
**Universidad Católica Andrés Bello**

**Facultad de Ingeniería**

**Escuela de Ingeniería Informática**

# Sistema de Monitoreo Acústico, para Identificar Sonidos y Generar Alertas de Emergencia

**Trabajo de Grado**

presentado ante la

**UNIVERSIDAD CATÓLICA ANDRÉS BELLO**

como parte de los requisitos para optar al título de

**INGENIERO EN INFORMÁTICA**

| | |
|---|---|
| Realizado por | Naim, Carmelo |
| | Sotillo, César |
| Tutor | Lárez, Jesús |
| Fecha | Marzo, 2025 |

## Índice de Contenido

# Índice de Tablas

## Índice de Figuras

# Índice de Fórmulas

**Universidad Católica Andrés Bello**

**Facultad de Ingeniería**

**Escuela de Ingeniería Informática**

**Sistema de Monitoreo Acústico, para Identificar Sonidos y Generar Alertas de Emergencia**

Autor: Naim, Carmelo
Sotillo, César
Tutor Académico: Lárez, Jesús
Fecha: Marzo, 2025

## Resumen

Este trabajo presenta el desarrollo de un sistema de monitoreo acústico basado en inteligencia artificial, que clasifica sonidos ambientales y genera alertas de emergencia al identificar eventos anómalos en tiempo real que podrían implicar que individuos se encuentren en una posición de vulnerabilidad. En este marco, el proyecto aborda la detección de eventos críticos, como cambios en la rutina o gritos de auxilio, con el fin de ofrecer una herramienta de detección que pueda alertar de forma temprana sobre situaciones anormales que podrían indicar una posición de vulnerabilidad del individuo. El sistema fue desarrollado utilizando dispositivos Edge como Raspberry Pi, integrando modelos preentrenados como YAMNet para clasificación de sonidos y Vosk para detección de palabras clave con modelos propios que perfilan el comportamiento acústico con Isolation Forest, LSTM y Transformers. La metodología empleada está basada en el modelo de desarrollo en espiral, lo cual permitió ciclos iterativos de análisis, diseño, implementación y validación, adaptándose a desafíos técnicos como la variabilidad del entorno sonoro y las limitaciones de hardware. Los resultados evidencian que el sistema es capaz de identificar patrones anómalos de sonido en tiempo real y generar notificaciones automáticas a contactos de emergencia. La solución considera la privacidad del usuario al no almacenar audios y procesar todo localmente. Se concluye que este sistema representa una herramienta ética para el monitoreo de entornos domésticos, alineándose con los Objetivos de Desarrollo Sostenible en salud, bienestar e inclusión tecnológica.

***Palabras clave—*** alertas de emergencia, dispositivos Edge, inteligencia artificial, monitoreo acústico, Raspberry Pi.

# Introducción

Este trabajo aborda el desarrollo de un sistema distribuido que, mediante el uso de inteligencia artificial, permite la detección de anomalías acústicas como herramienta de apoyo para la alerta temprana ante posibles emergencias. El sistema está enfocado en servir no solo a personas mayores o con discapacidades, sino también a cualquier individuo que se encuentre en una condición de vulnerabilidad. Su funcionamiento se basa en dispositivos que primero clasifican los sonidos ambientales en tiempo real para, posteriormente, analizar la secuencia de dichas clasificaciones. Este análisis permite al sistema caracterizar el comportamiento acústico "normal" de un entorno y, en consecuencia, identificar eventos que se desvían de esa normalidad para generar una alerta a contactos predefinidos. La importancia de este proyecto radica en su capacidad de alertar sobre situaciones anormales que podrían indicar una situación de emergencia, ofreciendo un sistema de alerta temprana que respeta la privacidad. Además, tiene el potencial de beneficiar a familias, cuidadores e instituciones y facilitar una intervención más oportuna de los servicios de emergencia, incluso en casos de violencia doméstica o el monitoreo de personas con depresión. A través de la identificación de situaciones anormales y la emisión de alertas, este sistema contribuye a una mejor calidad de vida.

Para el desarrollo del sistema, se empleó una metodología basada en el modelo de desarrollo en espiral, que permite un enfoque iterativo y flexible. Cada ciclo del proceso abarca fases de planificación, análisis de riesgos, desarrollo y evaluación, permitiendo ajustes continuos en el diseño y la implementación del sistema. Esta metodología es adecuada para proyectos con incertidumbre, ya que facilita la evolución del sistema según la retroalimentación obtenida, asegurando su mejora progresiva a lo largo del desarrollo.

El presente trabajo está estructurado en cinco capítulos, En el Capítulo I, referido al Planteamiento del Problema, se describe la problemática, se establecen los objetivos, justificación, alcance y limitaciones del proyecto. En el Capítulo II, se expone el Marco Teórico donde se recopilan los antecedentes y las bases teóricas que sustentan el trabajo. En el Capítulo III, se presenta el Marco Metodológico, que describe el tipo de investigación, técnicas e instrumentos de recolección de datos, metodología de desarrollo y el procedimiento metodológico. En el Capítulo IV, se expone el Desarrollo y Resultados, donde se describe como el procedimiento metodológico dio respuesta a cada uno de los objetivos planteados, y en el Capítulo V, se presentan las conclusiones y recomendaciones sobre el trabajo realizado. Finalmente se listan las

referencias bibliográficas utilizadas, seguidas de los anexos y apéndices.

# Capítulo I. El Problema

## Planteamiento del Problema

En la actualidad, existe un número significativo de casos de personas que sufrieron algún accidente o complicación de una condición de salud, producto de la detección tardía de estos eventos, una tendencia que ha ido en aumento debido al envejecimiento de la población, que de acuerdo a los expuesto por la CEPAL (2005), dice que: "el envejecimiento demográfico en América Latina se triplicará entre el 2000 y 2050, con un aumento significativo en la proporción de personas mayores de 60 años" y a la estructura familiar moderna, que según Peña (2016), "El aumento de los hogares unipersonales se ha convertido en un fenómeno extendido dentro del mundo occidental desarrollado". Esta situación, expone a las personas a múltiples riesgos, especialmente cuando se enfrentan a emergencias domésticas, problemas de salud o accidentes que comprometen su bienestar físico y emocional. La falta de asistencia inmediata en situaciones críticas puede transformar un incidente inicialmente manejable en una amenaza grave para la vida.

Algunos casos ilustran estos riesgos. Un ejemplo es el de un hombre octogenario en Zaragoza, España, cuyo fallecimiento en su domicilio pasó desapercibido durante una semana. No fue hasta que una vecina notó su ausencia que se alertó a las autoridades. Aunque la causa de su muerte fue natural, este caso pone en evidencia la vulnerabilidad de las personas que viven solas y la falta de un mecanismo que pueda detectar la inactividad prolongada o cambios abruptos en la rutina diaria (Ralla, 2024). En una situación como esta, el problema no es la atención médica, sino la ausencia de una alerta automática que notifique una anomalía grave (la total falta de actividad) a familiares o servicios de emergencia. De forma similar, en un caso no fatal como una caída, la falta de esa misma alerta inmediata sí puede derivar en graves complicaciones físicas, transformando un accidente manejable en una crisis de salud. Estos incidentes no son excepcionales, sino que representan una problemática real, especialmente en el contexto de los hogares unipersonales.

El envejecimiento de la población es un fenómeno natural que está incrementando la cantidad de personas mayores que viven en solitario. De acuerdo a lo comentado por la Naciones Unidas (2022): "se estima que para el 2050, el 16 % de la población mundial estará compuesta por personas mayores de 65 años, muchas de las cuales optarán o se verán forzadas a vivir sin compañía". Esta tendencia plantea un desafío crítico en cuanto a su seguridad y bienestar. La falta de asistencia oportuna en emergencias no solo

representa una vulnerabilidad significativa para la vida de estas personas ante eventos que pongan en riesgo su vida o integridad física, sino que, a largo plazo, puede contribuir a un deterioro progresivo de su estado de salud y autonomía. Además, situaciones como caídas, ataques cardíacos o accidentes domésticos, que podrían ser atendidos a tiempo, se convierten en amenazas letales debido a la ausencia de un sistema de detección temprana.

Aunque en los últimos años se han desarrollado algunas soluciones tecnológicas para abordar esta problemática, como el dispositivo Zoe Fall, que detecta caídas a través de ondas WiFi, o el Apple Watch, que incluye una función de detección de caídas, estas tecnologías no son accesibles para todos ni cubren una gama suficientemente amplia de emergencias. De acuerdo a lo comentado por Carmona (2024): “estas herramientas están mayormente enfocadas en eventos puntuales, como caídas, pero no proporcionan un monitoreo continuo y adaptable a otras situaciones que podrían requerir detección temprana”. Adicionalmente, la mayoría de estos dispositivos dependen de la capacidad del usuario para activarlos o configurarlos adecuadamente, lo cual representa una barrera tecnológica en personas que, por su edad o condición, pueden no estar familiarizadas con el uso de la tecnología.

La falta de mecanismos efectivos para detectar emergencias a tiempo no solo prolongará el sufrimiento de las personas en situaciones críticas, sino que también incrementará las tasas de mortalidad y discapacidad asociadas a estos eventos. La inacción en este sentido implicaría ignorar las necesidades de una población vulnerable, cuya seguridad y calidad de vida pueden verse directamente influenciadas por la imposibilidad de notificar estos eventos.

Ante este panorama, se evidencia la necesidad de desarrollar una solución tecnológica que supere las limitaciones de los dispositivos actuales, ofreciendo un monitoreo más integral, no invasivo y accesible. Por lo tanto, este trabajo se centra en investigar y proponer un sistema de monitoreo inteligente basado en el análisis del entorno acústico. La pregunta central que guía esta investigación es: ¿Es posible, utilizando inteligencia artificial en dispositivos de bajo costo, crear un sistema que caracterice el comportamiento sonoro habitual de un hogar y genere alertas fiables ante patrones anómalos, respetando al mismo tiempo la privacidad del usuario?

Este sistema, ofrecerá un monitoreo continuo, creando un perfil acústico de cada ambiente del hogar que permitirá diferenciar entre sonidos típicos y atípicos en cada entorno.

Al clasificar un sonido que se salga de los patrones típicos, el sistema enviará una consulta al individuo para confirmar si está en buen estado; en caso de no recibir respuesta o de que se confirme una situación de emergencia, enviará una alerta inmediata a los contactos de emergencia predefinidos.

Esto permite generar un perfil de audio adaptado a cada entorno del hogar, ya sea de una persona mayor, una persona con discapacidad, un hogar con muchos niños, o un entorno con vecinos que escuchan música alta con frecuencia. Este perfil facilita la detección de situaciones anómalas que puedan representar un riesgo, especialmente en viviendas donde viven personas solas, quienes pueden enfrentar desafíos por la falta de compañía o asistencia. Así, el sistema está diseñado no solo para adultos mayores o personas con discapacidad, sino para cualquier persona en situación de vulnerabilidad.

Esta solución tiene el potencial de reducir los riesgos asociados a la detección tardía de estos eventos, proporcionando a las personas un mayor nivel de seguridad sin comprometer su privacidad.

## Objetivos

### Objetivo General.

Desarrollar un sistema para generar alertas en casos de emergencia basado en el monitoreo acústico.

### Objetivos Específicos.

1. Analizar los conceptos asociados a la analítica de sonidos ambientales identificando los conceptos necesarios para diseñar el sistema.
2. Diseñar un sistema para generar alertas en casos de emergencia basado en el monitoreo acústico en función del análisis realizado.
3. Implementar el sistema para generar alertas en casos de emergencia basado en el monitoreo acústico según el diseño realizado.
4. Validar el sistema para generar alertas en casos de emergencia basado en el monitoreo acústico con respecto al análisis realizado.
5. Elaborar la documentación técnica y guías del usuario y sistema.

## Alcance

El proyecto tiene como objetivo desarrollar un sistema de monitoreo acústico que perfile el comportamiento de los habitantes de una vivienda, a través del reconocimiento y clasificación de los

sonidos ambientales en las diferentes estancias de la misma. El sistema generará alarmas en tiempo real que alertarán a contactos de emergencia predefinidos. Para ello se llevará a cabo un análisis de los conceptos asociados a la analítica de sonidos donde se identificarán los conceptos necesarios para diseñar el sistema.

Con base en los resultados del análisis, se procede a diseñar el sistema de monitoreo acústico capaz de reconocer sonidos. El diseño comprende la definición de la arquitectura del sistema, contemplando los componentes esenciales, como la red de micrófonos para la captura de sonidos y los mecanismos de alerta, detección de anomalías y de reconocimiento acústico. El diseño considera la importancia de la privacidad de los usuarios. Según los resultados del análisis realizado para el desarrollo del sistema de monitoreo acústico, se evalúa si es necesario crear un dataset desde cero, obtener uno de internet y modificarlo según sea necesario, o bien emplear un modelo ya preentrenado.

Una vez completado el diseño, se procede a la implementación del sistema de monitoreo acústico. Esta fase incluye la configuración e instalación de los componentes físicos, como los micrófonos distribuidos en las estancias de la vivienda, asegurando una cobertura adecuada para la captura de sonidos relevantes. Los algoritmos de procesamiento de señales y los modelos de inteligencia artificial, definidos en la fase de diseño, fueron desarrollados y adaptados para realizar el reconocimiento y clasificación de los sonidos.

Se desarrollaron los mecanismos de alerta y notificación, que emiten avisos a los contactos de emergencia predefinidos.

Luego se realiza la validación del sistema, se lleva a cabo pruebas funcionales en ambientes controlados y no controlados para comparar los resultados obtenidos con los objetivos planteados y realizar ajustes en caso de ser necesario.

Finalmente se elabora la documentación del sistema, manuales de usuario, manuales de sistema, descripción de los componentes, diagramas y cualquier otro documento necesario.

## Limitaciones

En la implementación del sistema, ciertas limitaciones que podrían influir en el desempeño de su desarrollo. A continuación, se describirán algunas de las restricciones que se podrían presentar:

Limitaciones tecnológicas: La precisión del sistema podrá verse afectada por la calidad de los sensores acústicos y la capacidad de procesamiento de los microcontroladores empleados.

Limitaciones de datos: La recolección de suficientes muestras de sonidos para entrenar el modelo

puede ser limitada, lo que puede comprometer el correcto funcionamiento del sistema en escenarios no previstos.

Limitaciones operativas: La inestabilidad del suministro eléctrico, especialmente en contextos donde se realizaron las pruebas, puede interrumpir la recolección continua de datos y afectar la disponibilidad del sistema para capturar eventos clave en su contexto temporal.

Limitaciones de hardware: El reentrenamiento de modelos complejos, como los basados en Transformers o LSTM, requiere recursos de cómputo significativos que superan la capacidad de dispositivos de bajo consumo como Raspberry Pi. Esto restringe la posibilidad de actualizar o adaptar los modelos directamente en el dispositivo, obligando a depender de un servidor externo para estas tareas.

## Justificación

En la actualidad, la atención a personas en condiciones de vulnerabilidad justifica el desarrollo de soluciones tecnológicas que permitan una respuesta rápida ante situaciones críticas, sin comprometer la privacidad ni depender de una supervisión constante.

Desde una perspectiva social, el sistema puede beneficiar a personas que, por condiciones físicas o emocionales, se encuentran en mayor riesgo de enfrentar emergencias sin posibilidad de pedir ayuda inmediata, como en casos de caídas, crisis de salud o situaciones de violencia doméstica. También contribuye a la labor de cuidadores y servicios de emergencia al ofrecer alertas tempranas.

En el ámbito institucional, el proyecto plantea una solución aplicable en hogares de cuidado, hospitales o comunidades, ya que su arquitectura de bajo costo y su implementación con componentes disponibles permiten su adaptación a contextos con recursos limitados. Esta característica puede ser relevante en la formulación de futuras regulaciones públicas que busquen integrar herramientas automatizadas en estrategias de atención ciudadana.

El sistema se vincula con los Objetivos de Desarrollo Sostenible (ODS), particularmente el ODS 3: Salud y bienestar, al aportar un mecanismo de alerta temprana en situaciones críticas, y el ODS 10: Reducción de desigualdades, al ofrecer una solución técnica accesible que no depende de infraestructura compleja. Su diseño se considera adaptable por ser independiente del ambiente donde se implemente, lo que lo hace funcional en diferentes contextos sociales, promoviendo mayor equidad en el acceso a herramientas de apoyo.

# Capítulo II. Marco Teórico

Este capítulo presenta las bases teóricas y las investigaciones previas que sustentan el desarrollo del Sistema de Monitoreo Acústico para Identificar Sonidos y Generar Alertas de Emergencia. A través de la revisión de fuentes documentales, se exponen los conceptos y enfoques principales relacionados con el procesamiento de señales acústicas, la clasificación de sonidos mediante técnicas de inteligencia artificial y los sistemas de alertas automatizadas.

El marco teórico no solo proporciona un contexto académico para el proyecto, sino que también justifica las decisiones técnicas y metodológicas adoptadas en el diseño del sistema. En las siguientes secciones se detallan los antecedentes investigativos, los modelos de IA aplicados a la clasificación de sonidos y los fundamentos teóricos que permiten la generación de alertas en situaciones de emergencia.

En las siguientes secciones se detallan los antecedentes y los conceptos clave que sustentan este trabajo.

## Antecedentes de la Investigación

### Real-time Audio Classification on an Edge Device [Clasificación de audio en tiempo real en un dispositivo perimetral].

Malmberg y Radszuweit (2021) Abordan la implementación de modelos de aprendizaje automatico en dispositivos edge para la clasificación de audio en tiempo real, Se centra en el uso del modelo YAMNet, que fue reentrenado para detectar eventos acústicos como disparos, rotura de cristales, animales y habla humana. Este modelo puede desplegarse en dispositivos edge en su versión completa con TensorFlow como en versiones optimizadas con TensorFlow Lite, con el objetivo de comparar la precisión, el tiempo de inferencia y el uso de memoria de cada variante.

Con el propósito de evaluar el desempeño, trabajaron en una serie de experimentos en los que compararon la precisión del modelo en ambas versiones de TensorFlow. Se encontraron con que, aunque existía una pérdida de precisión en la versión lite los resultados eran comparables a los de la versión completa. Esto implica que TensorFlow Lite es una opción viable y crea la posibilidad de trabajar con

dispositivos de bajo consumo y recursos limitados como ESP32

Este enfoque no solo demuestra la factibilidad de implementar modelos de clasificación de audio en tiempo real en entornos con recursos restringidos, sino que también abre la puerta a futuras investigaciones que busquen mejorar estos modelos. La capacidad de desplegar soluciones de inteligencia artificial en dispositivos edge amplía las posibilidades de aplicaciones en áreas como la seguridad, la vigilancia y el monitoreo ambiental, permitiendo respuestas más rápidas y reduciendo la dependencia de infraestructuras centralizadas.

### Analysing RMS and peak values of vibration signals for condition monitoring of wind turbine gearboxes [Análisis de valores RMS y pico de señales de vibración para la monitorización del estado de las cajas de engranajes de aerogeneradores].

Igba, Alemzadeh, Durugbo, y Eiriksson (2016) abordan el monitoreo de condición en aerogeneradores con el objetivo de detectar fallos en las cajas de engranajes mediante el análisis de valores RMS y picos de vibraciones. La investigación propone tres modelos: correlación de señales, vibración extrema e intensidad RMS, validados con datos en dominio del tiempo. A través del uso de la teoría de valores extremos, se identificaron indicadores que permiten detectar fallos en etapas tempranas, lo que facilita la planificación del mantenimiento y minimiza el tiempo de inactividad de los aerogeneradores. Los resultados demostraron que el monitoreo de vibraciones proporciona información clave sobre el estado de los componentes y que la precisión de cada técnica depende de la física de la falla, sugiriendo un enfoque integral que combine distintas estrategias para una evaluación más robusta de la salud de los aerogeneradores.

El uso del valor RMS en este estudio es fundamental para la monitorización de la condición de las cajas de engranajes en aerogeneradores, ya que permite evaluar el nivel global de vibración y detectar fallos progresivos, como el desgaste de rodamientos y grietas en ejes. Igba y cols. (2016) destacan que el RMS es una métrica confiable para identificar tendencias anómalas en las vibraciones, lo que facilita la detección temprana de fallos antes de que se conviertan en problemas críticos. A pesar de ciertas limitaciones, como su menor sensibilidad a fallos incipientes en los dientes de los engranajes, el análisis de RMS sigue siendo un pilar clave en la estrategia de mantenimiento basado en condición (CBM), al proporcionar información valiosa sobre la evolución del estado de los componentes mecánicos.

