



UNIVERSIDAD DE JAÉN  
*Escuela Politécnica Superior de Jaén*

Trabajo Fin de Grado

# **RECOLECCIÓN Y RECONOCIMIENTO DE EVENTOS DOMÉSTICOS MEDIANTE SENSORES DE SONIDO EN IoT**

**Alumno: José Manuel Vílchez Chiachío**

Tutores: Dr.D. Javier Medina Quero  
D<sup>a</sup>. Aurora Polo Rodríguez

Dpto: Arquitectura y Tecnología de  
Computadores

**Septiembre, 2021**

**Junio. 2021**





Universidad de Jaén  
Escuela Politécnica Superior de Jaén  
Departamento de Informática

D.Dr. Javier Medina Quero y D<sup>a</sup>. Aurora Polo Rodríguez tutores del Proyecto Fin de Grado titulado: **Recolección y reconocimiento de eventos domésticos mediante sensores de sonido en IoT**, que presenta José Manuel Vílchez Chiachío, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2021

El alumno:

Los tutores:

*José Manuel Vílchez Chiachío*

José Manuel VílchezChiachío

Javier Medina Quero

Aurora Polo Rodríguez

*Agradecimientos:*

*Transmitir mi más sincero agradecimiento a todos aquellos que me han ayudado a lo largo de esta etapa y han colaborado en esta investigación.*

*En primer lugar, a mi tutor, el Doctor Javier, por su ayuda en la planificación, información y organización en este Trabajo de Fin de Grado.*

*En segundo lugar, a mi familia, mi madre Josefa, mi padre Manuel Antonio y mis hermanos Marta y Antonio. A mis amigos y a mi pareja que han estado a lo largo de toda mi carrera apoyándome en todo momento y animándome a seguir adelante.*

*Desarrollar este estudio ha tenido un gran impacto en mi persona y es por eso que me gustaría agradecer a todas aquellas personas que me han apoyado durante este proceso.*

*A todos ellos, mil gracias.*

## Índice

1. Introducción.....	8
1.1. Motivación.....	8
1.2. Trabajos relacionados.....	10
1.3. Propuesta.....	12
1.4. Objetivos.....	13
1.5. Planificación temporal.....	13
1.6. Presupuesto.....	15
2. Estado del arte.....	18
2.1. Internet de las cosas.....	18
2.2. Redes neuronales convolucionales.....	21
2.3. Clasificación de sonidos ambientales.....	22
3. Proceso ingeniería software.....	24
3.1. Fases de la ingeniería del software.....	24
3.2. Especificación de requisitos.....	25
3.2.1. Requerimientos del sistema de recopilación de sonido ambiente.....	25
3.2.2. Requerimientos del sistema de segmentación de audios.....	26
3.2.3. Requerimientos del sistema etiquetado de clips de audios.....	26
3.2.4. Requerimientos del sistema algoritmos para el reconocimiento de sonidos.....	27
3.2.5. Requerimientos del sistema de evaluación offline del modelo.....	27
3.2.6. Requerimientos del sistema de evaluación online del modelo.....	28
3.3. Análisis del sistema.....	28
3.4. Diseño del sistema.....	36
3.4.1. Análisis de señales de audio.....	36
3.4.2. Etiquetado de audio.....	37
3.4.3. Herramientas para construir modelos de predicción.....	39
3.4.4. Evaluación de modelos.....	42
3.5. Implementación del sistema.....	47
3.5.1. Lenguaje de programación escogido.....	48
3.5.2. Preparación de la Raspberry pi.....	49
3.5.3. Entorno de programación usado.....	53
3.5.4. Recolección de datos y etiquetado.....	55
3.5.5. Conversión de audio a imagen (tratamiento de los datos).....	57

