen diversas soluciones para identificar y segmentar objetos. Una de ellas es Teachable Machine, una herramienta desarrollada por Google que permite entrenar modelos personalizados de clasificación de imágenes directamente en la web y exportarlos para su integración en el entorno deseado, (Google, 2019).

Procesamiento del lenguaje natural (NLP)

El procesamiento del lenguaje natural (NLP, por sus siglas en inglés) es una rama de las ciencias computacionales enfocada en la interacción entre las máquinas y el lenguaje humano. Combina aprendizaje automático, estadística y lingüística para analizar estructuras léxicas con el fin de detectar sentimientos, intenciones, generar texto, entre otros (Kumar, 2013b).

Automatización de envio de alertas

La automatización de alertas es un proceso mediante el cual un sistema detecta una anomalía y, sin intervención humana, genera una respuesta. Esta puede ser informativa —como el valor de humedad de un sensor— o puede activar una función, como encender una bomba si la humedad lo requiere. (Valinejadshoubi et al., 2021).

Seguridad y Privacidad

La recopilación, procesamiento y transmisión de datos como imágenes, audios o información personal plantea desafíos importantes en términos de seguridad y privacidad. A nivel técnico, la seguridad implica el uso de medidas como cifrado, autenticación y control de acceso. En el ámbito legal, existen normativas como el Reglamento a Ley Orgánica de Protección de Datos Personales que establecen principios para el tratamiento lícito de la información, incluyendo la protección del almacenamiento,el derecho de acceso o eliminación por parte del titular de los datos, (Parlamento, 2016).

Pandas

Pandas es una biblioteca de código abierto de Python que sirve para analizar y manipular datos. Su estructura permite trabajar con datos de tabla para la filtración, agrupación, eliminación de valores nulos, comúnmente usados desde archivo como CSV, Excel, SQL, JSON,etc, (McKinney et al., 2011).

UDP

User Datagram Protocol (UDP) es un protocolo de comunicación sin conexión para transportar paquetes a través de redes. Es usado por su velocidad y eficiencia en aplicaciones donde la latencia es crítica, como audios o videos en tiempo real, (AL-Dhief et al., 2018).

HTTPS/TLS

Hypertect Transfer Protocol Secure (HTTPS) es una versión segura de HTTP combinando con tecnología TLS (Transport Layer Security)/SSL (Secure Socket Layer); toda la información es cifrada y protegida contra accesos no autorizados. Aunque sean interceptados por un tercero, no podrá manipularlos sin la clave de cifrado, (Fielding et al., 2022).

JWT

JWT (JSON Web Token) es un estándar que define un formato seguro para la transmisión de información entre partes como un objeto JSON. Estos datos están firmados digitalmente mediante tokens, lo que garantiza su autenticidad, (Jones et al., 2015).

Teachable Machine

Teachable Machine es una herramienta creada por Google cuya funcionalidad es crear modelos de aprendizaje automático de manera ágil y sencilla. Funciona bajo aprendizaje por transferencia, donde se utilizan modelos previamente entrenados y se ajustan a tareas mediante carga de datos personalizados como imágenes, (Machine, 2024).

Tensorflow

TensorFlow es un framework desarrollado por Google para el desarrollo y ejecución de modelos de aprendizaje automático. Tiene varias aplicaciones como visión por computadora, procesamiento de lenguaje natural, entre otros, (TensorFlow Developers, 2025).

YOLOv5

YOLOv5 es un modelo de detección de objetos en tiempo real basado en redes neuronales. Su arquitectura permite analizar una imagen completa de manera extremadamente rápida, recomendada para vigilancia, conteo de personas o vehículos, seguridad, entre otros, (Jani et al., 2023).

Matriz de confusión

Permite visualizar y evaluar el desempeño de un modelo entrenado mediante una tabla la precisión obtenida por cada clase, respecto a su clasificación correcta e incorrecta. (Google, 2019).

Época

Se define cuando el modelo de preparación ha procesado todas las muestras del conjunto de datos de preparación una vez, se dice que se ha completado una época. (Google, 2019).

Medición de rendimiento

Son herramientas visuales donde se evalúa el comportamiento del modelo entrenado bajo diferentes condiciones:

- **Curva de precisión frente al umbral de confianza:** Permite identificar el punto óptimo de operación, es decir, el equilibrio entre minimizar los posibles falsos positivos y mantener un buen nivel de detección.
- **Precisión y pérdida por época:** Se refiere a la mejora progresiva del modelo y su reducción de errores según el entrenamiento.
- **Recall:** Es una métrica que mide la capacidad del modelo para identificar correctamente las instancias relevantes.
- **F1-score:** Es una métrica que combina la precisión y recall, donde existe un desbalance entre clases.

Whisper IA

Whisper es un modelo de inteligencia artificial desarrollado por OpenAI para la transcripción automática de audio a texto. Este modelo cubre un gran volumen de datos de audio multilingüe , lo que le permite reconocer y transcribir en diferentes idiomas, ademas de tener diferentes versiones según el requerimiento de recursos y precisión, (Spiller et al., 2023).

Vosk

Vosk es una librería de código abierto para el reconocimiento de voz desarrollada por Alpha Cephei que permite convertir audio en texto en tiempo real. Incluye modelos preentrenados que mejoran la precisión incluso en presencia de ruido o acentos, (Soni, 2025).

Scikit-Learn

Scikit-learn es una biblioteca de código abierto para aprendizaje automático en Python. Proporciona herramientas para clasificación, regresión, validación cruzada, utilizando algoritmos como SVM,regresión logística, entre otros. Es utilizado por su facilidad de uso y su fácil integración con otras bibliotecas como Numpy y pandas, que son eficaces para analizar datos, predecir tendencias, segmentar clientes, detectar anomalías, etc., (Kramer, 2016).

RoBERTa

RoBERTa (Robustly Optimized BERT Approach) es un modelo de lenguaje desarrollado por Facebook AI. Está basado en la arquitectura de transformer que ha revolucionado el NPL (procesamiento de lenguaje natural) gracias al mecanismo de atención y auto-atención, que permite que el modelo entienda el contexto de cada palabra según la oración, (Pavlov & Mirceva, 2022)

Flask

Flask es un microframework para Python que permite desarrollar aplicaciones web de manera sencilla y rápida. A pesar de no tener una estructura rígida, es altamente escalable, además de que incluye un servidor de desarrollo y es flexible con bibliotecas externas como base de datos, autenticación y APIs, (Flask, 2010).

INMP441

Es un módulo de micrófono omnidireccional de interfaz I2S MEMS (Sistemas Microelectromecánicos), utilizado para capturar audio digital de bajo consumo, (Shin, 2020).

ESP32+ESP32CAM

El ESP32 es un microcontrolador de buen rendimiento que cuenta con conectividad Wi-Fi y Bluetooth integrada, tiene soporte para interfaces como I2S,I2C, UART, lo que lo hace efectivo para proyectos IoT y sistemas embebidos, (Hercog et al., 2023).

NEO6m + ANT-555

El NEO6m es un módulo GPS de alto rendimiento que proporciona datos de localización en tiempo real. El módulo se comunica por UART y entrega mensajes que contienen información como latitud, altitud, longitud, cantidad de satélites,entre otros, (Moumen et al., 2023).

Telegram Bot

Telegram es una aplicación de mensajería instantánea que permite la comunicación de usuarios de forma rápida y segura. Los bots son pequeñas aplicaciones que se ejecutan completamente dentro de la aplicación Telegram. Permiten a los usuarios interactuar con los bots a través de interfaces flexibles que pueden soportar cualquier tipo de tarea o servicio. (Telegram, 2025).

Node-Red

Node-Red es una herramienta de desarrollo basada en web para la programacion de flujos de datos. Fue desarrollada por IBM y permite conectar dispositivos, APIs o servicios mediante un sistema de nodos, (Medina-Pérez et al., 2021).

Metodología CAPDA

Describe un flujo lógico para el funcionamiento de proyectos que integran IoT e Inteligencia Artificial. Aunque el término CAPDA no corresponde a un estándar técnico oficial reconocido por organismos como IEEE, ISO o ACM, su estructura deriva directamente de modelos ampliamente documentados en la literatura científica sobre IoT, sistemas ciberfísicos (CPS) y arquitecturas event-driven.

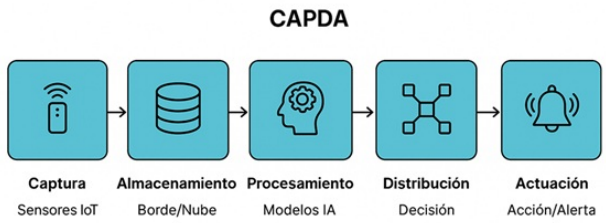


Figura 1.1: Metodología CAPDA

Algunos modelos equivalentes se encuentran en distintas fuentes académicas y técnicas, (Gubbi et al., 2013) describen un flujo para IoT compuesto por Data Acquisition, Data Transmission, Data Analysis, Decision Making y Actuation, que coincide funcionalmente con CAPDA.

En el ámbito de CPS, (Lee, 2008) presenta una arquitectura en capas formada por Perception Layer, Network Layer, Processing Layer y Application Layer, que sigue la misma lógica de captura, análisis y acción.

En consecuencia, aunque CAPDA no sea un modelo estandarizado, es una adaptación útil para describir el ciclo de operación de sistemas inteligentes con IoT e IA.

Teoría de colas FIFO

Es un sistema de espera con `c` servidores, donde los tiempos entre arribos y de servicio tienen distribución exponencial, la fuente y capacidad del sistema son infinitas, y la disciplina de servicio es, el primero que llega es el primero que se atiende. (Peraza Siqueiros, 2006)

CAPÍTULO 2

2. METODOLOGÍA

La metodología del proyecto se inspira en modelos sistemas ciberfísicos y adaptada a IoT + IA, conocido como CAPDA para el desarrollo de un sistema autónomo e inteligente de seguridad vehicular. Para ello, se utilizó un microcontrolador ESP32-CAM, equipado con una batería recargable que permite su operación independiente del vehículo, un módulo GPS para la obtención de coordenadas en tiempo real y un micrófono digital para la captura de audio ambiental.

El desarrollo metodológico se basa en tres fases principales:

1. Captura de datos.

2. Procesamiento y protección.

3. Alertas automatizadas.

El detalle del funcionamiento técnico y flujo de datos se presenta en el capítulo 3 correspondiente al diseño e implementación del sistema.

2.1. Metodología para el diseño

Comparación de modelos para transcripción de voz

Para la transcripción de audio, se evaluaron dos modelos de código abierto: Whisper, desarrollado por OpenAI, y Vosk una solución ligera para dispositivo de bajo consumo.

Métrica	**WhisperAI**	**Vosk**
Tasa de error	6-8 %	10-20 %
Requerimientos de RAM	5 GB	100 MB
Velocidad de inferencia	12 s	1.4 s
Tamaño del modelo	1.4 GB	50 MB
Idiomas soportados	95	20

Tabla 2.1: Comparativa de características entre WhisperAI y Vosk

En la tabla 2.1 se pueden observar sus comparativas técnicas en donde Whisper destaca por su amplia precisión de transcripción, a pesar de estar en ambientes ruidosos. Este modelo se puede usar en diferentes tamaños, por lo que se seleccionó la versión medium por su soporte nativo al español y su capacidad de manejar audios complejos. Por otro lado, Vosk ofrece una implementación ligera, rápida y fácil de integrar. Sin embargo, la precisión en la transcripción de frases en español, además en presencia de ruido, fue relativamente menor al otro modelo utilizado. Esto representa una gran desventaja en causar falsos positivos o negativos al momento del análisis que necesita una interpretación correcta ante posibles amenazas verbales.

Parámetros definidos en ambas herramientas de transcripción de voz

Para la evaluación de las herramientas de transcripción de audio, se definieron varios parámetros de prueba que nos ayudaron a elegir el más óptimo para el sistema. En este caso, usaremos varios audios con diferentes duraciones para observar la variación en el tiempo de respuesta. Además, se registró el consumo de RAM durante la ejecución, considerando los recursos necesarios.

Duración	**Tiempo**	**RAM**	**N° palabras**	**% correcto**
15	16.52 s	3.24 GB	21	99 %
20	15.59 s	3.21 GB	20	97 %
30	20.25 s	3.28 GB	39	98 %
40	24.26 s	3.12 GB	63	98 %
50	27.51 s	3.48 GB	87	95 %

Tabla 2.2: Parámetros de WhisperAI

En la tabla 2.2 se evidencia la alta precisión alcanzada por Whisper en la transcripción de voz. Si bien el tiempo de respuesta y el consumo de memoria RAM resultan relativamente elevados, el modelo mantiene un rendimiento eficaz y adecuado para los objetivos planteados en nuestro proyecto.

Duración	Tiempo	RAM	N° palabras	% correcto
15	1.25 s s	0.25 GB	21	80.95 %
20	1.20 s	0.22 GB	20	80.00 %
30	1.9 s	0.21 GB	39	76.92 %
40	2.36 s	0.23 GB	61	80.33 %
50	3.17 s	0.23 GB	82	77.73 %

Tabla 2.3: Parámetros de Vosk

En la tabla 2.3 se presentan los resultados obtenidos con el modelo Vosk. Aunque el tiempo de respuesta y el consumo de memoria RAM son considerablemente bajos, lo que lo convierte en una alternativa ligera y eficiente en términos de recursos, el porcentaje de acierto en la transcripción no alcanza un nivel suficientemente alto como para ser considerado viable dentro de los requerimientos de nuestro proyecto.

Comparación de modelos para la deteccion de objetos

Para la detección de objetos, se evaluaron dos modelos de inteligencia artificial: Teachable Machine, herramienta desarrollada por Google para el entrenamiento de un modelo basado en TensorFlow y YOLOv5 creado en el año 2016. El objetivo es identificar el modelo más eficiente en términos de precisión, velocidad y usabilidad en aplicaciones de la vida real, utilizando parámetros similares para su comparación.

Métrica	TensorFlow	YOLOv5
Precisión (arma)	100 %	98.5 %
Sensibilidad (Recall)	98.67 %	98.5 %
Entrenamiento (min)	10	45
Inferencia (ms/img)	110	400
Tamaño modelo (MB)	2.33	26.8
Formato	Keras (.h5)	Pytorch (.onnx)
Velocidad de entrenamiento	Rápida con CPU	Rápida con GPU

Tabla 2.4: Comparativa entre modelos de IA para detección de armas blancas

Basado en la comparación presentada, ambos modelos ofrecen ventajas particulares según el contexto de aplicación. Teachable Machine, se caracteriza por su facilidad de uso, precisión, bajo consumo de recursos computacionales e integración directa con TensorFlow, ideal para el

procesamiento de sus datos, donde no se necesita la detección precisa de la ubicación del arma, ofreciendo ventajas en términos de rapidez, portabilidad y bajo consumo de recursos. Durante las pruebas realizadas, este enfoque ofreció un tiempo de inferencia promedio de 110 ms por imagen y una precisión aceptable en condiciones controladas.

Por su lado, YOLOv5 ofrece escalabilidad y capacidad de detección en tiempo real dependiendo del nivel de procesamiento, diseñada para la detección precisa de objetos mediante bouding boxes, logró identificar correctamente la ubicación del arma blanca, incluso en escenarios con iluminación variable, aunque con un tiempo de inferencia promedio de 400 ms por imagen, ocasionando un mayor consumo de recursos.

