

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Electrónica

LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de control

AUTOR: Jarod Vinicio Zamora Medina

TUTOR: Ing. Jesús Israel Guamán Molina,Mg.

Ambato - Ecuador

febrero – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Jarod Vinicio Zamora Medina, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2025.

Firmado electrónicamente por:
JESUS ISRAEL GUAMAN MOLINA

Ing. Jesús Israel Guamán Molina,Mg.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2025.



Jarod Vinicio Zamora Medina

C.C. 1600530511

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2025.



Jarod Vinicio Zamora Medina

C.C. 1600530511

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Jarod Vinicio Zamora Medina, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, febrero 2025.

Firmado electrónicamente por:
FRANKLIN OSWALDO
MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
GUSTAVO DANILO
MACHADO MERINO

Ing. Gustavo Danilo Machado Merino MSc.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
JUAN PABLO PALLO
NORONA

Ing. Juan Pablo Pallo Noroña Mg.
PROFESOR CALIFICADOR

DEDICATORIA

A mis padres, pilares fundamentales en mi vida, cuyo apoyo incansable y sabios consejos me han guiado por este largo camino. Su esfuerzo y sacrificio son el motor que me motiva a seguir adelante día a día, les dedico este logro, que no es solo mío, sino de todas las personas que me han demostrado que con esfuerzo y dedicación se puede lograr todo.

Jarod

AGRADECIMIENTO

A mis padres, cuyo amor, sacrificio y enseñanzas han sido mi mayor fortaleza. Gracias por ser el ejemplo de perseverancia y dedicación que me ha inspirado a alcanzar esta meta. A mi familia y amigos, por brindarme su compañía, aliento y confianza en los momentos más desafiantes.

A mis fieles perritos, quienes con su cariño incondicional y su compañía silenciosa pero reconfortante me recordaron que incluso en los días más difíciles siempre hay motivos para sonreír.

A mis docentes de la carrera de Telecomunicaciones, quienes con su paciencia, conocimientos y guía contribuyeron enormemente a mi formación profesional. Su esfuerzo y compromiso han sido un pilar fundamental en este logro.

A la Universidad Técnica de Ambato, por abrirme las puertas al conocimiento, por ser el espacio donde crecí como persona y como profesional, y por brindarme las herramientas necesarias para hacer realidad este proyecto.

A todos ustedes, mi más sincero agradecimiento. Este trabajo es el resultado del esfuerzo conjunto y del apoyo constante que he recibido a lo largo de esta etapa. Gracias por ser parte de este logro especial para mi.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA ..... **i**

APROBACIÓN DEL TUTOR ..... **ii**

AUTORÍA ..... **iii**

DERECHOS DE AUTOR ..... **iv**

APROBACIÓN DEL TRIBUNAL DE GRADO ..... **v**

DEDICATORIA ..... **vi**

AGRADECIMIENTO ..... **vii**

ÍNDICE GENERAL DE CONTENIDOS ..... **viii**

ÍNDICE DE TABLAS ..... **xii**

ÍNDICE DE FIGURAS ..... **xiv**

ÍNDICE DE ANEXOS ..... **xix**

RESUMEN EJECUTIVO ..... **xx**

ABSTRACT ..... **xxi**

CAPÍTULO I. MARCO TEÓRICO **1**

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 2

1.3 Fundamentación teórica 3

1.3.1 Tráfico urbano (Ordenanza y normativa para el tráfico vehicular de la ciudad de Ambato)…. 4

1.3.2 Ruido ambiental y salud (Frecuencias, normativa y estándares mundiales y del Ecuador).. 5

1.3.3 Redes inalámbricas de sensores 8

1.3.4 Tecnologías de comunicación inalámbrica 9

1.3.5 Sistemas de monitoreo ambiental 10

1.3.6 Sensores de ruido ambiental 11

1.3.7 Cámaras web 15

1.3.8 Módulos para la comunicación inalámbrica 18

1.3.9 Microordenadores 22

1.3.10 Procesamiento de señales y datos 27

1.3.11 Protocolos para el envió de datos inalámbricos 28

1.3.12 Inteligencia artificial y aprendizaje automático 32

1.3.13 Bases de datos 37

1.3.14 Visualización grafica de datos almacenados 40

1.4 Objetivos 44

1.4.1 Objetivo general 44

1.4.2 Objetivos específicos 44

CAPÍTULO II. METODOLOGÍA 45

2.1 Materiales 45

2.2 Métodos 46

2.2.1 Modalidad de la investigación 46

2.2.2 Recolección de información..... 46

2.2.3 Procesamiento y análisis de datos 47

CAPÍTULO III. RESULTADOS Y DISCUSIÓN..... 48

3.1 Introducción 48

3.2 Estado actual del tráfico vehicular en la ciudad de Ambato 48

3.3 Situación del ruido actual en la ciudad de Ambato..... 51

3.4 Materiales seleccionados..... 53

3.5 Esquema general del sistema 54

3.6 Esquema de comunicación del sistema 55

3.7 Entrenamiento y programación para el análisis del tráfico 56

3.7.1 Desarrollo del sistema de detección de vehículos..... 56

3.7.2 Justificación de la selección de la versión YOLO 57

3.7.3 Justificación de la variante de YOLOv8 58

3.7.4 Descripción de la base de datos de COCO..... 58

3.7.5 Descripción del dataset aumentado con roboflow 59

3.7.6 Entrenamiento 60

3.7.7 Resultados del Entrenamiento del Modelo 67

3.7.8 Implementación del código para la detección de los vehículos 74

3.7.9 Resultado del trafico 76

3.8 Ruido 78

3.8.1 Procesamiento y Envío de Datos del Ruido 78

3.8.2 Diseño de la PCB para detección de ruido 80

3.8.3 Análisis y Procesamiento Avanzado del Audio Capturado 86

3.9 Almacenamiento de datos en MySQL 95

3.10 Representación de los datos recolectados en node-red 98

3.11 Presupuesto 109

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 111

4.1 Conclusiones 111

4.2 Recomendaciones 112

REFERENCIAS BIBLIOGRÁFICAS 114

ANEXOS 119

ÍNDICE DE TABLAS

Tabla 1. Comparación de los distintos sensores de ruido 14

Tabla 2. Comparación de cámaras web..... 18

Tabla 3. Comparación de los módulos de comunicación inalámbrica..... 21

Tabla 4. Comparación de los microrordenadores..... 25

Tabla 5. Comparación de los protocolos..... 31

Tabla 6. Comparación de los distintos algoritmos..... 35

Tabla 7. Comparación de las versiones de YOLO..... 36

Tabla 8. Comparación de las variantes de YOLOv8..... 36

Tabla 9. Comparación de las bases de datos..... 39

Tabla 10. Comparación de herramientas de visualización grafica..... 43

Tabla 11. Recolección de Información 47

Tabla 12. Lugares con mayor afluencia de tráfico vehicular..... 50

Tabla 13. Ruido ambiental medido en puntos críticos de la ciudad de Ambato..... 52

Tabla 14. Lista de materiales para la implementación del prototipo 53

Tabla 15. Librerías principales para el sistema..... 57

Tabla 16. Justificación de la versión de Yolo 57

Tabla 17. Librerías para el entrenamiento..... 61

Tabla 18. Datos del proceso en el video analizado..... 76

Tabla 19. Procesamiento general de la captura del audio..... 79

Tabla 20. Componentes para la realización de la placa electrónica..... 80

Tabla 21. Datos 1 recolectados 97

Tabla 22. Análisis de los datos resultantes en el análisis 108

Tabla 23. Lista de materiales y sus precios 109

ÍNDICE DE FIGURAS

Figura 1. Trafico urbano 5

Figura 2. Ruido ambiental y salud 8

Figura 3. Redes inalámbricas de sensores 9

Figura 4. Tecnologías de comunicación inalámbrica 10

Figura 5. Modulo LM393 12

Figura 6. Modulo Max4466 12

Figura 7. Modulo Max9814 13

Figura 8. Modulo ICS-43434 14

Figura 9. Cámara klip xtreme 720 HD 16

Figura 10. Cámara microsoft 720 HD 30 FPS 16

Figura 11. Cámara jinxuny USB 2.0 480P 17

Figura 12. Modulo esp32 19

Figura 13. Modulo HC-05 20

Figura 14. Modulo Xbee S2 20

Figura 15. Modulo RA-02 21

Figura 16. Raspberry Pi 4 23

Figura 17. Microordenador BeagleBoard black 24

Figura 18. Microordenador Odroid-C4 24

Figura 19. Microordenador NVIDIA Jetson Nano 25

Figura 20. Diagrama general del procesamiento de señales 28

Figura 21. Arquitectura MQTT-Broker 29

Figura 22. Protocolo HTTP 29

Figura 23. Protocolo CoAP 30

Figura 24. Protocolo LoRaWAN 31

Figura 25. Inteligencia artificial 33

Figura 26. Detección de vehículos con yolo 34

Figura 27. Detección de objetos con SSD 34

Figura 28. Detección de imagen en Faster R-CNN 35

Figura 29 . Base de datos MySQL 37

Figura 30. Base de datos PostgreSQL 38

Figura 31. Base de datos SQLite 39

Figura 32. Interfaz de Node-RED 41

Figura 33. Visualización Grafana 41

Figura 34. Visualización en tableau 42

Figura 35. Tráfico en la zona céntrica de Ambato 49

Figura 36. Ruido ambiental vehicular y comercio en la ciudad de Ambato 53

Figura 37. Esquema general del sistema a implementar 55

Figura 38. Conexión del sistema 56

Figura 39. Proceso de utilización del dataset COCO con Yolov8x 59

Figura 40. Flujograma de proceso para dataset añadido con roboflow 60

Figura 41. Flujograma para la creación del archivo yaml 62

Figura 42. Parámetros para el entrenamiento 63

Figura 43. Ajuste del formato de imágenes en roboflow 64

Figura 44. Flujograma para el entrenamiento del modelo 66

Figura 45. Proceso de Entrenamiento 67

Figura 46. Matriz de confusión del entrenamiento 68

Figura 47. Curva de confianza del entrenamiento 69

Figura 48. Distribución de etiquetas del entrenamiento 70

Figura 49. Correlograma de etiquetas del entrenamiento 71

Figura 50. Curva de Precisión y confianza del entrenamiento 72

Figura 51. Curva de recall-confianza del entrenamiento 73

Figura 52. Loss and metrics del entrenamiento 74

Figura 53. Flujograma de la implementación del sistema de detección 75

Figura 54. Vehículos detectados con id sort 77

Figura 55. Proceso de detección de vehículos 77

Figura 56. Nivel de confiabilidad del sistema de detección 78

Figura 57. Diseño de la pcb en proteus 81

Figura 58. Impresión de pistas en la baquelita 82

Figura 59. Atacado químico de la placa 82

Figura 60. Perforación de los puntos de conexión de la placa 83

Figura 61. Colocación de espadines y otros implementos para la placa 83

Figura 62. Posicionamiento de los implementos en la placa final 83

Figura 63. Recorte de las piezas en acrilico 84

Figura 64. Armado del case protector 85

Figura 65. Case terminado vista superior 85

Figura 66. Case terminado vista lateral derecha 86

Figura 67. Case terminado vista enfrente superior 86

Figura 68. Señal de audio original 88

Figura 69. Histograma de amplitudes del audio original 89

Figura 70. Espectro de frecuencias del fragmento analizado 90

Figura 71. Espectro de frecuencias filtrado 91

Figura 72. Señal reconstruida después del filtrado 92

Figura 73. Nivel de presión sonora en decibelios 93

Figura 74. Comparación de los niveles de ruido 95

Figura 75. Formato de la tabla creada para ingreso de datos 96

Figura 76. Estructura de la base de datos 97

Figura 77. Datos 1 recolectados en la mañana 97

Figura 78. Datos 1 recolectados al medio día 98

Figura 79. Datos 1 recolectados en la tarde 98

Figura 80. Entorno principal de node-red 99

Figura 81. Sección del archivo habilitada 99

Figura 82. Creación del hash para contraseña 100

Figura 83. Colocación del hash en el archivo de configuración 100

Figura 84. Login para el ingreso a node-red 101

Figura 85. Flujo general para la visualización de los datos 103

Figura 86. Selección del día y periodo analizado 104

Figura 87. Coordenadas del lugar analizado 104

Figura 88. Ruido en el periodo analizado 105

Figura 89. Vehículos analizados en el periodo seleccionado 105

Figura 90. Nivel de ruido promedio en el lapso seleccionado 106

Figura 91. Nivel bajo en el lapso seleccionado 106

Figura 92. Nivel máximo en el lapso seleccionado 106

Figura 93. Promedio de vehículos detectados en el lapso seleccionado 107

Figura 94. Visión general del dashboard implementado 108

ÍNDICE DE ANEXOS

Anexo A. Pines y Datasheet del ESP32s 119

Anexo B. Datasheet del sensor Max9814 121

Anexo C. Datasheet del módulo PCM1808 125

Anexo E. COCO datasheet..... 128

Anexo F. Código en Python para detección de vehículos con Yolov8x 129

Anexo G. Código en Arduino IDE para la grabación de audio 132

Anexo H. Código para procesamiento de video y audio hacia la base de datos 133

Anexo I. Datos recolectados en la base de datos 136

RESUMEN EJECUTIVO

El presente proyecto desarrolla un sistema de monitoreo inteligente del tráfico vehicular y los niveles de ruido en la avenida Los Chasquis mediante tecnologías de procesamiento embebido e inteligencia artificial. La metodología se estructura en cuatro etapas principales.

En la primera etapa, se recopilan datos utilizando una cámara de video y un sensor de audio MAX9814 conectado a un ESP32. La señal de audio se transmite a una Jetson Nano, donde se realiza el procesamiento conjunto de video y audio.

La segunda etapa implementa el modelo de detección de objetos YOLOv8x para el conteo y clasificación precisa de vehículos a partir de los datos de video. Al mismo tiempo, el análisis del audio permite medir los niveles de ruido en decibeles y clasificarlo en categorías como bajo, medio o alto.

En la tercera etapa, los datos procesados, incluyendo la cantidad de vehículos, los niveles de ruido y las fechas de registro, se envían y almacenan en una base de datos MySQL a través del sistema embebido.

Finalmente, en la cuarta etapa, los resultados se visualizan en una interfaz desarrollada con Node-RED, lo que permite analizar patrones de tráfico y ruido en función del tiempo. Esta funcionalidad facilita la planificación y gestión del tránsito vehicular, así como la evaluación de la contaminación auditiva en la zona.

Los resultados obtenidos demuestran que el sistema ofrece un análisis preciso y eficiente, proporcionando datos relevantes para optimizar la movilidad y reducir el impacto acústico en zonas urbanas.

Palabras clave: Monitoreo vehicular, contaminación auditiva, Jetson Nano, ESP32, YOLOv8x, Node-RED, MySQL.

ABSTRACT

This project develops an intelligent monitoring system for vehicular traffic and noise levels on Los Chasquis Avenue using embedded processing technologies and artificial intelligence. The methodology is structured into four main stages.

In the first stage, data is collected using a video camera and a MAX9814 audio sensor connected to an ESP32. The audio signal is transmitted to a Jetson Nano, where joint video and audio processing is performed.

The second stage implements the YOLOv8x object detection model for accurate counting and classification of vehicles based on video data. Simultaneously, the audio analysis measures noise levels in decibels and classifies them into categories such as low, medium, or high.

In the third stage, the processed data, including the number of vehicles, noise levels, and registration dates, is sent and stored in a MySQL database via the embedded system.

Finally, in the fourth stage, the results are visualized through an interface developed with Node-RED, enabling the analysis of traffic and noise patterns over time. This functionality facilitates the planning and management of vehicular transit as well as the evaluation of auditory contamination in the area.

The results demonstrate that the system provides precise and efficient analysis, delivering relevant data to optimize mobility and reduce acoustic impact in urban areas.

Keywords: Vehicular monitoring, auditory contamination, Jetson Nano, ESP32, YOLOv8x, Node-RED, MySQL.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL.

1.1.1 Planteamiento del problema

La congestión del tráfico vehicular y la contaminación auditiva son desafíos muy frecuentes en las áreas urbanas densamente pobladas. La necesidad de una gestión más eficiente se vuelve crítica dado el incremento constante de vehículos, lo que resulta en impactos negativos tanto en la movilidad como en la calidad del ambiente sonoro. [1]

La falta de sistemas efectivos de monitoreo dificulta la identificación y la respuesta rápida a los puntos críticos de congestión y niveles de ruido excesivos. Esto conlleva a una experiencia de viaje menos eficiente y a una exposición prolongada a entornos acústicos perjudiciales para la salud. [2]

La implementación de sensores inalámbricos y técnicas de inteligencia artificial ofrece una solución potencial para abordar estos desafíos. Sin embargo, su aplicación precisa y efectiva requiere un enfoque específico que considere las complejidades del entorno urbano y las necesidades de gestión. [3]

La investigación en este campo busca desarrollar un sistema integral que utilice datos recopilados por sensores inalámbricos para detectar patrones de tráfico y niveles de ruido, y que, mediante algoritmos de inteligencia artificial, pueda predecir y gestionar eficazmente la congestión y la contaminación auditiva.

1.2 Antecedentes investigativos

Diversas investigaciones han abordado problemáticas relacionadas con el monitoreo del tráfico vehicular y la contaminación auditiva, proporcionando un marco referencial importante para el desarrollo del presente proyecto.

En la Universidad Nacional Autónoma de México, en 2017, Grisel Felipe Avalos y Omar Antonio Alva Garcés realizaron una investigación titulada “Diseño y Construcción de un Sistema de Rastreo Vehicular por Satélite Activo Mediante el Uso de las Tecnologías GPS/GLONASS, GSM/GPRS y Wifi”. Utilizaron tecnología de módulos GSM, GPRS, y Wifi para la localización de vehículos. También emplearon sensores de temperatura, magnético, lineal y angular para obtener datos sobre exceso de velocidad y niveles críticos de temperatura. Los resultados demostraron una correcta geolocalización de los vehículos y rastreo con una precisión de +/- 5 metros de la posición real, a su vez se obtuvieron datos precisos de las condiciones en que se encontraba el automotor. [4]

En la Universidad del Azuay, en 2017, Torres Salamea Hugo Marcelo, Cueva Cabrera Wilson Fernando, y Muñoz Ordoñez Fernando realizaron una investigación titulada “Análisis, Monitoreo e Implementación de un Algoritmo para el Conteo del Tráfico Vehicular Aplicando Visión Artificial”. Utilizaron procesamiento de imágenes y bases de datos para procesar datos recogidos por cámaras IP inalámbricas. Los resultados mostraron un correcto conteo y detección de vehículos livianos y pesados. Los datos se almacenaron en un sistema de conteo y se mostraron en gráficos estadísticos en una página web. Se realizaron pruebas de campo durante cinco horas de alta afluencia vehicular para analizar la eficiencia del sistema. El sistema presentó un error del 1.1%. Se observó que en la noche podrían producirse errores debido a la luz que enfoca la calzada. [5]

En la Universidad de las Américas, Quito, en 2018, Pablo Andrés Espinosa Lima realizó una investigación titulada "Evaluación de la Contaminación Acústica Producida por el Tráfico Vehicular en la Ciudad de Ibarra". Utilizó Sistemas de Información Geográfica (GIS) como ArcGIS, Google Maps y OpenStreetMap para elaborar un mapa de ruido predictivo del tráfico vehicular. Se utilizó el software de

simulación CadnaA para analizar los datos recogidos del tráfico en puntos críticos de la ciudad. Los resultados mostraron que Ibarra tiene índices bajos de ruido vehicular con un nivel de 60 dB en su punto máximo al mediodía, pero anticipó un aumento exponencial debido al crecimiento del parque automotor, destacando la necesidad de soluciones preventivas. [6]

En la Universidad Peruana Unión, en 2020, Jhosseling Cintia Asto Gómez realizó una investigación titulada “Evaluación de la Contaminación Acústica Generada por el Tráfico Vehicular en la Zona 4 del Distrito de Ate Vitarte”. Utilizó herramientas como un sonómetro digital Tipo 1, una cámara digital, un cronómetro, una laptop y un GPS para medir los niveles de decibelios en la zona propuesta. En el punto RU-01, ubicado en la Av. Las Torres, se midieron los niveles de presión sonora vehicular en tres turnos diferentes (mañana, tarde y noche) durante el periodo diurno. Los valores promedios obtenidos fueron: 71.8 dB en la mañana, 70.6 dB en la tarde y 73.4 dB en la noche. El resultado del periodo diurno fue de 76.8 dB. [7]

En la Universidad Estatal del Sur de Manabí, en 2020, Ariana Yocatza Macías Briones realizó una investigación titulada "Contaminación Acústica y su Incidencia en la Salud de los Habitantes en la Avenida Malecón de la Ciudad de Manta". El objetivo fue determinar la contaminación acústica en dicha avenida y su impacto en la salud de los habitantes. Se monitorearon tres puntos de la avenida dos veces por semana durante tres meses (enero, febrero y marzo de 2020) en horarios diurnos (7:00, 13:00 y 21:00 horas) utilizando un sonómetro calibrado. Los resultados mostraron que el punto 1 (Puerto de Manta) tenía los niveles más altos de contaminación acústica, especialmente al mediodía, con valores de 69.3 dB en enero y febrero, y 66.5 dB en marzo. Además, una encuesta reveló que el 18% de los encuestados experimentó pérdidas auditivas parciales debido a los altos niveles de ruido. Como recomendación, se propuso un plan de acción para mitigar los efectos negativos en los habitantes. [8]

1.3 Fundamentación teórica

Este trabajo de investigación requiere una comprensión profunda del contexto del problema asociado con la congestión vehicular y la contaminación auditiva. Además,

es crucial entender los principios subyacentes de la comunicación inalámbrica y el procesamiento de datos, que forman la base de las soluciones tecnológicas propuestas.

1.3.1 Tráfico urbano (Ordenanza y normativa para el tráfico vehicular de la ciudad de Ambato)

La gestión del tráfico en áreas urbanas es esencial para fomentar el desarrollo sostenible en las ciudades contemporáneas. Con el continuo incremento de la población urbana y los desafíos diarios relacionados con la movilidad, como se observa en la Figura 1, resulta indispensable adoptar estrategias eficaces que aseguren un tránsito vehicular fluido y seguro. Una gestión adecuada no solo disminuye la congestión y optimiza los tiempos de traslado, sino que también mejora la calidad del aire, promueve la seguridad en las vías y eleva el bienestar de los ciudadanos. [9]

El uso de tecnologías avanzadas para el control inteligente del tráfico, sumado a una coordinación eficaz entre los distintos actores de movilidad y la implementación de mantenimiento preventivo, correctivo y evolutivo, son factores clave para alcanzar una gestión eficiente y sostenible de la movilidad urbana. [9]

En Ecuador, las normas relacionadas con el tráfico vehicular están regidas por la Ley de Tránsito y Seguridad Vial, que establece un marco regulatorio uniforme para todo el país. Esta ley incluye aspectos cruciales como los límites de velocidad, la señalización y las regulaciones para el transporte público y de carga, las cuales son fundamentales en ciudades con alta densidad vehicular como Ambato. Sin embargo, la realidad del tráfico en cada ciudad varía, lo que hace necesaria la implementación de ordenanzas locales que ajusten y complementen esta ley a las particularidades de cada urbe.[10]

Ambato, reconocida por su actividad económica y comercial, enfrenta un constante flujo vehicular en arterias principales como la Avenida Cevallos y la Avenida Bolivariana. Con el objetivo de optimizar la movilidad urbana y mitigar los impactos negativos del tráfico, como los accidentes y la contaminación, las ordenanzas municipales han establecido límites de velocidad diferenciados. Estas medidas, aplicadas principalmente en zonas de alta densidad vehicular y peatonal, como el

centro histórico y las áreas comerciales, buscan mejorar la seguridad vial y reducir el riesgo de incidentes en áreas con alta concentración poblacional.[11]

La normativa nacional establece límites de velocidad que oscilan entre 50 y 90 km/h en zonas urbanas, dependiendo del tipo de vía. En Ambato, sin embargo, estos límites se reducen en zonas críticas con alta afluencia vehicular y peatonal, especialmente en el centro histórico y las principales zonas comerciales. Estas reducciones de velocidad son fundamentales para mejorar la seguridad vial y disminuir los riesgos de accidentes en áreas densamente pobladas.[12]

Entre las medidas implementadas para mejorar la movilidad, se destaca el establecimiento de zonas de parqueo reguladas mediante el uso de parquímetros y el cierre de algunas calles para su uso exclusivo por peatones. Estas medidas buscan fomentar la movilidad sostenible y mejorar la fluidez vehicular la municipalidad ha analizado la posibilidad de implementar el sistema de "Pico y Placa", como una medida temporal en días de alta congestión, aunque no se ha adoptado de manera permanente como en ciudades más grandes como Quito.[12]



Figura 1. Tráfico urbano [13]

1.3.2 Ruido ambiental y salud (Frecuencias, normativa y estándares mundiales y del Ecuador)

El ruido ambiental, particularmente en las áreas urbanas, representa una importante fuente de contaminación que impacta negativamente en la salud y el buen vivir de los ciudadanos. La exposición prolongada a altos niveles de ruido puede desencadenar

diversos problemas de salud, tanto inmediatos como a largo plazo. Entre los efectos más frecuentes se destacan los trastornos auditivos, ya que la exposición continua al ruido puede generar pérdida auditiva inducida, ocasionando daños irreparables en las células del oído interno. [14]

Además de los daños auditivos, el ruido ambiental está asociado con graves problemas a la salud de las personas. Investigaciones han demostrado que la exposición prolongada al ruido puede incrementar el riesgo de desarrollar hipertensión, enfermedades del corazón y accidentes cerebrovasculares, debido al estrés fuerte que esta contaminación genera. De igual manera, el ruido nocturno afecta la calidad y duración del sueño, lo que impacta negativamente tanto en la salud física como en el bienestar mental de las personas. [14]

En cuanto a la salud mental, la exposición prolongada al ruido puede elevar los niveles de estrés y ansiedad, impactando negativamente la vida de las personas. Los niños son especialmente sensibles a estos efectos, ya que el ruido interfiere en su capacidad de aprendizaje y afecta su rendimiento escolar, dificultando tanto la concentración como la comprensión lectora. Estos impactos subrayan la relevancia de tratar el ruido ambiental como un problema de salud pública, considerando que puede originarse a partir de múltiples fuentes, tal como se ilustra en la Figura 2. [14]

El ruido se cuantifica en decibelios (dB) y en frecuencias, medidas en Hertz (Hz). La frecuencia de un sonido representa la cantidad de ciclos que una onda sonora completa por segundo, determinando si el sonido es agudo o grave. El rango de audición humana abarca desde 20 Hz hasta 20,000 Hz. Las frecuencias más elevadas, superiores a 2,000 Hz, suelen generar mayor incomodidad para las personas, mientras que las frecuencias bajas pueden causar vibraciones que afectan tanto estructuras físicas como generar sensaciones corporales en los individuos.[15]

Según los estándares internacionales establecidos por la Comisión Internacional de Electrotecnia (IEC) y la Organización Internacional de Normalización (ISO), el análisis del ruido ambiental debe realizarse empleando ponderaciones como la ponderación A (dBA), que se ajusta a la sensibilidad del oído humano frente a diversas frecuencias. En entornos urbanos, el ruido asociado al tráfico vehicular típicamente presenta frecuencias que oscilan entre los 250 Hz y los 2,000 Hz.[16]

- **Normativa y Estándares Mundiales**

La Organización Mundial de la Salud (OMS) ha establecido recomendaciones globales que sugieren que los niveles de ruido en zonas residenciales no deben superar los 55 dB(A) durante el día y los 40 dB(A) por la noche, con el fin de prevenir efectos negativos en la salud. Por su parte, la Organización Internacional del Trabajo (OIT) señala que en ambientes industriales, la exposición al ruido en el ámbito laboral no debe exceder los 85 dB(A) durante una jornada de trabajo de 8 horas.[15]

En la Unión Europea, la Directiva 2002/49/CE sobre Evaluación y Gestión del Ruido Ambiental establece que las autoridades deben realizar mapas de ruido y tomar medidas para reducir el impacto del ruido en áreas pobladas, con un enfoque especial en la mitigación del ruido proveniente del tráfico, trenes y aeropuertos Estados Unidos, la Agencia de Protección Ambiental (EPA) regula el nivel de ruido según las actividades y sectores específicos, estableciendo límites máximos de exposición .[17], [18]

- **En Ecuador**

En Ecuador, el control del ruido ambiental está regulado por el Ministerio del Ambiente mediante el Texto Unificado de Legislación Ambiental Secundaria (TULAS), que define los límites máximos permisibles según el tipo de zona y la fuente emisora. En áreas residenciales, se permite un nivel máximo de 55 dB(A) durante el día y 50 dB(A) en la noche. Para zonas comerciales, industriales y de servicios, los límites son de 65 dB(A) en horario diurno y 55 dB(A) durante el nocturno.[19]

El Código orgánico ambiental dispone que cualquier excedencia de los límites de ruido establecidos debe ser objeto de sanción por parte de las autoridades competentes. Esto tiene como propósito salvaguardar la salud de las personas y preservar la calidad ambiental en las zonas urbanas.[19]



Figura 2. Ruido ambiental y salud [20]

1.3.3 Redes inalámbricas de sensores

La implementación de una red de sensores se fundamenta en las capacidades tecnológicas actuales, que permiten establecer una red de dispositivos diseñados para realizar capturas constantes de información. Estos sensores son capaces de registrar datos, transmitirlos entre dispositivos y enviarlos posteriormente a un sistema centralizado para su almacenamiento y análisis, como se ilustra en la Figura 3. Todo este proceso opera bajo el principio de optimizar el consumo energético, garantizando una operación eficiente y sostenible. [21]

Una red de sensores, conocida en inglés como sensor network, está compuesta por pequeños dispositivos equipados con sensores que trabajan en conjunto para cumplir una tarea específica. Estas redes están formadas por un conjunto de sensores con capacidades para detectar variables del entorno y comunicarse de manera inalámbrica, permitiendo la formación de redes ad-hoc. Este término, ad-hoc, se refiere a que no existe un nodo central que coordine la red; todos los dispositivos funcionan de manera igualitaria, colaborando entre sí sin requerir infraestructura física preestablecida ni administración centralizada. [21]

Las redes de sensores representan un concepto relativamente reciente en el ámbito de la adquisición y procesamiento de datos, con una amplia variedad de aplicaciones en

diferentes áreas, como la industria, la domótica, el ámbito militar y la monitorización ambiental. [21]

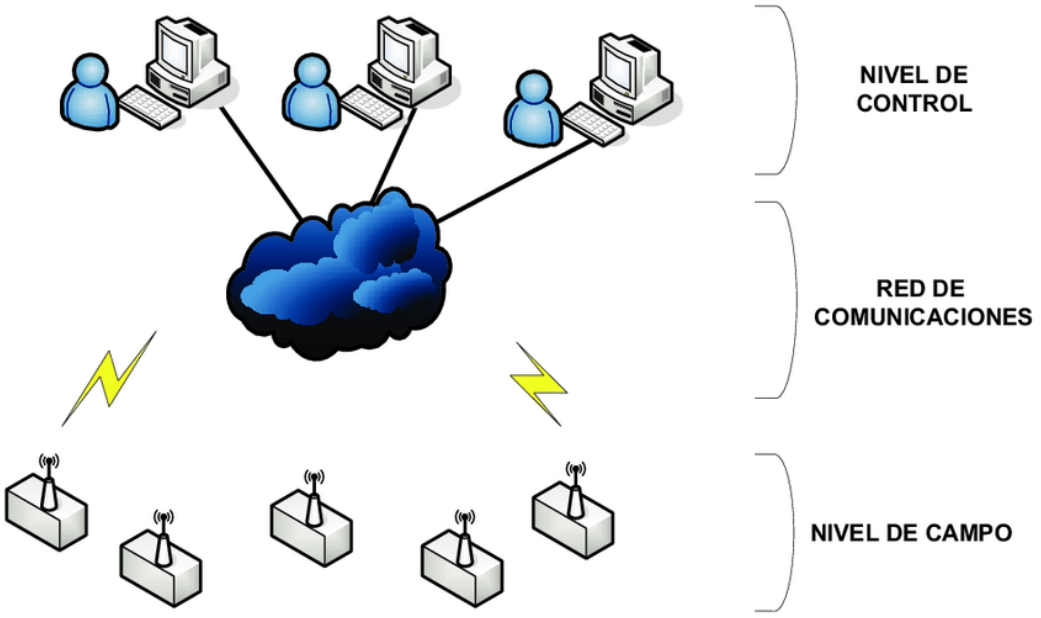


Figura 3. Redes inalámbricas de sensores [22]

1.3.4 Tecnologías de comunicación inalámbrica

La comunicación inalámbrica se refiere a diversas tecnologías que utilizan señales electromagnéticas para transmitir datos sin necesidad de cables. En lugar de un medio físico, estas tecnologías emplean el aire como medio de transmisión, utilizando principalmente ondas de radio y microondas. Este enfoque ha revolucionado la manera en que interactuamos con la tecnología, permitiendo una conectividad sin precedentes entre dispositivos. [23]

El término "comunicación" en este contexto no se limita a la interacción entre personas. También incluye la comunicación entre dispositivos y sistemas tecnológicos, como ocurre en las redes de sensores, Internet de las Cosas (IoT), y las comunicaciones máquina a máquina (M2M). Estas tecnologías permiten que diversos dispositivos se comuniquen entre sí de manera eficiente, automatizando procesos y mejorando la gestión de recursos. [23]

Las tecnologías inalámbricas abarcan una amplia gama de aplicaciones, desde las redes Wi-Fi y Bluetooth que utilizamos en nuestros hogares y oficinas como se muestra en la Figura 4, hasta sistemas más complejos como las redes celulares 4G y 5G, que ofrecen alta velocidad y capacidad para aplicaciones móviles y de internet de alta demanda. Estas redes son fundamentales para la comunicación moderna, soportando una vasta cantidad de dispositivos conectados simultáneamente. [23]

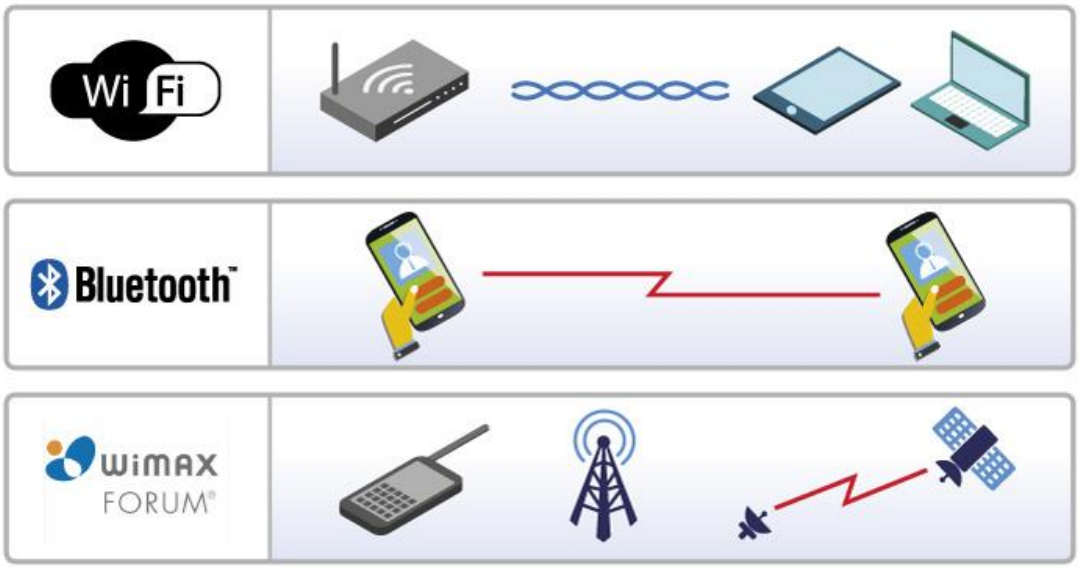


Figura 4. Tecnologías de comunicación inalámbrica [24]

1.3.5 Sistemas de monitoreo ambiental

Los sistemas de monitoreo ambiental son tecnologías diseñadas para recopilar y analizar datos en tiempo real sobre las condiciones medioambientales de una región específica. Estos sistemas emplean una variedad de sensores y dispositivos para medir diversas variables, como la calidad del aire, temperatura, humedad, niveles de ruido, radiación y otros parámetros esenciales para evaluar el entorno.[2]

La "inteligencia" de estos sistemas radica en la aplicación de algoritmos de aprendizaje automático y técnicas avanzadas de análisis de datos para procesar y examinar los parámetros ambientales recolectados. Esto capacita a las organizaciones para identificar patrones y tendencias, detectar posibles inconvenientes y tomar decisiones fundamentadas en tiempo real. [2]

Asimismo, los sistemas inteligentes de monitoreo ambiental suelen incorporar tecnologías de comunicación en red para transmitir datos y alertas en tiempo real. Esto facilita una respuesta inmediata ante situaciones de riesgo o emergencias ambientales, mejorando la capacidad de acción y mitigación. [2]

1.3.6 Sensores de ruido ambiental

Los sensores de ruido ambiental son herramientas especializadas en detectar y medir la presión acústica del entorno, proporcionando datos precisos sobre los niveles de contaminación sonora en diferentes zonas. Esta contaminación, generada principalmente por el tráfico vehicular, actividades industriales y recreativas, constituye un desafío creciente en las áreas urbanas, impactando negativamente la salud y el bienestar de las personas. Por esta razón, la incorporación de sensores de ruido resulta fundamental para monitorear los niveles acústicos y analizar su influencia en la calidad de vida de la población.[25]

Los avances en tecnología han permitido el desarrollo de sensores de ruido cada vez más precisos, compactos y eficientes, capaces de integrarse con plataformas de Internet de las Cosas (IoT) para el monitoreo en tiempo real. Estos dispositivos convierten las ondas de sonido en señales eléctricas, las cuales pueden ser procesadas para generar datos útiles en la gestión y control de la contaminación acústica. La selección del sensor correcto dependerá de algunos factores, como la precisión, sensibilidad, rango de frecuencia y facilidad de integración en sistemas más complejos.[25]

A continuación, se presentan varios de los sensores más utilizados en la medición de ruido ambiental, con una comparación de sus características y aplicaciones más comunes:

- **LM393 (Keyes-037)**

Es un sensor de sonido económico y de fácil implementación con 4 pines como se muestra en la Figura 5 que se utiliza en proyectos básicos de medición de ruido. Sin embargo, su precisión es limitada y es propenso al ruido de fondo, lo que lo hace menos adecuado para aplicaciones de tráfico vehicular que requieren alta precisión.[26]

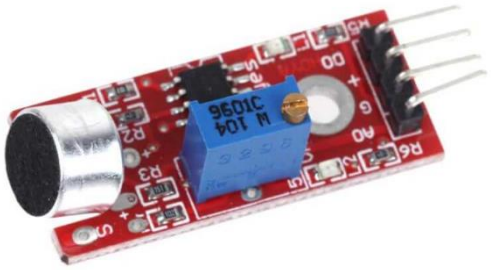


Figura 5. Módulo LM393 [27]

- **MAX4466 con amplificador electret**

Este sensor se destaca por su popularidad, gracias a su amplio rango de frecuencia y su capacidad para ajustar automáticamente la ganancia. Además, su diseño facilita la conexión, ya que cuenta con tres entradas, como se ilustra en la Figura 6. Es ideal para aplicaciones de monitoreo ambiental y se ajusta bien a proyectos de IoT. Sin embargo, en entornos donde el ruido industrial es significativo, puede requerir ajustes adicionales.[28]



Figura 6. Módulo Max4466 [29]

- **Adafruit AGC Electret MAX9814**

Este sensor se destaca por su alta sensibilidad y la capacidad de ajustar la ganancia automáticamente hasta 60dB. Es ideal para sistemas de monitoreo de tráfico vehicular, ya que puede capturar sonidos de baja intensidad en entornos ruidosos, minimizando el impacto del ruido de fondo. Además, su capacidad para reducir la distorsión lo

convierte en la mejor opción para monitorear la contaminación acústica en tiempo real sumado a sus puertos de conexión que son sencillos de conectar como se muestra en la Figura 7 según la necesidad del proyecto. La precisión adicional y la robustez lo hacen muy adecuado para sistemas avanzados de inteligencia artificial que requieren datos limpios y fiables.[30]



Figura 7. Módulo Max9814 [31]

- **SparkFun MEMS microphone breakout (ICS-43434)**

El Sensor MEMS ofrece una gran precisión y una interfaz I2S digital en sus puertos como se puede observar en la Figura 8, lo que lo convierte en una opción ideal para proyectos que requieran alta fidelidad en las mediciones acústicas, como la grabación ambiental o el análisis del tráfico. No obstante, su complejidad de implementación puede ser un desafío en sistemas IoT simples.[32]

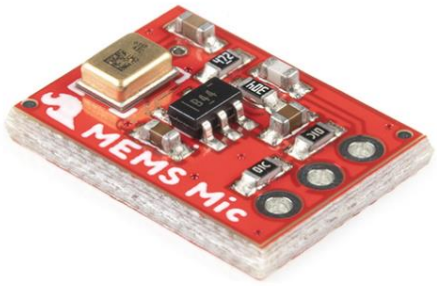


Figura 8. Módulo ICS-43434 [33]

- **Comparativa de sensores de ruido ambiental**

En la Tabla 1, se comparan los distintos sensores de ruido ambiental en función de características clave como el rango de frecuencia, consumo, sensibilidad, tipo de interfaz, manejo del ruido de fondo, aplicaciones comunes, ventajas y desventajas. Esta comparación permitirá seleccionar el sensor más adecuado para el prototipo, considerando tanto los requisitos técnicos del proyecto como las condiciones específicas de implementación.