### Metodología para la identificación de eventos sonoros anómalos.

Torija, Ruiz, y Ramos-Diao (2018) presentan una metodología para la detección de eventos sonoros anómalos en entornos urbanos. Su trabajo se centra en el análisis de sucesos acústicos que generan incrementos bruscos de energía sonora en el paisaje sonoro urbano, lo que puede provocar molestias significativas en la población expuesta. Según los autores, los eventos sonoros anómalos, como bocinas, gritos, explosiones o disparos, generan una percepción intensificada del ruido ambiental debido a su impacto en la focalización de la atención. Para abordar este problema, desarrollaron un modelo basado en la evaluación del tiempo de estabilización de la medición del ruido ambiental y el análisis del factor cresta, que permite estimar el incremento de energía sonora generado por estos eventos.

El estudio incluyó mediciones en 35 localizaciones de la ciudad de Granada, utilizando un sonómetro Brüel Kjaer tipo 1. La metodología aplicada permitió definir un evento sonoro anómalo como aquel que provoca un incremento del nivel de energía sonora de al menos un 25 % respecto al nivel de fondo caracterizado por el descriptor LA90.

Los resultados indicaron que el número de eventos sonoros anómalos presentes en una ubicación está altamente correlacionado con la variabilidad de la energía sonora en la zona. Además, se evidenció que el factor cresta es un parámetro clave para estimar la magnitud del impacto acústico.

### Edge AI for Real-Time Anomaly Detection in Smart Homes [Inteligencia artificial de borde para la detección de anomalías en tiempo real en hogares inteligentes].

Reis y Serôdio (2025) proponen un marco de trabajo para la detección de anomalías en tiempo real en dispositivos de borde (Edge AI), con el objetivo de superar las limitaciones de latencia y privacidad de los sistemas centralizados en la nube. Su enfoque se basa en una arquitectura híbrida que combina dos modelos de aprendizaje automático, Isolation Forest (IF), un modelo ligero para la detección rápida de eventos puntuales, y un Autoencoder de Memoria a Corto y Largo Plazo (LSTM-AE), un modelo de aprendizaje profundo para identificar patrones anómalos en secuencias de datos.

El sistema fue evaluado en plataformas de borde como Raspberry Pi y NVIDIA Jetson Nano para validar su desempeño en entornos con recursos limitados. Con el propósito de evaluar el rendimiento, los autores compararon la precisión y la eficiencia de ambos modelos. Los resultados demostraron que el modelo LSTM alcanzó una mayor precisión de detección (hasta un 93.6 %), pero con un costo computacional y

energético más elevado. Un hallazgo clave fue la aplicación de técnicas de cuantización sobre el modelo LSTM, lo que permitió una reducción del 76 % en el tiempo de inferencia y del 35 % en el consumo de energía. Esto confirma que es viable trabajar con versiones ligeras de modelos complejos en dispositivos edge sin sacrificar significativamente la precisión.

Este enfoque no solo demuestra la factibilidad de implementar sistemas de monitoreo de anomalías en tiempo real que sean rápidos, privados y eficientes energéticamente, sino que también subraya la ventaja de una arquitectura híbrida que equilibra velocidad y precisión. La capacidad de desplegar estas soluciones directamente en el hardware del hogar amplía sus aplicaciones a otros dominios como edificios inteligentes e IoT industrial, reduciendo la dependencia de la infraestructura en la nube y permitiendo respuestas más autónomas ante eventos críticos.

## Bases Teóricas

### Dispositivos Edge.

El incremento en la demanda de los servicios y aplicaciones en las últimas décadas del uso de la Internet, ha contribuido a un fuerte aumento de los requisitos de procesamiento y almacenamiento de datos. Son diversos, en términos de los recursos que requieren las diferentes aplicaciones y, por lo tanto, a menudo invocan soluciones a medida (Doluí y Kanti Datta, 2017).

Según Medina (2019) Edge Device, en este contexto, se refiere a elementos con capacidades limitadas que tiene su propio conjunto de recursos: CPU, memoria, almacenamiento, y red. Pueden ser Smartphone, Smartglasses, smartwatches, tablets, routers, vehículos autónomos, o cualquier dispositivo de IoT con capacidad de proceso. En este escenario, Edge Computing, representa una solución para enfrentar y aliviar la carga de procesamiento y almacenamiento en la nube, en aplicaciones y tecnologías que requieren de un ancho de banda exponencial y una baja o nula latencia.

Shi, Cao, Zhang, Li, y Xu (2016) definen Edge Computing como un paradigma de computación distribuida que acerca el procesamiento y almacenamiento de datos a la fuente de generación, es decir, al "borde" de la red. Este enfoque permite reducir la latencia, optimizar el uso del ancho de banda y mejorar la eficiencia en aplicaciones que requieren respuestas en tiempo real, como el Internet de las Cosas (IoT), la realidad aumentada, los vehículos autónomos y las ciudades inteligentes. Edge Computing se basa en dispositivos periféricos (edge devices) y nodos locales que realizan tareas de procesamiento y

almacenamiento, lo que reduce la dependencia de la infraestructura centralizada de la nube.

### Raspberry pi.

Según Richardson y Wallace (2016), la Raspberry Pi es un dispositivo de computación de bajo costo y alto rendimiento que ha ganado popularidad en diversos campos debido a su versatilidad y facilidad de uso. Este dispositivo, del tamaño de una tarjeta de crédito, está equipado con un procesador ARM, memoria RAM, puertos de entrada/salida y conectividad de red, lo que lo hace una opcion valida para proyectos educativos, de automatización y desarrollo de prototipos. La Raspberry Pi es capaz de ejecutar sistemas operativos basados en Linux, lo que permite a los usuarios programar y personalizar sus aplicaciones según sus necesidades (Richardson y Wallace, 2016).

Además, Upton y Halfacree (2020) destacan que la Raspberry Pi ha evolucionado significativamente desde su lanzamiento, con modelos más potentes como la Raspberry Pi 4, que ofrece mayores capacidades de procesamiento, almacenamiento y conectividad. Este modelo incluye soporte para redes Gigabit Ethernet, puertos USB 3.0 y salidas de video de alta definición, lo que lo convierte en una herramienta poderosa para aplicaciones más exigentes, como servidores domésticos, centros multimedia y sistemas embebidos avanzados (Upton y Halfacree, 2020).

### Python.

Python es un lenguaje de programación poderoso, elegante y fácil de leer, diseñado para simplificar la creación de programas mediante una sintaxis clara y estructurada. Según el documento, Python destaca por su versatilidad en aplicaciones del mundo real, su enfoque en la legibilidad del código y su capacidad para integrar paradigmas como la programación orientada a objetos y funcional. Además, es software libre con una comunidad activa y una implementación estándar consolidada (Yuill y Halpin, 2006).

### Tensorflow.

Según Goldsborough (2016b) TensorFlow es una biblioteca de software de deep learning de código abierto desarrollada por Google que permite definir, entrenar y desplegar modelos de machine learning mediante la representación de algoritmos como grafos computacionales.

La librería TensorFlow opera construyendo un grafo computacional en el que cada nodo representa

una operación (por ejemplo, una función matemática, una transformación o una capa de una red neuronal) y cada arista transporta un tensor, es decir, un arreglo multidimensional de datos. Esta arquitectura facilita diversas optimizaciones, como la eliminación de subgrafos redundantes, y permite distribuir la ejecución de la computación a lo largo de múltiples dispositivos (CPUs, GPUs, TPUs) e incluso en entornos distribuidos. De esta manera, se optimiza tanto el uso de memoria como el rendimiento, haciendo posible el entrenamiento y despliegue de modelos complejos a gran escala. (Goldsborough, 2016b).

### NestJs.

Según Sabo (2020) Sabo (2020) NestJS es un framework para el desarrollo de aplicaciones del lado del servidor basado en Node.js, que se escribe en TypeScript. Proporciona una estructura modular y escalable mediante el uso de patrones modernos como la inyección de dependencias, controladores y módulos, facilitando la creación de aplicaciones backend mantenibles y robustas.

NestJS aprovecha la solidez de Node.js y Express.js, pero se diferencia al introducir un enfoque inspirado en Angular para la organización de la aplicación. Gracias a su arquitectura basada en módulos, cada parte de la aplicación se encapsula en unidades independientes que facilitan la reutilización y la escalabilidad. Además, el framework implementa un avanzado sistema de inyección de dependencias, lo que permite gestionar y suministrar las instancias de servicios de manera automática, reduciendo el acoplamiento entre componentes y promoviendo un diseño orientado a pruebas. Otro pilar fundamental de NestJS es su fuerte integración con TypeScript, lo cual aporta tipificación estática y facilita la detección temprana de errores durante el desarrollo. La utilización de decoradores en NestJS permite agregar metadatos a clases, métodos y propiedades, lo que habilita la implementación de características como interceptores, pipes y controladores para la validación y transformación de datos, así como para el manejo de rutas HTTP. Además, la herramienta Nest CLI agiliza la generación de nuevos proyectos y componentes, garantizando que se siga una estructura coherente en toda la aplicación, lo que resulta especialmente útil en proyectos complejos y colaborativos (Sabo, 2020).

### Inteligencia artificial.

La Inteligencia Artificial (IA) se define como el estudio de agentes que perciben su entorno a través de sensores y actúan sobre él mediante actuadores, con el objetivo de maximizar su utilidad esperada. Según Russell y Norvig (2022), “ La IA es el estudio de agentes que reciben percepciones del entorno y

realizan acciones. Cada uno de estos agentes implementa una función que asigna secuencias de percepción a acciones, y cubrimos diferentes formas de representar estas funciones para lograr el mejor resultado esperado." (p. 19).

Los fundamentos de la IA se entrelazan con múltiples disciplinas. La filosofía aporta marcos éticos y lógicos, como señalan los autores: "El filósofo griego Aristóteles fue uno de los primeros en intentar codificar el "pensamiento correcto" sus silogismos proporcionaron patrones para estructuras argumentales que siempre produjeron conclusiones correctas." (p. 21). Las matemáticas y la estadística proporcionan herramientas para el razonamiento probabilístico: "La probabilidad rápidamente se convirtió en una parte invaluable de las ciencias cuantitativas, ayudando a lidiar con mediciones inciertas y teorías incompletas." (p. 26). La economía contribuye con teorías de decisión y utilidad: "La teoría de la decisión, que combina la teoría de la probabilidad con la teoría de la utilidad, proporciona un marco formal y completo para las decisiones individuales tomadas en condiciones de incertidumbre." (p. 28). La neurociencia y la psicología inspiran modelos cognitivos: "El campo interdisciplinario de la ciencia cognitiva reúne modelos informáticos de la IA y técnicas experimentales de la psicología para construir teorías precisas y comprobables de la mente humana." (p. 21). Por último, la ingeniería y la computación permiten implementar sistemas eficientes: "La historia de la IA es también la historia del diseño de arquitecturas cada vez más sofisticadas para programas de agentes." (p. 65).

La IA actual ha alcanzado hitos significativos en diversos dominios. De acuerdo a los expuesto por Russell y Norvig (2022): "Los sistemas que usan IA han alcanzado o superado el rendimiento humano en ajedrez, Go, póquer, Pac-Man, Jeopardy, detección de objetos ImageNet, reconocimiento de voz y diagnóstico de retinopatía diabética." (p. 46). En aplicaciones prácticas, destacan los vehículos autónomos: "Los vehículos de prueba de Waymo superaron la marca de 10 millones de millas recorridas en vías públicas sin sufrir accidentes graves." (p. 47), y sistemas de diagnóstico médico: "Los algoritmos de IA ahora igualan o superan a los médicos expertos en el diagnóstico de muchas afecciones, como el cáncer metastásico y las enfermedades oftálmicas." (p. 48). Además, herramientas como "Los sistemas de traducción automática ahora permiten la lectura de documentos en más de 100 idiomas, lo que genera cientos de miles de millones de palabras por día." (p. 47) evidencian su impacto global. No obstante, persisten retos éticos y técnicos, como la alineación de valores humanos y la escalabilidad en entornos complejos, que definen la frontera actual de investigación.

## Cadenas de Markov.

Fundamentos teóricos Las Cadenas de Markov, nombradas en honor al matemático ruso Andrei Andreevich Markov (1856–1922), son procesos estocásticos que modelan sistemas donde el futuro depende únicamente del estado presente, sin influencia directa del pasado (Matas Soberón, 2024, p. 13). Este principio, conocido como propiedad de Markov, fue inicialmente aplicado por Markov en el análisis de secuencias de vocales y consonantes en textos literarios, como Eugene Onegin de Pushkin, sentando las bases para su uso en campos como la física, biología y ciencias sociales (Matas Soberón, 2024, p. 11). Posteriormente, estas cadenas han sido utilizadas para modelar sistemas en epidemiología, teoría de colas y dinámica de poblaciones (Bobadilla Osses, 2010, p. 5).

Una Cadena de Markov se define formalmente como un proceso estocástico $X_n$ sobre un espacio de estados S (finito o numerable), donde la probabilidad de transición al siguiente estado depende exclusivamente del estado actual. Esta propiedad se traduce en una matriz de transición $P = (p_{ij})$, donde cada entrada $p_{ij}$ representa la probabilidad de pasar del estado i al estado j. La matriz P es estocástica, es decir, sus filas suman 1 y sus elementos son no negativos (Matas Soberón, 2024, p. 14-15). Además, en sistemas más avanzados, se pueden considerar Cadenas de Markov de tiempo continuo, las cuales presentan una relación con la teoría de semigrupos de operadores (Bobadilla Osses, 2010, p. 6).

Las cadenas de Markov pueden clasificarse en homogéneas, cuando la matriz de transición P permanece constante en el tiempo (Matas Soberón, 2024, p. 23). Un estado es recurrente si la cadena regresa a él infinitas veces, y transitorio si eventualmente lo abandona para siempre. Esto se determina mediante la representación canónica de P, que identifica clases cerradas (conjuntos de estados que no pueden abandonarse) (Matas Soberón, 2024, p. 24-25). En el contexto de sistemas biológicos, por ejemplo, una cadena de Markov puede utilizarse para modelar la propagación de una enfermedad dentro de una población, donde los estados pueden representar niveles de infección y la recurrencia indicar posibles rebrotes (Bobadilla Osses, 2010, p. 91).

Para cadenas regulares (matrices P con todas sus entradas positivas en alguna potencia), las probabilidades convergen a un estado estacionario w, único vector estocástico que satisface w = wP. Este resultado se fundamenta en el teorema de Perron-Frobenius y técnicas como la descomposición de Jordan (Matas Soberón, 2024, p. 29-30). Un caso particular de convergencia ocurre en modelos de colas, donde la distribución estacionaria describe la cantidad promedio de clientes en espera en el largo plazo (Bobadilla Osses, 2010, p. 84).

Los autores en sus respectivas publicaciones ilustran las utilidades de las Cadenas de Markov en casos reales. En fútbol, se utilizan para modelizar resultados (ganar, empatar, perder) en partidos de la liga española, calculando probabilidades de equilibrio para equipos ganadores y perdedores (Matas Soberón, 2024, p. 33-41). En póquer, se emplean para el análisis de rondas de apuestas y distribución de cartas, identificando estados recurrentes (como abandonar una partida) mediante matrices de transición (Matas Soberón, 2024, p. 42-48). En Google AdWords, se aplican para predecir el comportamiento de clics en anuncios, utilizando cadenas regulares para optimizar estrategias de marketing (Matas Soberón, 2024, p. 49-53). También se usan en el modelo de Reed-Frost en epidemiología para estudiar la propagación de enfermedades infecciosas, clasificando estados en susceptibles, infectados y recuperados (Bobadilla Osses, 2010, p. 91-107). Otro ejemplo es su aplicación en cadenas de colas, modelos de atención en sistemas de servicio como supermercados o sistemas de telecomunicaciones, donde la matriz de transición describe la dinámica de llegada y atención de clientes (Bobadilla Osses, 2010, p. 57-61).

## Agente.

Según Russell y Norvig (2022), un agente es cualquier entidad que percibe su entorno a través de sensores y actúa sobre él mediante actuadores. En este sentido, un agente puede ser un ser humano, un robot o un sistema de software, siempre que cumpla con esta función de percepción y acción. Por ejemplo, un agente humano posee sensores como los ojos y oídos, y actuadores como las manos y las piernas. Un agente robótico puede incluir cámaras y sensores infrarrojos como entrada, y motores como salida. Un software, en cambio, percibe datos en forma de archivos o paquetes de red y responde modificando archivos o enviando información a otros sistemas (Russell y Norvig, 2022).

El comportamiento de un agente está determinado por su función de agente, la cual define cómo actúa en función de la secuencia de percepciones que ha experimentado. En otras palabras, un agente toma decisiones basándose en su conocimiento previo y en los datos que recibe en tiempo real. Russell y Norvig (2022) también introducen el concepto de agente racional, que es aquel que elige acciones que maximizan su desempeño según un criterio predefinido. La complejidad del entorno en el que opera el agente influye directamente en su diseño y en su capacidad de tomar decisiones óptimas (Russell y Norvig, 2022).

## Ciencia de datos.

Según Provost y Fawcett (2013), la ciencia de datos se enfoca en el uso de datos para tomar decisiones informadas, utilizando herramientas como el aprendizaje automático, la minería de datos y el análisis estadístico. Los autores destacan que la ciencia de datos no solo se trata de manejar grandes volúmenes de datos, sino también de entender cómo estos pueden ser utilizados para generar valor en diferentes contextos, como en el ámbito empresarial, científico o social. Por su parte, Russell y Norvig (2022) enfatizan que la ciencia de datos es una disciplina que se apoya en la inteligencia artificial y el aprendizaje automático para modelar y predecir comportamientos a partir de datos. Los autores mencionan que la ciencia de datos es fundamental en la creación de sistemas inteligentes que pueden aprender de los datos y mejorar su desempeño con el tiempo. Además, resaltan la importancia de la ética en el manejo de datos, especialmente en aplicaciones que involucran la privacidad y la seguridad de las personas. García, Molina, y Berlanga (2018) complementan esta definición al señalar que la ciencia de datos es una disciplina que integra técnicas analíticas avanzadas, como el aprendizaje estadístico y la minería de datos, para resolver problemas complejos. Los autores destacan que la ciencia de datos no solo se limita al análisis de datos, sino que también incluye la preparación, limpieza y transformación de los datos para que puedan ser utilizados en modelos predictivos y descriptivos. Además, resaltan la importancia de la visualización de datos como una herramienta clave para comunicar los resultados de manera efectiva.

En el contexto del Sistema de Monitoreo Acústico para Identificar Sonidos y Generar Alertas de Emergencia, la ciencia de datos juega un papel fundamental en varias etapas del proyecto. En primer lugar, se utiliza para el procesamiento y análisis de señales acústicas, donde se aplican técnicas de aprendizaje automático para clasificar sonidos ambientales y detectar eventos anómalos. Como mencionan Provost y Fawcett (2013), el aprendizaje automático es una herramienta poderosa para la clasificación de datos, especialmente en aplicaciones que requieren respuestas en tiempo real, como es el caso de este sistema.

Además, la ciencia de datos es esencial en la creación y entrenamiento de modelos predictivos que permiten identificar patrones de sonidos asociados a situaciones de emergencia. Russell y Norvig (2022) destacan que los modelos de inteligencia artificial, como las redes neuronales y las cadenas de Markov, son fundamentales para la predicción y clasificación de eventos basados en datos históricos. En este proyecto, se utilizan modelos preentrenados, como YAMNet, y se adaptan para la detección de sonidos específicos, lo que demuestra la aplicación práctica de la ciencia de datos en la creación de sistemas inteligentes. Por último, García y cols. (2018) resaltan la importancia de la visualización de datos en la comunicación de

resultados. En este proyecto, la ciencia de datos no solo se limita al análisis de sonidos, sino que también incluye la generación de alertas visuales y notificaciones que permiten a los usuarios entender rápidamente la situación y tomar acciones adecuadas. Esto demuestra cómo la ciencia de datos puede ser utilizada para mejorar la usabilidad y efectividad de los sistemas de monitoreo.

### Técnicas predictivas para series temporales.

Según Hyndman y Athanasopoulos (2018), las series temporales son secuencias de datos ordenados en el tiempo, y su análisis implica la identificación de patrones como tendencias, estacionalidad y ciclos, para luego utilizarlos en la predicción de valores futuros. Las técnicas predictivas para series temporales son métodos estadísticos y de aprendizaje automático que permiten modelar y predecir el comportamiento futuro de datos que varían en el tiempo. Estas técnicas son fundamentales en aplicaciones donde es necesario anticipar tendencias, patrones o eventos futuros basados en datos históricos.

