UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

PLATAFORMA PARA LA MEDICIÓN Y CLASIFICACIÓN DE SERIES DE TIEMPO DE RUIDO GEOLOCALIZADO

POR

Ramón Ignacio Castillo Retamal

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Electrónico

Profesor guía
Mario Rubén Medina Carrasco

Profesional supervisor
Ingeniero Civil Electrónico Cristóbal Obregón Pérez

Junio 2024
Concepción (Chile)

Esta memoria está dedicada a todo aquel que desee profundizar en los temas aquí desarrollados. Espero les sea de utilidad.

Agradecimientos

Agradezco a mi mamá, por siempre estar ahí.

Agradezco a Julia Gonzales, por ser un pilar fundamental durante todo este proceso.

Agradezco a toda la gente de R9 ingeniería por darme la oportunidad de desarrollar con ellos.

Sumario

Los efectos negativos que la exposición prolongada a altos niveles de ruido puede provocar sobre nuestra salud es actualmente un tema de estudio, siendo el ruido un problema frecuente en zonas cercanas a carreteras con altos niveles de tráfico o en zonas de producción industrial, entre otros. Las normativas establecen limites sobre el nivel de ruido, y metodologías para medirlo, pero el alto costo de dispositivos en este tipo de medición dificulta su acceso por parte de la comunidad, la cual normalmente es la principal afectada con esta problemática.

En el presente trabajo se describe el desarrollo de una herramienta con la cual es posible caracterizar el ruido ambiental haciendo uso de un teléfono móvil, esta caracterización se hace en base a la medición de la intensidad o el nivel de ruido, la clasificación del ruido, la cual se obtiene haciendo uso de un modelo de clasificación de audio, y la ubicación en la cual se mide el ruido.

Para lo cual primero se investiga material bibliográfico respecto a qué es el ruido y cómo medirlo, clasificarlo y geolocalizarlo. Luego, se explica la arquitectura de la herramienta y los principales factores involucrados en su desarrollo. Después se mostrarán distintas pruebas realizadas con el objetivo de validar los métodos usados con los distintos propósitos antes mencionados. Finalmente se discutirán los resultados obtenidos y se entregarán conclusiones sobre el trabajo.

Como resultado de este trabajo se obtiene una herramienta de fácil acceso la cual permite medir el ruido con un alto nivel de precisión, con fines informativos, además de entregar una caracterización del ruido y la geolocalización de donde se hizo la medición para un posterior análisis de toda esta información.

Summary

The negative effects that prolonged exposure to high noise levels can have on our health are currently a topic of study, as noise is a common problem in areas near highways with heavy traffic or in industrial production zones, among others. Regulations establish limits on noise levels and methodologies for measuring it, but the high cost of devices for this type of measurement makes it difficult for the community, which is typically the primary affected party, to access them.

In this work, we developed a tool that allows the characterization of environmental noise using a mobile phone. This characterization is based on the measurement of noise intensity or level, noise classification obtained through an audio classification model, and the location where the noise is measured.

To achieve this, we first research relevant works on what noise is and how to measure, classify, and geolocate it. Next, we explain the architecture of the tool and the main factors involved in its development. Then, we present various tests conducted to validate the methods used for the aforementioned purposes. Finally, we present and discuss our results, and share our conclusions about this work.

As a result of this work, an easily accessible tool is obtained that allows noise measurement with a high level of precision for informational purposes, in addition to providing a characterization of the noise and the geolocation of where the measurement was taken for subsequent analysis of all this information.

Tabla de contenidos

Lista de tablas

Lista de figuras

Capítulo 1: Introducción

En este capítulo se presenta el problema a evaluar, junto con sus objetivos, hipótesis de entrada, metodología de trabajo y planificación.

1.1. Descripción del problema

En los últimos años, junto a un aumento general de la población, la concentración de habitantes en zonas urbanas e industriales ha crecido, exponiéndose así a la población a diferentes tipos de emisiones propias de cada una de estas zonas. Una de estas emisiones es la emisión acústica, la cual en los últimos años ha tomado relevancia debido a distintos estudios que la relacionan a una variedad de enfermedades. Uno de los efectos más conocidos producto de la exposición constante a altos niveles de ruido es la pérdida de audición. La OMS en un estudio de 2021 [1] estimó que más de un 5% de la población mundial (430 millones de personas) padece una pérdida de audición y requiere rehabilitación (entre ellos 34 millones de niños). Además, se estima que para 2050 esta cifra podría superar los 700 millones (una de cada diez personas). Otro estudio reciente [2] relaciona la contaminación acústica con problemas cardiovasculares, las cuales, según el departamento de estadística e información en salud, DEIS, ocupan los primeros puestos como causas de muerte en la población chilena. Otros estudios, como el de Hahad, *et al* [3], revisan la relación entre la exposición a contaminación acústica y la salud mental.

En una publicación de 2018 [4] la OMS recomienda ciertos límites para prevenir enfermedades asociadas a la contaminación acústica. El Decreto Supremo N°38/2011 del ministerio del medio ambiente en Chile regula la contaminación acústica producida por fuentes fijas y establece métodos para hacer mediciones. Estas mediciones se deben llevar a cabo con instrumentos previamente calibrados y certificados según la norma IEC 61672/1:2002 “Sonómetros”. El acceso a dichos instrumento y certificación requiere de una inversión monetaria importante, la cual la población principalmente afectada normalmente no está dispuesta a pagar.

Otro problema presente en las mediciones de contaminación acústica es el determinar su origen, tanto de forma espacial como determinar que la produce. Esta información complementaria permite determinar con un mayor nivel de certeza el problema y diseñar soluciones más precisas.

1.2. Objetivo general

Desarrollar una herramienta que permita medir y clasificar series de tiempo de ruido asociadas a una geolocalización, crear una aplicación para dispositivos Android que implemente la herramienta e integrar la herramienta en la plataforma de incidencias ambientales de la empresa R9 ingeniería.

1.3. Objetivos específicos

i. Implementar un algoritmo para medir la intensidad del ruido en una aplicación móvil para sistemas operativos Android y en un servicio web disponible dentro de la red de R9 ingeniería
ii. Implementar un algoritmo para clasificar ruido en una aplicación móvil para sistemas operativos Android y en un servicio web disponible dentro de la red de R9 ingeniería
iii. Registrar series de tiempo con los datos obtenidos con los algoritmos implementados y guardarlos en la plataforma Airviro
iv. Asociar las series de tiempo a una geolocalización
v. Almacenar los datos recopilados y mostrarlos en la plataforma de incidencias ambientales de R9 ingeniería
vi. Generar informes con los datos recopilados y mostrarlos en plataforma de incidencias ambientales de R9 ingeniería

1.4. Metodología de trabajo

La metodología contempla los siguientes puntos:

1. Revisión y análisis de publicaciones e investigaciones recientes sobre el efecto de la contaminación acústica en la salud de las personas. Se revisarán estudios y publicaciones científicas que propongan métodos para medir el ruido con herramientas más accesibles que las presentes actualmente en el mercado. Se revisarán estudios y publicaciones científicas de los últimos años sobre métodos para clasificar ruido ambiental.
2. Desarrollo de una herramienta que permita la medición del nivel de ruido con fines informativos, que obtenga la clasificación del ruido con un alto nivel de precisión y que

además permita obtener la ubicación geográfica del lugar donde se hace la medición. Además, esta herramienta debe complementar estos datos con información subjetiva de quien hace la medición y almacenar los datos para su posterior visualización y análisis.

3. Comparación de las mediciones del nivel de ruido hechas por la herramienta con las hechas por un sonómetro. Medición de la precisión en la clasificación usando un conjunto de datos de prueba. Además, se compararán las clasificaciones hechas por la herramienta con lo percibido durante la medición en distintos entornos. Verificación de que la ubicación geográfica obtenida por la herramienta coincida con el lugar en el que se hacen las mediciones.
4. Análisis de los resultados obtenidos y el estudio de la factibilidad de la herramienta para su uso en el monitoreo de contaminación acústica.
5. Elaboración de conclusiones obtenidas a partir del desarrollo de la herramienta y el análisis de los resultados.

Capítulo 2: Marco teórico

2.1. Ruido

La OMS define ruido como “sonido molesto”. Dado lo subjetivo de su definición, se puede presentar en diversas formas. La exposición prolongada a éste puede provocar distintas alteraciones en la salud [1] [2] [3] [4]. Entonces, para entender que es el ruido necesitamos definir que es el sonido.

2.1.1. Sonido

El sonido es un fenómeno físico dado por la alteración mecánica de partículas en un medio elástico que es capaz de provocar una sensación auditiva, las que se propagan generalmente por el aire transmitiéndose en forma de ondas sonoras. Estas, al llegar al oído, hacen vibrar la membrana del tímpano y, mediante una serie de mecanismos propios de la anatomía, finalmente, esta información llega al cerebro y provoca una sensación sonora. El nivel de presión sonora o NPS se expresa en decibeles (dB) y se define por la siguiente relación matemática.

$$L_p = 20\ Log_{10}\left(\frac{P1}{P}\right) \tag{1.1}$$

Donde P1 es el valor de la presión sonora medido y P el valor de una presión sonora de referencia, establecido en $3x10^{-5}$ $(\frac{N}{m^2})$, la cual corresponde al nivel de presión más bajo que el oído comúnmente puede percibir.

El sonido se puede caracterizar a través de distintas propiedades, estas son:

- Amplitud: Medida máxima de la alteración de las partículas respecto a una posición de equilibrio. Representa la intensidad del sonido, mayor amplitud implica un sonido más fuerte.
- Periodo: Tiempo que tarda una onda en completar un ciclo. Es inversamente proporcional a la frecuencia.
- Frecuencia: Numero de ciclos completos de una onda en un segundo. Se mide en Hertz (Hz). Es responsable del tono del sonido; sonidos de alta frecuencia tienen tonos agudos y sonidos con baja frecuencia tiene tonos grabes.

- Longitud de onda: Es la distancia que recorre una onda durante un ciclo completo. Se mide en metros.

2.1.2. Contaminación acústica

El ruido es un fenómeno que está presente de forma inevitable en todas las actividades que se realicen, en industrias, construcciones, aeropuertos, zonas urbanas, etc. Se dice que es el contaminante más barato y además el más fácil de emitir. A diferencia de otros contaminantes el ruido no deja residuo en el lugar físico, pero si puede tener un efecto acumulativo en la persona que está expuesta a este contaminante.

Cuando se habla de contaminación acústica se hace referencia a la presencia de ruidos o vibraciones en el ambiente, la cual trae molestias, riesgos o daños. No todo sonido se puede considerar como contaminación acústica, por lo que a nivel nacional existen distintas normativas que regulan el ruido generado tanto por fuentes fijas como por fuentes móviles. Además, existen otras regulaciones que abordan el problema del ruido desde otras instancias, tales como la planificación territorial, el uso de la bocina, o los ruidos conductuales a nivel local, entre otros. La regulación para fuentes móviles se divide en dos partes. La primera controla el ruido generado por buses de la locomoción colectiva, Norma de Emisión de Ruido para Buses de Locomoción Colectiva Urbana y Rural – Decreto Supremo N°129/2001 del Ministerio de Transporte y Telecomunicaciones. Esta norma se controla mediante mediciones en 3 instancias, con el modelo del bus nuevo antes de ingresar al parque vehicular, en las revisiones técnicas periódicas y en la vía pública. Por otra parte, para la regulación de fuentes móviles, existe la Norma de Emisión de Ruido para vehículos Livianos, Medianos y Motocicletas – Decreto Supremo N°7/2015 del Ministerio del Medio Ambiente. Esta es una norma de ingreso, la cual se controla sólo para los modelos nuevos de vehículos o motocicletas, que desean ingresar al parque vehicular nacional. Finalmente, la regulación de ruido generado por fuentes fijas se controla y fiscaliza tanto por denuncias, como mediante los programas de cumplimiento y a través del Sistema de Evaluación de Impacto Ambiental. La regulación vigente está dada por la Norma de Emisión de Ruidos Generados por Fuentes que Indica – Decreto Supremo N°38/2011 del Ministerio del Medio Ambiente. Si nos centramos en esta última podemos ver que los limites se establecen dependiendo del tipo de zona y la hora del día. La tabla 2.1 muestra estos valores.

Tabla 2.1: Niveles máximos de Presión Sonora Corregidos (Npc) en dB(A)

	De 7 a 21 horas	**De 21 a 7 horas**
Zona I	55 dB	45 dB
Zona II	60 dB	45 dB
Zona III	65 dB	50 dB
Zona IV	70 dB	70 dB

En donde,

- zona 1: aquella zona definida en el instrumento de planificación territorial respectivo y ubicada dentro del límite urbano, que permite exclusivamente uso de suelo residencial o bien este uso de suelo y alguno de los siguientes usos de suelo: espacio público y/o área verde.
- zona II: aquella zona definida en el instrumento de planificación territorial respectivo y ubicada dentro del límite urbano, que permite además de los usos de suelo de la zona I, equipamiento de cualquier escala.
- zona III: aquella zona definida en el instrumento de planificación territorial respectivo y ubicada dentro del límite urbano, que permite además de los usos de suelo de la zona II, actividades productivas y/o de infraestructura.
- zona IV: aquella zona definida en el instrumento de planificación territorial respectivo y ubicada dentro del límite urbano, que permite sólo usos de suelo de actividades productivas y/o de infraestructura.

2.2. ¿Cómo medir el ruido ambiental?

2.2.1. Dispositivos para medir el ruido ambiental

La parte fundamental de cada dispositivo orientado a la medición de ruido es el micrófono, siendo éste el que convierte las vibraciones de presión sonora en señales eléctricas proporcionales a la intensidad del sonido. Dicho esto, los dispositivos más comúnmente usados en la medición de ruido son:

- Cámara acústica:

Herramientas que permiten “ver” el ruido haciendo uso de la tecnología *beamforming*, la cual consiste en usar un arreglo de antenas. Tras procesar las señales recibidas en cada una de ellas permiten determinar la ubicación de la señal captada. Esta información se combina con la imagen logrando así saber con exactitud el origen del sonido.

Figura 2.1: Cámara acústica Fluke ii900

- Sonómetros

Estos son posiblemente los instrumentos más comúnmente usados en la medición de ruido pues son las herramientas normalmente exigidas por las normas sobre monitoreo de contaminación acústica. Se pueden clasificar dependiendo de la clase a la que pertenezca y de si es o no un sonómetro integrador. Un sonómetro de clase 1 es un instrumento de alta precisión que cumplen con los estándares más rigurosos establecidos, diseñado para hacer mediciones en entornos donde se requiere el mínimo error posible, estando este error en ±1.1 dB. Un sonómetro de clase 2 tiene menor precisión que un sonómetro clase 1, siendo también más barato. A pesar del margen de error ligeramente mayor, ±1.4 dB, los sonómetros de clase 2 son adecuados para la mayoría de las aplicaciones donde se requiera medir ruido en la vida diaria, por lo que también son aceptados por las normas. Un sonómetro integrador es capaz de registrar el NPS durante un periodo de tiempo determinado, permitiendo así avaluar la exposición acumulada del ruido. Por otro lado, un sonómetro no integrador mide el NPS en un momento especifico.

Figura 2.2: Sonómetro Optimus + de Cirrus Research

- Aplicaciones móviles

Los teléfonos inteligentes se han convertido en parte fundamental en nuestra vida, llegando al punto en que todos llevamos alguno de estos en cada momento del día. La evolución continua de los dispositivos móviles ha llegado al punto en que éstos integran una variedad de sensores de alta calidad para su uso en diversas tareas. Uno de estos sensores son los micrófonos. Hoy en día un teléfono móvil incorpora al menos un micrófono interno, cuyo nivel de sensibilidad varía dependiendo de la marca y el modelo del dispositivo. Gracias a lo anterior es posible encontrar una variedad de aplicaciones móviles que proponen medir el ruido ambiental con un nivel de error a veces tan bajo como el de un sonómetro.

Dada la variación en la sensibilidad que depende de la marca y modelo, estas mediciones son usadas con fines informativos y no se pueden usar como mediciones oficiales ante los instrumentos reguladores. No obstante, existen diversos estudios recientes [5] [6] [7] [8] que intentan demostrar lo útiles que pueden ser estas aplicaciones cuando se necesitan hacer mediciones no oficiales, dado su fácil acceso a un bajo costo.

Figura 2.3: Aplicación "Sonómetro" de Splend Apps en la Google Play Store

2.2.2. Medición de ruido

En cuanto a cómo medir el ruido, en Chile el Decreto Supremo N°38/2011 del ministerio del medio ambiente establece que esta medición se debe llevar a cabo haciendo uso de un sonómetro promediador – integrador, que debe cumplir con lo establecido en la norma IEC 61672/1:2002 "Sonómetros".

En la norma anteriormente mencionada entonces se define el nivel de presión sonora como el descrito en la ecuación 1.1. Luego el nivel de presión sonora equivalente para un periodo de tiempo T se describe en la ecuación 2.1.

$$L_{eqp} = 10\,Log_{10}\left(\frac{1}{T}\sum_{0}^{T}\sqrt{10^{\frac{L_{pi}}{10}}}\right) \qquad (2.1)$$

En la medición de NPS comúnmente se pide usar ponderaciones en frecuencia las cuales

permiten adaptar las mediciones a distintas situaciones en las que la percepción del sonido puede variar. La norma pide que un sonómetro tenga ponderación frecuencia tipo A, pues ofrece una buena correlación con la subjetividad de la humana. Además, existen otros tipos de ponderaciones con las que se debe contar en algunas situaciones específicas tal como la ponderación C, la cual se utiliza para medir niveles altos de presión sonora, y la ponderación Z, la cual considera un espectro de frecuencias mayor. La tabla 2.2 muestra estas ponderaciones frecuenciales y el límite de tolerancia.

Tabla 2.2: Ponderaciones frecuenciales y límites de incertidumbre

Frecuencia nominal Hz	**Ponderaciones frecuenciales dB**			**Límites de tolerancia (dB)**	
				Clase	
	A	**C**	**Z**	**1**	2
10	-70,4	-14,3	0,0	+3,5; -∞	+5,5; -∞
12,5	-63,4	-11,2	0,0	+3,5; -∞	+5,5; -∞
16	-56,7	-8,5	0,0	+3,5; -4,5	+5,5; -∞
20	-50,5	-6,2	0,0	±2,5	±3,5
25	-44,7	-4.4	0,0	+2,5; -2,0	±3,5
31,5	-39,4	-3,0	0,0	±2,0	±3,5
40	-34,6	-2,0	0,0	±1,5	±2,5
50	-30,2	-1,3	0,0	±1,5	±2,5
63	-26,2	-0,8	0,0	±1,5	±2,5
80	-22,5	-0,5	0,0	±1,5	±2,5
100	-19,1	-0,3	0,0	±1,5	±2,0
125	-16,1	-0,2	0,0	±1,5	±2,0
160	-13,4	-0,1	0,0	±1,5	±2,0
200	-10,9	0,0	0,0	±1,5	±2,0
250	-8,6	0,0	0,0	±1,4	±1,9
315	-6,6	0,0	0,0	±1,4	±1,9
400	-4,8	0,0	0,0	±1,4	±1,9
500	-3,2	0,0	0,0	±1,4	±1,9
630	-1,9	0,0	0,0	±1,4	±1,9
800	-0,8	0,0	0,0	±1,4	±1,9
1000	0	0	0	±1,1	±1,4

1250	+0,6	0,0	0,0	±1,4	±1,9
1600	+1,0	-0,1	0,0	±1,6	±2,6
2000	+1,2	-0,2	0,0	±1,6	±2,6
2500	+1,3	-0,3	0,0	±1,6	±3,1
3150	+1,2	-0,5	0,0	±1,6	±3,1
4000	+1,0	-0,8	0,0	±1,6	±3,6
5000	+0,5	-1,3	0,0	±2,1	±4,1
6300	-0,1	-2,0	0,0	+2,1; -2,6	±5,1
8000	-1,1	-3,0	0,0	+2,1; -3,1	±5,6
10000	-2,5	-4,4	0,0	+2,6; -3,6	+5,6; -∞
12500	-4,3	-6,2	0,0	+3,0; -6,0	+6,0; -∞
16000	-6,6	-8,5	0,0	+3,5; -17,0	+6,0; -∞
20000	-9,3	-11,2	0,0	+4,0; -∞	+6,0; -∞

Luego, para las frecuencias que no están en la tabla, las ponderaciones tipo C, A y Z deben calcularse según las ecuaciones 2.2, 2.3 y 2.4 respectivamente

$$C(f) = 20\log_{10}\left[\frac{f_4^2 f^2}{(f^2+f_1^2)(f^2+f_4^2)}\right] - C_{1000} \tag{2.2}$$

$$A(f) = 20\log_{10}\left[\frac{f_4^2 f^4}{(f^2+f_1^2)(f^2+f_2^2)^{1/2}(f^2+f_3^2)^{1/2}(f^2+f_4^2)}\right] - A_{1000} \tag{2.3}$$

$$Z(f) = 0 \tag{2.4}$$

En donde A_{1000} y C_{1000} son constantes normalizadas, en decibelios, que representan la ganancia eléctrica necesaria para proporcionar ponderaciones frecuenciales de 0 dB a 1kHz. La característica de la ponderación tipo C se realiza por dos polos de baja frecuencia a la frecuencia f_1, dos polos de alta frecuencia a la frecuencia f_4, y dos ceros a 0 Hz. Con estos polos y ceros, la respuesta de potencia para la característica de ponderación C, relativa a la frecuencia de referencia

$f_r = 1\ kHz$, será deducida por $D^2 = 1/2$ (aproximadamente -3 dB) a $f_L = 10^{1,5} Hz$ y $f_H = 10^{3,9} Hz$. La característica de ponderación A se realiza añadiendo dos filtros pasa alto de primer orden acoplados a la característica de ponderación tipo C. Para cada filtro pasa alto, la frecuencia de corte está dada por $f_A = 10^{2,45} Hz$. Las frecuencias f_1, f_2, f_3 y f_4 están dadas por las ecuaciones 2.5, 2.6, 2.7 y 2.8 respectivamente.

$$f_1 = \left[\frac{-b - \sqrt{b^2 - 4c}}{2}\right]^{1/2} \tag{2.5}$$

$$f_2 = \left[\frac{-b + \sqrt{b^2 - 4c}}{2}\right]^{1/2} \tag{2.6}$$

$$f_3 = \left[\frac{3 - \sqrt{5}}{2}\right] f_A \tag{2.7}$$

$$f_4 = \left[\frac{3 + \sqrt{5}}{2}\right] f_A \tag{2.8}$$

Con b y c tales que

$$b = \left(\frac{1}{1 - D}\right)\left[f_r^2 + \frac{f_L^2 f_H^2}{f_r^2} - D(f_L^2 + f_H^2)\right] \tag{2.9}$$

$$c = f_L^2 f_H^2 \tag{2.10}$$

Resolviendo se obtienen valores aproximados para las frecuencias a usar en las ecuaciones 2.2 y 2.3 tales que $f_1 = 20{,}6\ Hz$, $f_2 = 107{,}7\ Hz$, $f_3 = 737{,}9\ Hz$ y $f_4 = 12194\ Hz$. Además, A_{1000} y C_{1000} redondeadas al 0,001 dB más cercano, son -0,062 dB y -2,000 dB, respectivamente.

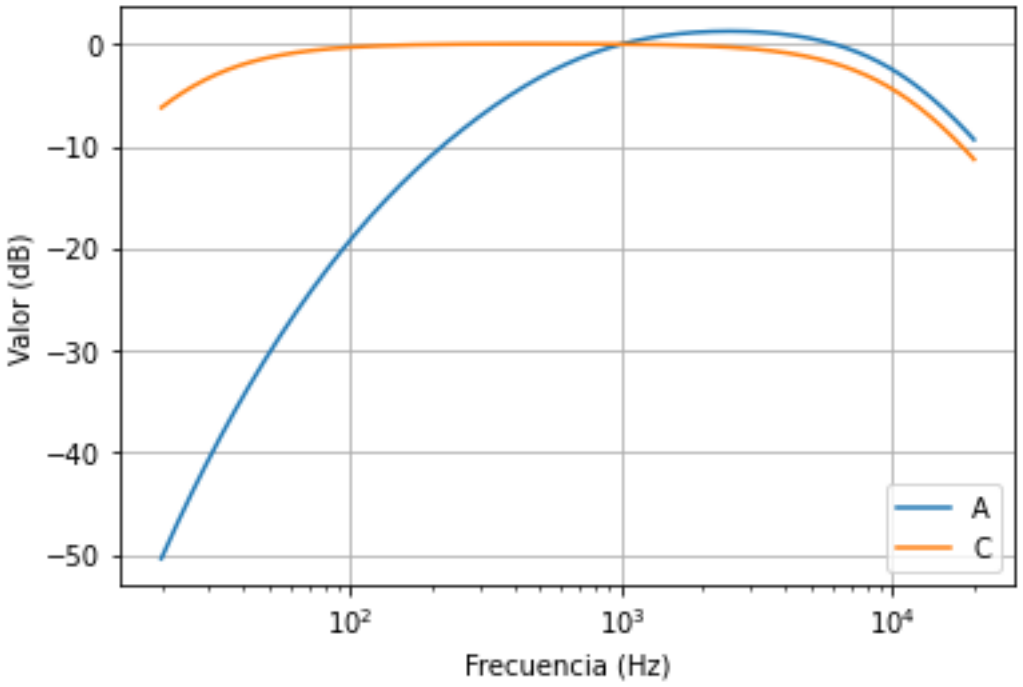


Figura 2.4: Curvas de ponderación A y C en función de la frecuencia

Entonces, al aplicar el filtro de ponderación tipo A, la ecuación 1.1 se convierte en

$$L_{Ap} = 20\ Log_{10}\left(\frac{P1}{P} * \Delta_{filtro}(f)\right) \tag{2.11}$$

Y la ecuación 2.1 queda como

$$L_{eqp} = 20\ Log_{10}\left(\frac{1}{T}\sum_{0}^{T}\sqrt{10^{\frac{L_{Api}}{10}}}\right) \tag{2.12}$$

Además, Δ_{filtro} se puede calcular como

$$\Delta_{filtro}(f) = \sqrt{10^{\frac{A(f)}{10}}} \tag{2.13}$$

2.2.3. Algoritmos para calcular el ruido

Zamora *et al.* [5] proponen 3 algoritmos para el cálculo del NPS en sistemas en dispositivos móviles, los cuales se pueden aplicar en cualquier dispositivo que cuente con un micrófono, un conversor análogo digital y una memoria para el muestreo y guardado de datos de presión sonora. Estos algoritmos son,

- Cálculo en el dominio de la frecuencia

  Consiste en leer la memoria en donde se guardan los valores asociados a la presión sonora y aplicarles la transformada de Fourier para trabajarlos en el dominio de la frecuencia. Luego, en el dominio de la frecuencia, se usa la ecuación 2.12 para calcular el ruido equivalente durante el periodo muestreado.

- Cálculo en el dominio del tiempo

  Este es el procedimiento más simple pues consiste en tomar los datos muestreados y aplicar la ecuación 1.1 para obtener el ruido instantáneo en el momento muestreado.

- Cálculo en el dominio del tiempo con datos normalizados

  Similar al método anterior, pero usando una presión referencia medida en Pascales de $p_{ref} = 20[\mu P]$, la cual es la referencia estándar para medir la presión del sonido, y normalizando los datos muestreados respecto a esto.

Zamora *et al.* [5] concluye, tras varios experimentos, que el algoritmo con mejores resultados es el que hace el cálculo en el dominio de la frecuencia.

Algoritmo 1: Calculo de dB(A) en el dominio de la frecuencia	
	Data: Buffer Data
	Input: SR (Sample Rate), BS (Buffer Size), T (Total time)
1	k = SR/BS
2	**for** c = 1 **to** T **do**:
3	dat = 0
4	**for** z = 1 **to** k **do**:
5	Leer los datos muestreados, almacenados en un buffer
6	**for** i = 0 **to** BS **do**:
7	w = $\frac{1}{2}(1 - \cos\left(\frac{2\pi i}{BS-1}\right))$
8	ArraySample[i] = Data[i]*w
9	**end**
10	fft = FFT(ArraySample)
11	**for** i = 0, j = 0; i<BS/2; i++, j+=2 **do**
12	mag[i] = $\sqrt{fft[j]^2 + fft[j+1]^2}$
13	dat = dat + $\sqrt{10^{\frac{L_{pmag[i]}}{10}}}$
14	**end**
15	**end**
16	return $20 \operatorname{Log}_{10} \frac{dat}{k}$
17	**end**

2.3. Modelos de clasificación de audio

Como consecuencia del aumento en la población y de la concentración de ésta en lugares cercanos a industrias, instituciones educacionales, carreteras, entre otros lugares con alto nivel de población, la necesidad de monitoreo y control se hace cada vez mayor. Es en este contexto que surge el concepto de "Ciudades Inteligentes" surge, IBM en un artículo [9] define una ciudad inteligente como "...una zona urbana en la que la tecnología y la recopilación de datos contribuyen a mejorar la calidad de vida, así como la sostenibilidad y el funcionamiento eficiente de la ciudad." En este contexto, uno de los principales requisitos es el poder caracterizar sonidos urbanos, lo que envuelve distintas tareas tales como la clasificación y segmentación. Esto ha implicado un aumento

creciente en el número de publicaciones hechas al respecto en los últimos años explorando distintas técnicas principalmente basadas en aprendizaje profundo o aprendizaje de máquina para resolver este problema (figura 2.6). Rodrigues *et al.* [10] hacen una revisión de los más destacados de estos trabajos y concluyen que son tres los pasos fundamentales en la clasificación de audio: preprocesado, extracción de características y clasificación; por otro lado, Abayomi-Alli *et al.* [20] hacen una revisión de los distintos métodos más usados en la clasificación, agregando un nuevo paso a los mencionados por Rodrigues, aumento de datos o "data augmentation". Dado lo recién mencionado, a continuación, se explicará brevemente en que consiste cada una de estas fases y se darán algunos ejemplos de las técnicas más utilizadas, pero antes se hará una introducción a distintos conjuntos de grabaciones de sonido ambiental disponibles. Un resumen de todo el proceso para crear un modelo de clasificación de audio se muestra en la figura 2.5.