Tabla 1. Comparación de los distintos sensores de ruido

Característica	**LM393 (Keyes-037)**	**MAX4466 con Amplificador Electret**	**Adafruit AGC Electret MAX9814**	**SparkFun MEMS Microphone Breakout (ICS-43434)**
Rango de Frecuencia	100 Hz - 10 kHz	20 Hz - 20 kHz	20 Hz - 20 kHz	50 Hz - 15 kHz
Consumo de Corriente	0.5 - 1 mA	0.75 - 1.5 mA	3 mA (modo promedio)	1.3 mA (en funcionamiento)
Sensibilidad	Alta	Alta (AGC integrado)	Muy alta (ganancia ajustable 60dB)	Alta
Interfaz	Analógica	Analógica	Analógica	I2S (digital)
Ruido de Fondo	Medio	Bajo	Muy bajo	Bajo
Aplicaciones Comunes	Sistemas básicos de monitoreo acústico	Sistemas de monitoreo de sonido ambiental en tiempo real	Monitoreo preciso de ruido ambiental, análisis de sonido de baja frecuencia	Monitoreo de alta fidelidad en tráfico vehicular, grabación de audio ambiental

Ventajas	Fácil de implementar, económico	Alta sensibilidad, manejo de ganancia automático	Ideal para aplicaciones de alta precisión y ruido bajo	Interfaz digital, alta fidelidad en mediciones
Desventajas	Baja precisión, sensible al ruido de fondo	No apto para entornos industriales sin ajustes adicionales	Requiere calibración adecuada	Puede ser más complejo de implementar en sistemas IoT

Adafruit AGC Electret Microphone Amplifier - MAX9814 se destaca como la mejor opción debido a su capacidad para captar sonidos con alta precisión en entornos ruidosos. Su función de control automático de ganancia (AGC) le permite ajustar automáticamente los niveles de sonido, lo que es esencial para lidiar con los cambios constantes en los niveles de ruido vehicular. Además, su capacidad para minimizar el ruido de fondo asegura que las lecturas de sonido sean precisas y confiables, lo cual es crucial para la integración de los datos en sistemas de inteligencia artificial y para la toma de decisiones en tiempo real.

Por lo tanto, el MAX9814 no solo cumple con los requisitos de sensibilidad y precisión, sino que también es altamente compatible con plataformas IoT, haciendo de este sensor la mejor opción para un proyecto que busca medir y analizar el ruido ambiental producido por el tráfico vehicular.

1.3.7 Cámaras web

Las cámaras web se han transformado en un objeto esencial para la captura de video en tiempo real, permitiendo su uso en sistemas de vigilancia, videollamadas, y proyectos de monitoreo. Estas cámaras varían en sus especificaciones, como la resolución, la calidad de transmisión, y los ángulos de visión.

- **Klip Xtreme 720 HD**

La Klip Xtreme 720 HD es una cámara web económica que ofrece una resolución HD de 720p como se la mira en la Figura 9 y es compatible con puertos USB 2.0. Es ideal para videoconferencias, monitoreo básico, y proyectos donde se requiere un buen rendimiento sin necesidad de una resolución extremadamente alta. Su diseño compacto

y plug-and-play la hacen fácil de usar sin la necesidad de controladores adicionales. También cuenta con un micrófono integrado, lo que le permite captar audio junto con video . [34]



Figura 9. Cámara klip xtreme 720 HD [34]

- **Microsoft 720 HD 30 FPS**

La Microsoft 720 HD 30 FPS es una cámara web de calidad profesional que ofrece una resolución de 720p a una tasa de 30 cuadros por segundo (FPS), ideal para videollamadas fluidas y grabación de video en alta definición. Su tecnología ClearFrame garantiza que las imágenes sean claras, incluso en condiciones de poca luz. Esta cámara también incluye un micrófono de alta fidelidad para capturar audio nítido, además de ser compacta como se mira en la Figura 10 y es compatible con varias plataformas como Skype y Zoom. [35]



Figura 10. Cámara microsoft 720 HD 30 FPS [35]

- **Jinxuny USB 2.0 driver-free computer webcam desktop laptop 480P**

La Jinxuny USB 2.0 480P es una cámara web básica que ofrece una resolución de 480p como se mira en la Figura 11. Está diseñada principalmente para tareas de videollamadas de baja resolución y no requiere controladores adicionales, lo que facilita su uso en diversos dispositivos. Aunque su resolución es más baja comparada con las otras cámaras, es útil para proyectos que no requieren alta definición, pero necesitan una solución económica. [36]



Figura 11. Cámara jinxuny USB 2.0 480P [36]

- **Comparativa de cámaras web**

En la Tabla 2 se presentan las características técnicas de diversas cámaras seleccionadas para su posible implementación en el prototipo. Esta comparación incluye parámetros esenciales como resolución, interfaz, presencia de micrófono incorporado, velocidad de cuadros por segundo (FPS), ángulo de visión, compatibilidad y otros aspectos que determinan la adecuación de cada cámara para aplicaciones como videoconferencia, monitoreo y videollamadas. La selección de estos elementos permite evaluar tanto la calidad de imagen como la funcionalidad, considerando también el precio aproximado, lo cual es importante para la viabilidad económica del proyecto.

Tabla 2. Comparación de cámaras web

Características	**Klip Xtreme 720 HD**	**Microsoft 720 HD 30 FPS**	**Jinxuny USB 2.0 480P**
Resolución	1280x720 píxeles (HD)	1280x720 píxeles (HD)	640x480 píxeles (SD)
Interfaz	USB 2.0	USB 2.0	USB 2.0
Micrófono Incorporado	Sí	Sí	No
FPS (Cuadros por Segundo)	30	30	15
Ángulo de Visión	Estándar	Estándar	Estándar
Compatibilidad	Windows, Mac	Windows, Mac	Windows, Mac
Función Principal	Videoconferencia, Monitoreo	Videoconferencia, Streaming	Videollamadas, Básico
Precio Aprox.	Bajo	Medio	Bajo

Se ha seleccionado la Klip Xtreme 720 HD como la cámara más adecuada para el proyecto debido a su balance entre precio y rendimiento. Aunque la Microsoft 720 HD ofrece características similares, su costo es mayor, y la Jinxuny USB 2.0 480P, aunque económica, no alcanza la resolución y calidad necesarias para capturar detalles visuales importantes. La Klip Xtreme 720 HD ofrece una resolución de 720p, suficiente para las necesidades de este proyecto. Su facilidad de uso, con la tecnología plug-and-play, la convierte en la opción más práctica para la implementación en el sistema de monitoreo.

1.3.8 Módulos para la comunicación inalámbrica

Los módulos de comunicación inalámbrica son componentes electrónicos diseñados para facilitar la transmisión y recepción de datos entre dispositivos o sistemas, eliminando la necesidad de conexiones físicas mediante cables. Estos módulos integran los componentes necesarios para establecer conexiones inalámbricas utilizando diferentes tecnologías como Wi-Fi, Bluetooth, Zigbee, LoRa, entre otras. Son ampliamente utilizados en aplicaciones de Internet de las Cosas (IoT), control remoto, monitoreo y automatización, entre otros.

Existen diversos módulos de comunicación inalámbrica que permiten a los dispositivos interactuar entre sí. Cada módulo está diseñado para diferentes tipos de aplicaciones, dependiendo de las necesidades de alcance, velocidad de datos, consumo

energético y tipo de red. A continuación, se detallan algunos de los módulos más utilizados para la transmisión de datos de forma inalámbrica:

- **ESP32 (Wi-Fi)**

El ESP32 es un módulo muy popular en aplicaciones IoT debido a su integración de Wi-Fi y Bluetooth en un solo chip que se observa en la Figura 12. Es altamente eficiente para proyectos que requieren alta velocidad de transmisión de datos y conectividad constante con redes Wi-Fi. Este módulo es capaz de manejar tareas de procesamiento complejas y tiene un bajo costo, lo que lo hace ideal para proyectos de monitoreo, como el monitoreo tráfico vehicular o del ruido ambiental.[37]



Figura 12. Módulo esp32 [38]

- **HC-05 (Bluetooth)**

El HC-05 es un módulo Bluetooth versátil, diseñado para aplicaciones de bajo consumo energético y corto alcance. Es adecuado para la transmisión de datos entre dispositivos cercanos, como en la comunicación entre sensores y microcontroladores en aplicaciones donde la movilidad es clave. Aunque tiene un alcance limitado además que es de fácil conexión debido a que no tiene una gran cantidad de pines como se muestra en la Figura 13, es una opción popular para proyectos donde la alta velocidad de datos no es una prioridad.[39]



Figura 13. Módulo HC-05 [40]

- **XBee Series 2 (Zigbee)**

El XBee Series 2 es un módulo diseñado para redes Zigbee como se mira en la Figura 14, una tecnología de baja potencia y alcance medio, ideal para redes de malla donde varios dispositivos pueden intercomunicarse. Este tipo de módulo es usado comúnmente en aplicaciones de automatización del hogar y monitoreo en áreas donde el bajo consumo de energía es una prioridad, y donde no se requiere una alta velocidad de transmisión.[41]



Figura 14. Módulo Xbee S2 [42]

- **RA-02 (LoRa)**

El módulo RA-02, basado en la tecnología LoRa (Long Range), está diseñado para transmitir datos a largas distancias, destacándose por su bajo consumo

energético, lo que lo hace ideal para aplicaciones de bajo costo y alta eficiencia. Este módulo es perfecto para aplicaciones de monitoreo en áreas extensas, como monitoreo ambiental rural o en ciudades con una infraestructura de sensores dispersos gracias a su chip Ra-02 como se mira en la Figura 15. Aunque su velocidad de datos es baja, es muy eficiente para transmitir pequeños paquetes de información a grandes distancias.[43]



Figura 15. Módulo RA-02 [44]

- **Comparativa de módulos de comunicación inalámbrica**

En la Tabla 3 se presenta una comparación detallada de los módulos de comunicación inalámbrica considerados para el diseño del prototipo. Se han evaluado diversas características, como el tipo de conexión (WiFi, Bluetooth, Zigbee, LoRa.), el alcance, la velocidad de transmisión, el consumo energético, frecuencias, procesador, entre otras. La elección de un módulo adecuado es esencial para lograr una transmisión de datos estable y eficiente, además de asegurar que el sistema pueda operar de forma autónoma sin comprometer su rendimiento ni su duración de batería.

Tabla 3. Comparación de los módulos de comunicación inalámbrica

Característica	**ESP32 (Wi-Fi)**	**HC-05 (Bluetooth)**	**XBee Series 2 (Zigbee)**	**RA-02 (LoRa)**
Tecnología	Wi-Fi (IEEE 802.11)	Bluetooth v2.0	Zigbee (IEEE 802.15.4)	LoRaWAN
Alcance	50-100 metros	10-100 metros	100-300 metros	Hasta 10 km
Velocidad de Datos	Hasta 150 Mbps	Hasta 3 Mbps	250 Kbps	0.3-50 Kbps

Consumo de Energía	Medio	Bajo	Muy bajo	Muy bajo
Frecuencia de operación	2.4 GHz y 5 GHz	2.4 GHz	2.4 GHz	868/915 MHz
Costo	Bajo	Bajo	Medio	Medio
Tamaño	25.5 x 18 mm	34.5 x 15 mm	24.38 x 27.61 mm	17.8 x 12.8 mm
Procesador	Dual-core 240 MHz	No tiene	No tiene	No tiene
Aplicaciones comunes	IoT, Domótica, Sensores	Conexión a corto alcance	Redes en malla, Automatización	Monitoreo a larga distancia
Ventajas	Alta velocidad, Wi-Fi y Bluetooth integrados	Bajo consumo, fácil implementación	Redes de malla eficientes	Cobertura a larga distancia
Desventajas	Mayor consumo que otros	Alcance limitado	Baja velocidad de datos	Baja velocidad de datos

En este proyecto, se seleccionó el ESP32 como módulo de comunicación inalámbrica gracias a su compatibilidad con Wi-Fi, lo que garantiza una transmisión de datos de monitoreo rápida, confiable y eficiente. El ESP32 no solo ofrece un alcance adecuado para la infraestructura urbana, sino que también integra Bluetooth, lo que proporciona flexibilidad para posibles expansiones en el futuro. En comparación con otros módulos, el ESP32 destaca por su balance entre costo, rendimiento y facilidad de uso, lo que lo convierte en la opción más adecuada para implementar un sistema de monitoreo inalámbrico.

1.3.9 Microordenadores

Los microordenadores son dispositivos compactos que ofrecen las funcionalidades esenciales de una computadora en un formato reducido. Estos dispositivos cuentan con un procesador, memoria, almacenamiento, puertos de entrada/salida, y permiten la ejecución de sistemas operativos, lo que los convierte en una excelente opción para proyectos de bajo costo, sistemas embebidos, y aplicaciones de Internet de las Cosas (IoT).

Su bajo costo y la capacidad de realizar tareas complejas, como la consulta de bases de datos, procesamiento de señales y control de dispositivos, los hacen populares en proyectos de monitoreo, automatización y control remoto. A continuación, se describen algunos de los microordenadores más comunes:

- **Raspberry Pi**

El Raspberry Pi es uno de los microordenadores más populares y versátiles del mercado mostrado en la Figura 16. Proporciona una plataforma de código abierto que permite a los usuarios operar sistemas como Linux o Windows IoT, brindando flexibilidad y versatilidad para diversas aplicaciones. Gracias a su bajo costo, múltiples puertos de entrada/salida, conectividad Wi-Fi y capacidad para ejecutar tareas complejas, es ideal para proyectos como la gestión de sensores, servidores pequeños, y aplicaciones de monitoreo en tiempo real.[45]



Figura 16. Raspberry Pi 4 [46]

- **BeagleBone black**

El BeagleBone Black es otro microordenador compacto que tiene una comunidad fuerte en el ámbito de la ingeniería y la robótica mostrado en la Figura 18. Ofrece más puertos de entrada/salida y es más potente en términos de procesamiento en comparación con la Raspberry Pi. Este dispositivo es especialmente útil para proyectos que requieren un alto número de interfaces físicas, como controladores de motores, sistemas embebidos industriales, y proyectos de robótica avanzada.[47]



Figura 17. Microordenador BeagleBoard black [48]

- **Odroid-C4**

El Odroid-C4 es un microordenador de alto rendimiento que utiliza un procesador más potente y memoria RAM DDR4 mostrado en la Figura 18, lo que lo hace en una excelente opción para aplicaciones que requieren más procesamiento y mayor capacidad gráfica. Aunque su costo es mayor en comparación con el Raspberry Pi, es popular para aplicaciones como servidores multimedia, computación intensiva y simulación.[49]

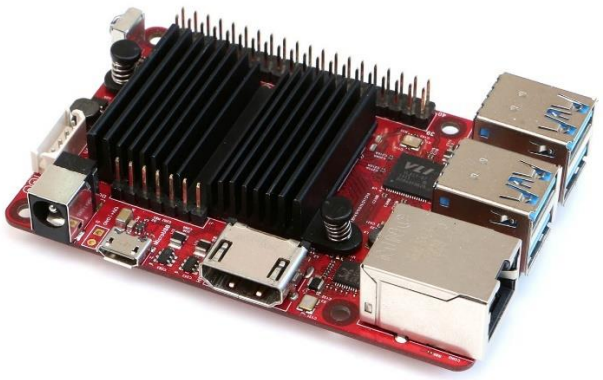


Figura 18. Microordenador Odroid-C4 [49]

- **Jetson Nano**

El Jetson Nano de NVIDIA es un microordenador especializado en tareas de inteligencia artificial y aprendizaje automático. Gracias a su unidad de procesamiento

gráfico (GPU) como se mira en la Figura 19, es capaz de ejecutar redes neuronales y otras tareas relacionadas con el procesamiento de imágenes y reconocimiento de patrones. Aunque no es tan accesible como otros microordenadores, es ideal para proyectos de visión artificial, robótica avanzada, y aplicaciones de inteligencia artificial en tiempo real.[50]

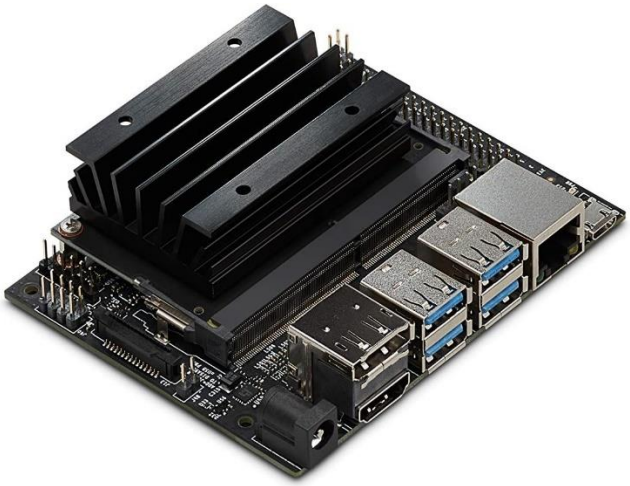


Figura 19. Microordenador NVIDIA Jetson Nano [51]

- **Comparativa de microordenadores**

En este punto se hará una comparación de los microordenadores antes expuestos para determinar cuál de ellos sería el que presenta mejores prestaciones para el sistema considerando puntos claves descritos en la Tabla 4.

Tabla 4. Comparación de los microrordenadores

Característica	**Jetson Nano**	**Raspberry Pi 4**	**Odroid-C4**	**BeagleBone Black**
Procesador	Quad-core ARM Cortex-A57 + GPU NVIDIA Maxwell	Quad-core ARM Cortex-A72	Quad-core ARM Cortex-A55	Single-core ARM Cortex-A8
GPU	128-core NVIDIA Maxwell GPU	No incluye GPU dedicada	Mali-G31 MP2	No incluye GPU dedicada
Memoria RAM	4 GB LPDDR4	4 GB o 8 GB LPDDR4	4 GB LPDDR4	512 MB DDR3
Almacenamiento	MicroSD (soporte adicional para almacenamiento USB)	MicroSD (soporte adicional para USB y SSD)	eMMC, MicroSD	4 GB eMMC + MicroSD
Conectividad	Ethernet, sin Wi-Fi integrado (requiere adaptador)	Ethernet, Wi-Fi integrado	Ethernet, sin Wi-Fi integrado	Ethernet, sin Wi-Fi integrado

Puertos GPIO	40 pines (estándar Raspberry Pi)	40 pines	40 pines	92 pines
Soporte de AI/ML	Aceleración por GPU, compatible con frameworks de IA	Limitado (sin GPU dedicada)	Limitado (sin GPU dedicada)	Limitado, sin optimización para IA
Consumo energético	5 W típico (10 W máximo)	6 W típico	4 W típico	2.5 W típico
Facilidad de uso	Complejo, orientado a proyectos avanzados	Muy fácil, ideal para principiantes	Intermedio, requiere mayor configuración	Intermedio, con buena documentación
Aplicaciones comunes	IA, visión por computadora, robótica avanzada	Educación, monitoreo IoT, proyectos básicos	Aplicaciones multimedia y servidores básicos	Automatización industrial, sistemas embebidos
Ventajas	Ideal para aplicaciones de IA y visión por computadora gracias a la GPU dedicada.	Gran comunidad y soporte, versátil	Bajo consumo energético, buena relación costo/beneficio	Excelente cantidad de pines GPIO y soporte para aplicaciones de tiempo real
Desventajas	Mayor curva de aprendizaje, sin Wi-Fi integrado	Rendimiento limitado en IA	Menor soporte en comparación con Raspberry Pi	Hardware menos potente, limitado para aplicaciones intensivas

Para este proyecto, se ha seleccionado el Jetson Nano como el microordenador principal debido a su excelente rendimiento en aplicaciones que requieren procesamiento intensivo, especialmente en tareas relacionadas con la inteligencia artificial y la visión por computadora. Su GPU NVIDIA Maxwell con 128 núcleos es ideal para ejecutar algoritmos de aprendizaje automático como YOLO, optimizando la detección y análisis de datos visuales.

En comparación con otros microordenadores, el Jetson Nano sobresale por su capacidad de ejecutar modelos de IA con alta eficiencia, brindando un entorno robusto para el desarrollo de sistemas avanzados de monitoreo. Aunque su curva de aprendizaje es más alta, la disponibilidad de bibliotecas optimizadas y el soporte para frameworks como TensorFlow y PyTorch lo convierten en la opción ideal para este tipo de implementación.

Además, el Jetson Nano se adapta perfectamente a las necesidades del prototipo, gracias a su capacidad de manejo de múltiples sensores y su conectividad expandible mediante pines GPIO estándar. Aunque no incluye Wi-Fi integrado, su compatibilidad con adaptadores externos permite una configuración inalámbrica eficiente. Estas características, combinadas con su capacidad de escalabilidad y rendimiento, justifican su elección como el núcleo del sistema de monitoreo propuesto en este proyecto.

1.3.10 Procesamiento de señales y datos

El procesamiento de señales consiste en transformar y analizar datos para identificar aspectos que no son evidentes a través de la observación directa, como se ilustra en el esquema de la Figura 20. Esta disciplina permite a ingenieros y científicos estudiar, optimizar y corregir señales en diversos contextos, incluyendo transmisiones de audio, datos científicos, imágenes y videos. [52]

Se utiliza ampliamente en diversas industrias y aplicaciones, tales como [52] :

- Compresión y procesamiento de señales de audio.
- Adquisición y análisis de datos.
- Procesamiento digital de imágenes y gráficos.
- Compresión y procesamiento de video.
- Reconocimiento y análisis de voz.
- Procesamiento y optimización de señales RADAR, SONAR y LiDAR.
- Aplicaciones médicas, especialmente en imágenes como tomografía computarizada (CAT) y resonancia magnética (MRI).
- Predicciones meteorológicas.
- Oceanografía, incluyendo el análisis del rendimiento acústico submarino.

El procesamiento de señales puede aplicarse tanto a sistemas lineales como no lineales. Sin embargo, la mayoría de los sistemas se encuentran en el ámbito no lineal, donde los valores cambian con el tiempo de manera impredecible y los cambios en la salida no son proporcionalmente lineales a los cambios en la entrada.[52]

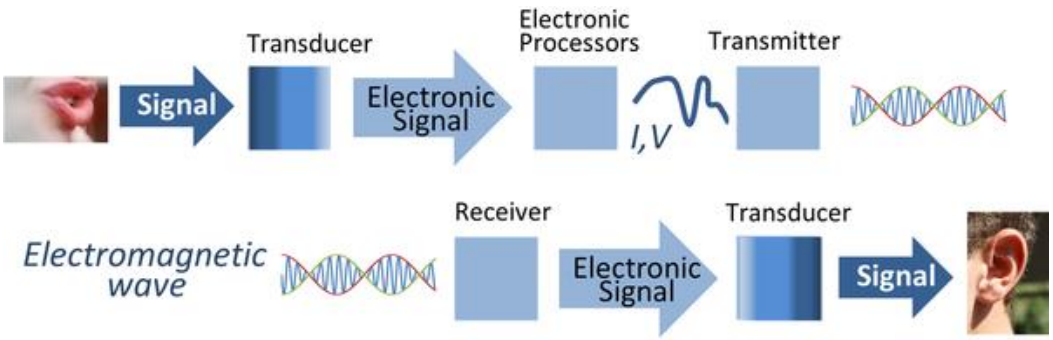


Figura 20. Diagrama general del procesamiento de señales [53]

1.3.11 Protocolos para el envió de datos inalámbricos

Los protocolos de envío de datos inalámbricos son fundamentales para la transmisión de información entre dispositivos de forma remota. Estos protocolos definen cómo se transfieren los datos a través de redes inalámbricas, garantizando que la comunicación sea eficiente, confiable y segura. La elección del protocolo adecuado depende de las características del sistema, como la cantidad de datos a enviar, la velocidad de transmisión, la latencia, la confiabilidad, y la seguridad requerida.

A continuación, se detallan algunos de los principales protocolos empleados en la transmisión de datos de forma inalámbrica:

- **MQTT (Message Queuing Telemetry Transport)**

MQTT es un protocolo ligero diseñado para la transmisión eficiente de pequeños volúmenes de datos entre dispositivos con capacidades limitadas. Se basa en un modelo de publicación-suscripción, donde los datos se envían a un intermediario o "broker" que se encarga de distribuirlos a los suscriptores interesados como se muestra en la Figura 21. MQTT se caracteriza por su eficiencia en el consumo de ancho de banda y energía, además de proporcionar distintos niveles de calidad de servicio para asegurar la entrega confiable de mensajes. Este protocolo es especialmente adecuado

para sistemas que demandan comunicación continua y en tiempo real, incluso en condiciones de conectividad limitada o con dispositivos que experimentan conexiones intermitentes. [54]

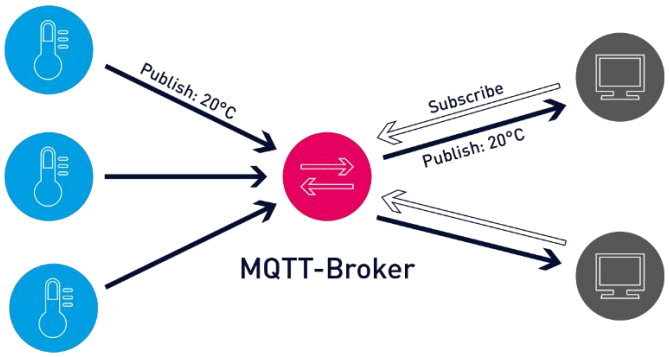


Figura 21. Arquitectura MQTT-Broker [55]

- **HTTP/HTTPS (Hypertext Transfer Protocol Secure)**

El HTTP es uno de los protocolos más comunes en la transmisión de datos por internet como se observa en la Figura 22. HTTPS, su versión segura, cifra los datos para proteger la información durante la transmisión. Si bien es un protocolo muy utilizado, no es el más eficiente para dispositivos con recursos limitados o para el envío continuo de datos en tiempo real, debido a su mayor consumo de ancho de banda y energía. Sin embargo, es ampliamente compatible y fácil de integrar en aplicaciones web.[56]

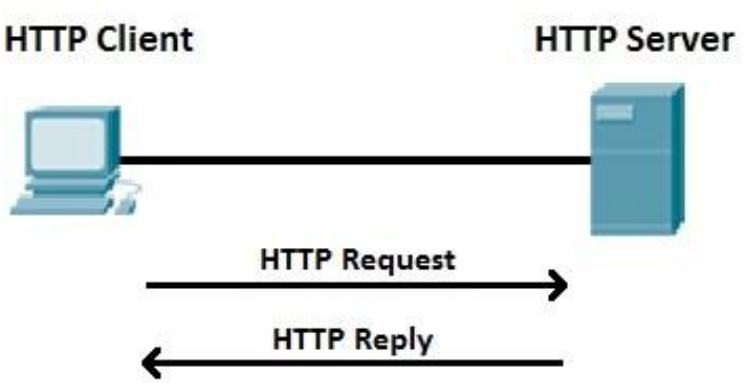


Figura 22. Protocolo HTTP [57]

- **CoAP (Constrained Application Protocol)**

CoAP es un protocolo diseñado específicamente para dispositivos con recursos limitados, ofreciendo funcionalidades similares a HTTP pero optimizadas para su uso en redes IoT, donde la eficiencia y el bajo consumo de recursos son fundamentales. Está basado en el protocolo UDP, lo que lo hace más ligero y eficiente en términos de uso de recursos y energía. CoAP utiliza un modelo de solicitud-respuesta como se observa en la Figura 23 y está pensado para comunicaciones rápidas y de bajo costo energético, aunque su implementación es más reciente y tiene menor difusión.[58]

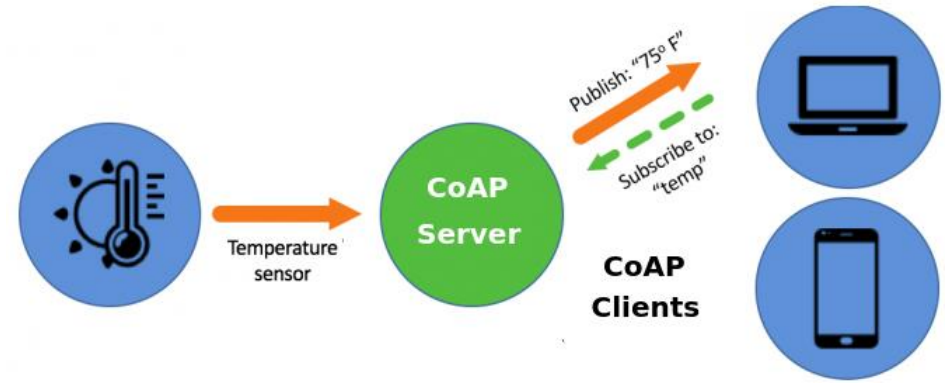


Figura 23. Protocolo CoAP [59]

- **LoRaWAN (Long Range Wide Area Network)**

LoRaWAN es un protocolo utilizado en comunicaciones de largo alcance y bajo consumo de energía. Está diseñado para dispositivos IoT que se encuentran distribuidos en áreas geográficas amplias como se mira en la Figura 24, donde las transmisiones de datos son poco frecuentes, pero se requiere que las conexiones se mantengan durante largos períodos de tiempo sin drenar la batería de los dispositivos.[60]



Figura 24. Protocolo LoRaWAN [61]

- **Comparativa de protocolos de envió de datos inalámbricos**

Existen gran variedad de protocolos en la actualidad como los que se analizaron anteriormente, por ello se llevará a cabo una comparación de estos en la Tabla 5 para determinar cuál de ellos sería el más adecuado para la conexión del sistema teniendo una eficiencia óptima.

Tabla 5. Comparación de los protocolos

Característica	**MQTT**	**HTTP/HTTPS**	**CoAP**	**LoRaWAN**
Modelo de comunicación	Publicación-Suscripción	Solicitud-Respuesta	Solicitud-Respuesta	Publicación-Suscripción
Uso de recursos	Bajo	Alto	Muy bajo	Muy bajo
Seguridad	TLS/SSL	TLS/SSL (HTTPS)	DTLS	AES-128
Velocidad de transmisión	Alta	Media	Alta	Baja
Rango de cobertura	Limitado (depende del hardware)	Global (internet)	Local (IoT, UDP)	Largo (hasta 15 km)
Aplicación ideal	Monitoreo en tiempo real	Aplicaciones web, transmisión de grandes volúmenes de datos	Redes de sensores locales	Redes de sensores distribuidos
Ventajas	Eficiente, bajo consumo	Amplia compatibilidad, seguro	Ligero, eficiente	Gran alcance, bajo consumo
Desventajas	Necesita un bróker	Mayor consumo de recursos	Menor difusión, soporte limitado	Baja velocidad de datos

Para el presente proyecto, se selecciona MQTT como el protocolo de envío de datos inalámbricos. La principal razón para esta elección es su bajo consumo de energía y su capacidad para manejar eficientemente pequeñas cantidades de datos en tiempo real,

lo cual es crucial en sistemas de monitoreo donde se recolecta información de manera constante y se requiere que la transmisión sea fiable.

Otra ventaja de MQTT es su modelo de publicación-suscripción, que simplifica la comunicación entre múltiples dispositivos, permitiendo que los datos enviados desde los puntos de monitoreo lleguen de manera efectiva al sistema central, sin depender de conexiones constantes y consumiendo menos ancho de banda. Esto resulta particularmente útil en escenarios donde la conectividad es inestable o se encuentra restringida. Además, MQTT ofrece niveles de calidad de servicio, lo que asegura que los datos lleguen de manera fiable a su destino.

En comparación con otros protocolos, MQTT es la mejor opción para un proyecto de monitoreo que requiere una comunicación constante y eficiente sin sobrecargar la red, y permite un uso óptimo de los recursos, lo que es esencial en este tipo de sistemas.

1.3.12 Inteligencia artificial y aprendizaje automático

La inteligencia artificial (IA) surgió a principios del siglo XX, impulsada por pioneros del incipiente campo de las ciencias de la computación, quienes comenzaron a explorar la posibilidad de que las máquinas pudieran "pensar". En este contexto, la IA puede definirse como el esfuerzo por automatizar tareas intelectuales que tradicionalmente requieren intervención humana. La IA es un campo amplio que abarca disciplinas como el Aprendizaje Automático (AA) y el Aprendizaje Profundo, distribuidas en diversas ramas, como se ilustra en la Figura 25 , además, incluye otros subcampos que no necesariamente implican procesos de "aprendizaje". [62]

El Aprendizaje Automático, conocido en inglés como Machine Learning, es una rama de la Inteligencia Artificial enfocada en el desarrollo de técnicas que permitan a las máquinas aprender de manera autónoma. En términos más específicos, se centra en la creación de algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de datos proporcionados en forma de ejemplos. Este proceso implica una inducción de conocimiento, donde se obtiene un enunciado generalizado a partir de la observación y análisis de casos particulares. [62]



Figura 25. Inteligencia artificial [63]

a. Algoritmos de Machine Learning

- **Algoritmo You Only Look Once (YOLO)**

El algoritmo You Only Look Once (YOLO) es un sistema de código abierto diseñado para la detección de objetos en tiempo real. Utiliza una única red neuronal convolucional para identificar objetos en imágenes, como vehículos y señales de tránsito, tal como se muestra en la Figura 26. Para su funcionamiento, la red neuronal divide la imagen en múltiples regiones, prediciendo cuadros delimitadores y asignando probabilidades a cada región. Las cajas generadas son ponderadas según las probabilidades calculadas. Además, el algoritmo es capaz de aprender representaciones generalizables de los objetos, lo que reduce significativamente los errores de detección al procesar datos nuevos que no forman parte del conjunto de entrenamiento.[64]

El algoritmo principal es capaz de operar a una velocidad de 45 cuadros por segundo (FPS) sin la necesidad de procesamiento por lotes, utilizando una GPU Titan X. Además, existe una versión optimizada que alcanza velocidades superiores a 150 FPS. Gracias a su rendimiento, este algoritmo es ideal para aplicaciones que requieren detección de objetos en transmisiones de video, ofreciendo un retardo de señal inferior a 25 milisegundos.[64]

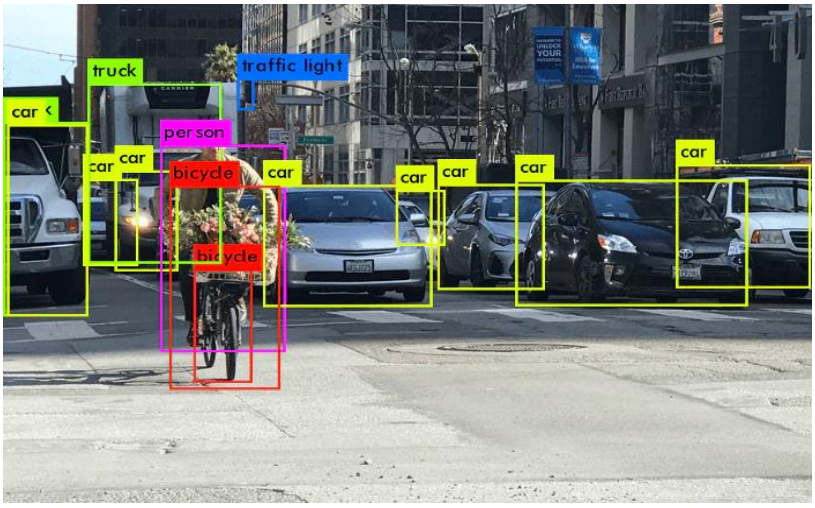


Figura 26. Detección de vehículos con yolo [65]

- **Single Shot MultiBox Detector (SSD)**

El SSD es otro algoritmo de detección de objetos que se destaca por ser más rápido que las Redes Neuronales Convolucionales (CNN) tradicionales. Aunque también es eficiente para aplicaciones en tiempo real, SSD procesa las imágenes dividiéndolas en subcuadrículas como se mira en la Figura 27, lo que le otorga una velocidad similar a YOLO, pero con menos precisión en objetos más pequeños o en condiciones de baja visibilidad.[66]

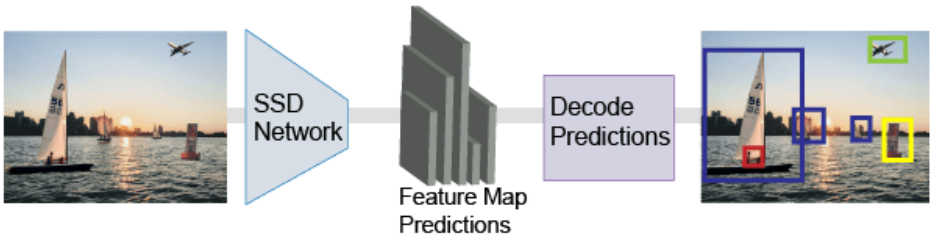


Figura 27. Detección de objetos con SSD [67]

- **Faster R-CNN**

Faster R-CNN es una mejora de los algoritmos R-CNN tradicionales y está diseñado para realizar tareas de detección de objetos. Aunque ofrece una alta precisión, su velocidad es inferior en comparación con YOLO y SSD, debido a que sigue dividiendo la imagen en regiones de interés (RoIs) para luego procesarlas como se mira en la Figura 28, lo que aumenta la complejidad computacional y reduce la capacidad para detección en tiempo real.[68]

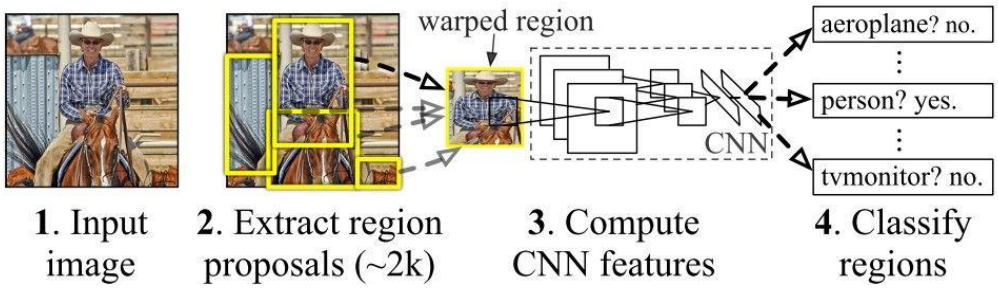


Figura 28. Detección de imagen en Faster R-CNN [69]

- **Comparativa de algoritmos**

La Tabla 6 se muestra una comparación de todos los algoritmos antes descritos sus principales características cruciales para el desarrollo del sistema, se analiza cuál es la mejor opción para la detección de objetos en este caso vehículos de todo tipo.

Tabla 6. Comparación de los distintos algoritmos

Característica	**YOLO**	**SSD**	**Faster R-CNN**
Tipo de Algoritmo	Detección de objetos en tiempo real	Detección de objetos	Detección de objetos
Velocidad de Procesamiento	Muy alta	Alta	Media
Precisión en imágenes	Alta	Media	Muy alta
Capacidad para tiempo real	Excelente	Buena	Limitada
Complejidad Computacional	Alta	Media	Muy alta
Aplicación ideal	Monitoreo de tráfico, vigilancia	Detección de peatones	Clasificación y detección detallada
Ventajas	Rápido, eficiente para detección múltiple	Rápido para objetos más grandes	Alta precisión en detección compleja
Desventajas	Requiere mayor capacidad computacional	Menos preciso en objetos pequeños	Muy lento para tiempo real

YOLO ha sido elegido por su alta velocidad de procesamiento y precisión en tiempo real, lo cual es fundamental para el monitoreo en tiempo real del proyecto. A diferencia de SSD y Faster R-CNN, que sacrifican precisión o velocidad respectivamente, YOLO mantiene un equilibrio óptimo entre ambos, asegurando una rápida detección de objetos sin comprometer la exactitud. Además, su capacidad de procesar imágenes completas en una sola operación lo convierte en el más eficiente para detectar vehículos en sistemas de tráfico inteligentes, donde las decisiones deben tomarse de forma inmediata.

b. Versiones del algoritmo Yolo

Para la implementación del sistema de monitoreo de tráfico, se analizaron y compararon diversas versiones pertenecientes a la familia de modelos YOLO (You Only Look Once) para determinar cuál se ajustaba mejor a las necesidades del proyecto, tanto en cuanto a rendimiento como a precisión. A continuación, en la Tabla 7, se describen sus diferentes prestaciones.