### Transformers.

Los Transformers son una arquitectura de red neuronal introducida por Vaswani y cols. (2017), cuyo principio central es el mecanismo de atención, que permite procesar secuencias en paralelo en lugar de manera secuencial, como ocurría en las redes recurrentes. Esto posibilita capturar dependencias de largo alcance en secuencias de datos, mejorando el rendimiento en tareas de procesamiento de lenguaje natural, visión y audio. La capacidad de atender de forma dinámica a distintas partes de la secuencia los hace más eficientes y escalables, superando las limitaciones de las RNN y LSTM en problemas de dependencia a largo plazo. Actualmente, constituyen la base de modelos avanzados de IA como BERT y GPT, así como en aplicaciones emergentes de audio y monitoreo acústico

### LSTM (Long Short-Term Memory).

Las redes neuronales recurrentes (RNN) fueron diseñadas para procesar datos secuenciales, pero sufrían el problema del desvanecimiento del gradiente, lo que dificultaba capturar dependencias de largo plazo (Heaton, 2018). Para resolverlo, Hochreiter y Schmidhuber introdujeron las LSTM, que incorporan un mecanismo de memoria a través de una celda con tres puertas: de olvido, entrada y salida. Estas puertas permiten seleccionar qué información descartar, almacenar o transmitir, logrando que el modelo aprenda patrones de secuencia de manera más robusta.

### Autoencoders.

Los autoencoders son redes neuronales no supervisadas diseñadas para aprender una representación comprimida de los datos de entrada (codificación) y luego reconstruirla (decodificación). Su objetivo es minimizar el error de reconstrucción, lo que permite capturar las características más importantes de los datos. En el caso del LSTM Autoencoder, se combinan las capacidades de los autoencoders con las LSTM, resultando especialmente útiles para datos secuenciales. Este tipo de modelo también sirve para simplificar los datos y encontrar patrones importantes, lo que ayuda a tareas como predecir valores futuros, comprimir información o mejorar otros modelos de aprendizaje (Malhotra, Vig, Shroff, Agarwal, y cols., 2015).

### Isolation Forest (Bosque de Aislamiento).

Isolation Forest es un método no supervisado de detección de anomalías que aísla observaciones mediante particiones aleatorias en bosques de árboles, donde las instancias atípicas requieren menos divisiones y, por tanto, muestran longitudes de camino esperadas más cortas. Su puntaje de anomalía deriva de la profundidad normalizada de aislamiento y no presupone distribuciones paramétricas ni datos etiquetados, lo que favorece su uso en flujos en línea y alta dimensionalidad (Aggarwal, 2016).

### Redis Pub/Sub.

Según Carlson (2013), Redis ofrece un sistema de publicación-suscripción (Pub/Sub) que permite a los clientes enviar mensajes a canales sin necesidad de conocer a los receptores. Esta arquitectura desacoplada es útil para sistemas distribuidos, notificaciones en tiempo real y flujos de eventos. Los clientes pueden suscribirse a uno o varios canales y recibir mensajes en tiempo real, sin almacenamiento ni persistencia. Carlson (2013) destaca que este mecanismo se caracteriza por su baja latencia y simplicidad: los mensajes no se almacenan, se entregan en mejor esfuerzo y se pierden si no hay suscriptores activos, lo que lo hace idóneo para eventos efímeros y señales transitorias.

## Bases Legales

El desarrollo del Sistema de Monitoreo Acústico se fundamenta en el cumplimiento de las normativas venezolanas vigentes relacionadas con la privacidad, protección de datos y seguridad

informática. A continuación, se detallan los instrumentos legales aplicables:

1. Constitución de la República Bolivariana de Venezuela (1999)
    - Artículo 60: Garantiza el derecho a la protección de la vida privada, intimidad, honor, propia imagen, confidencialidad y reputación.
    - Artículo 28: Establece el derecho a la protección de datos personales.
2. Ley Especial contra Delitos Informáticos (2001)
    - Artículo 6: Prohíbe el acceso no autorizado a sistemas informáticos.
    - Artículo 20: Sanciona la violación de la privacidad mediante la interceptación de comunicaciones.
3. Ley Orgánica de Telecomunicaciones (2010)
    - Artículo 3: Promueve el uso ético de las telecomunicaciones.

El proyecto prioriza la privacidad mediante:

- No almacenamiento de audios: Los sonidos capturados se procesan en tiempo real y se descartan inmediatamente después de su análisis.
- Base de datos local: La información técnica (como patrones de sonido y registros de alertas) se almacena en un servidor interno, sin conexión a internet, cumpliendo con el principio de confidencialidad.
- Transparencia: Los usuarios son informados sobre el funcionamiento del sistema y su finalidad, asegurando consentimiento informado conforme al artículo 60 constitucional.

# Capítulo III. Marco Metodológico

Este capítulo describe el conjunto de estrategias, técnicas y procedimientos empleados para el desarrollo del Sistema de Monitoreo Acústico para Identificar Sonidos y Generar Alertas de Emergencia. En él se detalla tanto el enfoque investigativo como las herramientas utilizadas para la recolección de datos y, fundamentalmente, la metodología de desarrollo adoptada, basada en el modelo espiral.

## Tipo de Investigación

El presente trabajo constituye una investigación de tipo proyectiva. De acuerdo con Hurtado (2000), la investigación proyectiva tiene como objetivo diseñar o crear propuestas dirigidas a resolver determinadas situaciones. Los proyectos de arquitectura e ingeniería, el diseño de maquinarias, la creación de programas de intervención social, el diseño de programas de estudio, los inventos, la elaboración de programas informáticos, entre otros, siempre que estén sustentados en un proceso de investigación, son ejemplos de investigación proyectiva. Este tipo de investigación potencia el desarrollo tecnológico (p. 325).

Basado en esta definición, el desarrollo de un sistema de monitoreo acústico para identificar sonidos y generar alertas de emergencia se adapta a este tipo de investigación. El sistema propuesto en este trabajo se fundamenta en un proceso de investigación que incluye la revisión documental de conceptos como edge computing, el uso de modelos pre-entrenados para la clasificación de audio y el análisis de secuencias temporales para la detección de anomalías, así como el diseño e implementación de un prototipo funcional. Esto cumple con los requisitos de una investigación proyectiva, ya que no solo se analiza y diagnostica el problema, sino que también se desarrolla una solución tecnológica aplicable en entornos reales.

## Técnicas e Instrumentos de Recolección de Datos

Según Hernández, Fernández, y Baptista (2014), las técnicas de recolección de datos son procedimientos sistemáticos que permiten obtener información relevante para la investigación, mientras que los instrumentos son las herramientas específicas utilizadas para registrar dicha información.

En este estudio, se emplearon técnicas tanto documentales como experimentales. La técnica

documental se utilizó para recopilar información teórica y antecedentes relacionados con los temas de la investigación. Para ello, se realizó una revisión bibliográfica de fuentes como artículos científicos, tesis y normativas legales, con el objetivo de construir el marco teórico que sustenta el proyecto. Esta técnica permitió fundamentar el diseño del sistema y justificar las decisiones técnicas adoptadas. Por otro lado, la técnica experimental se aplicó durante la fase de implementación y validación del sistema. Para ello, se diseñaron experimentos controlados en los que se capturaron y analizaron sonidos ambientales utilizando micrófonos y dispositivos Raspberry Pi. Los instrumentos utilizados en esta fase incluyeron software de procesamiento, así como protocolos de registro de datos para documentar los resultados de las pruebas. Como señala Tamayo (2004), "la recolección de datos debe ser precisa y sistemática para garantizar que la información obtenida sea válida y confiable" (p. 145). En este sentido, se implementaron protocolos estandarizados para la captura y clasificación de sonidos, asegurando que los datos fueran consistentes y representativos de los escenarios reales en los que se espera que opere el sistema.

## Metodología de Desarrollo Utilizada

Dado que al inicio del proyecto no se contaba con una definición completa de los requisitos, se necesitaba una metodología que fuera flexible y evolutiva. La elección del modelo de desarrollo se basó en la necesidad de gestionar activamente los riesgos técnicos, permitir una mejora continua a través de iteraciones y adaptar el sistema a los desafíos que surgieran durante la implementación.

Esto implicaba ciclos de retroalimentación constantes, validación continua y refinamiento iterativo del sistema, características esenciales para alcanzar una solución en un contexto tan dinámico. La elección de una metodología de desarrollo adaptable permitió no solo enfrentar la incertidumbre técnica, sino también ajustar oportunamente el diseño según los aprendizajes obtenidos en cada etapa del proceso. Con esto en mente comparamos las distintas metodologías basándonos en esas necesidades en la Tabla 1.

Aunque metodologías ágiles como Scrum también cumplen con estos criterios, se optó por el modelo Espiral debido a su énfasis explícito en el 'Análisis de Riesgos' como una fase formal en cada ciclo. Dada la incertidumbre técnica del proyecto, esta característica fue considerada fundamental para mitigar de manera proactiva cualquier desafío técnico o de gestión que pudiera surgir.

Según lo propuesto por Boehm (1988), es un enfoque iterativo y flexible que combina elementos del desarrollo incremental y el prototipado, permitiendo gestionar riesgos y adaptar el sistema a medida que avanza el proyecto. Este modelo ha sido ampliamente reconocido por su capacidad para gestionar el

| Metodologías | Flexible | Evolutivo | Gestión de riesgos | Iteración y mejora continua |
|---|---|---|---|---|
| Espiral | Sí | Sí | Sí | Sí |
| Cascada | No | No | No | No |
| Incremental | No | Sí | No | Sí |
| Scrum | Sí | Sí | Si | Sí |

Tabla 1: Cuadro Comparativo de metodologías

desarrollo de sistemas complejos, ya que combina la naturaleza iterativa del prototipado con la estructura sistemática del modelo en cascada (Pressman, 2010, p. 39). En cada iteración, se pueden realizar ajustes en el plan del proyecto, permitiendo adaptar el software a las necesidades emergentes y reducir riesgos antes de que se conviertan en problemas críticos (Pressman, 2010, p. 40).

Según Boehm (1988), el Modelo Espiral se caracteriza por su estructura cíclica, en la que cada iteración incluye cuatro fases principales:

1. **Planificación.** donde se identifican los objetivos del proyecto, los requisitos principales y las restricciones que pueden influir en su desarrollo (Boehm, 1988). Se consideran alternativas y estrategias para alcanzar estos objetivos.

2. **Análisis de riesgos.** Se identifican y evalúan los riesgos potenciales. Se definen acciones para reducir los riesgos identificados y se evalúan alternativas existentes partiendo de prototipos, simulaciones y softwares de análisis. En este ciclo, existen varios prototipos como plantillas de diseño o componentes funcionales.

3. **Desarrollo.** Una vez que los riesgos han sido evaluados, se llevan a cabo las actividades de ingeniería de software. Esta fase incluye el diseño detallado, la codificación del sistema y las pruebas del producto, desarrollando el incremento correspondiente a la iteración actual (Boehm, 1988).

4. **Evaluación.** se revisan los productos obtenidos en la iteración actual. Esta revisión implica la participación de los interesados para verificar que el sistema cumple con los requisitos establecidos y permite realizar ajustes para la siguiente iteración. Esta fase es clave para garantizar la mejora continua del software y la adaptación a nuevas necesidades (Pressman, 2010).

Como señala Boehm (1988), “el Modelo Espiral es particularmente adecuado para proyectos con altos niveles de incertidumbre y requisitos cambiantes, ya que permite incorporar retroalimentación continua y adaptar el diseño en función de los resultados obtenidos”. En este trabajo, esta metodología permitió gestionar los riesgos técnicos y asegurar que el sistema cumpliera con los objetivos planteados.

# Capítulo IV. Desarrollo y Resultados

En este capítulo se detalla el proceso de construcción del sistema de monitoreo acústico, siguiendo los ciclos de la metodología en espiral y su relación con los objetivos específicos. A continuación, se presentan las actividades y resultados obtenidos en cada iteración del proyecto.

## Procedimiento Metodológico

### Análisis del sistema de monitoreo acústico para generar alertas en casos de emergencia.

La definición de la metodología y los requerimientos del sistema se basó en un análisis de literatura existente, estudios previos, noticias de incidentes y la identificación de sus potenciales usuarios. Se determinó que el sistema está dirigido a cualquier individuo que, por un evento súbito como una caída o una agresión, se encuentre en una situación de vulnerabilidad que le impida solicitar ayuda. El análisis se enfocó en perfiles prioritarios con mayor exposición a estos riesgos, como adultos mayores que viven solos, personas con condiciones médicas que limitan su movilidad y potenciales víctimas de robos o violencia doméstica. La característica común de estos grupos es la posible incapacidad de interactuar con un dispositivo durante la emergencia, lo cual estableció el funcionamiento autónomo y pasivo como un requisito técnico clave. El sistema debe, por tanto, detectar la anomalía e iniciar una alerta sin depender de la acción del usuario, justificando así la elección de un monitoreo acústico continuo. Este requisito define el desafío técnico central el cómo lograr que el sistema interprete el entorno sonoro de manera inteligente. Para abordar dicho desafío, se realizó una revisión de artículos científicos y casos de estudio sobre el análisis de audio y la detección de anomalías. La investigación comenzó explorando la interpretación del audio crudo, donde se identificó que el uso de modelos de clasificación pre-entrenados, como la arquitectura YAMNet, era el punto de partida más efectivo para convertir el sonido ambiental en una secuencia de eventos etiquetados. Una vez resuelto “el qué” (la clasificación), el análisis se centró en cuándo un patrón es “anómalo”. Se evaluaron diferentes enfoques para analizar dicha secuencia, desde cadenas de Markov para cambios de estado simples hasta métodos no supervisados como Isolation Forest y

arquitecturas de aprendizaje profundo como las redes LSTM. Este recorrido llevó a la conclusión de que la estrategia más robusta era descomponer el problema, primero usar un clasificador de alto nivel para traducir el sonido y segundo, aplicar modelos de detección de anomalías sobre esa secuencia de datos estructurados para analizar el comportamiento.

### *Revisión de estudios previos y fundamentos teóricos.*

Se revisaron diversos estudios previos sobre el procesamiento de señales de audio y la clasificación de eventos sonoros anómalos. Entre ellos, destacan los trabajos de Malmberg y Radszuweit (2021), quienes abordaron la implementación de modelos de aprendizaje automático en dispositivos edge para la clasificación de audio en tiempo real. Su investigación se centró en el uso del modelo YAMNet, reentrenado para detectar eventos acústicos como disparos, rotura de cristales y habla humana. Este estudio demostró que, aunque existe una ligera pérdida de precisión al utilizar versiones ligeras como TensorFlow Lite, los resultados son comparables a los de la versión completa, lo que abre la posibilidad de trabajar con dispositivos de bajo consumo y recursos limitados, como la Raspberry Pi o el ESP32 (Malmberg y Radszuweit, 2021).

El método para detectar la actividad general en el entorno se inspiró en el trabajo de Torija y cols. (2018). Su investigación, aunque centrada en ambientes urbanos, proporcionó un marco de referencia para analizar los niveles de energía sonora, demostrando que los cambios bruscos en dichos niveles son indicadores fiables de eventos críticos. Este proyecto extiende ese principio para un análisis dual: no solo se utiliza para detectar “picos de actividad” (un incremento brusco de energía que puede sugerir un grito o un golpe), sino también para su opuesto, la “ausencia de actividad”. Un período de silencio prolongado e inusual, que representa una caída significativa por debajo del nivel de energía normal, es igualmente un potente indicador de una posible anomalía. Este doble enfoque se fundamenta en la premisa demostrada por Torija y cols. (2018), quienes establecieron que un análisis centrado en los niveles de energía sonora es un método eficaz para diferenciar el ruido de fondo de eventos acústicos críticos.

En cuanto a los fundamentos teóricos, el sistema se apoya en principios de la inteligencia artificial, campo que, según Russell y Norvig (2022), se enfoca en el desarrollo de sistemas capaces de percibir y actuar sobre su entorno. En este proyecto, dicha capacidad de percepción se logra mediante una combinación de modelos de aprendizaje profundo (deep learning). Para la clasificación inicial de los sonidos, se emplean arquitecturas como las redes neuronales convolucionales (CNN). Posteriormente, para el análisis de la secuencia de eventos y la detección de anomalías, se integra una combinación de modelos

no supervisados como Isolation Forest y redes neuronales recurrentes (LSTM). Esta arquitectura de software, que utiliza distintos componentes de IA para sus tareas de percepción y análisis, se fundamenta en los conceptos teóricos establecidos por autores de referencia en el campo (Russell y Norvig, 2022).

### *Selección de hardware y dispositivos Edge.*

Para la implementación del sistema, se realizó una evaluación de hardware con un enfoque en dispositivos de Edge Computing. La elección se centró en plataformas capaces de procesar datos localmente para reducir la latencia y la dependencia de la nube, como lo describen Shi y cols. (2016). En este contexto, se seleccionó la Raspberry Pi como la plataforma de desarrollo, dada su versatilidad para integrar componentes como micrófonos y su amplia conectividad, características que la hacen ideal para aplicaciones de IoT y análisis en tiempo real según Richardson y Wallace (2016).

En la siguiente tabla comparamos las características técnicas y costos de los dispositivos evaluados

### *Limitaciones y desafíos técnicos.*

Durante el análisis inicial, se identificaron varias limitaciones y desafíos técnicos que podrían afectar el desempeño del sistema. Entre ellos, destacan:

1. Limitaciones tecnológicas: El rendimiento del sistema está directamente condicionado por tres factores: la calidad del sensor acústico para una captura de audio fiel, la capacidad de procesamiento del hardware de borde para ejecutar los modelos sin latencia, y la presencia de ruido ambiental, que puede enmascarar o ser confundido con eventos anómalos.
2. Limitaciones de datos: La recolección de suficientes muestras de sonidos para entrenar el modelo puede ser limitada, lo que puede comprometer el funcionamiento del sistema en escenarios no previstos. Para abordar este desafío, se consideró la posibilidad de utilizar datasets públicos y el uso de modelos preentrenados.
3. Privacidad y seguridad: El desafío ético y técnico de monitorear un entorno privado. Se abordó estableciendo como requisito fundamental el procesamiento 100 % local (Edge AI), sin almacenar ni transmitir grabaciones de audio, para garantizar la confidencialidad del usuario.

### *Requerimientos Funcionales.*

1. El sistema debe capturar el audio ambiental de forma continua a través de los micrófonos designados.
2. El sistema debe procesar el audio capturado para clasificarlo en eventos sonoros predefinidos (ej. voz, silencio, golpe, etc.) utilizando sus modelos de IA.
3. El sistema debe aprender y mantener un perfil de la actividad acústica “normal” del entorno, basado en la secuencia y horario de los sonidos clasificados.
4. El sistema debe comparar la actividad en tiempo real con el perfil de normalidad para detectar patrones anómalos.
5. El sistema debe realizar una consulta verbal al usuario (ej. “¿Está todo bien?”) cuando se detecte una anomalía.
6. Permitir al usuario responder que está bien en caso de que haya un falso positivo o se presente una posible situación de emergencia.
7. El sistema debe enviar notificaciones de emergencia a una lista de contactos predefinidos si una anomalía es crítica o si no se recibe respuesta a la consulta de verificación.

### *Requerimientos no funcionales.*

1. Garantizar la seguridad y privacidad de los datos al no almacenar grabaciones de audio ni datos personales sensibles.
2. Ser escalable para adaptarse a diferentes entornos y necesidades, permitiendo la integración de nuevos micrófonos o dispositivos de procesamiento según sea necesario.
3. Mantener una alta disponibilidad para estar operativo las 24 horas del día, los 7 días de la semana, garantizando la detección continua de eventos sonoros críticos.
4. Mejorar el uso de recursos de hardware, como la memoria y el procesamiento, para funcionar en dispositivos de bajo consumo, como la Raspberry Pi.
5. Contar con una interfaz de usuario intuitiva que sea fácil de usar tanto para usuarios técnicos como no técnicos.

6. Ser robusto y confiable para funcionar en entornos ruidosos y bajo condiciones variables, manteniendo una alta precisión en la detección de eventos sonoros.
7. Facilitar el mantenimiento y actualización mediante un diseño modular que permita actualizaciones de software y hardware sin interrumpir su funcionamiento.

### *Selección de Tecnologías.*

Una vez definidos los requerimientos, se procedió a la selección de las tecnologías para cada capa del sistema. La elección se basó en criterios como el rendimiento en entornos de borde (Edge), la disponibilidad de código abierto, el costo y la compatibilidad entre componentes.

#### *Captura de Audio (Hardware).*

1. **Micrófonos:** Para garantizar una captura de audio clara, se seleccionaron micrófonos con alta sensibilidad. Como señalan Torija y cols. (2018), la calidad del sensor acústico es un factor determinante para la precisión en la detección de eventos.
2. **Plataforma de Procesamiento (Raspberry Pi):** Se eligió la Raspberry Pi como el dispositivo de borde principal debido a su bajo costo, su comunidad de soporte y su capacidad para ejecutar los modelos de IA. Su versatilidad y opciones de conectividad la hacen ideal para prototipar proyectos de IoT y análisis en tiempo real (Richardson y Wallace, 2016).