Ambos modelos tiene la capacidad de la detección y clasificación de imágenes con diferentes niveles de complejidad según los objetivos específicos del proyecto.

En la tabla 2.4, se puede observar una similitud entre los valores de precisión y recall en ambos modelos, por su parte en Teachable Machine se partió por el cálculo promedio en estas métricas, obteniendo resultados iguales.

En YOLOv5, la similitud proviene del ajuste del umbral de confianza y promedio del recall, permitiendo maximizar ambas métricas de forma equilibrada.

Parámetros definidos en ambos modelos para detección de objetos

Teachable Machine basado en Keras, fue entrenado con tres clases diferentes, incluyendo un total 3000 imágenes (1000 por clase), utilizando en su mayoría datasets públicos y complementando con imágenes personalizadas, ajustando su iluminación, diferentes posiciones y ángulos para mejorar su precisión. Durante su entrenamiento, se configuraron 20 epochs con un tamaño de lote de 16, dejando la tasa de aprendizaje de 0.0001, estos parámetros como indica en la tabla 2.5, quedaron establecidos debido a que permitieron alcanzar una precisión del 100 % en la clase de detección de arma, sin evidencias sobre ajuste. El modelo fue exportado en formato `.h5` para su posterior integración en aplicaciones basadas en TensorFlow.

Clases	3 tipos
Tamaño datasets	1000 imágenes/clase
Datasets	Públicos y personalizados
Número de época	20
Tamaño de lote	16
Tasa de aprendizaje	0.0001
Formato	Keras (.h5)

Tabla 2.5: Parámetros para el entrenamiento del modelo en Teachable Machine

YOLOv5 basado en Pytorch, su entrenamiento se realizó con dos clases diferentes, para la identificación de armas blancas, y rostro, incluyendo un total de 1000 imágenes, utilizando completamente datasets públicos para su entrenamiento, la entrada al modelo se re-dimensionó a 640x640 píxeles para optimizar el balance entre precisión y velocidad. Además de configurarse 25 épocas con un tamaño de lote de 16, estos parámetros fueron establecidos como se indica en la tabla 2.6. El modelo final se guardó en formato `.pt` compatible con PyTorch, listo para inferencia rápida en dispositivos con GPU, sin embargo para asegurar la compatibilidad con el sistema operativo Windows y facilitar la interoperabilidad con otras plataformas y herramientas, se modificó el formato de Pytorch a `.onnx`.

Clases	2 tipos
Tamaño datasets	1000 imágenes
Datasets	Públicos
Número de época	25
Tamaño de lote	16
Re-dimensión (px)	640x640
Formato	ONNX (.onnx)

Tabla 2.6: Parámetros para el entrenamiento del modelo YOLOv5

Comparación de modelos para análisis semántico

Para el análisis semántico del lenguaje y clasificación de frases se consideraron dos enfoques: el uso de un modelo de aprendizaje profundo como **RoBERTa**, y el uso de algoritmos de aprendizaje automático clásicos implementados con la biblioteca **Scikit-learn**.

Ambos modelos fueron entrenados y evaluados utilizando el mismo conjunto de datos, asegurando

condiciones de prueba idénticas para garantizar una comparación justa. Durante el proceso se aplicaron métricas como precisión, recall, F1-score, así como el tiempo de inferencia y el consumo de recursos computacionales (CPU, GPU y memoria).

Adicionalmente, se generaron visualizaciones complementarias para analizar el rendimiento de cada modelo:

- Para **Scikit-learn**, se construyeron matrices de confusión y curvas ROC por clase, calculando también el Área Bajo la Curva (AUC) para cuantificar la capacidad de discriminación.
- Para **RoBERTa**, se aprovecharon las métricas generadas durante el entrenamiento, incluyendo eval/accuracy, eval/f1, train/loss, train/learning_rate, eval/loss y eval/precision. Estas métricas fueron representadas gráficamente para evaluar la evolución del modelo y detectar posibles problemas como overfitting o underfitting.

Las pruebas se ejecutaron en un entorno controlado para minimizar factores externos que puedan influir en los resultados como se muestra en la Tabla 2.7 . De esta manera, se obtuvo un análisis comparativo objetivo que permitió identificar fortalezas y debilidades de cada modelo en relación con los objetivos del proyecto.

Métrica	**RoBERTa**	**Scikit-learn**
Precisión en lenguaje complejo	93.4 %	97.5 %
Requerimientos computacionales	GPU 8GB VRAM	CPU 2 núcleos
Velocidad de inferencia	3.2 s	0.4 s
Tamaño del modelo	480 MB	12 MB
Tiempo de entrenamiento	45 min	5 min
Pico de uso de memoria	6.8 GB	0.8 GB

Tabla 2.7: Comparativa de características entre RoBERTa y modelos clásicos de Scikit-learn

Parámetros definidos para los modelos de análisis semántico

Tanto Scikit-learn como RoBERTa fueron entrenados utilizando el mismo dataset personalizado, conformado por videos de referencia de casos delictivos obtenidos de diversas plataformas como TikTok y YouTube. Este conjunto de datos incluye más de 1500 frases clasificadas entre agresivas y no agresivas. Asimismo, los audios procesados para la evaluación son idénticos en ambos modelos, con el fin de garantizar la mayor equivalencia posible en la comparación de resultados, como se ve en la Tabla 2.8.

Clases	2 tipos
Tamaño datasets	1500 frases
Datasets	Personalizado
Número de época	25
Tamaño de lote	16
Formato	WAV

Tabla 2.8: Parámetros para el entrenamiento de los modelos de análisis semántico

2.2. Evaluación de métricas

Con el fin de asegurar que el sistema cumpla con las necesidades y requerimientos técnicos para la experiencia del usuario, se realizó una evaluación basada en métricas de desempeño en base al cumplimiento con los objetivos específicos.

Evaluación del Objetivo 1: Detección de armas blancas en tiempo real

Para el primer objetivo, que consiste en desarrollar un sistema de detección en tiempo real de armas blancas a partir de imágenes capturadas por cámara, se evaluó el desempeño del modelo utilizando métricas clásicas de clasificación.

Se define la matriz de confusión sobre el conjunto de prueba, considerando:

- **TP (True Positives):** imágenes con arma blanca correctamente clasificadas como "arma".
- **FP (False Positives):** imágenes sin arma clasificadas erróneamente como "arma".
- **TN (True Negatives):** imágenes sin arma clasificadas correctamente como "no arma".
- **FN (False Negatives):** imágenes con arma clasificadas erróneamente como "no arma".

A partir de estos valores, se calcularon las siguientes métricas:

Tasa de Verdaderos Positivos (Recall / Sensibilidad):

$$\text{TPR} = \frac{TP}{TP + FN}$$

Tasa de Falsos Positivos:

$$\text{FPR} = \frac{FP}{FP + TN}$$

Precisión (Precision):

$$\text{Precision} = \frac{TP}{TP + FP}$$

Exactitud (Accuracy):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

F1-Score:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Adicionalmente, dado que el sistema está diseñado para un entorno de tiempo real con recursos limitados, se midió el tiempo de inferencia por imagen (desde su captura hasta la alerta en el bot de Telegram) y se analizó el uso de recursos del sistema (memoria RAM, consumo de CPU y almacenamiento requerido para los datasets). Estos parámetros permitieron comprobar no solo la precisión del modelo, sino también su eficiencia práctica en escenarios con restricciones de hardware.

Evaluación del Objetivo 2: Detección de frases agresivas en tiempo real

Para el segundo objetivo específico, que consiste en desarrollar un sistema capaz de detectar frases amenazantes a partir de audio, se evaluó el desempeño del herramienta de transcripción y del modelo de clasificación de frases según su nivel de amenaza. Para la herramienta de transcripción se utilizó como métrica el Word Error Rate (WER). La métrica Word Error Rate se define como:

$$\text{WER} = \frac{S + D + I}{N}$$

Donde:

- S: número de sustituciones de palabras.
- D: número de eliminaciones (palabras omitidas).
- I: número de inserciones (palabras extra añadidas).
- N: número total de palabras en la transcripción de referencia.

Para la evaluación de métricas del modelo de análisis semántico se definirá la matriz de confusión considerando:

- **TP (True Positives):** frases amenazantes detectadas correctamente como amenazantes.
- **FP (False Positives):** frases no amenazantes clasificadas erróneamente como amenazantes.
- **TN (True Negatives):** frases no amenazantes detectadas correctamente como no amenazantes.
- **FN (False Negatives):** frases amenazantes clasificadas erróneamente como no amenazantes.

A partir de estos valores, se calcularon las siguientes métricas:

Precisión (Precision):

$$\text{Precision} = \frac{TP}{TP + FP}$$

Sensibilidad / Recall (TPR):

$$\text{Recall} = \frac{TP}{TP + FN}$$

Exactitud (Accuracy):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

F1-Score:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluación del Objetivo 3: Sistema de Alertas en Tiempo Real

Finalmente, para el último objetivo, relacionado con la implementación de un sistema basado en eventos que transmita información de manera autónoma ante posibles actos delictivos, se evaluará la integridad, consistencia y rapidez de los datos.

Se definirá la latencia por etapas, dividiendo el tiempo que se demora en procesar los datos por fase.

- Tiempo de captura: $t^{(\text{captura})}$
- Tiempo de procesamiento: $t^{(\text{Procesamiento})}$
- Tiempo de envío de alertas: $t^{(\text{Alerta})}$

En el tiempo de captura, se considera el envío de datos al servidor de procesamiento. En la fase de procesamiento se define el tiempo que tarda, desde la detección y clasificación de la IA para el envío a Node-RED. Finalmente, el tiempo de alertas corresponde al transcurso del flujo del orquestador de datos y envío a telegram.

Además, se define el cumplimiento de SLA (Service Level Agreement) de latencia.

$$\text{SLA}_{\text{lat}} = \frac{\#\{i : L_i \leq T\}}{M} \times 100\,\%$$

donde:

- L_i es la latencia observada en el evento i,
- T es el umbral de latencia definido en el SLA,
- M es el número total de eventos medidos,
- $\#\{i : L_i \leq T\}$ es el número de eventos cuyo tiempo de latencia cumple con el umbral.

Estas métricas permiten verificar que el sistema entregue alertas completas, consistentes y rápidas, asegurando la integridad de la información en la transmisión de datos de seguridad.

2.3. Recursos

- **Hardware:**
  - ESP32-CAM, cuya funcionalidad es capturar imágenes en un intervalo de tiempo.
  - ESP32, microcontrolador para integración de módulos de audio y GPS.
  - Módulo GPS, obtener la ubicación por medio de coordenadas.
  - Módulo micrófono, capturar el audio ambiente dentro del vehículo.
  - Fuente de alimentación portátil.
- **Software:**
  - Python con librería de IA para la detección de incidentes.
  - Flask, servidor de procesamiento.
  - Node-RED, flujo de datos para el envío de alertas.
  - Base de datos, MYSQL para el almacenamiento registros.
  - Bot de telegram, recepción de alertas.

2.4. Métodos de seguridad

- Confidencialidad de datos: Las imágenes y audios captados tienen propósito académico, al igual que los datos almacenados en la base de datos, cuyos registros fueron inventados.
- Seguridad de la información: La transmisión de sus datos se maneja bajo protocolos de cifrado (HTTPS/TLS) para evitar manipulaciones de terceros.
- Alcance: Nuestro proyecto se desarrolla dentro de una red local para tener un ambiente controlado de sus datos.

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

El diseño del proyecto se basa en un sistema de seguridad vehicular orientado al monitoreo en tiempo real mediante el uso de tecnologías IoT e inteligencia artificial. La implementación incluye la captura de imágenes y audios en intervalos cortos de tiempo de forma periódica con el objetivo de detectar situaciones de riesgo de forma automatizada, que ante la identificación de un posible incidente delictivo generará alertas enviadas a través de la API de Telegram a las autoridades que dispondrán del acceso inmediato a esta información para la toma de decisión.

3.1. Arquitectura del sistema

El sistema propuesto está diseñado para el envío de imágenes y registro de audio como evidencia un incidente delictivo hacia las autoridades, garantizando la seguridad y confiabilidad en sus datos. La transmisión se realiza de manera inmediata ante la presencia de un evento capturado con la utilización tecnologías IoT e inteligencia artificial. Su disponibilidad depende del acceso a internet para asegurar su comunicación en tiempo real.

La arquitectura general del sistema está compuesta por los siguientes módulos:

- **ESP32-CAM:** Este microcontrolador actúa como un capturador de imágenes en un intervalo corto de tiempo para ser enviado al servidor de procesamiento.
- **ESP32:** Este microcontrolador actúa como un capturador de voz y ubicación, gracias a la incorporación de un micrófono para la detección de alguna agresión en el audio ambiental dentro del vechículo, así como un módulo GPS que proporciona la ubicación en tiempo real mediante coordenadas, las cuales son enviadas al servidor de procesamiento en caso de un incidente.
- **Servidor Flask:** Este servidor basado en Python, tendrá incorporados modelos de aprendizaje en IA, para el procesamiento de imágenes en la detección de armas blancas y voz para la identificación de frases amenazadoras o agresivas.

- **Node-RED:** Este orquestador de datos, recibirá la información procesada por el servidor, el cual se encargará de ejecutar reglas automatizadas e integraciones con otro servicios para la recolección y envío de alertas al servidor de mensajería instantánea como Telegram, que será monitoreado por las autoridades.
- **Base de datos MySQL:** Esta base de datos alojada en la nube almacenará la información del propietario del vehículo, junto con sus características y datos relevantes.
- **Bot de Telegram:** Una vez completada la validación y enriquecimiento del evento, el flujo de Node-RED enviará una notificación automatizada a través de API REST de Telegram junto con la imagen capturada, ubicación GPS y datos del vehículo.

En la Figura 3.1 se ilustra el diagrama general de la arquitectura, que incluye las conexiones de los componentes involucrados en el sistema y el flujo de sus datos para la comunicación con las autoridades mediante el bot de telegram.