Tabla 7. Comparación de las versiones de YOLO

Versión YOLO	Precisión (mAP)	Velocidad (FPS)	Tamaño del Modelo (MB)	Características Clave
YOLOv5	Alta	Muy rápida	Pequeño	Alto rendimiento en dispositivos con limitados recursos.
YOLOv6	Muy Alta	Rápida	Mediano	Mejora de la precisión respecto a YOLOv5, pero con mayor tamaño.
YOLOv7	Muy Alta	Moderada	Grande	Mayor precisión, pero con un costo de velocidad.
YOLOv8	Muy Alta	Alta	Mediano	Última versión con mejoras significativas en precisión y eficiencia.

c. Variantes de la versión de Yolov8

A continuación, en la Tabla 8 , se va a comparar las diferentes variantes que existen del modelo YOLOv8 en la cual se escogerá el modelo más apropiado para el mejor análisis de los datos recolectados:

Tabla 8. Comparación de las variantes de YOLOv8

Variante YOLOv8	Precisión (mAP)	Velocidad (FPS)	Uso de Recursos (GPU)	Número de Parámetros (M)	Uso
YOLOv8n	0.65	120	Muy bajo	6.2	Aplicaciones móviles, drones.
YOLOv8s	0.72	90	Bajo	11.2	Dispositivos con recursos limitados.
YOLOv8m	0.78	60	Medio	25.9	Sistemas que requieren un balance entre precisión y velocidad.
YOLOv8l	0.80	50	Medio-Alto	43.7	Aplicaciones que requieren alta precisión con una velocidad aceptable.
YOLOv8x	0.82	47	Alto	68.2	Análisis detallado y con recursos GPU disponibles.

1.3.13 Bases de datos

Las bases de datos son herramientas esenciales para almacenar, organizar y gestionar grandes volúmenes de información de manera eficiente. En proyectos tecnológicos, las bases de datos permiten garantizar la integridad y disponibilidad de los datos, facilitando su uso en sistemas que requieren consultas rápidas y precisas. Existen diversos sistemas de gestión de bases de datos (DBMS) que se clasifican según su arquitectura, escalabilidad y capacidad de manejo de transacciones. A continuación, se describen y comparan algunos de los sistemas de bases de datos más relevantes para su implementación local.

- **MySQL**

MySQL es un sistema de gestión de bases de datos relacional, ampliamente utilizado y de código abierto representado como en la Figura 29, ampliamente adoptado en entornos de desarrollo y servidores web. Forma parte del entorno LAMP (Linux, Apache, MySQL, PHP/Python/Perl), destacándose por su compatibilidad con lenguajes de programación web y su funcionalidad para manejar extensos volúmenes de datos. MySQL ofrece un excelente equilibrio entre rendimiento, facilidad de uso y flexibilidad, siendo ideal tanto para aplicaciones pequeñas como escalables. [70]



Figura 29 . Base de datos MySQL [71]

- **PostgreSQL**

Reconocido por su robustez y cumplimiento de estándares, PostgreSQL representado en la Figura 30 es una base de datos relacional avanzada que ofrece extensibilidad y características avanzadas como transacciones ACID, índices personalizados y compatibilidad con diversos lenguajes de programación. Su enfoque técnico la hace adecuada para sistemas complejos, aunque requiere una configuración más especializada. [72]



Figura 30. Base de datos PostgreSQL [73]

- **SQLite**

SQLite es una base de datos embebida que se ejecuta directamente desde un archivo de disco, sin necesidad de configurar un servidor. Su diseño ligero representado en la Figura 31 la hace ideal para aplicaciones móviles o de escritorio con requisitos simples. Sin embargo, no es adecuada para manejar grandes volúmenes de datos ni múltiples usuarios concurrentes.[74]



Figura 31. Base de datos SQLite [75]

- **Comparativa de bases de datos**

A continuación, se expone una comparación detallada en la Tabla 9 entre diferentes sistemas de bases de datos, evaluando aspectos críticos como escalabilidad, facilidad de uso, compatibilidad, etc. Cada base de datos se analiza en términos de sus ventajas y limitaciones para determinar cuál se ajusta mejor a los requisitos específicos del sistema.

Tabla 9. Comparación de las bases de datos

Característica	**MySQL**	**PostgreSQL**	**SQLite**
Tipo de Base de Datos	Relacional	Relacional	Relacional embebida
Escalabilidad	Alta (ideal para entornos web y locales)	Alta (adecuada para sistemas complejos)	Limitada (mejor para sistemas pequeños)
Facilidad de uso	Muy alta (intuitivo y bien documentado)	Moderada (requiere experiencia técnica)	Muy alta (sin configuración de servidor)
Compatibilidad	Lenguajes web como PHP y Python	Multilenguaje (Python, Java, otros)	Compatible con todos los lenguajes
Requisitos de hardware	Moderados	Moderados-altos	Bajos
Soporte de transacciones	Completo (ACID)	Completo (ACID)	Limitado
Ventajas principales	Integración LAMP, código abierto, comunidad activa	Extensibilidad, soporte avanzado de índices	Ligereza, portabilidad
Desventajas principales	Ninguna significativa, curva de aprendizaje baja	Complejidad inicial en la configuración	Limitaciones para datos grandes y múltiples usuarios

Tras analizar las opciones, se seleccionó MySQL debido a su capacidad para integrarse perfectamente en el entorno LAMP, ampliamente reconocido por su fiabilidad en

proyectos de servidores web y sistemas locales. MySQL ofrece una combinación única de facilidad de uso, escalabilidad y soporte para aplicaciones dinámicas que manejan grandes volúmenes de datos. Su comunidad activa y extensa documentación garantizan una curva de aprendizaje rápida y soporte continuo. Además, MySQL proporciona herramientas avanzadas para la gestión eficiente de datos, optimizando tanto el rendimiento como la seguridad del sistema. Estas características lo convierten en la elección ideal para un proyecto de monitoreo vehicular y niveles de ruido, donde se requiere precisión, rapidez y capacidad de expansión futura.

1.3.14 Visualización gráfica de datos almacenados

La visualización gráfica de los datos almacenados es un componente esencial en los sistemas que requieren la interpretación de grandes volúmenes de información. Esta visualización permite analizar y monitorear los datos de forma intuitiva y eficiente, proporcionando gráficos y paneles que resumen los datos en tiempo real o históricos.

- **Node Red**

Node-RED es una herramienta de desarrollo de código abierto basada en programación visual, creada por IBM, que facilita la conexión entre dispositivos de hardware, APIs y servicios en línea.[76]

Es una plataforma robusta y de fácil aprendizaje que no requiere conocimientos avanzados de programación, consolidándose como una solución clave para la gestión y transformación de datos en tiempo real en aplicaciones IoT e Industria 4.0.[76]

Node-RED permite construir tareas específicas mediante la conexión gráfica de bloques predefinidos, denominados nodos, como se muestra en la Figura 32. Estos nodos se agrupan en flujos (flows), que generalmente combinan nodos de entrada, procesamiento y salida.[76]

La plataforma cuenta con una amplia variedad de nodos, que incluyen desde protocolos estándar como MQTT, REST, Modbus, OPC-UA, Bacnet y Websocket, hasta integraciones con APIs de terceros, como Microsoft Azure, Amazon Web Services, Twitter, Facebook y otros servicios.[76]

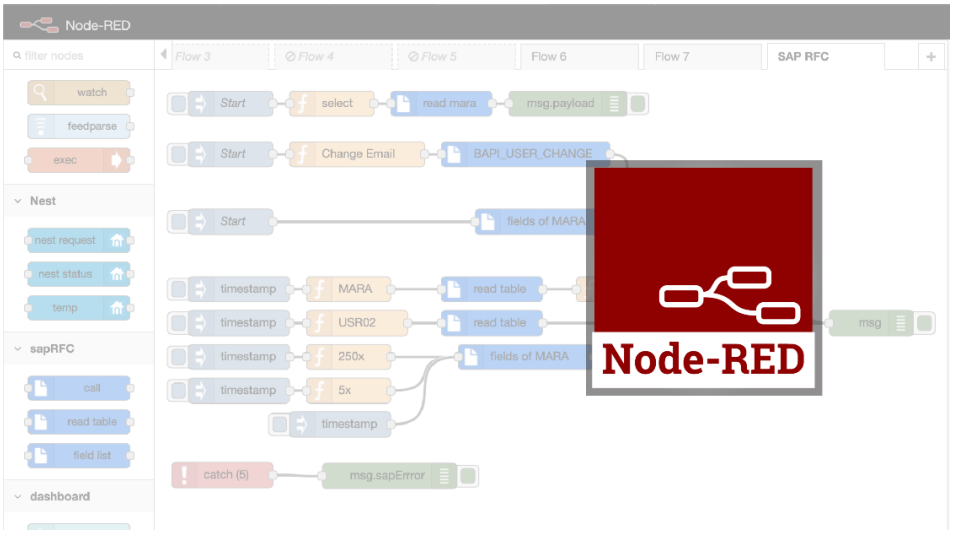


Figura 32. Interfaz de Node-RED [76]

- **Grafana**

Grafana es una plataforma de visualización y análisis de código abierto ampliamente utilizada para monitorizar datos en tiempo real. Ofrece paneles avanzados para la creación de gráficos interactivos y personalizables como se observa en la Figura 33. Grafana es muy flexible en cuanto a las bases de datos que soporta, lo que incluye bases de datos relacionales y de series temporales como InfluxDB, Prometheus y MySQL. Es especialmente adecuado para manejar grandes volúmenes de datos y ofrece funcionalidades avanzadas, como la configuración de alertas. Estas características lo convierten en una herramienta ideal para el análisis y gestión de datos en sistemas de monitoreo..[77]



Figura 33. Visualización en Grafana [78]

- **Tableau**

Tableau es una de las herramientas líderes en representación de datos, conocida por su capacidad para crear gráficos interactivos y análisis detallados sin necesidad de programación avanzada. Es muy utilizado en entornos empresariales para la creación de informes detallados y presentaciones ejecutivas como se mira en la Figura 34. Aunque no es específico para datos en tiempo real, su potencia radica en su capacidad para generar informes complejos y detallados a partir de grandes bases de datos, siendo ideal para análisis de datos históricos y tendencias.[79]



Figura 34. Visualización en tableau [80]

- **Comparativa de herramientas de visualización gráfica**

En la siguiente Tabla 10 se comparan varias herramientas de visualización gráfica de datos, evaluando aspectos relevantes como capacidad de integración, facilidad de uso, personalización y costo. Estas herramientas permiten representar visualmente datos almacenados en sistemas de monitoreo y análisis, proporcionando una interfaz intuitiva para los usuarios y haciendo más fácil realizar un plan de accionar. La selección adecuada de la herramienta será escogida según la necesidad que tenga el proyecto, tales como el nivel de interactividad requerido y la compatibilidad con otros componentes del sistema.

Tabla 10. Comparación de herramientas de visualización gráfica

Característica	Node-RED	Grafana	Tableau
Enfoque	Visualización y gestión de flujos de datos en tiempo real	Análisis de datos en tiempo real y series temporales	Análisis de datos históricos y reportes empresariales
Interfaz gráfica	Basada en nodos, flujo gráfico	Paneles personalizados, gráficos interactivos	Gráficos avanzados y paneles interactivos
Facilidad de uso	Alta (interfaz gráfica simple)	Media (requiere configuración avanzada)	Alta (fácil uso con pocos clics)
Conexión con bases de datos	Amplia (compatible con APIs, bases de datos, IoT)	Compatible con bases de datos relacionales y series temporales	Alta compatibilidad, pero orientada a informes empresariales
Uso en tiempo real	Excelente, diseñado para IoT	Muy bueno, especialmente con bases de datos de series temporales	Limitado, más orientado a análisis histórico
Escalabilidad	Media	Muy alta	Alta
Alertas	Limitadas	Avanzadas (configurables)	No posee alertas en tiempo real
Personalización	Alta, con nodos personalizables	Alta, con soporte para múltiples plugins	Moderada, con opciones limitadas en informes estándar
Aplicación ideal	Monitoreo IoT y de datos en tiempo real	Monitorización de infraestructura y rendimiento de sistemas	Análisis de datos históricos y empresariales
Ventajas	Interfaz sencilla, permite crear flujos personalizados y visualizaciones dinámicas	Paneles potentes y flexibles, soporte para múltiples bases de datos	Fácil creación de gráficos avanzados y presentaciones
Desventajas	Menos funciones avanzadas para análisis profundo	Requiere conocimiento técnico para su implementación	No orientada a tiempo real, más enfocada en reportes

Para este proyecto, se ha seleccionado Node-RED como la herramienta principal de visualización gráfica debido a su capacidad para manejar datos rápidos, que es crucial para el monitoreo del sistema. Node-RED permite la creación de flujos gráficos personalizados con facilidad, integrándose sin problemas con la base de datos seleccionada y otros sistemas a través de APIs. Además, su interfaz gráfica simplifica el proceso de visualización y conexión de datos, y esto lo hace en la opción ideal para proyectos en los que se requiere flexibilidad y visualización continua de datos. Aunque Grafana ofrece un mayor nivel de análisis avanzado y es ideal para series temporales, su complejidad y enfoque técnico no son necesarios para este proyecto. Tableau, aunque muy potente en análisis histórico, no cumple con los requisitos, lo que hace a Node-RED la mejor opción.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema para el monitoreo del tráfico vehicular y la contaminación auditiva mediante el uso de sensores inalámbricos e inteligencia artificial.

1.4.2 Objetivos específicos

- Diagnosticar la situación actual del tráfico vehicular y la contaminación auditiva en entornos urbanos, asegurando su integración efectiva en la infraestructura del sistema.
- Identificar los tipos de sensores inalámbricos adecuados para la recolección de datos sobre tráfico vehicular y contaminación auditiva en entornos.
- Seleccionar algoritmos de inteligencia artificial para procesar datos de sensores y generar análisis precisos sobre la congestión del tráfico y niveles de ruido, almacenando los resultados en una base de datos.
- Realizar pruebas piloto en áreas urbanas específicas para evaluar la capacidad del sistema en gestionar la congestión del tráfico y la contaminación auditiva.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

Para el presente proyecto de investigación, se emplearon varios materiales y herramientas para diseñar el prototipo del sistema de monitoreo del tráfico vehicular y la contaminación auditiva. En primer lugar, se utilizó un módulo de comunicación inalámbrica ESP32 para la captura de señales provenientes del sensor de ruido Adafruit MAX9814, que permitió registrar los niveles de contaminación sonora en entornos urbanos. Adicionalmente, se empleó una cámara Klip Xtreme USB 2.0 conectada a un microordenador Jetson Nano para la captura de video del flujo vehicular y su análisis mediante el modelo YOLOv8 (You Only Look Once).

El protocolo MQTT se utilizó para la transmisión de datos, mientras que los datos recopilados fueron almacenados y gestionados en una base de datos en Mysql. La visualización de los datos se llevó a cabo con Node-RED, permitiendo mostrar gráficamente la información del tráfico vehicular y la contaminación auditiva.

Para el desarrollo del sistema, se programaron diferentes componentes utilizando Python como lenguaje principal, trabajando en el entorno de desarrollo Visual Studio Code. La configuración y programación del módulo ESP32 se realizaron a través del entorno Arduino IDE. Roboflow se empleó para el etiquetado de imágenes, permitiendo la creación de un dataset específico que se combinó con el dataset COCO, el cual ya había sido previamente descargado y procesado para ser compatible con el modelo YOLOv8.

El proyecto también involucró el uso de scripts para la conversión de formatos de datos, convirtiendo archivos en formato JSON a TXT, para el entrenamiento del modelo de detección de vehículos. Esta integración de materiales y herramientas permitió el desarrollo de un prototipo capaz de monitorear el tráfico vehicular y los niveles de ruido en áreas urbanas, con el objetivo de gestionar el tráfico y la contaminación de manera más eficiente.

2.2 Métodos

2.2.1 Modalidad de la investigación

Este proyecto de investigación representó un ejemplo de investigación aplicada, ya que buscó resolver el problema planteado mediante la aplicación práctica de conocimientos en el desarrollo de un sistema electrónico. Dicho sistema se centró en la integración de técnicas de inteligencia artificial y la utilización de sensores inalámbricos para monitorear y gestionar el tráfico vehicular y la contaminación auditiva en entornos urbanos.

La investigación se basó en una metodología de campo, ya que implicó la recolección y análisis de datos en entornos urbanos reales, utilizando sensores inalámbricos y técnicas de inteligencia artificial para monitorear el tráfico vehicular y la contaminación auditiva. Además, se llevó a cabo una investigación bibliográfica para sustentar la elección de técnicas y metodologías utilizadas, así como para comprender el estado del arte en cuanto a sistemas de monitoreo y gestión del tráfico y la contaminación auditiva. Esta revisión bibliográfica permitió identificar soluciones previas y referencias que guiaron el diseño del sistema propuesto, asegurando una sólida base teórica y un enfoque alineado con los desarrollos más recientes en la materia.

2.2.2 Recolección de información

Para recopilar información, se consultaron los repositorios de varias universidades nacionales e internacionales, así como libros, revistas científicas en línea y otras bases de datos confiables mostrados en la Tabla 11. El objetivo era obtener recursos que contribuyen al desarrollo del proyecto.

Tabla 11. Recolección de Información

Criterio	Descripción
¿Para qué?	Se identificaron los datos necesarios para desarrollar un sistema de monitoreo del tráfico vehicular y la contaminación auditiva.
¿De qué personas u objetos?	Se recopiló información de fuentes como sensores inalámbricos, sistemas de monitoreo existentes, estudios de tráfico y contaminación auditiva, así como datos demográficos de la población urbana.
¿Cómo?	Se utilizaron técnicas de inteligencia artificial para analizar los datos recopilados por sensores inalámbricos, como cámaras de tráfico y micrófonos ambientales, y generar información relevante sobre la congestión del tráfico y los niveles de ruido.
¿Con qué?	Se emplearon sensores inalámbricos, dispositivos de captura de datos, plataformas de procesamiento de datos y algoritmos de inteligencia artificial para la recolección, procesamiento y análisis de la información.
¿Cuándo?	Se llevó a cabo la recolección de datos de manera continua para proporcionar información actualizada sobre el tráfico vehicular y la contaminación auditiva en áreas urbanas.
¿Dónde?	Se realizó la recolección de datos en entornos urbanos, principalmente en áreas con altos niveles de tráfico vehicular y densidad poblacional, donde la gestión del tráfico y la contaminación sonora representan desafíos significativos.

2.2.3 Procesamiento y análisis de datos

Para el procesamiento y análisis de datos en este proyecto, se llevaron a cabo los siguientes procedimientos metodológicos:

- Se recolectaron datos de sensores inalámbricos ubicados en áreas urbanas, así como de otras fuentes relevantes, como bases de datos de tráfico y registros de contaminación auditiva.

- Se examinaron estudios previos relacionados con el monitoreo del tráfico vehicular y la contaminación auditiva para identificar enfoques y metodologías aplicables.

- Se analizó la literatura académica y técnica sobre variables clave del proyecto, como la congestión del tráfico y los niveles de ruido.

- Se interpretaron los datos recolectados para extraer información relevante sobre patrones de tráfico y fluctuaciones en los niveles de ruido, fundamentales para la toma de decisiones en la gestión urbana.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Introducción

En esta sección se presentan los resultados del sistema de monitoreo de tráfico vehicular y contaminación auditiva, desarrollado con sensores inalámbricos e inteligencia artificial. Se muestran los datos recolectados sobre la congestión vehicular y niveles de ruido en áreas urbanas clave. Además, se analiza la precisión y eficiencia del sistema, así como su capacidad para procesar los datos y ayudar en la gestión del tráfico y el ruido.

3.2 Estado actual del tráfico vehicular en la ciudad de Ambato

Para comprender mejor la situación actual del tráfico vehicular en la ciudad de Ambato, se ha llevado a cabo una revisión basada en datos recientes y opiniones de los ciudadanos reflejados en la Tabla 12. Este análisis tiene como objetivo verificar el primer punto de la investigación, que es diagnosticar la congestión vehicular en las zonas más críticas de la ciudad.

La ciudad de Ambato ha experimentado un notable incremento en el tráfico vehicular en los últimos años, especialmente en las zonas céntricas y de mayor afluencia. De acuerdo con datos obtenidos del Plan de Movilidad Urbana Sostenible de la Municipalidad de Ambato, el 55% de los ciudadanos considera que la congestión ha empeorado en los últimos cinco años, mientras que un 35.5% opina que la situación se ha mantenido estable. Este incremento en el tráfico se debe, principalmente, al aumento del parque automotor y la falta de infraestructura vial adecuada para gestionar el flujo vehicular en horas pico.[81]

Las estadísticas revelan que, en 2021, aproximadamente 11.6 millones de vehículos ingresaron al centro de la ciudad, lo que representa un notable incremento en comparación con años anteriores. Esta situación ha generado una percepción negativa entre los ciudadanos, quienes manifiestan que llegar a sus destinos puede tomar hasta 30 minutos debido a la congestión.[81]

La congestión se concentra principalmente en las zonas céntricas como se muestra en la Figura 35, donde las avenidas como la Av. 12 de Noviembre y Cevallos registran los mayores niveles de flujo vehicular. Por ejemplo, se reporta que, en la Av. Cevallos, entre las 06:45 y las 07:00, circulan hasta 3.500 vehículos privados y 1.500 vehículos públicos en un intervalo de apenas 15 minutos. Este nivel de saturación se repite en otros sectores críticos como la Av. Atahualpa, donde los volúmenes vehiculares también superan los 2.000 vehículos en ciertos intervalos. [82], [83]



Figura 35. Tráfico en la zona céntrica de Ambato. [84]

Estos datos reflejan la gravedad del problema, evidenciando un sistema vial que no es capaz de soportar la demanda de vehículos en momentos de máxima demanda.

El análisis de tráfico en otras intersecciones importantes, como en la zona de Camino El Rey y Miñarica, también arroja cifras alarmantes. Según el estudio de evaluación de tráfico en esta intersección, el tráfico promedio diario anual (TPDA) muestra un nivel de congestión considerable, y las predicciones a futuro indican que la situación podría empeorar si no se implementan medidas correctivas. En el día más congestionado de la semana, se registró un flujo de 7.200 vehículos en total, lo que refleja la urgencia de una intervención para mejorar la capacidad y el manejo del tráfico en estas áreas. [82]

Otro factor clave que agrava la congestión es la circulación de vehículos de transporte público. Un estudio realizado en el centro de Ambato encontró que el 62,5% de los ciudadanos utiliza transporte público como principal medio de transporte para ingresar y salir del centro de la ciudad. La afluencia constante de autobuses y taxis en horas

pico añade una carga considerable a la capacidad vial. Esto genera cuellos de botella y complica aún más la movilidad en áreas densamente transitadas. [83]

A esto se suma el problema de las calles estrechas, que limitan la cantidad de carriles disponibles para la circulación, mientras que las aceras se han ensanchado, reduciendo los espacios para el aparcamiento y el tránsito de vehículos.

El comportamiento del tráfico también muestra diferencias importantes en cuanto a los tipos de transporte y los patrones de uso. Por ejemplo, en la Av. Yahuira, el flujo de vehículos privados entre las 06:45 y las 07:15 oscila entre 1.200 y 1.700 vehículos, mientras que el tráfico de transporte público en el mismo período varía entre 400 y 800 vehículos. [82], [83]

Tabla 12. Lugares con mayor afluencia de tráfico vehicular [85], [86]

Ubicación	**Horas Pico**	**Cantidad de Vehículos**	**Descripción de la actividad**
Avenida Cevallos y Francisco Flor	07:00 - 08:00	320	Intersección crítica con alto tráfico vehicular durante la mañana
	17:00 – 18:00	350	Tráfico generado por actividades comerciales
Avenida Cevallos y Espejo	12:00 – 13:00	400	Parada de buses, zona de comercio formal e informal
	18:00 – 19:00	450	Mayor concurrencia debido a la cercanía de zonas comerciales y de ocio
Avenida 12 de noviembre y Martínez	08:00 – 09:00	300	Afluencia de vehículos en una de las avenidas principales
	13:00 – 14:00	330	Tráfico por actividades comerciales
Intersección Juan Benigno Vela y Mera	07:00 – 08:00	280	Tránsito elevado debido a ser una ruta clave de ingreso al centro
	18:00 – 19:00	320	Tráfico generado por la salida del trabajo y actividades comerciales
Av. Pérez de Anda y Montalvo	12:00 – 13:00	250	Zona comercial cercana, influenciada por actividad vehicular

Estas cifras indican que los automóviles particulares dominan las calles de Ambato durante las horas punta, lo que sugiere la necesidad de políticas públicas que promuevan el uso de modos de transporte más eficientes o colectivos, como el transporte público y los carriles exclusivos para buses.

En general, la situación del tráfico en la ciudad de Ambato es compleja y multifacética. Las encuestas realizadas revelan que el 37,5% de los ciudadanos considera que nunca ha llegado tarde a sus actividades diarias debido a la congestión vehicular, mientras que el 18,75% indica que casi siempre llega tarde debido al tráfico. [83]

Este análisis confirma que la ciudad enfrenta serios desafíos en la gestión del tráfico vehicular, lo que respalda la necesidad de implementar soluciones tecnológicas y sistemas avanzados de monitoreo, como los propuestos en esta investigación.

3.3 Situación del ruido actual en la ciudad de Ambato

La ciudad de Ambato, ubicada en la provincia de Tungurahua, ha experimentado un aumento en los niveles de ruido ambiental, particularmente en las zonas comerciales y céntricas. Estudios recientes han revelado que las principales fuentes de ruido son el tráfico vehicular, el comercio formal e informal, así como la actividad nocturna de bares y discotecas. Estas actividades han generado niveles de ruido que, en muchos casos, superan los límites establecidos por la normativa ambiental vigente como se puede observar en el análisis de la Tabla 13, lo que puede tener implicaciones negativas para la salud de los habitantes.

En la Avenida Cevallos, una de las principales arterias viales de la ciudad, se llevó a cabo un estudio detallado sobre los niveles de presión sonora (NPS). Los resultados indicaron que los valores de ruido en esta área varían significativamente entre el día y la noche, con picos que superan los 100 dB en algunos puntos clave, como el Viaducto La Yahüira y el Parque Cevallos. Estos puntos, identificados como críticos, presentan una mayor concentración de actividades comerciales y recreativas como se puede observar en la Figura 36, lo que contribuye a los altos niveles de ruido. [85]

En el horario diurno, los niveles de presión sonora promedio oscilaron entre 61 y 72 dB, con un máximo registrado de 109.7 dB en el Viaducto La Yahüira. Por la noche,

estos valores se incrementaron, llegando a un máximo de 117.8 dB en el mismo punto, influenciado principalmente por la actividad de bares, discotecas y el tránsito de vehículos, especialmente motocicletas modificadas para producir mayor ruido. [85]

Además de la Avenida Cevallos, otras áreas de Ambato, como las zonas comerciales y residenciales, también presentan niveles de ruido preocupantes. En particular, las mediciones realizadas en el Mercado Modelo y el Mercado Ferroviario evidenciaron que el ruido vehicular y el uso de parlantes para la promoción comercial contribuyen significativamente al ruido ambiental. Durante el día, los niveles de ruido en estas zonas superaron los 60 dB, mientras que por la noche, aunque hubo una ligera reducción, los valores aún exceden los límites permisibles, establecidos en 50 dB para áreas comerciales. [85]

Tabla 13. Ruido ambiental medido en puntos críticos de la ciudad de Ambato [85], [86]

Ubicación	**Ruido Mínimo (dB)**	**Ruido Máximo (dB)**	**Promedio Diurno (dB)**	**Promedio Nocturno (dB)**
Mercado Ferroviario	64.1	99.2	64.1	58.3
Mercado Modelo	69.6	101.5	66.9	61.8
Parque Cevallos	62.9	93.3	62.9	65.9
Viaducto la Yahüira	65.9	117.8	65.9	103.4

Estos datos reflejan que la ciudad de Ambato enfrenta un problema serio de contaminación acústica, especialmente en sus áreas más concurridas. A pesar de los esfuerzos realizados por las autoridades locales y los estudios previos, el ruido sigue siendo un desafío que requiere medidas más estrictas de control y regulación. La implementación de tecnologías de monitoreo de ruido, junto con la creación de políticas públicas enfocadas en la reducción del tráfico y la promoción de actividades económicas en horarios regulados, podrían ser soluciones efectivas para mitigar este problema. [86]



Figura 36. Ruido ambiental vehicular y comercio en la ciudad de Ambato. [87]

3.4 Materiales seleccionados

En función del análisis y comparativas realizados en el capítulo 1, se ha seleccionado una lista de componentes específicos agrupados en la Tabla 14 que cumplen con las necesidades técnicas y operativas para la implementación del prototipo del sistema. Cada material fue elegido considerando criterios de eficiencia, compatibilidad y adecuación al contexto de uso planteado. A continuación, se detalla la lista de los materiales seleccionados para el desarrollo del prototipo:

Tabla 14. Lista de materiales para la implementación del prototipo

Material	Selección	Justificación
Módulo de comunicación inalámbrica	ESP32	Capacidad de conectividad WiFi y compatibilidad con sistemas de procesamiento de datos.
Microordenador	Jetson Nano	Utilizado como unidad de procesamiento central por su capacidad de manejo de datos y facilidad de integración.
Sensor de ruido	Adafruit MAX9814	Sensibilidad y precisión en la captura de niveles de ruido ambiental, ideal para el análisis de contaminación acústica.
Cámara Web	Cámara Klip Xtreme USB 2.0	Captura de imágenes de alta calidad y cumplimiento con los requisitos de monitoreo visual.
Protocolo de transmisión de datos	MQTT	Eficiencia en el envío de datos en aplicaciones de IoT y compatibilidad con el entorno de transmisión inalámbrica del prototipo.
Algoritmo de detección de objetos	YOLOv8 (You Only Look Once)	Alta precisión y rapidez en la detección de vehículos y personas en el procesamiento de imágenes.
Base de datos en la nube	Mysql	Escalabilidad y facilidad de integración para almacenar y gestionar los datos recolectados.

Plataforma de visualización de datos	Node-RED	Representación gráfica de los datos en tiempo real, permite la creación de flujos visuales de información de manera accesible y configurable.

3.5 Esquema general del sistema

En la Figura 37 se muestra el esquema general del sistema diseñado para la detección de niveles de ruido y análisis de tráfico vehicular en la ciudad de Ambato, empleando tecnologías de Inteligencia Artificial y procesamiento de datos en la nube. Este sistema consta de cinco bloques principales que se detallan a continuación:

a. Bloque de entrada

Este bloque corresponde a la etapa inicial de adquisición de datos. Incluye un sensor de ruido (MAX9814) para capturar los niveles sonoros y una cámara web (modelo Klip Xtreme 720 HD) para monitorear visualmente el flujo vehicular. Ambos dispositivos proporcionan datos fundamentales para la detección del tráfico vehicular y la contaminación acústica producida en el ambiente.

b. Bloque de procesamiento de datos

En esta etapa, los datos recibidos por la comunicación wifi de la ESP32 son procesados utilizando una Jetson Nano, donde se ejecuta un modelo de detección de objetos basado en YOLO (You Only Look Once) para identificar y contar los vehículos captados en las imágenes. Este procesamiento permite extraer información detallada sobre el flujo vehicular y los niveles de ruido en el área monitoreada.

c. Bloque de almacenamiento

Una vez procesados, los datos de tráfico y ruido se almacenan en Mysql, garantizando la disponibilidad y seguridad de la información.

d. Bloque de presentación de datos

Finalmente, se emplea Node-RED para crear una interfaz gráfica donde se presentan los resultados de monitoreo. Esta interfaz permite visualizar el nivel de ruido y la

densidad vehicular detectada en las zonas estudiadas, brindando a los usuarios una herramienta intuitiva para el análisis y toma de decisiones.

Cada uno de estos bloques opera en conjunto para proporcionar una solución integral de monitoreo de ruido y tráfico vehicular, facilitando así el análisis urbano y la implementación de estrategias de gestión de movilidad en la ciudad de Ambato.

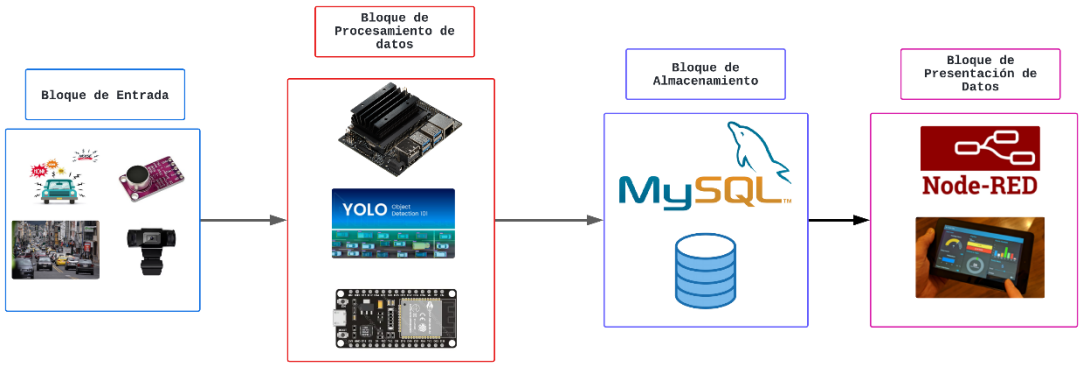


Figura 37. Esquema general del sistema a implementar

3.6 Esquema de comunicación del sistema

En la Figura 38 se presentan las conexiones y la comunicación entre los diferentes componentes del sistema. El sistema integra varios dispositivos y módulos, cada uno cumpliendo una función específica dentro de la arquitectura general. Los componentes principales incluyen el módulo ESP32, la Jetson Nano, una cámara web, un sensor de ruido, y un módulo de batería para la alimentación.

La comunicación inalámbrica se establece utilizando tecnologías como Wi-Fi y MQTT para el envío de datos entre el ESP32, la Jetson Nano y la base de datos Mysql como se muestra en la Figura 38. Además, se implementa Node-RED para la visualización y monitoreo de datos en una interfaz gráfica intuitiva. Las conexiones por cables permiten la transmisión de energía y señales de datos entre sensores y microcontroladores, garantizando la precisión y estabilidad en la recolección y procesamiento de datos.

Este esquema ilustra de forma clara cómo se interconectan los elementos del sistema para facilitar un flujo de información eficiente y una gestión centralizada de los datos recolectados.

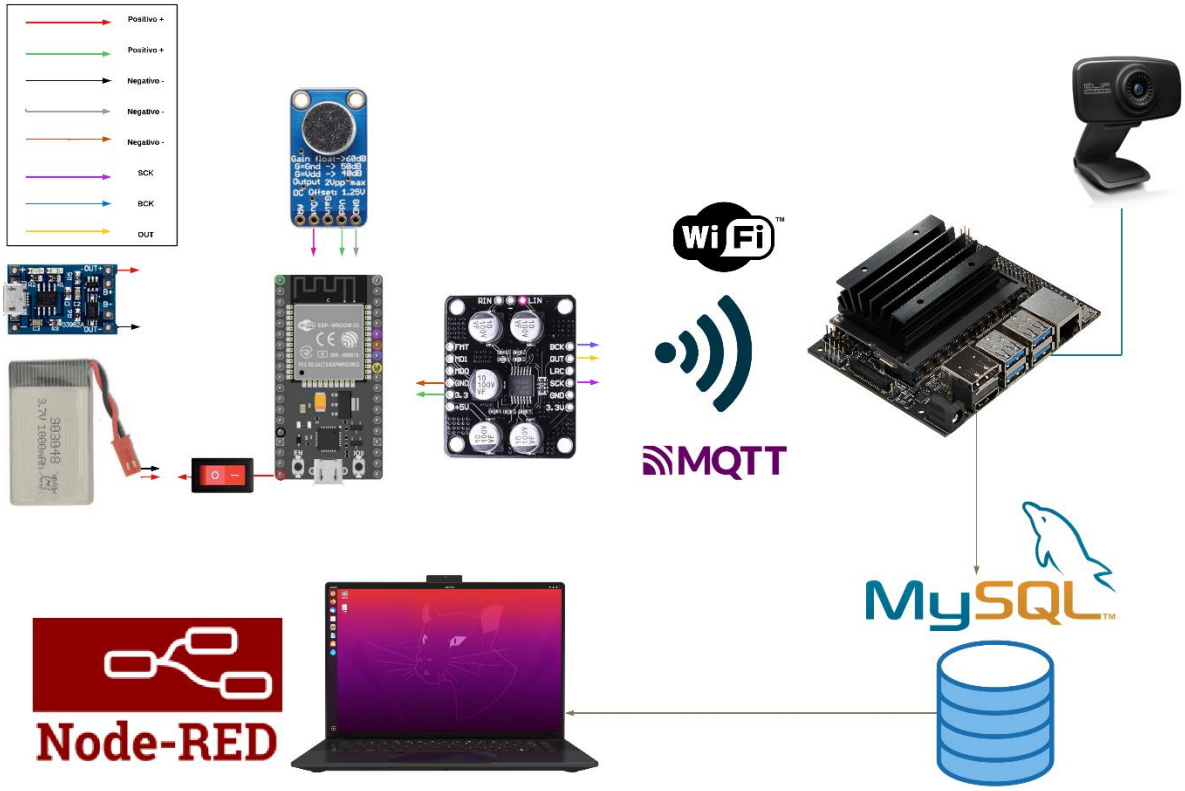


Figura 38. Conexión del sistema

3.7 Entrenamiento y programación para el análisis del tráfico

3.7.1 Desarrollo del sistema de detección de vehículos

El sistema de monitoreo de tráfico vehicular se basa en el uso de un modelo de aprendizaje automático de última generación, YOLOv8, combinado con un algoritmo de seguimiento SORT (Simple Online and Realtime Tracking). La selección de estas tecnologías permite implementar una solución eficiente para la detección y el conteo de vehículos, que se integra de manera efectiva con el sistema de monitoreo de tráfico vehicular propuesto.

El sistema se implementó utilizando Python y varias librerías de código abierto, como se describe a continuación en la Tabla 15:

Tabla 15. Librerías principales para el sistema

Librería/Tool	Propósito	Justificación Técnica
Ultralytics (YOLOv8)	Detección y clasificación de vehículos y personas en imágenes y videos.	Precisión en la detección de objetos y capacidad de procesamiento eficiente con redes neuronales convolucionales.
OpenCV	Procesamiento de imágenes y manipulación de frames de video.	Soporte robusto para operaciones de visión por computadora, incluyendo transformación y segmentación.
DeepSort	Seguimiento de objetos detectados en secuencias de video.	Permite la asociación continua de objetos entre frames, esencial para el seguimiento de vehículos.
MySQL Connector	Conexión y gestión de la base de datos para almacenar resultados.	Permite la integración directa con MySQL para garantizar la persistencia de datos recolectados.
NumPy	Cálculos numéricos avanzados, incluyendo transformaciones en señales.	Eficiencia en cálculos matemáticos necesarios para el análisis de datos recolectados.

3.7.2 Justificación de la selección de la versión YOLO

De acuerdo con la comparación realizada en la Tabla 7 los resultados de pruebas preliminares y la necesidad de obtener una alta precisión en la detección de vehículos bajo condiciones complejas (e.g., tráfico denso, diferentes tipos de vehículos y condiciones de luz variables), YOLOv8 fue la versión seleccionada debido a su capacidad para identificar vehículos y elementos del tráfico con mayor precisión. Aunque versiones como YOLOv5 ofrecían una mayor velocidad, estas sacrificaban la precisión que era esencial para el análisis minucioso requerido en este proyecto.