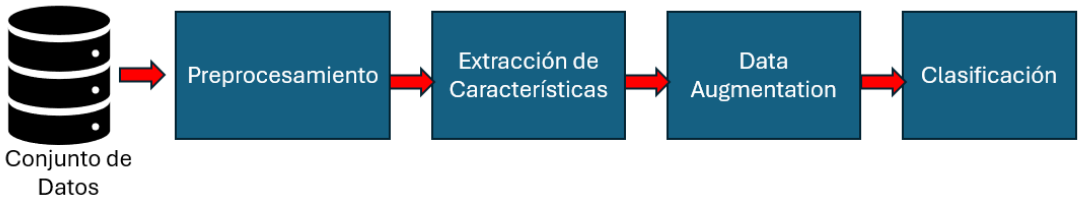


Figura 2.5: Resumen de los pasos para clasificar audio

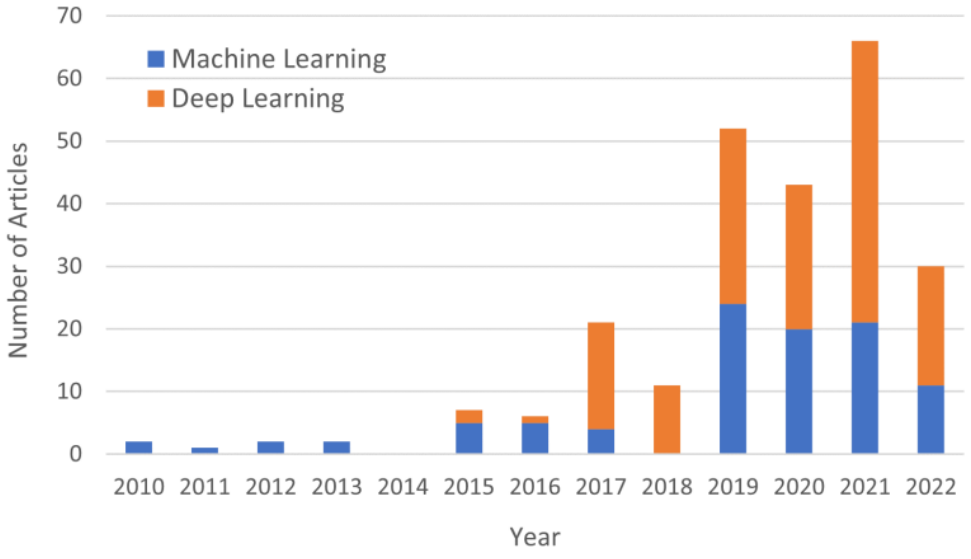


Figura 2.6 Evolución de artículos publicados basados en técnicas de aprendizaje profundo o aprendizaje de maquina relacionados al procesamiento o clasificación de sonido ambiental entre 2010-2022, extraído de [10]

2.3.1. Conjuntos de datos de sonido ambiental

Dada la necesidad de grandes conjuntos de datos de audio necesarios para el desarrollo y prueba de modelos de clasificación y a la extensiva tarea que implica la recolección y etiquetado de estos datos, existen, a disposición de cualquiera que los necesite, diferentes conjuntos de datos con audios de diferentes orígenes pensado cada cual para una tarea diferente. Rodrigues *et al.* [10] proporciona una recopilación de conjuntos de datos de sonido ambiental la que se muestra en la tabla 2.3.

Tabla 2.3: Resumen de conjuntos de datos con sonidos urbanos, extraído de [10]

Referencia	**Nombre**	**Tamaño**	**N° de clases**	**Duración**	**Descripción**
Salamon *et al*. [11]	UrbanSound8k	8732	10	≤4 s	Contiene la metadata, clases no balanceadas
Piczak [12]	ESC - 50	2000	30	≤5 s	Contiene la metadata, clases balanceadas
Piczak [12]	ESC - 10	400	10	≤5 s	Contiene la metadata, clases balanceadas
Piczak [12]	ESC – US	250.000	-	≤5 s	Conjunto sin etiquetar
Koizumi *et al* [13]	DCASE Task 2	8	-	≤10 s	Sonidos mecánicos anómalos
Cao *et al*. [14]	CREMA-D	7442	-	≤10 s	Una selección de 12 oraciones con emociones
Gemmeke *et al.* [15]	AudioSet	+2 Millones	632	≤10 s	Una gran colección de sonidos organizados por categorías
Mesaros et al. [16]	TUT Sound Event	24	15	≤15 s	Separado en pequeñas muestras
Rachman *et al.* [17]	MIREX	903	15	-	Conjunto de datos de estados de ánimo para

					clasificación de emociones
Fonseca *et al*. [18]	FSD50K	51.197	200	≤30 s	Eventos de sonido etiquetados

A continuación, se profundizará brevemente en los conjuntos UrbanSound8K, ESC – 50 y AudioSet, para entender de mejor manera como se organizan.

- UrbanSound8K:

Este conjunto de datos contiene 8732 sonidos etiquetados con duración menor a 4 segundos, las posibles etiquetas bajo las que están clasificados son 10: air_conditioner, car_horn, children_paying, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren y street_music. Todos los extractos son grabaciones en terreno subidos a Freesound [19]. Los archivos están preordenados dentro de 10 carpetas (con nombres fold1 a fold10). Además, el conjunto contiene un archivo CSV en donde se encuentra la metadata de cada extracto, la figura 2.7 muestra un extracto de esta metadata, la cual en general contiene el nombre del archivo, el directorio en donde encontrarlo, la clase a la que pertenece, entre otras cosas.

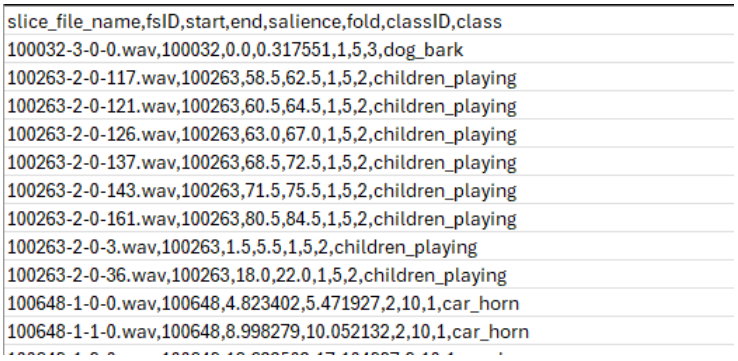


Figura 2.7: Extracto metadata de los archivos encontrados en el conjunto UrbanSound8K

- ESC – 50:

Es una colección de 2000 grabaciones de audio etiquetados dentro de 50 clases distintas, las que a su vez se pueden juntar en 5 mayores categorías. Estas se muestran en la figura 2.8.

Los extractos encontrados en este conjunto fueron manualmente extraídos de Freesound [19]. El conjunto se contiene dentro de 1 carpetas y se acompaña de un CSV que contiene la respectiva metadata, la figura 2.9 muestra un extracto de ésta metadata.

Animals	Natural soundscapes & water sounds	Human, non-speech sounds	Interior/domestic sounds	Exterior/urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

Figura 2.8: Organización de categorías encontradas en ESC – 50

```
filename,fold,target,category,esc10,src_file,take
1-100032-A-0.wav,1,0,dog,True,100032,A
1-100038-A-14.wav,1,14,chirping_birds,False,100038,A
1-100210-A-36.wav,1,36,vacuum_cleaner,False,100210,A
1-100210-B-36.wav,1,36,vacuum_cleaner,False,100210,B
1-101296-A-19.wav,1,19,thunderstorm,False,101296,A
1-101296-B-19.wav,1,19,thunderstorm,False,101296,B
1-101336-A-30.wav,1,30,door_wood_knock,False,101336,A
1-101404-A-34.wav,1,34,can_opening,False,101404,A
1-103298-A-9.wav,1,9,crow,False,103298,A
1-103995-A-30.wav,1,30,door_wood_knock,False,103995,A
1-103999-A-30.wav,1,30,door_wood_knock,False,103999,A
1-104089-A-22.wav,1,22,clapping,False,104089,A
1-104089-B-22.wav,1,22,clapping,False,104089,B
```

Figura 2.9: Extracto metadata de los archivos encontrados en ESC – 50

- AudioSet:

Es un conjunto de datos constantemente en expansión, el cual en este momento contiene 2.084.320 clips de audio etiquetados a mano y clasificado en 632 clases distintas extraídos de videos de YouTube. La figura 2.10 muestra la organización de algunas de las posibles clases que se pueden encontrar dentro de este conjunto.

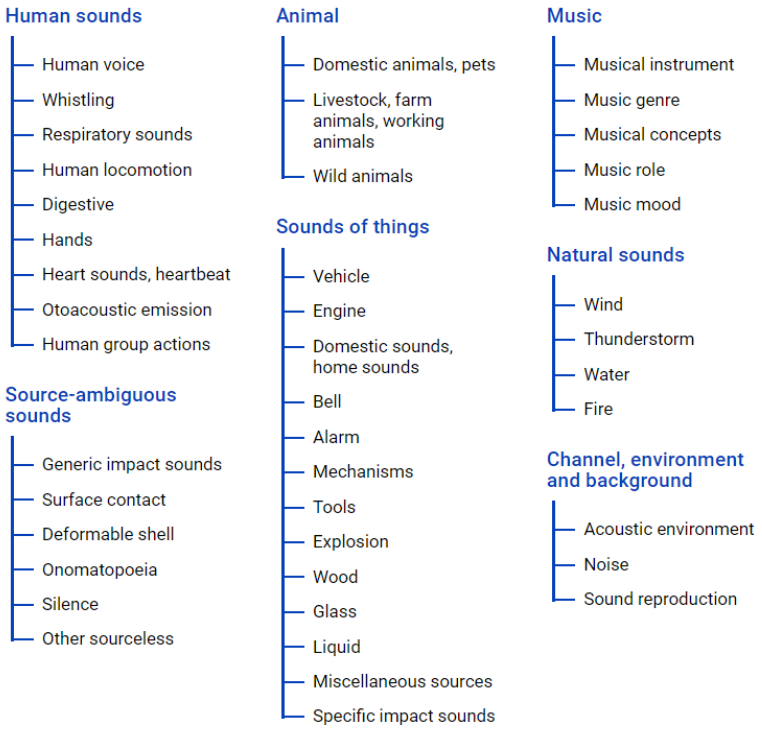


Figura 2.10: Organización principales clases dentro de AudioSet

2.3.2. Preprocesamiento del audio

Una vez en la etapa de clasificación, cada uno de los archivos será tratado como un vector. Para que la clasificación funcione correctamente es necesario que cada uno de estos vectores, con los que se alimenta un modelo de aprendizaje, tengan una forma definida, por lo que una parte esencial cuando se trabaja en la creación de algoritmos de clasificación usando alguna técnica de aprendizaje profundo es el preprocesamiento de datos. Esta acción consiste en preparar todo el conjunto de datos bajo un criterio específico para así facilitar procesos posteriores y agilizarlos. Generalmente consiste en definir el tamaño de los archivos, el número de canales, la frecuencia de

muestreo, el formato del audio. Se escoge el formato WAVE debido a no varía mucho la grabación del sonido original conservando así las características originales, entre otros. Algunas de las acciones que se llevan a cabo en este nivel son:

- Preparar los archivos: es necesario tener los archivos ordenados y crear una lista con el nombre de cada archivo y la clase a la que pertenece, tal como muestran las figuras 2.7 y 2.9.
- Re-muestreo y definición del número de canales: Las características que definen un archivo de audio son la frecuencia de muestreo y el número de canales usados, siendo uno (mono) o dos canales (estéreo) las opciones más frecuentes.
- Cambiar la duración de los archivos: Es necesario que cada archivo tenga la misma duración, para lo que si el archivo tiene una duración mayor a la establecida entonces simplemente basta con cortar la parte sobrante. Por el contrario, si el archivo tiene menor duración entonces es necesario aumentarla.

2.3.3. Extracción de características

En este nivel los audios son procesados con el fin de obtener algún tipo de representación característica de cada uno, por lo que, en general, las técnicas de extracción de características suelen basarse en obtener las componentes en frecuencia. Basado en las revisiones de literatura de Rodrigues *et al.* [10] y Abayomi-Alli *et al.* [20], a continuación, se explicarán los métodos más usados con este propósito.

- Transformada de Fourier en periodos cortos (STFT):

  Cómo el nombre lo indica, este método consiste en dividir un audio en pequeños segmentos de igual largo y calcular la transformada de Fourier sobre cada uno de ellos. Luego, se puede graficar el cambio en el espectro respecto al tiempo usando un espectrograma. Dicho lo anterior, la STFT en tiempo descrito se define según la ecuación 2.14, en donde $x[m]$ es el audio y $w[m]$ es una ventana. Jeon et al. [21], en la figura 2.11, nos da una visión general de este proceso.

$$X_{STFT}[k,n] = \sum_{m=0}^{N-1} x[m]\, w[m-k] e^{-inm} \tag{2.14}$$

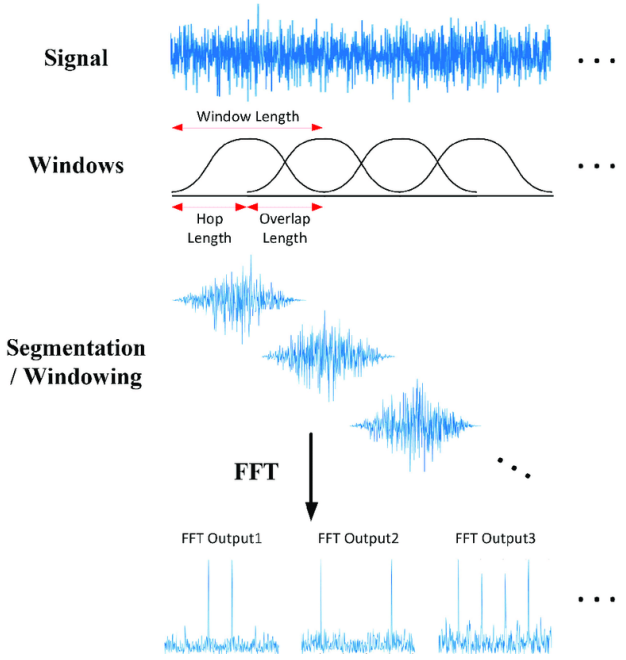


Figura 2.11: Vista general de cómo obtener la transformada de Fourier en periodos cortos, en tiempo discreto

- Espectrograma de Mel

La percepción del oído humano respecto a la variación de frecuencias de un sonido es logarítmica, por lo que notamos mejor la variación entre frecuencias bajas, mientras que la variación entre frecuencias altas es casi imperceptible. La ponderación de frecuencias tipo A es un intento de simular esta percepción al hacer mediciones del nivel de presión sonora. Otro método es usar la escala de Mel, en donde la conversión de la frecuencia a esta escala se hace según la ecuación 2.15.

$$mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.15}$$

Entonces, el espectrograma de Mel consiste en usar la STFT para obtener las componentes de frecuencia respecto al tiempo y luego estas frecuencias se convierten a

escala de Mel y se representan como un espectrograma.

- Log-Mel

Consiste en pasar la dimensión de la amplitud en el espectrograma de Mel a una escala logarítmica, para lo que primero se calcula la potencia (el cuadrado de la amplitud) y luego se le aplica el logaritmo en base 10, tal como muestra la ecuación 2.16, en dónde A_{mel-n} es la amplitud para el n-ésimo índice de la STFT. Esto aporta una mayor definición de las características principales de una señal.

$$\log\_mel_n = 20 \log_{10}(A_{mel-n}) \tag{2.16}$$

- Coeficientes Cepstrales de Frecuencia de Mel (MFCC)

La transformada del coseno discreta, DCT, se basa en la transformada de Fourier discreta, pero utiliza únicamente números reales, una de las características de la DCT es su gran capacidad de compactación de energía, es decir, consigue concentrar la mayor cantidad de información en unos pocos coeficientes. La figura 2.12 muestra un ejemplo grafico de esto. La ecuación 2.15 muestra la definición más típicamente usada para la DCT en donde L es el número de coeficientes que se desea obtener.

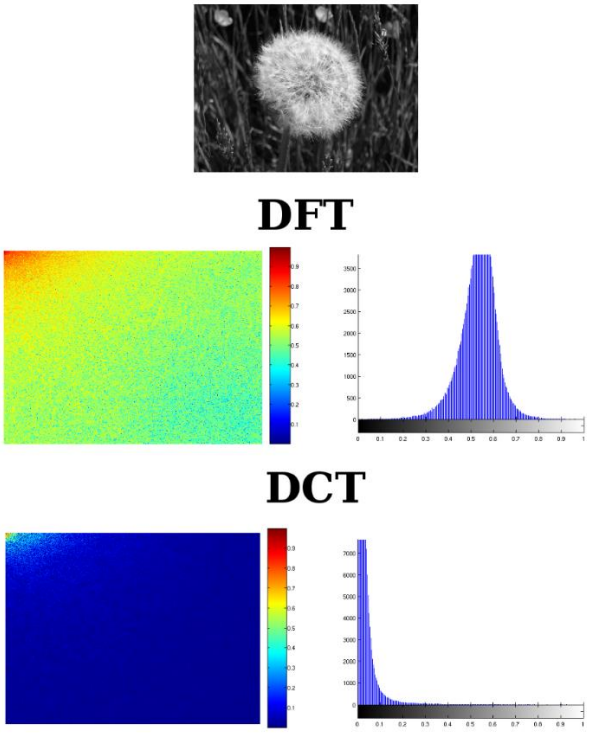


Figura 2.12: Comparación DFT versus DCT

$$DCT - II = \sum_{l=0}^{L} \mathrm{x_n} * \cos\left(\frac{\pi n(2l+1)}{2L}\right) \tag{2.15}$$

Entonces, los MFCC se consiguen al aplicar la transformada del coseno discreta al espectrograma de Mel en escala logarítmica, o Log-Mel, tal como muestra la ecuación 2.16

$$MFCC_n = \sum_{l=0}^{L} \log\_mel_n * \cos\left(\frac{\pi n(2l+1)}{2L}\right) \tag{2.16}$$

Actualmente, los MFCC son los que entregan mejores resultados en el ámbito de la clasificación de audio, siendo a veces usados junto a otros métodos [10].

2.3.4. Aumento de datos

Cómo ya se ha mencionado, uno de los mayores problemas a la hora de crear modelos de clasificación de sonido es la falta de conjunto de datos. Si bien existen conjuntos disponibles, este problema es más notorio cuando se quiere clasificar sonidos menos comunes en donde los datos son más escasos. Otro problema que se presenta por la poca variedad en los conjuntos de datos es la sobrealimentación, que provoca que los modelos entreguen buenos resultados sobre el conjunto de datos con los que fueron entrenados, pero que tengan un desempeño pobre cuando se usan en datos externos. Es en este contexto en donde surgen las técnicas para aumento de datos o "data augmentation", las cuales permiten aumentar la cantidad y variedad de datos de un conjunto reducido, mejorando así el resultado al usar dichos datos para entrenar modelos de clasificación. A continuación, se listan algunos de los métodos más usados según la literatura [10] [20]. Notar que la adición de ruido, el desplazamiento temporal, desplazamiento del tono y estiramiento temporal se aplican antes de la extracción de características, mientras que el enmascaramiento en tiempo y frecuencia se realizan después de la extracción de características. Además, se puede usar una mezcla de estos métodos para obtener mejores resultados.

- Adición de ruido: Consiste en mezclar el sonido con ruido, comúnmente ruido blanco (figura 2.13).
- Desplazamiento temporal: Las características del audio se mantiene, pero se desplaza, o rota, en el eje temporal (figura 2.14).
- Desplazamiento del tono: Se disminuye o aumenta el tono del audio manteniendo su duración (figura 2.15).
- Estiramiento temporal: Se modifica la duración del audio, extendiéndolo o acortándolo, el tono se mantiene igual (figura 2.16).
- Enmascaramiento en la frecuencia: De forma aleatoria, se enmascara un rango consecutivo de frecuencias añadiendo barras verticales en el espectrograma (figura 2.17).
- Enmascaramiento en el tiempo: De forma aleatoria, se enmascara un rango consecutivo de tiempo añadiendo barras horizontales en el espectrograma (figura 2.17).

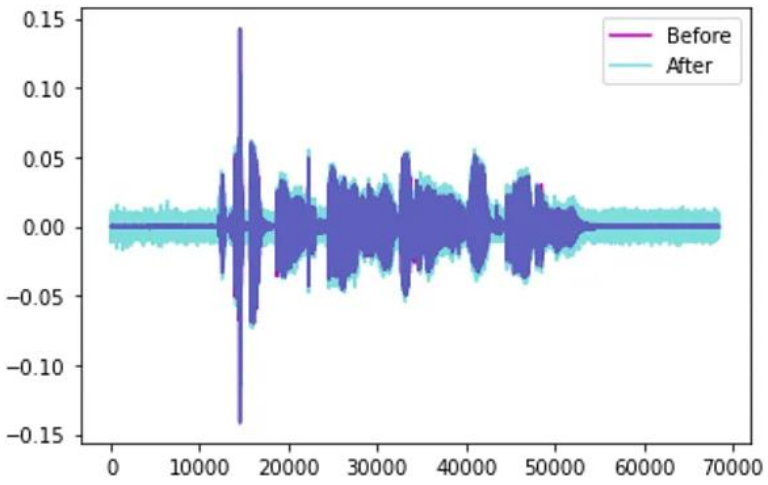


Figura 2.13: Aumento de datos, ejemplo de adición de ruido

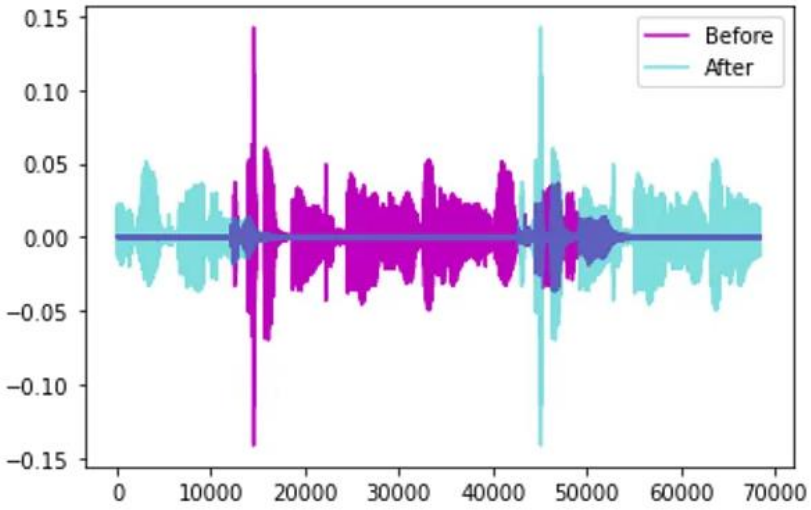


Figura 2.14: Aumento de datos, ejemplo de desplazamiento temporal

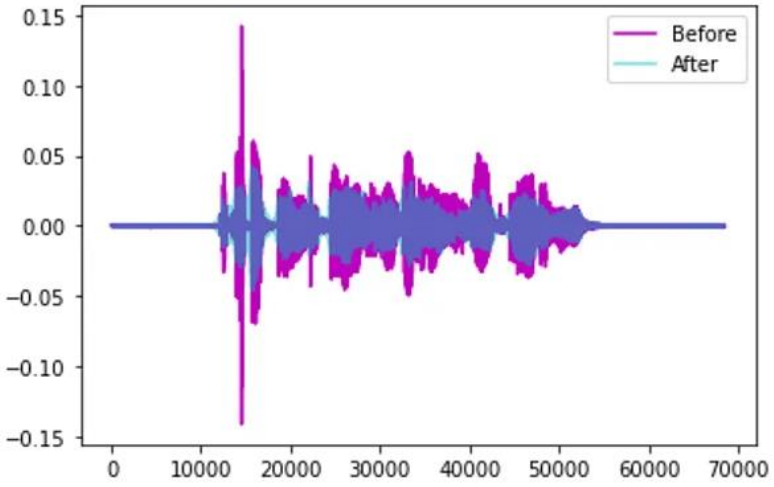


Figura 2.15: Aumento de datos, ejemplo de desplazamiento del tono

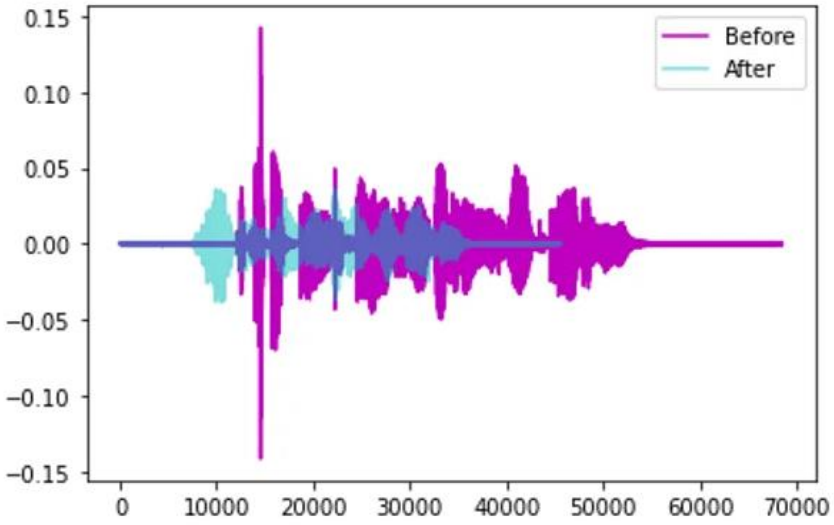


Figura 2.16: Aumento de datos, ejemplo de estiramiento temporal

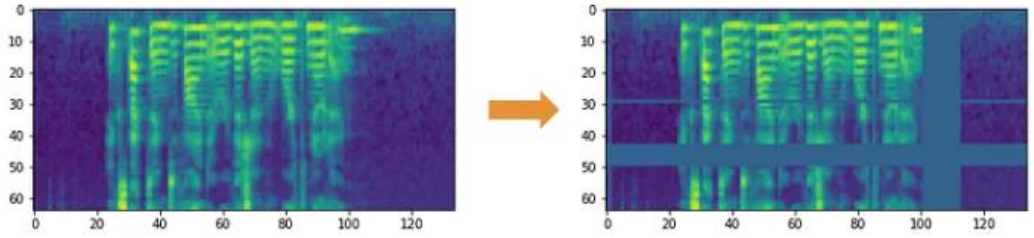


Figura 2.17: Aumento de datos, ejemplo de enmascaramiento den tiempo y frecuencia

2.3.5. Clasificación

Podemos definir clasificar, a grandes rasgos, cómo organizar algo por clases, en donde cada clase comparte un conjunto de características. En el campo del aprendizaje profundo el problema de la clasificación ha sido ampliamente abordado dada la diversidad de aplicaciones que los modelos de clasificación tienen, como por ejemplo en el campo de la visión por computadora. Producto de esto nace una gran cantidad de diferentes arquitecturas, cada una con diferentes ventajas y desventajas, Zaman *et al.* [23] presenta una revisión de las arquitecturas de redes de aprendizaje profundo más comunes en el campo de la clasificación de audio, siendo: Redes Neurales Convolucionales (CNN), Redes Neurales Recurrentes (RNN), Autoencoders, Transformers y modelos híbridos.