3.5.6. Arquitectura red neuronal .....	59
4. Evaluación.....	61
4.1. Pruebas.....	63
4.2. Casos.....	66
4.2.1. Modelo evaluado con la técnica de validación cruzada .....	66
4.2.2. Modelos para las escenas naturalistas.....	66
4.2.3. Tiempo de ejecución en Raspberry .....	68
4.3. Resultados .....	69
4.3.1. Modelo evaluado con la técnica de validación cruzada .....	69
4.3.2. Modelos para las escenas naturalistas.....	71
4.3.3. Tiempo de ejecución en Raspberry .....	78
5. Conclusiones y líneas de trabajo futuras .....	78
5.1. Conclusiones.....	78
5.2. Líneas futuras .....	80
Anexos .....	82
Scripts .....	82
Recolección de audio .....	82
Etiquetado.....	83
Segmentación de audio.....	84
Despliegue del sistema.....	84
Predicción en tiempo real.....	85
Bibliografía .....	87

## Índice de ilustraciones

Ilustración 1.1 Diagrama de Gantt de febrero a marzo .....	14
Ilustración 1.2 Diagrama de Gantt de abril a junio .....	15
Ilustración 1.3 Diagrama de Gantt de junio a agosto .....	15
Ilustración 3.1 Caso de uso 1: Recopilación de sonido ambiente .....	29
Ilustración 3.2 Caso de uso 2: Segmentación de audios .....	30
Ilustración 3.3 Caso de uso 3: Etiquetado de clips de audio.....	31
Ilustración 3.4 Caso de uso 4: Reconocimiento de sonidos.....	32
Ilustración 3.5 Caso de uso 5: Evaluación del modelo.....	34
Ilustración 3.6 Caso de uso 6 Evaluación del modelo en tiempo real .....	35
Ilustración 3.7 Ejemplo fichero CSV .....	37

Ilustración 3.8 Ejemplo fichero JSON .....	38
Ilustración 3.9 Ejemplo fichero XML .....	39
Ilustración 3.10 Arquitectura CNN .....	40
Ilustración 3.11 Capa convolución .....	41
Ilustración 3.12 Función activación relu .....	42
Ilustración 3.13 Descripción matriz de confusión .....	43
Ilustración 3.14 Ejemplo matriz confusión.....	44
Ilustración 3.15 Divisiones en cross validation (k-fold).....	45
Ilustración 3.16 Dominio de frecuencia del evento ducha .....	46
Ilustración 3.17 Espectrograma de mel del evento ducha.....	47
Ilustración 3.18 MFCC del evento ducha .....	47
Ilustración 3.19 Uso de lenguajes de programación para aprendizaje automático.....	48
Ilustración 3.20 RaspBerry PI .....	49
Ilustración 3.21 Interfaz aplicación balenaEtcher .....	50
Ilustración 3.22 Fichero configuración para la interfaz de red .....	51
Ilustración 3.23 Interfaz de desarrollo de jupyter notebook.....	55
Ilustración 3.24 Señal de audio de los eventos a clasificar .....	58
Ilustración 3.25 Espectrograma de mel de los eventos a clasificar .....	59
Ilustración 3.26 MFCC de los eventos a clasificar .....	59
Ilustración 4.1 Despliegue en escena 1 .....	64
Ilustración 4.2 Despliegue en escena 2 .....	64
Ilustración 4.3 Despliegue en escena 3 .....	65
Ilustración 4.4 Despliegue en escena 4 .....	65
Ilustración 4.5 Matriz confusión con la técnica validación cruzada .....	70
Ilustración 4.6 Matriz de confusión de la escena 1 .....	71
Ilustración 4.7 Matriz de confusión de la escena 2 .....	73
Ilustración 4.8 Matriz de confusión de la escena 3 .....	75
Ilustración 4.9 Matriz de confusión de la escena 4 .....	77
Ilustración 5.1 Publicación de artículo sobre este TFG .....	81

## Índice de tablas

Tabla 1.1 Coste de los Recursos Humanos.....	16
Tabla 1.2 Coste del Hardware. ....	17
Tabla 1.3 Coste del software y los consumibles. ....	17