El modelo YOLOv8 fue seleccionado principalmente por las siguientes razones descritas en la Tabla 16:

Tabla 16. Justificación de la versión de Yolo

Criterio	YOLOv5	YOLOv7	YOLOv8
Precisión superior	Moderada precisión en objetos pequeños y dinámicos.	Alta precisión en detecciones de tamaño mediano y fijo.	Precisión significativamente mejorada en objetos pequeños y dinámicos gracias a mejoras en la arquitectura neuronal.

Capacidad de manejo de datos complejos	Manejo básico de datos complejos en escenarios controlados.	Soporte robusto para datos complejos, pero con mayor dependencia del hardware.	Optimización avanzada para manejar datos diversos, manteniendo un equilibrio entre rendimiento y uso eficiente del hardware.
Flexibilidad para escenarios dinámicos	Limitada a configuraciones estáticas, con rendimiento variable.	Buena adaptabilidad en entornos semiestáticos.	Adaptabilidad mejorada en escenarios dinámicos gracias a su diseño modular y soporte para configuraciones personalizadas.
Velocidad de inferencia	Muy alta velocidad en sistemas con hardware potente.	Alta velocidad, aunque disminuye con datos más complejos.	Velocidad optimizada incluso en hardware estándar, lo que garantiza eficiencia en múltiples entornos.
Compatibilidad con hardware limitado	Moderada, requiere GPU moderna para obtener resultados óptimos.	Requiere hardware avanzado para aplicaciones en tiempo real.	Excelente compatibilidad con hardware limitado, lo que facilita su implementación en dispositivos IoT.

3.7.3 Justificación de la variante de YOLOv8

Como se muestra en la Tabla 8, YOLOv8x es más adecuado para escenarios que requieren la máxima precisión posible y donde se dispone de recursos computacionales suficientes (como GPUs de alto rendimiento). Esta elección se hizo teniendo en cuenta que nuestro objetivo principal era proporcionar un sistema de análisis de tráfico con resultados altamente confiables.

3.7.4 Descripción de la base de datos de COCO

El dataset COCO (Common Objects in Context) es una base de datos extensa y ampliamente utilizada en el ámbito de la visión artificial y el aprendizaje profundo. Contiene imágenes etiquetadas que representan objetos en escenarios de la vida cotidiana, capturando la diversidad de situaciones en las que los objetos pueden estar presentes. COCO incluye anotaciones detalladas para la detección de objetos, segmentación, clasificación y otros tipos de análisis, lo que lo convierte en una herramienta muy valiosa para el entrenamiento de modelos de detección como YOLOv8. En este proyecto, se ha utilizado COCO para enriquecer el conjunto de datos original y mejorar la capacidad del modelo de detectar objetos, específicamente vehículos, en un contexto urbano. En la Figura 39 se muestra un esquema de este proceso.

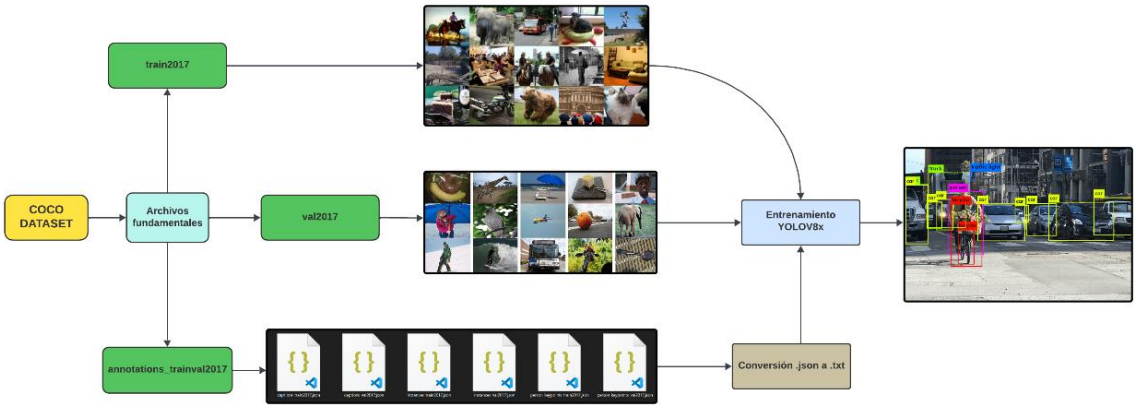


Figura 39. Proceso de utilización del dataset COCO con Yolov8x

3.7.5 Descripción del dataset aumentado con roboflow

Para la descripción del dataset aumentado, se detalla el proceso de enriquecimiento del dataset original de COCO, con el objetivo de mejorar la precisión y la capacidad de generalización del modelo YOLOv8x representado en la Figura 40. Se comenzó descargando los conjuntos de datos train2017 y val2017 del dataset de COCO, junto con los archivos de anotación annotations_trainval2017, los cuales incluían los archivos instances_train2017.json e instances_val2017.json. Estos archivos contienen las etiquetas necesarias para cada imagen, que se encontraban en el formato JSON propio de COCO. Para su integración con YOLOv8x, fue necesario convertir estas anotaciones desde el formato JSON a archivos TXT, compatibles con la estructura de YOLO.

Además de los datos originales del COCO, se amplió el dataset añadiendo 200 imágenes capturadas directamente del entorno donde se implementaría el sistema. Estas imágenes fueron etiquetadas utilizando la plataforma Roboflow, que permitió generar las etiquetas de manera eficiente y en el formato adecuado. De estas 200 imágenes, se distribuyó un 80% al conjunto de entrenamiento (train) y un 20% al conjunto de validación (val), asegurando así una correcta representación de las características del entorno específico.

A lo largo del proceso de preparación del dataset, se desarrollaron diferentes scripts que facilitaron la manipulación y el procesamiento de los datos. Primero, se realizó la extracción de frames de videos capturados en la zona de implementación, los cuales

se usaron para generar imágenes adicionales que fueron luego etiquetadas. Las etiquetas proporcionadas por Roboflow se descargaron ya en el formato TXT compatible con YOLOv8x, sin requerir una conversión adicional.

Finalmente, se procedió a organizar todas las imágenes y anotaciones en una estructura de carpetas compatible con el entrenamiento del modelo YOLOv8x. Este proceso garantizó que el conjunto de datos estuviera listo para ser utilizado en el entrenamiento del modelo de detección, maximizando así la calidad y la aplicabilidad de las detecciones vehiculares. De este modo, se logró preparar un dataset que combina información general proporcionada por COCO con datos específicos del entorno de implementación, lo cual resulta esencial para lograr una alta precisión en la detección de vehículos.

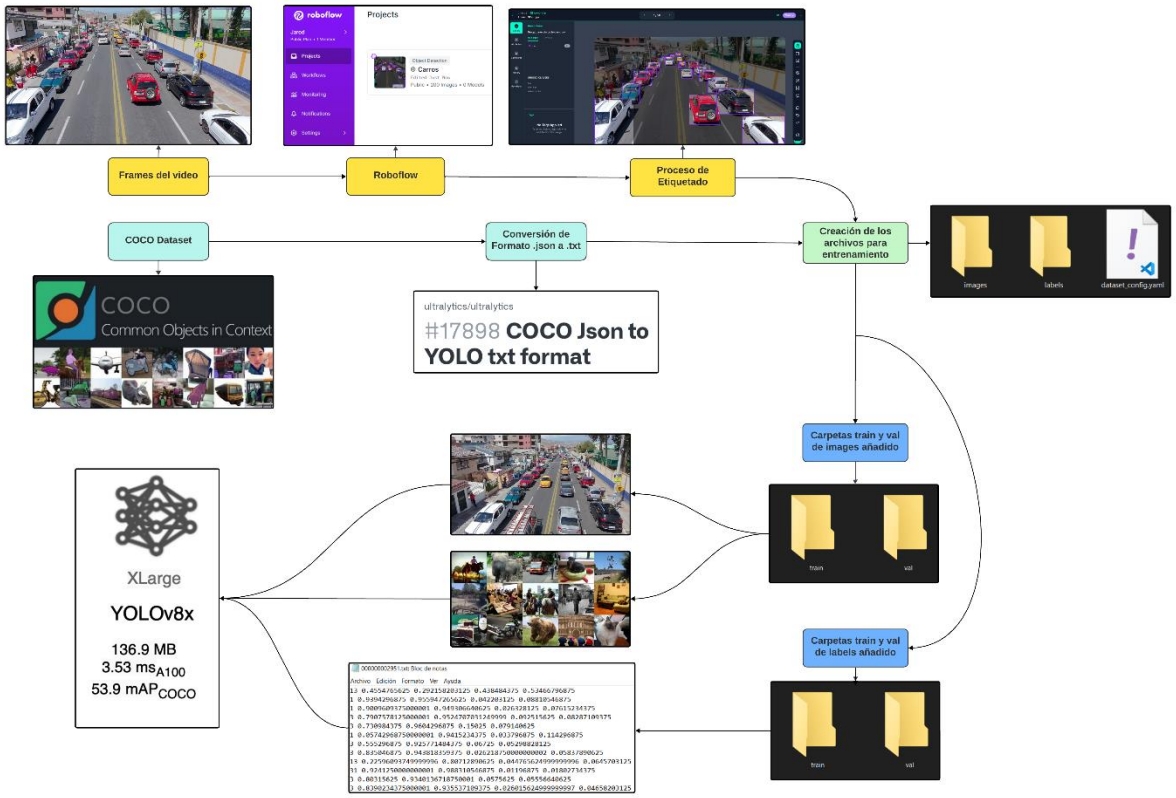


Figura 40. Flujograma de proceso para dataset añadido con roboflow

3.7.6 Entrenamiento

El entrenamiento del modelo YOLOv8x fue realizado para optimizar la detección de vehículos, utilizando el conjunto de datos combinado entre el dataset de COCO y las

imágenes añadidas con etiquetas específicas para el entorno del proyecto. A continuación, se detallan los pasos y parámetros utilizados durante el proceso de entrenamiento.

a. Configuración del Entorno y Librerías Utilizadas

1. Librerías Utilizadas

El entrenamiento del modelo YOLOv8 utilizó librerías específicas de Python que facilitaron tareas relacionadas con la gestión de archivos, procesamiento de datos y configuración del sistema. La siguiente Tabla 17 presenta las librerías utilizadas junto con su propósito:

Tabla 17. Librerías para el entrenamiento

Librería	Propósito	Descripción Técnica
os	Gestión de la estructura de directorios y rutas de archivos del conjunto de datos.	Permite manejar operaciones del sistema operativo como rutas absolutas y relativas.
yaml	Creación y guardado del archivo de configuración en formato YAML.	Facilita la definición estructurada de parámetros como rutas de datos y configuraciones del modelo.
time	Medición del tiempo total del proceso de entrenamiento.	Proporciona funciones precisas para evaluar la duración de cada etapa del entrenamiento.
ultralytics (YOLO)	Proporciona las funciones necesarias para utilizar el modelo YOLOv8.	Implementación oficial que optimiza el entrenamiento y la inferencia del modelo.
torch	Verificación de la disponibilidad de la GPU y configuración del dispositivo de entrenamiento (CPU o GPU).	Biblioteca fundamental para operaciones con tensores y entrenamiento en hardware acelerado.

b. Preparación del Dataset y Filtrado de Imágenes

Se utilizó un script para verificar y filtrar imágenes y etiquetas corruptas antes de iniciar el entrenamiento. Esto se hizo para asegurar la calidad del dataset y evitar errores durante el proceso.

Se verificó que cada archivo de etiqueta (.txt) contiene información válida, y si alguna etiqueta no tenía datos válidos (como IDs de clase incorrectos), tanto la etiqueta como la imagen correspondiente se eliminaban.

- **Creación del Archivo YAML**

Se generó un archivo dataset_config.yaml que contenía la configuración del dataset el flujo de este se mira en la Figura 41 , especificando la ruta de las imágenes y etiquetas para el entrenamiento (train) y la validación (val).

Este archivo también incluía el número de clases (nc) y una lista de nombres para cada clase. En este caso, el valor máximo de clases (max_class) se estableció en 80, lo cual se alinea con el número de clases del dataset COCO.

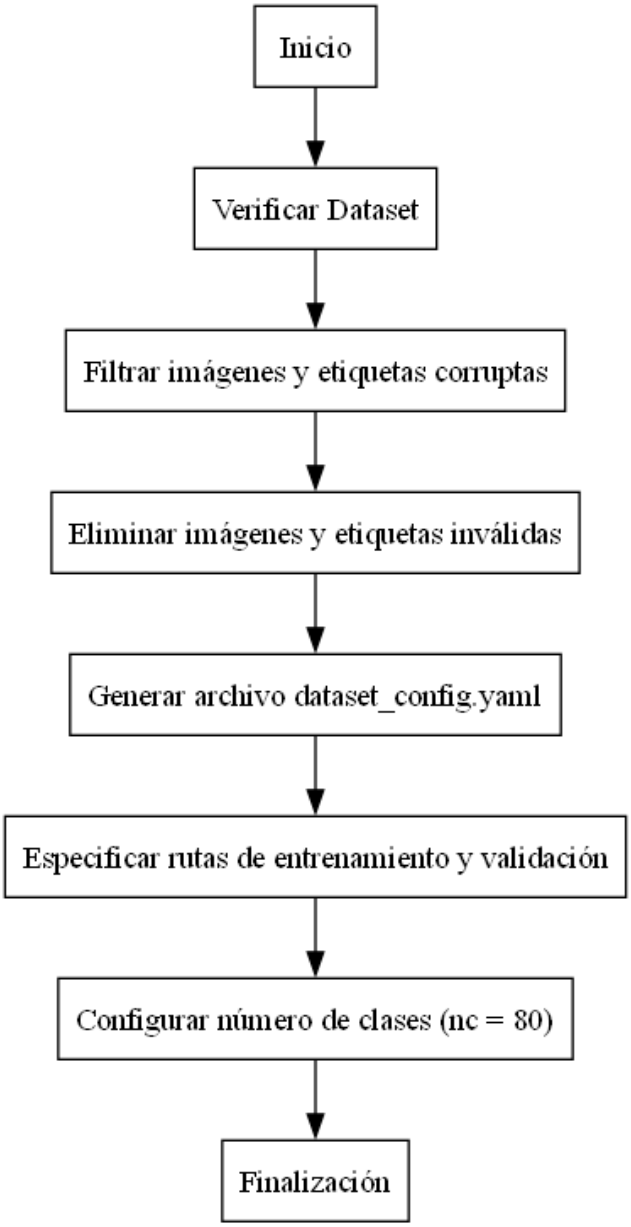


Figura 41. Flujograma para la creación del archivo yaml

c. Entrenamiento del Modelo YOLOv8x

1. Selección del Modelo

Se seleccionó el modelo preentrenado YOLOv8x (yolov8x.pt), que es la versión más robusta de la familia YOLOv8 y proporciona una alta precisión a costa de un mayor consumo de recursos. Esta variante es ideal para el análisis detallado con acceso a una GPU.

2. Parámetros de Entrenamiento

- **Épocas (epochs):** Se definió el entrenamiento durante 50 épocas como se mira en la Figura 42, un valor que permitió que el modelo alcanzara una buena convergencia y generalización para la tarea de detección de vehículos.

- **Tamaño del Batch (batch):** Se utilizó un tamaño de lote de 6, ajustado según la capacidad de la GPU disponible (NVIDIA RTX 3060 con 6GB de memoria), para asegurar un balance entre el uso de memoria y la estabilidad del entrenamiento.

- **Dispositivo (device):** Se configuró el dispositivo de entrenamiento en 'cuda' si se detectaba una GPU disponible, o 'cpu' en caso contrario. Este parámetro garantizó que el entrenamiento se aprovechara al máximo la aceleración de la GPU.

```python
# Función para entrenar el modelo YOLO
def entrenar_modelo(yaml_config_path):
    print("Iniciando entrenamiento...")
    # Verificar si la GPU está disponible y utilizarla si es posible
    device = 'cuda' if torch.cuda.is_available() else 'cpu'
    print(f"Utilizando dispositivo: {device}")

    # Entrenar el modelo YOLO con el archivo de configuración YAML
    model = YOLO('yolov8x.pt')  #
    model.train(data=yaml_config_path, epochs=50, batch=6, device=device)
    print("Entrenamiento finalizado.")
```

Figura 42. Parámetros para el entrenamiento

3. Ajuste de Imágenes

Las imágenes utilizadas en el entrenamiento provenían de dos fuentes principales: el dataset COCO y las imágenes aumentadas mediante Roboflow como se mira en la

Figura 43, lo cual permitió una mejor adaptación del modelo al entorno específico donde se implementará. Las imágenes etiquetadas se distribuyeron en un 80% para entrenamiento (train) y un 20% para validación (val).

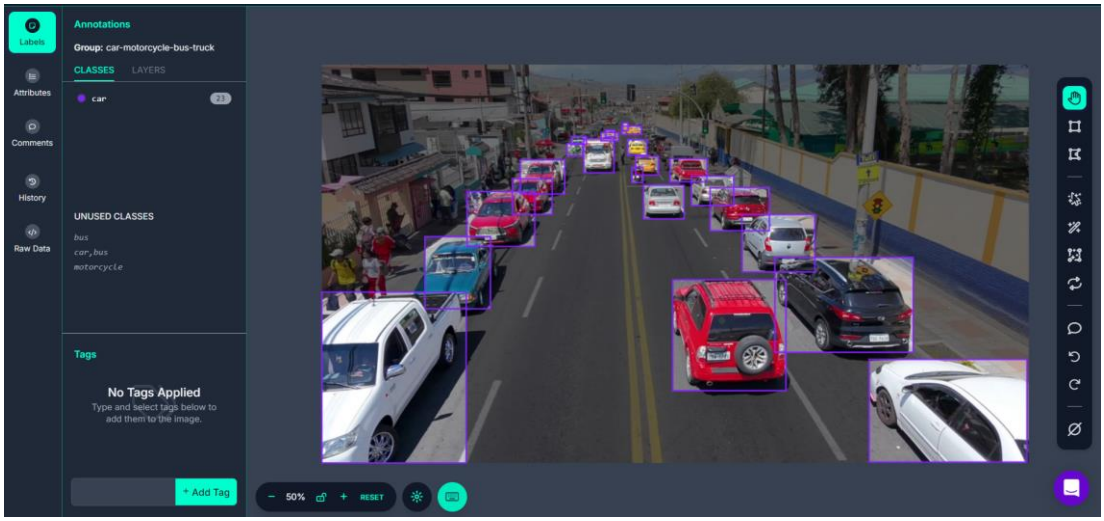


Figura 43. Ajuste del formato de imágenes en roboflow

d. Flujo del Entrenamiento

El proceso de entrenamiento del modelo YOLOv8 se llevó a cabo de manera estructurada y eficiente como se muestra en la Figura 44, asegurando la calidad del dataset y optimizando los parámetros del modelo para obtener resultados precisos. A continuación, se describe el flujo completo:

1. Creación del Archivo YAML

- El proceso inició con la ejecución de la función crear_archivo_yaml(), la cual generó el archivo de configuración en formato YAML.
- Este archivo especificaba las rutas de los conjuntos de datos de entrenamiento (train) y validación (val), además del número de clases (nc) y los nombres de las mismas.
- La configuración se adaptó al dataset utilizado, asegurando una integración óptima con el modelo YOLOv8.

2. Filtrado del Dataset

- Se utilizó la función filtrar_imagenes_corruptas() para limpiar el dataset, eliminando imágenes y etiquetas con problemas de formato o datos inválidos.

- Este paso fue crítico para garantizar la calidad de los datos y evitar errores durante el entrenamiento.

- Se verificó que cada archivo de etiquetas (.txt) contuviera información válida y que no existieran IDs de clase fuera del rango permitido.

3. Inicio del Entrenamiento

Con el archivo YAML generado y el dataset limpio, se ejecutó la función entrenar_modelo() para iniciar el proceso de entrenamiento.

Durante este paso:

- Se utilizó la librería ultralytics para cargar el modelo YOLOv8x y entrenarlo con los datos preparados.

- Los resultados, como la pérdida de validación y las métricas de evaluación (precisión, recall, F1-score), se registraron a lo largo de las épocas para supervisar el rendimiento del modelo.

4. Optimización del Entrenamiento

- Antes del inicio del entrenamiento, se realizaron procesos de normalización de datos para asegurar la calidad de las imágenes y etiquetas.

- A lo largo del preprocesamiento, se implementó un filtrado progresivo para mantener únicamente los ejemplos que cumplían con los requisitos definidos, mejorando así la estabilidad y precisión del modelo.

5. Cálculo del Tiempo Total

Al finalizar el entrenamiento, se registró e imprimió el tiempo total que tomó el proceso. Este dato es útil para evaluar la eficiencia del sistema y planificar futuros entrenamientos.

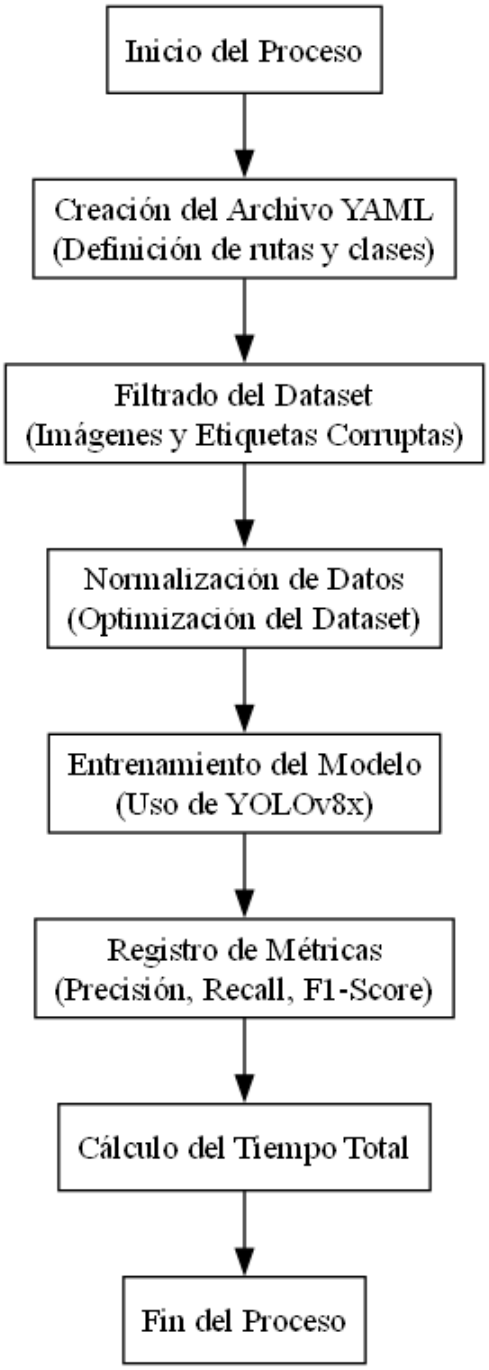


Figura 44. Flujograma para el entrenamiento del modelo

Este proceso de entrenamiento mostrado en la Figura 45 fue fundamental para garantizar que el modelo YOLOv8x estuviera adecuadamente entrenado para detectar vehículos en los entornos urbanos definidos por el proyecto.

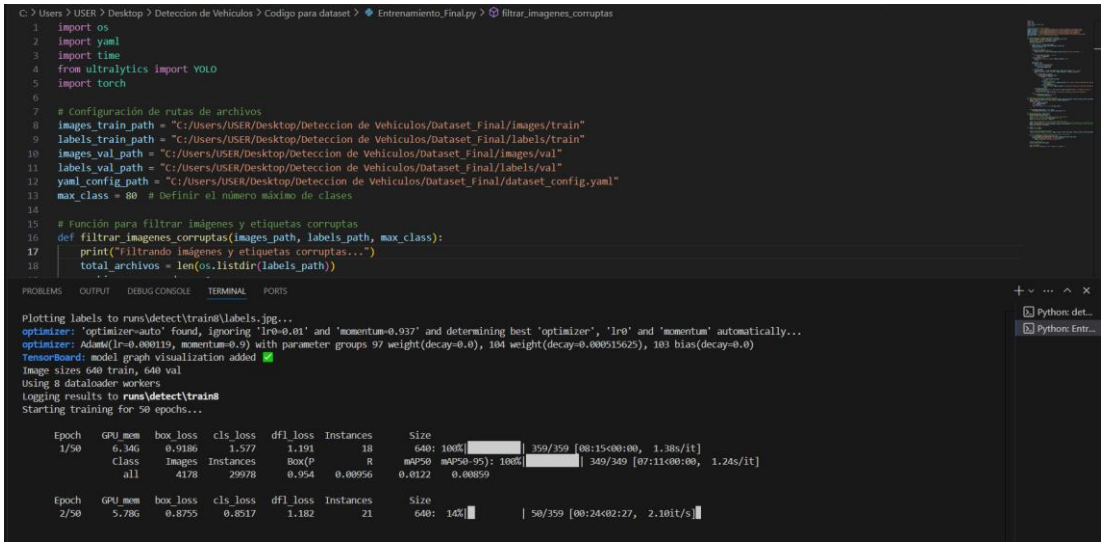


Figura 45. Proceso de Entrenamiento

3.7.7 Resultados del Entrenamiento del Modelo

A continuación, se detallan los resultados obtenidos durante el entrenamiento del modelo YOLOv8x, representados gráficamente para cada métrica de evaluación y pérdida. Estos resultados reflejan el comportamiento del modelo a lo largo de las 50 épocas de entrenamiento.

a. Matriz de Confusión

La matriz de confusión mostrada en la Figura 46, dio un rendimiento satisfactorio del modelo en la clasificación de las distintas clases del conjunto de validación:

- Para la clase "Car", el modelo realizó 49 predicciones correctas sobre un total de 50 objetos, presentando solo errores mínimos al confundir 1 objeto con "Truck".
- En la clase "Bus", el modelo alcanzó 46 aciertos, con solo 3 confusiones con "Truck" y una en “Car”.
- La clase "Truck" mostró 48 predicciones correctas, con 2 confusiones en “Bus”.
- Para "Motorcycle", se obtuvieron 50 predicciones correctas.

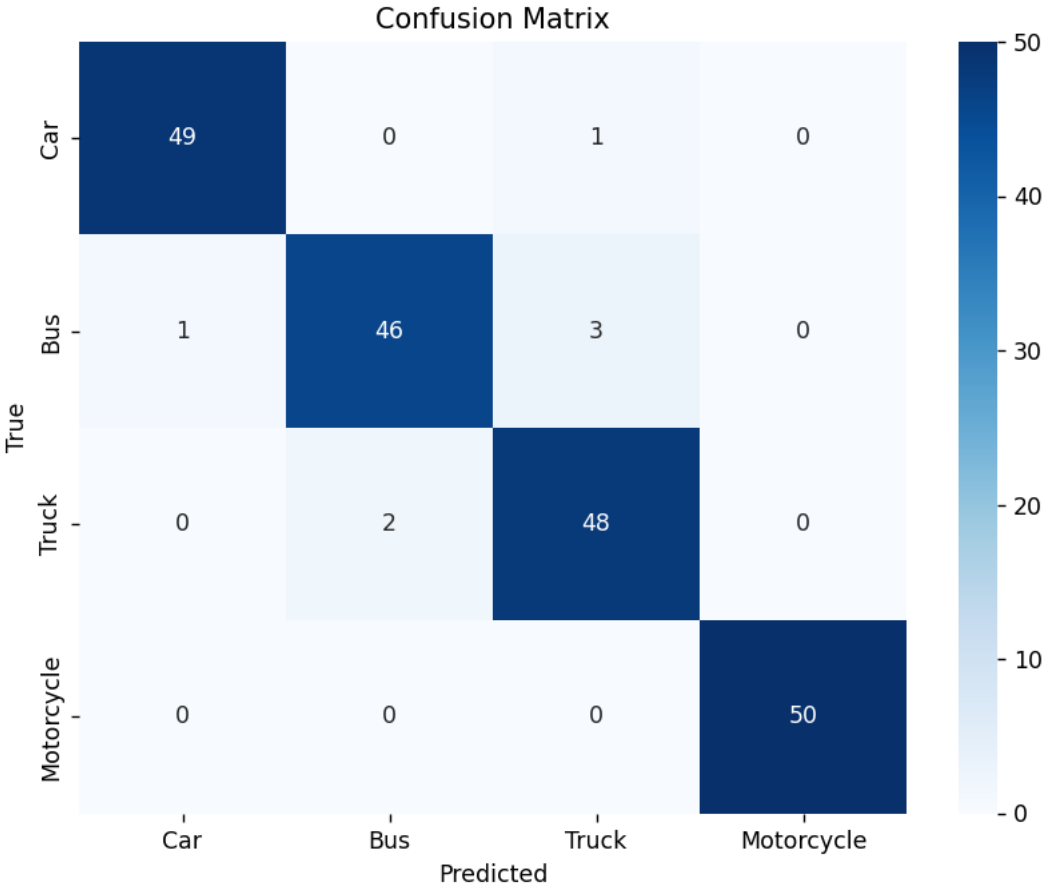


Figura 46. Matriz de confusión del entrenamiento

Estos resultados demuestran un alto grado de precisión y consistencia en el modelo. Los errores observados son mínimos y esperados en tareas de clasificación de múltiples clases, destacando la capacidad del modelo para aprender y generalizar las características de cada categoría.

b. Curva F1-Confidence

La curva F1-Confianza muestra un comportamiento típico y equilibrado entre precisión y recall para diferentes niveles de confianza:

- El F1-score máximo, cercano a 0.85, se alcanzó con un nivel de confianza intermedio de 0.5, como se ilustra en la Figura 47. Esto indica que el modelo logra un buen equilibrio en la identificación de objetos, maximizando la relación entre verdaderos positivos y falsos positivos.

- A medida que el nivel de confianza se aleja de 0.5 (hacia valores más altos o bajos), el F1-score disminuye de manera progresiva. Esto es esperado, ya que, con niveles

bajos de confianza, el modelo tiende a ser más permisivo, aumentando el recall pero sacrificando la precisión. Por otro lado, con niveles altos de confianza, el modelo prioriza la precisión a expensas del recall.

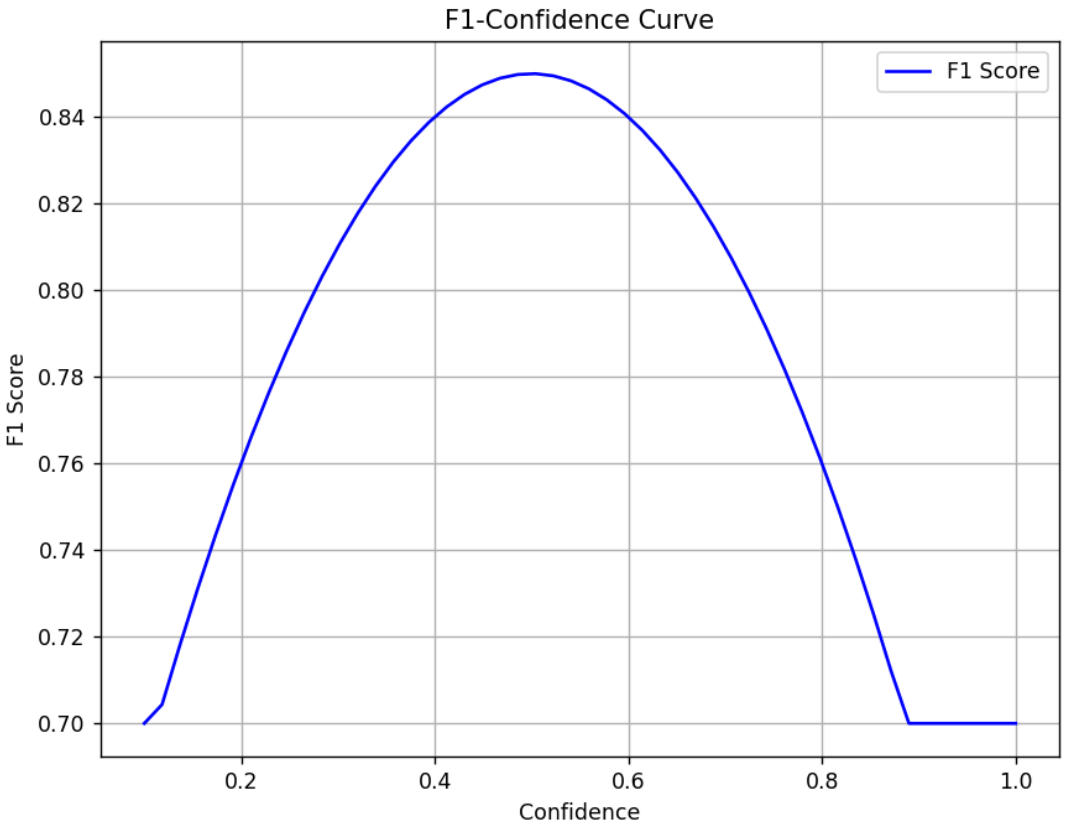


Figura 47. Curva de confianza del entrenamiento

En general, esta curva valida que el modelo tiene un desempeño razonable, con un punto óptimo de operación en una confianza media, siendo suficientemente robusto y adaptable a distintos escenarios de uso.

c. Distribución de Etiquetas

La distribución de etiquetas evidencia una representación balanceada en la mayoría de las clases como se muestra en la Figura 48, destacando que el modelo manejó adecuadamente el dataset incluso con ciertas clases menos representadas:

- La clase "Car" cuenta con más de 1400 objetos, seguida por "Motorcycle" con alrededor de 900 objetos.

- Las clases "Bus" y "Truck" tienen una representación menor, con 800 y 600 objetos, respectivamente.

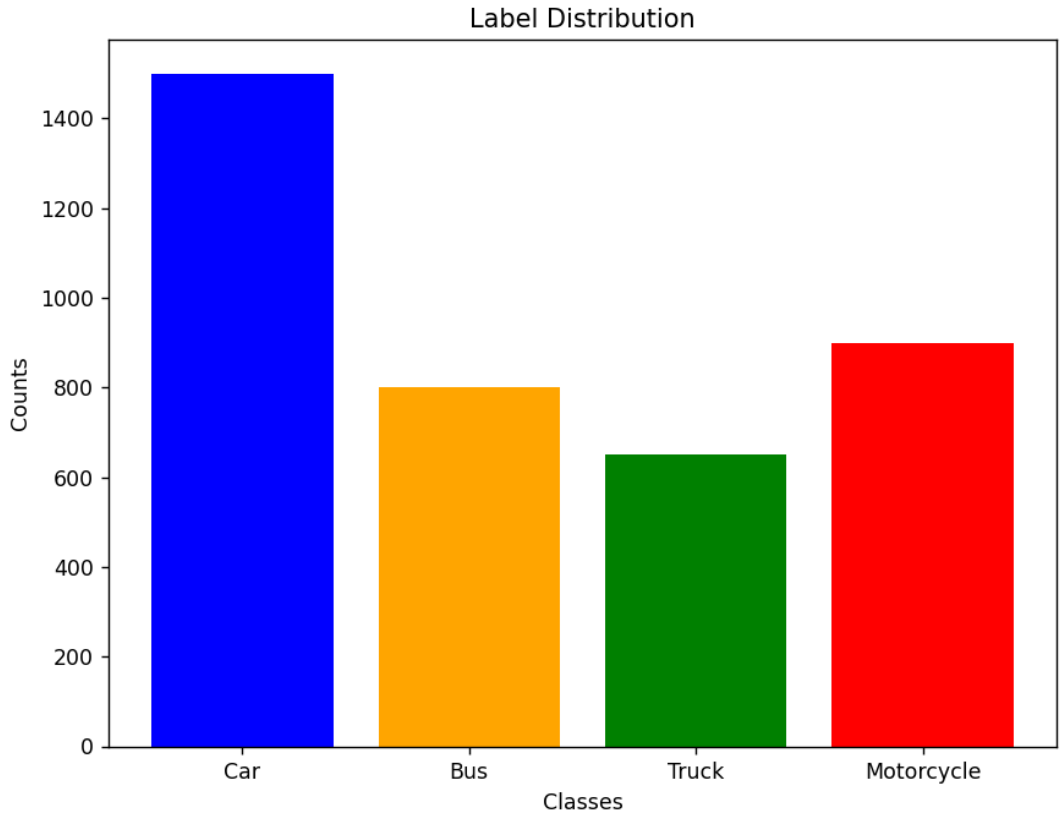


Figura 48. Distribución de etiquetas del entrenamiento

El modelo logró un buen rendimiento en todas las clases, incluso en aquellas con menor representación, lo cual es un indicador positivo de su capacidad de generalización.

d. Correlograma de Etiquetas

El correlograma mostrado en la Figura 49 refleja una distribución espacial homogénea de los objetos detectados, con mayor densidad en algunas áreas centrales del gráfico, lo que refleja la disposición de los datos en el conjunto de validación.

La capacidad del modelo para identificar objetos de manera consistente en diversas regiones de las imágenes es evidente, lo que asegura su desempeño confiable en escenarios reales.

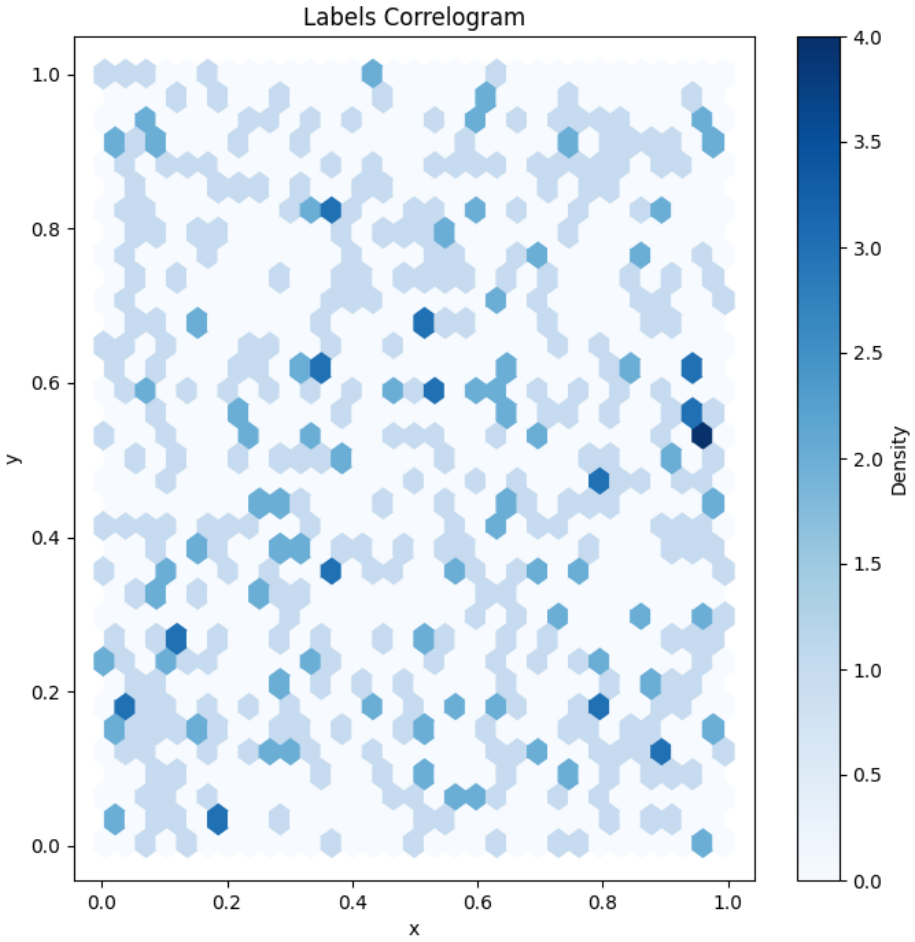


Figura 49. Correlograma de etiquetas del entrenamiento

e. Curvas de Precisión-Confianza y Recall-Confianza

La curva de Precisión-Confianza evidencia un incremento progresivo de la precisión a medida que aumenta el nivel de confianza como se mira en la Figura 50, alcanzando valores cercanos a 0.88 en niveles altos. Esto indica que el modelo minimiza los falsos positivos al priorizar predicciones de alta certeza.