#### *Procesamiento y Lógica de Detección (Software).*

1. Framework de IA (TensorFlow Lite): Para la ejecución de los modelos en el dispositivo, se optó por TensorFlow Lite. Esta es una versión ligera de TensorFlow diseñada específicamente para hardware con recursos limitados, permitiendo una inferencia rápida y eficiente (Goldsborough, 2016a).
2. Modelo de Clasificación de Sonido (YAMNet): Se seleccionó YAMNet como el clasificador de audio base. Al ser un modelo pre-entrenado por Google en un dataset masivo (AudioSet), proporciona una base robusta para identificar una amplia gama de eventos acústicos generales sin necesidad de un entrenamiento desde cero.

3. Modelos de Detección de Anomalías (Isolation Forest y LSTM): Para el análisis del comportamiento, se eligió una combinación de Isolation Forest, por su capacidad para aislar anomalías puntuales, y un Autoencoder LSTM, por su capacidad para identificar patrones anómalos en secuencias temporales.

4. Modelo de Detección de Palabras Clave (Vosk): Para la funcionalidad de activación por voz, se seleccionó la librería Vosk, un toolkit de reconocimiento de voz de código abierto que puede operar completamente offline en la Raspberry Pi, garantizando la privacidad

*Capa de Notificación (Backend).*

1. Framework del Servidor (NestJS): Para el sistema de envío de notificaciones, se seleccionó NestJS. Al ser un framework de Node.js escalable y basado en TypeScript, facilita la creación de un backend mantenible y modular, capaz de gestionar las alertas de forma segura (Sabo, 2020).

## Diseño del sistema de monitoreo acústico para generar alertas en casos de emergencia.

### *Diseño de la arquitectura del sistema.*

La arquitectura implementada se fundamenta en el modelo en capas, donde cada nivel cumple funciones específicas: captura de datos, clasificación de audio, detección de anomalías, gestión de eventos y almacenamiento. Estas capas no se ejecutan como módulos monolíticos, sino como procesos independientes que se comunican mediante un esquema pub/sub interno con Redis, lo que introduce un componente distribuido dentro del propio dispositivo. Según Tanenbaum y Van Steen (2007), este tipo de combinaciones puede entenderse como una arquitectura híbrida, al integrar un estilo cliente-servidor con mecanismos de colaboración descentralizados. Además, el uso de un Raspberry Pi como nodo principal aproxima la solución al concepto de servidores al borde, en los cuales el procesamiento ocurre cerca de la fuente de datos para reducir tiempos de respuesta y la transmisión de información irrelevante. De esta manera, la propuesta logra unir la claridad del enfoque en capas con la flexibilidad de un sistema distribuido basado en microservicios locales.

1. Capa de captación de audio: Esta capa corresponde al nivel más cercano al hardware y tiene como función principal la adquisición de datos acústicos mediante los micrófonos conectados al dispositivo

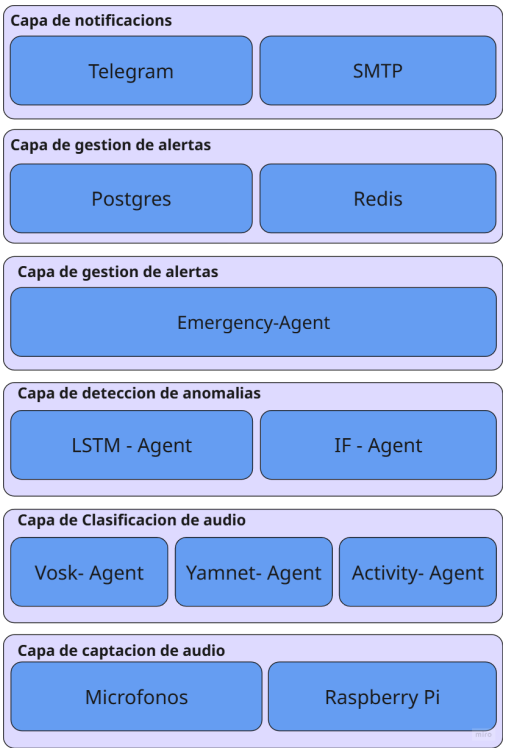


Figura 1: Arquitectura del sistema de monitoreo acústico para generar alertas en casos de emergencia

Raspberry Pi. Se encarga de gestionar la interacción con el sistema operativo para acceder a las señales de audio en tiempo real y ponerlas a disposición del sistema a través de canales de comunicación internos. Su objetivo es abstraer la complejidad del hardware y garantizar que los datos se entreguen de forma continua a las capas superiores.

2. Capa de clasificación de audio: Se encarga de analizar las señales acústicas para identificar patrones relevantes. En este nivel, múltiples agentes especializados procesan el flujo de audio con el fin de determinar la presencia de actividad, reconocer palabras clave y clasificar eventos característicos (como golpes, vidrios rotos o gritos). Los resultados de este procesamiento se estructuran en mensajes que representan descripciones semánticas de la señal, reduciendo el volumen de datos y facilitando la interpretación en etapas posteriores.

3. Capa de detección de anomalías: En esta capa se aplican modelos de aprendizaje automático previamente entrenados (como Isolation Forest y LSTM) para diferenciar entre eventos normales y anormales. El sistema utiliza los datos clasificados por la capa anterior como entrada, evaluando si el comportamiento acústico observado corresponde a patrones esperados dentro del entorno o si, por el contrario, representa una desviación significativa. Esta capa constituye el núcleo analítico del

sistema, ya que transforma datos objetivos en juicios de normalidad o anomalía.

4. Capa de gestión de alertas: La función de esta capa es decidir cuáles de las anomalías detectadas deben escalarse como alertas reales. Para ello, aplica un conjunto de reglas predefinidas que consideran factores contextuales, como los horarios (por ejemplo, diferenciar entre actividad normal en horas diurnas y actividad sospechosa durante la madrugada) o la naturaleza del evento (gritos de auxilio, sonidos de ruptura). Además, esta capa gestiona la interacción con el usuario mediante consultas verbales para validar la situación antes de enviar notificaciones externas. Su propósito es minimizar los falsos positivos y asegurar que solo las situaciones críticas generen una respuesta.

5. Capa de gestión de datos: Esta capa centraliza la persistencia y organización de la información recolectada y procesada por el sistema. Su propósito es almacenar los registros de eventos, tanto normales como anómalos, en estructuras que permitan su posterior análisis. Asimismo, prepara los conjuntos de datos necesarios para el reentrenamiento de los modelos de detección de anomalías, asegurando que el sistema pueda mejorar progresivamente su desempeño y adaptarse a nuevas condiciones acústicas en el entorno.

6. Capa de notificaciones: Tiene como responsabilidad la comunicación con el usuario final o con sistemas externos de monitoreo. Actualmente, este módulo envía alertas validadas a través de servicios de mensajería como Telegram y protocolos de correo electrónico (SMTP). Su diseño es extensible, lo que permite incorporar otros canales de comunicación en el futuro. La existencia de esta capa garantiza que la información crítica llegue de forma oportuna a los responsables de la toma de decisiones.

### *Diseño de la Inteligencia Artificial.*

En esta etapa, se diseñó un sistema de inteligencia artificial que combina modelos avanzados para el análisis de audio. El sistema utiliza YAMNet, un modelo pre-entrenado por Google, para la clasificación precisa de eventos sonoros. Para la detección no supervisada de anomalías en secuencias acústicas, integra autoencoders con LSTM, capaces de aprender patrones temporales, y Isolation Forest, un algoritmo de detección de anomalías. A continuación, se describen en detalle cada uno de estos componentes, su funcionamiento y cómo se integraron en el sistema.

*Yamnet.*

El modelo YAMNet fue seleccionado como la base para la clasificación de eventos sonoros debido a su capacidad comprobada en tareas de reconocimiento de audio. Desarrollado por Google, YAMNet es un modelo preentrenado disponible en formato TFLite, lo que lo hace ideal para su implementación en sistemas embebidos o de bajo consumo computacional. Este modelo fue descargado desde la plataforma Kaggle, en su página oficial. YAMNet está entrenado sobre el conjunto de datos AudioSet, que contiene más de 2 millones de segmentos de audio etiquetados con 521 clases de eventos sonoros, como gritos, ladridos, música, ruidos ambientales, entre otros. Esta amplia cobertura de clases lo convierte en una herramienta versátil para la detección de eventos críticos.

La arquitectura de YAMNet se basa en MobileNet v1, una red neuronal convolucional (CNN) diseñada específicamente para aplicaciones móviles y embebidas. Esta arquitectura permite ejecutar el modelo en dispositivos con limitaciones de hardware sin sacrificar el rendimiento. El modelo acepta como entrada una forma de onda de audio en formato mono, muestreada a 16 kHz, y normalizada en el rango de [-1.0, +1.0]. Internamente, YAMNet divide la forma de onda en ventanas de 0.96 segundos con un salto (hop) de 0.48 segundos, lo que permite analizar el audio en tiempo real. Cada ventana se procesa de manera independiente, generando tres tipos de salidas: scores, que contienen las puntuaciones de predicción para cada una de las 521 clases; embeddings, que representan características de alto nivel extraídas del audio.

Para adaptar YAMNet a la tarea específica de detección de eventos críticos, como gritos, caídas y rotura de cristales, se utilizó la salida de embeddings como entrada para un clasificador adicional. Este enfoque, conocido como transfer learning, permite aprovechar el conocimiento preentrenado de YAMNet y ajustarlo a las necesidades del sistema sin requerir un entrenamiento extensivo desde cero.

*LSTM.*

Las redes neuronales recurrentes (RNN) fueron diseñadas para procesar datos secuenciales, pero se enfrentaban a un desafío importante: la incapacidad de retener información a largo plazo debido al problema del desvanecimiento del gradiente (Heaton, 2018). Para superar esta limitación crítica, Hochreiter y Schmidhuber (1997) desarrollaron las redes LSTM. Esta arquitectura innovadora no solo procesa secuencias de manera efectiva, sino que también introduce un mecanismo de memoria que le permite recordar información relevante por largos periodos, lo que era un obstáculo insalvable para las

RNN tradicionales.

La capacidad distintiva de una red LSTM radica en su celda de memoria (cell state), que actúa como una cinta transportadora que lleva información relevante a lo largo de toda la secuencia de la red. A diferencia de una neurona simple, una celda LSTM está equipada con tres "puertas" reguladoras: la puerta de olvido (forget gate), la de entrada (input gate) y la de salida (output gate). Estas puertas, controladas por funciones sigmoides, deciden qué información debe ser eliminada, qué información nueva debe ser agregada y qué información debe ser utilizada para generar la salida, respectivamente (Heaton, 2018).

La interacción entre estas puertas es lo que le da a las LSTM su poder. La puerta de olvido revisa el estado anterior y la entrada actual para determinar qué datos ya no son necesarios, limpiando la memoria. Luego, la puerta de entrada evalúa la nueva información y decide si es lo suficientemente importante para ser almacenada en la celda de memoria. Finalmente, la puerta de salida toma decisiones sobre la información contenida en la celda de memoria y la entrada actual para generar el valor de salida de la célula, permitiendo que la red utilice el conocimiento aprendido para una tarea específica (Heaton, 2018).

Gracias a este de control de memoria, las LSTM han demostrado ser efectivas en una amplia gama de aplicaciones. En el procesamiento del lenguaje natural (NLP), son fundamentales para la traducción automática y la generación de texto, ya que pueden capturar las dependencias gramaticales a largo plazo. También son cruciales en la detección de anomalías en secuencias de datos, como el audio, donde pueden aprender los patrones de sonido normales para luego identificar desviaciones que indican la presencia de una anomalía. Su éxito se debe a que superan la limitación de las RNN, permitiendo un mejor manejo de las dependencias a largo plazo (Hochreiter y Schmidhuber, 1997).

### *Isolation Forest (Bosque de Aislamiento).*

A diferencia de la mayoría de los algoritmos de detección de anomalías, que intentan construir un perfil complejo de los datos "normales" para luego identificar lo que no encaja, Isolation Forest (IF) se basa en un principio mucho más directo. La premisa fundamental es que las anomalías son, por definición, "pocas y diferentes", lo que las hace más susceptibles a ser aisladas que los puntos de datos normales (Liu, Ting, y Zhou, 2012).

El algoritmo funciona construyendo un conjunto de árboles de decisión aleatorios, conocidos como "árboles de aislamiento" (isolation trees). Para cada árbol, el conjunto de datos se particiona recursivamente seleccionando un atributo al azar y luego un valor de división aleatorio entre el valor máximo y mínimo de

ese atributo. Este proceso de división aleatoria se repite hasta que cada punto de datos queda aislado en un nodo hoja del árbol. La idea central es que los puntos anómalos, al ser diferentes y estar más alejados de las concentraciones de datos normales, requerirán menos particiones para ser aislados (Liu y cols., 2012).

La “anomalía” de un punto se mide calculando la longitud del camino (path length) desde la raíz del árbol hasta el nodo hoja que lo contiene. Los puntos normales, que se encuentran en regiones densas, necesitarán muchos más cortes para ser aislados, resultando en caminos más largos. En contraste, las anomalías, que se encuentran en zonas de baja densidad, serán aisladas con muy pocos cortes, resultando en caminos muy cortos. El puntaje final de anomalía para cada punto se calcula promediando la longitud de su camino a través de todos los árboles del bosque, lo que hace que el resultado sea robusto y fiable (Liu y cols., 2012).

Gracias a su simplicidad y a no depender de cálculos de distancia o densidad, Isolation Forest es extremadamente rápido y tiene un bajo consumo de memoria. Estas características lo convierten en un candidato para la detección de anomalías en tiempo real sobre grandes volúmenes de datos, y es particularmente adecuado para ser implementado en dispositivos de borde (Edge) con recursos computacionales limitados, como es el caso de este proyecto.

## Implementación del sistema de monitoreo acústico para generar alertas en casos de emergencia.

### *Iteraciones.*

1. Prototipo basado en esp32 y detección de intensidad de sonido

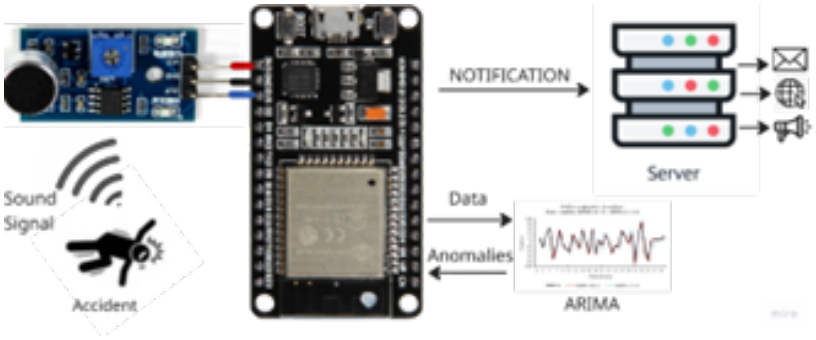


Figura 2: Prototipo basado en ESP32 y detección de intensidad de sonido

Para el primer ciclo de desarrollo, y siguiendo una recomendación del tutor del proyecto, el objetivo

fue validar la hipótesis de que se podían detectar anomalías a través del error de pronóstico de un modelo de series temporales. La estrategia consistía en utilizar un modelo estadístico como ARIMA para predecir el comportamiento sonoro "normal", de modo que cualquier diferencia significativa entre el valor predicho y el valor real observado fuera clasificada como una anomalía. El diseño del prototipo para probar este enfoque se inspiró en metodologías documentadas para sistemas IoT (Luis-García y Gómez, 2024) y consistió en un dispositivo de captura basado en un microcontrolador ESP32, que transmitía el audio a un servidor local (Raspberry Pi 4) para su análisis con el modelo ARIMA.

Sin embargo, durante la fase de evaluación, este enfoque demostró ser fundamentalmente inviable. El problema de fondo no fue solo que el modelo ARIMA tuviera limitaciones, sino que su propio proceso de parametrización resultó ser incompatible con la naturaleza del problema de investigación.

La parametrización de ARIMA busca encontrar una "receta" estadística fija (los valores p,d,q) que describa la estructura de la serie de tiempo. Este enfoque funciona bien cuando el proceso que genera los datos es estable y predecible. No obstante, el entorno acústico de un hogar es todo lo contrario, lo que reveló las siguientes incompatibilidades críticas:

- Naturaleza Dinámica y No Estacionaria: La principal barrera fue que las señales acústicas de un ambiente domestico son altamente no estacionarias. Parametrizar ARIMA es como tomar una foto de las reglas estadísticas del sonido en un momento dado, pero el sonido de un hogar es una "película" que cambia constantemente. La "receta" estadística que describe los sonidos de una mañana activa (conversaciones, desayuno) es completamente diferente a la de una tarde silenciosa o una noche con la televisión encendida. Un modelo con parámetros fijos, por definición, no puede adaptarse a estos cambios de contexto, lo cual viola el supuesto fundamental de estacionariedad que ARIMA necesita para operar de forma fiable.
- Complejidad y No Linealidad: El proceso de parametrización de ARIMA asume que las relaciones en los datos son, en su mayoría, lineales. Sin embargo, los eventos sonoros de una emergencia (un grito, un golpe fuerte, un cristal rompiéndose) son eventos abruptos y no lineales. Un modelo lineal es incapaz de capturar la complejidad de estas interacciones, lo que limita severamente su capacidad para detectar las anomalías más críticas.
- Carencia de Contexto Externo: La "receta" (p,d,q) de ARIMA solo se basa en los valores pasados de la propia señal de audio. El modelo es "ciego" a información contextual crucial, como la hora del día, el día de la semana o si hay personas en casa. Un sonido, como pasos

rápidos, puede ser normal a las 3 PM, pero una anomalía crítica a las 3 AM. Al no poder incorporar variables externas, ARIMA es incapaz de realizar esta distinción.

- Baja Adaptabilidad a Cambios Súbitos: Como consecuencia de su parametrización fija, ARIMA asume que el comportamiento pasado se mantendrá en el futuro. Esto lo hace poco adaptable a rupturas estructurales en el entorno, como el inicio de una reunión social o una tormenta, respondiendo pobremente ante eventos atípicos.

Estas limitaciones nos llevaron a considerar enfoques más flexibles y robustos, capaces de adaptarse a la dinámica cambiante del entorno doméstico y a las características complejas de las señales de audio.

2. Esp32 con micrófonos de señales de audio (Inmp441 Omnidireccional) y prophet

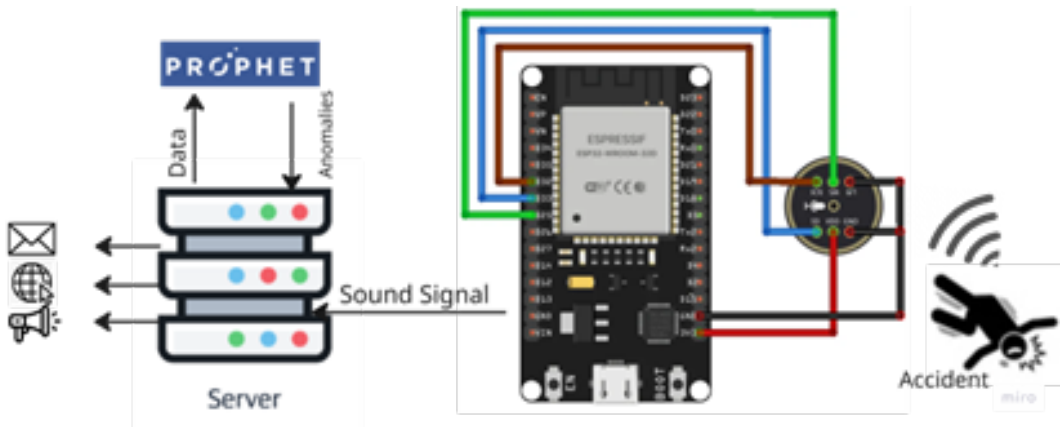


Figura 3: Diseño de prototipo de esp32 con micrófono de señales de audio y prophet

En una segunda iteración, se abandonó la rigidez del modelo ARIMA en favor de un enfoque más flexible. Aunque se mantuvo la arquitectura de hardware basada en el microcontrolador ESP32, se integraron micrófonos omnidireccionales INMP441 para garantizar una captura de señales acústicas de alta fidelidad. A nivel de software, se reemplazó el modelo ARIMA por Prophet, una librería de pronóstico de series temporales desarrollada por Facebook, diseñada específicamente para superar las limitaciones del enfoque anterior.

El modelo Prophet logró ajustarse a las variaciones periódicas del sonido ambiental, como los patrones de ruido diurnos y nocturnos. Sin embargo, durante las pruebas se identificó una limitación crítica: su mecanismo para suavizar datos e identificar tendencias a largo plazo provocaba que se descartaran los eventos de interés. Sonidos de corta duración y alta amplitud, como un golpe, un

grito o la rotura de un cristal, eran filtrados por el modelo y no generaban ninguna alerta, ya que no formaban parte de un patrón estacional o de una tendencia.