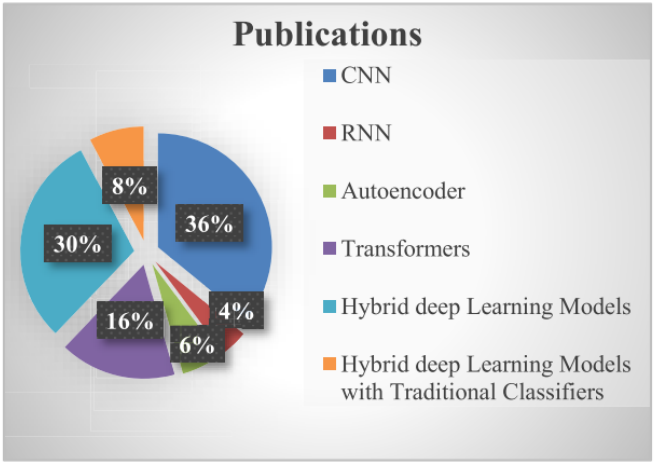


Figura 2.18: Proporción entre las distintas arquitecturas de aprendizaje profundo para clasificación de audio

Las redes neurales convolucionales generalmente están compuestas de múltiples capas convolucionales, una unidad de rectificación lineal (ReLU), capas de pooling (o submuestreo), capas de redes neurales totalmente conectadas y una capa Softmax tal como muestra la figura 2.19. La ventaja de este tipo de arquitectura es que permite tomar una entrada, generalmente 2D, y mediante la convolución con un núcleo se extraen sólo las características más relevantes, reduciendo así el número de parámetros a entrenar y por lo tanto haciendo que el proceso sea más rápido.

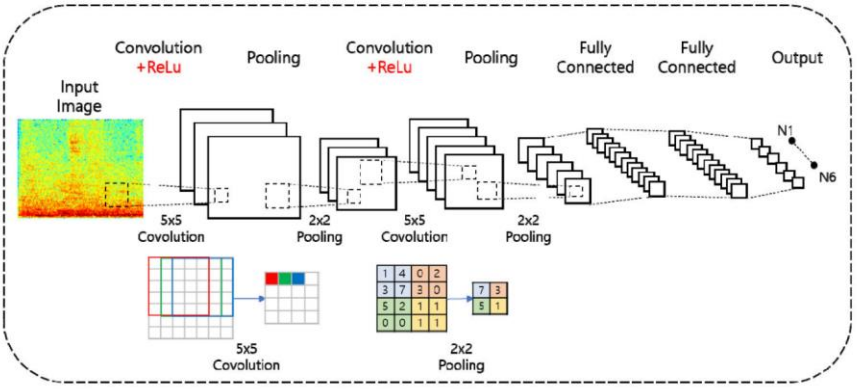


Figura 2.19: Arquitectura general de una red neural convolucional

Las redes neurales recurrentes se caracterizan (RNN) por su capacidad de extraer características temporales en largas secuencias de datos usando recurrencia. Uno de los principales problemas de las redes neurales es su corta memoria pues aprenden de características presentes en pequeños tramos. Esto hace que las RNN sean una herramienta muy útil en tareas que dependan de extraer el contexto de los datos, la figura 2.20 muestra el diagrama de una arquitectura de tipo memoria a corto-largo plazo o *Long Short-Term Memory* (LSTM) la cual es el tipo más representativo de redes neurales recurrentes. Esta consta de varias puertas o válvulas las cuales permiten mantener relaciones importantes de los datos en el tiempo. Primero está la compuerta *forget gate*, la cual permite decidir qué información se va a descartar. Luego se encuentra la compuerta de entrada, la cual permite añadir información nueva que se desea recordar y finalmente está la compuerta de salida, en la cual se obtienen los valores que entrarán en la siguiente capa.

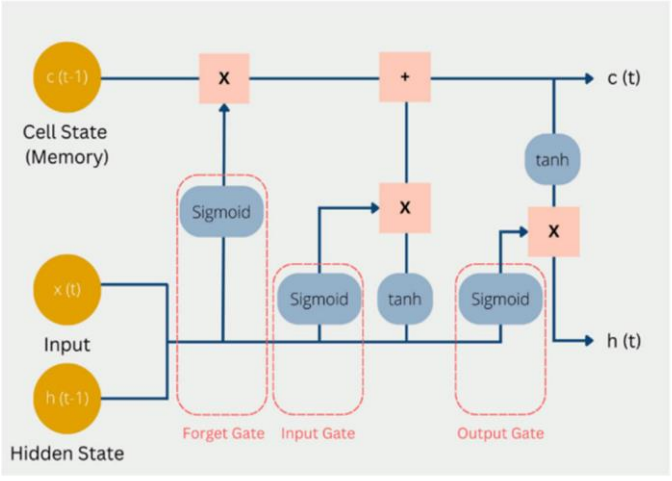


Figura 2.20: Arquitectura de una red de memoria a corto-largo plazo (LSTM)

Los Autoencoder son una técnica de aprendizaje no supervisado (clases sin etiquetar) la cual usa redes neurales en datos comprimidos. Constan de dos pasos (figura 2.21). Primero un codificador transforma la entrada en una representación de ésta de menor dimensión. Luego un decodificador recrea la salida a partir de los datos codificados.

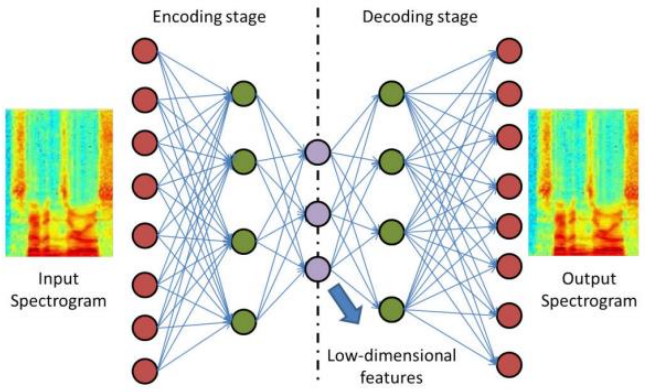


Figura 2.21: Arquitectura de un autoencoder

Los Transformers son un tipo de red neural profunda originalmente diseñados para técnicas de procesamiento del lenguaje natural y luego aplicados a imágenes y por lo tanto también a audio. Estos requieren grandes cantidades de datos para ser entrenados, pero a diferencia de las redes neurales convolucionales estos pueden manejar entradas de largo variable. La figura 2.22 muestra a rasgos generales la arquitectura de un transformer en donde se observa que la entrada es un espectrograma divido en "parches" de tamaño fijo y cada uno de estos es tratado como una entrada particular, como si fueran palabras en una oración en un modelo de procesamiento de lenguaje natural.

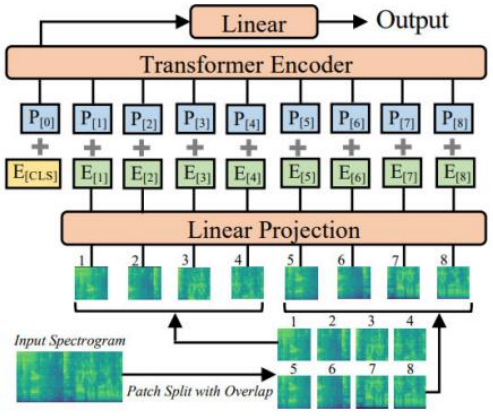


Figura 2.22: Arquitectura de un transformer

Por ultimo las arquitecturas hibridas, como el nombre lo indica, están compuestas por dos o más arquitecturas distintas, por ejemplo, muchos métodos de redes neurales recurrentes usan primero redes neurales convolucionales para extraer características espaciales a partir de espectrogramas y luego la red recurrente se encarga de identificar patrones en la salida de la red convolucional.

La figura 2.18 muestra un resumen de la proporción entre las arquitecturas antes mencionadas encontradas en los estudios revisados por Zaman *et al.* [23]. Como se puede ver, un 36% es representado por arquitecturas basadas en redes neurales convolucionales, otro porcentaje importante es representado por arquitecturas hibridas, y entre estas arquitecturas hibridas es común también encontrar redes neurales convolucionales. Éstas, gracias a una serie de ventajas, son actualmente el estado del arte en la clasificación de audio, es por esto por lo que a continuación se procederá a explicar con un poco más de detalle su funcionamiento y las capas que la conforman.

- Redes Neurales Convolucionales:

  Como se mencionó antes, una de las ventajas de las redes convolucionales es su capacidad para extraer aquella información que es relevante en el proceso de aprendizaje [26], reduciendo así la cantidad de información a procesar y por lo tanto aumentando la eficiencia. La figura 2.23 muestra de forma gráfica la arquitectura de una CNN comúnmente usada para clasificación de imágenes la cual consiste primero de numerosas capas de convolución + ReLU, seguidas de capas de submuestreo (pooling) para terminar con capas de redes neurales totalmente conectadas.

  La entrada a la red convolucional está generalmente organizada en tres dimensiones, altura, anchura y profundidad, o $m\ x\ m\ x\ r$, donde la altura y el ancho son iguales (m) y la profundidad es generalmente conocida como el numero de canales, por ejemplo, en una imagen RGB el número de canales, r, es 3.

  El kernel (o filtro) de la convolución se denota por k y sus dimensiones son $n\ x\ n\ x\ q$, donde $n < m$ y $q \leq r$. El proceso comienza por tomar la imagen original y convolucionarla con el kernel, dando como origen un mapa de características h para luego aplicar una función de activación, generalmente una función ReLU.

  El siguiente paso consiste en el submuestreo de cada mapa de características obtenido con cada kernel, esto tiene como resultado una disminución del número de parámetros en la red y por lo tanto aumenta la velocidad del proceso de entrenamiento.

La función de submuestreo (o pooling) es aplicada sobre cada mapa de características en áreas adyacentes de tamaño $p\ x\ p$, donde p es el tamaño del kernel.

La siguiente capa consiste en una red neural totalmente conectada que recibe las características obtenidas en la capa de submuestreo y crea una abstracción de alto nivel de estas.

La capa final de la red se encarga de convertir estas abstracciones en puntuaciones, donde cada puntuación representa la probabilidad de que la correspondiente entrada corresponda a cada una de las posibles clases.

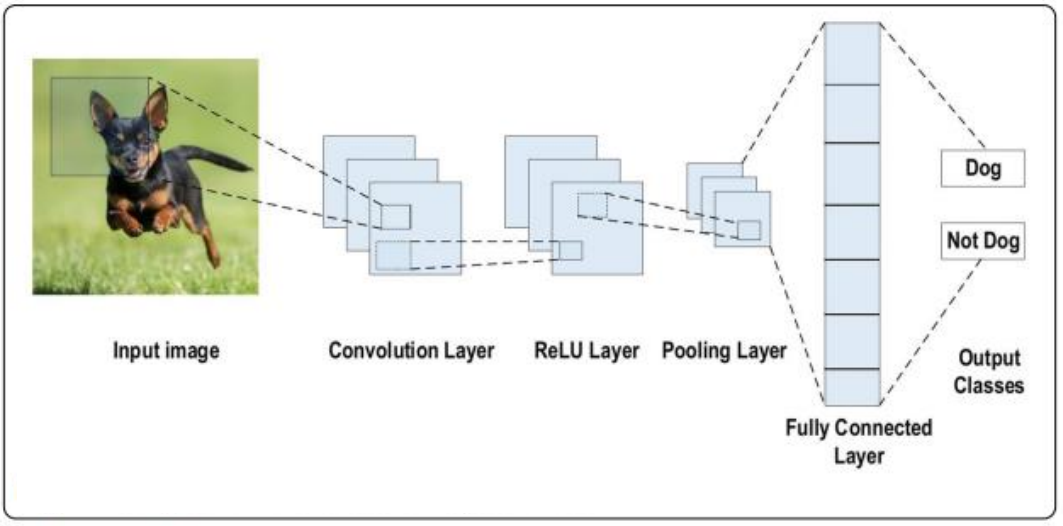


Figura 2.23: Ejemplo de una arquitectura CNN para clasificación de imágenes

- Capa de convolución

Esta es la capa más significativa en una red neural convolucional. Está compuesta por una serie de diferentes filtros o kernels los cuales en la primera iteración del proceso de entrenamiento reciben números aleatorios. Estos valores serán ajustados durante este proceso en cada iteración. A medida que estos valores se ajustan, los kernels aprenden a extraer características más importantes para el proceso de clasificación. Un ejemplo de la operación de convolución se muestra en la figura 2.24, en donde la entrada tiene dimensiones $4\ x\ 4\ x\ 1$ y el kernel tiene dimensiones $2\ x\ 2\ x\ 1$ y sus valores fueron inicializados de forma aleatorio. Primero el kernel se desplaza sobre la entrada sobre todo el eje vertical y horizontal y se calcula el producto punto entre el kernel y la porción

correspondiente de la entrada dando como resultado un escalar. Tras desplazar el kernel sobre toda la entrada, los resultados de cada producto punto conforman el mapa de características de la salida.

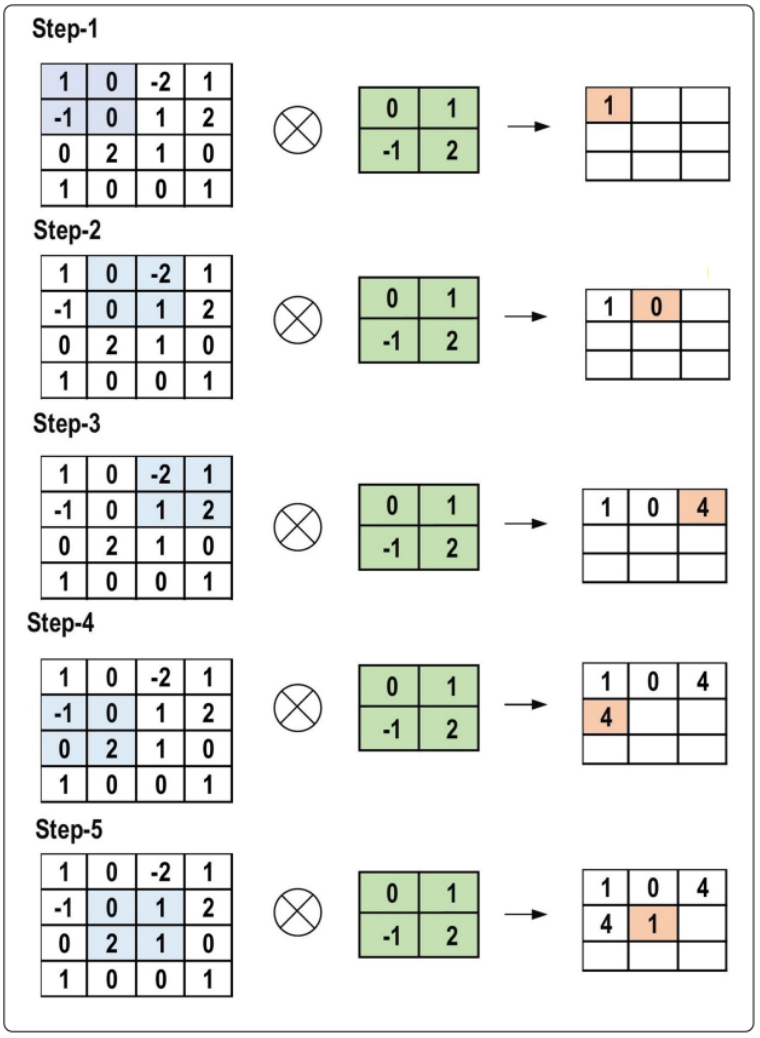


Figura 2.24: Ejemplo operación de convolución de una matriz con un núcleo dado

- Capa de submuestreo o pooling

En esta capa los mapas de características obtenidos en la capa de convolución son procesados para reducir sus dimensiones mediante un submuestreo creando así mapas de características de tamaños más reducidos, pero manteniendo las características más relevantes obtenidos en la capa anterior. Existen diferentes métodos de submuestreo, pero

los más comunes son el de medias, el de máximos y el de media global. Cada uno de estos métodos es gráficamente descrito en la figura 2.25.

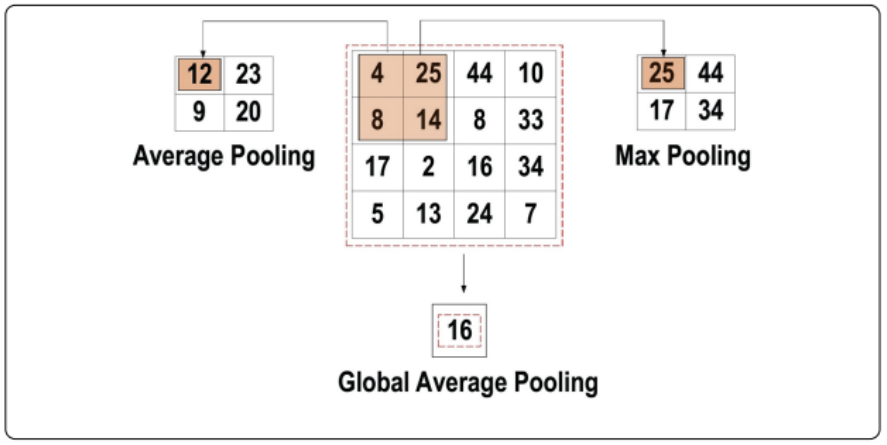


Figura 2.25: Tres tipos de operaciones de pooling

- Función de activación

La función de activación relaciona de forma no lineal la entrada de una neurona con su salida, esto es esencial para que la red pueda aprender patrones más complejos. El cálculo de la entrada se hace mediante una suma ponderada de todas las entradas a esa neurona, esto es, cada entrada se multiplica por un peso y es sumada a todas las entradas anteriores. Luego, esta entrada es pasada a la función de activación para calcular la salida. Las funciones de activación más comunes son la función sigmoide (ecuación 2.17) la cual convierte la entrada en valores entre cero y uno, la función tangente hiperbólica (ecuación 2.18) la cual restringe la entrada a valores entre -1 y 1, y la función ReLU (ecuación 2.19) la cual es la más usada en el contexto de las redes convolucionales. Esta retorna un cero si la entrada es negativa y retorna la entrada misma si ésta es mayor o igual a cero.

$$f(x)_{sign} = \frac{1}{1 + e^{-x}} \tag{2.17}$$

$$f(x)_{tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.18}$$

$$f(x)_{ReLU} = \max(0, x) \tag{2.19}$$

- Capa totalmente conectada

Generalmente localizada al final de cada CNN, cada neurona dentro de esta capa está conectada a todas las neuronas de la capa previa. Su función es tomar los mapas de características extraídos por las capas anteriores y usarlos para calcular la probabilidad de que la entrada pertenezca a cada una de las posibles categorías presentes. Para lo anterior, los mapas de características se aplanan antes de entrar a esta capa, proceso conocido como *flattening*. Por ejemplo, si la última capa de pooling produce 64 mapas de características de 7x7, entonces la entrada será un vector unidimensional de 64x7x7=3136 elementos. Cada uno de estos elementos se conecta a cada una de las neuronas de entrada de la capa totalmente conectada multiplicándose por un peso específico. La figura 2.26 resume este proceso.

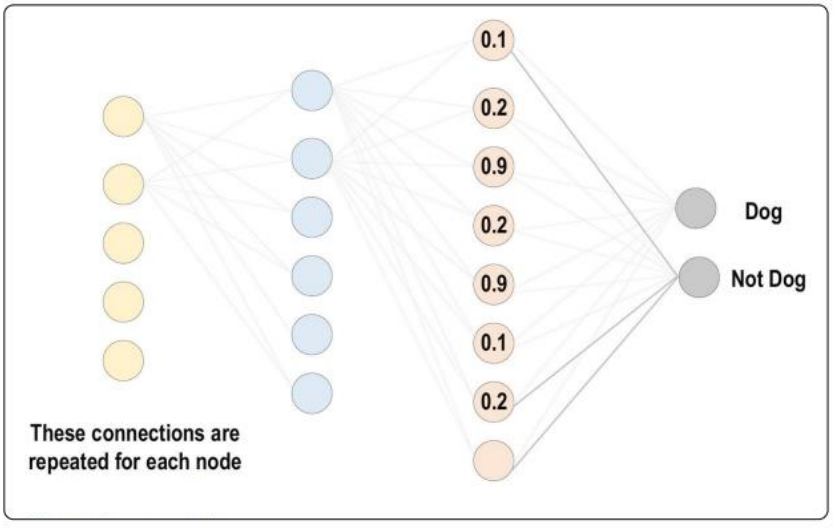


Figura 2.26: Esquema de una capa totalmente conectada

- Función de perdida

Para que el modelo aprenda, es necesario monitorear el error en cada iteración del proceso de entrenamiento. Entonces, la función de perdida (*loss function*) compara las predicciones hechas en cada iteración de este proceso con los valores reales obteniendo su diferencia. Existen varias posibles opciones para usar con este fin. Sea p_i la probabilidad de que la salida pertenece a una clase, definida por la ecuación 2.20, en donde e^{a_i} corresponde al valor de salida de una neurona y N el número total de neuronas en la capa de salida, podemos encontrar entonces la función de perdida de entropía cruzada en la que el error se calcula según la ecuación 2.21, la cual compara la probabilidad obtenida por la función softmax con la probabilidad real, y_i, de pertenecer a la clase i (toma valor 0 o 1).

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{2.20}$$

$$H(p, y) = -\sum_i y_i \log(p_i) \ \ con\ i\ \in [1, N] \tag{2.21}$$

2.4. Geolocalización

Actualmente, obtener la geolocalización de un dispositivo con alta precisión es común gracias al Sistema de Posicionamiento Global (GPS). Este sistema, presente en casi todos los dispositivos móviles, permite determinar la ubicación con un margen de error pequeño. El GPS funciona enviando señales desde al menos cuatro satélites a un dispositivo receptor y calcula la ubicación basándose en el tiempo que tardan en llegar las señales. Según la guía de desarrolladores de Android [29], los servicios de ubicación más precisos que proporciona Google suelen tener un margen de error de hasta 50 metros, aunque en la mayoría de los casos, el error es mucho menor, a menudo alrededor de 5 a 10 metros en condiciones óptimas.

Además, otros métodos de geolocalización incluyen la triangulación de señales de torres de telefonía móvil y la determinación de la ubicación a través de la dirección IP de una red Wi-Fi. Estos métodos, aunque menos precisos que el GPS, pueden proporcionar ubicaciones aproximadas cuando el GPS no está disponible o tiene una señal débil.

Capítulo 3: Desarrollo

La iniciativa del desarrollo de esta herramienta fue por parte de R9 Ingeniería [30], empresa especializada en el ámbito de la gestión ambiental en donde se desenvuelve desarrollando soluciones de software y brindado diferentes servicios, con un especial enfoque en el monitoreo de la calidad del aire. R9 se encuentra desarrollando una plataforma innovadora centrada en la gestión de incidencias ambientales la cual busca darles a los usuarios comunes una vía para reportar eventos ambientales molestos de origen no natural. Al momento, esta plataforma cuenta con la funcionalidad para reportar incidentes relacionados a malos olores a través de una plataforma web. El objetivo de este proyecto es integrar la herramienta de caracterización de ruido a ella para ser usada como un medio para reportar eventos de ruido. Esta integración consiste en añadir una sección para quejas de ruido en la plataforma ya existente por la cual acceder a la herramienta. Además, se desarrolla una aplicación para sistemas operativos Android mediante la que acceder a la herramienta.

3.1. Arquitectura de la solución

El diseño general de la herramienta de caracterización de ruido se puede observar en la figura 3.1. A rasgos generales la arquitectura consiste primero de una entrada/salida en donde el usuario puede ingresar reportes y visualizar información acerca de estos. Los reportes ingresados se procesan y luego almacenan en diferentes bases de datos para luego usarlos en la generación de reportes, alarmas, resúmenes, entre otros. Cada uno de los diferentes componentes de esta arquitectura es incorporado dentro de la plataforma de incidencias ambientales de R9 ingeniería y se presenta a continuación.

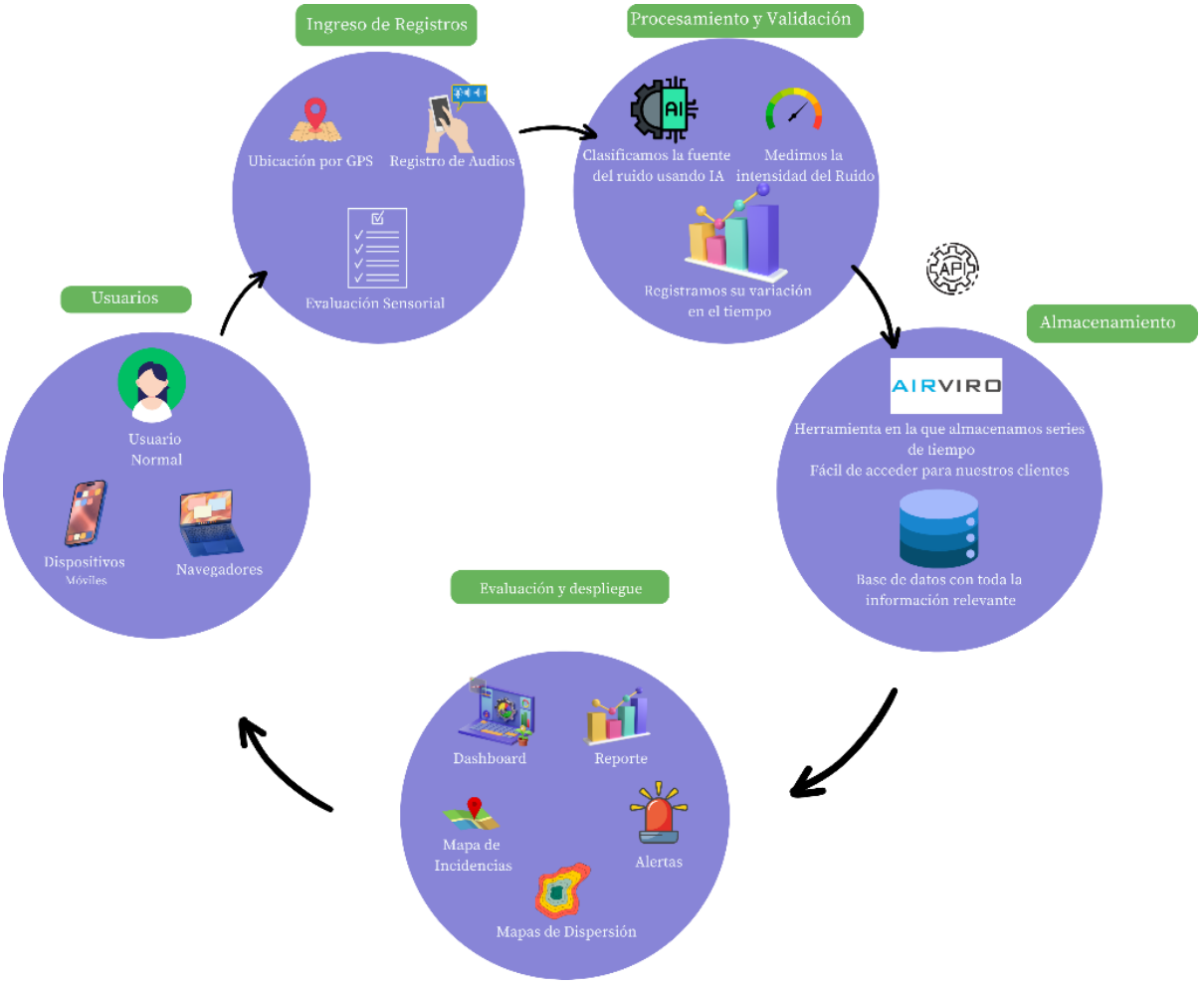


Figura 3.1: Diseño general de la herramienta para medición de ruido

3.1.1. Usuarios

Esta capa corresponde a las interfaces de usuario de la herramienta, tanto en versión móvil cómo en versión web. Aquí los usuarios pueden administrar su perfil, revisar registros de incidentes hechos por otros usuarios, recibir alertas y realizar sus propios registros. Tanto la aplicación móvil como la aplicación web tienen la opción para crear nuevos registros y ambas cuentan con un mapa en donde se muestran todos los registros hechos en un rango de tiempo junto a información básica de cada registro. La versión web además cuenta con información más detallada de cada registro y es la encargada de mostrar reportes hechos en base a la información recolectada y de generar notificaciones con alarmas.

3.1.2. Ingreso de registros

Al realizar un registro de ruido la información necesaria es:

- Ubicación: Corresponde al punto geográfico en el cual se están registrando los datos. Éste se obtiene accediendo a servicios de ubicación disponibles para cada uno de los dispositivos antes mencionados permitiendo así una geolocalización precisa del punto en el que el evento de ruido está ocurriendo.
- Evaluación sensorial: Corresponde a la evaluación subjetiva del usuario sobre el evento de ruido entregada al momento de subir el registro. Esta información se recolecta mediante el uso de una encuesta diseñada para ser fácil y rápida de responder y para recolectar toda la información que pueda ser relevante para un posterior análisis del evento.
- Registro de audio: Corresponde al audio registrado haciendo uso del micrófono del dispositivo usado. Este registro es guardado con extensión .wav.

Ambas plataformas, web y móvil, cuentan con las funcionalidades necesarias para la obtención de estos datos.

3.1.3. Procesamiento

La figura 3.2 muestra de forma gráfica en qué consiste esta etapa. Aquí el audio capturado en la etapa de registro y procesado para obtener la intensidad y la clasificación en relación con el tiempo, creando así dos series de tiempo, una con los valores de intensidad equivalente calculados según el algoritmo 1, y los valores de la clasificación del audio en cada segundo. Para clasificar se usó un modelo de clasificación de audio pre entrenado y de uso libre desarrollado por Google llamado YAMNet [28].