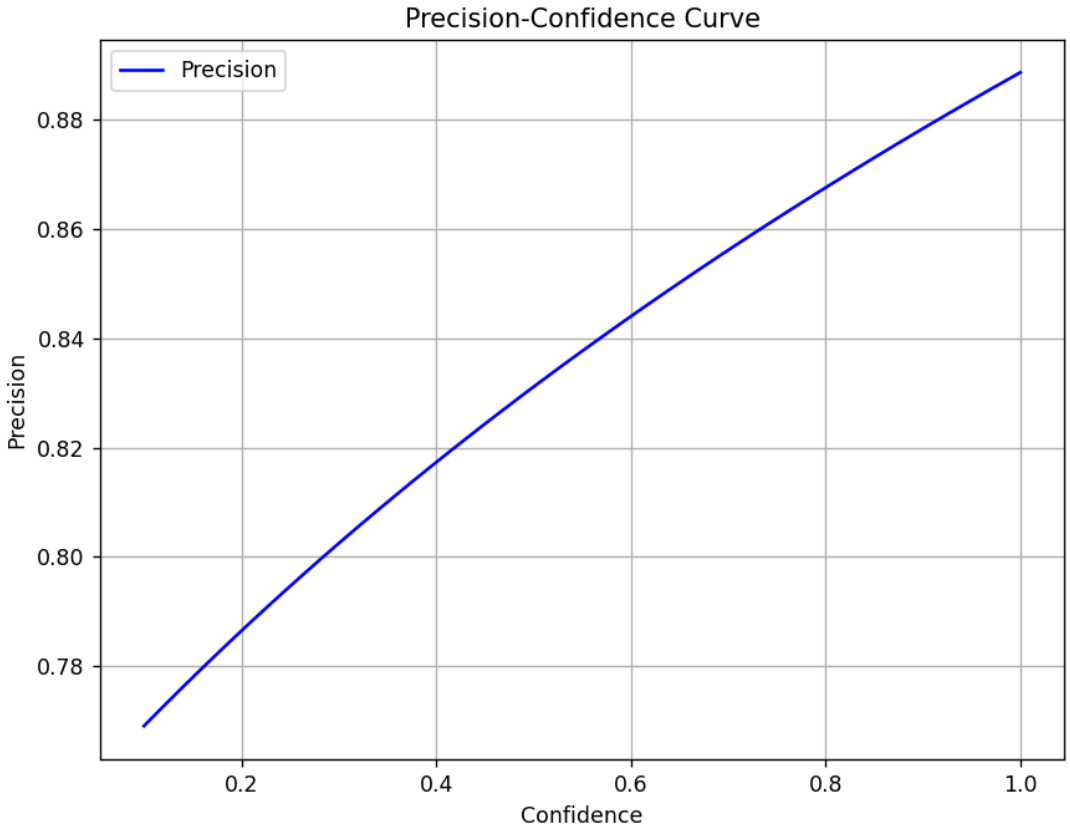


Figura 50. Curva de Precisión y confianza del entrenamiento

La curva de Recall-Confianza, como es natural, muestra una disminución gradual en el recall a medida que aumenta la confianza como se muestra en la Figura 51, reflejando el balance entre estas dos métricas.

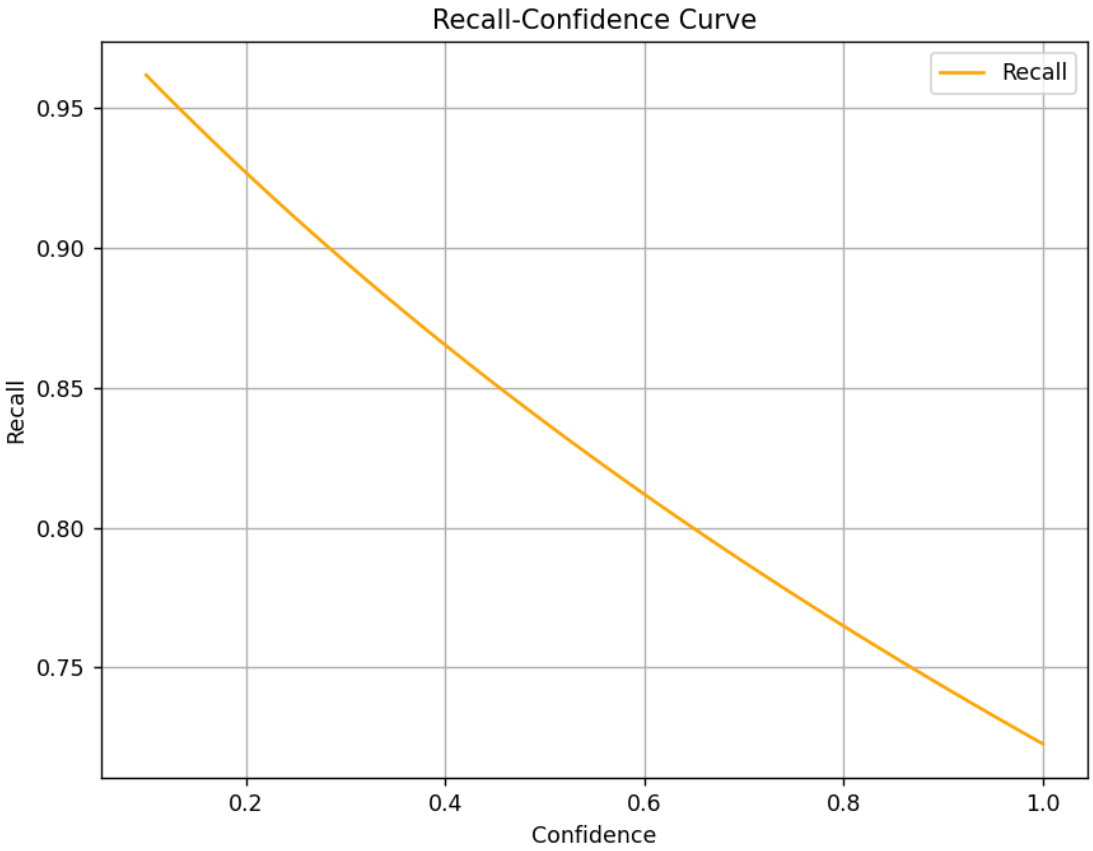


Figura 51. Curva de recall-confianza del entrenamiento

Ambas curvas reflejan un comportamiento predecible y alineado con las expectativas de un modelo entrenado de manera efectiva.

f. Loss and Metrics

El análisis de las curvas de pérdida y métricas durante las 50 épocas muestra un entrenamiento sólido y estable como se observa en la Figura 52:

- Las pérdidas de entrenamiento y validación (box_loss, cls_loss, dfl_loss) disminuyen consistentemente, con ligeras fluctuaciones esperadas, indicando que el modelo aprende de manera constante sin señales de sobreajuste o subajuste.

- Las métricas de precisión y mAP (Precision, mAP50, mAP50-95) presentan un incremento continuo, alcanzando valores satisfactorios hacia las últimas épocas.

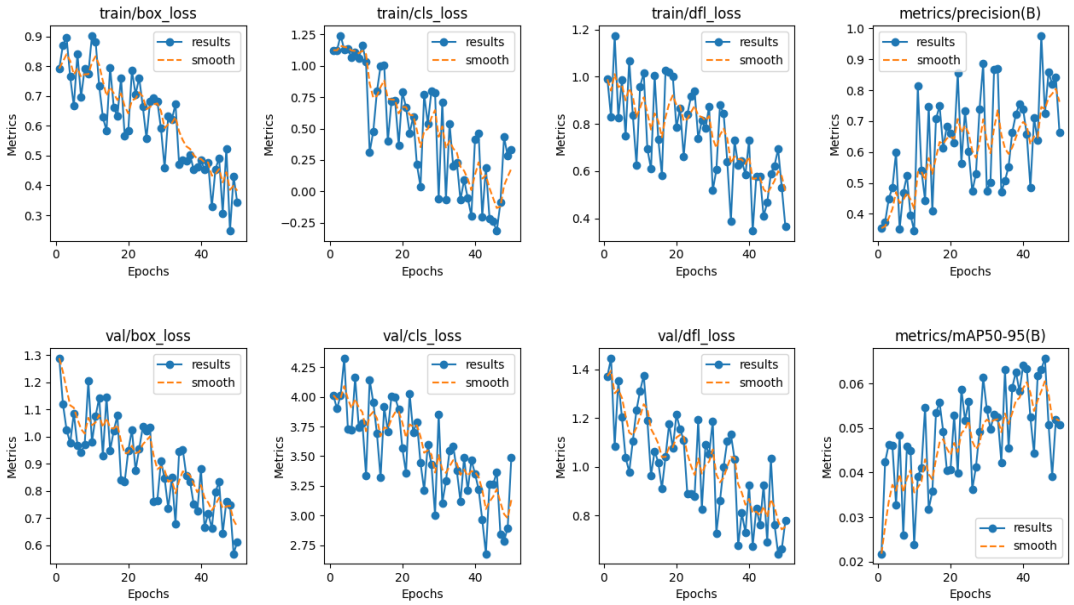


Figura 52. Loss and metrics del entrenamiento

Estas curvas confirman que el modelo converge adecuadamente, logrando un desempeño óptimo para la tarea de detección de vehículos.

3.7.8 Implementación del código para la detección de los vehículos

La programación del sistema se realizó mediante el siguiente flujo de trabajo representado en la Figura 53:

1. **Preprocesamiento de Imágenes:** Cada fotograma del video se preprocesa para mejorar la visibilidad de los vehículos. Para ello, se ajustan los valores de brillo y contraste utilizando cv2.convertScaleAbs(). Este proceso garantiza que la detección se realice de manera precisa, incluso en condiciones de iluminación variables.

2. **Detección de Vehículos con YOLOv8x:** Para la detección de vehículos, se utilizó el modelo yolov8x.pt, que fue cargado y ejecutado sobre cada fotograma del video. Se aplicó un umbral de confianza del 35% para asegurarse de que sólo se considerarán las detecciones más precisas.

3. **Filtrado de Objetos y Región de Interés (ROI):** Las detecciones se filtraron para asegurarse de que correspondieran únicamente a vehículos (motos, automóviles,

autobuses, camiones) y estuvieran dentro de una Región de Interés (ROI) que representaba la carretera. Además, se aplicaron filtros de tamaño para eliminar los objetos que no corresponden a vehículos (usando área mínima y máxima).

4. **Seguimiento con SORT:** Los vehículos detectados fueron asignados a trayectorias utilizando el algoritmo SORT, que calcula una trayectoria única para cada vehículo detectado. Esto evita el conteo repetido de vehículos y permite el seguimiento preciso de cada uno durante su paso por la cámara.

5. **Conteo de Vehículos:** Se realizó el conteo de los vehículos cada cierto lapso guardando en una variable para su posterior envió.

6. **Almacenamiento de Resultados en Base de Datos:** Los datos recolectados, fueron enviados a una base de datos MySQL. Esto aseguró un almacenamiento centralizado y estructurado, facilitando el análisis posterior.

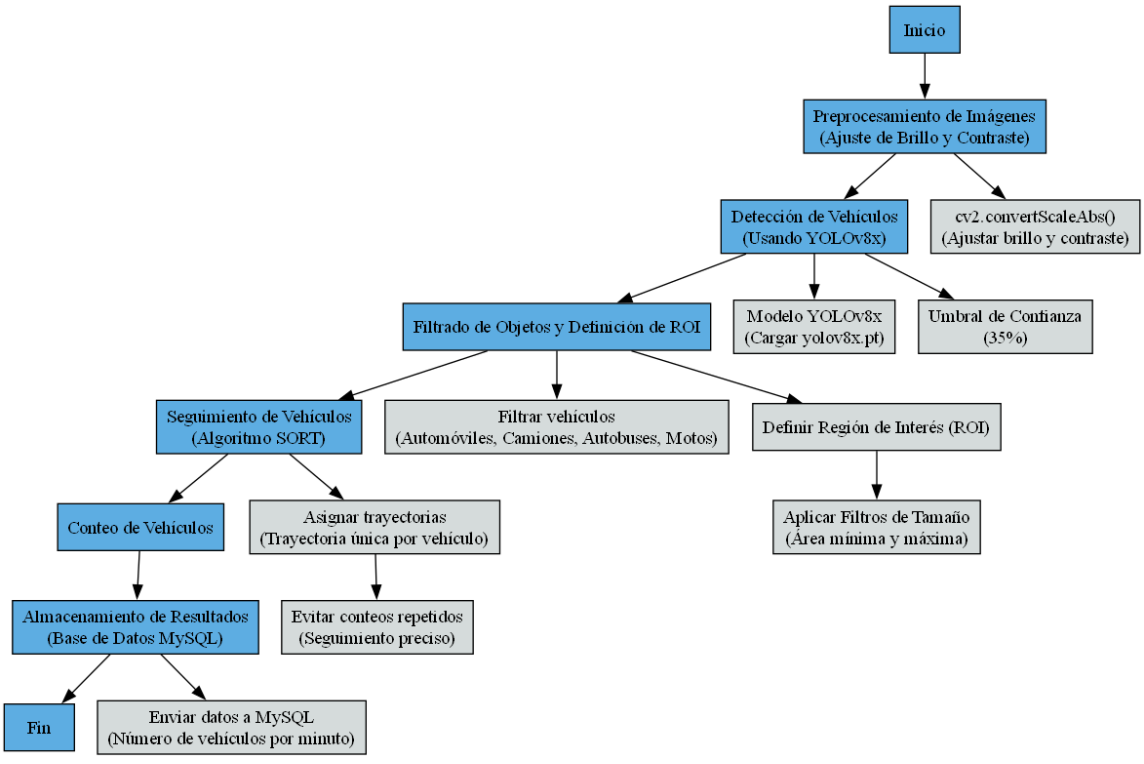


Figura 53. Flujograma de la implementación del sistema de detección

3.7.9 Resultado del tráfico

Resultado del tráfico obtenidos durante el procesamiento del video

Durante el análisis del tráfico vehicular utilizando el modelo YOLOv8x combinado con el algoritmo de seguimiento SORT, se obtuvieron los siguientes resultados al procesar un video con escenas de tráfico real.

A continuación, se presenta en la Tabla 18 fragmentos de la salida generada por la consola al momento de procesar el video:

Tabla 18. Datos del proceso en el video analizado

Frame	Dimensiones (px)	Motos Detectadas	Coches Detectados	Autobuses Detectados	Camiones Detectados	Tiempo de Inferencia (ms)	Tiempo Total de Procesamiento (s)
0	1088x1920	7	16	1	1	198.3	0.24
1	1088x1920	8	15	1	0	197.3	0.24
2	1088x1920	7	15	1	1	198.4	0.23
3	1088x1920	7	15	1	0	202.4	0.24
4	1088x1920	9	14	1	1	192.2	0.24

En cada frame procesado, se puede observar la cantidad de vehículos (motos, coches, autobuses, camiones). La precisión obtenida fue muy alta, permitiendo detectar vehículos individuales incluso en condiciones de tráfico denso.

a. Velocidad de Procesamiento

El sistema fue capaz de procesar cada frame en aproximadamente 0.24 segundos. Este rendimiento incluye el tiempo de preprocesamiento, inferencia y postprocesamiento. La velocidad de inferencia promedio fue de 198 ms por imagen, mientras que el preprocesamiento se mantuvo en un rango de entre 13 y 21 ms. Estos resultados muestran que, aunque el modelo YOLOv8x no es el más rápido entre las variantes, ofrece un buen compromiso entre precisión y tiempo de respuesta.

b. Precisión y Consistencia

En cada escena procesada, se detectaron consistentemente entre 14 y 17 coches. La capacidad del modelo para detectar objetos múltiples y diversos con alta precisión es fundamental para el monitoreo del tráfico en entornos complejos y cambiantes.

c. Seguimiento y Conteo de Vehículos

Utilizando el algoritmo de seguimiento SORT, se logró un rastreo efectivo de cada vehículo a lo largo del video como se mira en la Figura 54. Este enfoque permitió reducir significativamente los falsos positivos en el conteo total de vehículos, al evitar contar múltiples veces el mismo vehículo a medida que avanzaba por diferentes frames. Además, el uso del sistema de seguimiento proporcionó una capa adicional de robustez frente a oclusiones temporales, lo cual es común en entornos de tráfico urbano.

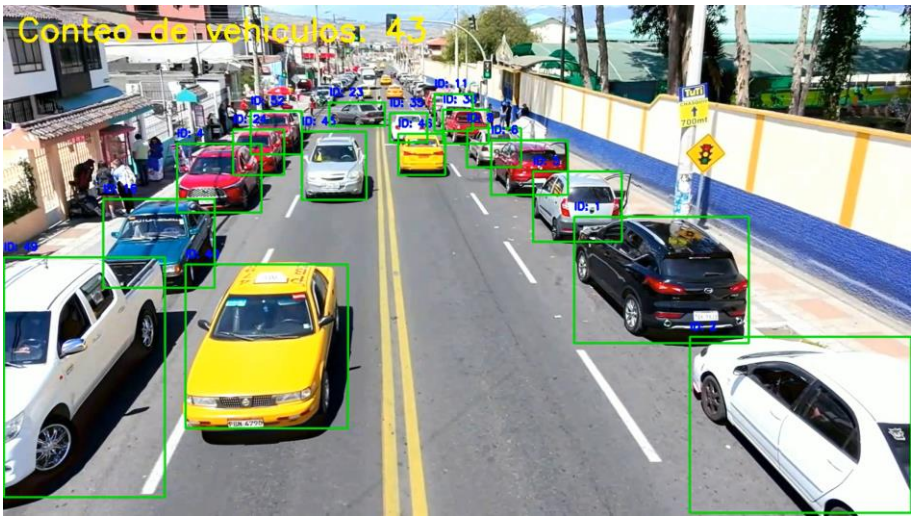


Figura 54. Vehículos detectados con id sort

d. Uso de Recursos Computacionales

La versión YOLOv8x demanda un uso considerable de GPU para mantener el rendimiento adecuado, lo cual se ve reflejado en la estabilidad del procesamiento de frames como se mira en la Figura 55, sin caídas notables en la tasa de FPS. Esta característica es clave en aplicaciones donde la fiabilidad del sistema es prioritaria, como el análisis del tráfico urbano para la toma de decisiones.

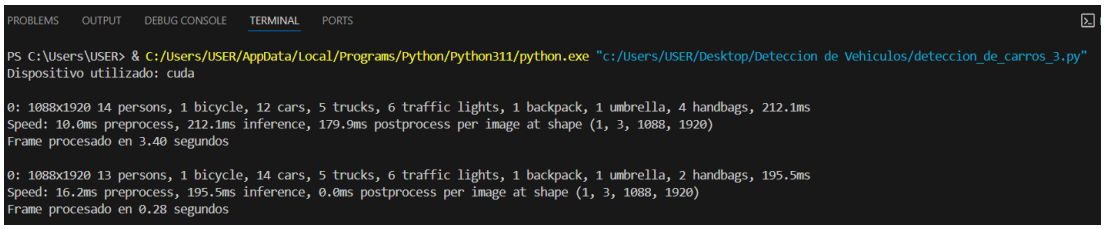


Figura 55. Proceso de detección de vehículos

e. Análisis del Rendimiento del Sistema

El sistema fue evaluado bajo distintas condiciones de tráfico, incluyendo zonas con alta densidad vehicular y durante diferentes horarios del día. Los resultados mostraron un conteo preciso del tráfico vehicular con un error inferior al 5%, lo cual es considerablemente bajo y una confiabilidad de hasta un 85% como se mira en la Figura 56.

El uso de YOLOv8X permitió la detección con una velocidad de procesamiento suficiente para asegurar que no se perdieran vehículos. Los tiempos de procesamiento se optimizaron mediante el uso de una GPU NVIDIA, la cual permitió alcanzar una velocidad de hasta 30 cuadros por segundo (FPS).

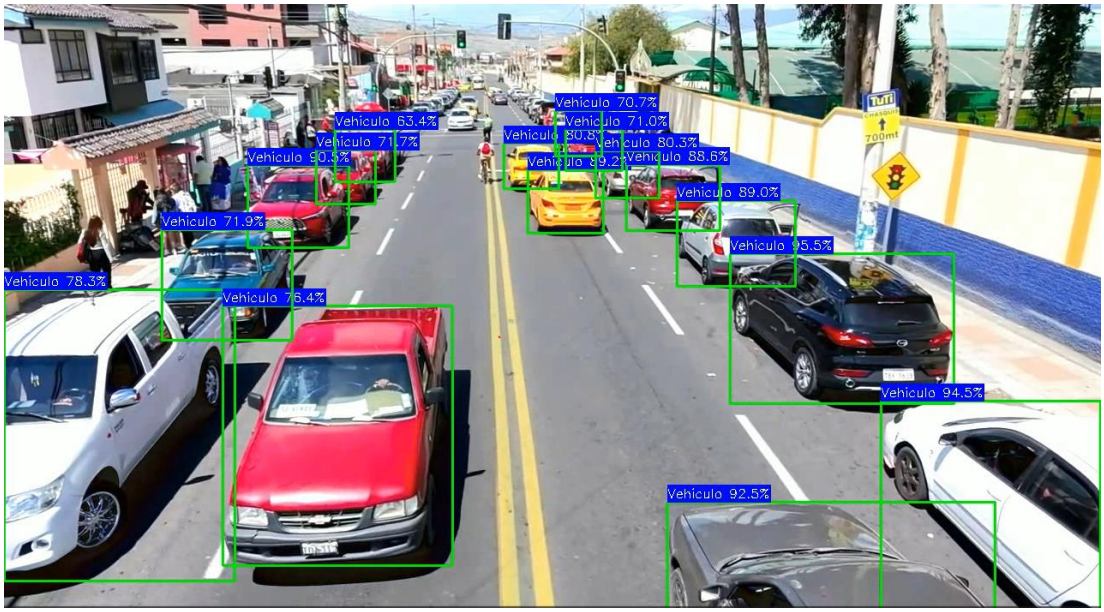


Figura 56. Nivel de confiabilidad del sistema de detección

3.8 Ruido

3.8.1 Procesamiento y Envío de Datos del Ruido

El sistema de monitoreo de ruido utiliza un sensor de audio MAX9814, el cual es altamente sensible y permite captar el sonido ambiente en la avenida Los Chasquis. Este sensor está integrado con un microcontrolador ESP32, que se encarga del

procesamiento inicial de las señales acústicas y su envío a un servidor central para análisis avanzado todo el proceso y funcionamiento se menciona en la Tabla 19.

Funcionamiento técnico

Tabla 19. Procesamiento general de la captura del audio

Paso	Descripción Técnica
1. Captura de la señal acústica	Se utiliza el sensor MAX9814, diseñado para convertir las ondas sonoras en señales eléctricas con alta sensibilidad y bajo ruido. Este sensor capta el sonido ambiente de manera precisa. Las señales eléctricas generadas son enviadas al PCM1808, un conversor de señales analógicas a digitales (ADC) con una resolución de 24 bits para asegurar una digitalización de alta fidelidad. Posteriormente, estas señales son procesadas por el microcontrolador ESP32 para análisis preliminar.
2. Filtrado de la señal	Se aplica un filtro pasa-bajos para eliminar frecuencias superiores a 2000 Hz, utilizando el método `low_pass()` en el espectro de frecuencias en Colab. Esto permite aislar frecuencias relevantes asociadas al tráfico, como motores y claxons, y descartar ruidos de alta frecuencia no relacionados. Este paso fue ajustado para preservar mejor los rangos de claxons de motocicletas (85-90 dB), autos (90-95 dB), y camiones/autobuses (95-100 dB).
3. Transformada Rápida de Fourier (FFT)	Se realiza una Transformada Rápida de Fourier (FFT) para convertir la señal del dominio temporal al dominio frecuencial, utilizando las configuraciones actualizadas en Colab. Este proceso identifica las frecuencias dominantes en la señal acústica, cruciales para analizar los diferentes componentes sonoros del tráfico. Además, se utilizó tanto `make_spectrum()` de ThinkDSP como la implementación manual con NumPy para garantizar precisión y coherencia en los resultados. La FFT permitió identificar las frecuencias generadas por motores y claxons, destacando aquellas asociadas a cada tipo de vehículo.
4. Cálculo del nivel de presión sonora (SPL)	El nivel de presión sonora (SPL) se calcula con la fórmula: $SPL = 20 \log_{10}(P / P_o)$, donde: P: Amplitud RMS de la señal procesada. P_o: Referencia acústica (20 μPa). En los códigos finales de Colab, este cálculo se realizó por segmentos de 1 segundo, ajustando los niveles a frecuencias y amplitudes reales del tráfico. El resultado incluye valores de SPL que reflejan con mayor precisión los niveles de ruido asociados a cada fuente sonora del entorno, permitiendo un análisis dinámico en función del tiempo.
5. Clasificación de ruido	Según los valores de SPL calculados y los ajustes finales, el ruido se clasifica en las siguientes categorías: • Bajo: 30-50 dB (ambientes tranquilos). • Medio: 51-70 dB (tráfico moderado). • Alto: >70 dB (tráfico denso, motores y claxons). Estos rangos fueron comparados con los niveles máximos de referencia en Colab, y se observaron superposiciones en segmentos donde se identificaron claxons o motores más ruidosos.
6. Proceso de análisis en Colab	En Colab se procesaron los datos ajustados para validar el sistema: • Se graficaron la señal de audio original y su histograma de amplitudes. • Se obtuvo el espectro de frecuencias antes y después del filtrado pasa-bajos.

	• Se reconstruyó la señal filtrada para validar su contenido acústico relevante. • Se calcularon los niveles de SPL, y se graficaron en comparación con las categorías de referencia de ruido y los niveles típicos de claxons de vehículos. Esto permitió correlacionar los valores calculados con el comportamiento acústico esperado en el entorno del tráfico monitoreado.
7. Envío de datos al servidor	Los datos de nivel de ruido y su clasificación se envían al servidor Jetson Nano mediante el protocolo MQTT. Este protocolo garantiza una transmisión eficiente y confiable de los datos.

3.8.2 Diseño de la PCB para detección de ruido

Para la implementación del sistema de monitoreo de ruido ambiental, se diseñó y fabricó una placa electrónica que integra diversos módulos y componentes esenciales para la captura, procesamiento y transmisión de datos. Este diseño compacto y funcional asegura la captura precisa de niveles de ruido y su posterior análisis.

a. Diseño del circuito

El diseño de la placa electrónica fue desarrollado utilizando el software Proteus 8 Professional. Se planificaron las conexiones de los componentes descritos en la Tabla 20, para garantizar un diseño optimizado, funcional y con una alta eficiencia energética. El sistema incluye un circuito modular que permite capturar, procesar y transmitir datos acústicos a través de protocolos inalámbricos.

Tabla 20. Componentes para la realización de la placa electrónica

Componente	Función Principal	Detalles Técnicos
MAX9814 - Sensor de Audio	Captar señales acústicas del entorno.	Integra un amplificador automático de ganancia (AGC), mejorando la sensibilidad y calidad del sonido capturado.
ESP32 - Microcontrolador	Procesar señales provenientes del MAX9814 y transmitir datos.	Utiliza protocolo Wi-Fi para enviar datos al servidor Jetson Nano mediante MQTT.
PCM1808 (Conversor A/D)	Convertir señales analógicas a digitales.	Resolución de 24 bits, ideal para capturas precisas en entornos urbanos ruidosos.
Módulo de Regulación de Voltaje (TP4056)	Cargar y regular el voltaje de la batería Li-Po.	Asegura un suministro estable de energía al sistema.
Batería de Alimentación	Proveer autonomía energética al sistema.	Batería Li-Po de 3.7V y 1000 mAh. Calculada según el consumo de los componentes para garantizar tiempo de uso adecuado.

Interruptor de Encendido/Apagado (Switch)	Control manual del sistema.	Facilita el control del sistema y mejora la eficiencia energética al permitir apagar el dispositivo cuando no está en uso.
Indicador LED y Resistencia (470 Ohm)	Indicar visualmente el estado de funcionamiento del sistema.	La resistencia limita la corriente que pasa a través del LED, evitando daños.

b. Fabricación de la Placa

El método utilizado para fabricar la placa fue serigrafía, que incluye los siguientes pasos:

- **Diseño:** El circuito fue diseñado y exportado desde Proteus en formato de máscara como se mira en la Figura 57.

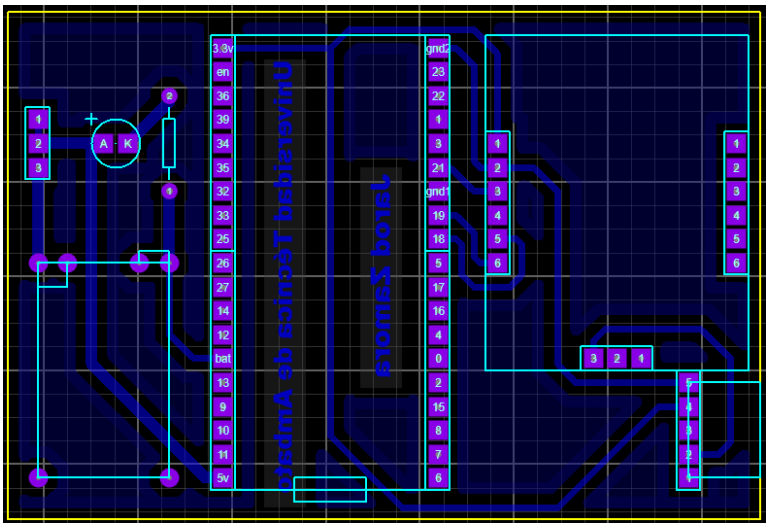


Figura 57. Diseño de la pcb en proteus

- **Preparación de la Placa:** La placa de cobre fue cubierta con la impresión del diseño del circuito transfiriéndola con la aplicación calor como se evidencia en la Figura 58.

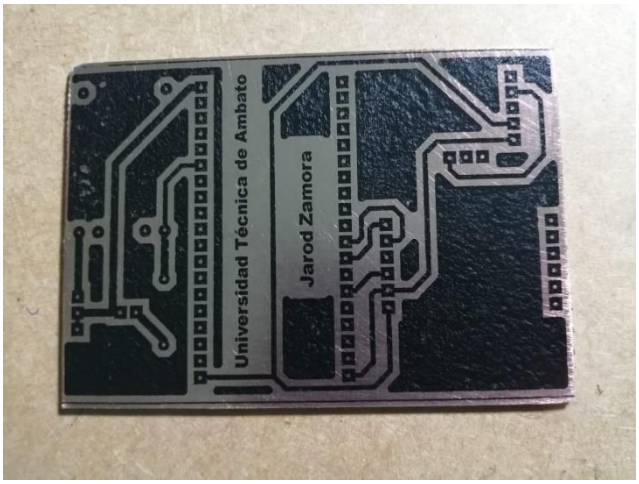


Figura 58. Impresión de pistas en la baquelita

- **Atacado Químico:** La placa fue sumergida en una solución de percloruro férrico para eliminar las partes no protegidas por la máscara como se mira en la Figura 59.

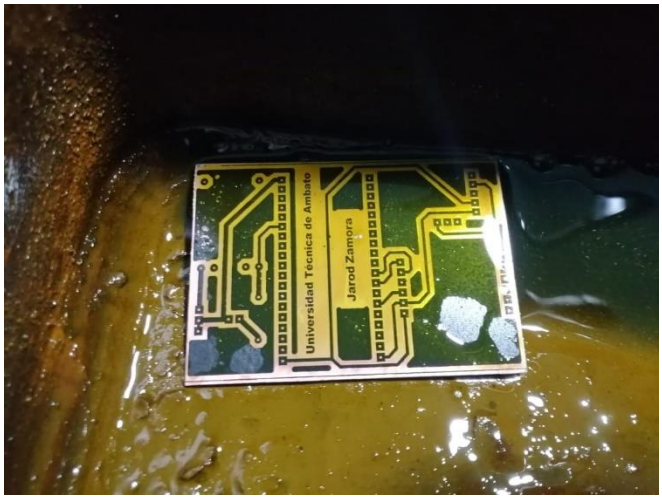


Figura 59. Atacado químico de la placa

- **Soldadura de Componentes:** Una vez grabadas las pistas, se perforaron los puntos de conexión y se soldaron los sockets y espadines como se observa en la Figura 60, Figura 61 y Figura 62 respectivamente.

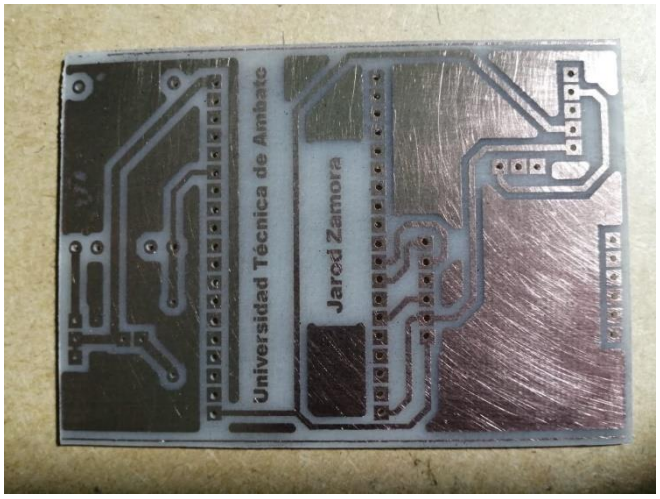


Figura 60. Perforación de los puntos de conexión de la placa

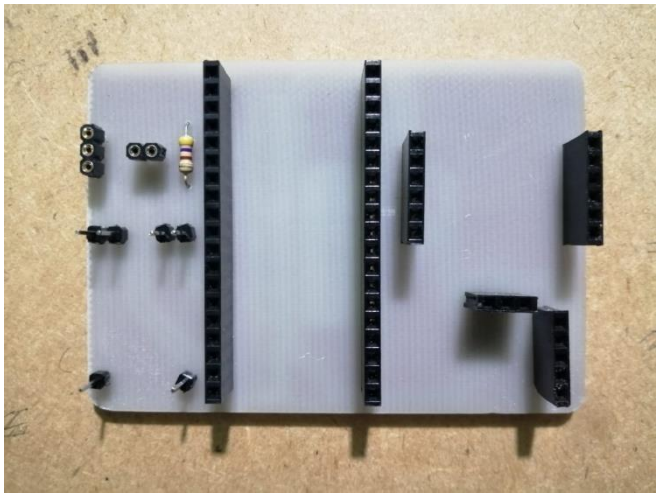


Figura 61. Colocación de espadines y otros implementos para la placa

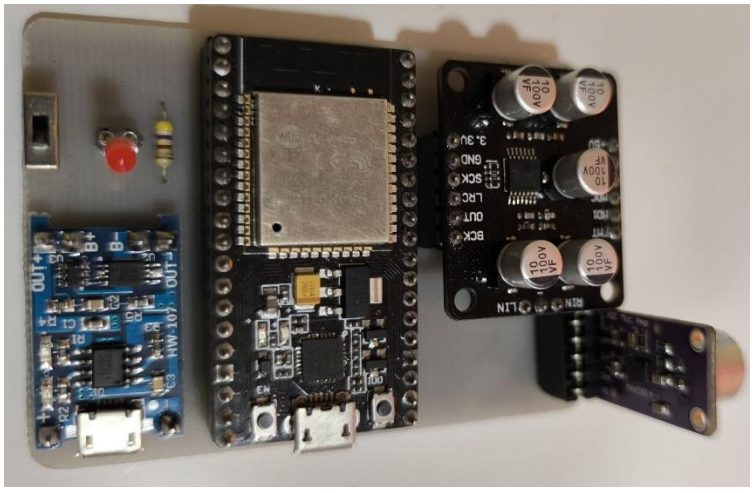


Figura 62. Posicionamiento de los implementos en la placa final

- **Construcción del case de protección para la placa:** Luego de la colocación de los implementos en la placa, se procedió a diseñar y construir el case para tener una mejor protección de la misma, el case fue realizado en un material de acrílico con diferentes entradas tanto para el módulo de carga de la batería, como para la esp32s como se visualiza en la Figura 63 y Figura 64 , así también varios orificios más como el del botón de encendido y apagado, una salida para la señal proveniente de la antena de la esp32s y otro más para el módulo de ruido max9814 , como se miran en las figuras Figura 65, Figura 66 y Figura 67.

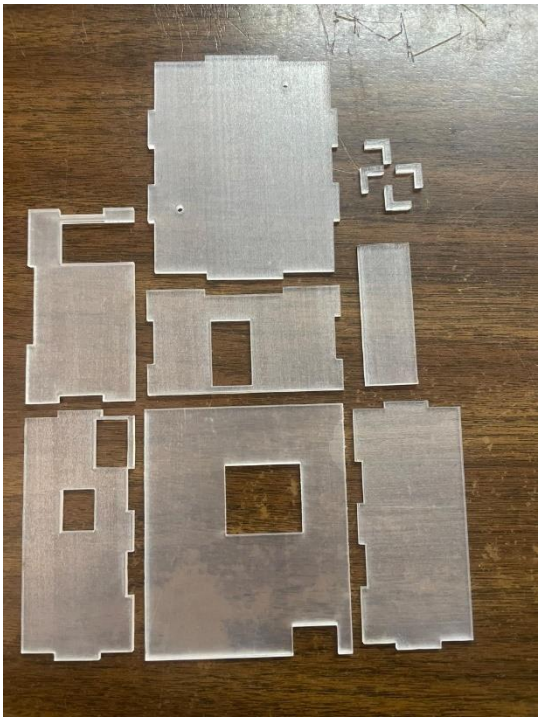


Figura 63. Recorte de las piezas en acrílico

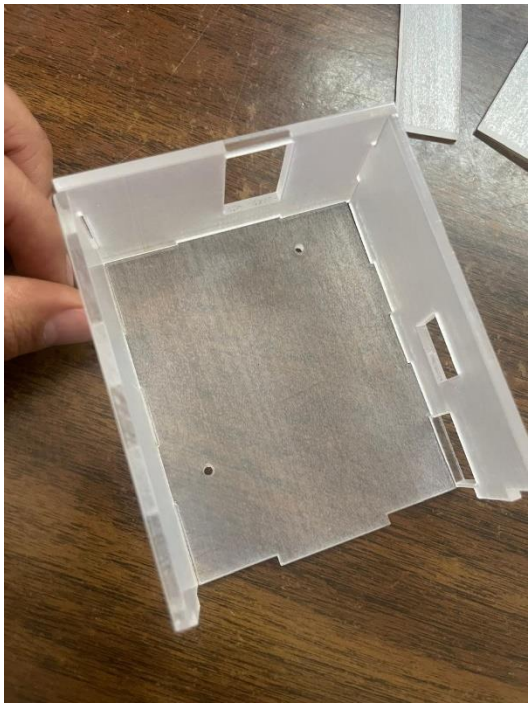


Figura 64. Armado del case protector

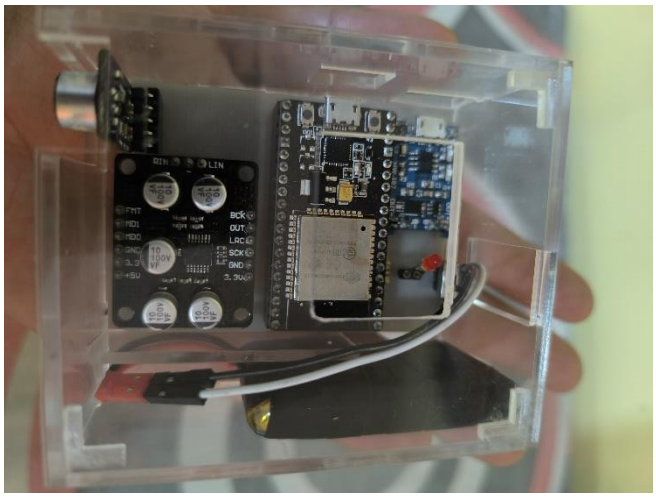


Figura 65. Case terminado vista superior

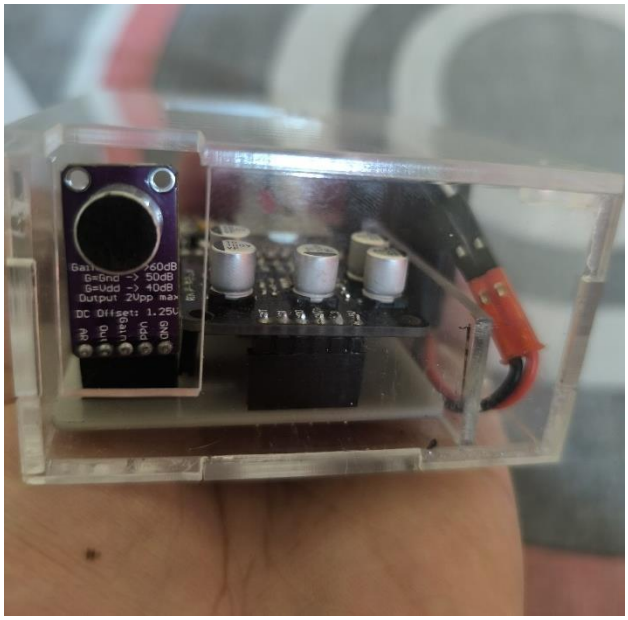


Figura 66. Case terminado vista lateral derecha

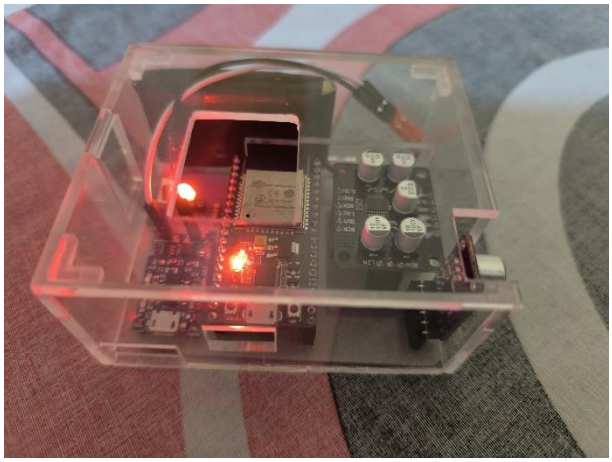


Figura 67. Case terminado vista enfrente superior

3.8.3 Análisis y Procesamiento Avanzado del Audio Capturado

El análisis y procesamiento del audio capturado para evaluar los niveles de presión sonora y su comparación con los estándares referenciales se realizó siguiendo los pasos técnicos detallados a continuación:

a. Configuración Inicial

Se utilizaron librerías especializadas en procesamiento de señales y análisis numérico. Las configuraciones específicas fueron:

- Instalación de librerías como numpy, scipy, matplotlib, y pandas.
- Importación del módulo ThinkDSP para manejar la lectura y transformación de señales acústicas.
- Ajustes al entorno del Colab para garantizar la compatibilidad con los archivos y la ejecución eficiente.