Esta observación llevó a la conclusión de que el enfoque de pronóstico de series temporales, independientemente del modelo utilizado, no era el adecuado para el objetivo del proyecto. El problema no era predecir la evolución del nivel de ruido, sino detectar eventos instantáneos y atípicos en el contenido de la señal acústica. Por lo tanto, se redefinió el problema como uno de detección de anomalías en tiempo real, lo que exigió un cambio completo en la estrategia de software.

Para implementar esta nueva estrategia de detección de anomalías, se propuso un enfoque basado en una arquitectura de redes neuronales. La hipótesis de trabajo es que un modelo de este tipo puede aprender a caracterizar el perfil acústico "normal" de un entorno a partir de la extracción de características espectrales del audio. De esta forma, el sistema podría operar en tiempo real identificando cualquier desviación significativa de esa línea base aprendida y marcarla como una posible anomalía.

3. Desarrollo de con dataset propio y modelo personalizado

   Inicialmente, la estrategia se centró en el desarrollo de un modelo de clasificación personalizado a partir de un conjunto de datos propio. El plan consistía en capturar y etiquetar una colección de sonidos de emergencia para entrenar una red neuronal diseñada específicamente para los escenarios de interés del proyecto.

   Sin embargo, a medida que avanzaba el desarrollo, la investigación sobre la robustez de los clasificadores acústicos reveló la verdadera magnitud del desafío. Se hizo evidente que para que un modelo pudiera generalizar y operar de manera fiable, necesitaría ser entrenado con un volumen de datos masivo. La literatura académica confirma que la efectividad de los modelos de aprendizaje profundo para audio depende directamente de la escala y diversidad de los datos, requiriendo a menudo millones de ejemplos para capturar la enorme variabilidad de los sonidos del mundo real (Gemmeke y cols., 2017). La tarea de construir un dataset de esa magnitud era logísticamente inviable para el proyecto.

   Fue precisamente durante esta etapa de investigación que se identificó YAMNet, una arquitectura de clasificación acústica ya pre-entrenada por Google. Este modelo fue entrenado sobre el corpus AudioSet, un extenso conjunto de datos que, como describen Gemmeke y cols. (2017), contiene más de dos millones de clips de audio de 10 segundos etiquetados a partir de una ontología de 527 clases

de eventos sonoros. El descubrimiento de YAMNet ofrecía una solución directa al problema de la adquisición de datos a gran escala que se había identificado como el principal obstáculo.

Ante esta situación, continuar con el desarrollo de un modelo propio habría significado un esfuerzo redundante. Por lo tanto, se tomó la decisión de descartar el enfoque personalizado y pivotar hacia la integración de YAMNet. Esta medida es una aplicación del paradigma de aprendizaje por transferencia (transfer learning), que permite aprovechar el conocimiento de un modelo ya entrenado en un gran dataset para resolver un problema específico, tal como analizan Pons Puig y cols. (2019) en su trabajo sobre el impacto del tamaño del dataset en audio. Esto permitió enfocar los recursos del proyecto en la implementación del sistema final sobre una base tecnológica ya validada.

4. Raspberry Pi, Yamnet, Cadenas de Markov

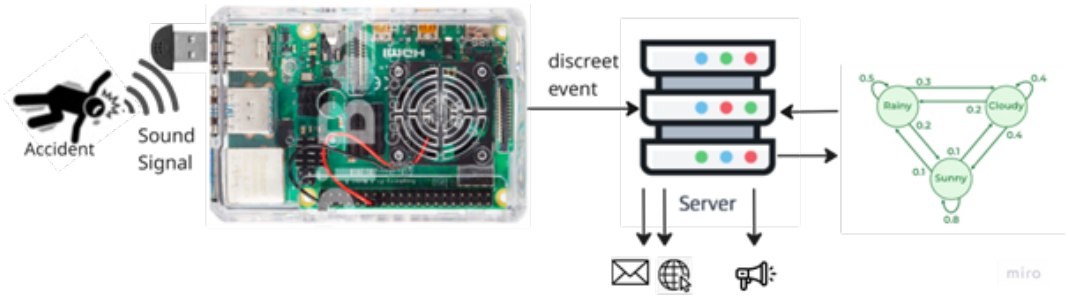


Figura 4: Prototipo con Raspberry Pi, Yamnet y Cadenas de Markov

En esta cuarta iteración del prototipo, se migró la lógica de procesamiento desde el ESP32 hacia una Raspberry Pi, con mayor capacidad computacional, lo que permitió trabajar con señales de audio completas en lugar de limitarse a la intensidad. Se integraron micrófonos digitales de rango completo, capaces de capturar audio a frecuencias más altas y con mayor fidelidad, lo que permitió acceder a un contenido espectral más rico y útil para clasificación detallada.

Con esta infraestructura, se incorporó YAMNet, un modelo de clasificación de audio en tiempo real basado en redes neuronales convolucionales entrenado con el dataset AudioSet de Google. YAMNet permite detectar y clasificar una gran variedad de eventos acústicos o (más de 500 categorías) como gritos, portazos, pasos, objetos cayendo, entre otros. Esta capacidad permitió dar un salto importante en la interpretación contextual del entorno sonoro.

No obstante, Si bien YAMNet ofrecía una detección puntual efectiva de eventos sonoros, surgió la necesidad de dotar a dichos eventos de un significado secuencial y contextual. Para atender esta limitación, se optó por integrar un módulo de modelado probabilístico basado en cadenas de Márkov,

con el fin de analizar la secuencia temporal de eventos y determinar si un patrón resultaba coherente con el comportamiento normal del entorno o, por el contrario, representaba una posible anomalía.

La decisión de emplear cadenas de Márkov estuvo fundamentada en dos aspectos principales: por un lado, en la existencia de estudios previos donde este tipo de modelos han sido utilizados para la detección de anomalías en secuencias temporales, destacando por su capacidad de representar probabilísticamente las transiciones entre estados inmediatos. En particular, investigaciones como la de Boldt, Borg, Ickin, y Gustafsson (2020), muestran cómo las cadenas de Márkov permiten identificar patrones irregulares en secuencias de eventos al modelar las transiciones esperadas y comparar estas con la realidad observada. Por otro lado, la elección también estuvo motivada por la recomendación del profesor de estadistica Omar Castro, quien sugirió explorar este enfoque como un primer acercamiento al modelado secuencial de los eventos detectados.

No obstante, también surgieron limitaciones importantes. Si bien éramos capaces de detectar situaciones potencialmente peligrosas a partir de categorías específicas identificadas por YAMNet, el intento de modelar secuencias completas de eventos mediante cadenas de Márkov resultó ineficaz, ya que se quedaba saltando en ciclos infinitos de cambios de estado, por ejemplo (luego de Habla el siguiente evento mas probable era Habla). Estas cadenas se limitan a representar transiciones entre estados inmediatos, es decir, predicen solo el evento más probable siguiente, sin tener en cuenta el historial completo de eventos previos.

Intentar forzar el modelado de secuencias más largas con este enfoque llevó a comportamientos problemáticos, como ciclos repetitivos sin resolución lógica, o secuencias que carecían de coherencia contextual, especialmente cuando el espacio de eventos posibles crecía. Esta rigidez estructural impidió representar adecuadamente escenarios más complejos o ambiguos, donde la interpretación depende no solo del último evento detectado, sino de una serie de interacciones acústicas encadenadas en el tiempo.

Este hallazgo marcó un punto de inflexión en el desarrollo del sistema, motivando la exploración de enfoques más robustos y dinámicos, capaces de capturar dependencias de largo plazo en secuencias acústicas. En particular, se consideraron modelos como LSTM (Long Short-Term Memory) y transformers , que permiten aprender patrones secuenciales con memoria y contexto más amplio.

5. Raspberry Pi, Yamnet, Isolation Forest y LSTM

   En esta quinta iteración, nos enfocamos en el desarrollo de una solución más robusta para la detección de anomalías, como respuesta a las limitaciones encontradas con las cadenas de Markov.

Para ello, basamos nuestro enfoque en el trabajo de Reis y Serôdio (2025), cuya investigación se centra en arquitecturas de Inteligencia Artificial en el borde (Edge AI) para el monitoreo en tiempo real. Específicamente, su estudio valida el uso de dos tecnologías que adoptamos: Isolation Forest (IF), un modelo para la detección de eventos anomalos basado en arboles, y un Long Short-Term Memory Autoencoder (LSTM), una red neuronal para el análisis de patrones secuenciales complejos.

Para la implementación de estos modelos en nuestro sistema, se replicaron las configuraciones de hiperparámetros validadas en dicho estudio para asegurar el rendimiento en la Raspberry Pi.

En el caso del Isolation Forest, adoptamos los hiperparámetros del estudio, configurando el modelo con 100 árboles ($n_estimators$) y un parámetro de contaminación del 3 %. Este último valor asume una proporción esperada de anomalías en nuestros datos, alineándose con los escenarios simulados en la investigación de referencia.

El segundo modelo implementado es un LSTM Autoencoder, una arquitectura de red neuronal no supervisada especialmente eficaz para la detección de anomalías en datos secuenciales (Malhotra y cols., 2015). Su objetivo es aprender una representación comprimida de los patrones "normales" presentes en los datos de entrenamiento para luego identificar desviaciones significativas. De igual manera, se adoptaron los hiperparámetros recomendados por Reis y Serôdio (2025), que incluyen un tamaño de lote de 32 y 50 épocas de entrenamiento. La arquitectura específica del modelo consta de 3 capas LSTM en el codificador y 3 capas LSTM en el decodificador, con tamaños de capa decrecientes de 32, 16 y 8 unidades respectivamente. y activacion del ReLu para ajustar la tasa de aprendizaje y evitar el sobreajuste. Para el desarrollo y aplicación del modelo, se adoptó un enfoque metodológico inspirado en el trabajo de Reis y Serôdio (2025), el cual se estructuró en un proceso de tres etapas: preparación de los datos, construcción y entrenamiento de la arquitectura, y definición del mecanismo de detección.

*a*) Preparación de los Datos: Para que nuestro modelo LSTM pudiera interpretar los datos de los eventos, fue necesario "traducirlos" a un formato numérico que la red neuronal pudiera procesar. Este paso es crucial, ya que el modelo no entiende de fechas o etiquetas de texto por sí mismo; solo trabaja con números y mientras mas sentido y consistencia tengan esos números, mejor podrá aprender. Por ejemplo, si convertimos los días de la semana a números (domingo=0, lunes=1, ..., sábado=6), el modelo interpretaría que el sábado (6) y el domingo (0) son valores muy distantes, cuando en realidad son adyacentes. Este tipo de malentendido

impide que la red aprenda patrones que ocurren en la transición del fin de semana. Para resolver este problema, el enfoque consiste en representar las variables temporales de una manera que refleje su naturaleza cíclica. En lugar de ver el tiempo como una línea recta, lo tratamos como un círculo, similar a las manecillas de un reloj. Esto se logra mediante funciones matemáticas (seno y coseno) que asignan a cada instante una coordenada única en un círculo. De esta forma: El sábado queda matemáticamente "cerca" del domingo. Las 23:59 quedan justo al lado de las 00:00. Este preprocesamiento es fundamental, pues permite que el modelo LSTM capture patrones de comportamiento continuos y lógicos, como los que ocurren durante la madrugada o en el cambio de un día para otro. Convirtiendo Categorías en Señales Claras De manera similar, el modelo no puede procesar etiquetas de texto como "uso de microondas" o "sensor de movimiento". Un error común sería asignarles un número a cada una (ej: microondas=1, sensor=2), ya que esto crearía una falsa relación matemática (como si el "sensor" fuera el doble del "microondas"). La solución es una técnica llamada One-Hot Encoding. Funciona como un panel de interruptores, donde cada evento posible tiene su propio interruptor. Cuando ocurre un evento, su interruptor se "enciende" (toma el valor de 1) mientras que todos los demás permanecen "apagados" (con valor de 0). De esta forma, el modelo recibe una señal clara y sin ambigüedades sobre qué evento específico sucedió, sin crear jerarquías o relaciones numéricas que no existen. Finalmente, el verdadero potencial de un modelo LSTM es su capacidad para entender el contexto. Un evento aislado, como "se enciende una luz", no dice mucho. Pero si ocurre dentro de una "historia" o secuencia como "sensor de movimiento activado $\rightarrow$ se abre la puerta $\rightarrow$ se enciende la luz", el patrón es mucho más revelador. Por esta razón, en lugar de analizar los eventos de forma individual, los agrupamos en secuencias superpuestas de una longitud fija (por ejemplo, 10 eventos consecutivos). Es como pasar de ver fotografías individuales a ver pequeños videoclips. Al entregarle los datos en este formato, el modelo LSTM no solo aprende "qué" pasó, sino "en qué orden" y "en relación con qué" otros eventos ocurrieron, permitiéndole así aprender los patrones complejos que definen un comportamiento normal. Con los datos ya "traducidos" a este formato numérico, cíclico y secuencial, el modelo está listo para comenzar su fase de entrenamiento.

*b*) Arquitectura del Modelo: Utilizamos un tipo especial de red neuronal llamado Autoencoder LSTM. La idea es bastante intuitiva. En lugar de predecir un valor futuro, el objetivo de un autoencoder es aprender a reproducir su propia entrada. Podemos imaginarlo como intentar

crear una figura circular con un lápiz y un compas, normalmente lo hariamos con el compas y el resultado es bastante predecible, pero al hacerlo a mano, el resultado va variar. Nuestro modelo funciona igual, se le entrena para ser un experto en reconstruir secuencias de eventos normales. Cuando se encuentra con una secuencia anómala, su reconstrucción es de mala calidad, y ese "error de reconstrucción" es la señal que nos alerta de una anomalía. La arquitectura del autoencoder se divide en dos partes simétricas: El Codificador (El Resumidor): Esta primera mitad de la red toma la secuencia de entrada (nuestro "videoclip" de 10 eventos) y la comprime progresivamente en una representación mucho más pequeña y densa. Es como si leyera un párrafo completo y lo resumiera en una sola idea clave. En nuestro caso, dos capas LSTM reducen la dimensionalidad de 32 a 16 bits, creando esa "idea" o resumen. El Decodificador (El Reconstructor): Esta segunda mitad toma la idea clave generada por el codificador y realiza el proceso inverso: intenta reconstruir la secuencia original de 10 eventos con la mayor fidelidad posible. Actúa como un espejo del codificador, expandiendo la representación de 16 de vuelta a 32 bits. Para mejorar la robustez del modelo y evitar que simplemente "memorice" los datos de entrenamiento, se incluyeron capas de Dropout. Estas capas "apagaban" aleatoriamente algunas neuronas durante el entrenamiento, forzando al modelo a aprender patrones más generales y significativos del comportamiento normal.

*c*) Entrenamiento y Detección: Una vez definida la arquitectura, el modelo entra en la fase de entrenamiento, que es donde aprende el comportamiento "normal". El objetivo del entrenamiento es simple, hacer que el modelo sea bueno reconstruyendo las secuencias de eventos normales y muy malo haciendo lo mismo con las anómalas. Para ello, se le alimenta con una serie constante de datos que representan la normalidad. El modelo procesa los datos en un ciclo de "ensayo y error" intentando reconstruir una secuencia normal, luego compara la secuencia original con su reconstrucción y calcula el "error de reconstrucción" (qué tan diferente es su copia del original), usando una métrica llamada Error Cuadrático Medio (MSE). Luego Ajusta sus parámetros internos para reducir ese error en el siguiente intento, este proceso se repite miles de veces para hacer al modelo más inteligente, durante este proceso se utilizan dos mecanismos de supervisión para evitar que el modelo se sobreentrene o se estanque:

1) EarlyStopping (Parada Temprana): Es un supervisor que detiene el entrenamiento automáticamente si el modelo deja de mejorar. Esto evita el sobreajuste (que el modelo "memorice" en lugar de "aprender") y ahorra tiempo de cómputo.

2) ReduceLROnPlateau (Reducción de Tasa de Aprendizaje): Si el modelo se estanca, este mecanismo reduce la magnitud de los ajustes que realiza, permitiéndole un aprendizaje más fino y preciso para superar el estancamiento.

Luego de que el modelo ha sido entrenado y puede reconstruir consistentemente las secuencias normales con bajo error, está listo para la fase de detección de anomalías. Para ello, calculamos nuestro umbral de decisión basado en la distribución estadística de los errores de reconstrucción obtenidos durante el entrenamiento. Este umbral define el límite entre lo que consideramos un comportamiento normal y una posible anomalía. Este umbral se establece en el percentil 99.95 de los errores de reconstrucción en los datos normales, lo que significa que solo el 0.05 % de las secuencias normales tendrán un error mayor que este valor. Cualquier secuencia que el modelo intente reconstruir y que produzca un error de reconstrucción superior a este umbral será clasificada como una anomalía. Este enfoque estadístico asegura que el modelo sea sensible a desviaciones significativas del comportamiento aprendido, es necesario para reducir las falsas alarmas y garantizar que solo los eventos verdaderamente atípicos sean señalados para una revisión más detallada.

### *Evaluacion de los Modelos.*

En esta etapa de evaluacion, nos centramos por completo en el desempeño de detección del modelo. El propósito es analizar qué tan bien la arquitectura puede distinguir entre patrones normales y anómalos. Para ello, se utilizó un conjunto de datos de prueba independiente, que incluye tanto secuencias normales como una serie de anomalías introducidas artificialmente para simular eventos críticos. Este conjunto no fue utilizado durante el entrenamiento, asegurando así una evaluación objetiva del modelo.

#### *Evaluacion del LSTM Autoencoder.*

Para evaluar el rendimiento del modelo, se empleó un conjunto de datos de prueba, el cual es una porción de los datos que el modelo no observó durante su entrenamiento. Este conjunto contenía tanto secuencias de comportamiento normal como una serie de anomalías que fueron introducidas artificialmente en el dataset para simular eventos atípicos. El proceso consistió en pasar cada una de las 45,841 secuencias de prueba a través del autoencoder ya entrenado y calcular su error de reconstrucción (MSE). Posteriormente, este error fue comparado con un umbral de anomalía definido estadísticamente a partir de la distribución de

errores del conjunto de entrenamiento (específicamente, el percentil 99.95). Cualquier secuencia cuyo error de reconstrucción superara este umbral fue clasificada como una anomalía.

La selección de este percentil (99.95) no fue un valor arbitrario. Para definir el umbral de decisión, se realizó un análisis de la Curva ROC (Receiver Operating Characteristic). Esta herramienta permite evaluar el balance entre la capacidad de detectar anomalías reales (Tasa de Verdaderos Positivos) y la tasa de falsas alarmas (Tasa de Falsos Positivos) para diferentes umbrales. Se probaron varios percentiles y se seleccionó aquel que maximizaba el Área Bajo la Curva (AUC), una métrica que resume la capacidad global del modelo para distinguir entre clases. Como se muestra en la gráfica, el umbral del 99.95 % permitió alcanzar un AUC de 0.792, un valor cercano al objetivo de 0.8, lo que confirma que el umbral elegido posiciona al modelo en un punto de operación con un sólido equilibrio entre sensibilidad y especificidad.

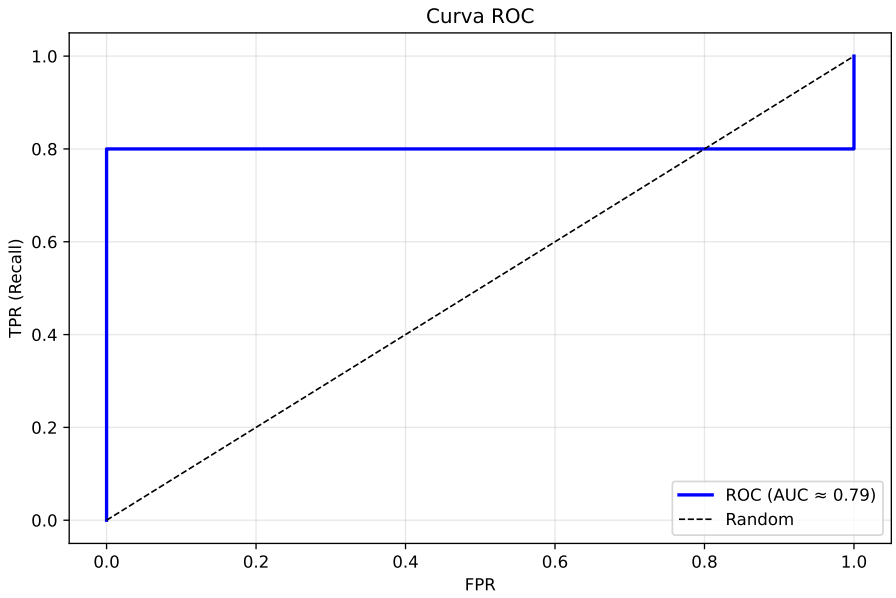


Figura 5: Curva ROC y AUC del LSTM Autoencoder

El resultado principal del proceso de validación se puede observar directamente en el siguiente gráfico, que muestra el momento exacto en que el modelo detecta una anomalía.