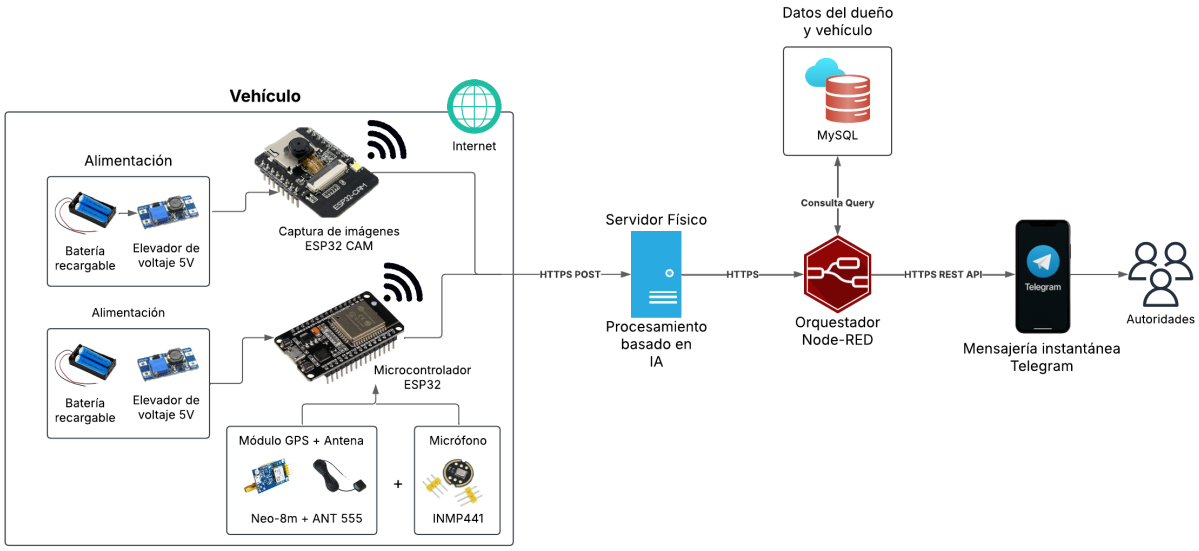


Figura 3.1: Arquitectura general del sistema propuesto

En la Figura 3.2 se desarrolló un diagrama de red que muestra la disposición de los puertos, direcciones IP y flujos de comunicación entre los diferentes componentes del sistema. Aunque los módulos de análisis de imágenes y análisis de audio operan de forma independiente, existe una integración funcional entre ambos. El microcontrolador NodeMCU32s, encargado de la captura de audio, también envía el identificador del vehículo y las coordenadas GPS. Esta información geográfica es utilizada tanto para la clasificación de audio como por el módulo ESP32-CAM en el análisis de imágenes, permitiendo correlacionar eventos de audio e imagen bajo un mismo contexto espacial y temporal. De esta manera, el sistema puede generar alertas más completas y precisas,

ya que las evidencias recogidas por ambos subsistemas se integran en el orquestador Node-RED, centralizando la información para su posterior envío a las autoridades a través de la plataforma de Telegram.

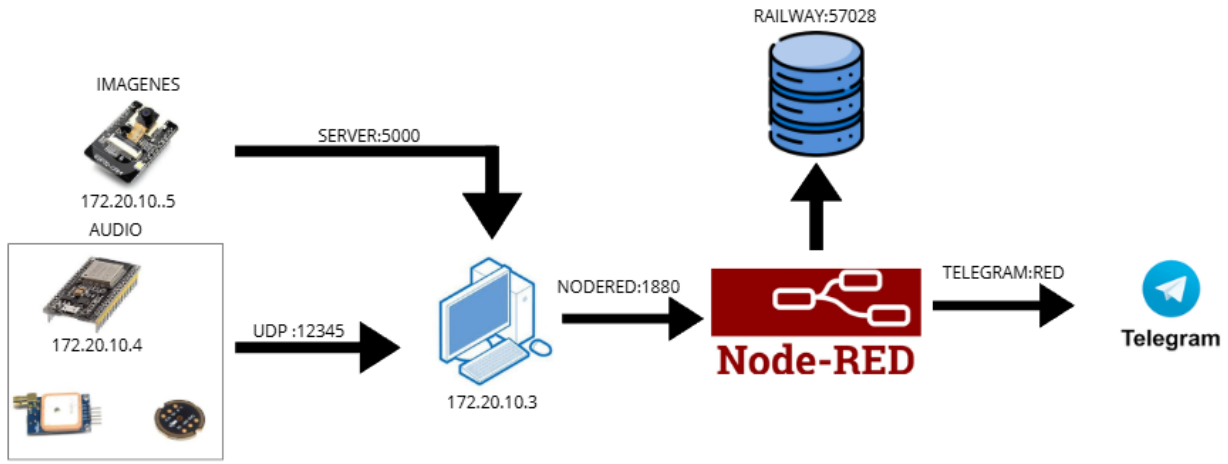


Figura 3.2: Diagrama de Puertos e IPs

3.2. Flujo de datos

Fase 1: Captura de datos

Mediante el uso del microcontrolador ESP32-CAM, se realiza la captura periódica de datos en intervalos cortos de tiempo, haciendo uso de la cámara integrada. Paralelamente, un microcontrolador adicional con un micrófono externo conectado se encarga de capturar el registro del audio ambiente, así como un módulo GPS capaz de obtener las coordenadas precisas en tiempo real del vehículo; ambos dispositivos funcionan de forma paralela con el fin de mejorar la eficacia del sistema, como se muestra en la figura 3.4. Es decir, si la imagen capturada detecta la presencia de un arma blanca, el sistema activará automáticamente una alerta, de igual manera, si el audio grabado contiene una frase amenazante o agresiva, se genera una alarma inmediata para el aviso a las autoridades. Estos datos obtenidos son enviados a un servidor de procesamiento basado en IA mediante Flask, donde son analizados por modelos de aprendizaje automático entrenados previamente para estos escenarios actuando conforme el caso identificado.

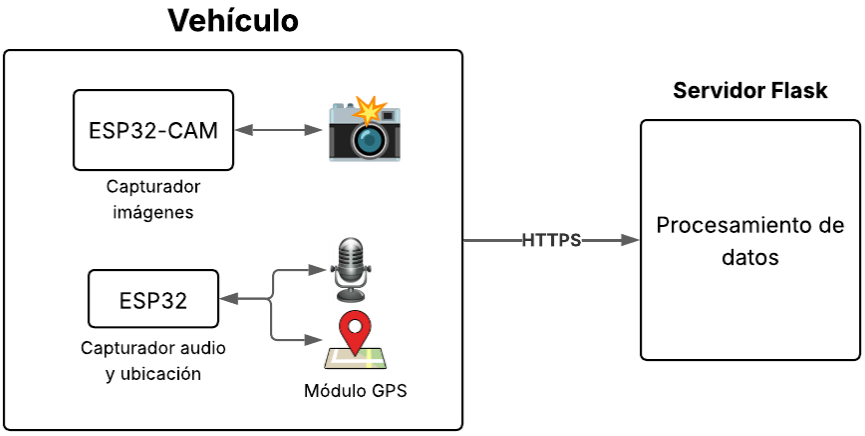


Figura 3.3: Captura y envío de datos

Fase 2: Procesamiento y protección de datos

El sistema desarrollado se encarga de la recopilación y transmisión de datos como imágenes y audios capturadas por el microcontrolador, las evidencias visuales, auditivas y la ubicación GPS se encuentran relacionadas con personas actuando en un incidente delictivo, por lo que es indispensable garantizar la protección durante el flujo del procesamiento de sus datos. Con el objetivo de asegurar la confiabilidad de esta información enviada al servidor de procesamiento, se implementa un protocolo de cifrado de datos mediante HTTPS/TLS, evitando la intercepción y manipulación de terceros durante su transmisión.

El servidor se encarga de recibir y procesar estos datos, ejecutándose en este entorno modelos de inteligencia artificial entrenados previamente para la detección de armas blancas e identificación de frases agresivas o amenazantes, el procesamiento se realiza en tiempo real, sin almacenamiento persistente de los archivos, es decir, si se considera como un dato no relacionado a un incidente delictivo es descartado, con el objetivo de minimizar riesgos de filtración o uso indebido. Como se indica en la figura 3.4.

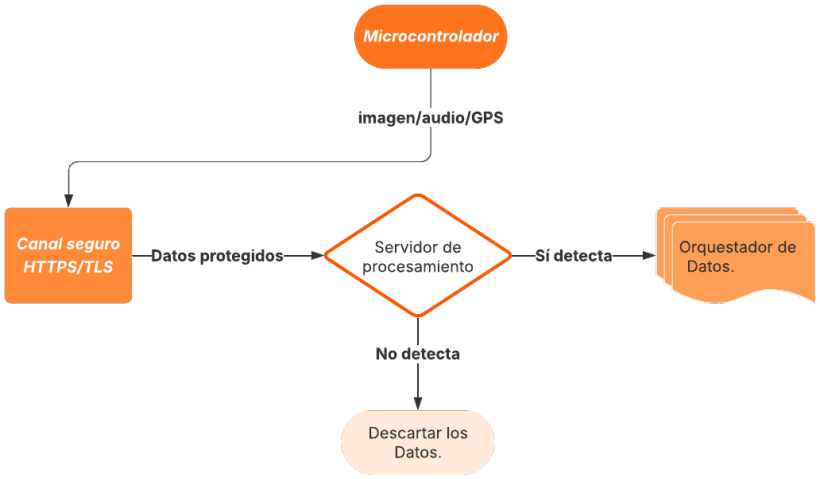


Figura 3.4: Flujo de procesamiento

Fase 3: Alertas automatizadas

Una vez que los datos han sido procesados por el servidor, aquellos que cumplen con los parámetros establecidos por los modelos de IA entrenados, serán enviados a Node-RED, mediante una petición HTTP POST con el propósito de orquestar el flujo de estos datos, generando una alerta automatizada. En este entorno, se realiza una consulta query a la base de datos MySQL alojada en la nube para la extracción de estos datos previamente registrados de manera manual por parte del administrador del sistema, esto corresponde a la tabla `datos_vehiculo` (Véase en la Figura 3.5) incluyendo campos como nombre, apellido, placa, año del vehículo, marca, modelo, color, reportado como robado y multas pendientes.

La información asignada hace referencia a datos del propietario del vehículo y sus características asociadas para ser enviados por medio de un bot de telegram con la imagen adjunta capturada en este incidente (Véase en la Figura 3.6). Este flujo automatizado asegura que el sistema no solo detecte incidentes en tiempo real, sino que también contextualice la alerta y la comunique eficientemente a través de una plataforma segura y accesible a las autoridades.

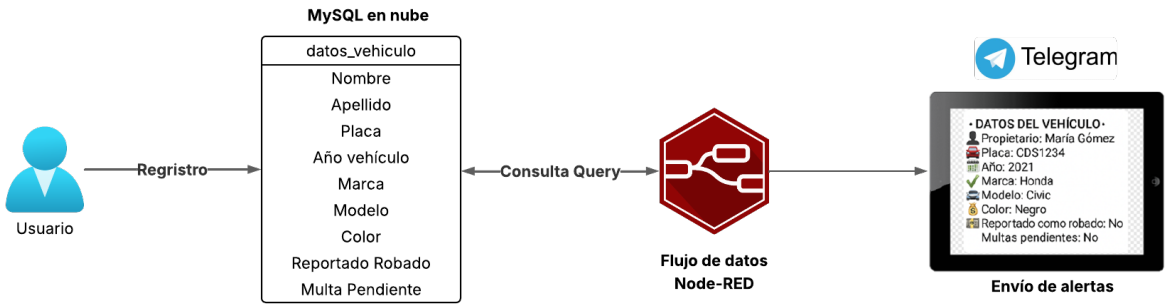


Figura 3.5: Registro base de datos

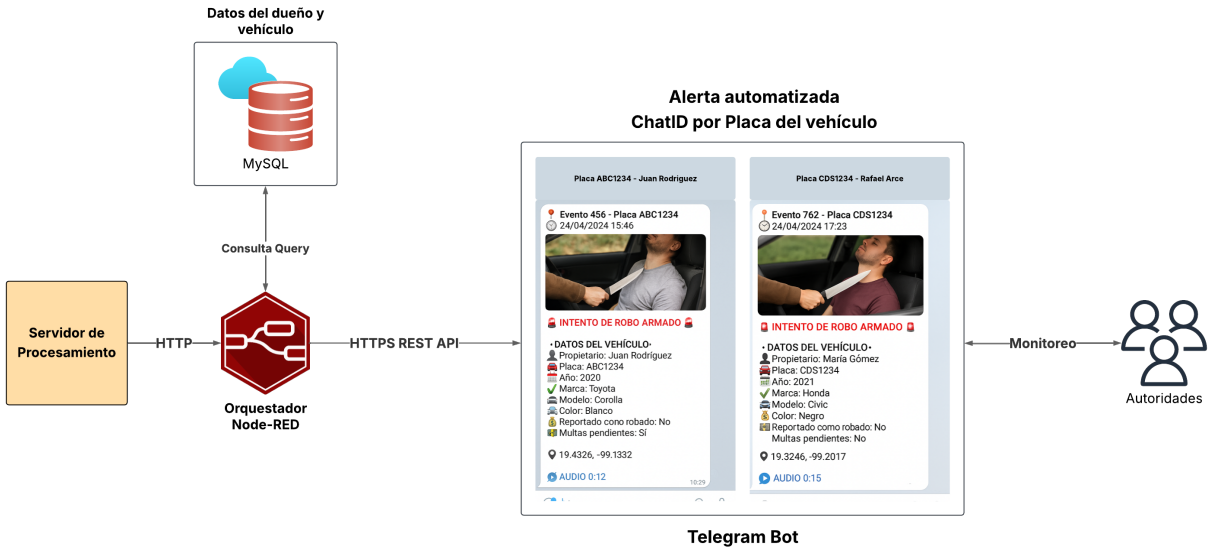


Figura 3.6: Envío de Alertas

3.3. Implementacion del sistema

Sistema para análisis de imágenes

El servidor de procesamiento de imágenes tendrá embebido el modelo entrenado en formato `.h5` debido a su menor tiempo de procesamiento de acuerdo a los datos enviado por el capturador, además de la inclusión de un sistema de colas para evitar el congestionamiento en la recepción de varios datos de manera simultánea, usando el tipo de cola FIFO.

Una vez clasificada la imagen en el servidor, es enviada al orquestador de datos Node-RED (Véase en la Figura 3.7) siempre y cuando la clase corresponda a `con arma`. Este flujo se encarga de enviar un mensaje personalizado en señal de alerta con los datos del propietario del vehículo y sus características a las autoridades, junto con su ubicación actual por medio de coordenadas GPS y enlace de Google Maps para mayor precisión, estos datos serán monitoreados mediante la aplicación de Telegram.

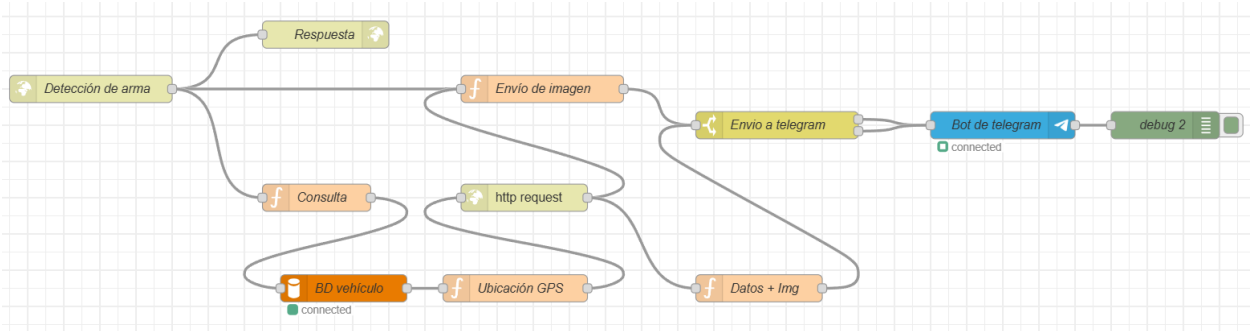
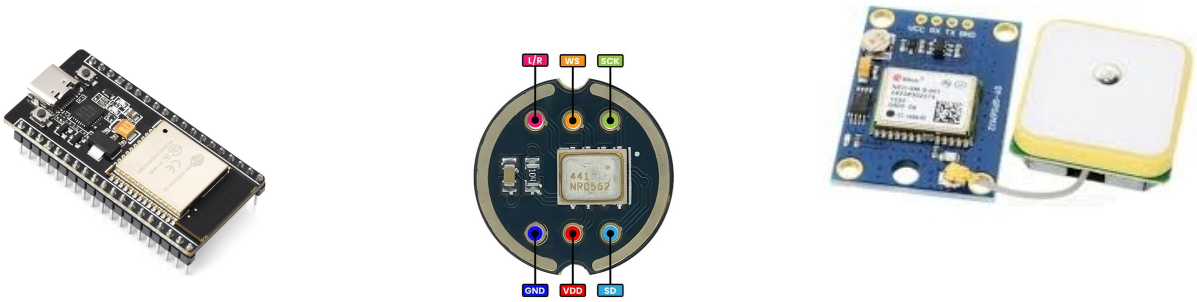


Figura 3.7: Node-RED Orquestador de datos

El sistema funciona en una red local durante las pruebas realizadas, por lo que se tiene previsto incorporar protocolos de seguridad para la utilización comercial como HTTPS con la utilización de cifrado SSL/TLS de extremo a extremo para la protección de los datos durante su transmisión.