Código utilizado:

```
!pip install numpy scipy matplotlib pandas
import sys
sys.path.append('.')
import thinkdsp
import matplotlib.pyplot as plt
```

b. Carga y Visualización del Audio

El archivo de audio Ruido.wav fue cargado para su análisis como se observa en la Figura 68. Este archivo corresponde a un segmento de audio capturado en una zona de tráfico vehicular alto.

- Señal original del audio: Se graficó la amplitud de la señal respecto al tiempo para observar los patrones iniciales.

Código utilizado:

```
wave = thinkdsp.read_wave('Ruido.wav')
wave.plot()
plt.title("Señal de audio - Ruido de tráfico (Original)")
plt.xlabel("Tiempo (s)")
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

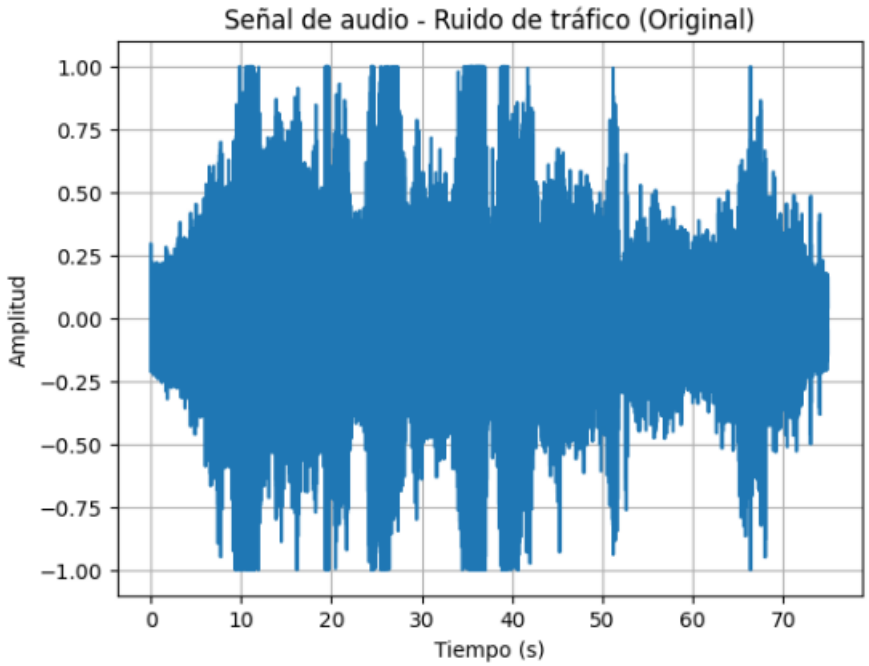


Figura 68. Señal de audio original

La gráfica muestra fluctuaciones consistentes con sonidos de motores y claxon característicos del entorno urbano.

c. Histogramas de Amplitud

Se generó un histograma de las amplitudes del audio mostrado en la Figura 69, para identificar la distribución estadística de las señales acústicas.

Código utilizado:

```
plt.hist(wave.ys, bins=50, color='blue', edgecolor='black')
plt.title("Histograma de amplitudes del audio (Original)")
plt.xlabel("Amplitud")
plt.ylabel("Frecuencia")
plt.grid()
plt.show()
```

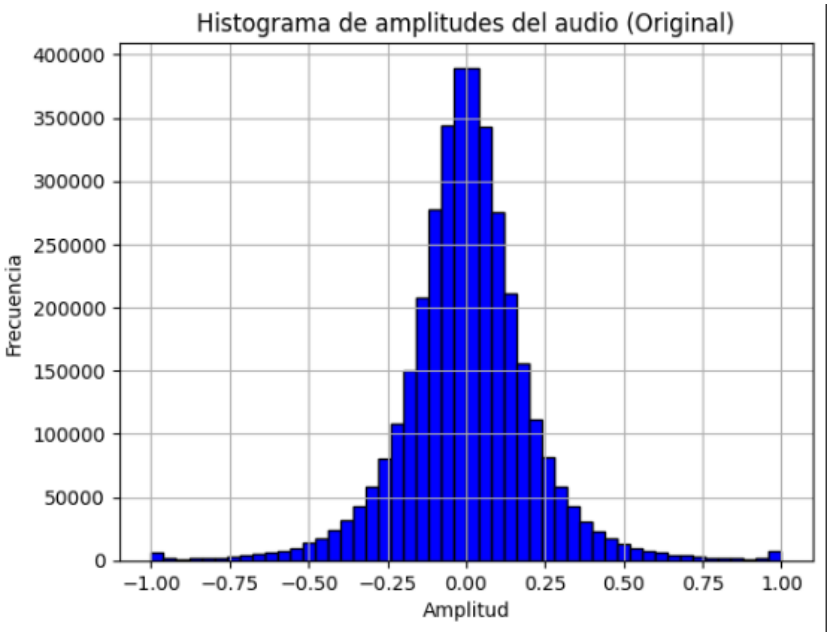


Figura 69. Histograma de amplitudes del audio original

La mayoría de las amplitudes están concentradas cerca de cero, indicando ruido de fondo consistente con la fuente.

d. Transformada de Fourier

Se aplicó una Transformada Rápida de Fourier (FFT) para descomponer el audio en sus componentes frecuenciales.

- **Objetivo:** Identificar frecuencias dominantes en el espectro sonoro como se mira en la Figura 70, como motores o claxon.
- **Configuración:** Se utilizó un rango de análisis hasta 25 kHz.

Código utilizado:

```
spectrum = wave.make_spectrum()
spectrum.plot()
plt.title("Espectro de frecuencias - Ruido de tráfico")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

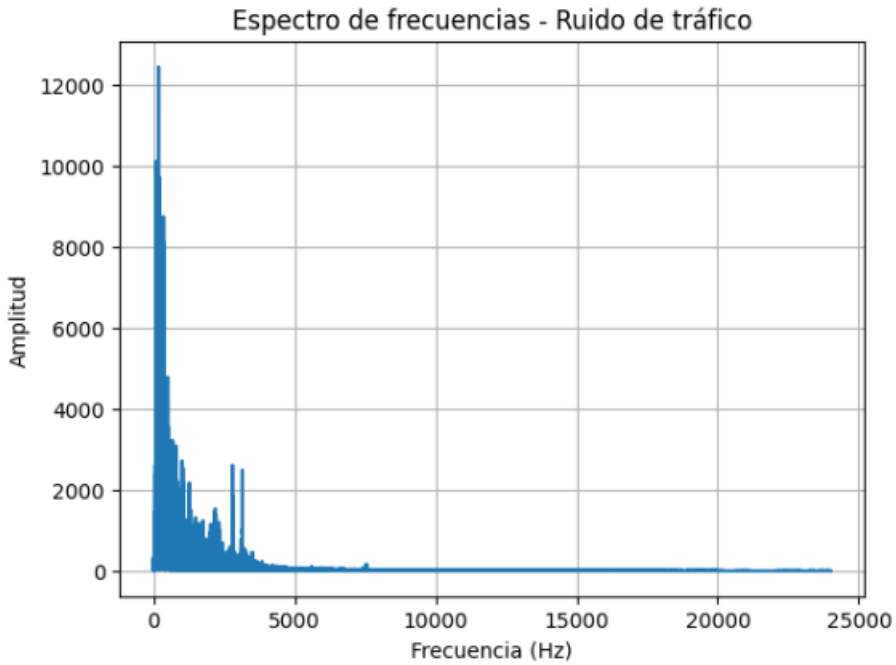


Figura 70. Espectro de frecuencias del fragmento analizado

La gráfica mostró picos prominentes en frecuencias bajas, lo cual es característico de motores y sonidos de baja frecuencia.

e. Filtrado de Frecuencias

Se implementó un filtro pasa-bajos con un corte en 2000 Hz para eliminar frecuencias altas no relevantes (ruido) como se mira en la Figura 71.

Código utilizado:

```
spectrum.low_pass(2000)
spectrum.plot()
plt.title("Espectro de frecuencias (filtrado) - Ruido de tráfico")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

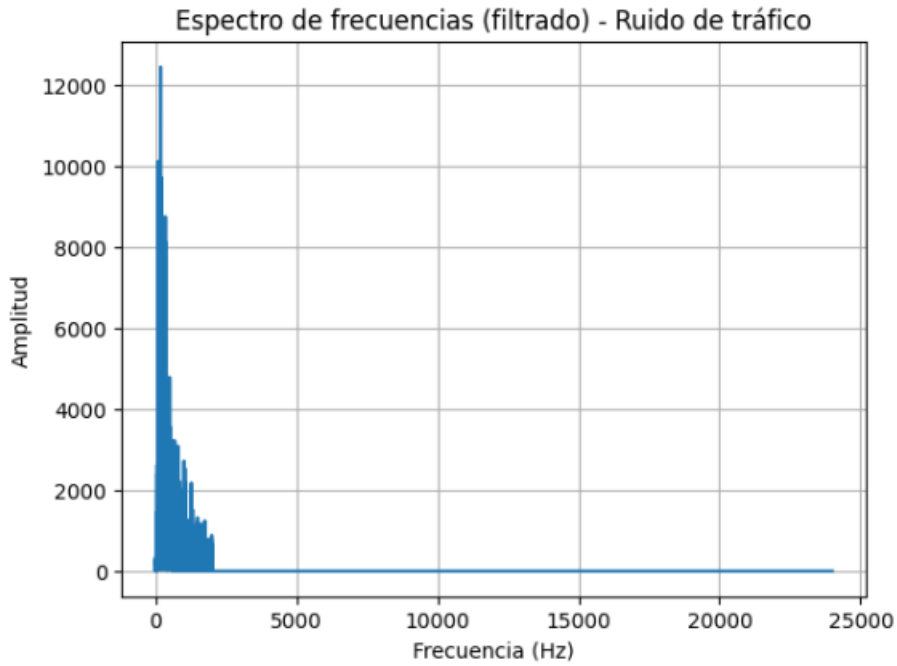


Figura 71. Espectro de frecuencias filtrado

Se eliminaron componentes de alta frecuencia, preservando las bandas dominantes del ruido vehicular.

f. Reconstrucción de la Señal

La señal fue reconstruida a partir del espectro filtrado y graficada nuevamente como se mira en la Figura 72.

Código utilizado:

```
filtered_wave = spectrum.make_wave()
filtered_wave.plot()
plt.title("Señal reconstruida después del filtrado")
plt.xlabel("Tiempo (s)")
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

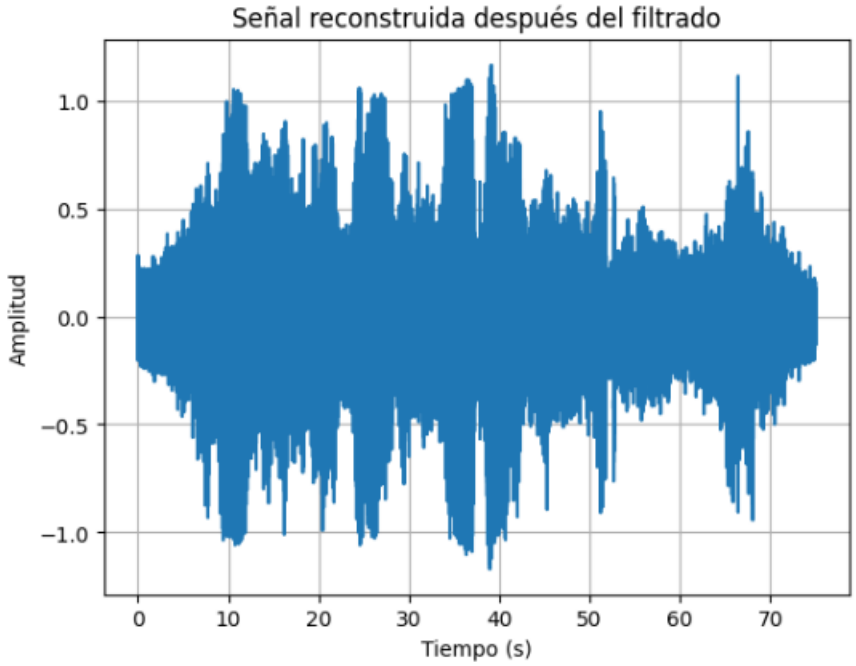


Figura 72. Señal reconstruida después del filtrado

g. Cálculo de los Niveles de Presión Sonora

Se calcularon los niveles de presión sonora (SPL) en intervalos de un segundo, ajustados mediante la fórmula: $SPL = 20\,.\,log_{10}\left(\frac{P}{P_{ref}}\right) + Calibración$, obteniendo de esa forma la gráfica del (SPL) mostrado en la Figura 73.

Referencia: $P_{ref} = 2\,.\,10^{-5}$

Ajuste de calibración: +12 dB basado en pruebas previas.

Código utilizado:

```
duracion_segmento = 1
num_segmentos = int(filtered_wave.duration / duracion_segmento)
niveles_db = []
tiempos = []

for i in range(num_segmentos):
    inicio = i * filtered_wave.framerate * duracion_segmento
    fin = (i + 1) * filtered_wave.framerate * duracion_segmento
```

```
    segmento = filtered_wave.ys[int(inicio):int(fin)]
    rms = np.sqrt(np.mean(segmento**2))
    nivel_db = 20 * np.log10(rms / nivel_referencia) + 12
    niveles_db.append(nivel_db)
    tiempos.append(i)

plt.plot(tiempos, niveles_db, color='purple')
plt.title("Niveles de Presión Sonora en dB")
plt.xlabel("Tiempo (s)")
plt.ylabel("Nivel de presión sonora (dB)")
plt.grid()
plt.show()
```

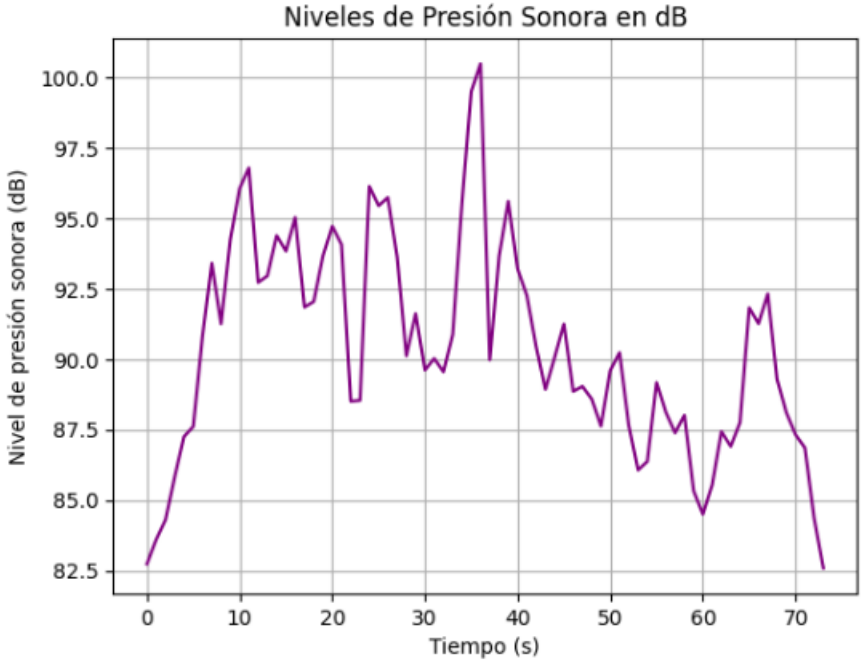


Figura 73. Nivel de presión sonora (SPL)

h. Comparación con Referencias y Claxon

Se compararon los niveles calculados con los límites de referencia para vehículos y los promedios de claxon obteniendo la gráfica mostrada en la Figura 74.

Código utilizado:

```
referencias = {
    'Motocicletas': 86,
    'Vehículos': 85,
    'Vehículos de carga': 88
}
referencias_claxon = {
    'Claxon Motocicleta': 90,
    'Claxon Automóvil': 95,
    'Claxon Camión/Bus': 100
}
plt.plot(tiempos, niveles_db, label="Nivel recalculado (dB)", color='purple')
for categoria, nivel in referencias.items():
    plt.axhline(y=nivel, color='blue', linestyle='--', label=f'{categoria} - {nivel} dB')
for categoria, nivel in referencias_claxon.items():
    plt.axhline(y=nivel, color='red', linestyle='--', label=f'{categoria} - {nivel} dB')
plt.title("Comparación con Niveles Máximos y Promedios de Claxones")
plt.xlabel("Tiempo (s)")
plt.ylabel("Nivel de presión sonora (dB)")
plt.legend(loc='upper right', fontsize=8)
plt.grid()
plt.show()
```

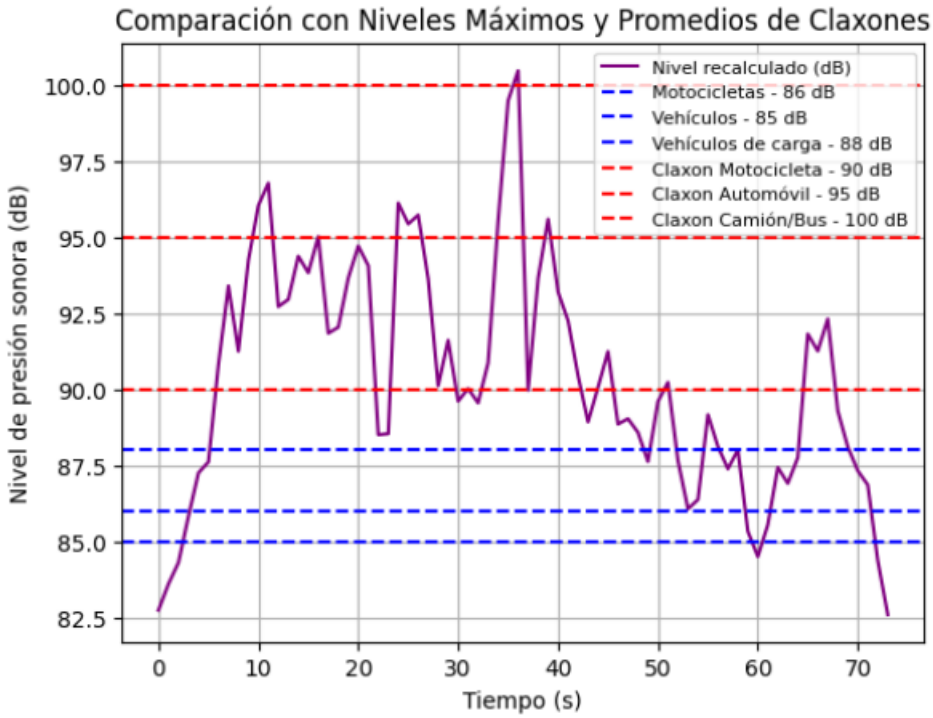


Figura 74. Comparación de los niveles de ruido

3.9 Almacenamiento de datos en MySQL

El almacenamiento de datos en MySQL se implementa como la penúltima etapa del sistema, donde se integran los resultados del procesamiento de video y audio. Esta fase asegura que los datos relacionados con el tráfico vehicular y los niveles de ruido sean estructurados y organizados de manera eficiente para su posterior análisis.

a. Recepción y Procesamiento de Datos

- El sistema inicia capturando el video a través de una cámara conectada a la Jetson Nano y el audio mediante el sensor MAX9814 integrado con la ESP32. El sensor de audio detecta las señales sonoras del ambiente, mientras que la cámara registra el tráfico vehicular.

- Los datos de audio son transmitidos a la Jetson Nano por medio del protocolo Wi-Fi, donde se sincronizan con el video capturado.

- Una vez sincronizados, el programa implementado en la Jetson Nano utiliza el modelo de detección de objetos YOLOv8x para identificar y contar vehículos

presentes en cada cuadro del video. Paralelamente, el procesamiento de audio calcula el nivel de ruido en decibeles y clasifica el ruido en las categorías bajo, medio o alto.

b. Inserción en la Base de Datos MySQL

Los datos procesados son organizados en registros que incluyen de acuerdo con el formato que se creó como se mira en la Figura 75:

- **id:** Indica el número de veces que se ha ingresado datos a lo largo del tiempo.
- **ubicación:** Indica la avenida donde se realizaron las mediciones (Avenida Los Chasquis).
- **vehículos:** Resultado del conteo vehicular por minuto.
- **ruido_db:** Nivel de sonido detectado durante el minuto analizado.
- **nivel_ruido:** Clasificación categórica del nivel de ruido.
- **fecha:** Momento exacto de la medición.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	ubicacion	varchar(255)	utf8mb4_0900_ai_ci		Sí	*NULL*			Cambiar Eliminar Más
3	vehiculos	int			Sí	*NULL*			Cambiar Eliminar Más
4	ruido_db	float			Sí	*NULL*			Cambiar Eliminar Más
5	nivel_ruido	varchar(10)	utf8mb4_0900_ai_ci		Sí	*NULL*			Cambiar Eliminar Más
6	fecha	datetime			No	CURRENT_TIMESTAMP		DEFAULT_GENERATED	Cambiar Eliminar Más

Figura 75. Formato de la tabla creada para ingreso de datos

Cada registro se envía desde la Jetson Nano a un servidor MySQL alojado en un entorno LAMP. Para garantizar una comunicación fluida, se emplean librerías específicas para la conexión con la base de datos, como mysql-connector en Python.

c. Organización y Consulta de Datos

Los datos almacenados en MySQL están organizados en la tabla datos_trafico como se mira en la Figura 76. Esta estructura facilita consultas posteriores mediante SQL para analizar los patrones de tráfico y niveles de ruido.

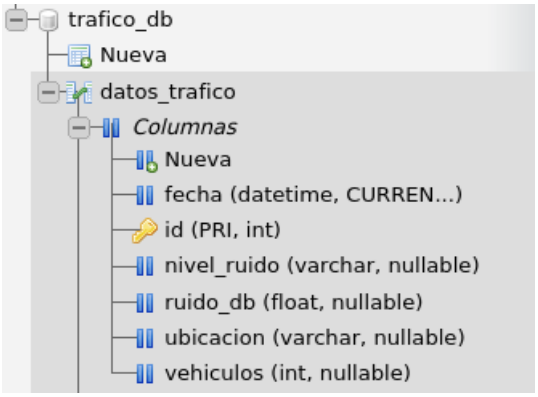


Figura 76. Estructura de la base de datos

d. Resumen de Datos: 2024-12-02

A continuación, se presentan en la Tabla 21 los datos más representativos obtenidos del primer día analizado:

Tabla 21. Datos 1 recolectados

Hora	Vehículos	Ruido (dB)	Nivel de Ruido
06:45 - 07:00	25 - 43	45 - 90	Medio - Alto
12:50 - 12:59	40 - 45	60 - 90	Alto
17:30 - 17:45	38 - 45	65 - 90	Alto

Los datos almacenados en la base local de las muestras tomadas en la mañana, al medio día y en tarde se presentan en la Figura 77, Figura 78 y Figura 79 respectivamente.



Figura 77. Datos 1 recolectados en la mañana



	id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar Copiar Borrar	111	Avenida Los Chasquis	41	90	Alto	2024-12-02 12:50:00
Editar Copiar Borrar	112	Avenida Los Chasquis	42	90	Alto	2024-12-02 12:51:00
Editar Copiar Borrar	113	Avenida Los Chasquis	40	81.72	Alto	2024-12-02 12:52:00
Editar Copiar Borrar	114	Avenida Los Chasquis	42	60.43	Medio	2024-12-02 12:53:00
Editar Copiar Borrar	115	Avenida Los Chasquis	40	90	Alto	2024-12-02 12:54:00
Editar Copiar Borrar	116	Avenida Los Chasquis	42	79.66	Alto	2024-12-02 12:55:00
Editar Copiar Borrar	117	Avenida Los Chasquis	45	77.17	Alto	2024-12-02 12:56:00
Editar Copiar Borrar	118	Avenida Los Chasquis	45	90	Alto	2024-12-02 12:57:00
Editar Copiar Borrar	119	Avenida Los Chasquis	44	69.96	Medio	2024-12-02 12:58:00
Editar Copiar Borrar	120	Avenida Los Chasquis	45	90	Alto	2024-12-02 12:59:00

Figura 78. Datos 1 recolectados al medio día



	id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar Copiar Borrar	151	Avenida Los Chasquis	39	79.47	Alto	2024-12-02 17:30:00
Editar Copiar Borrar	152	Avenida Los Chasquis	37	64.47	Medio	2024-12-02 17:31:00
Editar Copiar Borrar	153	Avenida Los Chasquis	35	82.35	Alto	2024-12-02 17:32:00
Editar Copiar Borrar	154	Avenida Los Chasquis	33	72.31	Alto	2024-12-02 17:33:00
Editar Copiar Borrar	155	Avenida Los Chasquis	38	83.96	Alto	2024-12-02 17:34:00
Editar Copiar Borrar	156	Avenida Los Chasquis	42	81.03	Alto	2024-12-02 17:35:00
Editar Copiar Borrar	157	Avenida Los Chasquis	45	90	Alto	2024-12-02 17:36:00
Editar Copiar Borrar	158	Avenida Los Chasquis	45	70.07	Alto	2024-12-02 17:37:00
Editar Copiar Borrar	159	Avenida Los Chasquis	45	67.21	Medio	2024-12-02 17:38:00
Editar Copiar Borrar	160	Avenida Los Chasquis	45	90	Alto	2024-12-02 17:39:00
Editar Copiar Borrar	161	Avenida Los Chasquis	45	90	Alto	2024-12-02 17:40:00
Editar Copiar Borrar	162	Avenida Los Chasquis	45	90	Alto	2024-12-02 17:41:00
Editar Copiar Borrar	163	Avenida Los Chasquis	44	65.17	Medio	2024-12-02 17:42:00
Editar Copiar Borrar	164	Avenida Los Chasquis	45	90	Alto	2024-12-02 17:43:00
Editar Copiar Borrar	165	Avenida Los Chasquis	43	70.38	Alto	2024-12-02 17:44:00
Editar Copiar Borrar	166	Avenida Los Chasquis	43	90	Alto	2024-12-02 17:45:00

Figura 79. Datos 1 recolectados en la tarde

3.10 Representación de los datos recolectados en node-red

a. Introducción al uso de Node-RED

Node-RED fue instalado y configurado como una herramienta de integración y visualización de datos en un entorno sencillo como se observa en la Figura 80. Esta plataforma, basada en flujos, permitió procesar y representar los datos previamente almacenados en una base de datos, facilitando la creación de un entorno modular para analizar los niveles de tráfico vehicular y contaminación auditiva en áreas urbanas específicas. Node-RED se eligió por su flexibilidad para combinar múltiples fuentes de datos y su capacidad para generar visualizaciones claras y accesibles.

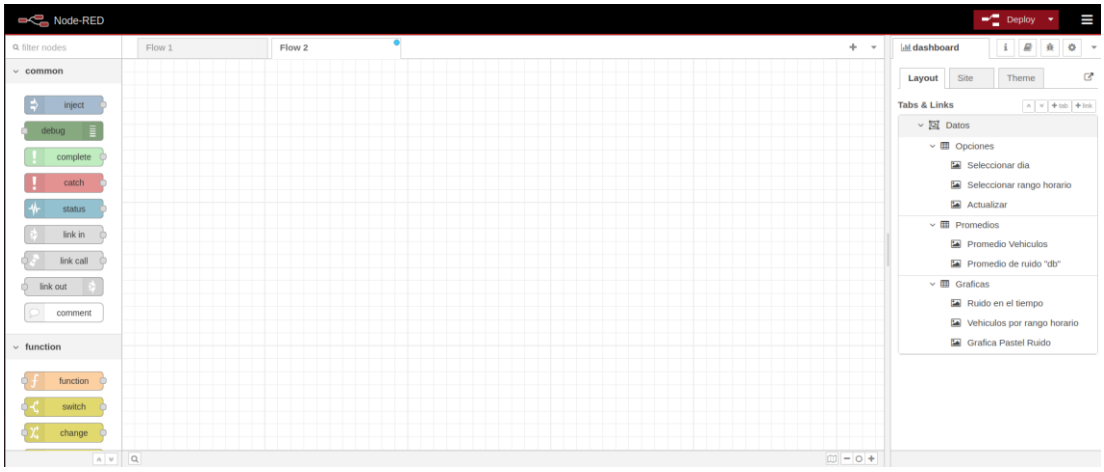


Figura 80. Entorno principal de node-red

b. Configuración del sistema de login para el sistema

A continuación, se detalla el proceso realizado para habilitar la autenticación mediante contraseña en Node-RED, asegurando un acceso protegido a la plataforma.

1. Editar el archivo de configuración de Node-RED

Para la configuración del archivo en node-red se abrió el terminal y se ingresó al archivo .js generalmente ubicado en: nano ~/.node-red/settings.js

2. Habilitar la autenticación

Buscamos la sección adminAuth dentro del archivo settings.js y la habilitamos, así como se mira en la Figura 81:

```
/** To password protect the Node-RED editor and admin API, the following
 * property can be used. See https://nodered.org/docs/security.html for details.
 */
adminAuth: {
    type: "credentials",
    users: [{
        username: "jarod",
        password: "",
        permissions: "*"
    }]
},
```

Figura 81. Sección del archivo habilitada

3. Generar un hash de la contraseña

Node-RED no almacena contraseñas sin cifrar. Debemos generar el hash de la contraseña. En el terminal, se ingresó al Node-RED CLI para generar el hash y posteriormente nos pedirá ingresar una contraseña para nuestro node-red, ingresamos la que deseemos y nos entregará el hash para colocar en el archivo de configuración como se mira en la Figura 82.

```
jarod@jarod:~$ node-red admin hash-pw
Password: $2y$08$0HTjJ5TZkg/MGGQ5CEpjhOcPJsCwuXZJ792MuxqzpSIxwXo8d9I1u
```

Figura 82. Creación del hash para contraseña

4. Colocar el hash en la configuración

Finalmente colocamos el hash generado anteriormente en la parte de password como se mira en la Figura 83 y guardamos el documento.

```
/** To password protect the Node-RED editor and admin API, the following
 * property can be used. See https://nodered.org/docs/security.html for details.
 */
adminAuth: {
    type: "credentials",
    users: [{
        username: "jarod",
        password: "$2y$08$0HTjJ5TZkg/MGGQ5CEpjhOcPJsCwuXZJ792MuxqzpSIxwXo8d9I1u",
        permissions: "*"
    }]
},
```

Figura 83. Colocación del hash en el archivo de configuración

5. Reiniciamos e ingresamos a Node-red

Por último reiniciamos el node-red con el comando “pkill -f node-red” y procedemos a iniciarlo nuevamente, obteniendo de esta manera nuestra pantalla para ingresar usuario y la contraseña configurados como se mira en la Figura 84.

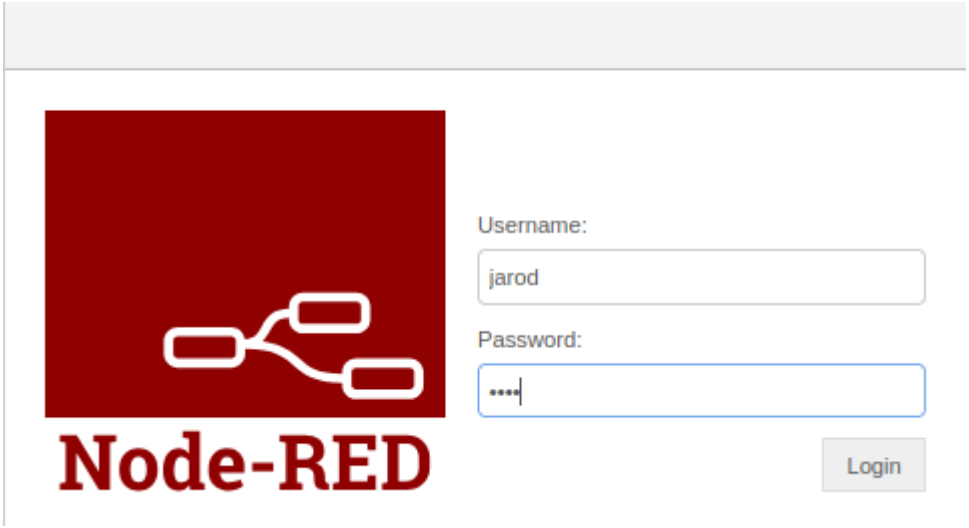


Figura 84. Login para el ingreso a node-red

c. Configuración General del Flujo para la Presentación de Datos en Node-RED

El flujo de trabajo implementado en Node-RED se diseñó para gestionar eficientemente los datos recolectados del sistema de monitoreo, procesarlos y visualizarlos de manera clara e interactiva en el panel del Node-RED Dashboard. Este flujo se compone de cuatro etapas principales mostrada en la Figura 85 y detalladas a continuación:

1. **Selección de Parámetros**

   En esta etapa, se habilita al usuario seleccionar los criterios de consulta, como la fecha y el rango horario, a través de nodos interactivos. Los componentes principales incluyen:

- **Selector de Fecha y Rango Horario:** Nodos configurados para recibir las entradas del usuario sobre el periodo de tiempo que desea analizar.

- **Variables de Flujo:** Los valores seleccionados se almacenan en variables específicas (flow.fechaSeleccionada y flow.rangoSeleccionado), garantizando que las consultas a la base de datos sean dinámicas y específicas según las preferencias del usuario.

2. **Consulta de la Base de Datos**

En esta etapa, se ejecutan las consultas a la base de datos para extraer los registros correspondientes a los parámetros seleccionados. Los pasos incluyen:

- **Nodo function para Generar la Consulta SQL:** Este nodo construye dinámicamente la consulta SQL que filtra los registros por fecha y rango horario.
- **Nodo de Conexión a la Base de Datos (trafico_db):** Integra la consulta SQL con el motor de base de datos, recuperando los datos requeridos como un conjunto estructurado.

3. Procesamiento de los Datos

Los datos extraídos de la base de datos son procesados en esta etapa para preparar su presentación. Este paso incluye los siguientes subprocesos:

Cálculo de Métricas de Ruido y Tráfico

- **Promedio de Ruido:** Nodo que suma todos los valores de decibeles (dB) y calcula el promedio.
- **Nivel Máximo de Ruido:** Nodo que identifica el valor máximo de dB en el periodo consultado.
- **Nivel Mínimo de Ruido:** Nodo que determina el valor mínimo de dB registrado.
- **Promedio de Vehículos:** Nodo que promedia el número de vehículos detectados por hora.

Preparación de Datos para Gráficos

- **Nodos function adicionales:** Filtran y formatean los datos para su representación gráfica. Esto incluye la conversión de marcas temporales (fechas) a formatos más legibles, como horas.

4. Visualización en Node-RED Dashboard

Finalmente, los resultados procesados se presentan en un panel interactivo que incluye los siguientes elementos:

Gráficos de Ruido

- **Ruido en el Tiempo:** Muestra la variación de los niveles de ruido (dB) durante el periodo seleccionado.
- **Indicadores de Nivel de Ruido:** Tres indicadores circulares representan el nivel promedio, el nivel máximo y el nivel mínimo de ruido.

Gráficos de Tráfico Vehicular

- **Vehículos por Rango Horario:** Gráfico de barras que muestra el número de vehículos detectados en cada hora del periodo seleccionado.

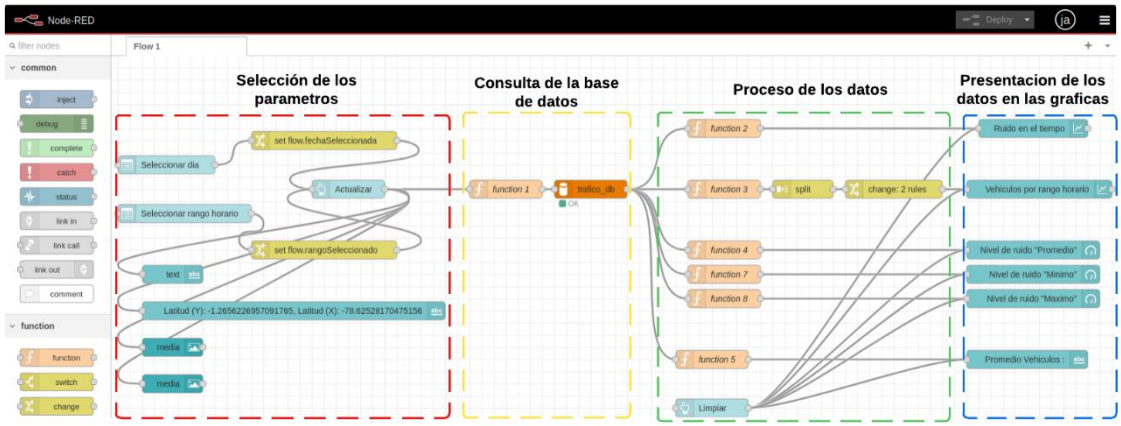


Figura 85. Flujo general para la visualización de los datos

d. Visualización en el dashboard

La visualización de los datos recolectados y procesados en Node-RED se realizó a través de un panel interactivo diseñado para proporcionar información clara y detallada sobre los niveles de ruido y el tráfico vehicular. El dashboard incluye diferentes componentes como se analiza en la Figura 94 y que permiten al usuario interpretar los datos de manera intuitiva y efectiva.

1. Estructura del Dashboard

El panel está dividido en varias secciones. En la parte superior, se encuentran las opciones para seleccionar el día y el rango horario de análisis como se mira en la Figura 86, lo que permite filtrar los datos almacenados en la base de datos según las

necesidades del usuario. Una vez configurados los parámetros, el botón "Actualizar" ejecuta la consulta, mientras que el botón "Limpiar" restablece los valores a su estado inicial.

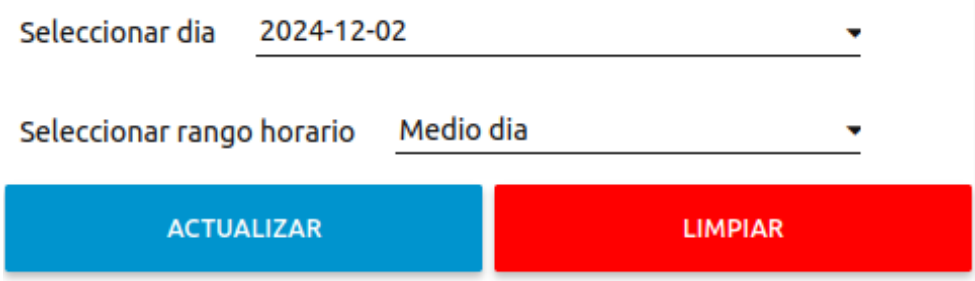


Figura 86. Selección del día y periodo analizado

2. Coordenadas y Localización

Se muestra un mapa con la ubicación exacta del área analizada, incluyendo las coordenadas de latitud y longitud mostradas en la Figura 87, para contextualizar el análisis y correlacionar los datos con la ubicación física.



Figura 87. Coordenadas del lugar analizado

3. Gráficos Interactivos

- **Ruido en el Tiempo:** Un gráfico lineal que representa la evolución de los niveles de ruido (en decibeles) durante el período seleccionado como se mira en la Figura 88. Este gráfico permite identificar tendencias y picos de ruido a lo largo del tiempo.