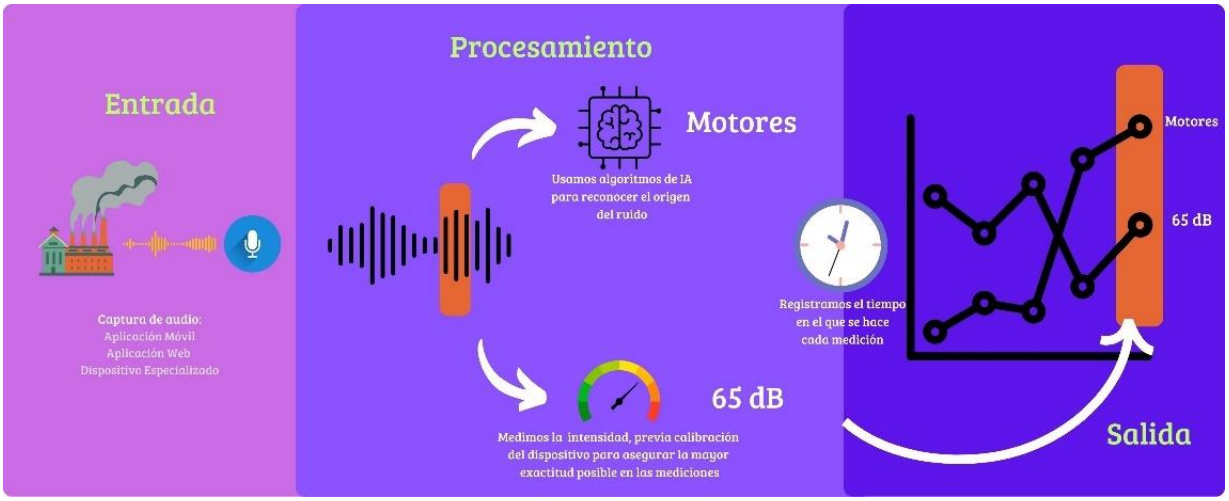


Figura 3.2: Procesamiento de audio

YAMNet es un modelo pre entrenado que clasifica entre 521 clases distintas. El conjunto de datos usado para su entrenamiento fue el AudioSet Corpus [15]. Toma como entrada un audio en formato WAVE muestreado a 16kHz en un solo canal y obtiene el espectrograma computando la STFT en intervalos de 25 ms para luego calcular 64 coeficientes cepstrales de Mel (MFCC) para frecuencias entre 125 a 7500 Hz. La arquitectura usada por YAMNet es una CNN cuyo diseño se puede ver en detalle en un resumen entregado por MathWorks [27].

3.1.4. Almacenamiento

En esta etapa el usuario envía la información recolectada durante la etapa previa a los servidores de R9. La información guardada se divide en dos lugares distintos

- Airviro: Herramienta que permite el almacenamiento y manejo de grandes volúmenes de información, conectado a distintas redes de monitoreo, permite centralizar el almacenamiento de series de tiempo y facilita su posterior análisis, aquí guardamos las series de tiempo obtenidas como salida en la etapa de procesamiento, es decir, las clasificaciones y la intensidad respecto al tiempo. En una serie de tiempo de Airviro cada punto o entrada de tiempo está asociado a tres valores, `timestamp`, `value` y `statusCode`, los que indican el tiempo en formato de estampas de tiempo de unix, el valor, y un código asociado a la validez del dato, respectivamente. Para aprovechar esta estructura y no crear dos series de tiempo distintas, se asociará cada clase posible del modelo a un valor numérico, este se guardará en la variable `statusCode`.

- Base de datos SQL: Aquí se almacena toda la información asociada a cada evento registrado, es decir, fecha, hora, ubicación, usuario, además de las respuestas a la encuesta de percepción del usuario.

La aplicación móvil se comunica con ambos medios de almacenamiento a través de una API Rest. Por otra parte, la aplicación web se aloja en los mismos servidores que ambas bases de datos por lo que la comunicación es directa.

3.1.5. Evaluación y despliegue

Los datos almacenados en ambas bases de datos son consumidos en la generación de reportes y alertas. La aplicación móvil muestra un mapa con los registros hechos junto a información básica de estos. En la aplicación web, además del mapa en donde ubicar los registros de eventos, también cuenta con una serie de reportes hechos en base a la información recopilada e información de carácter técnico de cada evento en particular.

3.2. Aplicación móvil

La aplicación se desarrolló para sistemas operativos Android, usando el lenguaje de programación Kotlin y usando el entorno de desarrollo Android Studio. El anexo A muestra todas las vistas de la aplicación. La figura 3.4 muestra la interacción entre la aplicación móvil, las bases de datos y la aplicación web. Como se puede ver, el procesamiento del audio se hace en el dispositivo para luego subir los datos a sus respectivas bases de datos y desde ahí ser consumidos. El anexo B muestra las funciones implementadas para el cálculo de la intensidad según el algoritmo 1. El anexo C muestra la función implementada para ejecutar el modelo de clasificación. Para esto se usó TensorFlow Lite [31] y una versión optimizada para dispositivos móviles de YAMNet disponible en el hub de Tensorflow, Kaggle [28].

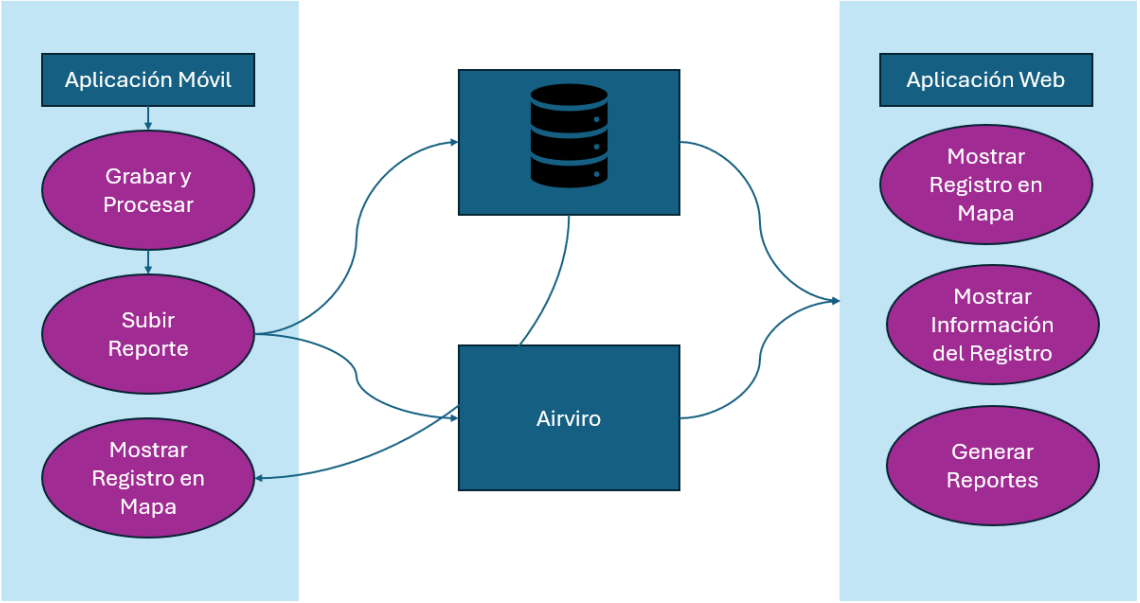


Figura 3.3: interacción aplicación móvil - base de datos – aplicación web

3.3. Aplicación Web

La base de la aplicación web ya existía por lo que sólo hubo que agregarle la opción de registros de Ruido y la generación de reportes en base a estos registros. El anexo D muestra las diferentes pantallas agregadas, relacionadas a los reportes de ruido. La figura 3.5 muestra las interacciones de la aplicación web con las bases de datos y la aplicación móvil. A diferencia de la aplicación móvil, los audios hechos durante la etapa de registro se envían a un proceso externo después de ser grabados para ser procesados. Ese proceso se encarga de almacenar las series de tiempo en Airviro, devolviéndole a la base de datos la ruta para encontrarlas. El código asociado al procesamiento de audio se muestra en el anexo E.

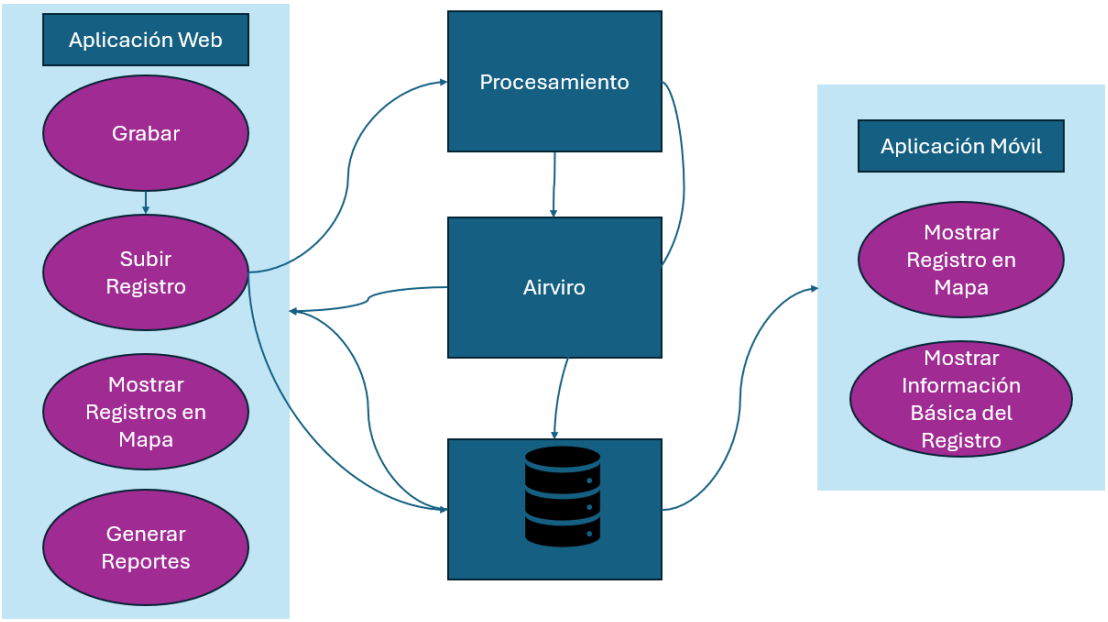


Figura 3.4: Interacción aplicación web – base de datos - móvil

Capítulo 4: Resultados

Para validar los algoritmos desarrollados, se realizaron pruebas para evaluar la precisión en las mediciones de nivel de ruido, de la clasificación, y de la geolocalización, todas estas tanto para la aplicación móvil cómo para la aplicación web. Finalmente se probaron las tres funcionalidades en conjunto en un contexto real.

4.1. Evaluación de las mediciones de nivel de ruido

4.1.1. Aplicación móvil

Para evaluar la precisión en las mediciones de nivel de ruido, se presenta los resultados de medir el ruido usando la aplicación instalada en 3 dispositivos móviles diferentes con las mediciones hechas por un sonómetro. El sonómetro usado es el SmartSensor AS844+, el cual cuenta con un rango de medición de 30-130 dBA con un rango de frecuencias de 31.5 Hz – 8 kHz con una precisión de ±1.5 dB. La aplicación VoiceLab se comunica con el sonómetro mediante USB para leer los datos y exportarlos a un archivo junto a sus estampas de tiempo correspondientes. La aplicación móvil, para efecto de comparación, los datos medidos en un archivo de texto junto a sus respectivas estampas de tiempo para así poder comparar de manera sencilla con lo obtenido por el sonómetro.

Como primer dispositivo móvil se presta un Samsung Galaxy A32 con el que, a diferencia de los otros dispositivos se hicieron 3 pruebas, ya que es al que se tuvo acceso más tiempo durante el desarrollo. Los resultados se muestran en las figuras 4.1, 4.2 y 4.3. El eje y corresponde a los valores de intensidad en decibeles, y el eje x corresponde al tiempo en segundos de cada respectiva grabación. El resumen de estas tres pruebas se muestra en la tabla 4.1.

El siguiente dispositivo con el que se hizo una prueba es un Samsung Z3 flip. El valor equivalente de toda la medición hecha por el sonómetro fue de 60 dB y por la aplicación con el ajuste del algoritmo 1 fue de 64.8 dB, dando un error de 4.8 dB (figura 4.4).

El tercer dispositivo fue un Samsung Galaxy A7. Los resultados de la prueba se muestran en la figura 4.5, en donde el valor equivalente de toda la medición hecha por el sonómetro fue de 68.2 dB y por la aplicación con el ajuste del algoritmo 1 fue de 63.7 dB, dejando un error de -4.7 dB.

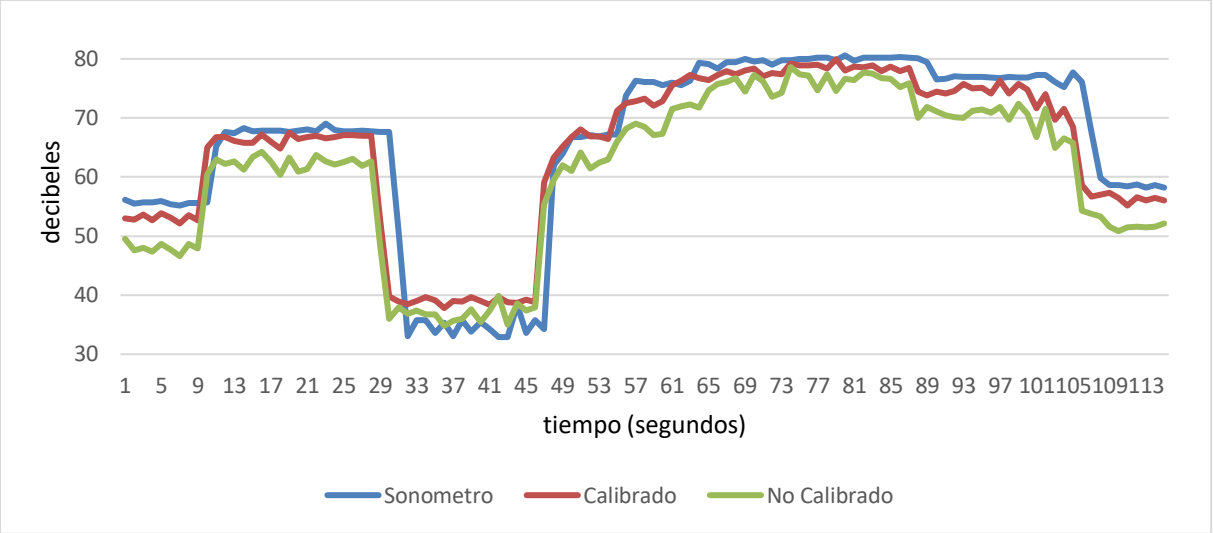


Figura 4.1: Samsung Galaxy A32, Prueba 1

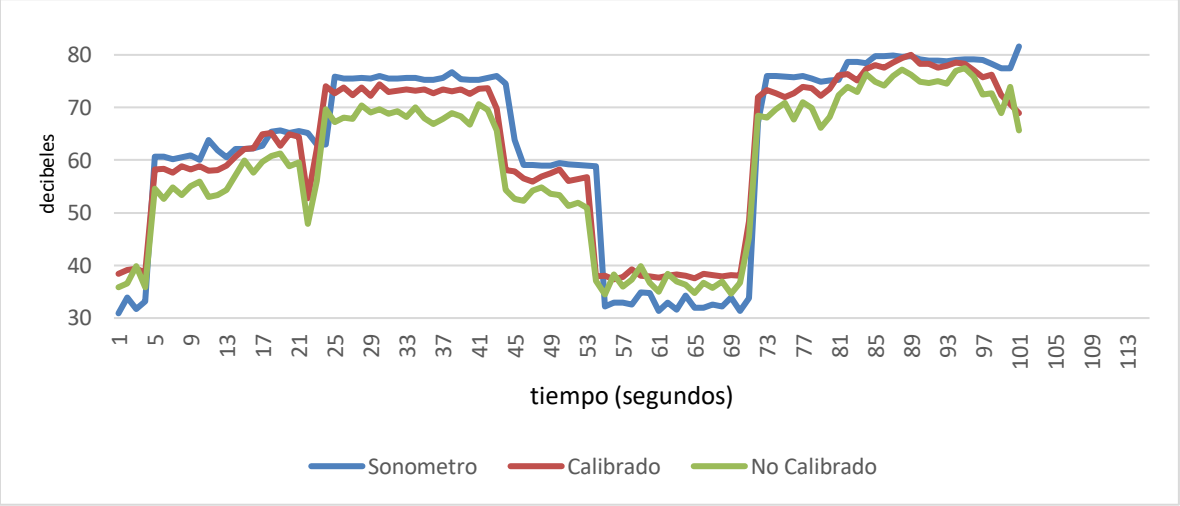


Figura 4.2: Samsung Galaxy A32, Prueba 2

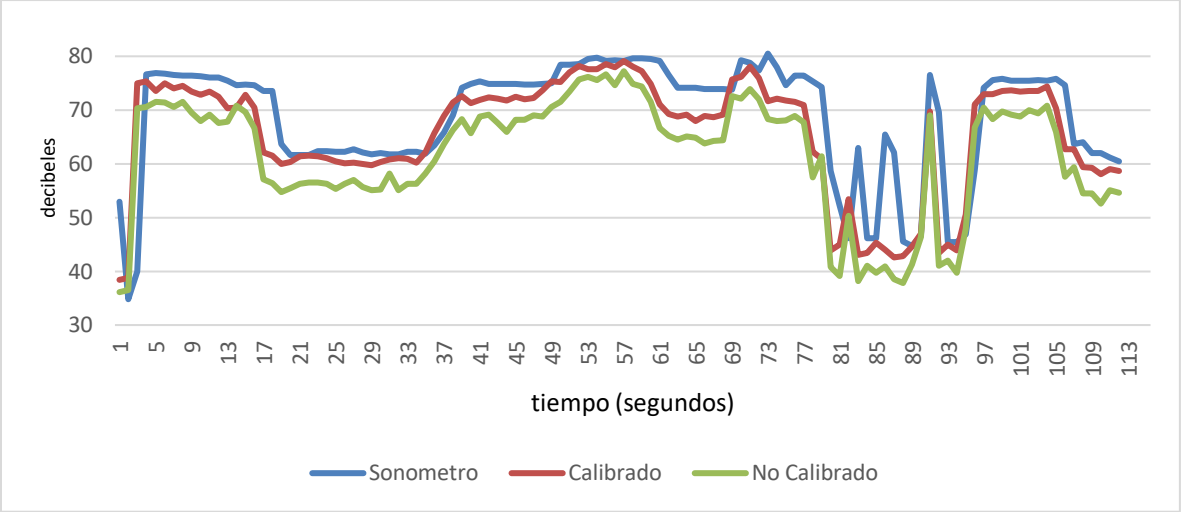


Figura 4.3: Samsung Galaxy A32, Prueba 3

Tabla 4.1: Promedio logarítmicos de las mediciones con Samsung Galaxy A32

N° Prueba	Sonómetro (dB)	Con Ajuste (dB)	Sin Ajuste (dB)	Error con ajuste (dB)	Error sin ajuste (dB)
1	65.8	64.5	60.9	1.3	4.9
2	63.1	62.3	59	0.8	4.1
3	68.7	65.6	61.8	3.1	6.9

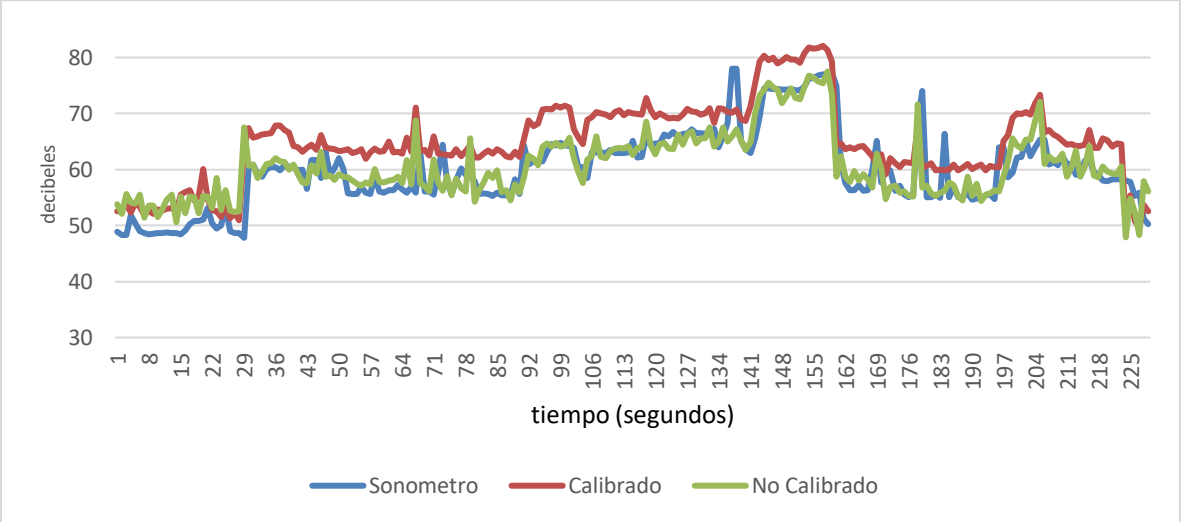


Figura 4.4: Prueba con Samsung Z3 Flip

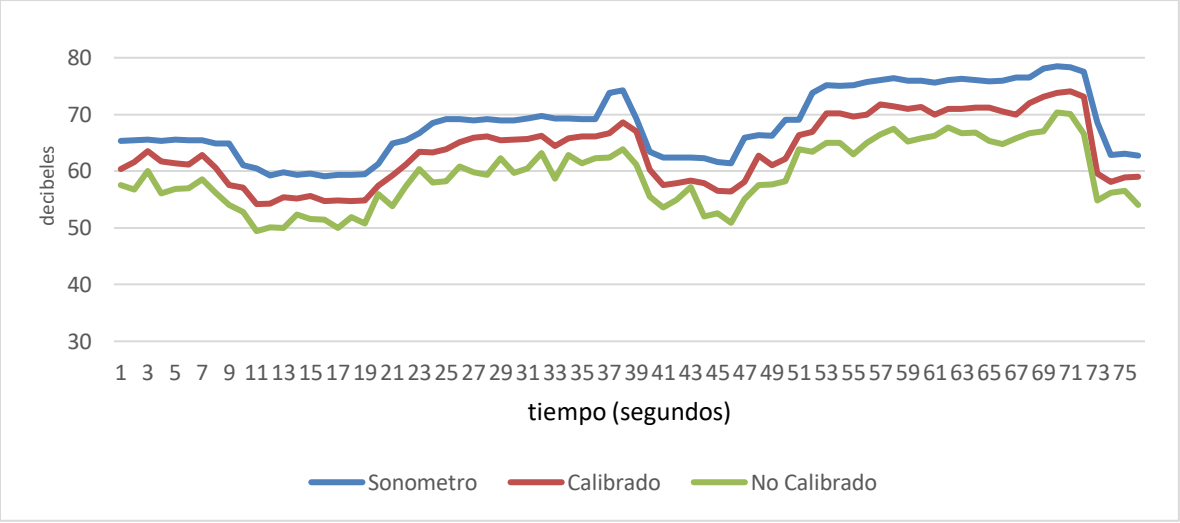


Figura 4.5: Prueba Samsung Galaxy A7

4.1.2. Aplicación web

A diferencia de un dispositivo móvil, ajustar el volumen de la entrada de audio en un computador es extremadamente fácil para el usuario (figura 4.6). Esta configuración no puede ser controlada desde la plataforma web, lo que introduce una variabilidad significativa en las mediciones de intensidad de ruido. Debido a esta falta de control sobre la configuración de entrada de audio en los computadores, las pruebas para determinar la precisión en la medición de intensidad resultan poco fiables, ya que los resultados dependerán completamente de cómo el usuario configure su dispositivo.

En una prueba, utilizando la configuración por defecto del sistema operativo y el navegador, se observó que las mediciones de intensidad de ruido eran significativamente menores en comparación con las realizadas por un sonómetro. Esta disparidad sugiere que la variabilidad en las configuraciones de volumen de entrada de audio puede influir drásticamente en los resultados obtenidos, haciendo que la plataforma web sea menos precisa para mediciones de intensidad de ruido en comparación con los dispositivos móviles.

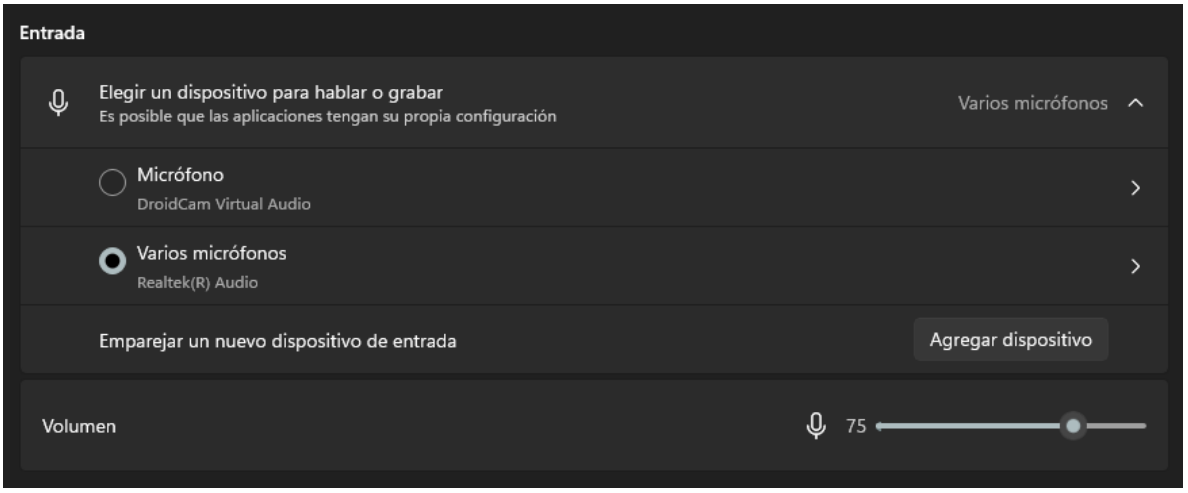


Figura 4.6: Configuración nivel de volumen del micrófono en un dispositivo con sistema operativo Windows 11

4.2. Evaluación del modelo de clasificación

La salida del modelo de clasificación es totalmente independiente de si la grabación se hace desde la aplicación móvil o la aplicación web pues la extracción de características se hace con respecto a la frecuencia. Por lo tanto, todas las pruebas para validar la eficiencia del modelo aplican para ambas plataformas.

La primera prueba consistió en evaluar la clasificación que el modelo hace sobre otros conjuntos de datos diferentes a con el que este fue entrenado. Los conjuntos elegidos para esto fueron UrbanSound8k [11] y ESC – 50 [12]. Cada audio en estos conjuntos fue procesado para eliminar las partes silenciosas. Además, para calcular la precisión de la clasificación sobre una clase presente en los conjuntos de datos elegidos, resultan validas todas aquellas clasificaciones que tengan alguna relación con la clase correcta. Por ejemplo, en el ESC – 50 tenemos la clase “perro”, posibles clasificaciones para esto pueden ser “Animal”, “perro”, “mascota”, entre otros, y todas con correctas.

La tabla 4.2 muestra los resultados obtenidos al usar YAMNet para clasificar el contenido del conjunto ESC – 50. El nombre de la clase muestra la etiqueta de los audios dentro del conjunto sobre el que se está evaluando, la precisión muestra que tan acertadas fueron las clasificaciones y clasificaciones consideradas muestra aquellas clasificaciones hechas por el modelo sobre el conjunto de datos que se consideran correctas. La tabla 4.3 muestra lo mismo, pero sobre el conjunto UrbanSound8K.