Sistema para análisis de audios

Para la captura de audio se utilizó un microcontrolador NodeMCU32s junto con un micrófono digital INMP441. La captura se realizó de forma periódica, en intervalos de 15 segundos, durante los cuales se enviaron paquetes mediante el protocolo UDP. Estos paquetes contenían la placa del vehículo, el audio codificado en bytes, y las coordenadas geográficas obtenidas mediante un sensor GPS Neo6M.



(a) NodeMCU32s (b) INMP441 (c) NEO-6M

Figura 3.8: Componentes utilizados para la captura de audio y ubicación

Además de las imágenes de los componentes de la Figura 3.8, se elaboró un diagrama físico que ilustra las conexiones entre ellos. El NodeMCU32s se conecta al micrófono digital INMP441 mediante los pines 31, 25 y 14, que corresponden respectivamente a las líneas de datos y reloj SD, WS y SCK, junto con las conexiones de GND y la alimentación a 3.3 V. Por su parte, el sensor GPS Neo6M se enlaza al microcontrolador a través de los pines TX y RX del puerto UART, con su alimentación suministrada a 5 V como se muestra en la Figura 3.9.

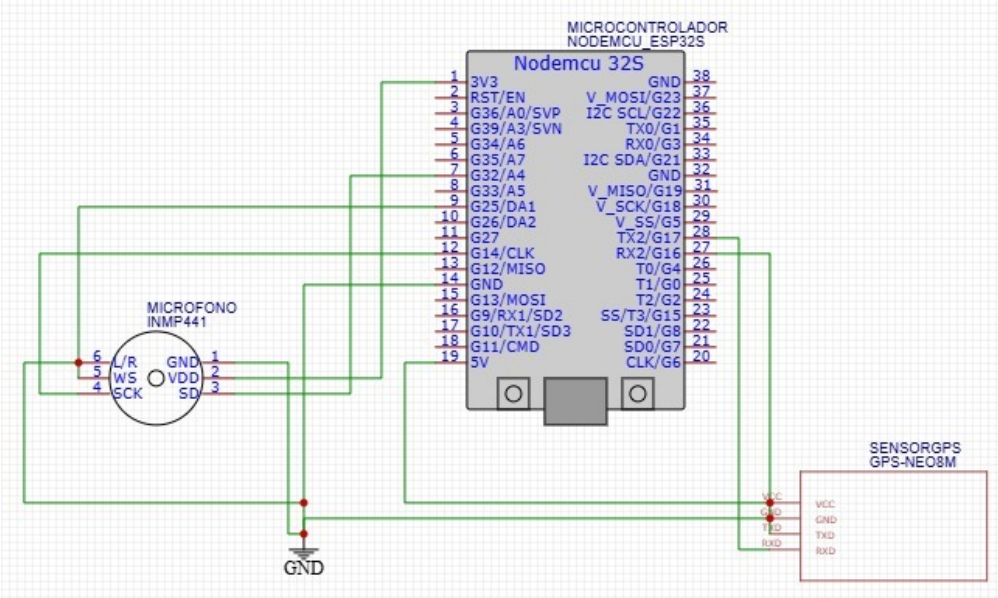


Figura 3.9: Diagrama de conexiones

La información será recibida por el servidor encargado del procesamiento de audio, el cual ejecutará el algoritmo correspondiente para su análisis que en este caso se utilizo Scikit-learn. En caso de que el audio sea clasificado como agresivo, se generará un evento. Cada evento podrá ser enviado hasta dos veces a Node-RED, aunque igualmente será registrado en el sistema. Además, se establece que un evento tiene una duración de una hora; si ocurre otra anomalía fuera de ese periodo, esta será considerada como un nuevo evento independiente.

La nueva información será enviada a Node-RED, incluyendo la placa del vehículo, la URL para acceder al audio grabado y la ubicación geográfica correspondiente. Node-RED cuenta con un flujo diseñado para procesar esta información: extrae la placa del vehículo, consulta la base de datos para obtener los datos del usuario asociado y, con esta información, genera una alerta que será enviada a un bot de Telegram. (Véase la Figura 3.10)

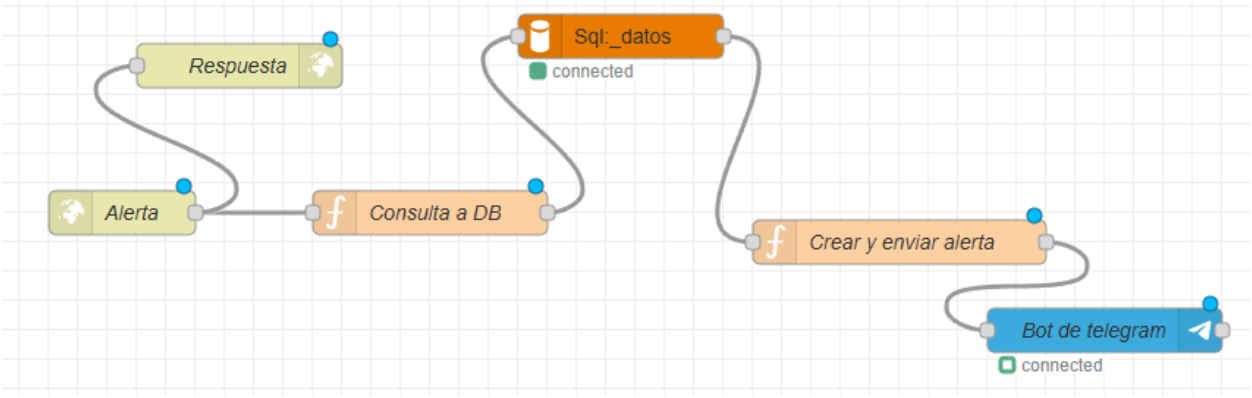


Figura 3.10: Node-RED Orquestador de datos

Vista de audios

En la interfaz del servidor Flask de la Figura se implementará una sección dedicada a la visualización y gestión de los audios capturados. Estos se organizarán de acuerdo con la placa del vehículo y el tipo de evento detectado, permitiendo una identificación rápida. Cada vez que el sistema detecte un nuevo audio clasificado como agresivo, este será automáticamente cargado y visible en la interfaz, manteniendo la información actualizada en tiempo real. (Véase la Figura **??**)

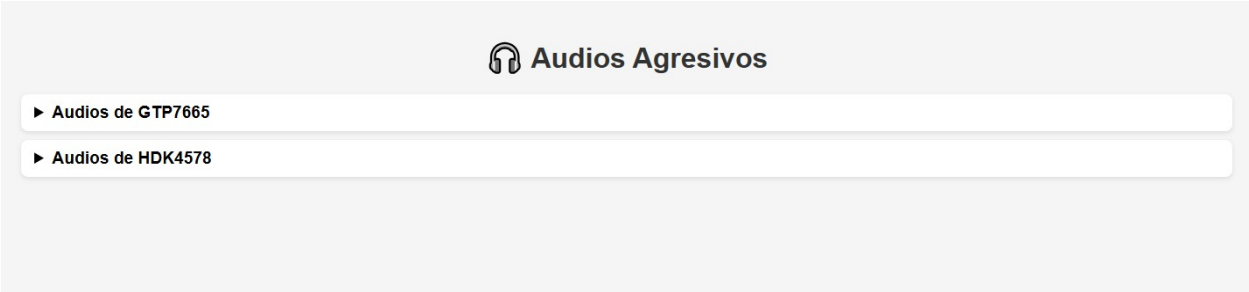


Figura 3.11: Interfaz para visualizar audios

Vista de ubicación a tiempo real

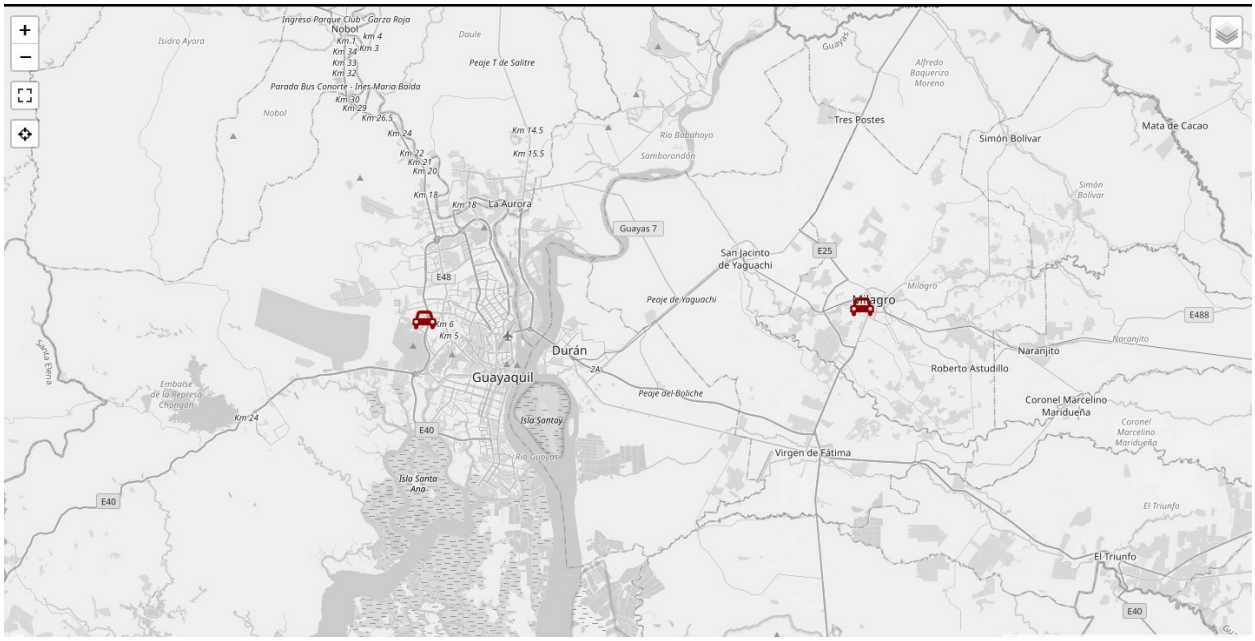


Figura 3.12: Ubicación en tiempo real

La ubicación geográfica se mostrará en un mapa interactivo implementado en Node-RED mediante el módulo worldmap como se muestra en la Figura 3.12. Esta visualización se activará automáticamente en dos casos: 1.Cuando se detecte un audio clasificado como agresivo. 2.Cuando se identifique visualmente la presencia de un arma blanca.

En ambos casos, el mapa presentará la posición exacta del evento en tiempo real, facilitando el

seguimiento inmediato y la toma de decisiones rápidas.

3.4. Implementacion de modelos de IA

Modelos para la detección de objetos

Basado en las comparativas realizadas anteriormente entre la herramienta Teachable Machine basado en Tensorflow y Yolov5 implementado en PyTorch, se pretende evaluar el desempeño real de estos modelos mediante su implementación práctica utilizando los mismos parámetros y recursos actuales del sistema. Ambos modelos fueron entrenados con datasets púbicos recientes, con la inclusión de imágenes con armas blancas y objetos similares en ambos casos con el fin de evaluar su capacidad de clasificación y detección correcta durante las situaciones de riesgo.

- **Teachable Machine basado en Tensorflow**

  El modelo utilizado fue exportado en formato `.h5` correspondiente a Keras, compatible con el framework Tensorflow, debido a su tiempo mínimo en el procesamiento de imágenes por el servidor, utilizando librerías propias del framework, tales como Keras, Numpy, Pillow (PIL) e io, esta última utilizada para la lectura de imágenes enviadas en formato bytes.

  La función de predicción se realiza sobre las imágenes recibidas por el capturador en formato binario, siendo preprocesadas para la conversión 224x224 y normalizadas según los requisitos del modelo Keras. Además se previene el bloqueo del servidor al recibir y procesar varias imágenes de forma simultánea con la implementación de una arquitectura basada en colas (`queue.Queue`) y procesamiento de hilos.

  Finalmente, si el modelo detecta la clase `con arma` con un nivel de confianza del 85 % es enviado al orquestador de datos para su posterior notificación. Durante las pruebas, el modelo brinda un nivel de exactitud del 99 % en la clase `con arma` y un 100 % en el resto de las clases, demostrando un alto nivel de fiabilidad para ser tomado en consideración dentro del sistema de seguridad vehicular, cuyos datos serán demostrados en el capítulo 4, bajo el desarrollo de sus métricas.

  En la figura 3.13 se puede observar los parámetros establecidos para su entrenamiento, lo que conlleva al nivel de precisión indicado.

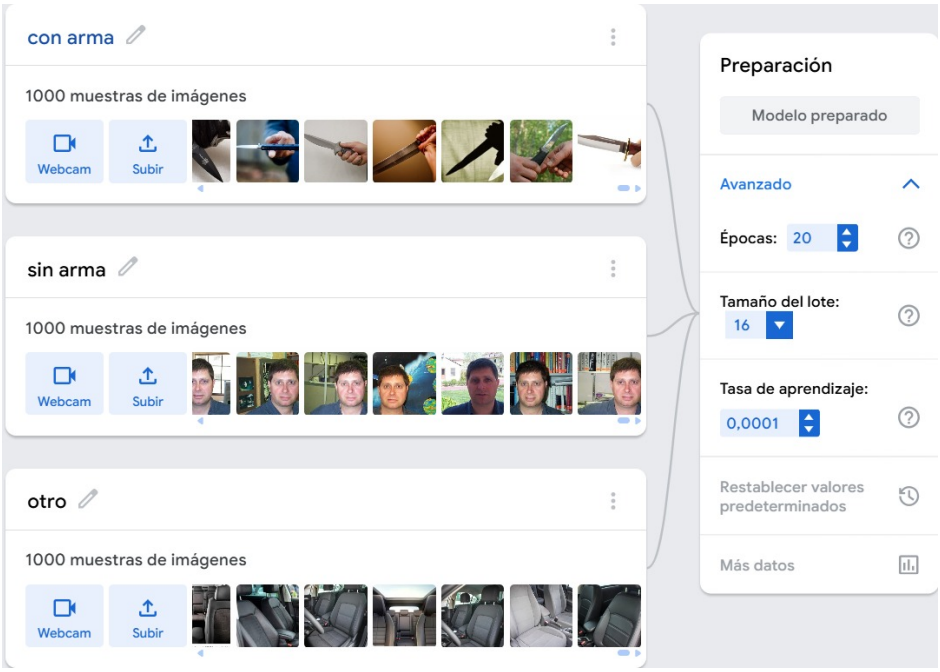


Figura 3.13: Parámetros de entrenamiento en Teachable Machine

- **YOLOv5 implementado en PyTorch**

  Para su implementación práctica del modelo de detección YOLOv5, se utilizó OpenCV junto con el modelo `.pt` obtenido de PyTorch, pero que fue convertido a ONNX con el objetivo de mejorar la compatibilidad en su procesamiento debido al entorno basado en Windows en el que fue ejecutado, permitiendo visualizar la detección de objetos en tiempo real mediante el uso de la cámara del capturador de imágenes.