En la gráfica de la figura 6, se visualiza el error de reconstrucción para cada secuencia del conjunto de prueba. La línea discontinua roja representa el umbral de anomalía (MSE = 0.001646). Los puntos rojos marcan las 21 secuencias cuyo error superó este límite, siendo clasificadas correctamente como anomalías. Este gráfico confirma visualmente que el modelo es efectivo para asignar una puntuación de error mucho más alta a los eventos que se desvían del comportamiento normal aprendido.

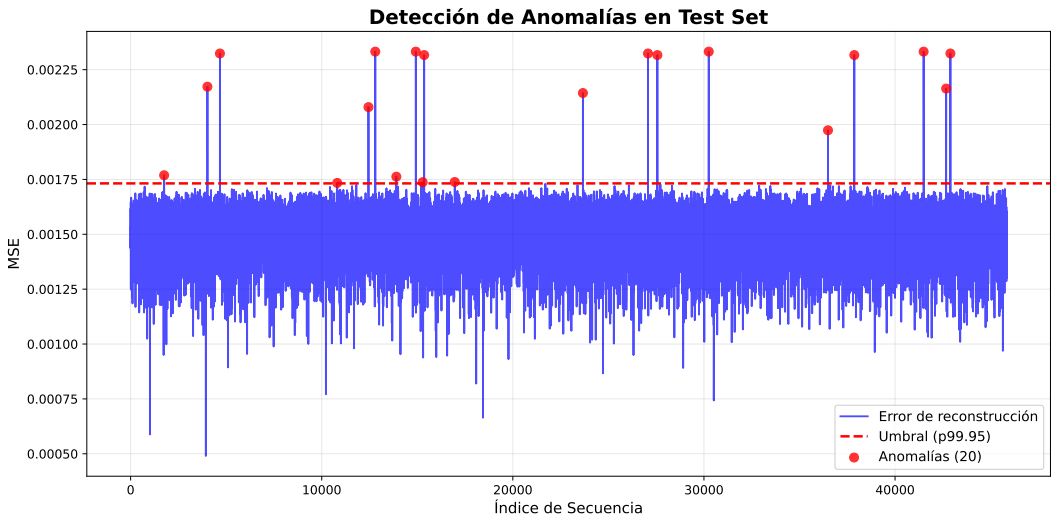


Figura 6: Gráficas de MSE por secuencia para el LSTM Autoencoder.

Para analizar en detalle el rendimiento del modelo, no es suficiente con saber el número total de anomalías detectadas. Es fundamental comprender la naturaleza de sus aciertos y errores, y para ello se utiliza la matriz de confusión. Esta herramienta es una tabla que desglosa los resultados al comparar las predicciones del modelo con la realidad del conjunto de prueba, permitiéndonos visualizar cuatro escenarios clave:

- Verdaderos Positivos (VP): Las anomalías que el modelo identificó correctamente.
- Verdaderos Negativos (VN): Los eventos normales que el modelo clasificó correctamente como normales.
- Falsos Positivos (FP): Eventos normales que el modelo etiquetó erróneamente como anomalías (las "falsas alarmas").
- Falsos Negativos (FN): Las anomalías reales que el modelo no fue capaz de detectar (los errores más críticos).

VP (Verdaderos Positivos): 18

FP (Falsos Positivos): 3

FN (Falsos Negativos): 6

VN (Verdaderos Negativos): 45814

Estas cifras se traducen en las siguientes métricas de rendimiento:

| | Esperado | |
|---|---|---|
| **Real** | **P** | **N** |
| V | 18 | 45814 |
| F | 3 | 6 |

Tabla 2: Matriz de Confusión del LSTM Autoencoder

- Exactitud (Accuracy): proporción de predicciones correctas respecto al total de casos evaluados.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{(18 + 45814)}{(18 + 45814 + 3 + 6)} \approx 0{,}9998 \tag{1}$$

- Tasa de error: proporción de predicciones incorrectas respecto al total.

$$Error = \frac{FP + FN}{TP + TN + FP + FN} = \frac{(3 + 6)}{(18 + 45814 + 3 + 6)} \approx 0{,}0002 \tag{2}$$

- Sensibilidad (Recall o Verdaderos Positivos): proporción de eventos de anomalos correctamente detectados.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- Precisión: La proporción de casos realmente positivos entre todos los casos que el modelo predijo

como positivos.

$$Precision = \frac{TP}{TP + FP} = \frac{18}{18 + 3} \approx 0{,}8571 \tag{4}$$

- Especificidad: proporción de eventos no críticos correctamente descartados.

$$Specificity = \frac{TN}{TN + FP} = \frac{45814}{45814 + 3} \approx 0{,}9999 \tag{5}$$

- F1 Score: media armónica entre la precisión y la sensibilidad, útil cuando es importante equilibrar ambas.

$$F1_{score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0{,}8571 * 0{,}75}{0{,}8571 + 0{,}75} \approx 0{,}8 \tag{6}$$

El análisis de las métricas de rendimiento cuantifica el desempeño del modelo. La exactitud general es del 99.98 %, aunque este valor es menos indicativo en conjuntos de datos con clases desbalanceadas. Un análisis más específico muestra una Sensibilidad (Recall) del 75.0 %, lo que indica que el modelo identifica a tres de cada cuatro anomalías reales. A su vez, la Precisión es del 85.7 %, lo que significa que sus alertas positivas son correctas en esa proporción. La puntuación F1, como media armónica de ambas, se sitúa en 80.0 %, reflejando el balance entre la capacidad de detección y la fiabilidad de sus predicciones.

*Evaluacion del Isolation Forest.*

Para la evaluación del modelo Isolation Forest (IF), una técnica no supervisada que opera de manera fundamentalmente distinta al autoencoder. En lugar de medir un error de reconstrucción, este modelo asigna una "puntuación de anomalía" a cada evento basándose en qué tan fácil es aislarlo del resto de los datos. Los eventos que son más fáciles de separar (requieren menos divisiones en los árboles de decisión) se consideran más anómalos.

Funciona construyendo múltiples árboles de decisión (en este caso, 100) estos arboles se construyen al dividir aleatoriamente los datos en subconjuntos. Funciona bajo la premisa de que las anomalías son puntos de datos que se encuentran en regiones menos densas del espacio de características, por lo que son más fáciles de aislar. Cada evento recibe una puntuación basada en la profundidad promedio a la que aparece en los árboles; cuanto más cerca esté la puntuación de 1, más anómalo es el evento.

El proceso de evaluación se aplicó sobre el mismo conjunto de datos, compuesto por 100,809 registros, de los cuales 5,041 (5 %) son anomalías reales. A diferencia del LSTM Autoencoder, el umbral de decisión en el Isolation Forest no se calcula estadísticamente a posteriori, sino que se define a priori mediante el hiperparámetro contamination. Este valor se estableció en 0.05 (5 %), una práctica estándar para configurar este algoritmo. Cualquier evento cuya puntuación de anomalía superara el umbral derivado de esta configuración fue clasificado como anómalo.

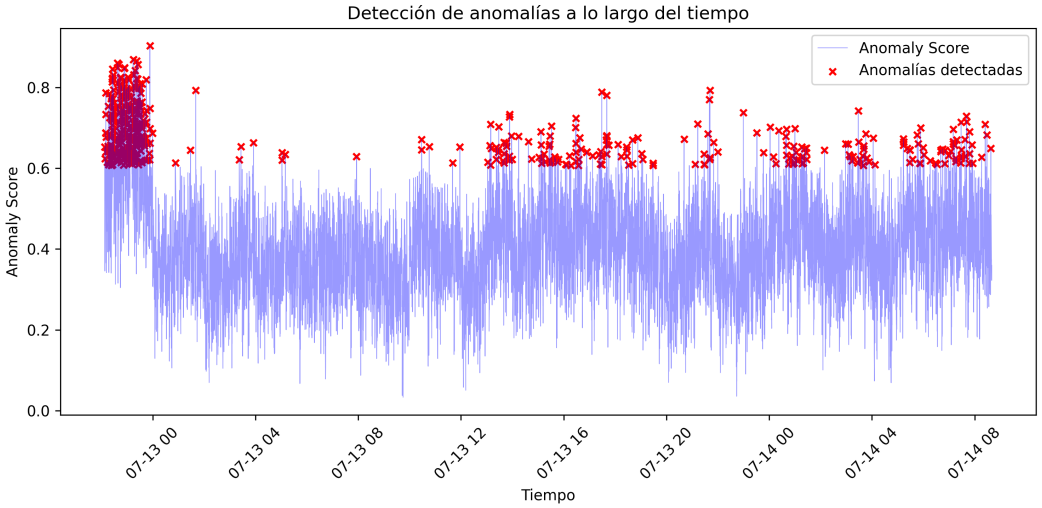


Figura 7: Gráficas de puntuación de anomalía por evento para el Isolation Forest.

- Análisis de la Detección

El análisis de los resultados del Isolation Forest revela un rendimiento significativamente inferior al del LSTM Autoencoder. El modelo identificó un total de 5,041 eventos como anómalos. Sin embargo, un desglose más profundo a través de la matriz de confusión es necesario para entender la verdadera efectividad de estas detecciones.

Para analizar en detalle el rendimiento del modelo, se utiliza la matriz de confusión. Esta herramienta desglosa los resultados al comparar las predicciones del modelo con la realidad del conjunto de prueba, permitiéndonos visualizar cuatro escenarios clave:

| | **Esperado** | |
|---|---|---|
| **Real** | **P** | **N** |
| V | 2877 | 87737 |
| F | 2164 | 8031 |

Tabla 3: Matriz de Confusión del Isolation Forest

VP (Verdaderos Positivos): 2,877

FP (Falsos Positivos): 2,164

FN (Falsos Negativos): 8,031

VN (Verdaderos Negativos): 87,737

- Métricas de Rendimiento

$$Accuracy = \frac{(2877 + 87737)}{(2877 + 87737 + 2164 + 8031)} \approx 0{,}8989 \tag{7}$$

$$Error = \frac{(2164 + 8031)}{(2877 + 87737 + 2164 + 8031)} \approx 0{,}101 \tag{8}$$

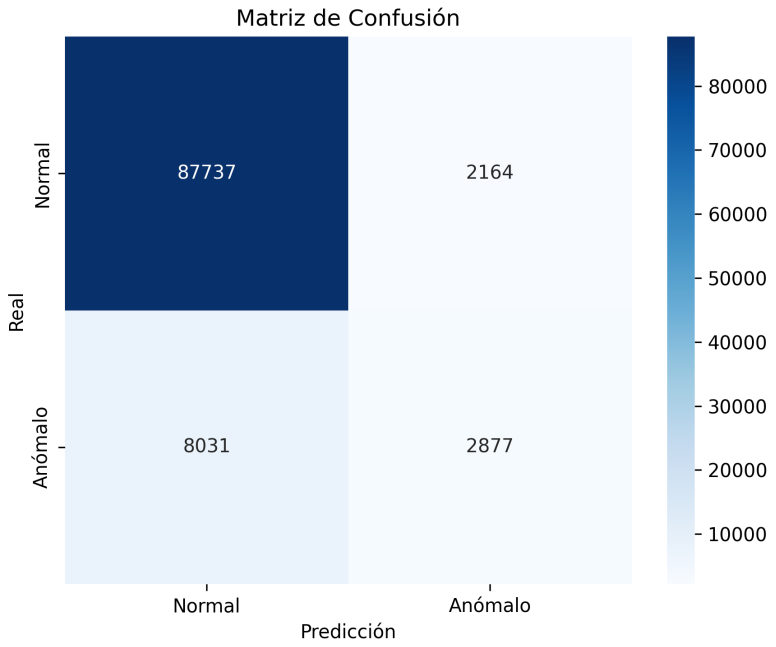


Figura 8: Matriz de Confusión del Isolation Forest

$$Recall = \frac{2877}{2877 + 8031} \approx 0{,}2638 \tag{9}$$

$$Precision = \frac{2877}{2877 + 2164} \approx 0{,}5707 \tag{10}$$

$$Specificity = \frac{87737}{87737 + 2164} \approx 0{,}9759 \tag{11}$$

$$F1_{score} = 2 \cdot \frac{Precision * Recall}{Precision + Recall} \tag{12}$$

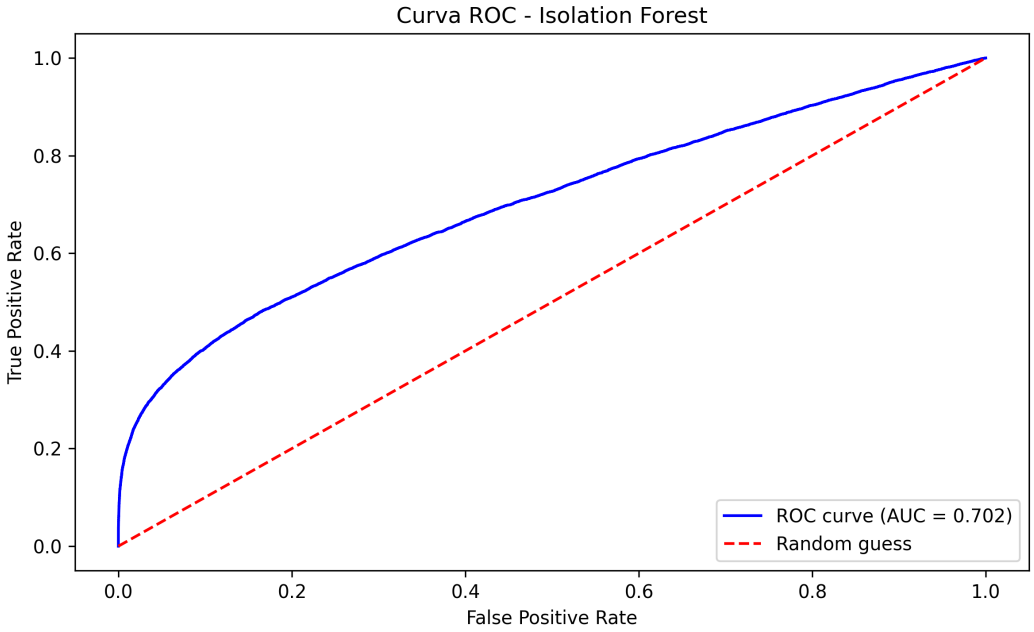


Figura 9: Curva ROC y AUC del Isolation Forest.

El análisis de las métricas de rendimiento cuantifica el desempeño del modelo. La exactitud general del 89.9 % es un valor engañoso, inflado por la gran cantidad de eventos normales correctamente identificados. Las métricas más importantes revelan un rendimiento deficiente: una Sensibilidad (Recall) de solo el 26.4 %, lo que indica que el modelo no detectó a más de 3 de cada 4 anomalías reales. A su vez, la Precisión es del 57.1 %, lo que significa que casi la mitad de sus alertas son falsas alarmas. La puntuación

F1 resultante, del 36.1 %, confirma que el modelo, en su configuración actual, carece de la capacidad necesaria para una detección fiable de anomalías en este contexto.

### Validación del sistema de monitoreo acústico para generar alertas en casos de emergencia.

Para validar el funcionamiento del sistema de monitoreo acústico y su capacidad de generar alertas de emergencia, se procedió con un proceso de verificación. Este proceso se basó en el contraste directo del comportamiento del prototipo con los requerimientos operativos definidos, detallados en la Tabla 4.

La validación se concretó a través de la evaluación de los modelos de clasificación acústica en un entorno controlado mediante el uso de datos sintéticos. Se simularon escenarios acústicos, abarcando tanto condiciones de operación normal como eventos anómalos. El propósito fue verificar de manera objetiva si la salida del sistema, la clasificación del evento y la consecuente activación de la alerta correspondían con el resultado esperado, asegurando la correspondencia con su especificación.

Los resultados obtenidos de esta verificación se consolidan en la Tabla 4.

# Capítulo V. Conclusiones y Recomendaciones

Como resultado de esta investigación, se desarrolló un sistema de monitoreo acústico basado en inteligencia artificial, diseñado para identificar sonidos relevantes en entornos domésticos y generar alertas oportunas en casos de emergencia. Este sistema integra tecnologías de procesamiento en el borde (Edge Computing) y modelos de clasificación de audio preentrenados, adaptándose a las condiciones variables de los espacios habitacionales, sin comprometer la privacidad de los usuarios. A continuación, se exponen las principales conclusiones y recomendaciones derivadas del estudio.

## Conclusiones

La revisión teórica y el análisis de enfoques previos permitieron establecer las bases conceptuales y tecnológicas para diseñar un sistema capaz de detectar patrones acústicos asociados a situaciones de riesgo. La integración de modelos como YAMNet y Vosk demostró que es posible realizar detección y clasificación de eventos sonoros en tiempo real desde dispositivos de bajo consumo, como Raspberry Pi.

La arquitectura del sistema, basada en una red distribuida de micrófonos conectados a nodos de procesamiento local, permitió generar un perfil acústico del entorno, diferenciando entre sonidos rutinarios y eventos anómalos. La incorporación de cadenas de Márkov como modelo probabilístico inicial brindó un nivel de razonamiento temporal básico, aunque sus limitaciones estructurales derivaron en la exploración de arquitecturas más avanzadas, como redes LSTM, capaces de modelar secuencias acústicas de forma más coherente.

Durante la validación, el sistema mostró una alta tasa de éxito en pruebas con eventos claramente definidos, como gritos, cristales rotos o sonidos fuertes, alcanzando niveles del 100 % de aciertos. Sin embargo, se identificaron desafíos en la detección contextualizada de palabras clave o en sonidos menos estructurados, lo que evidencia la necesidad de seguir mejorando la sensibilidad y especificidad del sistema en escenarios complejos.

El desarrollo adoptó la metodología espiral, lo cual permitió iterar entre diseño, implementación y validación de forma progresiva. Esta estrategia fue clave para ajustar la solución a las restricciones del

entorno real, abordando problemas como la gestión de ruido ambiental, las limitaciones de hardware y los requisitos legales sobre privacidad de datos.

A nivel social y ético, el sistema representa una herramienta accesible y no invasiva, especialmente relevante para personas en situación de vulnerabilidad, como adultos mayores o personas con discapacidad. Al evitar el almacenamiento de datos sensibles y operar en tiempo real, se garantiza el cumplimiento con principios constitucionales de confidencialidad y autonomía.

## Recomendaciones

- Incorporar modelos de aprendizaje secuencial más avanzados: La integración de arquitecturas como transformers adaptados al procesamiento de audio permitiría mejorar la comprensión del contexto acústico y reducir la incidencia de falsos positivos o negativos en escenarios ambiguos.
- Desarrollar interfaces más intuitivas y accesibles: Se sugiere implementar una interfaz de usuario que permita a los usuarios configurar umbrales de alerta, definir eventos críticos personalizados y gestionar contactos de emergencia de forma amigable, incluyendo opciones accesibles para personas mayores.
- Implementar una funcionalidad de alarma inteligente basada en contexto y perfil de actividad: se recomienda el desarrollo de una funcionalidad de alarma inteligente que no solo se active ante eventos sonoros abruptos, sino que también considere el contexto situacional completo del entorno. Esta funcionalidad debería integrar indicadores como la presencia o ausencia de personas en la vivienda, utilizando sensores de movimiento, horarios habituales o incluso rutinas declaradas por el usuario. Igualmente, se sugiere incluir mecanismos que permitan detectar silencios prolongados e inusuales, los cuales podrían indicar caídas, pérdida de conciencia o situaciones de emergencia silenciosas, especialmente en usuarios con movilidad reducida.
- Ampliar los mecanismos de notificación mediante módulos GSM y servicios de notificaciones push: se recomienda diversificar e incrementar la robustez de los canales de envío de alertas mediante la incorporación de módulos GSM (Global System for Mobile Communications), los cuales permitirían el envío de mensajes SMS directamente desde el dispositivo sin necesidad de conexión a internet. Esto resulta especialmente útil en contextos con baja conectividad o en situaciones donde la red local pueda fallar durante una emergencia. De igual forma, se sugiere integrar notificaciones push móviles, a través de plataformas como Firebase Cloud Messaging (FCM), que permitirían alertas más inmediatas, personalizadas y con capacidad de interacción desde teléfonos inteligentes.

- Uso de ensamblajes de modelos: La combinación de múltiples modelos de detección, como CNNs y LSTMs, podría mejorar la precisión y robustez del sistema al aprovechar las fortalezas individuales de cada arquitectura.

- Desarrollo de un agente controlador: Un agente controlador que gestione las alertas de emergencia, priorizando y filtrando las notificaciones para evitar falsas alarmas y garantizar que las alertas críticas se envíen de manera oportuna.

- Sistema de retroalimentación del usuario: Implementar un mecanismo que permita a los usuarios confirmar o descartar alertas, ayudando al sistema a aprender y mejorar su precisión con el tiempo.