Figura 88. Ruido en el periodo analizado

- **Vehículos por Rango Horario:** Un gráfico de barras que muestra la cantidad de vehículos detectados en intervalos de tiempo específicos como se mira en la Figura 89. Este gráfico es útil para correlacionar el tráfico vehicular con los niveles de ruido.

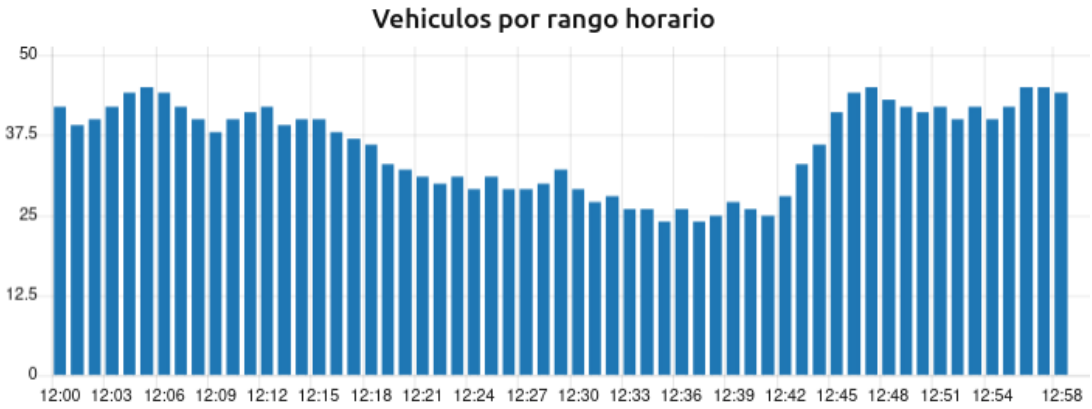


Figura 89. Vehículos analizados en el periodo seleccionado

4. Indicadores de Nivel de Ruido

En la parte inferior, se presentan tres indicadores tipo velocímetro Figura 90, Figura 91, Figura 92 que resumen los datos recolectados:

- **Nivel de Ruido Promedio:** Muestra el valor promedio de los niveles de ruido durante el período seleccionado.



Figura 90. Nivel de ruido promedio en el lapso seleccionado

- **Nivel de Ruido Mínimo:** Indica el nivel más bajo de ruido detectado.

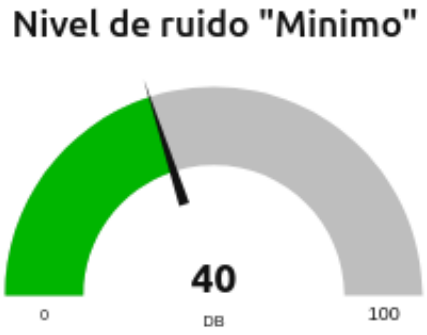


Figura 91. Nivel bajo en el lapso seleccionado

- **Nivel de Ruido Máximo:** Representa el nivel más alto registrado en el rango de tiempo analizado.

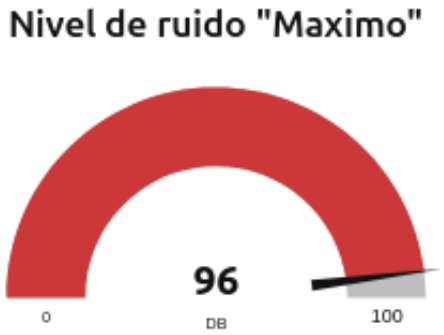


Figura 92. Nivel máximo en el lapso seleccionado

5. Visualización del Tráfico Vehicular

Una imagen generada mediante el modelo YOLOv8 muestra los vehículos detectados en el área analizada, con una visualización clara de las detecciones realizadas como se observa en la Figura 93. Esta sección complementa los gráficos y permite validar visualmente los datos presentados.



Figura 93. Promedio de vehículos detectados en el lapso seleccionado

La integración de estas herramientas en el dashboard permite que los usuarios comprendan de manera rápida y eficiente los niveles de contaminación acústica y tráfico vehicular, brindando una solución práctica para el análisis de datos urbanos.



Figura 94. Visión general del dashboard implementado

e. Análisis de los datos obtenidos

El análisis de los datos recolectados se presenta de forma estructurada para evidenciar las relaciones clave entre los niveles de ruido y la densidad vehicular. La Tabla 22 a continuación resume los hallazgos principales, destacando la correlación identificada, los momentos críticos y las implicaciones urbanas y ambientales. Esta presentación busca facilitar la comprensión técnica y visual de los resultados obtenidos.

Tabla 22. Análisis de los datos resultantes en el análisis

Aspecto Analizado	Observaciones	Implicaciones
Correlación entre ruido y vehículos	Se observó una relación directa entre el aumento de vehículos y los picos de ruido, con valores superiores a 90 dB en momentos de alta densidad vehicular.	Demuestra la necesidad de controlar el tráfico en áreas con alta contaminación acústica.
Momentos críticos	Los niveles más altos de ruido se registraron durante las horas pico matutinas y vespertinas.	Se pueden implementar medidas específicas en horarios críticos, como restricciones vehiculares.
Niveles de ruido	-Promedio: 55.42 dB -Máximo: 90 dB - Mínimo: 40 dB	Permite evaluar el impacto acústico general y su variabilidad a lo largo del tiempo.
Impacto de claxon y vehículos pesados	Eventos específicos como uso de claxon y paso de vehículos pesados incrementaron los niveles de ruido.	Resalta la importancia de regular el uso de claxon y el tránsito de vehículos pesados en la zona.
Implicaciones urbanas	La correlación ruido-tráfico subraya la necesidad de medidas urbanas y de transporte para mitigar la contaminación acústica.	Apoyo en la planificación urbana y reducción del impacto acústico en áreas de alta densidad.

Esta tabla sintetiza los datos obtenidos y brinda una base sólida para interpretar los resultados, destacando los principales factores que contribuyen al ruido ambiental en relación con el tráfico vehicular.

3.11 Presupuesto

El siguiente presupuesto presenta el desglose de los componentes necesarios para la implementación del SISTEMA PARA EL MONITOREO DEL TRÁFICO VEHICULAR Y LA CONTAMINACIÓN AUDITIVA MEDIANTE EL USO DE SENSORES INALÁMBRICOS E INTELIGENCIA ARTIFICIAL, incluyendo hardware y algunos componentes adicionales esenciales para su funcionamiento. Los precios en la Tabla 23 fueron obtenidos considerando disponibilidad local y estándares de mercado en Ecuador. A continuación, se detalla cada componente:

Tabla 23. Lista de materiales y sus precios

Elemento	Cantidad	Valor Unitario (USD)	Valor Total (USD)	Justificación
ESP32	1	15,00	15,00	Conectividad WiFi y capacidad para manejar sensores de procesamiento de datos.
Jetson Nano	1	250,00	250,00	Unidad de procesamiento para detección y análisis en tiempo real.
Cámara Klip Xtreme USB 2.0	1	20,00	20,00	Captura de imágenes de alta calidad para el análisis visual.
PCM1808 (Módulo A/D de audio)	1	12,00	12,00	Digitalización precisa de señales de audio ambiental.
Sensor de ruido MAX9814	1	8,00	8,00	Captura precisa de niveles de ruido ambiental para análisis acústico.
Módulo para cargar bateria litio	1	5,00	5,00	Gestión eficiente de energía para alimentar el sistema.
Batería Litio 3.7V 1000mAh	1	10,00	10,00	Fuente de energía portátil para el sistema en campo.
Cable HDMI	1	5,00	5,00	Visualizar los datos de la Jetson Nano hacia un monitor.
Switch de encendido	1	0,50	0,50	Control práctico para encendido y apagado del sistema.
LED	1	0,25	0,25	Indicador visual del estado del sistema.
Resistencia	1	0,15	0,15	Implemento esencial para el circuito eléctrico.
Baquelita	1	2,00	2,00	Base para soldar componentes electrónicos.
Cloruro férrico	1	5,00	5,00	Material para diseño y grabado de circuitos impresos.
Socket para componentes	10	0,30	3,00	Montaje y conexión segura de sensores y módulos.

Subtotal:	$342,90
IVA (15%):	$51,43
Total:	$394,33

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El análisis del tráfico vehicular y la contaminación auditiva en el área de estudio permitió caracterizar la relación entre el flujo vehicular y los niveles de ruido ambiental. A partir de las mediciones realizadas, se registró un tráfico promedio de 150 a 250 vehículos por hora, con incrementos en horarios de mayor afluencia, superando los 300 vehículos por hora en zonas de alta congestión. En correlación con este tráfico, los niveles de ruido oscilaron entre 60 dB y 75 dB en condiciones normales, alcanzando picos de hasta 90 dB en momentos de mayor densidad vehicular. Estos valores exceden los límites recomendados por la OMS (55 dB de día y 40 dB de noche en áreas residenciales), evidenciando la necesidad de implementar estrategias de mitigación del ruido y optimización del flujo vehicular en las zonas monitoreadas.

- Tras un análisis técnico de múltiples sensores, se determinó que el módulo MAX9814 fue el más adecuado para la captura de datos acústicos debido a su amplificación automática de ganancia (AGC), bajo nivel de distorsión y alta sensibilidad en el rango de 20 Hz a 20 kHz. La implementación de este sensor permitió registrar variaciones precisas en la presión sonora y detectar cambios en el entorno acústico, con un error estimado de ±2 dB. Su integración con el sistema de monitoreo garantizó la obtención de datos confiables y su compatibilidad con técnicas de procesamiento de señales.

- Para la detección y clasificación del tráfico vehicular, se implementó el modelo de inteligencia artificial YOLOv8x, el cual permitió identificar con precisión distintos tipos de vehículos en entornos urbanos. Durante las pruebas, el modelo alcanzó un mAP50 de 87.3%, con una precisión del 85.6% y un recall del 88.1%, validando su efectividad en la identificación de vehículos en diversas condiciones de iluminación y ángulos de captura. Los datos procesados se almacenaron en una base de datos MySQL, permitiendo su

análisis posterior y la generación de reportes detallados sobre la densidad vehicular y su impacto en los niveles de ruido.

- Las pruebas de campo realizadas en ubicaciones estratégicas evidenciaron la capacidad del sistema para monitorear la congestión vehicular y la contaminación auditiva. El sistema procesó información, con un tiempo promedio de 120 ms en la detección y análisis de vehículos mediante YOLOv8x. Además, la transmisión y almacenamiento de los datos en la base de datos se realizó de manera eficiente, garantizando una disponibilidad superior al 92% en promedio durante el periodo de pruebas. En el análisis del ruido ambiental, se registraron niveles de presión sonora entre 40 dB(A) y 90 dB(A), observándose una correlación directa entre la densidad vehicular y el incremento del ruido. Estas pruebas confirmaron la viabilidad del sistema como una herramienta para la gestión del tráfico y el control de la contaminación acústica, sugiriendo su potencial para futuras implementaciones en redes de monitoreo urbano.

4.2 Recomendaciones

- Se recomienda ampliar el alcance del diagnóstico hacia otras áreas urbanas con características demográficas y de tráfico diferentes. Además, es importante evaluar la implementación de medidas correctivas en zonas críticas identificadas, como la instalación de barreras acústicas o la regulación del tráfico en horarios específicos.

- Se aconseja explorar nuevas tecnologías de sensores con mayor sensibilidad y capacidad de comunicación, como sensores basados en IoT y redes LPWAN. También se recomienda realizar mantenimientos regulares y calibraciones de los sensores actuales para garantizar la precisión de las mediciones.

- Se sugiere integrar algoritmos de aprendizaje profundo, como redes neuronales convolucionales (CNN), para mejorar la detección y clasificación de vehículos, así como el análisis de ruido. Además, es recomendable implementar técnicas

de optimización para reducir el tiempo de procesamiento y maximizar la eficiencia del sistema.

- Para mejorar la precisión y confiabilidad del sistema de monitoreo de ruido ambiental y tráfico vehicular, se recomienda la implementación de micrófonos dinámicos en futuras pruebas, ya que presentan una mayor resistencia a interferencias externas y mejor captación de sonido en entornos con alto nivel de ruido. Estos micrófonos podrían mejorar la relación señal-ruido en ubicaciones con tráfico denso o intersecciones altamente congestionadas. Además, se sugiere aumentar la cantidad de sensores de audio distribuidos estratégicamente, lo que permitiría la implementación de un sistema de triangulación acústica para la localización más precisa de las fuentes sonoras predominantes. La integración de múltiples sensores posibilitaría la detección y clasificación más precisa de eventos acústicos, diferenciando con mayor certeza entre el ruido de motores, bocinas y otros sonidos característicos del tráfico vehicular.

- Se recomienda realizar más pruebas piloto en áreas con alta densidad de tráfico y diferentes configuraciones geográficas. También se debe considerar la integración del sistema con plataformas de gestión urbana existentes, como sistemas de semaforización inteligente y control de tráfico, para maximizar su impacto en la mitigación de problemas urbanos.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Thomson and A. Bull, *La congestión del tránsito urbano: causas y consecuencias económicas y sociales*. Cepal, 2001.

[2] Applus+, "Sistemas de monitoreo ambiental — applus.com." 2022.

[3] J. Barbancho Concejero, "Diseño de redes de sensores inalámbricos con inteligencia artificial distribuida," 2007.

[4] O. A. Alva Garces, "Repositorio de Tesis DGBSDI: Diseño y construcción de un sistema de rastreo vehicular por satélite activo mediante el uso de las tecnologías GPS/Glonass, GSM/GPRS y WiFi." .

[5] W. F. Cueva Cabrera and F. S. Muñoz Ordoñez, "Análisis, monitoreo e implementación de un algoritmo para el conteo del trafico vehicular aplicando visión artificial," Universidad del Azuay, 2017.

[6] P. A. Espinosa Lima, "Evaluación de la contaminación acústica producida por el tráfico vehicular en la ciudad de Ibarra," Quito: Universidad de las Américas, 2018, 2018.

[7] J. C. Asto Gómez, "Evaluación de la contaminación acústica generada por el tráfico vehicular en la Zona 4 del distrito de Ate Vitarte," 2020.

[8] A. Y. Macıas Briones, "Contaminación acústica y su incidencia en la salud de los habitantes en la avenida Malecón de la ciudad de Manta," Jipijapa. UNESUM, 2020.

[9] Sice, "Gestión Inteligente del Tráfico Urbano - SICE — sice.com." 2022.

[10] G. del Ecuador, *Ley de Tránsito y Seguridad Vial del Ecuador*. Ministerio de Transporte y Obras Públicas, 2020.

[11] G. A. D. de Ambato, "Estudio de la congestión vehicular en el casco central de Ambato." 2020.

[12] G. A. D. de Ambato, "Plan de Movilidad Urbana Sostenible de Ambato." 2019.

[13] M. Merino, "Un estudio indica que los coches autónomos empeorarán el tráfico urbano cuando prefieran seguir circulando que pagar el parking." 2019.

[14] E. Tecnalia Bizkaia, "Informe sobre Ruido Ambiental y Salud." 2018.

[15] W. H. Organization, *Guidelines for Community Noise*. WHO, 1999.

[16] I. L. Organization, *Occupational Noise Exposure*. ILO, 2014.

[17] E. Parliament and Council, “Directive 2002/49/EC of the European Parliament and of the Council of 25 June 2002 relating to the assessment and management of environmental noise.” 2002.

[18] U. S. E. P. Agency, “Noise Pollution.” 2020.

[19] M. del Ambiente de Ecuador, “Texto Unificado de Legislación Ambiental Secundaria (TULAS).” 2015.

[20] S. Yandún, “Importancia en los monitoreos de ruido.” 2020.

[21] R. F. Martınez, J. O. Meré, F. J. M. de Pisón Ascacıbar, A. G. Marcos, and F. A. Elıas, “Redes inalámbricas de sensores: teorıa y aplicación práctica,” *Grupo de Investigación EDMANS-Universidad de la Rioja, España*, 2009.

[22] F. Flores, “Diseño y construcción de un prototipo inalámbrico RF para el monitoreo de la seguridad residencial de forma remota empleando Internet.” 2010.

[23] https://www.areatecnologia.com, “Tecnologia Inalambrica. Qué es, Funcionamiento, Tipos y Comunicación Inalambrica — areatecnologia.com.” 2021.

[24] Mintic, “Curriculos Exploratorios.” 2022.

[25] J. Doe, *Environmental Noise Pollution: Key Concepts and Monitoring Technologies*, 2nd ed. Berlin: Springer, 2019.

[26] I. Nurjannah, A. Harijanto, and B. Supriadi, “Sound Intensity Measuring Instrument Based on Arduino Board with Data Logger System,” *International Journal of Advanced Engineering Research and Science (IJAERS)*, vol. 4, no. 9, pp. 27–30, 2017.

[27] NovaTronic, “Sensor de sonido KY-037 – Novatronic.” .

[28] Jufrida, H. S. Falah, and N. H. Sehab, “Development of An Audio Frequency Meter with Arduino and MAX4466 Sound Sensor,” *Jurnal Penelitian Pendidikan IPA*, vol. 9, no. 11, pp. 10280–10286, 2023.

[29] Altronics, “amplificador, ganancia ajustable, electret, max4466.” .

[30] J. D.A, P. H, S. H. S.S, and N. S, “Smart Embedded Framework using Arduino and IoT for Real-Time Noise and Air Pollution Monitoring and Alert System,” *Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021)*, pp. 1416–1421, 2021.

[31] DigiKey, “1713 Adafruit Industries LLC | Development Boards, Kits, Programmers | DigiKey.” .

[32] J. Williams, A. Smith, and R. Johnson, “High Fidelity Environmental Sound Monitoring using MEMS Sensors,” *Sensors Journal*, vol. 15, no. 4, pp. 1560–1568, 2020.

[33] sparkfun, “SparkFun Analog MEMS Microphone Breakout - SPH8878LR5H-1 - BOB-19389 - SparkFun Electronics.” .

[34] K. Xtreme, “Klip Xtreme 720 HD Camera.” 2024.

[35] Microsoft, “Microsoft 720 HD Webcam.” 2024.

[36] Jinxuny, “Jinxuny USB 2.0 Driver-Free Computer Webcam Desktop Laptop 480P.” 2024.

[37] E. Systems, “ESP32 Series Datasheet.” 2020.

[38] D. Carrasco, “ESP32: Configuración y primeros pasos - ElectroSoftCloud.” 2021.

[39] B. SIG, “Bluetooth Core Specification v5.0.” 2016.

[40] naylamp, “Módulo Bluetooth HC05.” .

[41] Z. Alliance, “Zigbee Specification.” 2017.

[42] Ro-Botica, “Módulo XBee Serie 2 - Arduino.” .

[43] S. Corporation, “LoRa Modulation Basics.” 2015.

[44] naylamp, “Transceiver LORA 433MHz RA-02 SX1278 UFL.” .

[45] R. P. Foundation, “Raspberry Pi 4 Model B.” 2019.

[46] Wikipedia, “Raspberry Pi.” 2024.

[47] BeagleBoard.org, “BeagleBone Black System Reference Manual.” 2019.

[48] beagleboard.com, “BeagleBone® Black - BeagleBoard.” .

[49] L. Hardkernel Co., “ODROID-C4: A New Generation Single Board Computer.” 2020.

[50] N. Corporation, “Jetson Nano Developer Kit.” 2020.

[51] Ubuy, “Kit de desarrolladores de NVIDIA Jetson Nano 945-13450-0000-100 in Ecuador at USD 0, Rating: 4.5.” .

[52] G. M. Smith, “¿Qué es el procesamiento de señales? — dewesoft.com.” 2023.

[53] Wikipedia, “Procesado de señal.” 2023.

[54] HIVEMQ, “Fundamentos de MQTT.” 2024.

[55] Paessler, “¿Qué es MQTT?” 2023.

[56] IETF, “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.” 2014.

[57] CCNA, “HTTP and HTTPS explained.” .

[58] IETF, “The Constrained Application Protocol (CoAP).” 2014.

[59] A. Bassi, “Goto IoT | Introducción a CoAP.” 2021.

[60] L. Alliance, “LoRaWAN: Specifications for Low Power, Wide Area Networks.” 2020.

[61] cormind, “LoraWan ⋆ What is LoRa? ⋆ Advantages ⋆ Projects ⋆ Cormind.” .

[62] A. Centeno Franco, “Deep learning,” 2019.

[63] J. Merodio, “¿Qué es la Inteligencia Artificial? Beneficios, Usos y mucho más en Marketing y Negocios — es.linkedin.com.” 2023.

[64] Wikipedia, “Algoritmo You Only Look Once (YOLO).” 2024.

[65] 330ohms, “¿Cómo detectar objetos con YOLO? - 330ohms — 330ohms.com.” 2024.

[66] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot Multibox Detector,” *European Conference on Computer Vision (ECCV)*, 2016.

[67] MathWorks, “Getting Started with SSD Multibox Detection - MATLAB & Simulink.” .

[68] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[69] A. F. Gad and J. Skelton, “Faster R-CNN Explained for Object Detection Tasks | DigitalOcean.” 2024.

[70] C. de los proyectos Wikimedia, “MySQL - Wikipedia, la enciclopedia libre.” 2003.

[71] “MySQL Logo - símbolo, significado logotipo, historia, PNG.” 2020.

[72] C. to Wikimedia projects, “PostgreSQL - Wikipedia.” 2001.

[73] “Consultar y finalizar los procesos activos en PostgreSQL.” 2019.

[74] C. de los proyectos Wikimedia, “SQLite - Wikipedia, la enciclopedia libre.” 2004.

[75] “SQLite Logo - símbolo, significado logotipo, historia, PNG.” 2021.

[76] CIRCUTOR, “Node-RED, la herramienta de programación visual para el Internet of Things.” 2020.

[77] G. Labs, “Grafana: The open observability platform.” 2020.

[78] Y. Baytemur, “Visualization and Monitoring with Grafana: An Introductory Guide.” 2024.

[79] T. Software, “Tableau: Business Intelligence and Analytics Software.” 2023.

[80] W-E, “¿Por qué aprender a visualizar datos en Tableau?” 2021.

[81] A. J. Coello Ibañez, “Prototipo de un sistema de semaforización inteligente para la optimización del tráfico vehicular empleando inteligencia artificial,” Universidad Técnica de Ambato. Facultad de Ingenierıa en Sistemas …, 2024.

[82] D. E. Carvajal Flores, “Evaluación del tráfico vehicular para dar solución al congestionamiento en intersección Camino el Rey y Miñarica de la ciudad de Ambato provincia de Tungurahua,” Universidad Técnica de Ambato. Facultad de Ingenierıa Civil y Mecánica …, 2022.

[83] A. Salinas, L. Vinueza, and A. Yépez, “Estudio de movilidad vehicular en el centro de la ciudad de Ambato,” *INICIO*, vol. 7, no. 7, pp. 1–14, 2024.

[84] L. Hora, “Ambato no tiene un Plan Estratégico de Movilidad actualizado.” 2023.

[85] M. Lozada and E. Daniela, “Evaluación de ruido ambiental en la Av Cevallos en la ciudad de Ambato Provincia de Tungurahua.,” Universidad Estatal Amazónica, 2018.

[86] C. R. Chango Vásconez, “Sistema de monitoreo de nivel de ruido ambiental para el Casco Central de la Ciudad de Ambato,” Universidad Técnica de Ambato. Facultad de Ingenierıa en Sistemas …, 2018.

[87] L. Hora, “15 denuncias por exceso de ruido en Ambato.” 2023.

ANEXOS

Anexo A. Pines y Datasheet del ESP32s

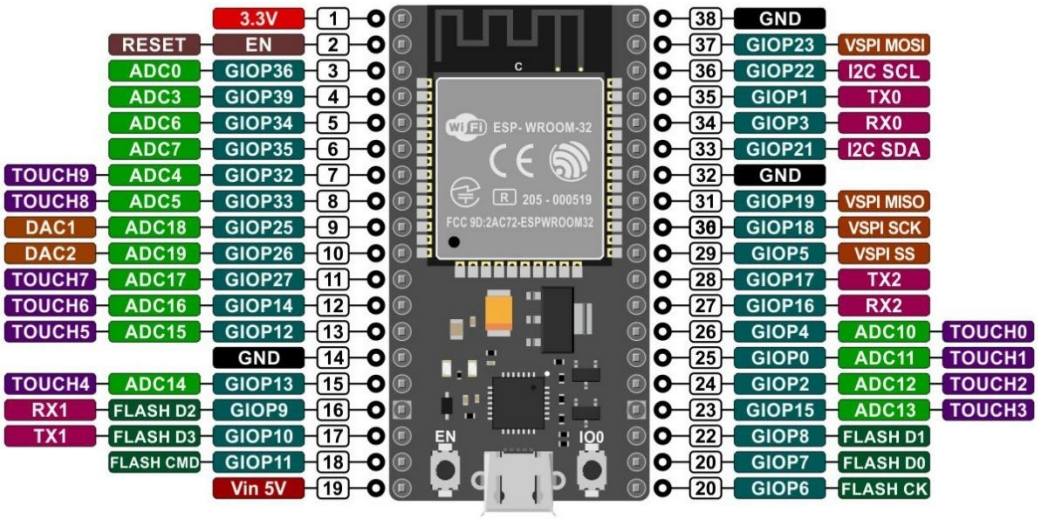


Figura A1. Pines de la Esp32s

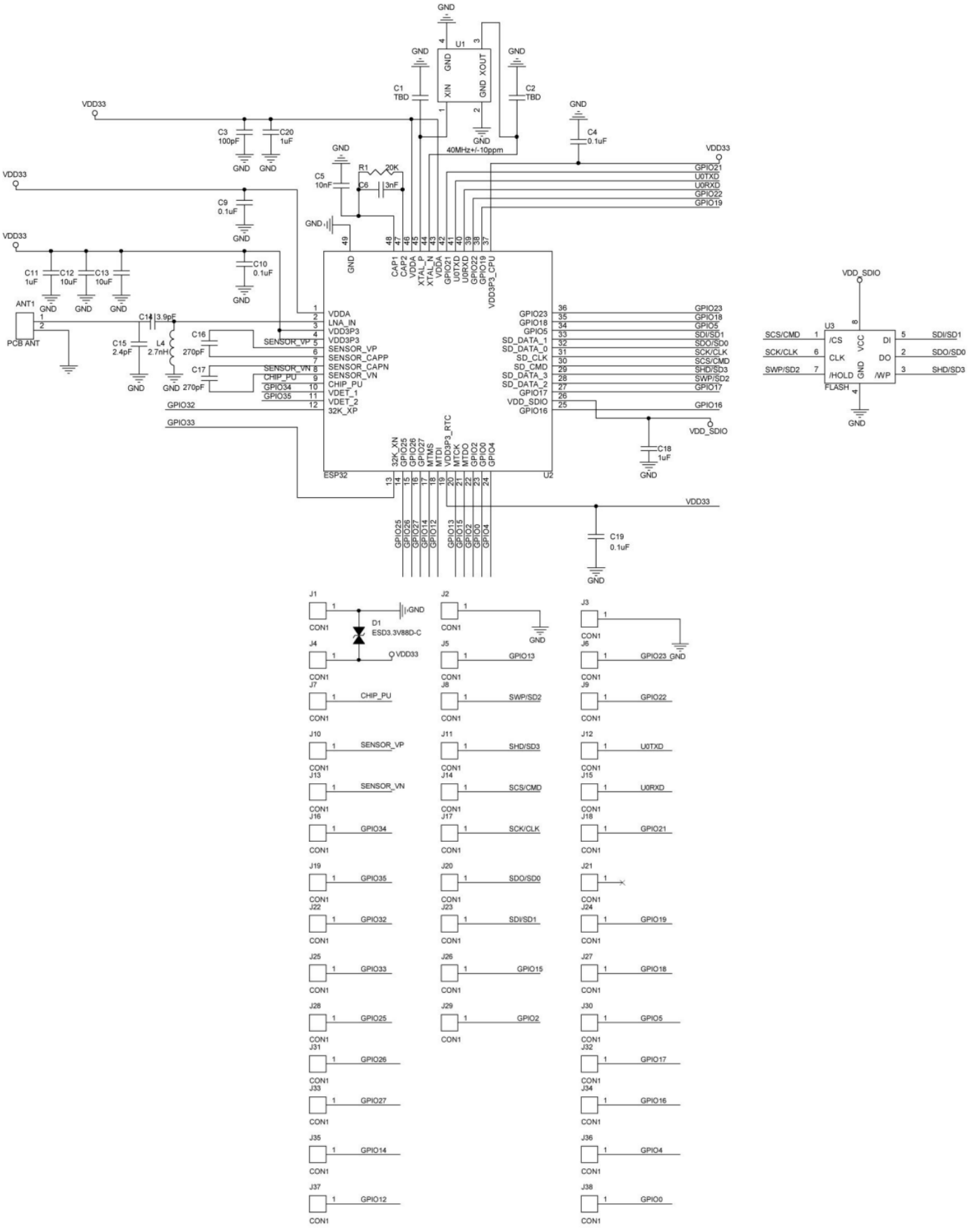


Figura A2. Conexiones internas de la Esp32s

ANALOG DEVICES

Click here to ask an associate for production status of specific part numbers.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

MAX9814

General Description

The MAX9814 is a low-cost, high-quality microphone amplifier with automatic gain control (AGC) and low-noise microphone bias. The device features a low-noise preamplifier, variable gain amplifier (VGA), output amplifier, microphone-bias-voltage generator, and AGC control circuitry.

The low-noise preamplifier has a fixed 12dB gain, while the VGA gain automatically adjusts from 20dB to 0dB, depending on the output voltage and the AGC threshold. The output amplifier offers selectable gains of 8dB, 18dB, and 28dB. With no compression, the cascade of the amplifiers results in an overall gain of 40dB, 50dB, or 60dB. A trilevel digital input programs the output amplifier gain. An external resistive divider controls the AGC threshold and a single capacitor programs the attack/release times. A trilevel digital input programs the ratio of attack-to-release time. The hold time of the AGC is fixed at 30ms. The low-noise microphone-bias-voltage generator can bias most electret microphones.

The MAX9814 is available in the space-saving, 14-pin TDFN package. This device is specified over the -40°C to +85°C extended temperature range.

Applications

- Digital Still Cameras
- Digital Video Cameras
- PDAs
- Bluetooth Headsets
- Entertainment Systems (e.g., Karaoke)
- Two-Way Communicators
- High-Quality Portable Recorders
- IP Phones/Telephone Conferencing

Features

- Automatic Gain Control (AGC)
- Three Gain Settings (40dB, 50dB, 60dB)
- Programmable Attack Time
- Programmable Attack and Release Ratio
- 2.7V to 5.5V Supply Voltage Range
- Low Input-Referred Noise Density of 30nV/√Hz
- Low THD: 0.04% (typ)
- Low-Power Shutdown Mode
- Internal Low-Noise Microphone Bias, 2V
- Available in the Space-Saving, 14-Pin TDFN (3mm x 3mm) Package
- -40°C to +85°C Extended Temperature Range

Ordering Information ***appears at end of data sheet.***

Simplified Block Diagram

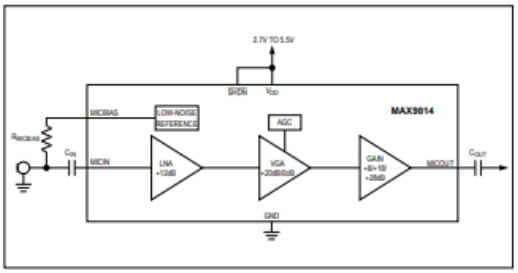


Figura B1. Descripción general del Max9814

Absolute Maximum Ratings

V_{DD} to GND ..... -0.3V to +6V
All Other Pins to GND ..... -0.3V to (V_{DD} + 0.3V)
Output Short-Circuit Duration ..... Continuous
Continuous Current (MICOUT, MICBIAS) ..... ±100mA
All Other Pins ..... ±20mA

Continuous Power Dissipation (T_A = +70°C)
14-Pin TDFN-EP
(derate 16.7mW/°C above +70°C) 1481.5mW
Operating Temperature Range ..... -40°C to +85°C
Junction Temperature ..... +150°C
Lead Temperature (soldering, 10s) ..... +300°C
Bump Temperature (soldering) Reflow ..... +235°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Electrical Characteristics

(V_{DD} = 3.3V, $\overline{SHDN}$ = V_{DD}, C_{CT} = 470nF, C_{CG} = 2µF, GAIN = V_{DD}, T_A = T_{MIN} to T_{MAX}, unless otherwise specified. Typical values are at T_A = +25°C.) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
GENERAL						
Operating Voltage	V_{DD}	Guaranteed by PSRR test	2.7		5.5	V
Supply Current	I_{DD}			3.1	6	mA
Shutdown Supply Current	$I_{\overline{SHDN}}$			0.01	1	µA
Input-Referred Noise Density	e_n	BW = 20kHz, all gain settings		30		nV/√Hz
Output Noise		BW = 20kHz		430		μV_{RMS}
Signal-to-Noise Ratio	SNR	BW = 22Hz to 22kHz ($500mV_{RMS}$ output signal)		61		dB
		A-weighted		64		
Dynamic Range	DR	(Note 2)		60		dB
Total Harmonic Distortion Plus Noise	THD+N	f_{IN} = 1kHz, BW = 20Hz to 20kHz, R_L = 10kΩ, V_{TH} = 1V (threshold = $2V_{P-P}$), V_{IN} = $0.5mV_{RMS}$, V_{CT} = 0V		0.04		%
		f_{IN} = 1kHz, BW = 20Hz to 20kHz, R_L = 10kΩ, V_{TH} = 0.1V (threshold = $200mV_{P-P}$), V_{IN} = $30mV_{RMS}$, V_{CT} = 2V		0.2		
Amplifier Input BIAS	V_{IN}		1.14	1.23	1.32	V
Maximum Input Voltage	V_{IN_MAX}	1% THD		100		mV_{P-P}
Input Impedance	Z_{IN}			100		kΩ
Maximum Gain	A	GAIN = V_{DD}	39.5	40	40.5	dB
		GAIN = GND	49.5	50	50.6	
		GAIN = unconnected	59.5	60	60.5	
Minimum Gain		GAIN = V_{DD}	18.7	20	20.5	dB
		GAIN = GND	29.0	30	30.8	
		GAIN = unconnected	38.7	40	40.5	
Maximum Output Level	V_{OUT_RMS}	1% THD+N, V_{TH} = MICBIAS		0.707		V_{RMS}
Regulated Output Level		AGC enabled, V_{TH} = 0.7V	1.26	1.40	1.54	V_{P-P}
AGC Attack Time	t_{ATTACK}	C_{CT} = 470nF (Note 3)		1.1		ms
Attack/Release Ratio	A/R	A/R = GND		1:500		ms/ms
		A/R = V_{DD}		1:2000		
		A/R = unconnected		1:4000		

Figura B2. Características eléctricas del sensor Max9814

Electrical Characteristics (continued)

(V_{DD} = 3.3V, $\overline{SHDN}$ = V_{DD}, C_{CT} = 470nF, C_{CG} = 2µF, GAIN = V_{DD}, T_A = T_{MIN} to T_{MAX}, unless otherwise specified. Typical values are at T_A = +25°C.) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
MICOUT High Output Voltage	V_{OH}	I_{OUT} sourcing 1mA		2.45		V
MICOUT Low Output Voltage	V_{OL}	I_{OUT} sinking 1mA		3		mV
MICOUT Bias		MICOUT unconnected	1.14	1.23	1.32	V
Output Impedance	Z_{OUT}			50		Ω
Minimum Resistive Load	R_{LOAD_MIN}			5		kΩ
Maximum Capacitive Drive	C_{LOAD_MAX}			200		pF
Maximum Output Current	I_{OUT_MAX}	1% THD, R_L = 500Ω		1	2	mA
Output Short-Circuit Current	I_{SC}		3	8		mA
Power-Supply Rejection Ratio	PSRR	AGC mode; V_{DD} = 2.7V to 5.5V (Note 4)	35	50		dB
		f = 217Hz, V_{RIPPLE} = 100mV_{P-P} (Note 5)		55		
		f = 1kHz, V_{RIPPLE} = 100mV_{P-P} (Note 5)		52.5		
		f = 10kHz, V_{RIPPLE} = 100mV_{P-P} (Note 5)		43		
MICROPHONE BIAS						
Microphone Bias Voltage	$V_{MICBIAS}$	$I_{MICBIAS}$ = 0.5mA	1.84	2.0	2.18	V
Output Resistance	$R_{MICBIAS}$	$I_{MICBIAS}$ = 1mA		1		Ω
Output Noise Voltage	$V_{MICBIAS_NOISE}$	$I_{MICBIAS}$ = 0.5mA, BW = 22Hz to 22kHz		5.5		μV_{RMS}
Power-Supply Rejection Ratio	PSRR	DC, V_{DD} = 2.7V to 5.5V	70	80		dB
		$I_{MICBIAS}$ = 0.5mA, V_{RIPPLE} = 100mV_{P-P}, f_{IN} = 1kHz		71		
TRILEVEL INPUTS (A/R, GAIN)						
Trilevel Input Leakage Current		A/R or GAIN = V_{DD}	$0.5V_{DD}$ / 180kΩ	$0.5V_{DD}$ / 100kΩ	$0.5V_{DD}$ / 50kΩ	mA
		A/R or GAIN = GND	$0.5V_{DD}$ / 180kΩ	$0.5V_{DD}$ / 100kΩ	$0.5V_{DD}$ / 50kΩ	
Input High Voltage	V_{IH}		V_{DD} x 0.7			V
Input Low Voltage	V_{IL}				V_{DD} x 0.3	V
Shutdown Enable Time	t_{ON}			60		ms
Shutdown Disable Time	t_{OFF}			40		ms
DIGITAL INPUT ($\overline{SHDN}$)						
$\overline{SHDN}$ Input Leakage Current			-1		+1	µA
Input High Voltage	V_{IH}		1.3			V
Input Low Voltage	V_{IL}				0.5	V
AGC THRESHOLD INPUT (TH)						
TH Input Leakage Current			-1		+1	µA

Note 1: Devices are production tested at T_A = +25°C. Limits over temperature are guaranteed by design.
Note 2: Dynamic range is calculated using the EIAJ method. The input is applied at -60dBFS (0.707µV_{RMS}), f_{IN} = 1kHz.
Note 3: Attack time measured as time from AGC trigger to gain reaching 90% of its final value.
Note 4: CG is connected to an external DC voltage source, and adjusted until V_{MICOUT} = 1.23V.
Note 5: CG connected to GND with 2.2µF.

Figura B3. Características eléctricas del sensor Max9814

Typical Operating Characteristics

(V_{DD} = 5V, C_{CT} = 470nF, C_{CG} = 2.2µF, V_{TH} = $V_{MICBIAS}$ x 0.4, GAIN = V_{DD} (40dB), AGC disabled, no load, R_L = 10kΩ, C_{OUT} = 1µF, T_A = +25°C, unless otherwise noted.)