  Esta arquitectura, trabaja mediante bounding boxes para la localización precisa del objeto que se pretende detectar, ocasionando un mayor trabajo computacional en comparación con el anterior modelo de clasificación global. Debido a la complejidad en la detección de varios objetos en una misma imagen junto con su generación de cajas limitadoras para cada clase establecida, la velocidad de inferencia se vio limitada al ejecutarse en un entorno sobre CPU.

  La selección adecuada del umbral de confianza en este escenario depende de la necesidad del sistema, es así que para evitar falsos positivos se podría utilizar un umbral alto del 94 % considerando un menor recall, es decir, el modelo no detectará algunos casos verdaderos debido a una menor certeza. Para mantener un balance entre precisión y recall, es preferible considerar un umbral del 48 % evitando descartar datos que pueden ser correctos en la detección de armas blancas frente a un incidente.

  En la figura 3.14 se puede observar los parámetros establecidos para su entrenamiento, lo que conlleva al nivel de precisión indicado.

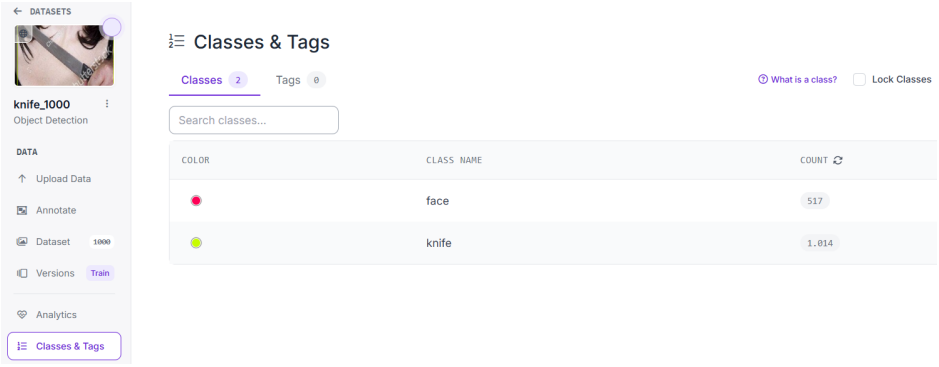


Figura 3.14: Parámetros de entrenamiento YOLOv5

Tras la implementación práctica de ambos modelos, se optó por utilizar el brindado por la herramienta Teachable Machine, basado bajo la compatibilidad con los recursos actuales del sistema, fácil integración con el servidor de procesamiento y bajo requerimiento computacional, además de no ser tan indispensable el uso de la localización del objeto a detectar, alineándose con el objetivo y desarrollo del proyecto.

Modelos para la transcripción de audios

Para la implementación del modelo Whisper, se configuró la versión medium preentrenada en español para procesar audios de 20 segundos. Se integró el modelo en el sistema de prueba, permitiendo la transcripción de cada archivo de audio de manera automática. Durante la implementación, se ajustaron los parámetros necesarios para manejar audios con presencia de ruido de fondo, asegurando que el modelo pudiera generar transcripciones consistentes y completas para su posterior análisis.

```
Cargando modelo Whisper medium...
RAM usada por el modelo: 3.99 GB

--- Resultados para medium ---
Tiempo total: 14.82 segundos
RAM máxima usada: 4.03 GB
Número total de palabras en el audio: 20
```

Figura 3.15: Salida de WhisperIA

Como se muestra en la figura 3.15, con la ayuda de Python se calcularon algunos parámetros iniciales, como el número de palabras, el uso de RAM y el tiempo de procesamiento. Este mismo procedimiento se aplicará a un conjunto más amplio de audios, permitiendo obtener un análisis completo de las métricas de rendimiento del modelo Whisper sobre diferentes archivos y condiciones de audio.

Modelos para el análisis semántico

Para identificar y clasificar posibles amenazas verbales en las transcripciones de audio, se utilizó Scikit-learn, entrenando clasificadores tradicionales como la regresión logística y las máquinas de soporte vectorial (SVM). El modelo se entrenó con un dataset personal compuesto por frases etiquetadas que representan los niveles de agresividad verbal.

Este enfoque permite una respuesta rápida y eficiente ante entradas textuales. Aunque su precisión puede ser menor en comparación con modelos basados en transformers, su velocidad de inferencia, facilidad de entrenamiento y bajo consumo de recursos lo hacen adecuado para entornos con necesidades de respuesta en tiempo real.

La implementación práctica permitió evaluar ventajas y limitaciones en cuanto a entrenamiento, rendimiento y compatibilidad con el sistema. Scikit-learn destacó por su rapidez en el procesamiento y facilidad de integración, lo que lo convierte en una opción eficiente para sistemas con recursos limitados, demostrando un desempeño sólido para la identificación de frases amenazantes.

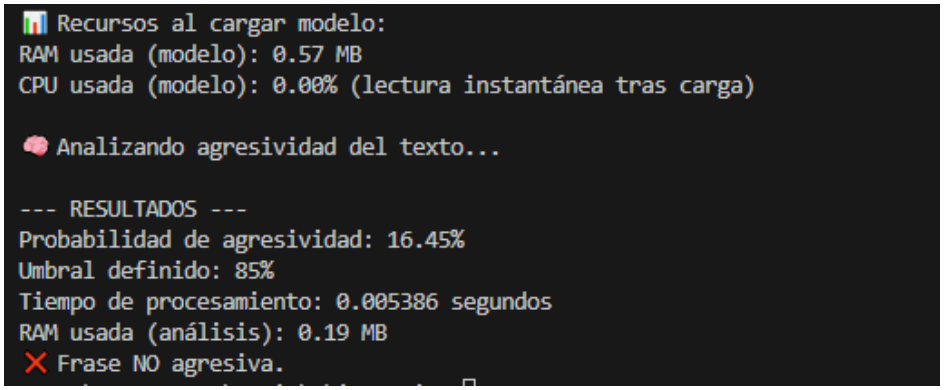


Figura 3.16: Salida de Scikit-Learn

El sistema desarrollado combina Whisper y Scikit-learn para identificar y clasificar posibles amenazas verbales en transcripciones de audio. Primero, el audio capturado por el ESP32 es procesado por Whisper, que se encarga de generar la transcripción textual de manera automática. Posteriormente, el texto resultante pasa por un clasificador entrenado con Scikit-learn, capaz de identificar frases agresivas o no agresivas según un dataset previamente etiquetado.

Esta integración permite aprovechar la capacidad de transcripción avanzada de Whisper y la rapidez y eficiencia de los clasificadores de Scikit-learn, obteniendo un flujo continuo desde la captura del audio hasta la determinación del nivel de agresividad.

A continuación, en la Figura 3.17, se muestra el diagrama de flujo que representa este proceso completo, destacando cada etapa desde la adquisición del audio hasta la generación del resultado final.

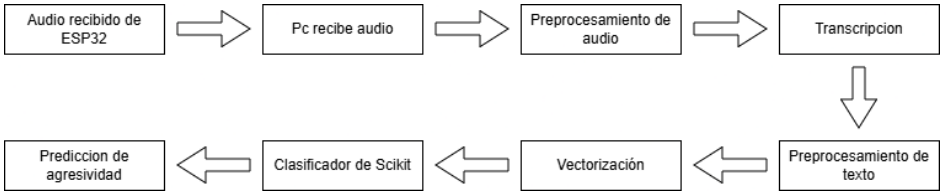


Figura 3.17: Diagrama del análisis semántico

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

4.1. Métrica para el objetivo 1

Para el cumplimiento del primer objetivo basado en un sistema de detección de armas blancas, se utilizó el modelo de IA en formato Keras, entrenado en Teachable Machine debido a la precisión en sus clases y sensibilidad al momento de la detección y clasificación.

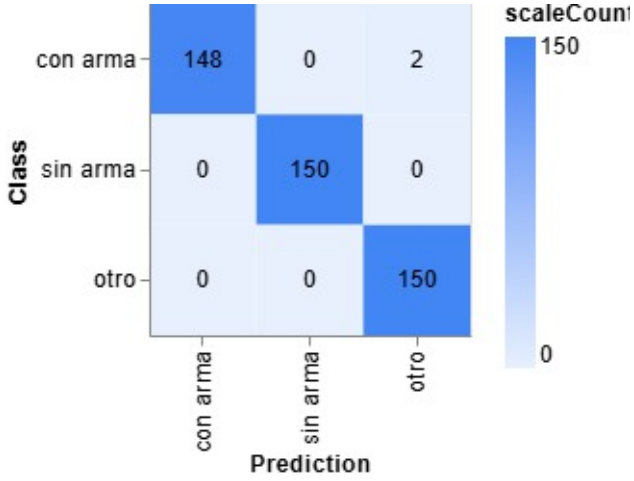


Figura 4.1: Matriz de confusión

Para el cálculo del Recall de la clase objetivo `con arma`, se utiliza la siguiente ecuación basado en la matriz de confusión (Véase la Figura 4.1):

VP: Verdaderos Positivos

FN: Falsos Negativos

FP: Falsos positivos

VN: Verdaderos Negativos

$$\text{Recall} = \left(\frac{VP}{VP + FN}\right) * 100 \tag{4.1}$$

$$\text{Recall} = \left(\frac{148}{148+2}\right) * 100 = 98{,}67\,\%$$

Con respecto a la ecuación 4.1 se obtuvo un recall de 98.67 %, indicando que el modelo detecta correctamente casi todos los verdaderos positivos de esta clase, con una pérdida menor.

Para calcular los verdaderos negativos en cada clase, se hace uso de la siguiente fórmula 4.2.

$$\text{VN} = 450 - (VP + FP + FN) = 450 - (148 + 0 + 2) = 300 \text{ - con arma} \tag{4.2}$$

Ahora, para el cálculo de la tasa de falsos positivos en la clase objetivo basado en la matriz de confusión, se hace uso de la ecuación 4.3.

$$\text{Tasa FP} = \left(\frac{FP}{FP + VN}\right) * 100 \tag{4.3}$$

$$\text{Tasa FP} = \frac{1}{3}\left(\frac{2}{2+300}\right) * 100 = 0{,}66\,\%$$

Para la precisión del modelo entrenado, se hace uso de la fórmula 4.4, obteniendo

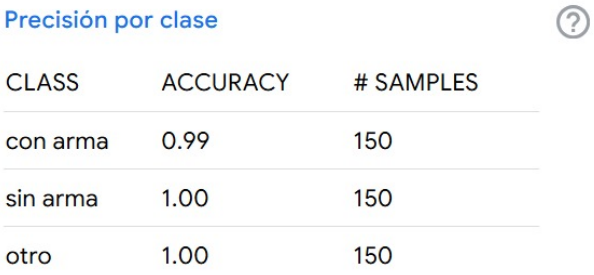
$$\text{Precisión} = \left(\frac{VP}{VP + FP}\right) * 100 \tag{4.4}$$

$$\text{Precisión} = \left(\frac{148}{148+0}\right) * 100 = 100\,\%$$

Basado en el resultado de la fórmula precisión, se obtiene el 100 %, indicando que las predicciones positivas detectadas por el modelo, son correctas.

En conjunto, esto muestra que el modelo es altamente confiable para identificar una arma blanca dentro de una imagen.

Además, la figura 4.2 indica que la exactitud de la clase `con arma` es del 99 %, capaz de detectar la mayoría de los casos positivos reales y realizar una clasificación correcta de esta clase.



Precisión por clase

CLASS	ACCURACY	# SAMPLES
con arma	0.99	150
sin arma	1.00	150
otro	1.00	150

Figura 4.2: Precisión del modelo

En la gráfica 4.3 se puede observar que en la época 17, el test de precisión se acerca a 1, y luego se mantiene constante, es decir, no hay un sobre ajuste del modelo al momento de ser entrenado, evitando su eficiencia y correcto balanceo.

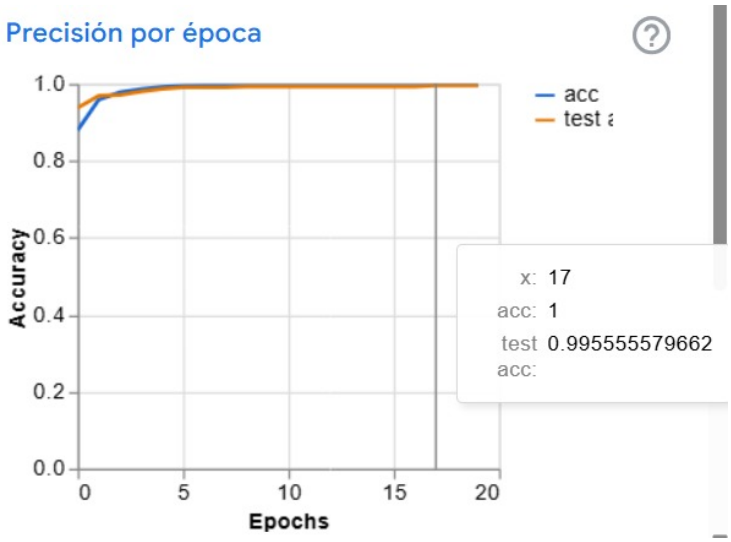


Figura 4.3: Precisión del modelo

F1-Score

$$\mathrm{F1} = 2 * \left(\frac{Precision * Recall}{Precision + Recall} \right) \tag{4.5}$$

$$\mathrm{F1} = 2 * \left(\frac{1 * 0{,}9867}{1 + 0{,}9867} \right) * 100 = 99{,}33\,\%$$

Para el cálculo de F1-Score se hace uso de la ecuación 4.5, indicando un modelo altamente eficiente en su entrenamiento según la clase `con arma`, logrando un gran balance para evitar en su mayoría falsos positivos y negativos, cabe mencionar que la iluminación es importante para la toma precisa de sus datos.