Tabla 4.2: Evaluación de YAMNet sobre el conjunto ESC – 50

Nombre de la clase	Precisión	Clasificaciones consideradas		Nombre de la clase	Precisión	Clasificaciones consideradas
Dog	97,5%	Animal, Dog, Domestic animals, pets		Footsteps	32,5%	Walk, footsteps
Rooster	92,5%	Animal		Laughing	42,5%	Laughter
Pig	88,5%	Animal, Pig, farm animals		Brushing teeth	32,5%	Toothbrush
Cow	65,0%	Animal, farm animals, bovinae, farm animals		Snoring	90,0%	Snoring, Breathing
Frog	87,5%	Animal, Frog		Glass Breacking	70,0%	Glass, Crack
Cat	87,5%	Animal, Cat, Domestic animals, pets		Door Knock	57,5%	Knock
Hen	100,0%	Animal, farm animals		Mouse Click	22,5%	Typing, clink
Insects	70,0%	Animal, Insect		Keyboard Typing	92,5%	Typing
Sheep	90,0%	Animal, Sheep, farm animals		Door wood creaks	2,5%	Knock
Crow	100,0%	Animal, Crow		Can opening	0,0%	
Rain	87,5%	Water, Rain		Washing machine	12,5%	Mechanisms, Blender
Sea waves	82,5%	Water, Boat, Ocean		Vacuum cleaner	25,0%	Vaccum claener, Mechanisms
Crackling fire	20,0%	Fire, Crack, Crickling		Clock alarm	87,5%	Alarm clock, Jingle bell, Telephone, Alarm, Telephone bell ringing, Ringtone, Beep, bleep
Crickets	23,0%	Insect		Clock tick	30,0%	Clack, Alarm clock
Chriping birds	92,5%	Anima, Bird, Wild animals		Glass Breaking	80,0%	Glass, Crack, Breacking
Water drops	20,0%	Drip, Plop		Helicopter	97,5%	Vehicle, Helicopter
Wind	7,5%	Wind		Chainsaw	62,5%	Chainsaw, Engine, Light engine
Pouring Water	95,0%	Water, Drip, Liquid		Siren	85,0%	Siren, Emergency vehicle, Alarm, Police car (siren)
Toilet Flush	90,0%	Toilet flush		Car Horn	82,5%	Vehicle, Car horn
Thunderstorm	90,0%	Thunderstorm		Engine	90,0%	Engine, Vehicle
Crying baby	85,0%	Crying		Train	97,5%	Vehicle, Rail transport
Sneezing	37,5%	Sneez		Church Bells	97,5%	Bell
Clapping	77,5%	Clapping, Applause, Hands, Cheering		Airplane	87,5%	Vehicle
Breathing	32,5%	Breathing		Fireworks	87,5%	Fireworks, Explsion
Coughing	77,5%	Cough		Hand Saw	80,0%	Sawing, Wood, Rub

Tabla 4.3: Evaluación YAMNet sobre el conjunto UrbanSound8K

Nombre de la clase	Precisión	Clasificaciones consideradas
air conditioner	70,0%	Vehicle
car horn	70,0%	car horn, vehicle
children playing	70,0%	Speech, Laughter
dog bark	79,0%	Animal, Dog, Domestics animals
drilling	29,4%	Tools, Engine, Noise
engine indeling	70,0%	Vehicle
gun shot	72,0%	Exploxion, Gunshot, Fireworks
jackhammer	47,6%	Jackhammer, Tools, Engine
siren	93,5%	Emergency vehicle, vehicle, Siren, Alarm, Ambulance (siren), Police (car)
street music	75,7%	Music

Por otra parte, también se hicieron pruebas en terreno para revisar el comportamiento del modelo de clasificación en un entorno real. Entonces, se usó la aplicación en dos lugares distintos y se extrajeron las salidas del modelo de clasificación. La primera prueba se realizó en el parque Ecuador durante la tarde de un día de semana. La prueba duró 30 minutos. La figura 4.7 muestra los resultados. La segunda prueba se realizó en las oficinas de R9 durante horario laboral y duró dos horas, los resultados se resumen en la figura 4.8.

Figura 4.7: Resultados del modelo de clasificación tras 30 minutos de grabación en el parque ecuador

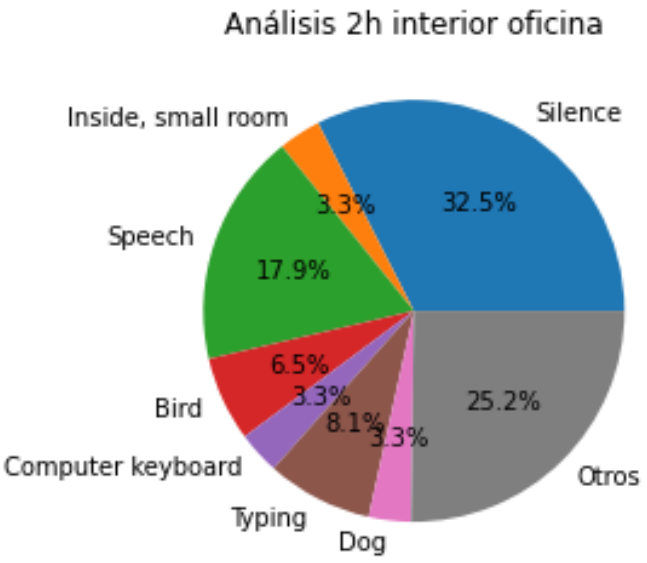


Figura 4.8: resultados del modelo de clasificación tras 2 horas de grabación en las oficinas de R9

4.3. Evaluación de la obtención de geolocalización

4.3.1. Aplicación móvil

Para acceder a la geolocalización por parte de la aplicación móvil se usan los servicios de ubicación de Google, los cuales según la documentación [29], tienen un margen de error de 50 metros. Durante el desarrollo de la aplicación se hicieron numerosas pruebas de funcionalidad, muchas de estas desde las oficinas de R9. la figura 4.7 muestra la ubicación en el mapa obtenida por la aplicación durante la realización de estas pruebas.

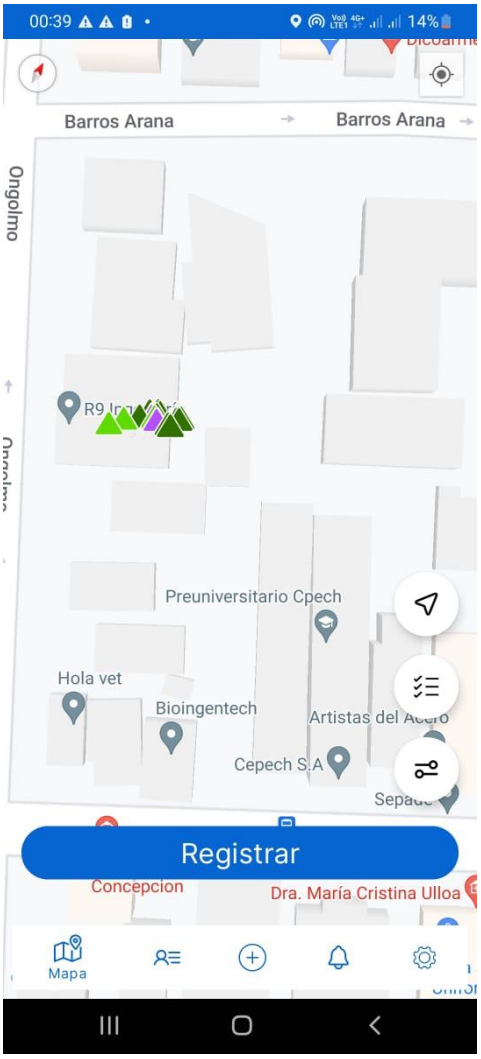


Figura 4.9: Evaluación de la geolocalización obtenida por la aplicación móvil

4.3.2. Aplicación web

La figura 4.8 muestra la precisión de Google Maps a la hora de obtener la geolocalización desde un navegador en un dispositivo que no cuenta con GPS versus la ubicación real, como se puede ver, el error es muy significativo. Por lo anterior la aplicación web sigue un enfoque diferente al de la aplicación móvil en donde se le pide al usuario ingresar manualmente la ubicación del evento (figura 4.9).

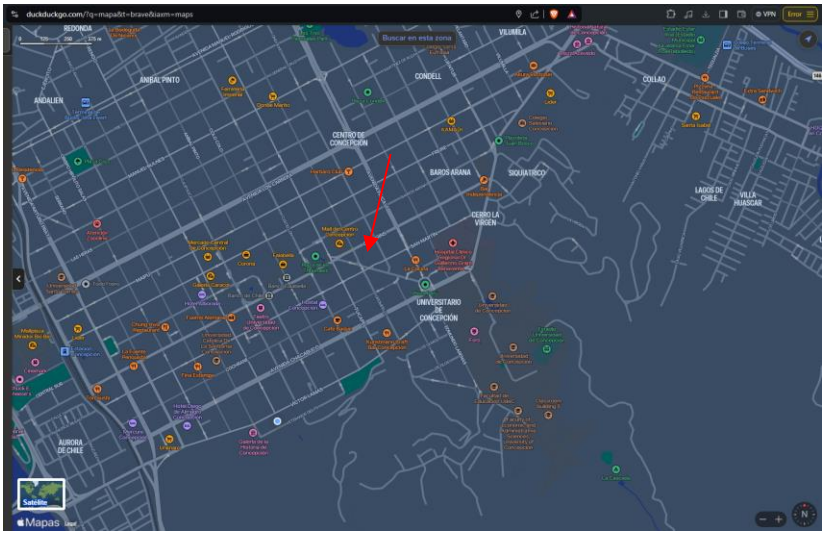


Figura 4.10: Geolocalización dada por Google Maps (punto azul) versus ubicación real (flecha roja) en un navegador

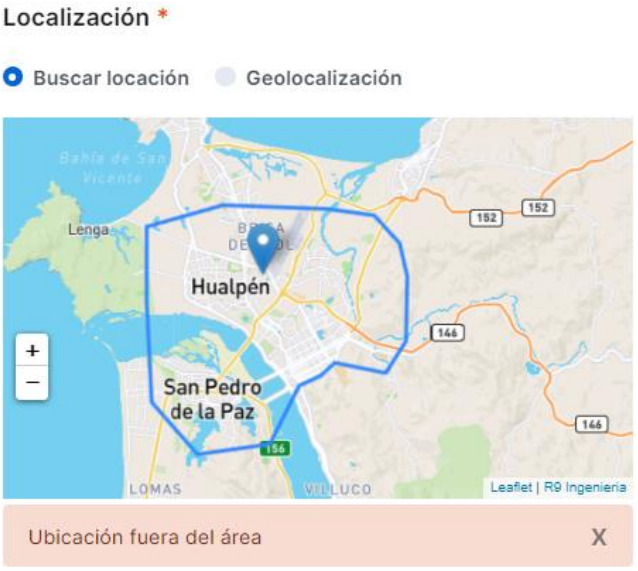


Figura 4.11: Ingreso manual de la localización en la plataforma de incidencias ambientales de R9

4.4. Evaluación en situación de uso real

Para la evaluación general de la plataforma de incidencias, se planificaron pruebas con el fin de simular situaciones de uso real. El primer tipo de prueba consistió en capturar eventos en un punto especifico con varios dispositivos móviles distintos y comparar el resultado obtenido por cada uno de ellos. El anexo F muestra al detalle la planificación de estas pruebas. Este tipo de prueba se realizó en tres ubicaciones de interés diferentes, fuera de la empresa “Cementos Biobío”, fuera de la

refinería ENAP Concepción y fuera de una construcción ubicada en avenida Chacabuco, concepción.

Previo a la realización de estas pruebas, con el fin de usar un dispositivo móvil como referencia, se instaló la aplicación en un Xiaomi Redmi R3 para comparar las mediciones que la aplicación hace en este dispositivo con las mediciones hechas por un sonómetro, el sonómetro Testo 816-1, de clase 2, calibrado según la norma IEC 61672-1 con un error de ±1.4 dB, la figura 4.10 muestra los resultados de esta medición. El error estimado fue de ±0.6 dB por lo que el Xiaomi Redmi R3 se usa como referencia para las pruebas mencionadas en esta sección.

Los resultados de la prueba realizada fuera de Cementos Biobío al detalle se pueden ver en el anexo G. En cuanto a la medición de intensidad, la media fue 66 dB y la desviación estándar fue de 2.8 dB. En cuanto a la clasificación, debido a la cercanía con una autopista las clasificaciones más repetidas fueron “vehículo” y “hablando”, las cuales no están relacionadas a la industria. La tercera clasificación más frecuente fue “ventilador mecánico”, clasificación que coincide con el sonido emitido por los procesos llevados a cabo dentro de la industria. En cuanto a la ubicación, todos los dispositivos fueron ubicados correctamente.

Los resultados de la prueba realizada fuera de la refinería ENAP se muestran en detalle en anexo H. En cuanto a la medición de intensidad, la media fue 72 dB y la desviación estándar fue de 1.5 dB. En cuanto a la clasificación, debido a la cercanía con una autopista las clasificaciones más repetidas fueron “vehículo” y “hablando”, las cuales no están relacionadas a la industria. La tercera clasificación más frecuente fue “viento”, clasificación que coincide con el sonido emitido por la antorcha presente la refinería. En cuanto a la ubicación, todos los dispositivos fueron ubicados correctamente.

Los resultados de la prueba realizada fuera de una construcción en avenida Chacabuco se muestran en detalle en el anexo I. En cuanto a la medición de intensidad, la media fue 75 dB y la desviación estándar fue de 3 dB. En cuanto a la clasificación, debido a la cercanía con una avenida principal, la clasificación más repetida fue “vehículo”, seguida de “hablando”. Cuando no había nada de ruido externo y solo quedaba el ruido de fondo emitido por la construcción la clasificación fue “transporte ferroviario”, lo que podría deberse a sonidos metálicos fuertes emitidos por la construcción. En cuanto a la ubicación, todos los dispositivos fueron ubicados correctamente.

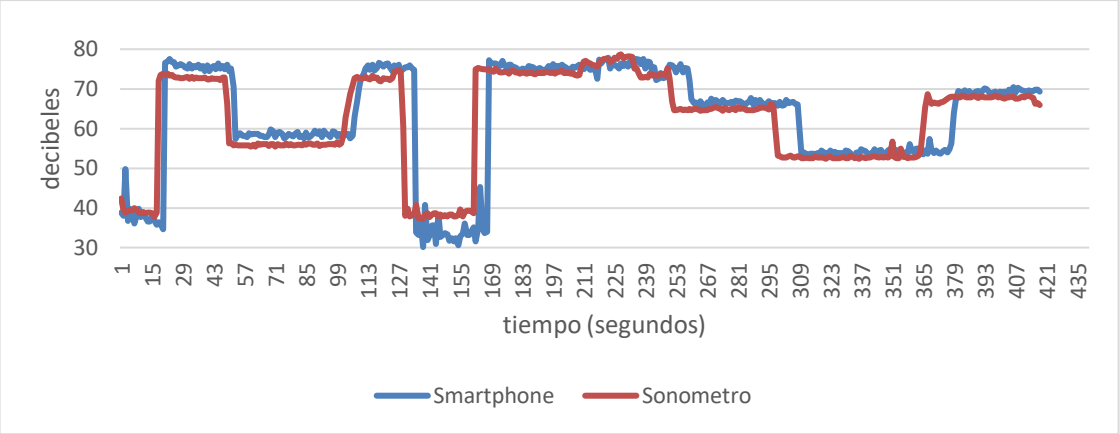


Figura 4.12: Medición en dB versus tiempo. Xiaomi Redmi R3 con sonómetro testo 816-1

4.5. Discusión de los resultados

La medición de la intensidad usando la aplicación móvil tiene un rango de error de ±5dB, lo que es suficiente para una medición de tipo informativa. Sin embargo, este error descarta el uso de teléfonos móviles en mediciones de carácter oficial o con el objetivo de aplicar la norma. Al aplicar la ponderación de tipo A en la medición, esta se adapta mejor a lo percibido por el oído humano.

El uso de modelos pre entrenados en la tarea de clasificación de audio acelera el proceso de desarrollo de la herramienta. Sin embargo, dado que el conjunto de datos con los que el modelo fue entrenado son datos de carácter general entonces se tiene como resultado que las clasificaciones no son tan precisas al poner el modelo a prueba en situaciones muy específicas. Por ejemplo, se observó que al medir la refinería ENAP, sonidos que son comunes tales como vehículos o personas hablando son clasificados si problema, pero un sonido que es característico de ese lugar tal como el sonido que emite la antorcha se clasifica como viento pues no existe una clase específica para este sonido que es poco común. No obstante, estas clasificaciones pueden ser suficientes para hacerse una idea de lo que está pasando si se ésta se analiza en conjunto con la ubicación y la intensidad, dándole un contexto.

La obtención de la geolocalización en dispositivos móviles es fácil de obtener y el rango de error es menor a 50 metros, siendo este error mucho menor en la mayoría de los casos. La obtención de la geolocalización en dispositivos que no cuentan con tecnología GPS es mucho menos precisa, con errores que pueden fácilmente superar los 100 metros. La solución de dejar que los usuarios ingresen la ubicación de los eventos de forma arbitraria puede significar la creación de eventos con

una ubicación errónea.

Finalmente, al poner la aplicación a prueba en un entorno real, la variación de la medición de intensidad es notablemente dependiente de la diferencia de componentes de cada dispositivo, sin embargo, el error no supera los ±5dB. La precisión de las clasificaciones varía dependiendo de lo común que sea el ruido que se clasifique, pero es suficiente para hacerse una idea de lo que ocurre durante el evento registrado, y la ubicación se obtiene con un rango de error pequeño, permitiendo así ubicar el evento con precisión.

Capítulo 5: Conclusiones

El ruido ambiental es un problema en aumento por lo que es necesario generar herramientas que faciliten su prevención y monitoreo y que faciliten la creación de medidas de mitigación.

En base a los resultados obtenidos, se puede concluir que la herramienta desarrollada, aunque es efectiva y útil, presenta limitaciones que impiden catalogarla como excelente. La aplicación móvil, si bien permite realizar mediciones de intensidad de ruido con un rango de error de ±5 dB, no es adecuada para mediciones oficiales. Sin embargo, esta precisión es suficiente para obtener una medida informativa que puede ser utilizada para propósitos generales de monitoreo.

Se identificaron varios puntos mejorables, como la precisión de la clasificación de audio en entornos específicos y la precisión de la geolocalización en dispositivos sin GPS. El uso de modelos pre entrenados ha demostrado ser eficaz para la clasificación de sonidos comunes, pero se requiere una mayor especialización para ruidos industriales específicos.

El uso combinado de todas las variables medidas - intensidad, clasificación de audio y geolocalización - permite obtener una mayor exactitud en la caracterización del ruido ambiental. La versión móvil de la herramienta resulta ser más precisa y conveniente para la recolección de datos en el campo debido a su capacidad de medición directa y la facilidad de obtener geolocalización con un margen de error reducido.

La versión web, por su parte, es útil para el análisis y la visualización de los datos recolectados, permitiendo una mayor flexibilidad en la manipulación de estos y de la creación de reportes.

La herramienta creada proporciona una solución accesible y efectiva para el monitoreo y la caracterización de ruido. Aunque existen áreas de mejora, la combinación de las capacidades de la aplicación móvil y la plataforma web ofrece una herramienta robusta que puede contribuir significativamente a la prevención y mitigación de los efectos negativos del ruido en la salud pública.

Finalmente, se logró crear una herramienta que permite medir la intensidad del ruido, clasificarlo y geolocalizarlo usando una aplicación móvil y desplegar el análisis de los datos y generar reportes en base a estos en una aplicación web.

Referencias

[1] World Health Organization. World report on hearing. Geneva. 2021.

[2] T. Münzel, S. Kröller-Schön, M. Oelze, T. Gori, F. P. Schmidt, S. Stevn, O. Hahad, M. Röösli, J. Wunderli, A. Daiber, M. Sørensen. Adverse Cardiovascular Effects of Traffic Noise with a Focus on Nighttime Noise and the New WHO Noise Guidelines. Annual Review of Public Health, vol 41, pp. 409-328, 2020. doi: 10.1146/annurev-publhealth-081519-062400

[3] O. Hahad, M. Kuntic, S. Al-Kindi, *et al*. Noise and mental health, mechanisms, and consequences. J Expo Sci Environ Epidemiol. 2024. doi: 10.1038/s41370-024-00642-5.

[4] World Health Organization. Environmental noise Guidelines for the European Region. 2018.

[5] W. Zamora, C. Calafate, J. Cano, P. Manzoni. Accurate Ambient Noise Assessment Using Smartphomes. Sensors, vol 17, pp. 917-935, 2017. doi: 10.3390/s17040917.

[6] A. Boumchich, J. Picaut, P. Aumond, A. Can, E. Bocher. Blind Calibration of Environmental Acoustic Measurements Using Smartphones. Sensors, vol 24, pp. 1255-1279, 2024. doi: 10.3390/s24041255.

[7] R. Doumoulin, J. Voix. Calibration of smartphone-based devices for noise exposure monitoring: methodology, uncertainties of measurement and implementation. Proceedings of Meetings on Acoustics, vol 19, pp. 54-64, 2013. doi: 10.1121/1.4800063.

[8] B. Roberts, C. Kardous, R. Neitzel. Improving the Accuracy of Smart Devices to Measure Noise Exposure. J Occup Environ Hyg, vol 13, pp. 840-846, 2016. doi: 10.1080/15459624.2016.1183014.

[9] IBM. "¿Qué es una ciudada inteligente?". 2023. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/smart-city. [Accedido: 15-May-24]

[10] A. Rodrigues, H. Olivera, J. Machado, J. Tavares. Sound Classification and Processing of Urban Environments: A Systematic Literature Review. Sensors, vol 22, pp. 8608-8638, 2022. doi: 10.3390/s22228608.

[11] J. Salamon, C. Jacoby, J. Bello. A dataset and taxonomy for urban sound research. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA. 3–7 November 2014. pp. 1041-1044. Doi: 10.1145/2647868.2655045.

[12] J. Piczak. ESC: Dataset for environmental sound classification. In Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane, Australia, 26–30 October 2015, pp. 1015–1018. Disponible en: https://www.karolpiczak.com/papers/Piczak2015-ESC-

Dataset.pdf. [Accedido: 20-Jul-24]

[13] Y. Koisumi, Y. Kawagushi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, *et al*. Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring. Detection and Classification of Acoustic Scenes and events, Tokyo, Japan, 2-3 November 2020, pp. 81-85. Disponible en: https://arxiv.org/pdf/2006.05822. [Accedido: 20-Jul-24]

[14] H. Cao, D. Cooper, M. Keutmann, R. Gur, A. Nenkova, R. Verma. Crema-d: Crowd-sourced emotional multimodal actors' dataset. IEEE Trans Affect Comput, vol 5, pp. 388-390, 2014. doi: 10.1109/TAFFC.2014.2336244.

[15] J. Gemmeke, D. Ellis, D. Freedman, A. Jansen, W. Lawrance, R. Moore, M. Plakal, M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). New Orleans, LA, USA, 5–9 March 2017, pp. 776–780. doi: 10.1109/ICASSP.2017.7952261

[16] A. Mesaro, T. Heittola, T. Virtanen. TUT Database for Acoustic Scene Classification and Sound Event Detection. In Proceedings of the 24th European Signal Processing Conference 2016 (EUSIPCO 2016), Budapest, Hungary, 29 August–2 September 2016, pp. 1128-1132. doi: 10.1109/EUSIPCO.2016.7760424.

[17] F. Rachman, R. Sarno, C. Fatichah. Music Emotion Classification based on Lyrics-Audio using Corpus based Emotion. International Journal of Electrical and Computer Engineering, vol 8, pp. 1720-1730, 2018. Doi: 10.11591/ijece.v8i3.pp1720-1730.

[18] E. Fonseca, X. Favory, J. Pons, F. Font, X. Serra. FSD50K: An open dataset of human-labeled sound events. IEEE/ACM Transactions on Audio, Speech and Language Processing, vol 30, pp. 829-852, 2022. Doi: 10.48550/arXiv.2010.00475.

[19] FreeSound. "freesound.org Find any sound you like". 2024. [En línea]. Disponible en: https://freesound.org/. [Accedido: 16-May-24]

[20] O. Abayomi-Alli, R. Damaševičius, A. Qazi, M. Adedoyin-Olowe, S. Misra. Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review. Electronics, vol 11, pp. 3795-3827, 2022. doi: 10.3390/electronics11223795.

[21] H. Jeon, Y. Jung, S. Lee, Y. Jung. Area-Efficient Short-Time Fourier Transform Processor for Time-Frequency Analysis of Non-Stationary Signals. Applied Sciences, vol 10, pp. 7208-7218 2020. doi: 10.3390/app10207208.

[22] K. Doshi. Audio Deep Learning Made Simple (Part 3): Data preparation and Augmentation.

Medium. 2021. [En línea]. Disponible en: https://towardsdatascience.com/audio-deep-learning-made-simple-part-3-data-preparation-and-augmentation-24c6e1f6b52. [Accedido: 22-May-24]

[23] K. Zaman, M. Sah, C. Direkoglu, M. Unoki. A Survey of Audio Classification Using Deep Learning. IEEE Access, vol 11, pp. 106620-106649, 2023. doi: 10.1109/ACCESS.2023.3318015.

[24] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, L. Farhan. Review of deep learning: concepts, CNN, architectures, challenges, applications, future directions. Journal of Big Data, vol 8, pp. 53-127, 2021. doi: 10.1186/s40537-021-00444-8.

[25] S. Harshey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, K. Wilson. CNN Architectures for large-scale audio classification. International Conference on Acoustics, Speech and Signal Processing (ICASPP), New Orleans, LA, USA, 2017, pp. 131-135. doi: 10.1109/ICASSP.2017.7952132.

[26] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang. Recent Advances in Convolutional Neural Networks. Pattern Recognit, vol 77, pp. 354-377, 2015. doi: 0.1016/J.PATCOG.2017.10.013.

[27] MathWorks. "yamnet". Disponible en: https://es.mathworks.com/help/deeplearning/ref/yamnet.html. [Accedido: 06-junio-2024].

[28] Kaggle. "yamnet. An audio evento classifier trained on the AudioSet dataset to predict audio events from AudioSet". Dsiponible en: https://www.kaggle.com/models/google/yamnet/frameworks/tensorFlow2/variations/yamnet/versions/1?tfhub-redirect=true . [Accedido: 06-junio-2024].

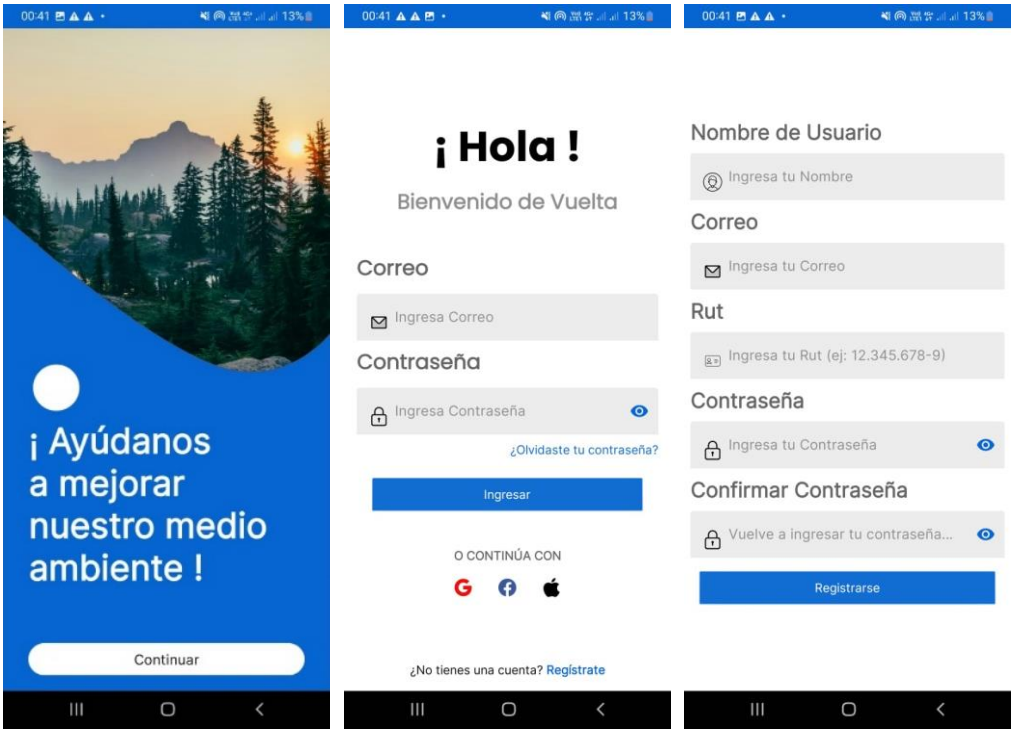
[29] Android Developers. "Solicita permisos de ubicación". Disponible en: https://developer.android.com/develop/sensors-and-location/location/permissions?hl=es-419. [Accedido: 06-junio-2024].

[30] R9 Ingeniería. "Soluciones y Software Para Gestión Ambiental". Disponible en: https://r9.cl/. [Accedido: 06-junio-2024].