- Incorporar el uso de sensores de presencia como el C1001 mmWave Human Detection Sensor de Acconeer, que utiliza tecnología de radar para detectar la presencia y el movimiento de personas en un entorno. Pudiendo detectar caídas, movimientos inusuales o la ausencia de movimiento, ritmo cardíaco y respiración, sin la necesidad de estar fisicamente unido al cuerpo del usuario. Esto podría complementar el sistema de identificación de sonidos proporcionando una capa adicional de validacion de estado del usuario.

## Referencias


Aggarwal, C. C. (2016). An introduction to outlier analysis. En *Outlier analysis* (pp. 1–34). Springer.

Bobadilla Osses, A. (2010). *Cadenas de markov.* Descargado 2025-09-08, de https://repositoriobibliotecas.uv.cl/serveruv/api/core/bitstreams/f4406bb7-5f33-4a41-8333-7d822360a0ed/content

Boehm, B. (1988). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, *11*(4), 14–24.

Boldt, M., Borg, A., Ickin, S., y Gustafsson, J. (2020). Anomaly detection of event sequences using multiple temporal resolutions and markov chains. *Knowledge and Information Systems*, *62*(2), 669–686.

Carlson, J. L. (2013). Action [j]. *Media. johnwiley. com. au*.

Carmona, J. A. (2024). *Un "enchufe" wifi que avisa de caídas en casa. este es el innovador invento que quiere ayudar a los mayores.* Descargado 2025-09-08, de https://www.xatakahome.com/seguridad-en-el-hogar/enchufe-wifi-que-avisa-caidas-casa-este-innovador-invento-que-quiere-ayudar-a-mayores

CEPAL. (2005). *Dinámica demográfica y desarrollo en américa latina y el caribe.* Descargado de https://fiapam.org/wp-content/uploads/2012/10/lcl2235e-p.pdf

Doluí, K., y Kanti Datta, S. (2017). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. En *Global internet of things summit (glots).* doi: 10.1109/giots.2017.8016213

García, J., Molina, J., y Berlanga, A. (2018). *Ciencia de datos: Técnicas analíticas y aprendizaje estadístico en un enfoque práctico*. Alfaomega. Descargado de https://d1wqtxts1xzle7.cloudfront.net/64031156/Ciencia_de_datos_2018-libre.pdf

Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., ... Ritter, M.

(2017). Audio set: An ontology and human-labeled dataset for audio events. En *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 776–780).

Goldsborough, P. (2016a, 10). A tour of tensorflow. *arXiv:1610.01178 [cs]*. Descargado de https://arxiv.org/pdf/1610.01178 doi: 10.48550/arXiv.1610.01178

Goldsborough, P. (2016b). *A tour of tensorflow, proseminar data mining.* Descargado 2025-09-08, de https://arxiv.org/pdf/1610.01178

Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. *Genetic programming and evolvable machines*, *19*(1), 305–307.

Hernández, R., Fernández, C., y Baptista, P. (2014). *Metodología de la investigación* (6.ª ed.). McGraw-Hill. Descargado de https://www.esup.edu.pe/wp-content/uploads/2020/12/2.%20Hernandez,%20Fernandez%20y%20Baptista-Metodologa%20Investigacion%20Cientifica%206ta%20ed.pdf

Hochreiter, S., y Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hurtado, J. (2000). *Metodología de la investigación holística.* Descargado 2025-09-08, de https://ayudacontextos.wordpress.com/wp-content/uploads/2018/04/jacqueline-hurtado-de-barrera-metodologia-de-investigacion-holistica.pdf

Hyndman, R. J., y Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2.ª ed.). OTexts. Descargado de https://otexts.com/fpp2/

Igba, J., Alemzadeh, K., Durugbo, C., y Eiriksson, E. T. (2016). Analysing rms and peak values of vibration signals for condition monitoring of wind turbine gearboxes. *Renewable Energy*, *91*, 90–106.

Liu, F. T., Ting, K. M., y Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(1), 1–39.

Luis-García, L. C., y Gómez, A. R. T. (2024). Desarrollo de aplicaciones iot: metodologías y estrategias. *European Public & Social Innovation Review*, *9*, 1–18.

Malhotra, P., Vig, L., Shroff, G., Agarwal, P., y cols. (2015). Long short term memory networks for anomaly detection in time series. En *Proceedings* (Vol. 89, p. 94).

Malmberg, C., y Radszuweit, D. (2021). *Real-time audio classification on an edge device - using yamnet and tensorflow lite.* Descargado 2025-09-08, de https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1605037&dswid=6895

Matas Soberón, J. J. (2024). *Una introducción a las cadenas de markov y sus aplicaciones.* Descargado 2025-09-08, de https://dspace.uib.es/xmlui/bitstream/handle/11201/151803/Memoria_EPSU0697.pdf?sequence=1&isAllowed=y

Medina, M. (2019). *Edge computing para iot.* https://openaccess.uoc.edu/bitstream/10609/91207/7/mmedinabarTFM0119memoria.pdf. (Accessed: 2025-09-08)

Naciones Unidas. (2022). *Envejecimiento.* Descargado 2025-09-08, de https://www.un.org/es/global-issues/ageing

Peña, F. A. V. (2016). Reciente evolución de los hogares unipersonales en españa: una aproximación sociológica. *WPS Review International on Sustainable Housing and Urban Renewal: RI-SHUR*(3), 38–55.

Pons Puig, J., y cols. (2019). Deep neural networks for music and audio tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. Descargado de https://jordipons.me/media/PhD_Thesis_JordiPons.pdf

Pressman, R. (2010). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill. Descargado de https://www.academia.edu/87171392/Ingeniera_de_software_enfoque_practico_Roger_Pressman_7ma_edicin

Provost, F., y Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media. Descargado de https://www.researchgate.net/publication/256438799_Data_Science_for_Business

Ralla, E. (2024). *Hallan el cadáver en zaragoza de un hombre que llevaba muerto varios días en su casa.* Descargado 2025-09-08, de https://www.heraldo.es/noticias/aragon/zaragoza/2024/03/22/encuentran-cadaver-zaragoza-hombre-muerto-varios-dias-casa-1720901.html

Reis, M. J., y Serôdio, C. (2025). Edge ai for real-time anomaly detection in smart homes. *Future*

*Internet*, *17*(4), 179.

Richardson, M., y Wallace, S. (2016). *Getting started with raspberry pi: An introduction to the fastest-selling computer in the world* (4.ª ed.). Maker Media, Inc. Descargado de https://books.google.co.ve/books/about/Getting_Started_With_Raspberry_Pi.html?id=w4CkDAAAQBAJ&redir_esc=y

Russell, S., y Norvig, P. (2022). *Artificial intelligence: A modern approach* (4.ª ed.). Pearson. Descargado de http://lib.ysu.am/disciplines_bk/efdd4d1d4c2087fe1cbe03d9ced67f34.pdf

Sabo, M. (2020). *Nestjs*. Descargado 2025-09-08, de https://zir.nsk.hr/islandora/object/mathos:441

Shi, W., Cao, J., Zhang, Q., Li, Y., y Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. doi: 10.1109/JIOT.2016.2579198

Tamayo, M. (2004). *El proceso de la investigación científica*. Limusa. Descargado de https://www.gob.mx/cms/uploads/attachment/file/227860/El_proceso__de_la_investigaci_n_cient_fica_Mario_Tamayo.pdf

Tanenbaum, A. S., y Van Steen, M. (2007). Distributed systems: principles and paradigms. *Pearson Prentice Hall*.

Torija, A., Ruiz, P., y Ramos-Diao, A. (2018). Metodología para la identificación de eventos sonoros anómalos. En *Congreso nacional de acústica – tecniacústica 2018*. Descargado de https://documentacion.sea-acustica.es/publicaciones/Coimbra08/id206.pdf

Upton, E., y Halfacree, G. (2020). *Raspberry pi user guide* (4.ª ed.). John Wiley & Sons. Descargado de https://www.perlego.com/book/997788/raspberry-pi-user-guide-pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Yuill, S., y Halpin, H. (2006). *Python*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb. (Accessed: 2025-09-08)

Apéndices

Apéndice A. Representación cíclica del tiempo mediante funciones trigonométricas

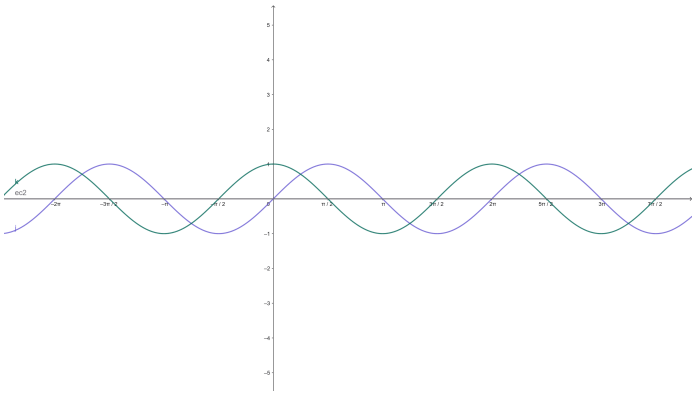

Figura 10: Visualización de las funciones seno y coseno

GeoGebra: Generación de las funciones seno y coseno usando las expresiones:
`Sequence((x, sin(x)), x, 0, 2π, π/6)` y
`Sequence((x, cos(x)), x, 0, 2π, π/6)`

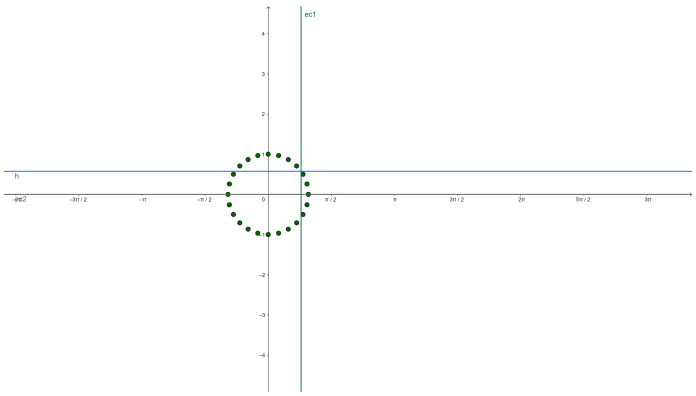

Figura 11: Visualización discreta del comportamiento cíclico del tiempo

GeoGebra: Generación de puntos discretos sobre la circunferencia usando la expresión:
`Sequence((cos(2π * n / 86400), sin(2π * n / 86400)), n, 0, 86400, 3600)`

La figura incluye dos líneas auxiliares que recorren la circunferencia: una horizontal (coseno) y una vertical (seno), generadas dinámicamente mediante un deslizador $t$ con paso de 600 segundos. Estas líneas intersectan en el punto $(x, y)$, representando la posición temporal proyectada en coordenadas trigonométricas.

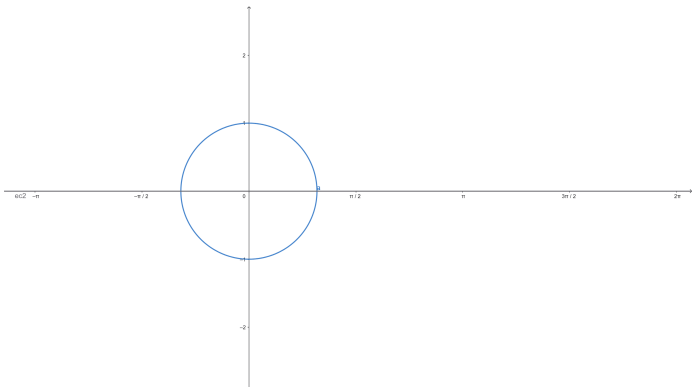

Figura 12: Curva paramétrica continua del tiempo sobre el círculo unitario

GeoGebra: Generación de la curva mediante las expresiones:
`x(t) = cos(2π * t / 86400),` `y(t) = sin(2π * t / 86400),`
`Curve(x(t), y(t), t, 0, 86400)`

La figura muestra una curva paramétrica continua que representa el tiempo sobre el círculo unitario mediante funciones trigonométricas. Cada instante se proyecta como un punto único definido por $x(t) = \cos\left(\frac{2\pi t}{86400}\right)$ y $y(t) = \sin\left(\frac{2\pi t}{86400}\right)$, donde $t$ es el número de segundos desde las 00:00. Esta codificación garantiza continuidad angular, permitiendo que modelos de simulación o aprendizaje automático interpreten correctamente la naturaleza cíclica del tiempo sin ambigüedades en los extremos del día.

Apéndice B. Algorimos de los distintos agentes

```
Algoritmo Agente Isolation Forest

Agente_IsolationForest()
    LeerConfiguración()  // tasas, umbrales, ventanas
    InicializarEstado()  // colas, flags, historial

Escuchar()
    Suscribir(canal_audio)
    Mientras true
        msg ← LeerMensaje()
        Si msg válido (texto + timestamp)
            evento ← NormalizarEvento(msg)   // agrega date y label
            Agregar(buffer, evento)
            Si |buffer| > MAX_BUFFER
                QuitarMasAntiguo(buffer)
        EsperarBreve()

Procesar()
    Mientras true
        Si |buffer| > 0
            lote ← Copia(buffer)
            df ← ConvertirADataFrame(lote)
            X ← GenerarCaracterísticas(df)
            resultados ← Predecir(Modelo, X) // etiquetas y puntajes
            anomalías ← Filtrar(resultados, pred = -1)
            Para cada label único en anomalías
                ConstruirMensaje(label, fecha, score)
                Publicar(canal_alertas, mensaje)
        EsperarIntervaloProceso()
```

Figura 13: Algoritmo del agente de detección de anomalías basado en Isolation Forest

```
Algoritmo Agente LSTM

Agente_LSTM()
    LeerConfiguración()  // tasas, umbrales, ventanas
    InicializarEstado()  // colas, flags, historial

Escuchar()
    Suscribir(canal_entrada)
    Mientras true
        msg ← LeerMensaje()
        Si msg es válido (tiene texto y timestamp)
            evento ← ConstruirEvento(msg)
            Agregar(buffer, evento)
            Si |buffer| > SEQ_LEN
                EliminarMásAntiguo(buffer)
        EsperarBreve()

Analizar()
    Mientras true
        Si |buffer| ≥ SEQ_LEN
            serie ← Copia(buffer)
            Para cada evento en serie
                CalcularFechaYHora()
                DerivarCaracterísticasCíclicas(día, tiempo)
            labels_enc ← CodificarEtiquetas(serie.labels)
            X ← Combinar(labels_enc, características_temporales)
            X2 ← ReconstrucciónModelo(X)
            mse ← ErrorCuadráticoMedio(X, X2)
            Si mse > umbral
                alerta ← ConstruirAlerta(serie, mse)
                Publicar(canal_alertas, alerta)
                Limpiar(buffer)
        EsperarIntervaloCorto()
```

Figura 14: Algoritmo del agente de predicción basado en LSTM

```
Algoritmo Agente Silencio

Agente_Silencio()
    LeerConfiguración()  // tasas, umbrales, ventanas
    InicializarEstado()  // colas, flags, historial
    IniciarStreamAudio(callback_audio)

CapturaYProcesamientoAudio()
    Mientras true
        frame ← ObtenerDeCola(timeout)
        variance ← CalcularVarianza(frame)
        actividad ← variance > UMBRAL
        ActualizarUltimaActividad(actividad)
        AñadirHistorial(frame, variance, actividad)
        RecortarHistorialSiExcedeVentana()
        Si HistorialInsuficiente()
            Continuar
        recent ← Últimos(STABILITY_FRAMES)
        activity_count ← ContarActividad(recent)
        silence_count ← STABILITY_FRAMES - activity_count
        frames_sugieren_actividad ← activity_count > silence_count
        hold ← TiempoDesdeUltimaActividad() ≤ HOLD_TIME
        estado_actual_actividad ← frames_sugieren_actividad O hold
        Si CambioASilencio(estado_actual_actividad)
            PublicarEstado("silence", métricas)
        Si CambioAActividad(estado_actual_actividad)
            PublicarEstado("activity", métricas)

PublicarEstado(tipo, datos)

Fin
```

Figura 15: Algoritmo del agente de detección de silencio

Apéndice C. Verificación del sistema de monitoreo acústico

| **Requerimiento** | **Descripción** | **Entrada** | **Salida** | **Criterio de Aceptación** |
|---|---|---|---|---|
| R1 | Captura continua del audio ambiental a través de micrófonos. | Señales de audio del entorno a través del hardware del micrófono. | Eventos discretos etiquedados del audio. | El sistema debe estar activo y registrando datos en todo momento. |
| R2 | Procesamiento del audio para clasificarlo en eventos sonoros predefinidos. | Flujo de datos de audio. | Clasificación del sonido (ej. voz, silencio, golpe). | El sistema debe etiquetar correctamente la señal de audio que recibe en todo momento. |
| R3 | El sistema debe conocer en todo momento el estado de actividad del entorno. | Señal de audio. | Hay ruido o silencio. | El sistema reconoce cuando hay ruido o silencio. |
| R4 | Comparación de la actividad en tiempo real con el perfil de normalidad para detectar patrones anómalos. | Secuencia de eventos en tiempo real | Identificación de una anomalía o evento atípico. | Detectar si una secuencia de eventos es normal o anómala. |

| Requerimiento | Descripción | Entrada | Salida | Criterio de Aceptación |
|---|---|---|---|---|
| R5 | El sistema realiza una consulta verbal al usuario al detectar una anomalía. | Señal de audio. | Emisión de una pregunta de voz pregrabada (ej. “¿Está todo bien?”). | El sistema consulta el estado del usuario antes de enviar una alerta. |
| R6 | Permitir cancelar una alerta de emergencia por comando de voz. | Respuesta de voz del usuario (ej. “Estoy bien”). | No se envía alerta. | El sistema cancela la alerta. |
| R7 | El sistema envía notificaciones de emergencia si la anomalía es crítica o el usuario no responde. | Falta de respuesta del usuario o gravedad de la anomalía | Envío de notificaciones. | El sistema es capaz de enviar una alerta sin la intervención del usuario. |

Tabla 4: Requerimientos del sistema acústico

Apéndice D. Cuadro comparativo de microcontroladores

| **Característica** | **Raspberry Pi 4 model B** | **NVIDIA Jetson Nano** | **Google Coral (TPU)** | **Arduino (ej. UNO/Mega)** | **ESP32** |
|---|---|---|---|---|---|
| Capacidad de Cómputo | Procesador ARM de alto rendimiento multinúcleo. | Procesador ARM de cuatro núcleos y GPU con 128 núcleos NVIDIA CUDA. | Procesador ARM, pero su poder de IA proviene de una TPU especializada. | Microcontrolador de 8 o 32 bits. | Microcontrolador de 32 bits con Wi-Fi y Bluetooth. |
| Memoria y almacenamiento | 2-8 GB RAM, microSD hasta 1 TB | 4 GB RAM, microSD hasta 128 GB | 1 GB RAM, microSD, almacenamiento interno limitado | 2-8 KB RAM, sin almacenamiento persistente | 512 KB RAM, soporte microSD externo |
| Costo | 65$ (Amazon) | 165$ (AliBaba) | 144$ | 27$ | Entre 10$ y 20$ (Amazon) |

Tabla 5: Cuadro comparativo de microcontroladores

Apéndice E. Manual de Usuario

## Introducción

Este manual está dirigido a las personas que se encuentran en los ambientes monitoreados por Alertify. Su propósito es explicar de forma sencilla cómo interactuar con el sistema por medio de la voz, qué frases utiliza para solicitar ayuda o cancelar una alerta, y cómo interpretar las señales y respuestas del sistema.

A través de esta guía, aprenderá a usar el sistema de manera segura y efectiva, aprovechando la tecnología de reconocimiento vocal para su tranquilidad y rápida atención en caso de ser necesario.

## ¿Qué es un agente?

Un agente en el contexto del siguiente sistema es un programa inteligente que escucha, interpreta y actúa ante sonidos del ambiente. No es un robot ni una persona, sino un software que:

- Tienen la capacidad de escuchar el sonido del ambiente a través de micrófonos para la detección de palabras clave (“Auxilio”, “ayuda”, “estoy bien”)
- Tienen la capacidad de detectar eventos de interés y periodos de silencio prolongados
- Al detectar algún evento relevante, envía datos al servidor para que este los notifique

## Acciones que puede realizar el usuario

El sistema está diseñado para responder a comandos de voz específicos que le permiten solicitar asistencia o cancelar una alerta de manera inmediata, sin necesidad de interactuar con ningún dispositivo físico. Su voz actúa como el control principal del sistema, facilitando su uso en situaciones donde pueda tener las manos ocupadas o dificultad para moverse.