Resultados de la detección y clasificación de imágenes

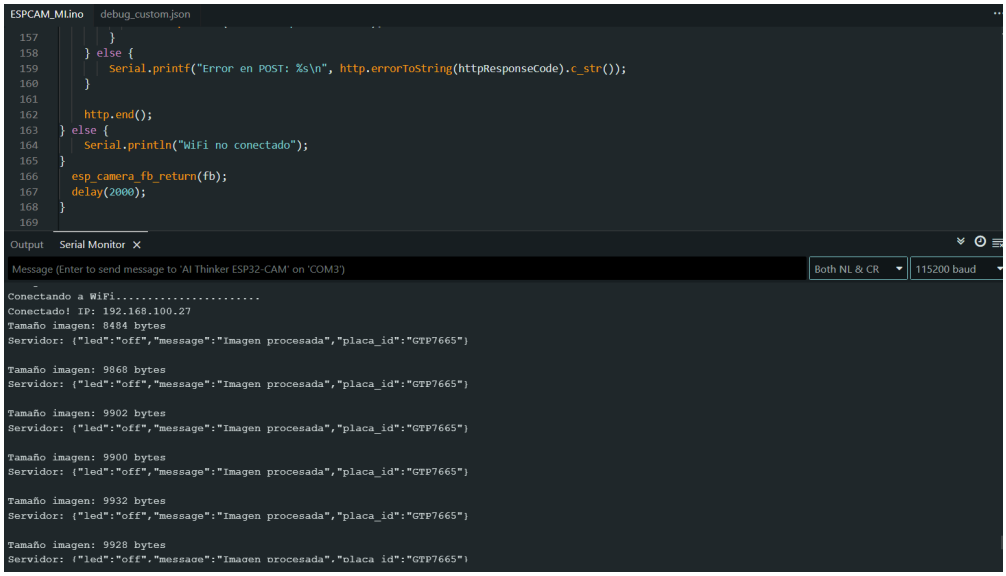


Figura 4.4: Envío de imágenes

En la Figura 4.4 se observa el proceso de envío de imágenes al servidor Flask desde la ESP32-CAM, la cual es puesta en cola antes de ser procesada para su posterior clasificación, asignándole como identificador único la placa del vehículo. Asimismo, el sistema realiza la captura periódica de imágenes en un intervalo de dos segundos, lo que permite un flujo continuo de datos para su análisis posterior.

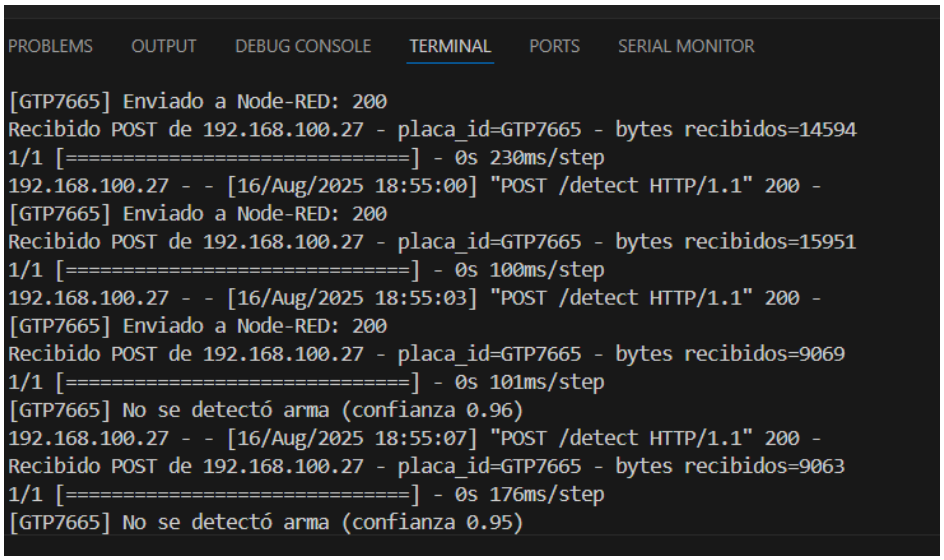


Figura 4.5: Procesamiento del servidor

En la Figura 4.5 se observa el procesamiento de las imágenes en el servidor Flask, junto con el tiempo de inferencia de imagen una vez detectada y clasificada con la categoría `con arma` es enviado al orquestador implementado en Node-RED, el cual se encarga de gestionar el envío de la alerta correspondiente al bot de Telegram. Además, para reforzar la señal de notificación de manera visual,

se activa brevemente el flash integrado en el microcontrolador.

4.2. Métrica para el objetivo 2

Para el segundo objetivo, que consiste en desarrollar un sistema de detección de frases amenazantes en tiempo real, se utilizó Scikit-learn debido a su rapidez y eficiencia durante la ejecución del sistema. La Figura 4.6 presenta la matriz de confusión, la cual refleja un desempeño excelente, mostrando de manera clara la correcta clasificación tanto de las frases consideradas agresivas como de aquellas no agresivas.

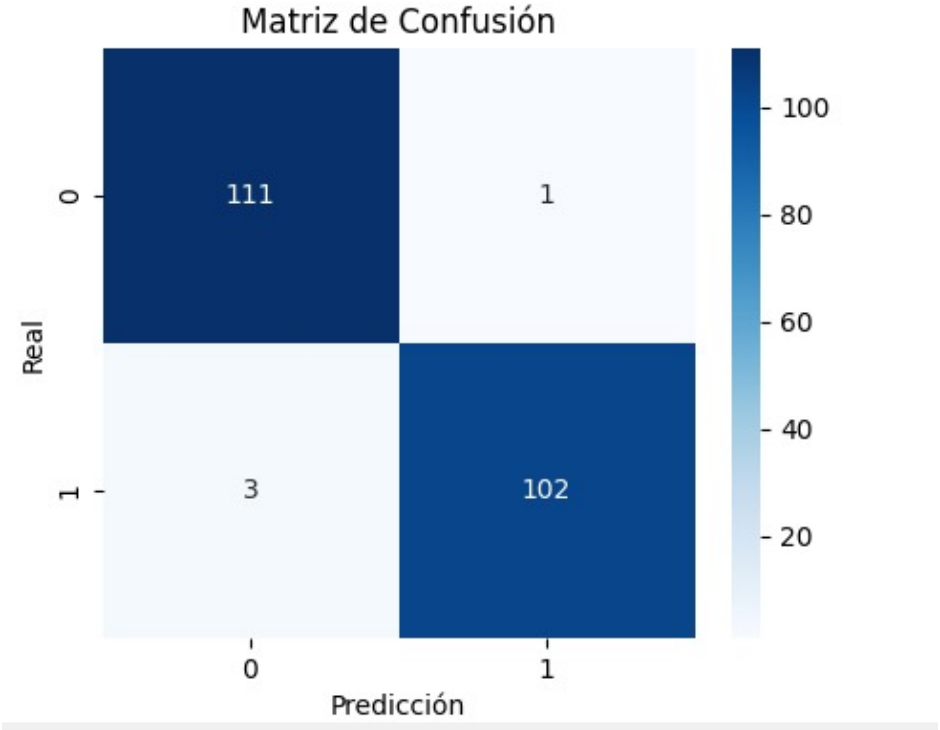


Figura 4.6: Matriz de confusion Scikit-learn

Donde se define:

- **TP:** frases amenazantes correctamente clasificadas como amenazantes = 102
- **FP:** frases no amenazantes clasificadas erróneamente como amenazantes = 1
- **TN:** frases no amenazantes correctamente clasificadas como no amenazantes = 111
- **FN:** frases amenazantes clasificadas erróneamente como no amenazantes = 3

A partir de estos valores, se calcularon las métricas de desempeño del modelo:

Precisión (Precision):

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{102}{102 + 1} \approx 0{,}9903 \approx 99{,}03\,\%$$

Sensibilidad / Recall (TPR):

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{102}{102 + 3} \approx 0{,}9714 \approx 97{,}14\,\%$$

Exactitud (Accuracy):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{102 + 111}{102 + 111 + 1 + 3} \approx 0{,}9815 \approx 98{,}15\,\%$$

F1-Score:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0{,}9903 \cdot 0{,}9714}{0{,}9903 + 0{,}9714} \approx 0{,}9807 \approx 98{,}07\,\%$$

Como se observa, el modelo muestra un alto desempeño en la identificación de frases amenazantes, logrando un equilibrio adecuado entre precisión y sensibilidad.

Resultados de la detección y clasificación de audios

Figura 4.7: Interfaz de resultados de audios guardados

En la Figura 4.7 se muestra la interfaz del sistema en el momento en que un audio es clasificado como agresivo. En este punto, el archivo se almacena junto con su respectiva etiqueta y se genera la alerta correspondiente. Tal como se indicó previamente, la alerta será enviada un total de tres veces con el fin de no saturar el tráfico de la red; sin embargo, las imágenes continuarán siendo registradas y almacenadas.

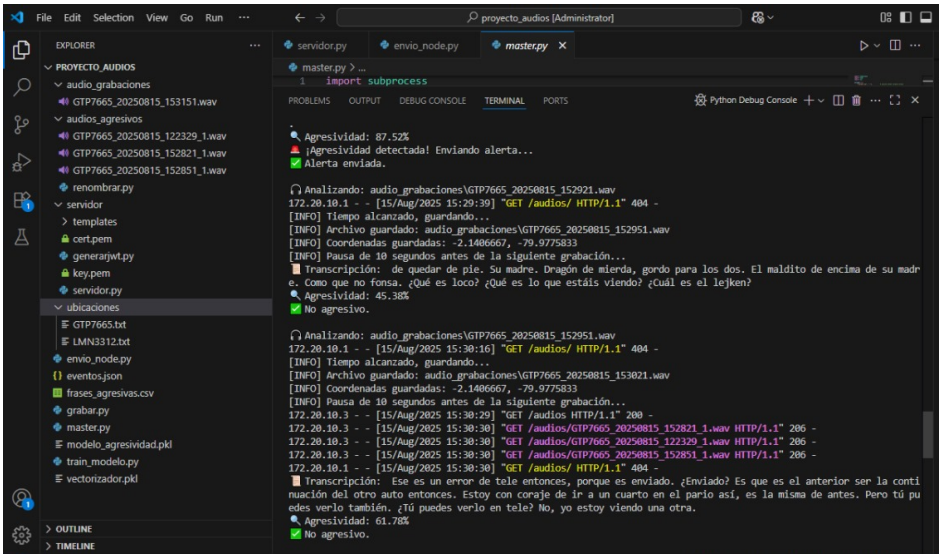


Figura 4.8: Resultados de Python

Posteriormente, en la Figura 4.8 se observa la salida en consola del programa desarrollado en Python. Allí se detalla el flujo del proceso: la recepción del audio, su almacenamiento, el análisis realizado por el modelo de clasificación y finalmente la confirmación del envío de la alerta.

4.3. Métrica para el objetivo 3

Para el tercer objetivo, relacionado con la implementación de un sistema basado en eventos que transmita información de manera autónoma ante posibles actos delictivos, se evaluará el tiempo en segundos, del procesamiento completo del sistema, dividido en fases, según la tabla 4.1, relacionada con la captura de imágenes.

N° Eventos	Captura de datos (s)	Procesamiento (s)	Envío de alertas (s)	Tamaño imagen (bytes)
#1	0.59	1.61	0.51	8051
#2	0.69	1.08	0.59	8060
#3	0.59	1.80	0.55	8065
#4	0.66	1.50	0.56	8060
#5	0.70	1.41	0.48	8045

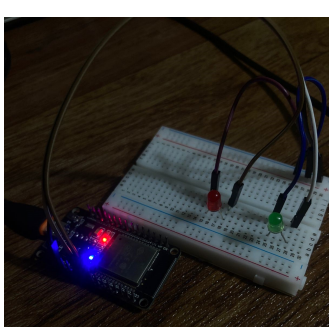
Tabla 4.1: Tiempo de procesamiento por fase

Estos datos fueron obtenidos de manera experimental, con la ayuda de un cronómetro y leds visuales para garantizar su precisión.

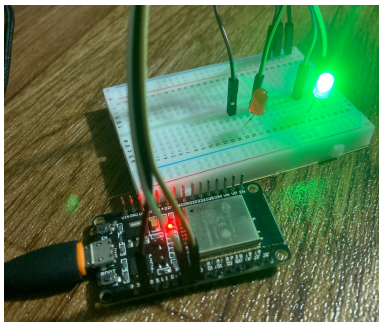
En la figura 4.9a se puede visualizar el encendido del led azul, cuando la imagen es enviada y recibida por el servidor de procesamiento.

En la segunda fase 4.9b, se observa el encendido del led verde cuando la imagen ha pasado por el modelo de IA, clasificándolo como arma blanca, enviándolo al orquestador de datos.

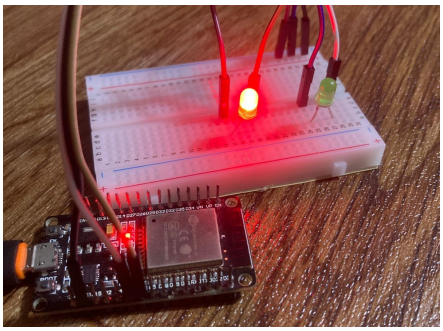
Por último, en la figura 4.9c, se observa el encendido del led rojo, cuando se generó la alerta automatizada y fue recibida por el bot de telegram, llegando el mensaje a las autoridades.



(a) Captura de datos



(b) Procesamiento



(c) Envío alertas

Figura 4.9: Procesamiento de datos visuales

Para el cumplimiento de latencia, se estableció un umbral de latencia por fase, como se detalla a continuación:

$T_{\text{fase 1}}$: 1 segundo.

$T_{\text{fase 2}}$: 1.50 segundos.

$T_{\text{fase 3}}$: 1 segundos.

Dando un total de 3.50 segundos desde la captura de los datos, hasta su envío de alertas al chat de Telegram.

Se calculará el SLA por fase, para determinar el valor máximo de tiempo aceptable para el procesamiento en cada tramo, siguiendo la siguiente ecuación 4.6.

$$\text{SLA}_{\text{lat}} = \frac{\#\{i : L_i \leq T_{\text{captura}}\}}{M} \times 100\,\% \tag{4.6}$$

$$\text{SLA}_{\text{fase1}} = \frac{5}{5} \times 100\,\% = 100\,\%$$

Cumpliendo el 100 % de los datos con la SLA de latencia, debido a que todos los eventos obtenidos, se encuentran por debajo del umbral establecido en la fase 1.

$$\text{SLA}_{\text{fase2}} = \frac{3}{5} \times 100\,\% = 60\,\%$$

Cumpliendo el 60 % de los datos con la SLA de latencia, debido a solo 3 eventos, se encuentran por debajo del umbral establecido en la fase 2.

$$\text{SLA}_{\text{fase3}} = \frac{5}{5} \times 100\,\% = 100\,\%$$

Cumpliendo el 100 % de los datos con la SLA de latencia, debido a que todos los eventos obtenidos, se encuentran por debajo del umbral establecido en la fase 3.

Entonces, si el umbral total fue de 3.5 segundos, según la suma de sus fases, el porcentaje de SLA de latencia se calcula sobre los eventos que cumplen con este límite. Contando con 13 eventos de 15 que cumplen con el umbral establecido.

$$\text{SLA}_{\text{lat}} = \frac{13}{15} \times 100\,\% = 86{,}67\,\%$$

Siendo este un porcentaje aceptable para el envío completo de las alertas por medio del bot de Telegram.