Tabla 1.4 Coste total del proyecto. ....	17
Tabla 3.1 Arquitectura de red del modelo CNN + MFCC .....	60
Tabla 4.1 Tabla de parámetros del DataSet utilizado para generar y evaluar el modelo.....	62
Tabla 4.2 Descripción eventos clasificados .....	63
Tabla 4.3 Objetos con los que se interactúa en la escena 1 .....	66
Tabla 4.4 Objetos con los que se interactúa en la escena 2 .....	67
Tabla 4.5 Objetos con los que se interactúa en la escena 3 .....	68
Tabla 4.6 Objetos con los que se interactúa en la escena 4 .....	68
Tabla 4.7 Resultados evaluación cruzada .....	69
Tabla 4.8 Resultados evaluación escena 1 .....	72
Tabla 4.9 Resultados evaluación escena 2 .....	74
Tabla 4.10 Resultados evaluación escena 3 .....	76
Tabla 4.11 Resultados evaluación escena 4 .....	78
Tabla 4.12 Tiempo evaluación de validación cruzada en RaspBerry .....	78



## 1. Introducción

### 1.1. Motivación

La **clasificación de sonidos ambientales** es un tema activo en la investigación, en reciente crecimiento y de interés debido al crucial rol que tienen los sonidos en nuestra interacción con el entorno [1]. El sonido es una rica fuente de información que se puede utilizar para inferir el contexto de una persona en la vida diaria [2]. Los patrones que producen ciertos eventos en el sonido hacen que podamos inferir qué actividades está realizando un usuario.

Por otro lado, las **redes neuronales convolucionales** han tenido resultados innovadores durante la última década en una variedad de campos relacionados con el reconocimiento de patrones [3]. Extraer características de los sonidos que producen los eventos del usuario puede desarrollarse mediante un modelo visual previamente entrenado con una red neuronal convolucional para predecir un resumen estadístico del sonido asociado [4]. Es por esto que la investigación de este proyecto está enfocado a la detección de sonidos ambientales en los hogares inteligentes, con el objetivo de mejorar la calidad de vida permitiendo que las personas permanezcan independientes en sus propios hogares durante el mayor tiempo posible [5].

El **Internet de las cosas (Internet of Things - IoT)** tiene como objetivo unificar todo en nuestro mundo bajo una infraestructura común, dándonos no sólo el control de las cosas que nos rodean, sino también manteniéndonos informados de su estado [6]. Esto último resulta para nosotros de especial interés, ya que queremos inferir los eventos producidos por cada sonido. Por ello, el estudio de un sensor de sonido en un dispositivo IoT para extracción de características de los sonidos producidos por un evento doméstico, y la clasificación que se hace de ellos con nuestros propios modelos previamente entrenados, tiene una especial motivación como enfoque de este trabajo fin de grado.

En la comunidad científica, la aparición de conjuntos de datos públicos ha dado lugar a avances en la detección acústica y la clasificación de escenas y eventos [7]. Para que el modelo que se ha desarrollado en este trabajo fin de grado pueda

realizar predicciones a través de sonidos, se ha procedido a recolectar sonidos que se han grabado cuando han ocurrido ciertos eventos en el hogar, ya que en algunos casos los datos no son públicos o lo suficientemente buenos. De esta manera, se genera un amplio Dataset correctamente etiquetado que posteriormente se ha utilizado para el entrenamiento de nuestros modelos.

Adicionalmente, también resulta de especial interés el hecho de que con un solo dispositivo se pueda aportar conocimiento extra que permita inferir la actividad humana en un hogar inteligente, ya que tiene como consecuencia una disminución de costes en este tipo de sistemas en hogares debido a la sustitución de sensores que se encargaban de inferir estos eventos producidos por el usuario.