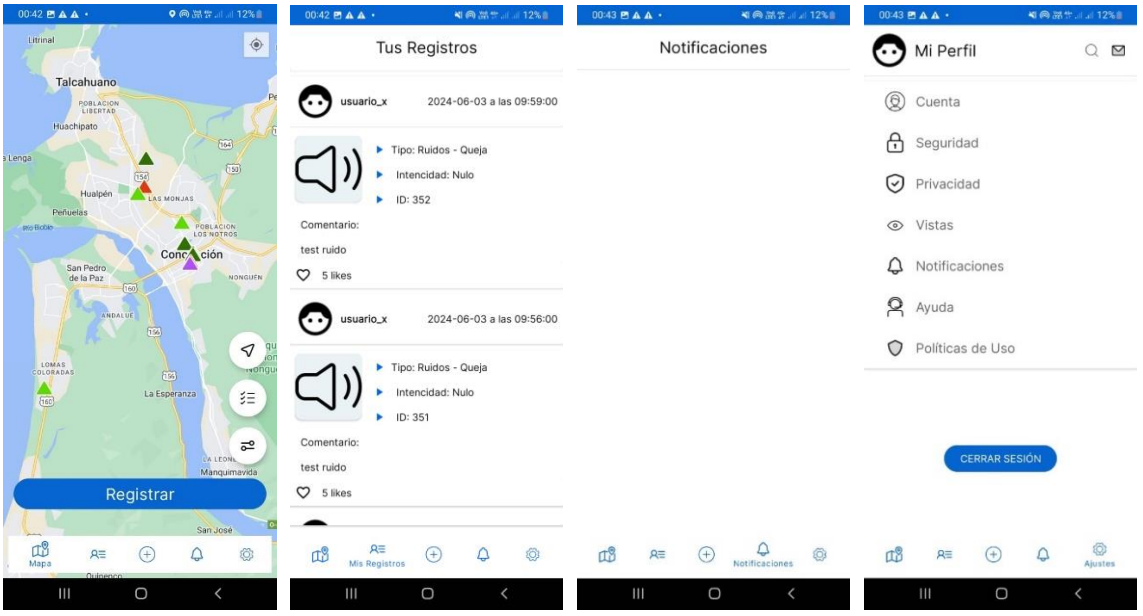
[31] TensorFlow. "TensorFlow Lite. Implementa modelos de aprendizaje automático en dispositivos móviles y perimetrales". Disponible en: https://www.tensorflow.org/lite?hl=es-419. [Accedido: 07-junio-2024].

Anexo A: Aplicación Móvil

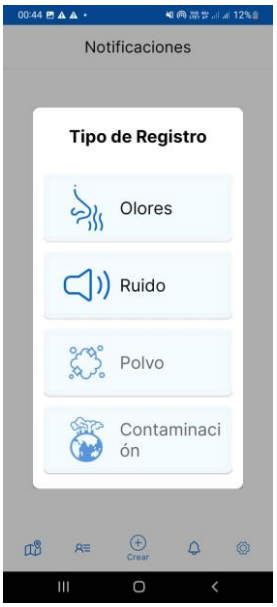
- Pantalla de lanzamiento, registro e inicio de sesión



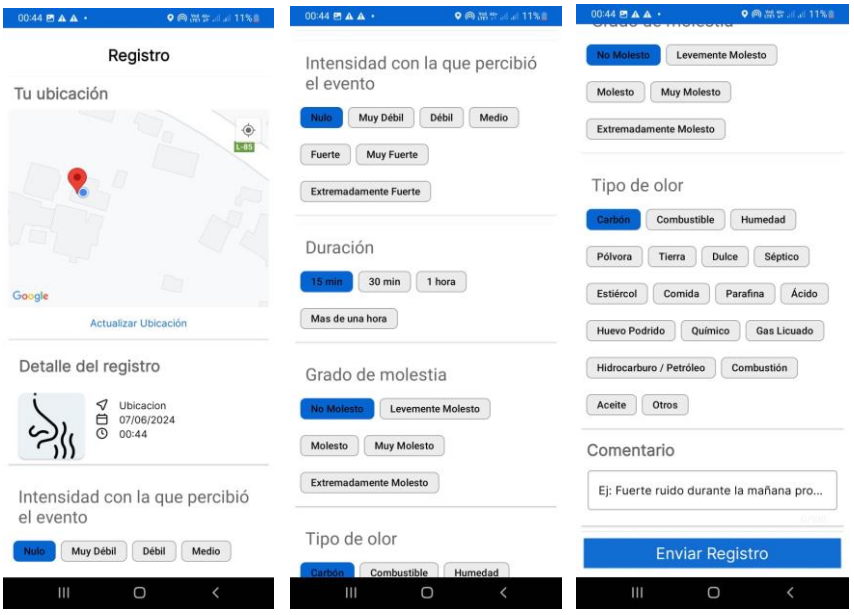
- Pantallas de mapa, registros del usuario, notificaciones y configuraciones



- Pantalla de opciones de registro

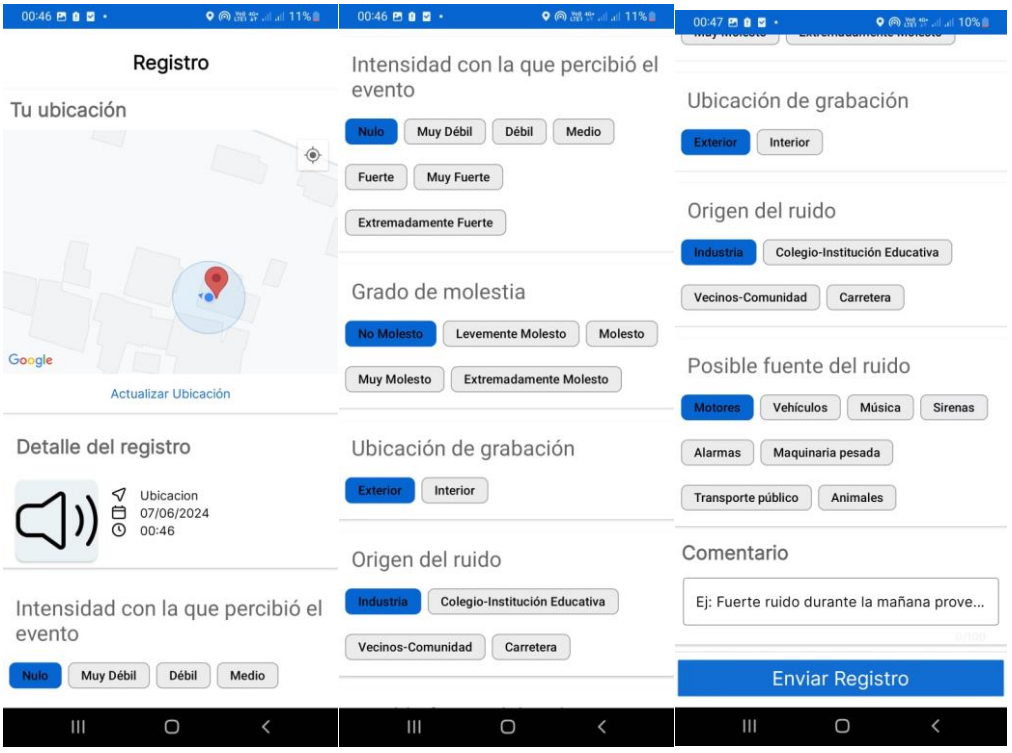


- Pantalla de Registro de Olores



- Pantalla de grabación.

- Pantalla de registro de ruidos



Anexo B: Cálculo de la intensidad en la aplicación móvil

```
    private suspend fun writeAudioDataToStorage() {
        val bufferSize = AudioRecord.getMinBufferSize(
            waveConfig.sampleRate,
            waveConfig.channels,
            waveConfig.audioEncoding
        )
        val data = ByteArray(bufferSize)
        val file = File(filePath)
        val outputStream = file.outputStream()
        while (isRecording) {
            val operationStatus = audioRecorder.read(data, 0, bufferSize)

            if (AudioRecord.ERROR_INVALID_OPERATION != operationStatus) {
                if (!isPaused) outputStream.write(data)

                withContext(Dispatchers.Main) {
                    onAmplitudeListener?.let {
                        it(calculateAmplitudeMax(data))
                    }
                    onTimeElapsed?.let {
                        val audioLengthInSeconds: Long = file.length() / timeModulus
                        it(audioLengthInSeconds)
                    }
                }
            }
        }

        outputStream.close()
        noiseSuppressor?.release()
    }
```

```
    private fun calculateAmplitudeMax(data: ByteArray): Double {
        val shortData = ShortArray(data.size / 2)
        ByteBuffer.wrap(data).order(ByteOrder.LITTLE_ENDIAN).asShortBuffer()
            .get(shortData)
        val noise = Noise.real(data.size/2)
        val src = FloatArray(data.size/2)
        val dst = FloatArray(data.size/2 + 2)

        for(i in 0 until data.size/2){
            val w = (1.0/2.0)*(1 - Math.cos((2 * Math.PI * i) / (data.size / 2 - 1)))
            src[i] = (shortData[i]*w).toFloat()
        }
        val fft = noise.fft(src, dst)
        val mag =  DoubleArray(data.size/2+2)
        for(i in 1 until data.size/4 -1){
            mag[i] = kotlin.math.sqrt(fft[i*2]*fft[i*2] + fft[i*2+1]*fft[i*2+1]).toDouble()
            var delta = calculateAWeightingDeltas(waveConfig.sampleRate, mag.size, i)
            delta = Math.sqrt(Math.pow(10.0, delta / 10))
            mag[i] *= delta
        }

        return 20.0* kotlin.math.log10(mag.average())
    }
```

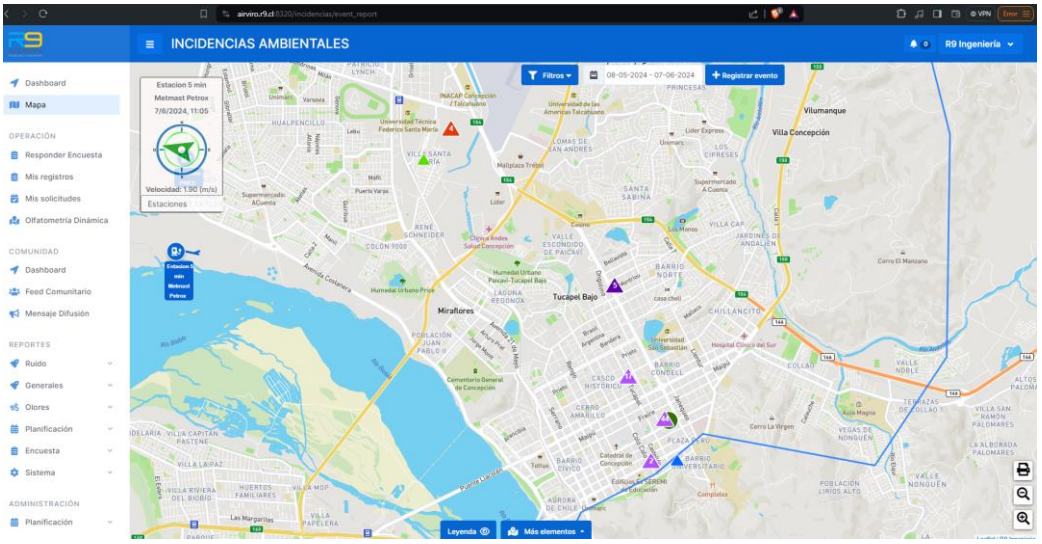
```
    fun calculateAWeightingDeltas(sampleRate: Int, fftSize: Int, i: Int): Double {
        val frequencyResolution = sampleRate.toDouble() / fftSize.toDouble()
        val f1 = 20.60
        val f2 = 107.7
        val f3 = 737.9
        val f4 = 12194.0
        val A1000 = -2
        if(i != 0){
            val frequency = i * frequencyResolution

            val num = Math.pow(f4, 2.0) * Math.pow(frequency, 4.0)
            val den = (Math.pow(frequency, 2.0) + Math.pow(f1, 2.0)) *
                    Math.sqrt((Math.pow(frequency, 2.0) + Math.pow(f2, 2.0))) *
                    Math.sqrt((Math.pow(frequency, 2.0) + Math.pow(f3, 2.0))) *
                    ((Math.pow(frequency, 2.0) + Math.pow(f4, 2.0)))

            val A = 20 * Math.log10(num / den) - A1000
            return A
        }else{
            val frequency = frequencyResolution

            val num = Math.pow(f4, 2.0) * Math.pow(frequency, 4.0)
            val den = (Math.pow(frequency, 2.0) + Math.pow(f1, 2.0)) *
                    Math.sqrt((Math.pow(frequency, 2.0) + Math.pow(f2, 2.0))) *
                    Math.sqrt((Math.pow(frequency, 2.0) + Math.pow(f3, 2.0))) *
                    ((Math.pow(frequency, 2.0) + Math.pow(f4, 2.0)))
            val A = 20 * Math.log10(num / den) - A1000
            return A
        }
    }
```

Anexo C: Implementación del modelo de clasificación en la aplicación móvil

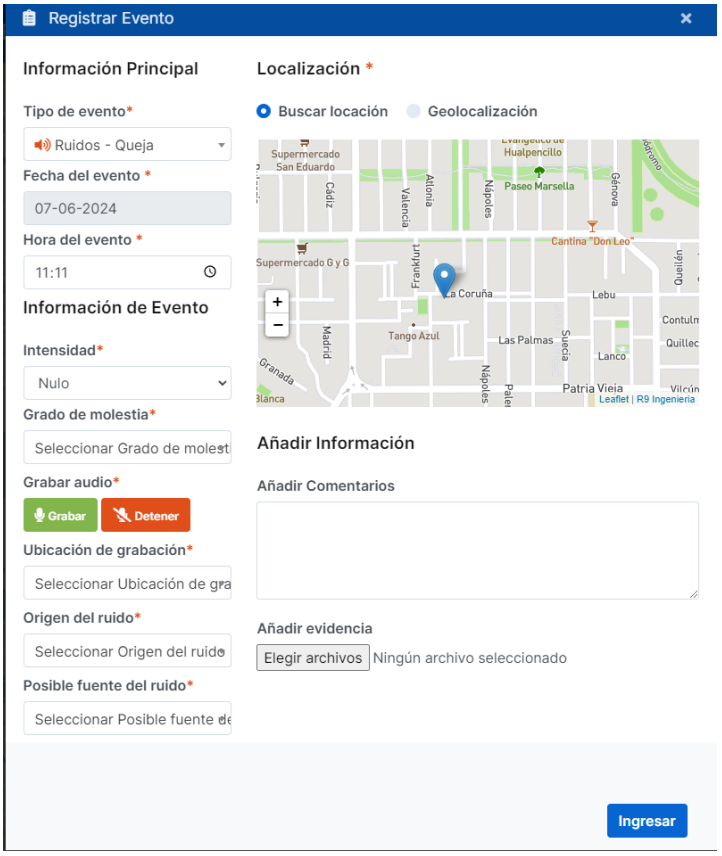
```
1   private fun startClassificationModel() {
2       val classifier = AudioClassifier.createFromFile(this, modelPath)
3       val tensor = classifier.createInputTensorAudio()
4       val format = classifier.requiredTensorAudioFormat
5       audioRecord = classifier.createAudioRecord()
6       audioRecord.startRecording()
7       classifierTimerTask = Timer().scheduleAtFixedRate(delay = 1, period = 1000) {
8           tensor.load(audioRecord)
9           val output = classifier.classify(tensor)
10          val filteredModelOutput = output[0].categories.filter { it.score > probabilityThreshold }
11          val outputStr = filteredModelOutput.sortedBy { -it.score }
12              .joinToString(separator = "\n") { "${AppConstants.label_codes_spanish[it.index]} -> ${it.score} " }
13
14
15          classificationDataString.postValue(AppConstants.label_codes_spanish[filteredModelOutput.first().index])
16          classificationDataIndex.postValue(filteredModelOutput.first().index)
17      }
18  }
```

Anexo D: Aplicación Web

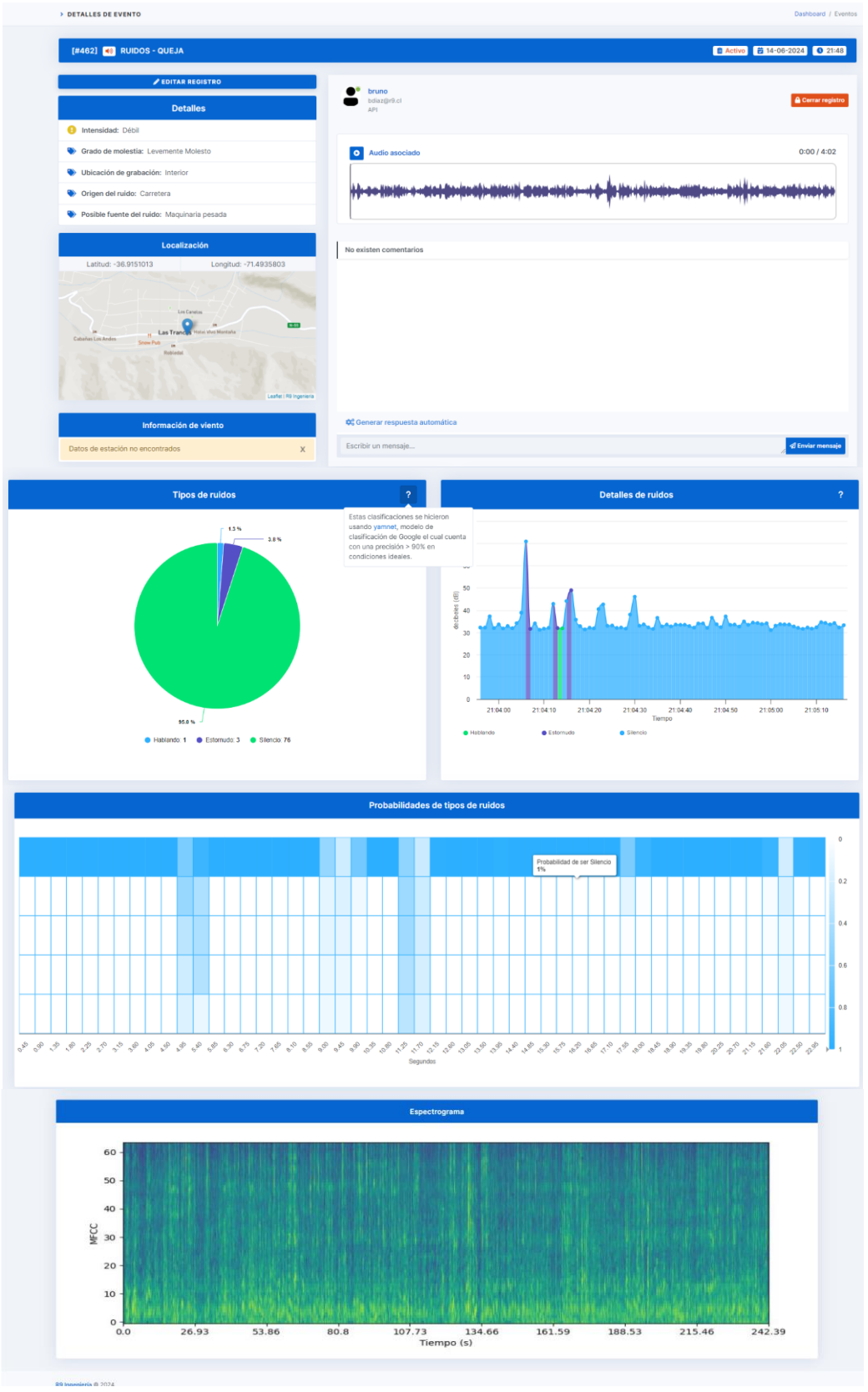
- Pantalla del Mapa



- Pantalla de Registros (Tipo de registro seleccionable)

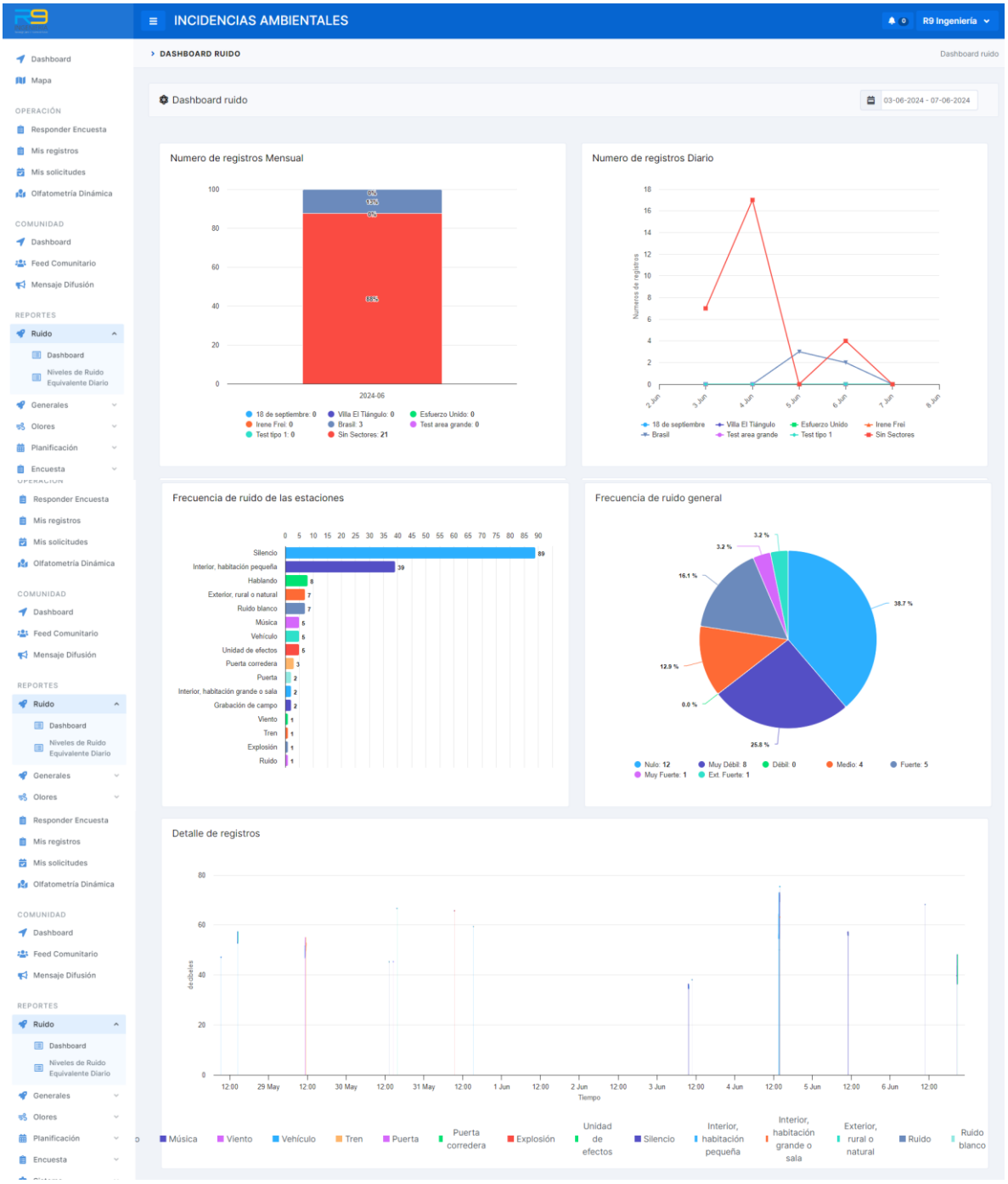


- Pantalla de detalles de evento

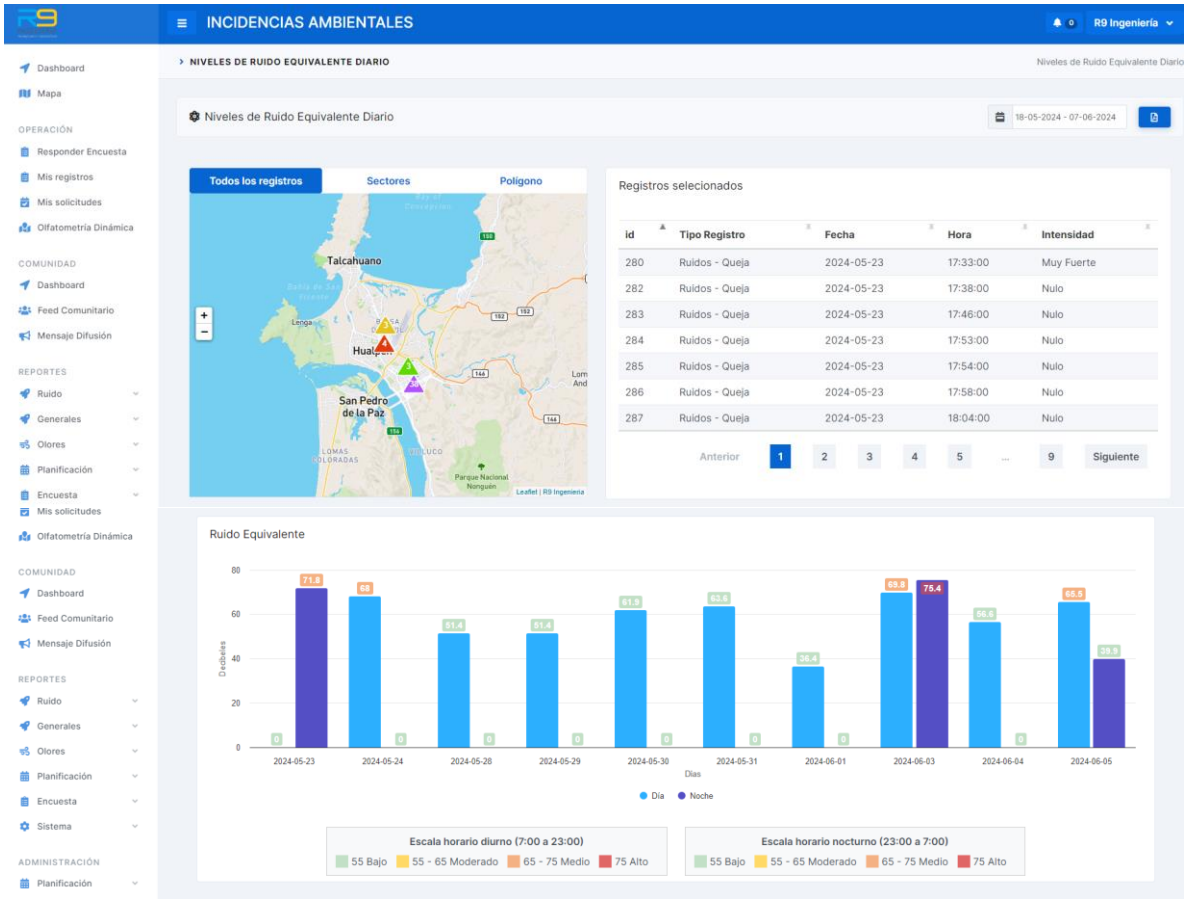


D.2

- Dashboard de registros de Ruido



- Pantalla de Reporte de Ruido equivalente Diario



Anexo E: Script para procesar audios

```
112  def execute_model(audio_file_paath):
113      wav_file_name = audio_file_paath
114      sample_rate, wav_data = wavfile.read(wav_file_name, 'rb')
115      sample_rate, wav_data = ensure_sample_rate(sample_rate, wav_data)
116      duration = len(wav_data)/sample_rate
117      Audio(wav_data, rate=sample_rate)
118      waveform = wav_data / tf.int16.max
119      output = model(waveform)
120      scores = output['output_0'].numpy()
121      embeddings = output['output_1'].numpy()
122      spectrogram = output['output_2'].numpy()
123      return scores, embeddings, spectrogram, waveform
124
125  def ensure_sample_rate(original_sample_rate, waveform,
126                         desired_sample_rate=16000):
127    """Resample waveform if required."""
128    if original_sample_rate != desired_sample_rate:
129      desired_length = int(round(float(len(waveform)) /
130                                 original_sample_rate * desired_sample_rate))
131      waveform = resample(waveform, desired_length)
132    return desired_sample_rate, waveform
133
```

```
134  def calculate_a_weighting_deltas(sample_rate, fft_size, i):
135      f1 = 20.60
136      f2 = 107.7
137      f3 = 737.9
138      f4 = 12194.0
139      A1000 = -2
140
141      frequency_resolution = sample_rate / fft_size
142      frequency = i * frequency_resolution if i != 0 else frequency_resolution
143
144      num = f4 ** 2 * frequency ** 4
145      den = (frequency ** 2 + f1 ** 2) * math.sqrt(frequency ** 2 + f2 ** 2) * math.sqrt(frequency ** 2 + f3 ** 2) * (frequency ** 2 + f4 ** 2)
146
147      A = 20 * math.log10(num / den) - A1000
148      return A
```

```python
def get_intensity(file_path):
    dB_arr = []
    audio = AudioSegment.from_wav(file_path)
    sample_rate = audio.frame_rate
    samples = np.array(audio.get_array_of_samples())

    segment_duration_ms = 1000
    segment_size = int(sample_rate * (segment_duration_ms / 1000.0))  # Número de muestras en 1 segundo


    for i in range(0, len(samples), segment_size):
        segment = samples[i:i + segment_size]

        # Aplicar la ventana Hanning
        window = hann(len(segment))
        windowed_segment = segment * window

        # Realizar la FFT
        fft_result = rfft(windowed_segment)

        # Calcular la magnitud del espectro
        magnitudes = np.abs(fft_result)

        # Aplicar la ponderación A
        for j in range(1, len(magnitudes) // 2):
            delta = calculate_a_weighting_deltas(sample_rate, len(magnitudes), j)
            delta = math.sqrt(10 ** (delta / 10))
            magnitudes[j] *= delta

        # Calcular el nivel equivalente de ruido (Leq) en decibelios
        Leq = 20 * np.log10(np.mean(magnitudes))
        dB_arr.append(Leq)

    return dB_arr