Alerta captada de arma blanca

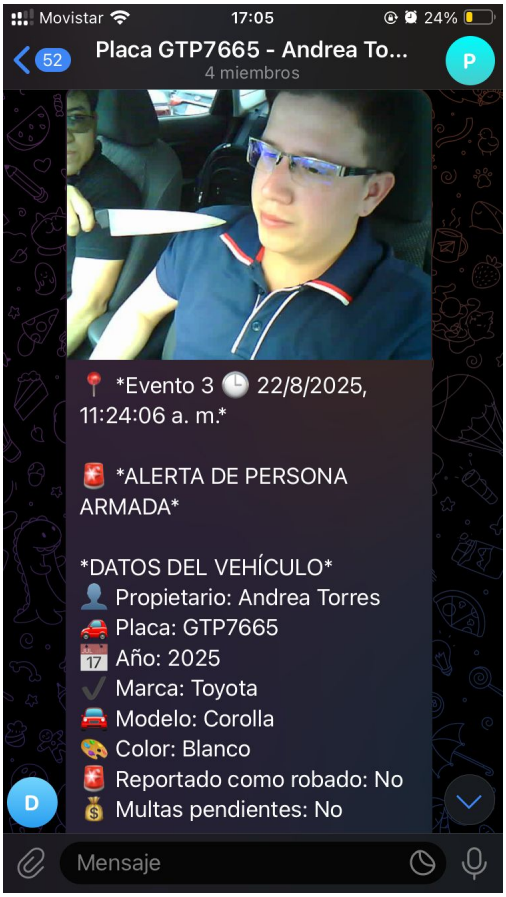


Figura 4.10: Notificación de alerta automatizada

En la figura 4.10 se visualiza la notificación de alerta en Telegram, correspondiente al chat según la placa del vehículo, donde se reflejan los datos necesarios del propietario e información relevante del vehículo, así como las coordenadas GPS necesarias para su localización.

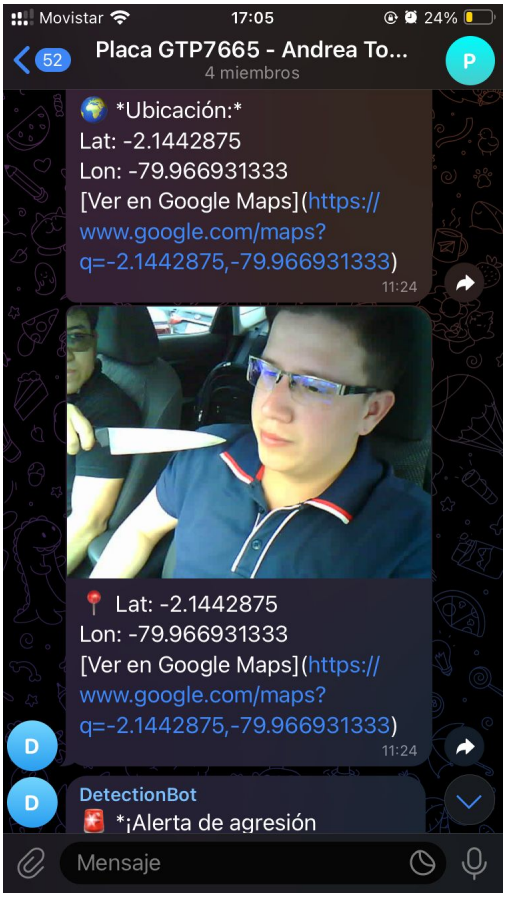


Figura 4.11: Actualización de GPS

Además, cada vez que el sistema detecte un arma blanca dentro de la imagen, se enviará a Telegram, junto con la constante actualización de las coordenadas por GPS, para llevar un control en tiempo real de su ubicación (Véase en la Figura 4.11).

Continuando con la evaluación de la Métrica 3, además de la parte de imágenes, también se consideró el procesamiento de audios de 20 segundos, como se muestra en la Tabla 4.2.

N° Evento	**Captura (s)**	**Procesamiento (s)**	**Envío de alerta (s)**
1	21.74	18.27	0.54
2	22.07	17.84	0.67
3	21.64	17.98	0.49
4	20.45	17.74	0.56
5	20.72	18.05	0.54
6	20.55	17.50	0.53

Tabla 4.2: Tiempo de procesamiento de audio por fase

Se definieron los siguientes umbrales por fase:

- **Captura:** 22 s
- **Procesamiento:** 18 s
- **Envío de alerta:** 1 s

Los SLA por fase se calcularon con la ecuación:

$$SLA_{lat} = \frac{\#\{i : L_i \leq T\}}{M} \times 100\,\%$$

donde L_i es el tiempo observado en el evento i, T es el umbral definido, y M el número total de eventos.

- **Fase 1 (captura):**

$$SLA_{fase1} = \frac{6}{6} \times 100\,\% = 100\,\%$$

- **Fase 2 (procesamiento):**

$$SLA_{fase2} = \frac{4}{6} \times 100\,\% = 66{,}67\,\%$$

- **Fase 3 (alerta):**

$$SLA_{fase3} = \frac{6}{6} \times 100\,\% = 100\,\%$$

Para el SLA total, considerando el umbral global de 41 segundos, 5 de los 6 eventos estuvieron dentro del límite:

$$SLA_{lat} = \frac{5}{6} \times 100\,\% = 83{,}33\,\%$$

Alerta audio agresivo

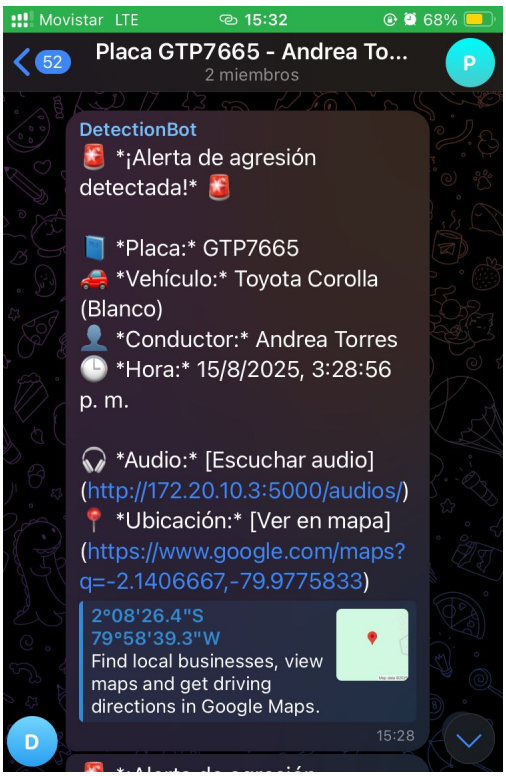


Figura 4.12: Mensaje creado a partir de alerta por audio

Por último, la Figura 4.12 presenta el mensaje generado por el bot de Telegram a partir de la detección del audio. Este mensaje es enviado de manera automática, facilitando la notificación en tiempo real del evento detectado y asociándolo con la persona o vehículo correspondiente.

CAPÍTULO 5

5. CONCLUSIONES Y LINEAS FUTURAS

5.1. Conclusiones

Conclusiones por objetivos

- **Objetivo 1:** Desarrollar un sistema de detección en tiempo real de armas blancas capaz de capturar y enviar imágenes a un servidor de procesamiento basado en IA. Se cumplió este objetivo mediante la implementación del módulo ESP32-CAM en conjunto con un modelo entrenado en Teachable Machine, alcanzando una detección precisa de armas blancas en escenarios controlados. Las imágenes capturadas fueron procesadas en el servidor y posteriormente enviadas como evidencia visual junto con los datos del vehículo al bot de Telegram, verificando la eficacia del flujo planteado.
- **Objetivo 2:** Desarrollar un sistema de detección de frases amenazantes en tiempo real, enviando los audios captados a un servidor de procesamiento basado en IA. Este objetivo se alcanzó utilizando un ESP32 con micrófono INMP441 para la captura de audio, que fue enviado al servidor para su transcripción mediante el modelo Whisper. Posteriormente, se aplicó un modelo de clasificación basado en Scikit-learn para identificar frases amenazantes, logrando la generación de alertas con datos del vehículo, ubicación y evidencia asociada. El desempeño obtenido en pruebas controladas fue satisfactorio, demostrando baja tasa de falsos positivos.
- **Objetivo 3:** Diseñar un sistema basado en eventos que, ante un posible acto delictivo, envíe automáticamente información del vehículo, tales como ubicación en tiempo real (coordenadas GPS), placa, modelo, año, propietario y antecedentes, con el fin de facilitar una intervención oportuna e inmediata de las autoridades. Este objetivo se cumplió al analizar los resultados tanto de imágenes como de audio, en donde se observa que el sistema cumple de manera satisfactoria en la captura de datos y el envío de alertas, alcanzando en ambas modalidades valores de SLA del 100 %. La mayor limitación se encuentra en la fase de procesamiento,

tanto en imágenes (60 %) como en audio (66.67 %), lo que indica que esta etapa requiere mayor optimización para garantizar una mayor consistencia. En términos globales, el sistema logra un SLA de 86.67 % en imágenes y 83.33 % en audio, lo cual refleja un nivel aceptable de desempeño para un sistema de alertas autónomo en tiempo casi real, manteniendo la integridad y rapidez en la transmisión de la información de seguridad.

Conclusión general del proyecto

En conclusión, el proyecto logró implementar un sistema de seguridad vehicular inteligente que integra tecnologías IoT e inteligencia artificial para la detección de armas blancas y frases amenazantes, así como para el envío de información en tiempo real a las autoridades mediante Node-RED y Telegram. Los resultados demostraron que la arquitectura planteada es viable y efectiva, permitiendo una detección autónoma sin necesidad de intervención humana. De esta manera, se evidencia el potencial de esta solución como herramienta de apoyo en la seguridad ciudadana, aportando al desarrollo de sistemas tecnológicos innovadores que pueden complementar las estrategias de prevención y respuesta en entornos de alta inseguridad.

5.2. Recomendaciones

Para proyectos futuros

El sistema implementado demostró ser funcional en escenarios controlados; sin embargo, se recomienda ampliar las capacidades de los modelos de inteligencia artificial mediante el uso de bases de datos más extensas y variadas. Esto permitirá que la detección de armas blancas y frases amenazantes sea más robusta frente a cambios de iluminación, ruido ambiental y diferentes acentos o expresiones locales. Asimismo, se sugiere trabajar en la optimización de los modelos ya implementados, con el fin de mejorar la eficiencia en el tiempo de respuesta y reducir el consumo de recursos computacionales.

Otra línea importante de mejora está relacionada con la infraestructura. La incorporación de microcontroladores con mayor capacidad de procesamiento o el uso de servidores en la nube especializados en inteligencia artificial permitiría escalar el sistema y garantizar un rendimiento estable en escenarios con múltiples vehículos monitoreados de manera simultánea.

Para implementación práctica

Se recomienda establecer un plan de mantenimiento periódico que contemple tanto el estado físico de los dispositivos (cámaras, micrófonos y módulos GPS) como la actualización de los modelos de IA. Esto asegurará que el sistema conserve su precisión a lo largo del tiempo y que no se vea afectado

por el desgaste de los componentes.

Adicionalmente, se recomienda fortalecer la seguridad operativa del sistema mediante el uso de controles de acceso y autenticación de usuarios que administren el entorno. Del mismo modo, se sugiere implementar registros de auditoría que permitan monitorear el uso del sistema y detectar intentos de manipulación o uso indebido. Finalmente, sería conveniente definir protocolos claros para la gestión y almacenamiento de datos sensibles, asegurando su resguardo, anonimización cuando sea necesario y cumplimiento con la normativa vigente en materia de protección de datos