Algunas de las tecnologías en las que nos basamos para la propuesta realizada que también motiva este proyecto son:

- **RaspBerry Pi 3 model B:** Dispositivo IoT en el que se acopla el sensor de sonido y se conecta a la red WiFi.
- **Lenguaje programación Python:** Lenguaje en auge para el desarrollo de redes neuronales convolucionales que nos permite crear el modelo con el que hacemos la clasificación de los eventos de cada sonido.
- **FIFINE Micrófono USB:** Sensor de audio que se utilizará para la grabación de sonidos generados por el usuario en un hogar y para su posterior etiquetación.

Gracias a estas tecnologías, hemos llevado a cabo este Trabajo de Fin de Grado (TFG), de modo que se ha obtenido un modelo capaz de predecir eventos acústicos en los distintos lugares del hogar.

## 1.2. Trabajos relacionados

Integrar la tecnología ubicua e inmersiva para mejorar nuestra vida diaria pasando a un segundo plano en nuestras vidas, ha sido conocida como la era de la tecnología tranquila [13].

Desde esta perspectiva visionaria de principios de los 90 hasta nuestros días de Internet de las cosas, se han explotado dos características clave durante los últimos 30 años:

- Baja invasividad o inmersión dentro de dispositivos integrados (tanto en el cuerpo como en el entorno).
- Dispositivos inteligentes conectados que proporcionan resultados interpretables a partir de la información recopilada por sensores.

Como describimos anteriormente, con el propósito de implementar sensores inmersivos, se propusieron sensores binarios ambientales para describir las actividades diarias en espacios interiores [14] proporcionando resultados alentadores con un conjunto de datos etiquetado [15] en el marco de enfoques basados en datos [9]. Hoy en día, el creciente crecimiento de dispositivos está promoviendo los sensores multimodales, que normalmente integra sensores de video, audio y dispositivos portátiles [16], y otros dispositivos de IoT con capacidades informáticas cada vez más altas. Las nuevas tendencias convergen hacia sensores sintéticos [17], que están conectados para detectar todo en una habitación determinada, uso de tecnologías de detección de propósito general para monitorear actividades mediante la combinación de sensores. En este contexto, el procesamiento de audio por micrófonos inteligentes para el etiquetado de eventos audibles está abriendo un campo de investigación prometedor en la realidad aumentada (AR - Augmented Reality) [8].

Sobre la arquitectura de componentes para el aprendizaje y la comunicación de dispositivos, los paradigmas de Edge Computing [18] o Fog Computing [19] han ubicado los datos y servicios dentro de los dispositivos donde se recopilan los sensores, proporcionando *recursos virtualizados y servicios comprometidos basados en la ubicación, en el borde de las redes móviles*. Bajo una nueva perspectiva de

Internet de las cosas (IoT) para desarrollar objetos inteligentes colaborativos que *interactúen entre sí y cooperen con sus vecinos para alcanzar objetivos comunes* [20]. En los modelos de aprendizaje automático para AR, la descripción de la información del sensor en enfoques basados en datos ha dependido del tipo de sensores, ya sean inerciales [21] o sensores binarios [15], se han propuesto metodologías para explotar características espaciales temporales [22]. Además, el aprendizaje profundo (DL - Deep Learning) también se ha mostrado como un enfoque adecuado en AR para descubrir y extraer características de los sensores [23]. DL está relacionado con el reconocimiento multimodal de sensores, tales como, visión y audio donde la obtención de características jerárquicas para reducir la complejidad son clave. En sensores de visión se propone una visión térmica para garantizar la privacidad, previniendo circunstancias peligrosas como caídas haciendo uso de redes neuronales convolucionales (CNN - Convolutional Neural Networks) [24]. En el campo del reconocimiento de audio, la combinación de redes neuronales convolucionales (CNNs) [25] con el uso del espectrograma como representación del sonido [26] se ha demostrado que genera resultados alentadores en el reconocimiento de sonido útil para la clasificación de sonidos ambientales [27-29] y análisis de señales musicales [30,31]. En concreto, se ha propuesto tanto el uso del espectrograma log-mel (LM) como el coeficiente cepstral de frecuencia mel (MFCC) tiene una representación robusta en la clasificación de sonido [32].