Para solicitar ayuda en caso de emergencia, simplemente pronuncie en voz clara y audible alguna de las palabras clave predefinidas: .ªuxilio.º .ªyuda”. El sistema está configurado para reconocer estos

términos de manera prioritaria. Al detectar cualquiera de estas palabras, el sistema activa inmediatamente el protocolo de emergencia, enviando una notificación automática a la central de monitoreo y a los contactos designados para tales situaciones. Adicionalmente, el sistema puede generar un anuncio sonoro en la central operativa para asegurar una respuesta rápida por parte del personal responsable. Si ha activado una alerta por error o la situación de emergencia ha sido resuelta, puede cancelar el estado de alarma utilizando frases como .estoy bien", "todo bien.o simplemente "bien". Al reconocer estas expresiones, el sistema desactiva las notificaciones activas asociadas a ese ambiente específico e informa a los operadores sobre la cancelación, evitando así movilizaciones innecesarias mientras mantiene el historial del evento para su posterior revisión. Para garantizar el correcto funcionamiento del sistema, se recomienda hablar directamente en dirección al micrófono del ambiente, manteniendo un tono de voz claro y natural. En espacios con niveles elevados de ruido ambiental, puede ser necesario repetir la frase o acercarse ligeramente al dispositivo para asegurar una captura óptima del audio. El sistema está calibrado para reconocer específicamente los comandos antes mencionados, por lo que se sugiere utilizar exclusivamente estas expresiones para interactuar con él.

## Señales que puede emitir el sistema

1. Aviso de Emergencia: se activa cuando el sistema identifica de manera clara una situación de riesgo inminente. Esto ocurre principalmente cuando detecta alguna de las palabras clave de auxilio como .ayuda.o .auxilio", pero también puede activarse cuando reconoce otros sonidos críticos preconfigurados, como golpes fuertes, cristales rompiéndose o sonidos atípicos que se salen del patrón normal del ambiente. Cuando se genera esta señal, el sistema prioriza la notificación enviando alertas inmediatas a todos los contactos configurados y, dependiendo de la configuración, puede activar un anuncio de voz en la central de monitoreo que especifica el ambiente exacto donde se originó la emergencia. Esta señal está diseñada para obtener una respuesta rápida y directa del personal de atención.

2. Aviso de Silencio Prolongado: funciona como un mecanismo preventivo que se activa cuando el sistema detecta una ausencia inusual de actividad sonora en el ambiente durante un período de tiempo determinado. A diferencia del aviso de emergencia, esta señal no indica necesariamente una situación crítica, sino que sirve como recordatorio para verificar que todo se encuentra en orden. Este tipo de aviso es particularmente útil en ambientes donde se espera cierta regularidad de

actividad, permitiendo identificar posibles situaciones como personas que podrían necesitar asistencia, pero no pueden vocalizar una petición de ayuda. El sistema envía esta notificación como una alerta de menor prioridad que no activa los protocolos de emergencia completos, pero que merece una verificación por parte del personal designado.

Cada tipo de señal está calibrada según el nivel de urgencia que representa, permitiendo al personal de monitoreo priorizar su respuesta, mientras mantiene una supervisión comprehensiva del ambiente monitoreado.

## Escenarios comunes

El sistema está preparado para manejar diversas situaciones que pueden presentarse en el día a día. Comprender cómo actuar en cada escenario le permitirá interactuar de manera efectiva con el sistema y obtener la respuesta adecuada según sus necesidades. A continuación, se describen tres situaciones frecuentes y el procedimiento recomendado para cada una.

Cuando necesite ayuda de manera inmediata, ya sea por una caída, un malestar repentino o cualquier situación que considere una emergencia, su respuesta debe ser clara y directa. En estos casos, pronuncie en voz alta y firme alguna de las palabras clave que el sistema reconoce como prioritaria: .ªuxilio.º .ªyuda". Inmediatamente después de dar esta instrucción vocal, el sistema procesará su solicitud y activará los protocolos establecidos. Es importante que, una vez realizada la solicitud, espere pacientemente la confirmación o la llegada de la atención por parte del personal o los contactos designados, ya que el sistema notificará de manera automática y rápida a las personas encargadas de brindarle asistencia.

Si ya se encuentra fuera de peligro o la situación de emergencia ha sido resuelta, es fundamental que cancele el estado de alerta para evitar movilizaciones innecesarias. Para ello, simplemente exprese en voz clara alguna de las frases de cancelación, como .ºestoy bien", "todo bien.º çancelar". Al hacerlo, el sistema interpretará que la situación ha vuelto a la normalidad y procederá a desactivar todas las notificaciones activas asociadas a ese evento. Esta acción informa automáticamente al personal de monitoreo que ya no se requiere intervención, lo que les permite actualizar el estado del incidente y concentrarse en otras posibles emergencias.

En caso de que el sistema active una alerta de manera accidental, lo que se conoce como un falso positivo, es igualmente importante que cancele la notificación. Los falsos positivos pueden ocurrir cuando un ruido fuerte o una conversación es interpretada erróneamente por el sistema como una palabra de auxilio. Si

esto sucede, no es necesario que espere a que el personal se comunique con usted; puede tomar la iniciativa y pronunciar cualquiera de las frases de cancelación mencionadas anteriormente. Al hacerlo, el sistema detendrá el envío de notificaciones y registrará el evento como una activación no válida, lo que contribuye a que el sistema aprenda y mejore su precisión con el tiempo.

Apéndice F. Manual de Sistema

## Introducción

El siguiente documento es la guía de instalación de Alertify, un sistema de monitoreo acústico inteligente diseñado para la identificación de sonidos ambientales y la generación automática de alertas ante eventos sonoros anómalos que puedan indicar una situación de emergencia o vulnerabilidad.

El sistema está especialmente orientado a entornos domésticos o asistidos, proporcionando una solución no invasiva para el monitoreo continuo de personas que puedan requerir asistencia inmediata, como adultos mayores, personas con discapacidad o cualquier individuo en situación de riesgo.

La arquitectura del sistema se compone de los siguientes módulos principales:

1. **Agentes de captura y detección**, que ejecutan modelos especializados de inteligencia artificial para:
    - Reconocimiento de voz en tiempo real (Vosk)
    - Clasificación de eventos acústicos (YAMNet)
    - Detección de anomalías en secuencias sonoras (LSTM, Transformer, Isolation Forest)
    - Monitoreo de períodos de silencio prolongado (Silents Agent)
    - Persistencia de eventos (Data Agent)
    - Gestión de notificaciones de emergencia (Emergency Agent)
2. **Servidor central**, que actúa como coordinador del sistema, gestionando la comunicación en tiempo real vía WebSocket, el almacenamiento en base de datos PostgreSQL, y la ejecución de tareas programadas.

La arquitectura está realizada para operar en dispositivos de edge computing como el Raspberry Pi 4 modelo B, garantizando que el procesamiento de audio se realice de forma local y privada, sin almacenar grabaciones ni depender de servicios en la nube para el análisis sensible.

Este manual está dirigido a administradores de sistemas y usuarios técnicos, y proporciona instrucciones completas para la instalación, configuración, operación y resolución de problemas.

## Requisitos

### Hardware

1. Raspberry Pi 4 Modelo B con las siguientes especificaciones:
    - Procesador: Broadcom BCM2711, SoC de cuatro núcleos Cortex-A72 (ARM v8) de 64 bits a 1,8 GHz
    - Memoria Ram: 4GB
    - Almacenamiento: MicroSD 32GB Clase 10 o superior
    - Puertos: 2 × USB 3.0, 2 × USB 2.0, 2 × micro-HDMI
    - Audio: Entrada de micrófono via USB
    - Microfonos: Adafruit Mini Microfono USB 3367
2. Servidor central (mínimo).
    - Procesador: CPU 2 núcleos / 4 hilos
    - Memoria Ram: 4GB
    - Almacenamiento: 50GB

### Software

1. Raspberry Pi 4 Modelo B:
    - Sistema operativo: Raspberry Pi OS (64-bit) Bullseye o superior
    - Python: Versión 3.9 o 3.10
    - Librerias a utilizar: pandas, joblib, scikit-learn, tensorflow, numpy, sounddevice, python-socketio, pyaudio, vosk, aiohttp, psycopg2, aioredis, pyttsx3
2. Servidor:
    - Node.js: Versión 18 o superior
    - NestJs: Framework principal del servidor

- PostgreSQL: Versión 14 o superior (para persistencia de eventos)
- Redis: Versión 6 o superior (para mensajería Pub/Sub)
- Docker: Para contenerización de servicios (opcional pero recomendado)

### Infraestructura de red

1. Puertos requeridos
   - Redis: 6379
   - PostgreSQL: 5432
   - Servidor NestJS: 3000
   - pgAdmin (opcional): 8888
2. Conectividad
   - Red local estable entre Raspberry Pi y servidor
   - Acceso a internet para notificaciones externas (Telegram, SMTP)
   - Ancho de banda mínimo para comunicación WebSocket en tiempo real

### Modelo de IA y recursos.

1. Modelos Preentrenados
   - Yamnet: yamnet-agent/yamnet.tflite, yamnet-agent/yamnet_class_map.csv
   - VOSK (español): vosk-agent/vosk-model-small-es-0.42/
   - Modelo de Anomalias:
     *a*) lstm-agent/lstm_autoencoder_no_duplicates_p99.keras o lstm-agent/lstm_autoencoder_p99.keras, y lstm-agent/umbral_no_duplicates_p99.pkl o lstm-agent/umbral_p99.pkl
2. Conectividad
   - Red local estable entre Raspberry Pi y servidor

- Acceso a internet para notificaciones externas (Telegram, SMTP)
- Ancho de banda mínimo para comunicación WebSocket en tiempo real

## Instalación

### Servidor (alertify-server).

1. Clonar repositorio y abrir carpeta alertify-server.
2. Crear/validar archivo .env (se incluye uno de ejemplo). Revise credenciales y puertos.
3. Instalar dependencias (PNPM).
4. Levantar Redis y PostgreSQL con Docker Compose.
5. Iniciar el servidor NestJS.

### Cliente (alertify-client).

1. Clonar y abrir carpeta Alertify-client.
2. Verificar Python 3.9+ y crear un entorno virtual (opcional).
3. Asegurar que los siguientes modelos estén en sus rutas correspondientes.
4. Ejecutar los agentes y scripts principales

## Configuración

### Redis

Los agentes usan por defecto redis://default:alertify@localhost:6379. Si cambia el host o las credenciales, actualice los scripts o variables de entorno correspondientes.

### Configuración de audio

En silents-agent/silents-agent.py, verificar y ajustar según sea necesario:

- SAMPLE_RATE
- VARIANCE_THRESHOLD
- Otros parámetros relacionados con la captura de audio

**Visualización y monitoreo.**

- WebSocket: conectarse con un cliente Socket.IO al servidor para recibir emergency-alert
- pgAdmin (si se usa Docker): http://¡host¿:8888 para ver base de datos
- Logs de agentes en consola. Para Windows puede usar varias terminales o un gestor de procesos.

**Carpetas y Archivos (alertify-client).**

**Archivos en la raíz**

Componentes principales para orquestar y documentar el cliente.

| Archivo | Descripción |
|---|---|
| main.py | Lanza los agentes como procesos independientes, maneja señales de parada y espera su finalización. |
| README.md | Introducción y notas del cliente |
| requirements.txt | Lista de dependencias Python necesarias para ejecutar los agentes (pandas, tensorflow, aioredis, vosk, etc.). |
| .gitignore | Reglas de exclusión de Git (archivos temporales, modelos, entornos, etc.). |

**CSV**

Catálogos en CSV utilizados por agentes de audio para mapear clases y categorías.

| Archivo | Descripción |
|---|---|
| Clasificación de categorías.csv | Tabla de categorías de eventos/sonidos normalizados para etiquetado y análisis. |
| yamnet_class_map.csv | Etiqueta legible de YAMNet usado por el agente de clasificación. |

**data_agent**

Ingesta de eventos desde Redis, persistencia en PostgreSQL y generación periódica de dataset

| Archivo | Descripción |
|---|---|
| config.py | Lee data_agent/database.ini con ConfigParser y devuelve un dict de conexión PostgreSQL. |
| data_agent.py | Suscribe a Redis (alertas_audio, alertas_vosk), inserta eventos en tabla event y actualiza data/dataset.csv periódicamente. |
| generate_csv.py | Genera data/dataset.csv on-demand; normaliza zona horaria (America/Caracas) y guarda fechas sin tz. |
| postgres_connection.py | Gestiona una conexión global a PostgreSQL con psycopg2 usando la configuración cargada. |

**emergency_agent**

Agente que gestiona las alertas de emergencia.

| Archivo | Descripción |
|---|---|
| emergency_agent.py | Se suscribe a múltiples canales Redis, agrega alertas, permite pausar/reanudar, hace TTS y envía notificaciones a Telegram; expone control por vosk_commands / emergency_control. |

**Isolation-forest-agent**

Detección de anomalías de secuencias de eventos con Isolation Forest.

| Archivo | Descripción |
|---|---|
| encoder.pkl | Codificador para transformar etiquetas de eventos a vectores. |
| isolation-forest-agent.py | Escucha alertas_audio, construye features temporales (día/tiempo cíclico), detecta anomalías por batch y publica en alertas_isolation_forest; si falta modelo, entrena y guarda anomalies_model.pkl. |

**lstm-agent**

Detección de anomalías con Autoencoder LSTM en ventanas temporales

| Archivo | Descripción |
|---|---|
| encoder.pkl | Codificador para transformar etiquetas a vectores de entrada del modelo. |
| lstm_autoencoder_improved.keras | Modelo Keras de Autoencoder LSTM (versión mejorada). |
| lstm_autoencoder_no_duplicates_p99.keras | Variante entrenada filtrando duplicados consecutivos; calibrada a percentil 99. |
| lstm_autoencoder_p99.keras | Variante calibrada a percentil 99 con datos estándar. |
| lstm-agent.py | Consume alertas_audio, arma secuencias (longitud fija), calcula MSE de reconstrucción y publica series anómalas en alertas_lstm si supera el umbral. |
| umbral_improved.pkl | Umbral de MSE recomendado para lstm_autoencoder_improved.keras. |
| umbral_no_duplicates_p99.pkl | Umbral calibrado para la variante sin duplicados. |
| umbral_p99.pkl | Umbral calibrado para la variante p99 estándar. |

**silent-agent**

Detección en tiempo real de silencio/actividad a partir de audio del micrófono.

| Archivo | Descripción |
|---|---|
| silents-agent.py | Captura audio, calcula varianza por frame, aplica ventana de estabilidad y periodo de retención, y publica cambios de estado (silencio/actividad) en alertas_silencio con métricas. |

### Vosk-agent

Reconocimiento de voz offline con Vosk en español y disparo de eventos por palabras clave.

| Archivo | Descripción |
|---|---|
| vosk-agent.py | Escucha micrófono con sounddevice y Vosk, detecta palabras clave (auxilio/Estoy bien), publica en alertas_vosk y envía comandos a vosk_commands. |

**Vosk-model-small-es-0.42**

| Archivo | Descripción |
|---|---|
| README | Información del modelo y notas del proyecto Vosk. |
| am/final.mdl | Modelo acústico entrenado (acoustic model). |
| conf/mfcc.conf | Configuración de extracción de características MFCC. |
| conf/model.conf | Configuración general del modelo. |
| graph/disambig_tid.int | Archivo de desambiguación para el grafo de decodificación. |
| graph/Gr.fst | Grafo FST principal para decodificación. |
| graph/HCLr.fst | Composición HCL para el decodificador (fonética/lexicón). |
| graph/phones/word_bounday.int | Límites de palabras por audio. |
| ivector/final.dubm | UBM para i-vectors. |
| ivector/final.ie | Extracción de i-vectors. |
| ivector/final.mat | Matriz de proyección para i-vectors. |
| ivector/global_cmvn.stats | Estadísticas para normalización CMVN global. |
| ivector/online_cmvn.conf | Configuración CMVN online. |
| ivector/splice.conf | Configuración de splicing de características. |

**Yamnet-agent**

Clasificación de audio en streaming con YAMNet (TensorFlow Lite) y publicación de eventos.

| Archivo | Descripción |
|---|---|
| yamnet_class_map.csv | Archivo con el nombre de cada clase para interpretar predicciones. |
| yamnet-agent.py | Captura audio, infiere con TFLite, marca emergencias por índices críticos y publica eventos en alertas_audio. |
| yamnet.tflite | Modelo YAMNet en formato TensorFlow Lite usado por el agente. |

**Carpetas y Archivos (alertify-server).**

**Archivos en la raíz**

Archivos base de configuración, tooling y metadatos del proyecto NestJS/TypeORM.

| Archivo | Descripción |
|---|---|
| package.json | Scripts de desarrollo/producción (Nest, TypeORM, seeds), dependencias y configuración de Jest. |
| pnpm-lock.yaml | Bloqueo de dependencias (pnpm). |
| docker-compose.yml | Servicios contenedorizados (p.ej. PostgreSQL/Redis si aplica) para desarrollo. |
| nest-cli.json | Configuración del CLI de Nest (paths de compilación). |
| tsconfig.json | Configuración TypeScript principal. |
| tsconfig.build.json | Configuración TypeScript para compilación a producción (dist). |
| .eslintrc.js | Reglas de ESLint. |
| .prettierrc | Reglas de formateo Prettier. |
| .gitignore | Exclusiones de Git. |
| README.md | Documentación del servidor. |

### SRC

Código fuente del servidor (NestJS), organizado por módulos y configuración.

| Archivo | Descripción |
|---|---|
| main.ts | Inicializa contexto transaccional, configura Swagger en /docs, prefijo api/v1 y levanta el servidor en PORT. |
| app.module.ts | Carga .env, registra TypeORM y el módulo Events, habilita @nestjs/schedule. |
| config/ | Configuración de entorno y base de datos (TypeORM, DataSource, variables). |
| modules/ | Módulos de funcionalidad (actualmente events). |

### src/config

Configuraciones de entorno y base de datos

| Archivo | Descripción |
|---|---|
| environment.ts | Cargar variables desde .env y las exporta como objeto envConfig. |
| database.config.ts | Opciones DataSourceOptions para PostgreSQL: host/puerto/credenciales/DB, entities, synchronize, migraciones y timezone. |
| database.ts | Vuelve a exportar propiedades básicas de conexión a DB a partir de environment. |
| datasource.config.ts | Instancia DataSource (TypeORM) desde database.config para migraciones/CLI. |
| typeorm.config.ts | Config asíncrono de TypeORM para Nest; integra typeorm-transactional con addTransactionalDataSource. |

**src/modules/events**

Módulo responsable de WebSockets (Socket.IO), gestión de eventos, emergencias, actividad y configuración de alarmas.

| Archivo | Descripción |
|---|---|
| events.module.ts | Declara EventsGateway y EventsService, registra entidades TypeORM y configura MailerModule con SMTP (env). |
| events.gateway.ts | Gateway WebSocket: maneja conexión/desconexión, comandos im-ok, help-call, alarmOn/Off, emite emergency-alert y reproduce TTS (voz). |
| events.service.ts | Lógica de negocio: CRUD de eventos/emergencias, cron jobs para silencios y emergencias, builder de matriz de Markov por environment y notificaciones (Telegram/Email/TTS). |
| enums/events.enums.ts | Enumeración Calls para tipos de llamadas (imOk, helpCall). |
| querys/events.querys.ts | Consultas SQL: ocurrencias por categoría y transiciones entre categorías (para matriz de Markov). |

**Dto**

| Archivo | Descripción |
|---|---|
| events.module.ts | Declara EventsGateway y EventsService, registra entidades TypeORM y configura MailerModule con SMTP (env). |
| events.gateway.ts | Gateway WebSocket: maneja conexión/desconexión, comandos im-ok, help-call, alarmOn/Off, emite emergency-alert y reproduce TTS (voz). |
| events.service.ts | Lógica de negocio: CRUD de eventos/emergencias, cron jobs para silencios y emergencias, builder de matriz de Markov por environment y notificaciones (Telegram/Email/TTS). |
| enums/events.enums.ts | Enumeración Calls para tipos de llamadas (imOk, helpCall). |
| querys/events.querys.ts | Consultas SQL: ocurrencias por categoría y transiciones entre categorías (para matriz de Markov). |

**Entities**

| Archivo | Descripción |
|---|---|
| entities/event.entity.ts | Entidad Event: environment, category, score, emergency, rms, date y createdAt. |
| entities/emergencies.entity.ts | Entidad Emergency: environment, category, createdBy, timestamps emited/cancelled y createdAt. |
| entities/activity.entity.ts | Entidad Activity: environment, rms, date y createdAt. |
| entities/alarm-settings.entity.ts | Entidad AlarmSettings: estados de alarmas, límite de silencio en horas inHome y createdAt. |

## Links a los repositorios

1. Alertify-client: https://github.com/Cariea/alertify-client

2. Alertify-server: https://github.com/Cariea/alertify-server