```

```python
def create_timeserie(begin, intensities, classes, user_id):
    time = int(begin)
    time_arr = []
    timeserie = []



    interpolated_classes = interpolate_classes(classes)

    size_of = min(len(intensities), len(interpolated_classes))



    for i in range(size_of):
        timepoint = {
                "timestamp": time,
                "value": intensities[i],
                "statusCode": 2*(500 + interpolated_classes[i]) # 2*(500+caract...) para acegurar pares > 1000 en airviro
            }
        time_arr.append(time)
        time += 1
        timeserie.append(timepoint)


    data = {

      "data": {
        "timeserie":timeserie
     }
    }


    end = time_arr[-1]
    start = time_arr[0]

    user_code = hex(int(user_id)).split("x")[1]
    instance = ""

    if len(user_code) < 3:
        for i in range(0, 3-len(user_code)):
            instance += "0"
        instance += user_code

    station = "PR1SMLEQ5" + instance

    url = base_url + station + f"/{start}/{end}"+"?force=true"

    print(url)
    response = requests.post(url, headers=headers, data=json.dumps(data))
    print(response.text)

    if response.status_code == 200 or response.status_code == 201:  # La respuesta fue exitosa
        #data = response.json()  # Convertir la respuesta a formato JSON
        return True, int(begin), end, station
    else:
        return False, 0, 0, ""

```

```python
@app.route('/upload_audio', methods=['POST'])
@error_handling
def upload_audio():
    if 'audio_file' not in request.files:
        return jsonify({'error': 'No se proporcionó ningún archivo de audio'}), 400
    audio = request.files['audio_file']
    if audio.filename == '':
        return jsonify({'error': 'No se subió ningún archivo de audio'}), 400
    if not allowed_file(audio.filename):
        return jsonify({'error': 'Formato de archivo de audio no válido'}), 400

    filename = secure_filename(audio.filename)
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    audio.save(filepath)

    #Manejar archivo comrpimido
    if(filename.endswith('.gzip') or filename.endswith('.zip') or filename.endswith('.gz')):
        with ZipFile(filepath) as zip_ref:
            zip_ref.extractall(app.config['UPLOAD_FOLDER'])
        os.remove(filepath)

        extracted_files = zip_ref.namelist()
        wav_files = [f for f in extracted_files if allowed_file(f)]
        if not wav_files:
            return jsonify({'error': 'El archivo no contiene archivos de audio válidos'}), 400
        first_wav_file = os.path.join(app.config['UPLOAD_FOLDER'], wav_files[0])
        return jsonify({'message': 'Archivo de audio subido exitosamente', 'file': first_wav_file}), 200
    #----------------

    return jsonify({'message': 'Archivo de audio subido exitosamente'}), 200


@app.route('/process_audio/<file_name>/<begin_timestamp>/<user_id>', methods=['GET'])
@error_handling
def process_audio(file_name, begin_timestamp, user_id):
    file_path = "uploads/" + file_name

    if os.path.exists(file_path):
       pass
    else:
        return "El archivo no existe en el directorio."

    begin = begin_timestamp
    print(begin)
    scores, embeddings, spectrogram, waveform = execute_model("uploads/"+file_name)
    classes = []
    #classes = [int(score.argmax()) for score in scores]

    for i in range(0, int(len(scores)-1) ):

        classes.append(int(scores[i].argmax()))

```

```
        max_score = [score[score.argmax()] for score in scores]
        intensities = get_intensity("uploads/"+file_name)



        airviro_response = create_timeserie(begin, intensities, classes, user_id)


        n_samples, n_classes = scores.shape

        top_n = 5
        reduced_scores = np.zeros((n_samples, top_n))
        reduced_classes = np.zeros((n_samples, top_n), dtype=int)

        for i in range(n_samples):
            top_indices = np.argsort(scores[i])[-top_n:][::-1]
            reduced_scores[i] = scores[i, top_indices]
            reduced_classes[i] = top_indices

        reduced_scores_list = reduced_scores.tolist()
        reduced_classes_list = reduced_classes.tolist()
        pair_index_class = []

        for i in range(0, len(reduced_scores_list)):
            pair_index_class.append([])
            for j in range(0, len(reduced_scores_list[i])):
                pair_index_class[i].append((reduced_classes_list[i][j], reduced_scores_list[i][j]))


        name =file_name.split(".")[0]
        iamge_path = f"images/{name}.jpg"

        graph_spectrogram(spectrogram, iamge_path)
```

```
359         sum = 0
360
361         for intensity in intensities:
362             sum += math.pow(10, intensity/10)
363         sum /= len(intensities)
364
365         Leq = 10*math.log10(sum)
366
367
368         output = {
369
370             "average_amplitude": Leq,
371             "max_amplitude": max(intensities),
372             "duration": len(intensities),
373             "airviro_station" : airviro_response[3],
374             "airviro_start": airviro_response[1],
375             "airviro_end": airviro_response[2],
376             "model_data": {
377                     "scores": pair_index_class,
378                     "spectrogram": name
379                 }
380         }
381
382         if airviro_response[0]:
383
384             return output
385
386         else:
387             return jsonify({'error': 'No se pudo subir la serie de tiempo a Airviro'}, airviro_response),400
```

```python
@app.route('/get_audio_classification/<file_name>', methods=['GET'])
@error_handling
def get_audio_classification(file_name):
    file_path = "uploads/" + file_name

    if os.path.exists(file_path):
       pass
    else:
        return "El archivo no existe en el directorio."

    scores, embeddings, spectrogram, waveform = execute_model("uploads/"+file_name)



    n_samples, n_classes = scores.shape

    top_n = 5
    reduced_scores = np.zeros((n_samples, top_n))
    reduced_classes = np.zeros((n_samples, top_n), dtype=int)

    for i in range(n_samples):
        top_indices = np.argsort(scores[i])[-top_n:][::-1]
        reduced_scores[i] = scores[i, top_indices]
        reduced_classes[i] = top_indices

    reduced_scores_list = reduced_scores.tolist()
    reduced_classes_list = reduced_classes.tolist()
    pair_index_class = []

    for i in range(0, len(reduced_scores_list)):
        pair_index_class.append([])
        for j in range(0, len(reduced_scores_list[i])):
            pair_index_class[i].append((reduced_classes_list[i][j], reduced_scores_list[i][j]))



    name =file_name.split(".")[0]
    iamge_path = f"images/{name}.jpg"

    graph_spectrogram(spectrogram, iamge_path)

    output = {
        "model_data": {
                "scores": pair_index_class,
                "spectrogram": name
            }
    }


    return output
```

```python
@app.route('/download_spectrogram/<file_name>', methods=['GET'])
@error_handling
def download_spectrogram(file_name):
    name = file_name.split(".")[0]
    image_path = f"images/{name}.jpg"

    if os.path.exists(image_path):
        return send_file(image_path, mimetype='image/jpeg')
    else:
        return "El archivo no existe en el directorio.", 404




if __name__ == "__main__":
    app.run()#(host='0.0.0.0', port=3000)
```

Anexo F: Plan de pruebas en un escenario real

Planificación mediciones de Ruido

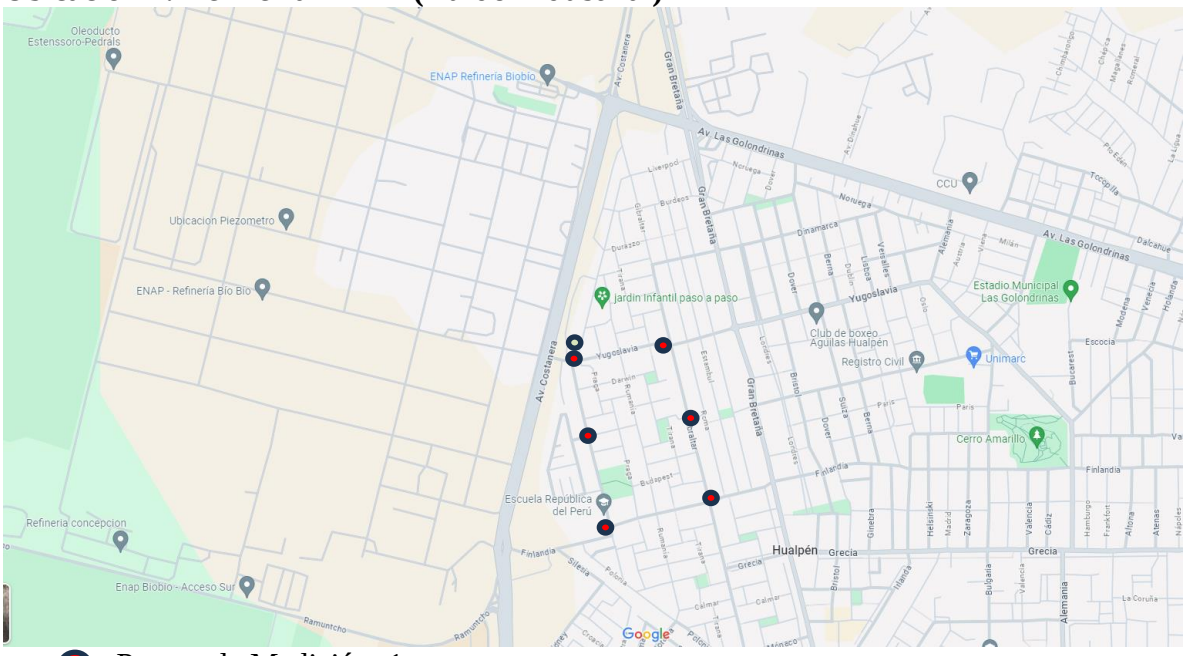
Tipos de prueba:
Cada una de las siguientes pruebas se deben llevar en cada una de las ubicaciones definidas más abajo, cada una de las mediciones hechas en cada prueba tendrá una duración de 5 minutos.

- Prueba 1: Comparación de error entre distintos dispositivos
  - Objetivo: Comparar mediciones hechas por distintos dispositivos en un mismo punto y en un mismo periodo de tiempo y determinar un margen de error común.
  - Detalle: En el punto definido para la ubicación correspondiente, se medirá durante 5 minutos con todos los dispositivos disponibles durante el mismo periodo de tiempo.
  - Procedimiento: Se definirá una hora de inicio común, una vez llegada esa hora todos comenzaran a medir, una vez pasados 5 minutos se detendrá la medición y se subirán los datos.
  - Resultados esperados: En un mismo punto y en el mismo periodo de tiempo, todos los dispositivos deberían medir un nivel de intensidad similar, con un rango de ±5dB de error, las clasificaciones obtenidas deberían ser similares.
- Prueba 2: Visualización de la distribución espacial del ruido
  - Objetivo: Obtener mediciones durante un periodo de tiempo definido en distintos puntos espaciales cercanos.
  - Detalle: Cada una de las personas se ubicará en cada uno de los puntos definidos, en una relación 1:1, y todos realizaran la medición durante un mismo rango de tiempo definido. Cada persona realizará la medición en solo uno de esos puntos.
  - Procedimiento: Se definirá una hora de inicio común para la medición, cada una de las personas se ubicará en uno de los puntos marcados, y al llegar el tiempo de inicio definido todos comenzarán a medir, al cumplirse el tiempo definido para la medición, 5 minutos, todos detendrán la medición y subirán los datos.
  - Resultados esperados: Las mediciones de intensidad obtenidas en distintos puntos en el mismo periodo de tiempo deberían depender de las distancias a la que el punto de medición se encuentre de la posible fuente.

Dispositivos por persona:

Nombre	Marca celular	Modelo Celular
Rodrigo	Samsung	S23
Alexis	Samsung	A15
Vicente	Samsung	Z Flip 5
Felipe	Samsung	Z Flip 5
Álvaro	Huawei	MAR-LX3A
Gustavo	Samsung	A50
Don Nelson	Samsung	S20
Ramón	Xiaomi	Redmi R3

Ubicación 1: Refinería ENAP (Ruido Industrial)



: Puntos de Medición, 1 por persona.
: Punto de medición general.

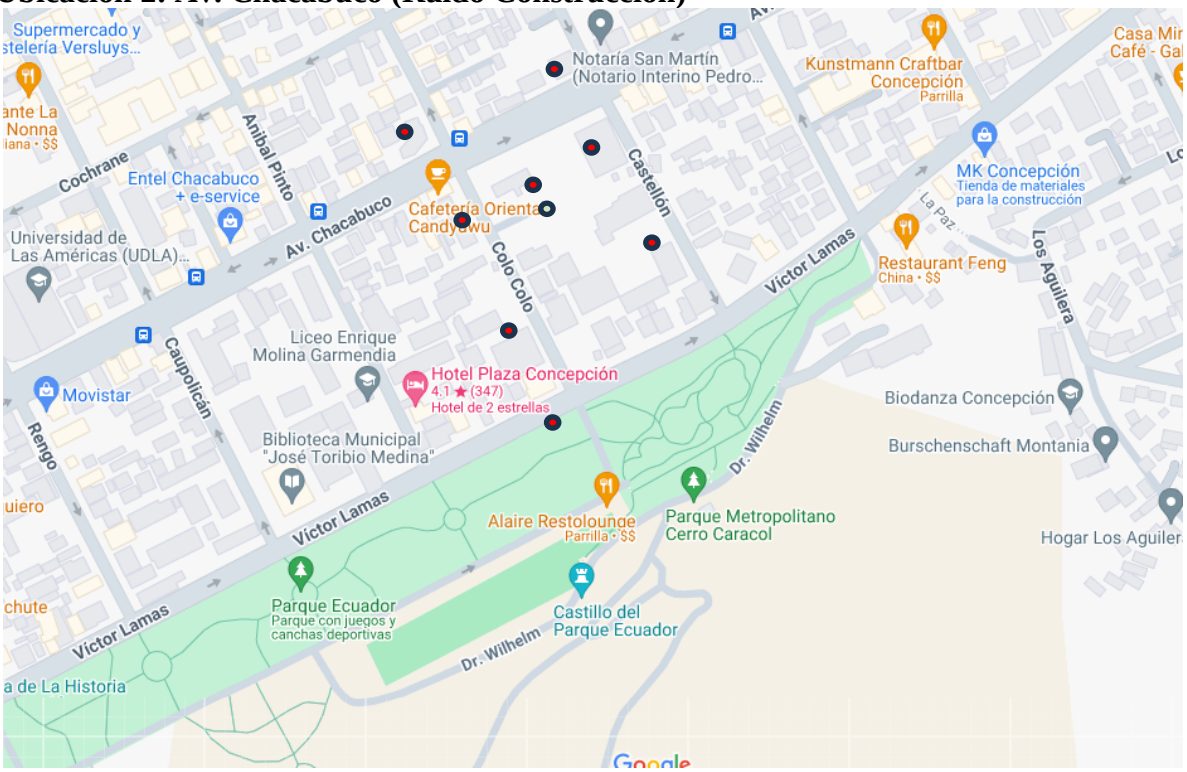
Resultados Prueba 1:

- Estado: Realizada

Resultado Prueba 2:

- Estado: Realizada

Ubicación 2: Av. Chacabuco (Ruido Construcción)



: Puntos de Medición, 1 por persona.
: Punto de medición general.

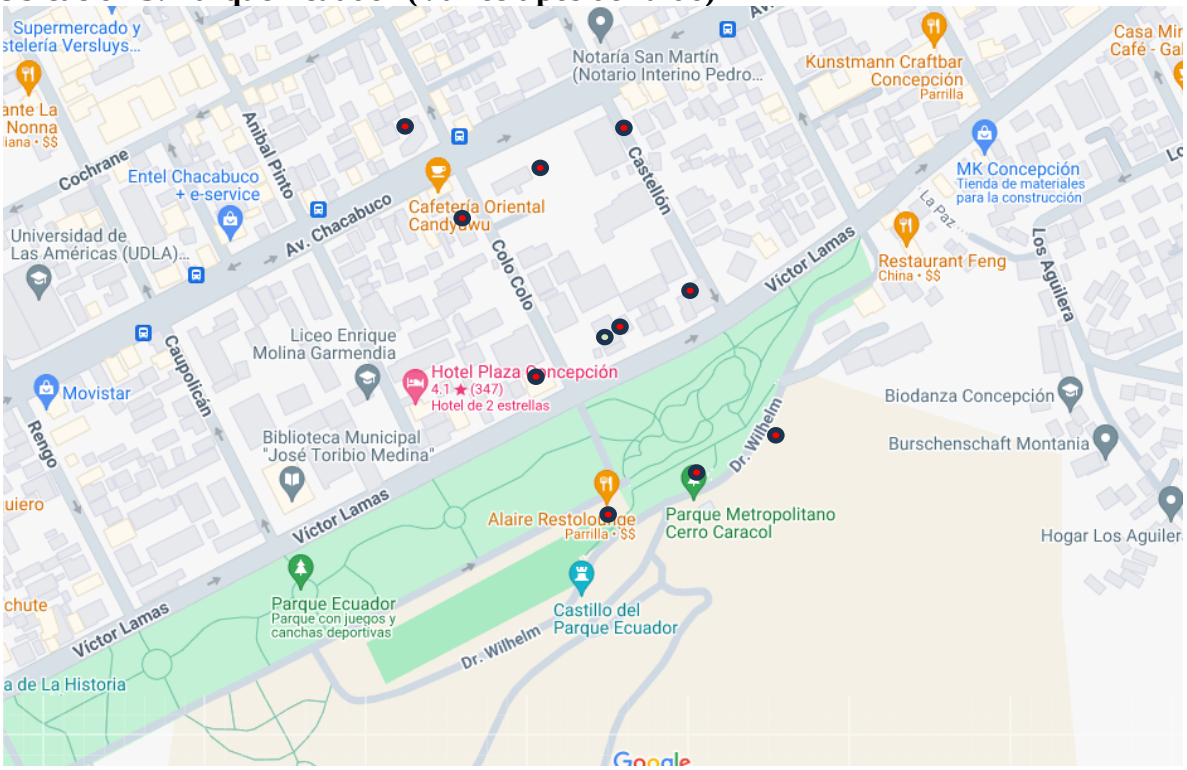
Resultados Prueba 1:

- Estado: Realizada

Resultado Prueba 2:

- Estado: Realizada

Ubicación 3: Parque Ecuador (Varios tipos de ruido)



: Puntos de Medición, 1 por persona.
: Punto de medición general.

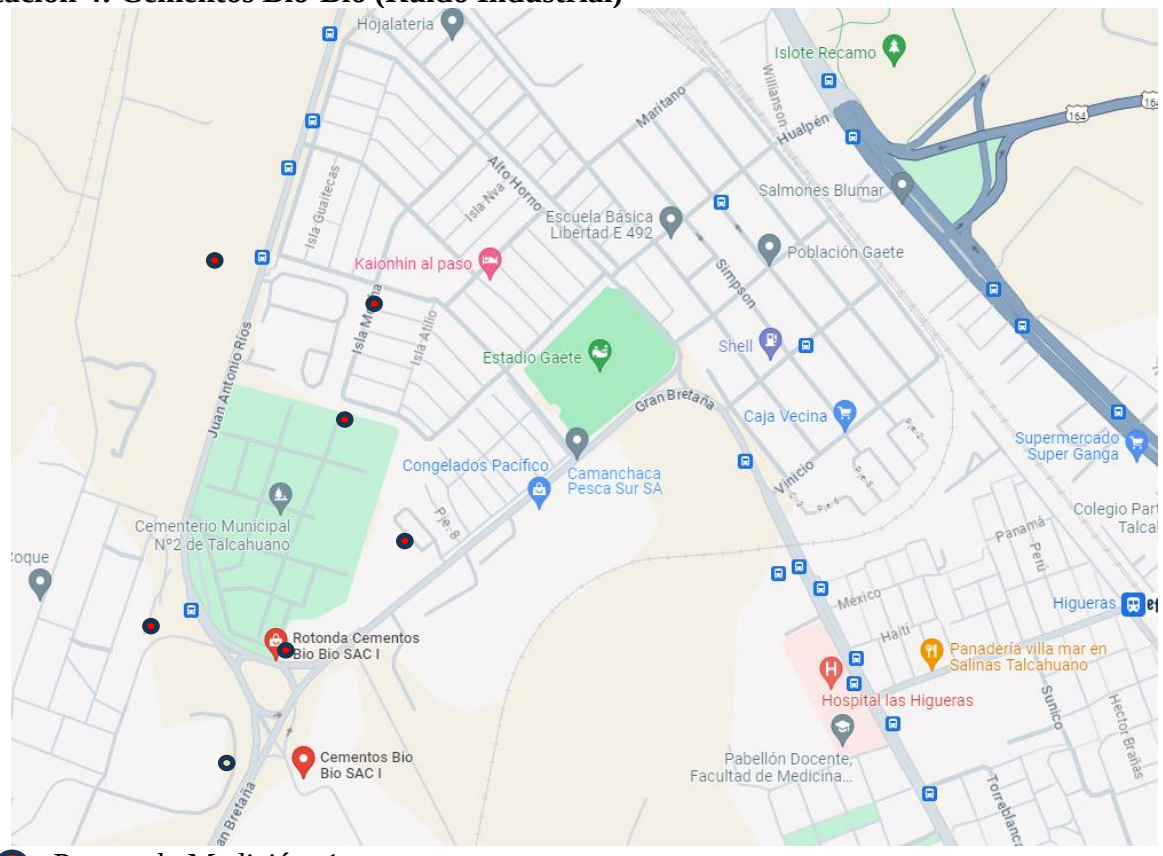
Resultados Prueba 1:

- Estado: No Realizada

Resultado Prueba 2:

- Estado: Realizada

Ubicación 4: Cementos Bio-Bio (Ruido Industrial)



: Puntos de Medición, 1 por persona.
: Punto de medición general.

Resultados Prueba 1:

- Estado: Realizada

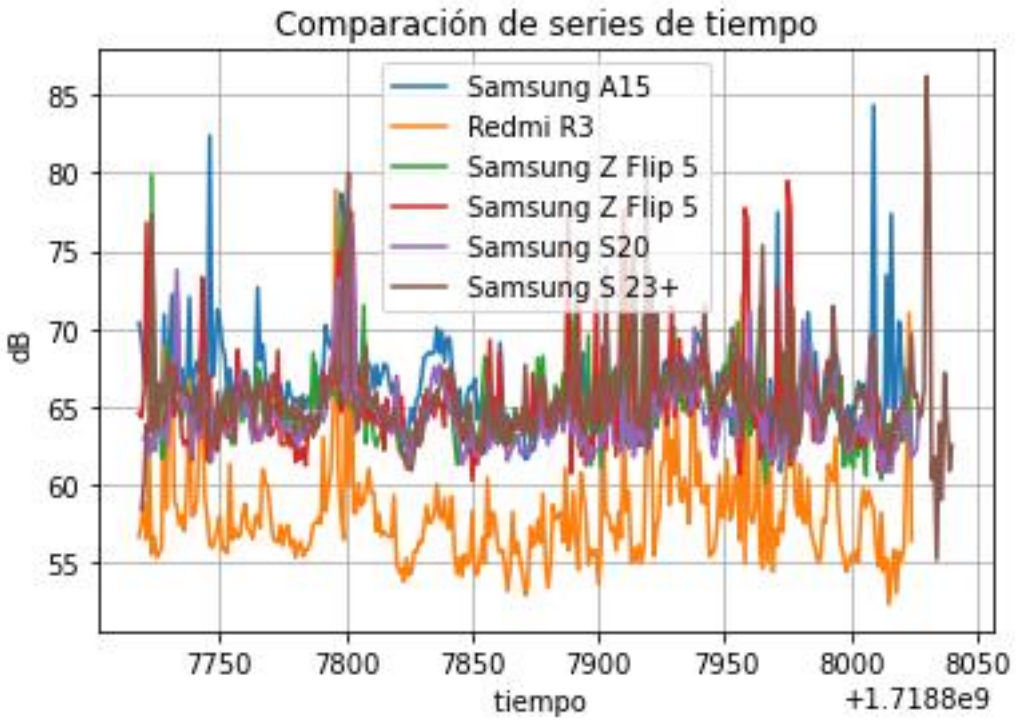
Resultado Prueba 2:

- Estado: Realizada

Anexo G: Resultados mediciones fuera de Cementos Biobío

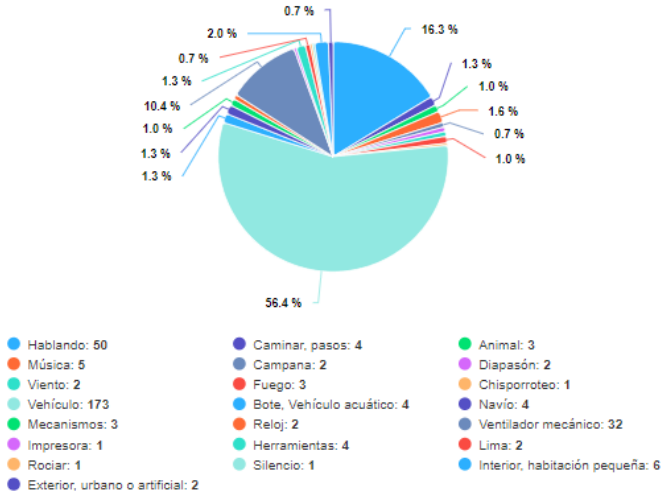
- Ubicación:

- Intensidad

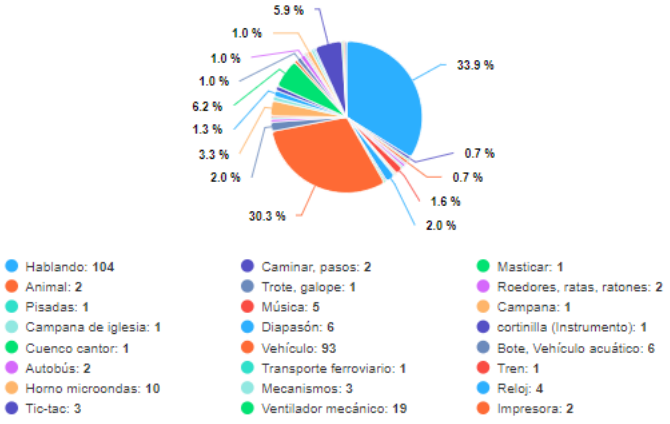


- Clasificaciones

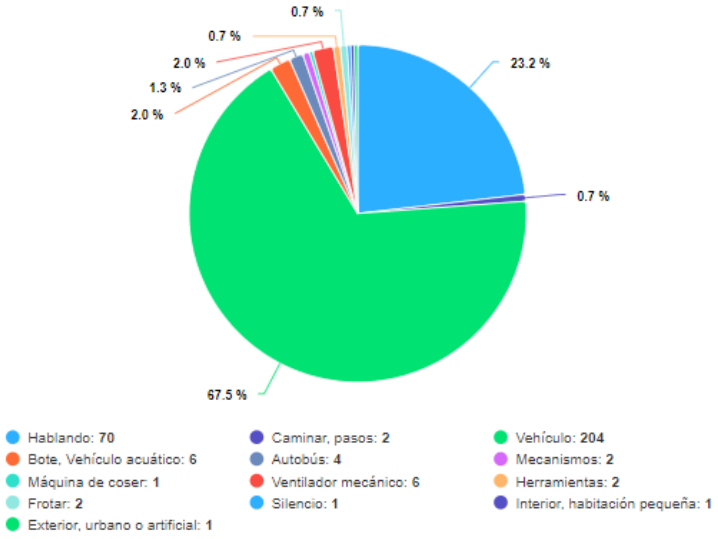
  - Samsung A15



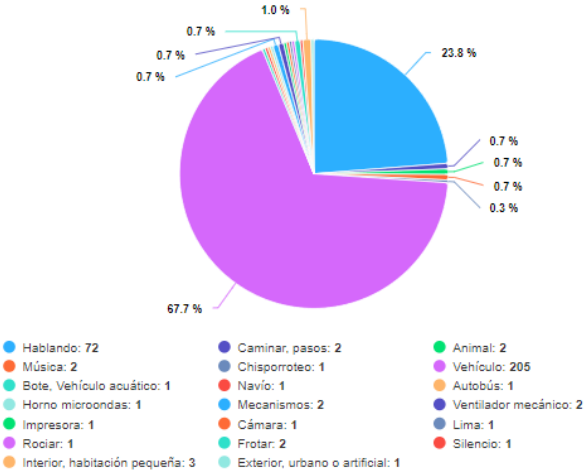
  - Redmi R3



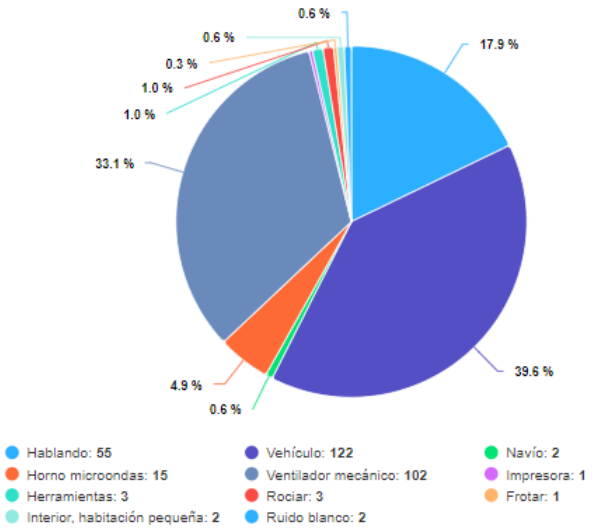
- Samsung Z Flip 5 (1)