En el campo de los sonidos ambientales en los espacios interiores, destacamos los siguientes enfoques: en reconocimiento de eventos, como una pelota que rebota o un cricket, ha sido desarrollada utilizando la representación de sonido espectral con características de nivel de frame que fue aprendido por Markov Models. Por otro lado en condiciones naturalistas dentro de una residencia geriátrica se clasificaron dos categorías de sonido [33] (tapping y washing hands) donde se reconocen usando espectrogramas e histogramas de sonidos por SVM. En parte del trabajo [8], los micrófonos direccionales espaciales 3D permiten la captura de audio multidireccional de alta calidad para detectar eventos y la ubicación del sonido en un entorno. Para eso, los coeficientes cepstrales se calculan como características espaciales que están relacionadas con eventos usando Modelos de Markov ocultos gaussianos. En [34], se recopilaron 30 eventos para reconocer la siete habitaciones

o espacios donde el habitante desarrolla actividades (baño, dormitorio, casa de entrada, cocina, oficina, exterior y taller).

La extracción de características tiene un gran efecto en el entrenamiento y el reconocimiento de audio en el sistema de reconocimiento de audio. El algoritmo MFCC es un método de extracción de características típico con un rendimiento estable y una alta tasa de reconocimiento [35].

Por otra parte, las funciones de MFCC se utilizan para producir una mayor precisión de reconocimiento de sonidos ambientales. Se llevan a cabo extensos experimentos para demostrar la efectividad de estas funciones para entornos ambientales no estructurados, clasificación de sonido, incluso pruebas de audición en humanos para estudiar sus capacidades de reconocimiento. El sistema de reconocimiento implementado con MFCC ha demostrado producir un rendimiento comparable al de los oyentes humanos [36].

Para cada sonido a clasificar, los espectrogramas log mel también se evaluaron junto con un modelo de DL (VGG-16) con audios de Youtube que dieron un resultado alentador, donde se demostró que la precisión cambia notablemente en condiciones controladas y en la vida real. Con base en los trabajos y enfoques relacionados en esta sección, en este trabajo presento un conjunto de datos centrado en eventos diarios en espacios interiores que pueden mejorar la AR utilizando placas inteligentes de IoT. El modelo propuesto para el reconocimiento de audio se basa en información espectral de muestras de audio junto con el aprendizaje de las CNN, proporcionando un alto reconocimiento de rendimiento con extracción automática de características espaciales. Además, se han evaluado escenas en condiciones naturalistas para analizar el impacto del reconocimiento de eventos diarios en tiempo real.

### **1.3. Propuesta**

El propósito de este trabajo fin de grado es:

- Recopilar un conjunto de datos de muestras de audio relacionadas con eventos diarios que se generan dentro de los espacios interiores.

- Evaluar el rendimiento de los modelos de aprendizaje profundo en el reconocimiento del entorno, eventos en tiempo real y offline y desarrollo de actividades diarias en condiciones naturalistas.
- En último lugar, evaluar la posibilidad de integrar el modelo en tiempo real incluyendo un caso de estudio de una escena real.

#### 1.4. Objetivos

Los objetivos específicos de este TFG son:

- Configuración de un dispositivo IoT con sensores de sonido inmersivos.
- Desarrollo de un software de recolección y etiquetado de muestras de sonido sobre diferentes eventos diarios en el dispositivo IoT.
- Diseño e implementación de un modelo basado en Deep Learning que permita clasificar los sonidos.
- Evaluación de la clasificación de sonidos ambientales en un entorno doméstico.
- Evaluar el modelo en tiempo real incluyendo un caso de estudio de una escena real.