APÉNDICES

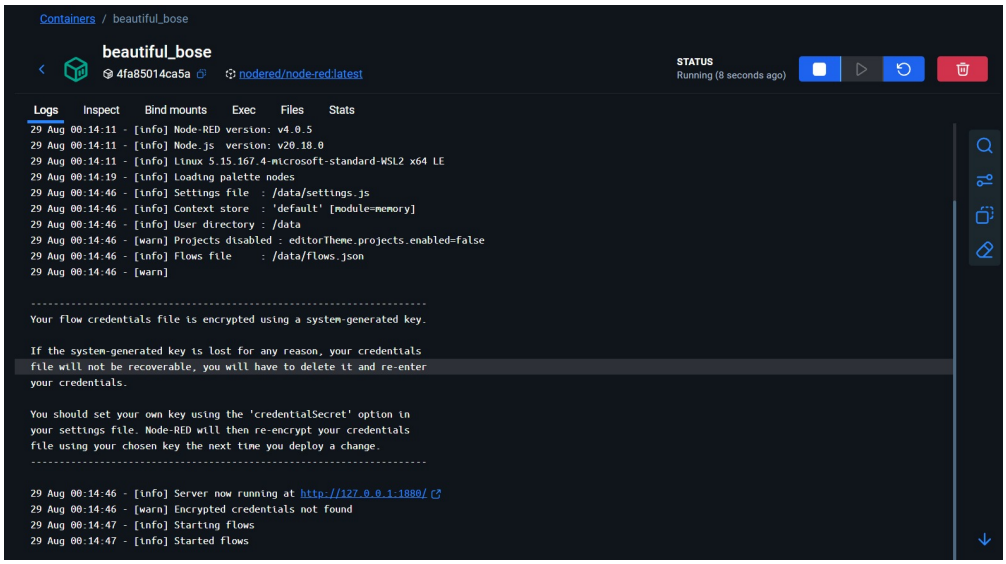


Figura 1: Dockerización del sistema

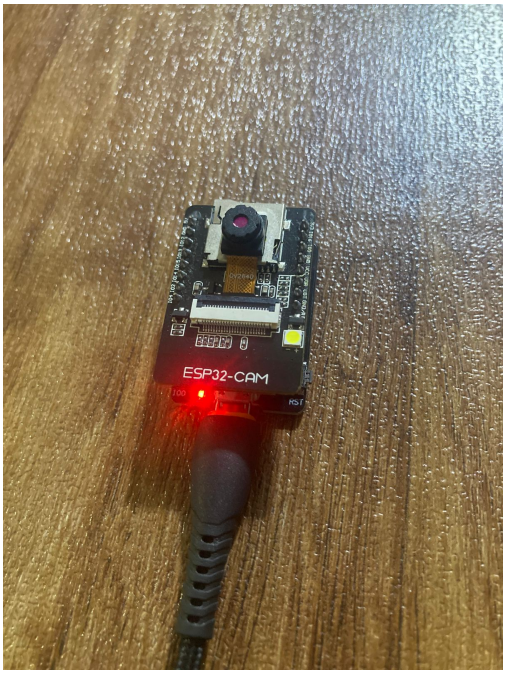


Figura 2: ESP32 CAM

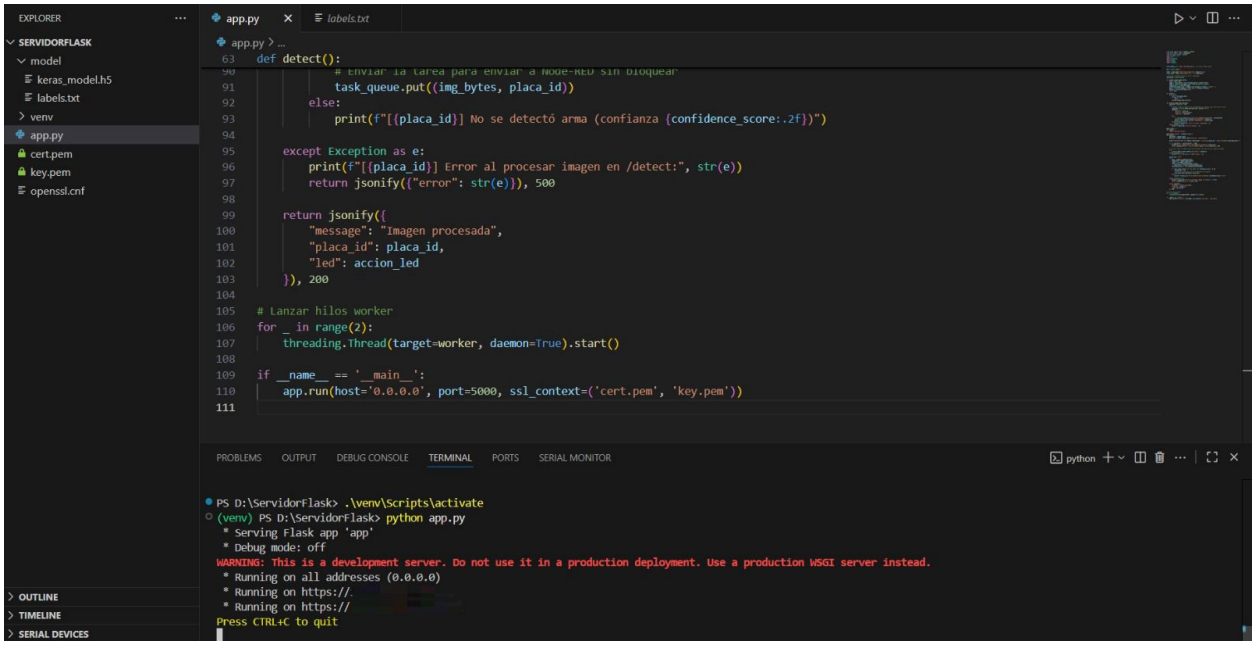


Figura 3: Servidor de procesamiento iniciado

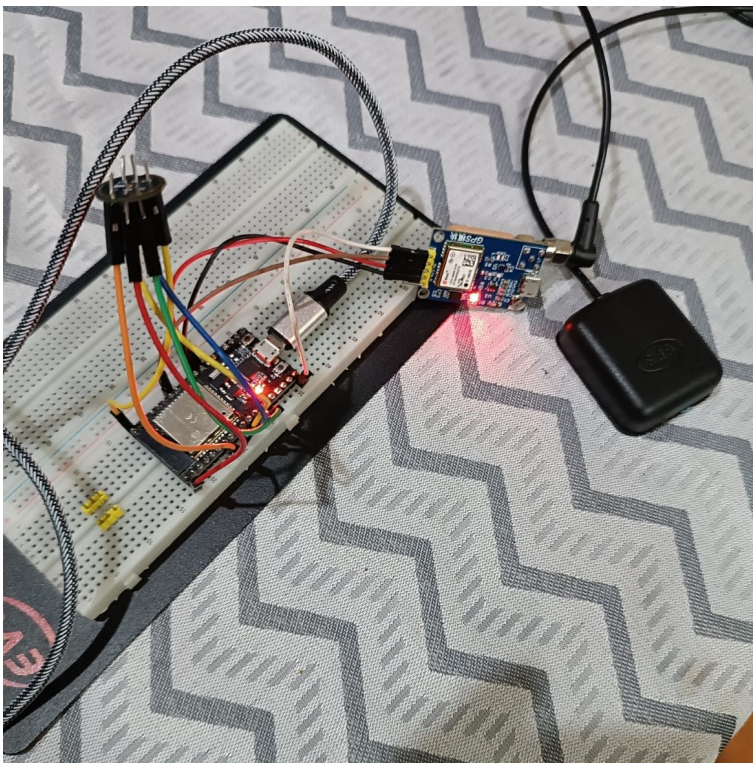


Figura 4: Circuito para capturar audio y gps

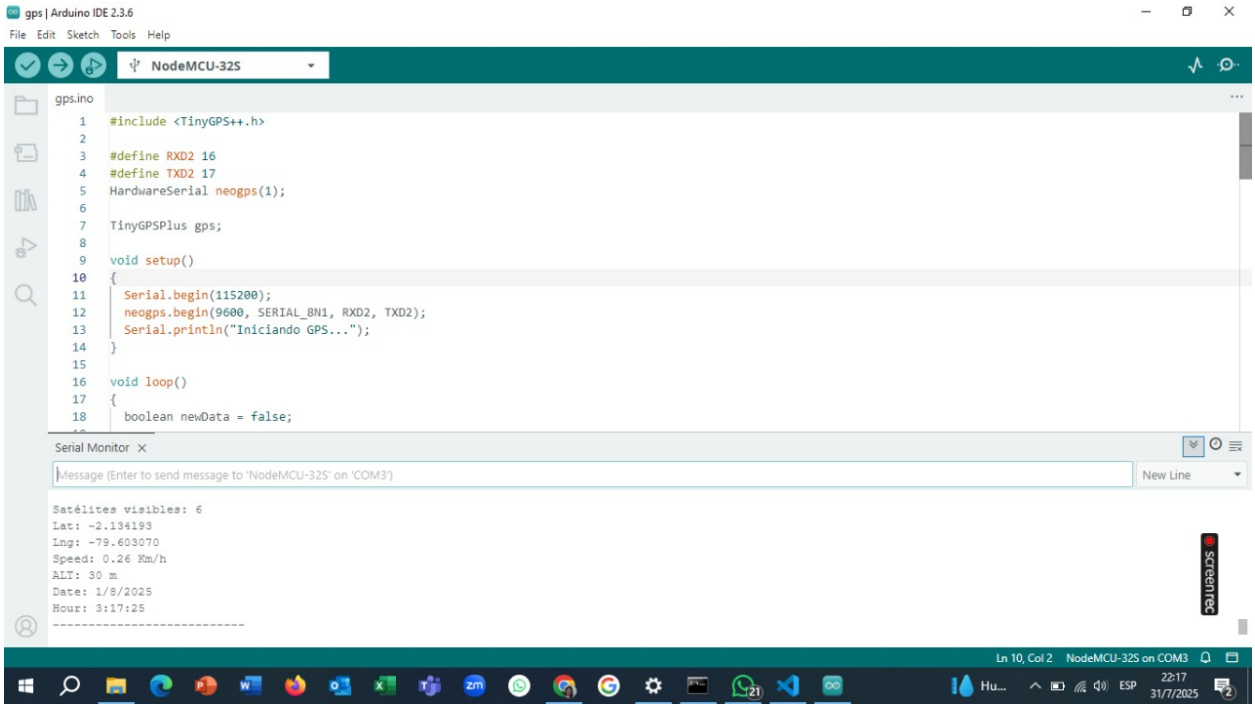


Figura 5: Datos captados por el gps

BIBLIOGRAFÍA

AL-Dhief, F. T., Sabri, N., Latiff, N. A., Malik, N., Abbas, M., Albader, A., Mohammed, M. A., AL-Haddad, R. N., Salman, Y. D., Khanapi, M., et al. (2018). Performance comparison between TCP and UDP protocols in different simulation scenarios. *International Journal of Engineering & Technology*, *7*(4.36), 172-176.

Alpaydin, E. (2021, agosto). *Machine Learning, revised and updated edition* [Google-Books-ID: Eyk5EAAAQBAJ]. MIT Press.

Chitravanshi, D., Malik, A., Saini, H., Avasthi, S., Agrawal, K., & Grover, Y. (2024). Weapon Detection from images using YOLO and OpenCV. *2024 First International Conference on Technological Innovations and Advance Computing (TIACOMP)*, 560-565. https://doi.org/10.1109/TIACOMP64125.2024.00098

Corrales Barragán, D. W. (2020). Las cámaras y botones de pánico de seguridad ciudadana en el transporte público masivo en Quito, 2016 – 2018. Consultado el 4 de junio de 2025, desde http://repositorio.iaen.edu.ec/handle/24000/6197

Crisgar, P. V., Wijaya, P. R., Pakpahan, M. D. F., Syamsuddin, E. Y., & Hasanuddin, M. O. (2021). GPS-Based Vehicle Tracking and Theft Detection Systems using Google Cloud IoT Core & Firebase. *2021 International Symposium on Electronics and Smart Devices (ISESD)*, 1-6. https://doi.org/10.1109/ISESD53023.2021.9501928

Diario Correo. (2022, junio). *Pese a que el 80 % de taxis sí tiene cámaras, ninguno cuenta con la asistencia del ECU-911*. Diario Correo. https://diariocorreo.com.ec/71488/ciudad/pese-a-que-el-80--de-taxis-si-tiene-camaras-ninguno-cuenta-con-la-asistencia-del-ecu-911

Directo, E. E. (2025, mayo). Sector del transporte amenazado por la creciente inseguridad - Ecuador en Directo. Consultado el 5 de junio de 2025, desde https://ecuadorendirecto.com/2025/05/27/sector-del-transporte-amenazado-por-la-creciente-inseguridad/

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (2022). HTTP/1.1. *Hypertext Transfer Protocol-HTTP/1.1*.

Flask. (2010). Welcome to Flask — Flask Documentation (3.1.x). Consultado el 2 de julio de 2025, desde https://flask.palletsprojects.com/en/stable/

García Torres, I., Castillo León, R. E., Domínguez de la Torre, J., & Parra López, R. A. (2020). Sistema de alerta usando Módulo de Reconocimiento de voz para detectar problemas de robo de vehículos. *Journal of business and entrepreneurial studies: JBES*, *4*(1 (Enero - Junio)), 20. Consultado el 29 de mayo de 2025, desde https://dialnet.unirioja.es/servlet/articulo?codigo=7472735

Google. (2019). Teachable Machine. Consultado el 7 de junio de 2025, desde https://teachablemachine.withgoogle.com/

Goyal, S. B., Bedi, P., Kumar, J., & Ankita. (2022). Realtime Accident Detection and Alarm Generation System Over IoT. En R. Kumar, R. Sharma & P. K. Pattnaik (Eds.), *Multimedia Technologies in the Internet of Things Environment, Volume 2* (pp. 105-126, Vol. 93). Springer Singapore. https://doi.org/10.1007/978-981-16-3828-2_6

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645-1660. https://doi.org/10.1016/j.future.2013.01.010

Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). Design and Implementation of ESP32-Based IoT Devices. *Sensors*, *23*(15), 6739. https://doi.org/10.3390/s23156739

Husni, E., & Hasibuan, F. (2018). Driver Supervisor System with Telegram Bot Platform. En N. T. Nguyen, E. Pimenidis, Z. Khan & B. Trawiński (Eds.), *Computational Collective Intelligence* (pp. 436-444, Vol. 11055). Springer International Publishing. https://doi.org/10.1007/978-3-319-98443-8_40

Jani, M., Fayyad, J., Al-Younes, Y., & Najjaran, H. (2023). Model Compression Methods for YOLOv5: A Review. https://doi.org/10.48550/ARXIV.2307.11904

Jemai, F., Hayouni, M., & Baccar, S. (2021). Sentiment Analysis Using Machine Learning Algorithms. *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 775-779. https://doi.org/10.1109/IWCMC51323.2021.9498965

Jones, M., Bradley, J., & Sakimura, N. (2015). *Json web token (jwt)* (inf. téc.).

Kramer, O. (2016). Scikit-learn. En *Machine learning for evolution strategies* (pp. 45-53). Springer.

Kumar, E. (2013a, diciembre). *Artificial Intelligence*. I. K. International Pvt Ltd.

Kumar, E. (2013b, diciembre). *Natural Language Processing* [Google-Books-ID: FpUBFNFuKWgC]. I. K. International Pvt Ltd.

Laínez Apolinario, W. P., & Navarrete Buste, G. E. (2024). *Implementación de un Prototipo de Monitoreo para la Visualización en Tiempo Real de Vehículos utilizando GPS y ESP32* [bachelorThesis]. Consultado el 6 de junio de 2025, desde http://dspace.ups.edu.ec/handle/123456789/29212

Lee, E. A. (2008). Cyber Physical Systems: Design Challenges. *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 363-369. https://doi.org/10.1109/ISORC.2008.25

Machine, T. (2024). Teachable machine. *Google. Accessed: Feb*, *22*.

McKinney, W., et al. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, *14*(9), 1-9.

Medina-Pérez, A., Sánchez-Rodríguez, D., & Alonso-González, I. (2021). An Internet of Thing Architecture Based on Message Queuing Telemetry Transport Protocol and Node-RED: A Case Study for Monitoring Radon Gas. *Smart Cities*, *4*(2), 803-818. https://doi.org/10.3390/smartcities4020041

Montalvo, D. (2023). Ecuador registra los niveles más altos de crimen, inseguridad y delincuencia del continente. Consultado el 3 de junio de 2025, desde https://www.participacionciudadana.org/web/wp-content/uploads/2024/02/A1-Ecuador-registra-los-niveles-mas-altos-de-crimen.pdf

Moumen, I., Rafalia, N., Abouchabaka, J., & Aoufi, M. (2023). Real-time GPS Tracking System for IoT-Enabled Connected Vehicles (S. Bourekkadi, M. Kerkeb, O. El Imrani, N. Rafalia, O. Zubareva, S. Khoulji & J. Abouchabaka, Eds.). *E3S Web of Conferences*, *412*, 01095. https://doi.org/10.1051/e3sconf/202341201095

Parlamento. (2016). Reglamento a Ley Orgánica de Protección de Datos Personales – Ministerio de Telecomunicaciones y de la Sociedad de la Información. Consultado el 7 de junio de 2025, desde https://www.telecomunicaciones.gob.ec/ley-y-reglamento-de-la-ley-de-proteccion-de-datos-personales/

Pavlov, T., & Mirceva, G. (2022). Covid-19 fake news detection by using bert and roberta models. *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, 312-316.

Peraza Siqueiros, G. F. (2006). *Introducción a la teoría de colas y su simulación* [Tesis de Licenciatura]. Universidad de Sonora.

Primicias. (2023, abril). Ni las aplicaciones móviles evitan asaltos a taxistas en Guayaquil. Consultado el 3 de junio de 2025, desde https://www.primicias.ec/noticias/sucesos/aplicaciones-taxistas-asaltos-guayaquil-inseguridad/

Rojas, L., Celso, E., Lopez, S., John, W., Quispe, H., & Maycon, J. (2017). Creación de un sistema de seguridad vehicular mediante bloqueo electrónico vía wifi. Consultado el 4 de junio de 2025, desde http://repositorio.avansys.edu.pe/handle/AVANSYS/24

Salama, R., Al-Turjman, F., Aeri, M., & Yadav, S. P. (2023). Internet of Intelligent Things (IoT) – An Overview. *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, 801-805. https://doi.org/10.1109/CICTN57981.2023.10141157

Shin, W. (2020). Implementation of Cough Detection System Using IoT Sensor in Respirator. *International journal of advanced smart convergence*, *9*(4), 132-138. https://doi.org/10.7236/IJASC.2020.9.4.132

Soni, A. A. (2025). Improving Speech Recognition Accuracy Using Custom Language Models with the Vosk Toolkit. *arXiv preprint arXiv:2503.21025*.

Spiller, T. R., Rabe, F., Ben-Zion, Z., Korem, N., Burrer, A., Homan, P., Harpaz-Rotem, I., & Duek, O. (2023). Efficient and accurate transcription in mental health research-A tutorial on using whisper AI for audio file transcription. *OSF Preprints*.

Telegram. (2025). Bots: Introduction for Developers. Consultado el 20 de agosto de 2025, desde https://core.telegram.org/bots

TensorFlow Developers. (2025, julio). TensorFlow. https://doi.org/10.5281/ZENODO.4724125

Valinejadshoubi, M., Moselhi, O., Bagchi, A., & Salem, A. (2021). Development of an IoT and BIM-based automated alert system for thermal comfort monitoring in buildings. *Sustainable Cities and Society*, *66*, 102602. https://doi.org/10.1016/j.scs.2020.102602

Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212-3232. https://doi.org/10.1109/TNNLS.2018.2876865