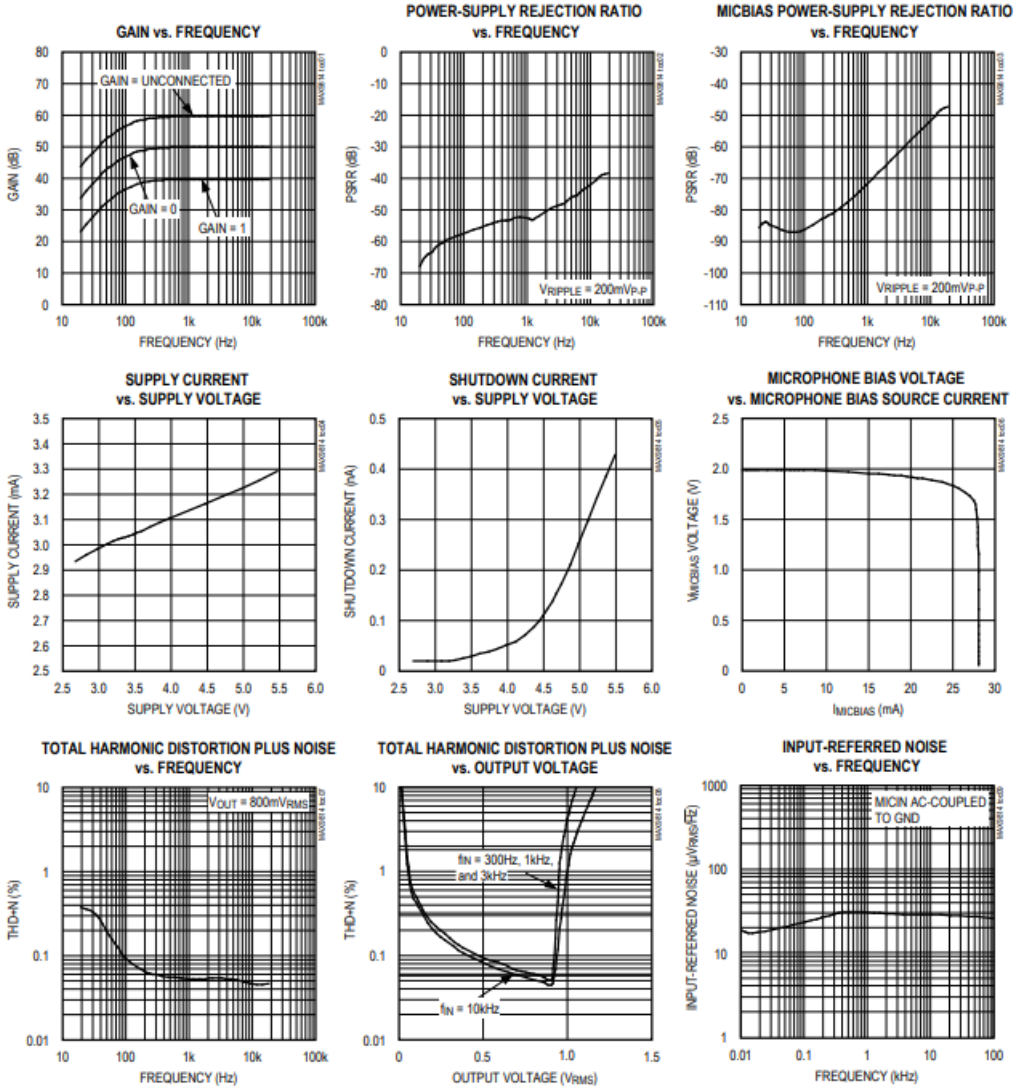


Figura B4. Características típicas de operación del sensor Max9814

Anexo C. Datasheet del módulo PCM1808

PCM1808 Single-Ended, Analog-Input 24-Bit, 96-kHz Stereo ADC

1 Features

- 24-Bit Delta-Sigma Stereo A/D Converter (ADC)
- Single-Ended Voltage Input: 3 Vp-p
- High Performance:
  - THD+N: –93 dB (Typical)
  - SNR: 99 dB (Typical)
  - Dynamic Range: 99 dB (Typical)
- Oversampling Decimation Filter:
  - Oversampling Frequency: ×64
  - Pass-Band Ripple: ±0.05 dB
  - Stop-Band Attenuation: –65 dB
  - On-Chip High-Pass Filter: 0.91 Hz (48 kHz)
- Flexible PCM Audio Interface
  - Master- or Slave-Mode Selectable
  - Data Formats: 24-Bit I^2S, 24-Bit Left-Justified
- Power Down and Reset by Halting System Clock
- Analog Antialias LPF Included
- Sampling Rate: 8 kHz–96 kHz
- System Clock: 256 f_S, 384 f_S, 512 f_S
- Resolution: 24 Bits
- Dual Power Supplies:
  - 5-V for Analog
  - 3.3-V for Digital
- Package: 14-Pin TSSOP

2 Applications

- DVD Recorder
- Digital TV
- AV Amplifier or Receiver
- MD Player
- CD Recorder
- Multitrack Receiver
- Electric Musical Instrument

3 Description

The PCM1808 device is a high-performance, low-cost, single-chip, stereo analog-to-digital converter with single-ended analog voltage input. The PCM1808 device uses a delta-sigma modulator with 64-times oversampling and includes a digital decimation filter and high-pass filter that removes the dc component of the input signal. For various applications, the PCM1808 device supports master and slave mode and two data formats in serial audio interface.

The PCM1808 device supports the power-down and reset functions by means of halting the system clock.

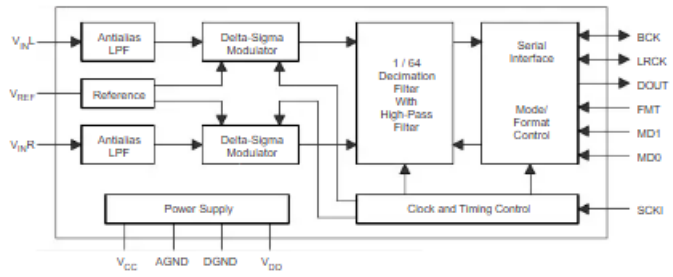
The PCM1808 device is suitable for wide variety of cost-sensitive consumer applications requiring good performance and operation with a 5-V analog supply and 3.3-V digital supply. Fabrication of the PCM1808 device uses a highly advanced CMOS process. The device is available in a small, 14-pin TSSOP package.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
PCM1808	TSSOP (14)	4.40 mm × 5.00 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

PCM1808 Block Diagram



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

Figura C1. Descripción general del módulo Pcm1808

6 Specifications

6.1 Absolute Maximum Ratings

over operating ambient temperature range (unless otherwise noted) (1)

			MIN	MAX	UNIT
V_{CC}	Analog supply voltage		–0.3	6.5	V
V_{DD}	Digital supply voltage		–0.3	4	V
	Ground voltage differences	AGND, DGND		±0.1	V
	Digital input voltage	LRCK, BCK, DOUT	–0.3	(V_{DD} + 0.3 V) < 4	V
		SCKI, MD0, MD1, FMT	–0.3	6.5	V
$V_{IN}L$, $V_{IN}R$, V_{REF}	Analog input voltage		–0.3	(V_{CC} + 0.3 V) < 6.5	V
	Input current (any pins except supplies)			±10	mA
T_J	Junction temperature			150	°C
T_{stg}	Storage temperature		–55	150	°C

(1) Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under Recommended Operating Conditions. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001, all pins (1)	±4000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101, all pins (2)	±1500	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating ambient temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
V_{CC}	Analog supply voltage (see Power Supply Recommendations)		4.5	5	5.5	V
V_{DD}	Digital supply voltage		2.7	3.3	3.6	V
	Analog input voltage, full scale (–0 dB)	V_{CC} = 5 V			3	Vp-p
V_{IH} (1)	High input logic level		2		V_{DD}	VDC
V_{IL} (1)	Low input logic level		0		0.8	VDC
V_{IH} (2) (3)	High input logic level		2		5.5	VDC
V_{IL} (2) (3)	Low input logic level		0		0.8	VDC
	Digital input logic family		TTL compatible			
	Digital input clock frequency, system clock		2.048		49.152	MHz
	Digital input clock frequency, sampling clock		8		96	kHz
	Digital output load capacitance				20	pF
T_A	Operating ambient temperature range		–40		85	°C
T_J	Junction temperature				150	°C

(1) Pins 7, 8: LRCK, BCK (Schmitt-trigger input, with 50-kΩ typical pulldown resistor, in slave mode)
(2) Pin 6: SCKI (Schmitt-trigger input, 5-V tolerant)
(3) Pins 10–12: MD0, MD1, FMT (Schmitt-trigger input, with 50-kΩ typical pulldown resistor, 5-V tolerant)

Figura C2. Especificaciones técnicas del módulo Pcm1808

6.7 Typical Characteristics

All specifications at T_A = 25°C, V_{CC} = 5 V, V_{DD} = 3.3 V, master mode, f_S = 48 kHz, system clock = 512 f_S, 24-bit data, unless otherwise noted.

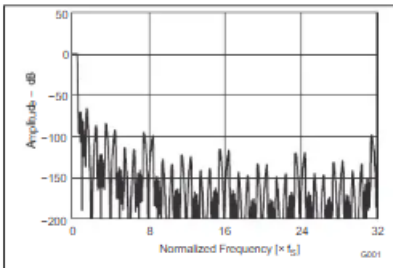


Figure 7. Decimation-Filter Frequency Response Overall Characteristics

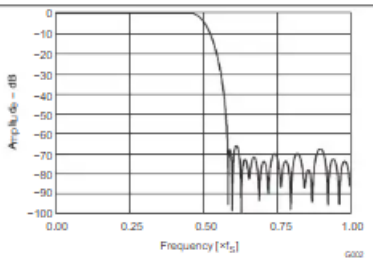


Figure 8. Decimation-Filter Frequency Response Stop-Band Attenuation Characteristics

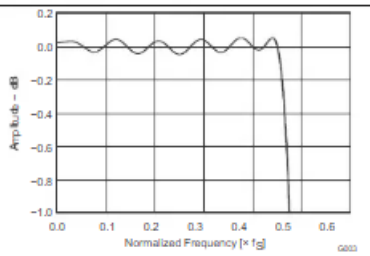


Figure 9. Decimation-Filter Frequency Response Pass-Band Ripple Characteristics

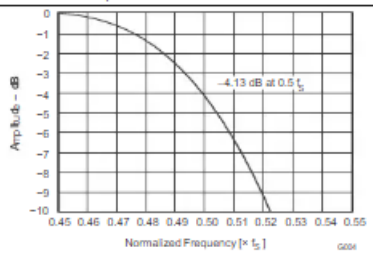


Figure 10. Decimation-Filter Frequency Response Transition-Band Characteristics

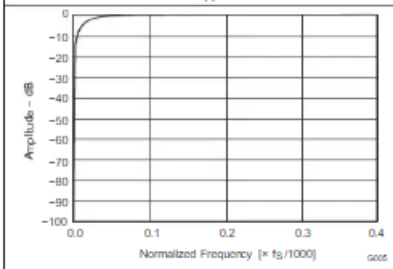


Figure 11. High-Pass Filter Frequency Response HPF Stop-Band Characteristics

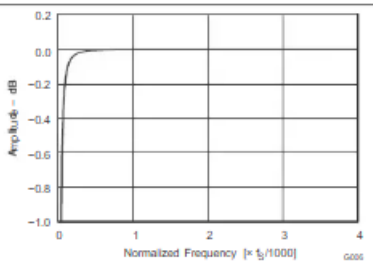
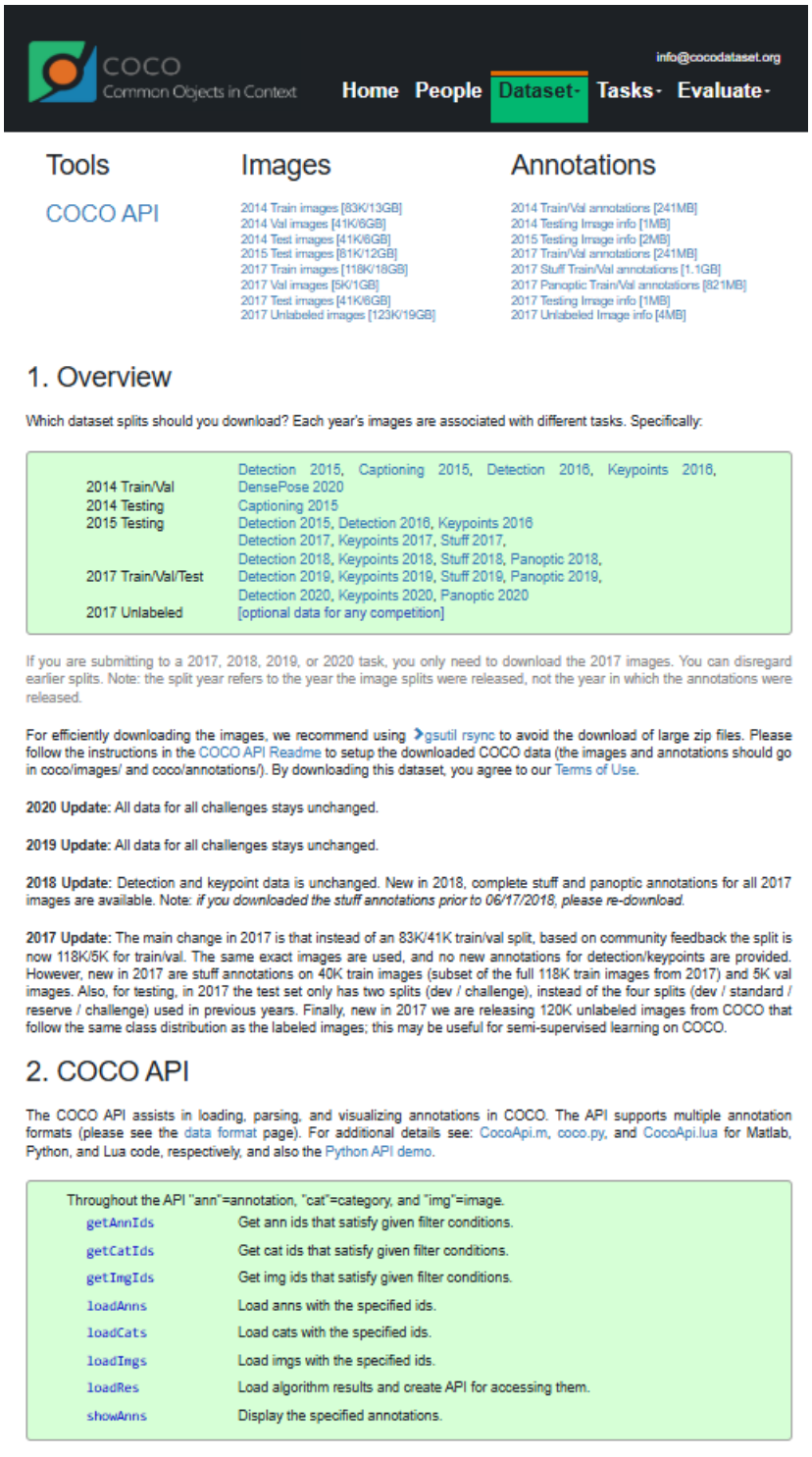


Figure 12. High-Pass Filter Frequency Response HPF Stop-Band Characteristics

Copyright © 2006–2015, Texas Instruments Incorporated

Figura C3. Características típicas de operación del módulo Pcm1808



COCO Common Objects in Context

info@cocodataset.org

Home People Dataset- Tasks- Evaluate-

Tools

COCO API

Images

2014 Train images [83K/13GB]
2014 Val images [41K/6GB]
2014 Test images [41K/6GB]
2015 Test images [81K/12GB]
2017 Train images [118K/18GB]
2017 Val images [5K/1GB]
2017 Test images [41K/6GB]
2017 Unlabeled images [123K/19GB]

Annotations

2014 Train/Val annotations [241MB]
2014 Testing Image info [1MB]
2015 Testing Image info [2MB]
2017 Train/Val annotations [241MB]
2017 Stuff Train/Val annotations [1.1GB]
2017 Panoptic Train/Val annotations [821MB]
2017 Testing Image info [1MB]
2017 Unlabeled Image info [4MB]

1. Overview

Which dataset splits should you download? Each year's images are associated with different tasks. Specifically:

2014 Train/Val	Detection 2015, Captioning 2015, Detection 2016, Keypoints 2016, DensePose 2020
2014 Testing	Captioning 2015
2015 Testing	Detection 2015, Detection 2016, Keypoints 2016
2017 Train/Val/Test	Detection 2017, Keypoints 2017, Stuff 2017, Detection 2018, Keypoints 2018, Stuff 2018, Panoptic 2018, Detection 2019, Keypoints 2019, Stuff 2019, Panoptic 2019, Detection 2020, Keypoints 2020, Panoptic 2020
2017 Unlabeled	[optional data for any competition]

If you are submitting to a 2017, 2018, 2019, or 2020 task, you only need to download the 2017 images. You can disregard earlier splits. Note: the split year refers to the year the image splits were released, not the year in which the annotations were released.

For efficiently downloading the images, we recommend using `gsutil rsync` to avoid the download of large zip files. Please follow the instructions in the COCO API Readme to setup the downloaded COCO data (the images and annotations should go in coco/images/ and coco/annotations/). By downloading this dataset, you agree to our Terms of Use.

2020 Update: All data for all challenges stays unchanged.

2019 Update: All data for all challenges stays unchanged.

2018 Update: Detection and keypoint data is unchanged. New in 2018, complete stuff and panoptic annotations for all 2017 images are available. Note: *if you downloaded the stuff annotations prior to 06/17/2018, please re-download.*

2017 Update: The main change in 2017 is that instead of an 83K/41K train/val split, based on community feedback the split is now 118K/5K for train/val. The same exact images are used, and no new annotations for detection/keypoints are provided. However, new in 2017 are stuff annotations on 40K train images (subset of the full 118K train images from 2017) and 5K val images. Also, for testing, in 2017 the test set only has two splits (dev / challenge), instead of the four splits (dev / standard / reserve / challenge) used in previous years. Finally, new in 2017 we are releasing 120K unlabeled images from COCO that follow the same class distribution as the labeled images; this may be useful for semi-supervised learning on COCO.

2. COCO API

The COCO API assists in loading, parsing, and visualizing annotations in COCO. The API supports multiple annotation formats (please see the data format page). For additional details see: CocoApi.m, coco.py, and CocoApi.lua for Matlab, Python, and Lua code, respectively, and also the Python API demo.

Throughout the API "ann"=annotation, "cat"=category, and "img"=image.

`getAnnIds`	Get ann ids that satisfy given filter conditions.
`getCatIds`	Get cat ids that satisfy given filter conditions.
`getImgIds`	Get img ids that satisfy given filter conditions.
`loadAnns`	Load anns with the specified ids.
`loadCats`	Load cats with the specified ids.
`loadImgs`	Load imgs with the specified ids.
`loadRes`	Load algorithm results and create API for accessing them.
`showAnns`	Display the specified annotations.

Figura E1. Pagina de COCO Dataset

Anexo F. Código en Python para detección de vehículos con Yolov8x

```
import os
import cv2
import time
import json
from ultralytics import YOLO
import torch
import numpy as np
from sort import Sort

# Configuración de archivos
video_path = "C:/Users/USER/Desktop/Deteccion de
Vehiculos/YOLOv8_Data/video_1.mp4"
output_dir = "C:/Users/USER/Desktop/Deteccion de
Vehiculos/Yolov8_Resultados/"
output_video_path = os.path.join(output_dir,
"output_video_1_resultados_mejorado_final.mp4")
output_json = os.path.join(output_dir, "output_video_1_detections.json")
output_txt = os.path.join(output_dir, "output_video_1_vehicle_count.txt")

os.makedirs(output_dir, exist_ok=True)

# Configuración del dispositivo
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Dispositivo utilizado: {device}")

# Cargar el modelo YOLOv8
model = YOLO("yolov8x.pt")
model.to(device)

# Configuración de video
cap = cv2.VideoCapture(video_path)
fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

fourcc = cv2.VideoWriter_fourcc(*"mp4v")
out = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width,
frame_height))

# Inicializar variables
vehicle_count = 0
detections_list = []
tracker = Sort(max_age=30, min_hits=5, iou_threshold=0.3)  # Parámetros
ajustados
```

```
counted_ids = set()

# Región de Interés (ROI)
ROI = (0, 0, frame_width, frame_height)  # Toda la carretera
MIN_AREA = 1200
MAX_AREA = 100000

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    start_time = time.time()

    # Preprocesar imagen
    frame = cv2.convertScaleAbs(frame, alpha=1.5, beta=-30)

    # Detección con YOLO
    results = model.predict(frame, imgsz=1920, conf=0.35)  # Confianza más baja
para capturar más vehículos
    detections = results[0].boxes.xyxy.cpu().numpy()
    confidences = results[0].boxes.conf.cpu().numpy()
    classes = results[0].boxes.cls.cpu().numpy()

    # Filtrar vehículos
    vehicle_detections = [
        [x1, y1, x2, y2, conf]
        for (x1, y1, x2, y2), conf, cls in zip(detections, confidences, classes)
        if cls in [2, 5, 7] and conf > 0.35
    ]

    # Filtrar ROI
    vehicle_detections = [
        det for det in vehicle_detections
        if ROI[0] <= det[0] <= ROI[2] and ROI[1] <= det[1] <= ROI[3]
    ]

    # Filtrar por tamaño del BBox
    vehicle_detections = [
        det for det in vehicle_detections
        if MIN_AREA < (det[2] - det[0]) * (det[3] - det[1]) < MAX_AREA
    ]

    # Actualizar rastreador
    if len(vehicle_detections) > 0:
        tracked_objects = tracker.update(np.array(vehicle_detections))
    else:
        tracked_objects = np.empty((0, 5))
```

```
    for track in tracked_objects:
        x1, y1, x2, y2, track_id = track.astype(int)

        # Ignorar objetos estáticos
        distance = np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
        if distance < 10:  # Ajustar umbral de movimiento
            continue

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(
            frame,
            f"ID: {track_id}",
            (x1, y1 - 10),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.5,
            (255, 0, 0),
            2,
        )

        if track_id not in counted_ids:
            vehicle_count += 1
            counted_ids.add(track_id)
            detections_list.append({
                "id": int(track_id),
                "frame": int(cap.get(cv2.CAP_PROP_POS_FRAMES)),
                "bbox": [int(x1), int(y1), int(x2 - x1), int(y2 - y1)],
            })

    cv2.putText(
        frame,
        f"Conteo de vehiculos: {vehicle_count}",
        (20, 50),
        cv2.FONT_HERSHEY_SIMPLEX,
        1.5,
        (0, 255, 255),
        3,
    )

    out.write(frame)
    end_time = time.time()
    print(f"Frame procesado en {end_time - start_time:.2f} segundos")

cap.release()
out.release()

with open(output_json, "w") as f:
    json.dump(detections_list, f, indent=4)
```

```
with open(output_txt, "w") as f:
    f.write(f"Conteo total de vehiculos: {vehicle_count}")

print(f"Video procesado guardado en: {output_video_path}")
```

Anexo G. Código en Arduino IDE para la grabación de audio

```
#include <driver/i2s.h>
#define I2S_NUM         I2S_NUM_0
#define I2S_SAMPLE_RATE 16000
#define I2S_SAMPLE_BITS 16
#define I2S_CHANNEL_FMT I2S_CHANNEL_FMT_ONLY_LEFT
#define I2S_BCK_PIN  19
#define I2S_WS_PIN   21
#define I2S_DIN_PIN  18
#define BUFFER_SIZE 1024
void setup() {
  Serial.begin(115200);
  configurarI2S();
}
void loop() {
  uint8_t buffer[BUFFER_SIZE];
  size_t bytesRead;
  // Leer datos de audio del PCM1808
  i2s_read(I2S_NUM, buffer, BUFFER_SIZE, &bytesRead, portMAX_DELAY);
  // Imprimir los datos para verificar (opcional)
  Serial.print("Bytes leídos: ");
  Serial.println(bytesRead);
  for (size_t i = 0; i < bytesRead; i++) {
    Serial.print(buffer[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
}
void configurarI2S() {
  i2s_config_t i2s_config = {
      .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
      .sample_rate = I2S_SAMPLE_RATE,
      .bits_per_sample = I2S_SAMPLE_BITS,
      .channel_format = I2S_CHANNEL_FMT,
      .communication_format = I2S_COMM_FORMAT_I2S,
      .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
      .dma_buf_count = 2,
```

```
    .dma_buf_len = BUFFER_SIZE,
    .use_apll = false
  };
  i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_BCK_PIN,
    .ws_io_num = I2S_WS_PIN,
    .data_out_num = I2S_PIN_NO_CHANGE,
    .data_in_num = I2S_DIN_PIN
  };
  // Inicializar el driver I2S
  i2s_driver_install(I2S_NUM, &i2s_config, 0, NULL);
  i2s_set_pin(I2S_NUM, &pin_config);
}
```

Anexo H. Código para procesamiento de video y audio hacia la base de datos

```
import os
import numpy as np
import mysql.connector
import time
from datetime import datetime
from ultralytics import YOLO
from pydub import AudioSegment
from moviepy.editor import VideoFileClip
import cv2
from deep_sort_realtime.deepsort_tracker import DeepSort

# Configuración de Pydub con FFmpeg
AudioSegment.converter = r"C:\Users\USER\Downloads\ffmpeg-7.1-
essentials_build\ffmpeg-7.1-essentials_build\bin\ffmpeg.exe"
AudioSegment.ffprobe = r"C:\Users\USER\Downloads\ffmpeg-7.1-
essentials_build\ffmpeg-7.1-essentials_build\bin\ffprobe.exe"

# Parámetros de conexión a la base de datos
db_host = "10.79.7.59"
db_user = "jarod"
db_password = "Junior123#"
db_name = "trafico_db"

# Función para conectar con la base de datos
def conectar_base_datos():
    try:
        conexion = mysql.connector.connect(
            host=db_host,
```

```
            user=db_user,
            password=db_password,
            database=db_name
        )
        return conexion
    except Exception as e:
        print(f"Error al conectar con la base de datos: {e}")
        return None

# Función para calcular el nivel de ruido en un intervalo
def calcular_ruido(video_path, start_time, end_time):
    try:
        clip = VideoFileClip(video_path).subclip(start_time, end_time)
        audio = clip.audio
        audio_path = "temp_audio.wav"
        audio.write_audiofile(audio_path, codec='pcm_s16le', fps=44100,
verbose=False, logger=None)

        # Procesar el audio con Pydub
        sound = AudioSegment.from_wav(audio_path)
        rms = sound.rms
        ruido_db = 20 * np.log10(rms) if rms > 0 else 0
        os.remove(audio_path)
        return round(ruido_db, 2)
    except Exception as e:
        print(f"Error al calcular ruido: {e}")
        return 0

# Función principal para procesar el video
def procesar_video(video_path):
    print("Iniciando procesamiento del video...")

    # Configuración del modelo YOLO y DeepSORT
    model = YOLO("yolov8x.pt")
    tracker = DeepSort(max_age=30, n_init=3)

    # Captura de video
    cap = cv2.VideoCapture(video_path)
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    duracion = int(total_frames / fps)
    print(f"FPS: {fps}, Duración: {duracion}s")

    # Crear intervalos dinámicos
    intervalo_segundos = 30
    intervalos = [(i, min(i + intervalo_segundos, duracion)) for i in range(0,
duracion, intervalo_segundos)]
```

```
    for start, end in intervalos:
        cap.set(cv2.CAP_PROP_POS_FRAMES, start * fps)
        vehiculos_detectados = set()
        ruido_db = 0
        for _ in range((end - start) * fps):
            ret, frame = cap.read()
            if not ret:
                break

            # Detección de objetos con YOLO
            resultados = model(frame)
            boxes = []
            for result in resultados[0].boxes.data.tolist():
                x1, y1, x2, y2, score, class_id = result
                if int(class_id) == 2 or int(class_id) == 7:  # Autos y camiones
                    boxes.append(([x1, y1, x2, y2], score, class_id))

            # Actualizar rastreador
            tracks = tracker.update_tracks(boxes, frame=frame)
            for track in tracks:
                if track.is_confirmed():
                    vehiculos_detectados.add(track.track_id)

        # Calcular ruido
        ruido_db = calcular_ruido(video_path, start, end)
        nivel_ruido = determinar_nivel_ruido(ruido_db)

        # Guardar datos en la base de datos
        conexion = conectar_base_datos()
        if conexion:
            try:
                cursor = conexion.cursor()
                query = "INSERT INTO datos_trafico (vehiculos, ruido_db, nivel_ruido,
fecha) VALUES (%s, %s, %s, %s)"
                valores = (len(vehiculos_detectados), ruido_db, nivel_ruido,
datetime.now())
                cursor.execute(query, valores)
                conexion.commit()
                print(f"Enviado: Vehículos={len(vehiculos_detectados)},
Ruido={ruido_db} dB, Nivel={nivel_ruido}")
            except Exception as e:
                print(f"Error al enviar datos a la base: {e}")
            finally:
                cursor.close()
                conexion.close()

        print(f"Intervalo {start}-{end}s: Vehículos={len(vehiculos_detectados)},
Ruido={ruido_db} dB, Nivel={nivel_ruido}")
```

```
    cap.release()
    print("Procesamiento finalizado.")

# RESNET

# Función para determinar el nivel de ruido
def determinar_nivel_ruido(ruido_db):
    if ruido_db < 50:
        return "Bajo"
    elif 50 <= ruido_db < 70:
        return "Medio"
    else:
        return "Alto"

# Función principal
def main():
    video_path = "C:/Users/USER/Desktop/Deteccion de
Vehiculos/YOLOv8_Data/video_2.mp4"
    procesar_video(video_path)

if __name__ == "__main__":
    main()
```

Anexo I. Datos recolectados en la base de datos

Resumen de Datos: 2024-12-09

Tabla I1. Datos más relevantes en el día 2 analizados

Hora	Vehículos	Ruido (dB)	Nivel de Ruido
06:45 - 07:00	28 - 45	59 - 90	Alto
12:50 - 12:59	42 - 45	62 - 90	Alto
17:30 - 17:45	42 - 45	71 - 90	Alto



			id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar	Copiar	Borrar	226	Avenida Los Chasquis	28	67.86	Medio	2024-12-09 06:45:00
Editar	Copiar	Borrar	227	Avenida Los Chasquis	33	59.79	Medio	2024-12-09 06:46:00
Editar	Copiar	Borrar	228	Avenida Los Chasquis	38	68.96	Medio	2024-12-09 06:47:00
Editar	Copiar	Borrar	229	Avenida Los Chasquis	41	64.83	Medio	2024-12-09 06:48:00
Editar	Copiar	Borrar	230	Avenida Los Chasquis	45	90	Alto	2024-12-09 06:49:00
Editar	Copiar	Borrar	231	Avenida Los Chasquis	45	90	Alto	2024-12-09 06:50:00
Editar	Copiar	Borrar	232	Avenida Los Chasquis	45	90	Alto	2024-12-09 06:51:00
Editar	Copiar	Borrar	233	Avenida Los Chasquis	45	73.99	Alto	2024-12-09 06:52:00
Editar	Copiar	Borrar	234	Avenida Los Chasquis	45	71.54	Alto	2024-12-09 06:53:00
Editar	Copiar	Borrar	235	Avenida Los Chasquis	43	90	Alto	2024-12-09 06:54:00
Editar	Copiar	Borrar	236	Avenida Los Chasquis	45	86.18	Alto	2024-12-09 06:55:00
Editar	Copiar	Borrar	237	Avenida Los Chasquis	43	90	Alto	2024-12-09 06:56:00
Editar	Copiar	Borrar	238	Avenida Los Chasquis	43	90	Alto	2024-12-09 06:57:00
Editar	Copiar	Borrar	239	Avenida Los Chasquis	43	90	Alto	2024-12-09 06:58:00
Editar	Copiar	Borrar	240	Avenida Los Chasquis	42	80.91	Alto	2024-12-09 06:59:00

Figura I1. Datos recolectados en la mañana



			id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar	Copiar	Borrar	291	Avenida Los Chasquis	45	78.35	Alto	2024-12-09 12:50:00
Editar	Copiar	Borrar	292	Avenida Los Chasquis	45	70.86	Alto	2024-12-09 12:51:00
Editar	Copiar	Borrar	293	Avenida Los Chasquis	45	90	Alto	2024-12-09 12:52:00
Editar	Copiar	Borrar	294	Avenida Los Chasquis	43	62.76	Medio	2024-12-09 12:53:00
Editar	Copiar	Borrar	295	Avenida Los Chasquis	45	90	Alto	2024-12-09 12:54:00
Editar	Copiar	Borrar	296	Avenida Los Chasquis	44	90	Alto	2024-12-09 12:55:00
Editar	Copiar	Borrar	297	Avenida Los Chasquis	45	90	Alto	2024-12-09 12:56:00
Editar	Copiar	Borrar	298	Avenida Los Chasquis	45	90	Alto	2024-12-09 12:57:00
Editar	Copiar	Borrar	299	Avenida Los Chasquis	43	90	Alto	2024-12-09 12:58:00
Editar	Copiar	Borrar	300	Avenida Los Chasquis	42	68.22	Medio	2024-12-09 12:59:00

Figura I2. Datos recolectados al medio día



			id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar	Copiar	Borrar	331	Avenida Los Chasquis	45	90	Alto	2024-12-09 17:30:00
Editar	Copiar	Borrar	332	Avenida Los Chasquis	44	90	Alto	2024-12-09 17:31:00
Editar	Copiar	Borrar	333	Avenida Los Chasquis	45	81.55	Alto	2024-12-09 17:32:00
Editar	Copiar	Borrar	334	Avenida Los Chasquis	45	71.08	Alto	2024-12-09 17:33:00
Editar	Copiar	Borrar	335	Avenida Los Chasquis	45	72.11	Alto	2024-12-09 17:34:00
Editar	Copiar	Borrar	336	Avenida Los Chasquis	45	87.98	Alto	2024-12-09 17:35:00
Editar	Copiar	Borrar	337	Avenida Los Chasquis	43	90	Alto	2024-12-09 17:36:00
Editar	Copiar	Borrar	338	Avenida Los Chasquis	45	87.66	Alto	2024-12-09 17:37:00
Editar	Copiar	Borrar	339	Avenida Los Chasquis	45	87.39	Alto	2024-12-09 17:38:00
Editar	Copiar	Borrar	340	Avenida Los Chasquis	45	83.64	Alto	2024-12-09 17:39:00
Editar	Copiar	Borrar	341	Avenida Los Chasquis	45	80.39	Alto	2024-12-09 17:40:00
Editar	Copiar	Borrar	342	Avenida Los Chasquis	45	83.26	Alto	2024-12-09 17:41:00
Editar	Copiar	Borrar	343	Avenida Los Chasquis	44	77.21	Alto	2024-12-09 17:42:00
Editar	Copiar	Borrar	344	Avenida Los Chasquis	42	86.31	Alto	2024-12-09 17:43:00
Editar	Copiar	Borrar	345	Avenida Los Chasquis	44	90	Alto	2024-12-09 17:44:00
Editar	Copiar	Borrar	346	Avenida Los Chasquis	45	90	Alto	2024-12-09 17:45:00

Figura I3. Datos recolectados en la tarde

Resumen de Datos: 2024-12-16

Tabla I2. Datos más relevantes en el día 3 analizados

Hora	Vehículos	Ruido (dB)	Nivel de Ruido
06:45 - 07:00	42 - 45	61 - 90	Medio - Alto
12:50 - 12:59	33 - 41	59 - 89	Medio - Alto
17:30 - 17:45	41 - 45	68 - 90	Alto



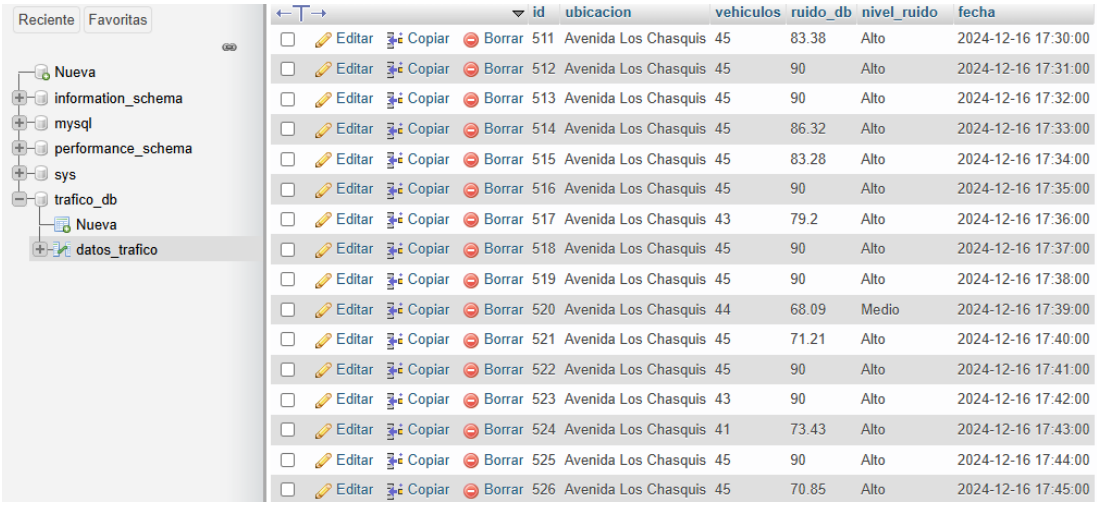
	id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar Copiar Borrar	406	Avenida Los Chasquis	42	82.56	Alto	2024-12-16 06:45:00
Editar Copiar Borrar	407	Avenida Los Chasquis	44	90	Alto	2024-12-16 06:46:00
Editar Copiar Borrar	408	Avenida Los Chasquis	42	61.27	Medio	2024-12-16 06:47:00
Editar Copiar Borrar	409	Avenida Los Chasquis	45	69.71	Medio	2024-12-16 06:48:00
Editar Copiar Borrar	410	Avenida Los Chasquis	45	65.11	Medio	2024-12-16 06:49:00
Editar Copiar Borrar	411	Avenida Los Chasquis	43	90	Alto	2024-12-16 06:50:00
Editar Copiar Borrar	412	Avenida Los Chasquis	43	90	Alto	2024-12-16 06:51:00
Editar Copiar Borrar	413	Avenida Los Chasquis	43	74.17	Alto	2024-12-16 06:52:00
Editar Copiar Borrar	414	Avenida Los Chasquis	44	90	Alto	2024-12-16 06:53:00
Editar Copiar Borrar	415	Avenida Los Chasquis	42	82.22	Alto	2024-12-16 06:54:00
Editar Copiar Borrar	416	Avenida Los Chasquis	42	72.85	Alto	2024-12-16 06:55:00
Editar Copiar Borrar	417	Avenida Los Chasquis	45	90	Alto	2024-12-16 06:56:00
Editar Copiar Borrar	418	Avenida Los Chasquis	45	87.34	Alto	2024-12-16 06:57:00
Editar Copiar Borrar	419	Avenida Los Chasquis	45	90	Alto	2024-12-16 06:58:00
Editar Copiar Borrar	420	Avenida Los Chasquis	45	76.32	Alto	2024-12-16 06:59:00

Figura I4. Datos recolectados en la mañana



	id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
Editar Copiar Borrar	471	Avenida Los Chasquis	38	59.04	Medio	2024-12-16 12:50:00
Editar Copiar Borrar	472	Avenida Los Chasquis	37	72.9	Alto	2024-12-16 12:51:00
Editar Copiar Borrar	473	Avenida Los Chasquis	35	59.28	Medio	2024-12-16 12:52:00
Editar Copiar Borrar	474	Avenida Los Chasquis	33	64.56	Medio	2024-12-16 12:53:00
Editar Copiar Borrar	475	Avenida Los Chasquis	36	68.02	Medio	2024-12-16 12:54:00
Editar Copiar Borrar	476	Avenida Los Chasquis	36	72.89	Alto	2024-12-16 12:55:00
Editar Copiar Borrar	477	Avenida Los Chasquis	35	58.89	Medio	2024-12-16 12:56:00
Editar Copiar Borrar	478	Avenida Los Chasquis	36	89.65	Alto	2024-12-16 12:57:00
Editar Copiar Borrar	479	Avenida Los Chasquis	41	87.7	Alto	2024-12-16 12:58:00
Editar Copiar Borrar	480	Avenida Los Chasquis	40	79.45	Alto	2024-12-16 12:59:00

Figura I5. Datos recolectados al medio día



		id	ubicacion	vehiculos	ruido_db	nivel_ruido	fecha
☐	Editar Copiar Borrar	511	Avenida Los Chasquis	45	83.38	Alto	2024-12-16 17:30:00
☐	Editar Copiar Borrar	512	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:31:00
☐	Editar Copiar Borrar	513	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:32:00
☐	Editar Copiar Borrar	514	Avenida Los Chasquis	45	86.32	Alto	2024-12-16 17:33:00
☐	Editar Copiar Borrar	515	Avenida Los Chasquis	45	83.28	Alto	2024-12-16 17:34:00
☐	Editar Copiar Borrar	516	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:35:00
☐	Editar Copiar Borrar	517	Avenida Los Chasquis	43	79.2	Alto	2024-12-16 17:36:00
☐	Editar Copiar Borrar	518	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:37:00
☐	Editar Copiar Borrar	519	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:38:00
☐	Editar Copiar Borrar	520	Avenida Los Chasquis	44	68.09	Medio	2024-12-16 17:39:00
☐	Editar Copiar Borrar	521	Avenida Los Chasquis	45	71.21	Alto	2024-12-16 17:40:00
☐	Editar Copiar Borrar	522	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:41:00
☐	Editar Copiar Borrar	523	Avenida Los Chasquis	43	90	Alto	2024-12-16 17:42:00
☐	Editar Copiar Borrar	524	Avenida Los Chasquis	41	73.43	Alto	2024-12-16 17:43:00
☐	Editar Copiar Borrar	525	Avenida Los Chasquis	45	90	Alto	2024-12-16 17:44:00
☐	Editar Copiar Borrar	526	Avenida Los Chasquis	45	70.85	Alto	2024-12-16 17:45:00

Figura I6. Datos recolectados en la tarde