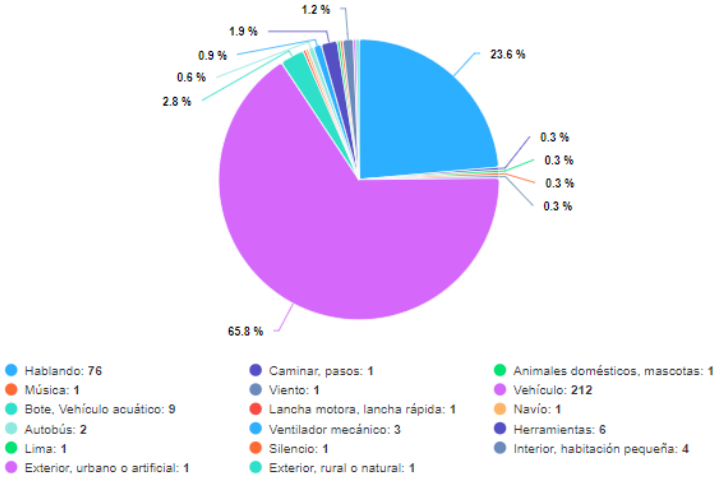
- Samsung Z Flip 5 (2)



- Samsung S20

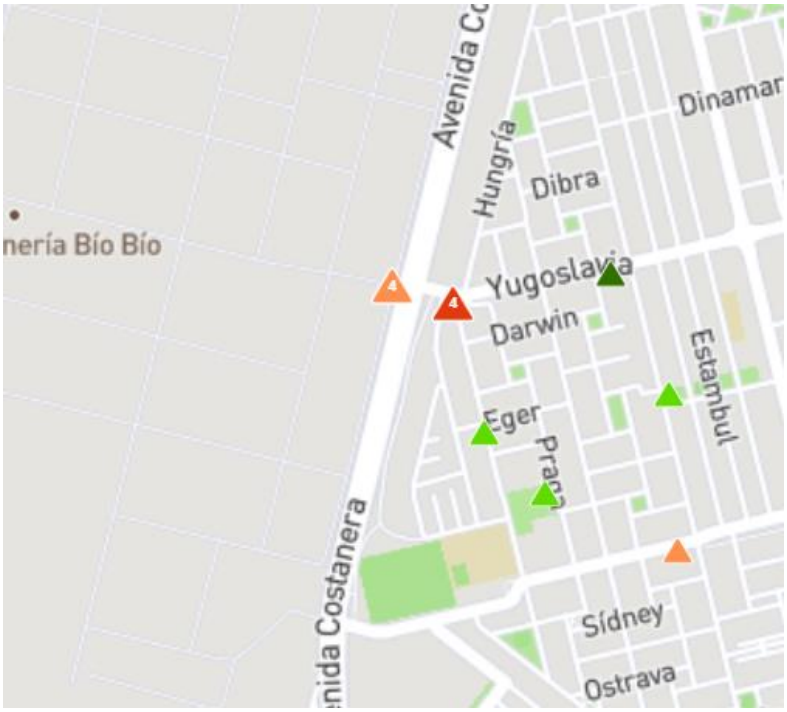


- Samsung S23

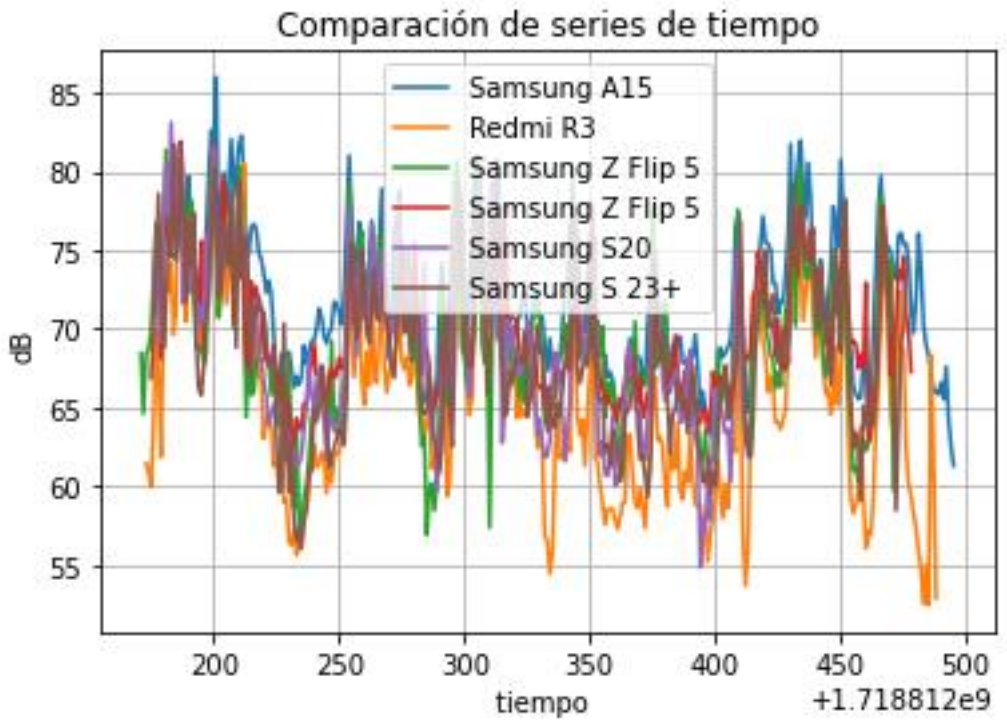


Anexo H: Resultados Mediciones fuera refinería ENAP Concepción

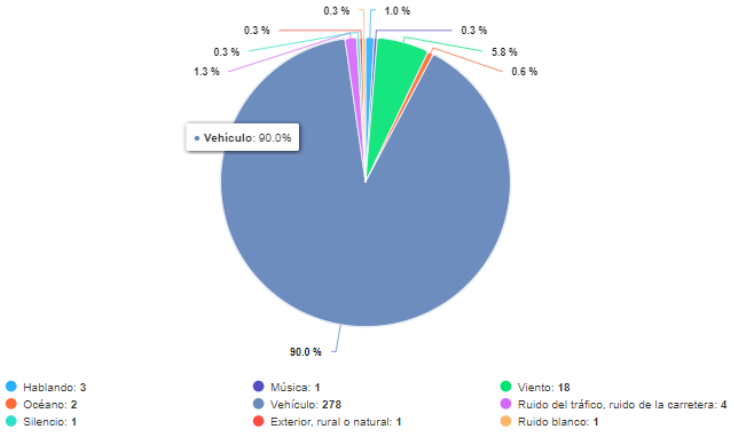
- Ubicación:



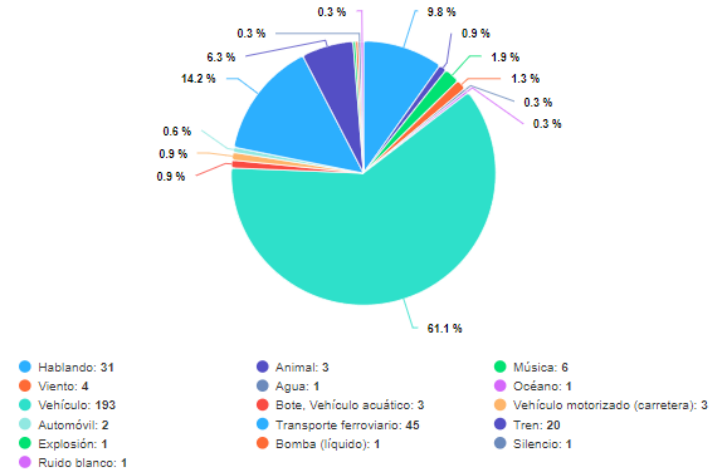
- Intensidad



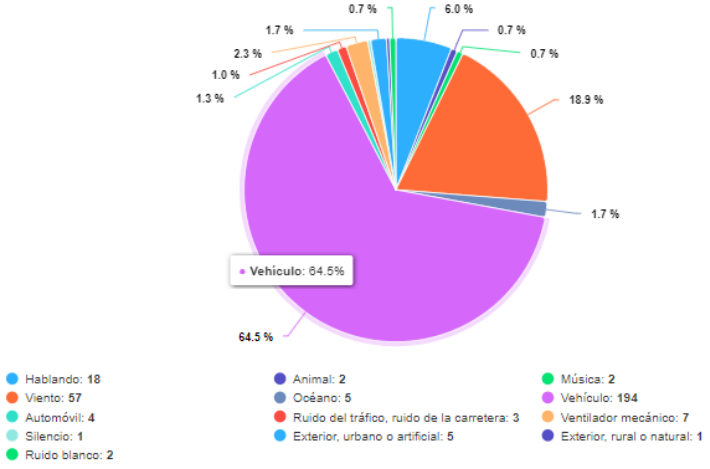
- Clasificaciones
  - Samsung A15



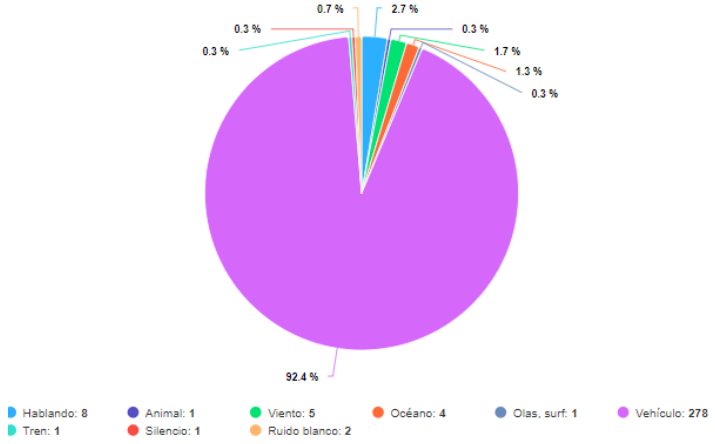
  - Redmi R3



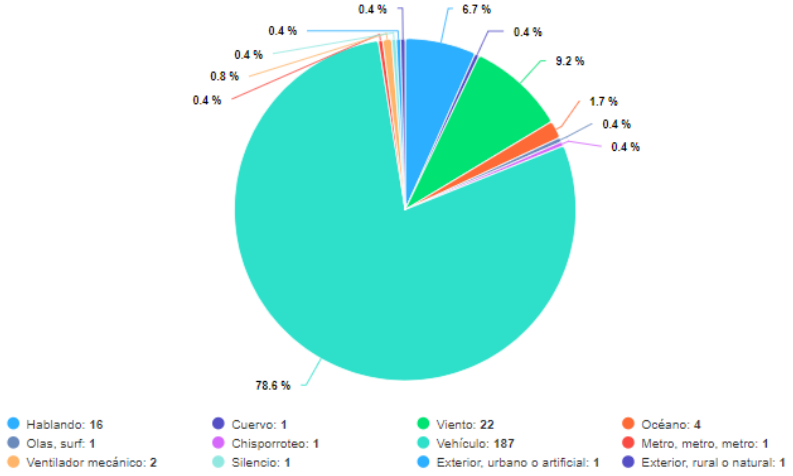
  - Samsung Z Flip 5 (1)



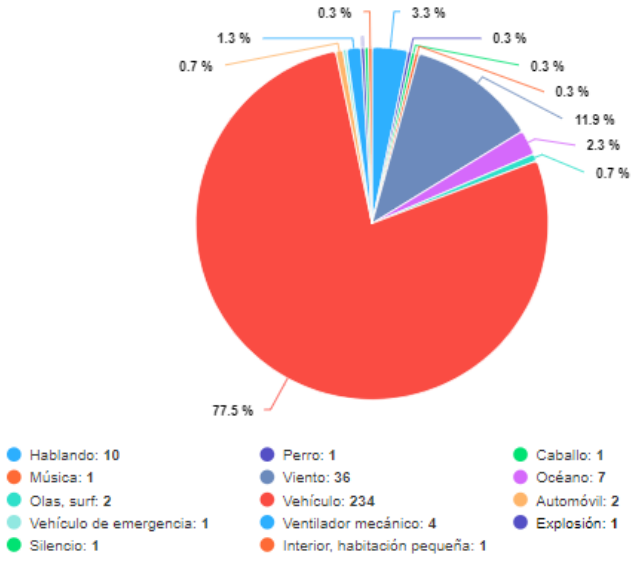
- Samsung Z Flip 5 (2)



- Samsung S20

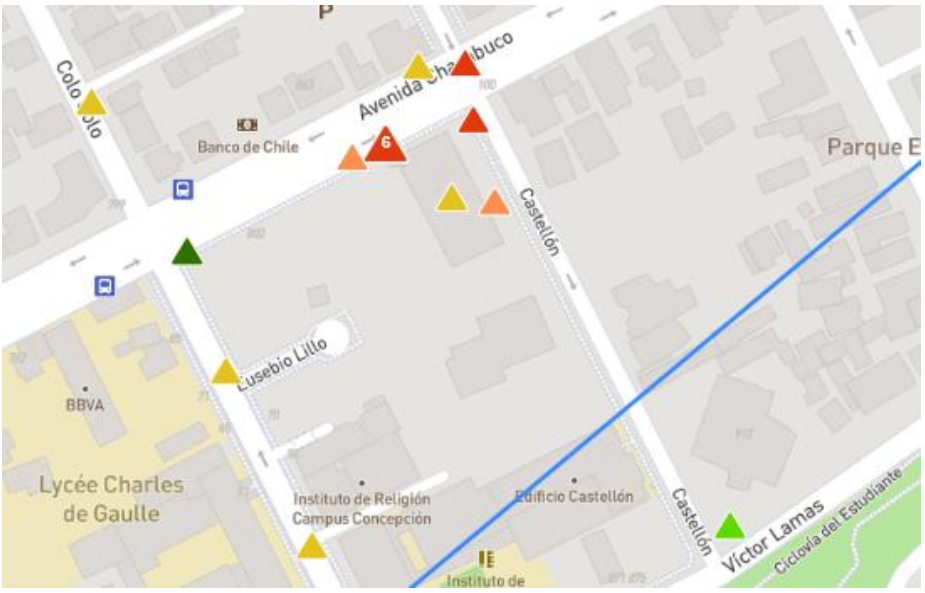


- Samsung S23

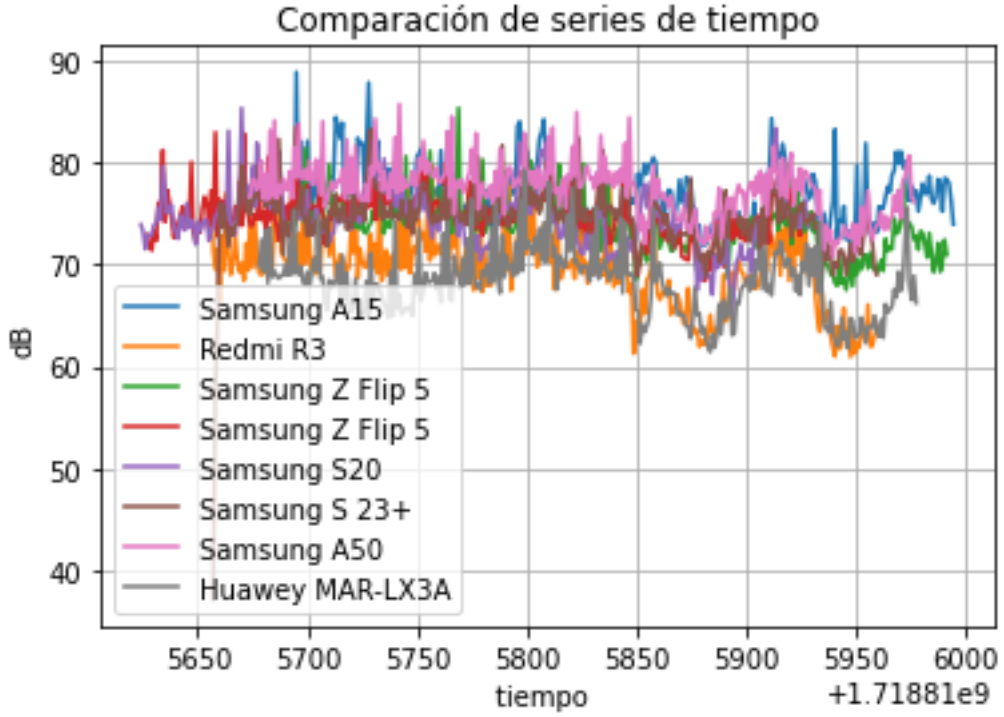

0.3 %
3.3 %
1.3 %
0.3 %
0.7 %
0.3 %
0.3 %
11.9 %
2.3 %
0.7 %
77.5 %
Hablando: 10
Perro: 1
Caballo: 1
Música: 1
Viento: 36
Océano: 7
Olas, surf: 2
Vehículo: 234
Automóvil: 2
Vehículo de emergencia: 1
Ventilador mecánico: 4
Explosión: 1
Silencio: 1
Interior, habitación pequeña: 1

Anexo I: Resultados Mediciones fuera de construcción en Avenida Chacabuco, Concepción

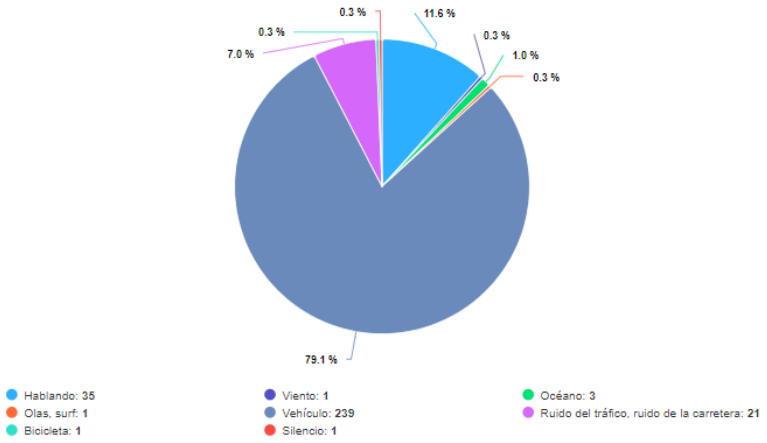
- Ubicación:



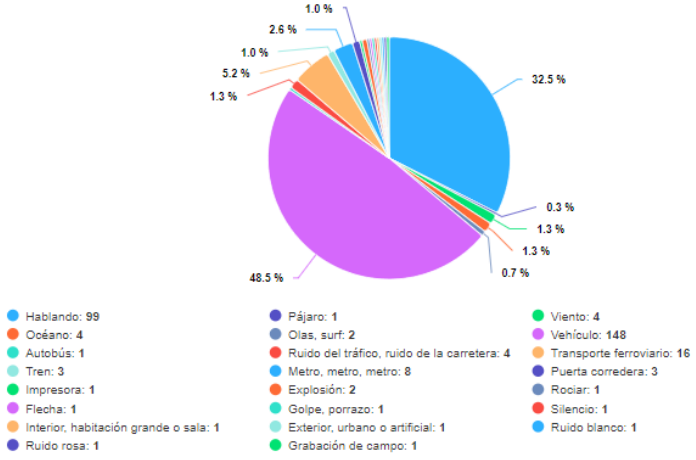
- Intensidad



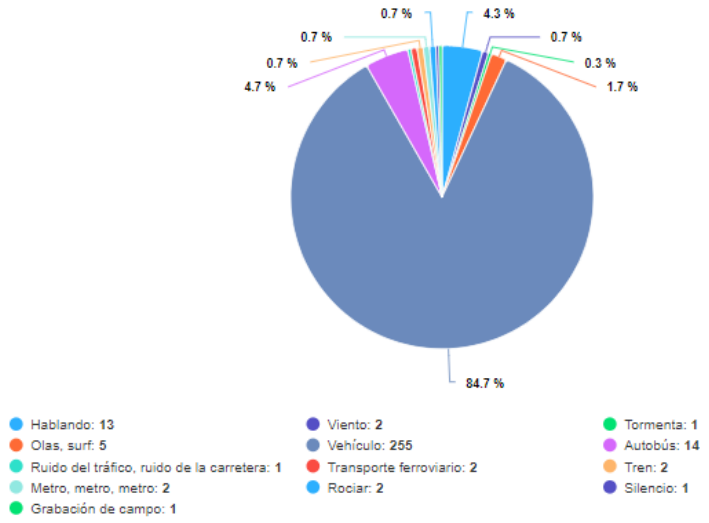
- Clasificaciones
  - Samsung A15



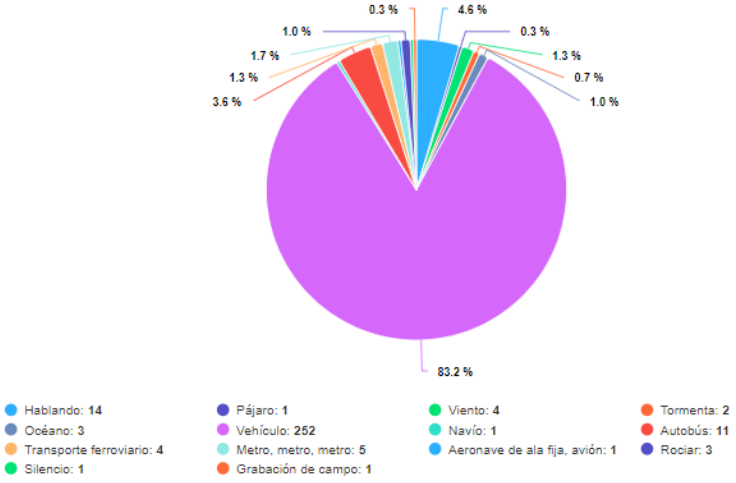
  - Redmi R3



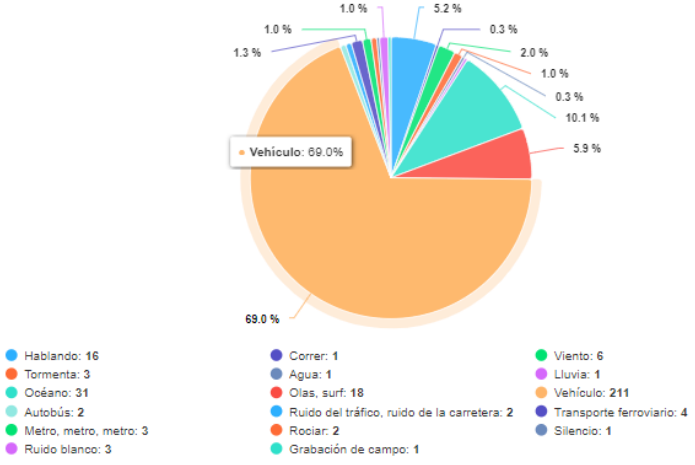
- Samsung Z Flip 5 (1)



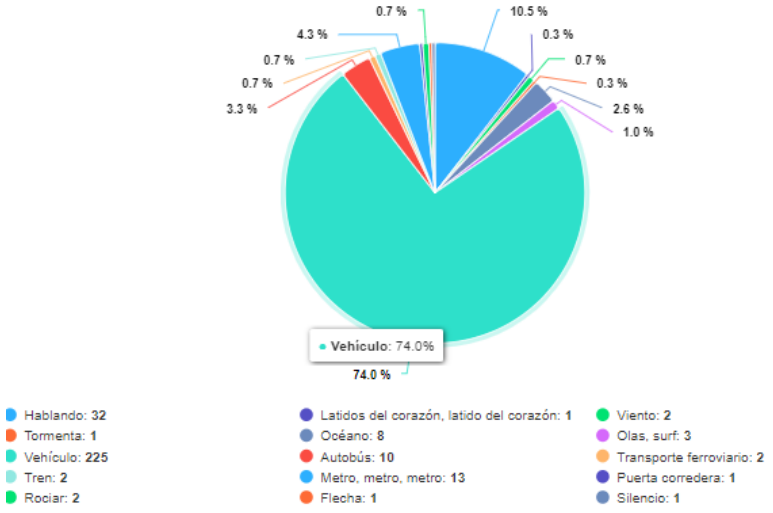
- Samsung Z Flip 5 (2)



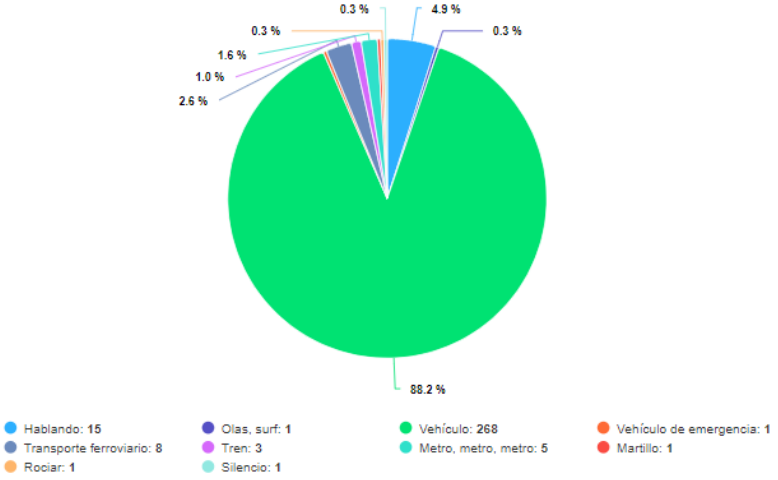
- Samsung S20



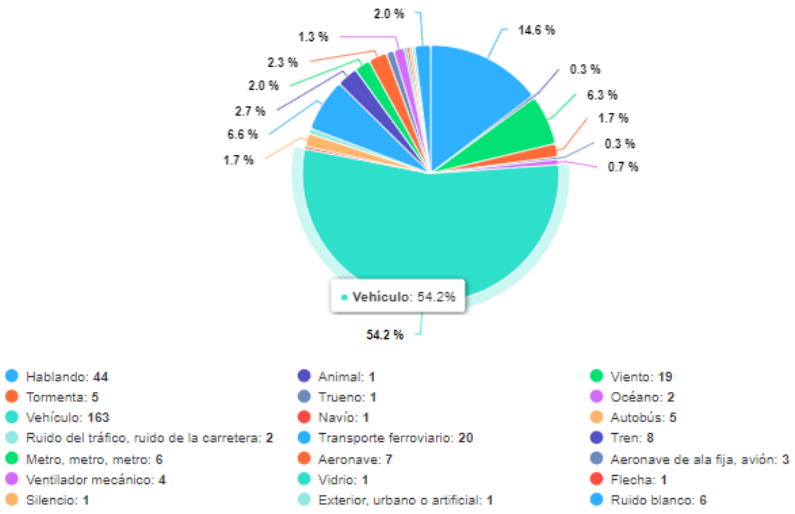
- Samsung S23



- Samsung A 50



- Huawey MAR-LX3A



UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA
RESUMEN DE MEMORIA DE TITULO

Departamento	**:** Departamento de Ingeniería Eléctrica
Carrera	: Ingeniería civil electrónica
Nombre del memorista	: Ramón Ignacio Castillo Retamal
Título de la memoria	**:** Plataforma para la medición y clasificación de series de tiempo de ruido geolocalizado
Fecha de la presentación oral	: 29 /08 / 2024
Profesor guía	: Mario Rubén Medina Carrasco
Profesor(es) revisor(es)	: Sebastián Godoy M. y Sergio Sobarzo G.
Concepto	**:**
Calificación	**:**

Resumen (máximo 200 palabras)

Los efectos negativos que la exposición prolongada a altos niveles de ruido puede provocar sobre nuestra salud es actualmente un tema de estudio, siendo el ruido un problema frecuente en zonas cercanas a carreteras con altos niveles de tráfico o en zonas de producción industrial, entre otros. Las normativas establecen limites sobre el nivel de ruido, y metodologías para medirlo, pero el alto costo de dispositivos en este tipo de medición dificulta su acceso por parte de la comunidad, la cual normalmente es la principal afectada con esta problemática.

En el presente trabajo se describe el desarrollo de una herramienta con la cual es posible caracterizar el ruido ambiental haciendo uso de un teléfono móvil, esta caracterización se hace en base a la medición de la intensidad o el nivel de ruido, la clasificación del ruido, la cual se obtiene haciendo uso de un modelo de clasificación de audio, y la ubicación en la cual se mide el ruido.

Para lo cual primero se investiga material bibliográfico respecto a qué es el ruido y cómo medirlo, clasificarlo y geolocalizarlo. Luego, se explica la arquitectura de la herramienta y los principales factores involucrados en su desarrollo. Después se mostrarán distintas pruebas realizadas con el objetivo de validar los métodos usados con los distintos propósitos antes mencionados. Finalmente se discutirán los resultados obtenidos y se entregarán conclusiones sobre el trabajo.

Como resultado de este trabajo se obtiene una herramienta de fácil acceso la cual permite medir el ruido con un alto nivel de precisión, con fines informativos, además de entregar una caracterización del ruido y la geolocalización de donde se hizo la medición para un posterior análisis de toda esta información.