#### 1.5. Planificación temporal

En este apartado se mostrará un diagrama de Gantt que es una herramienta que nos permitirá exponer el tiempo dedicado para cada tarea que se ha realizado en este trabajo fin de grado, incluido las fechas establecidas para cada tarea. Las tareas en las que se divide la organización de este proyecto son las siguientes:

- **Búsqueda bibliográfica:** Esta fase se buscó documentación sobre las tecnologías utilizadas en este proyecto, dispositivos IoT de sonido, tensorflow y entornos de desarrollo como jupyter notebook.
- **Recopilación de audio:** Se construyó un DataSet de audios de los eventos domésticos que queríamos clasificar, estos audios fueron recolectados con una raspberry pi y un sensor de audio en las

habitaciones de más interés para el estudio y realización de este trabajo fin de grado.

- **Desarrollo de software:** Abarca todo el desarrollo, depuración, pruebas, aprendizaje sobre las tecnologías utilizadas mencionadas anteriormente, etiquetado de audios y grabación de escenas naturalistas.
- **Documentación:** Se ha redactado una memoria que recoge toda la información anteriormente citada, exponiendo resultados y conclusiones sacadas de la elaboración de esta investigación.

A continuación, se adjunta unas ilustraciones del diagrama de Gantt desglosado:

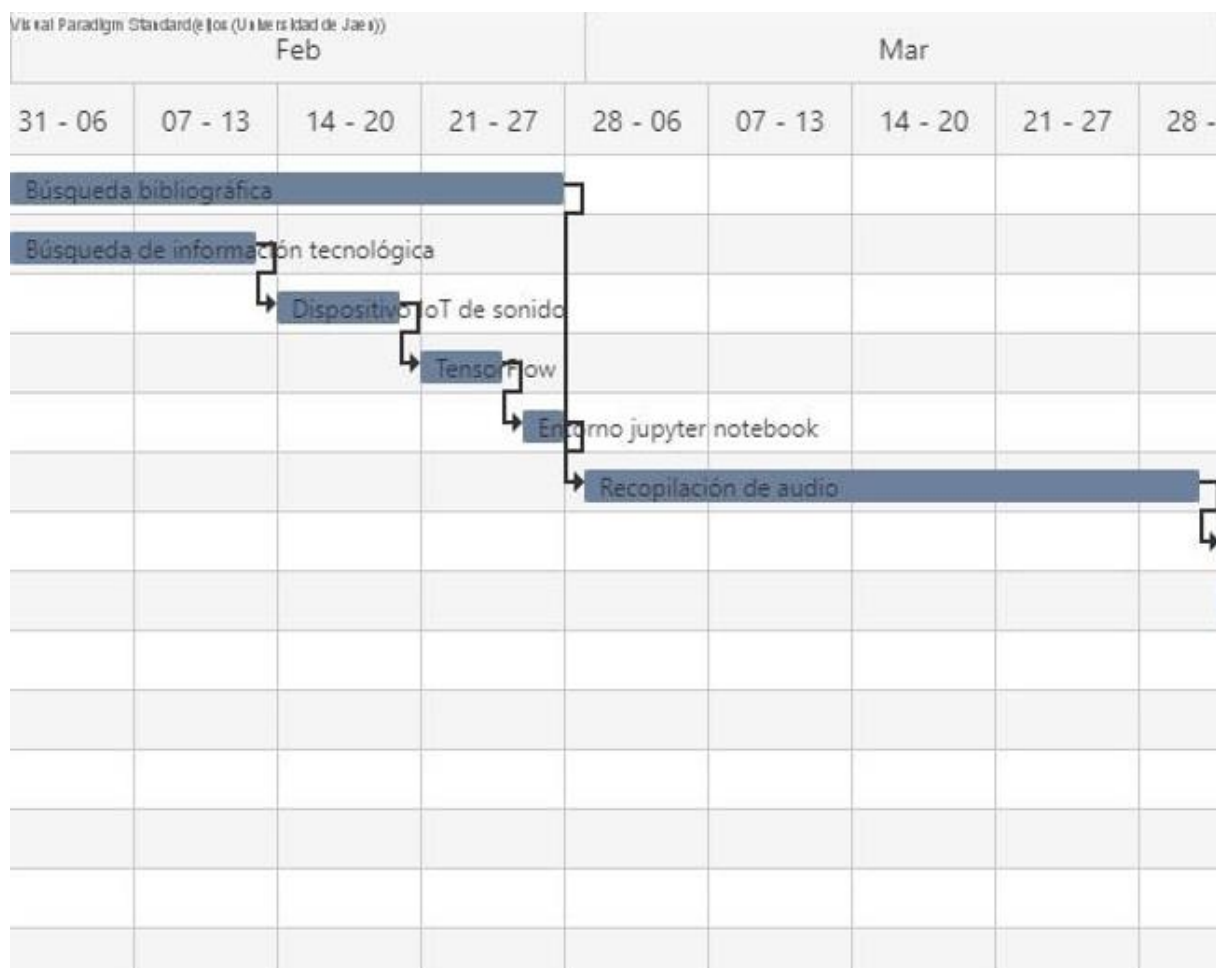


Ilustración 1.1 Diagrama de Gantt de febrero a marzo

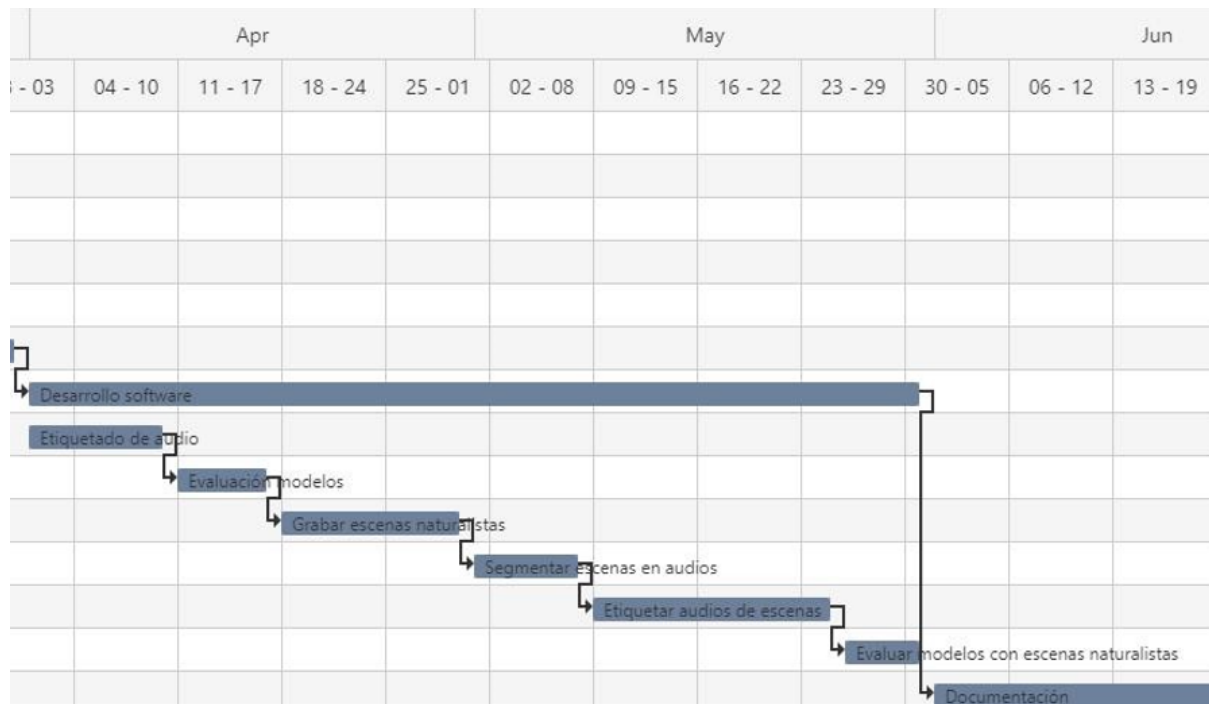


Ilustración 1.2 Diagrama de Gantt de abril a junio

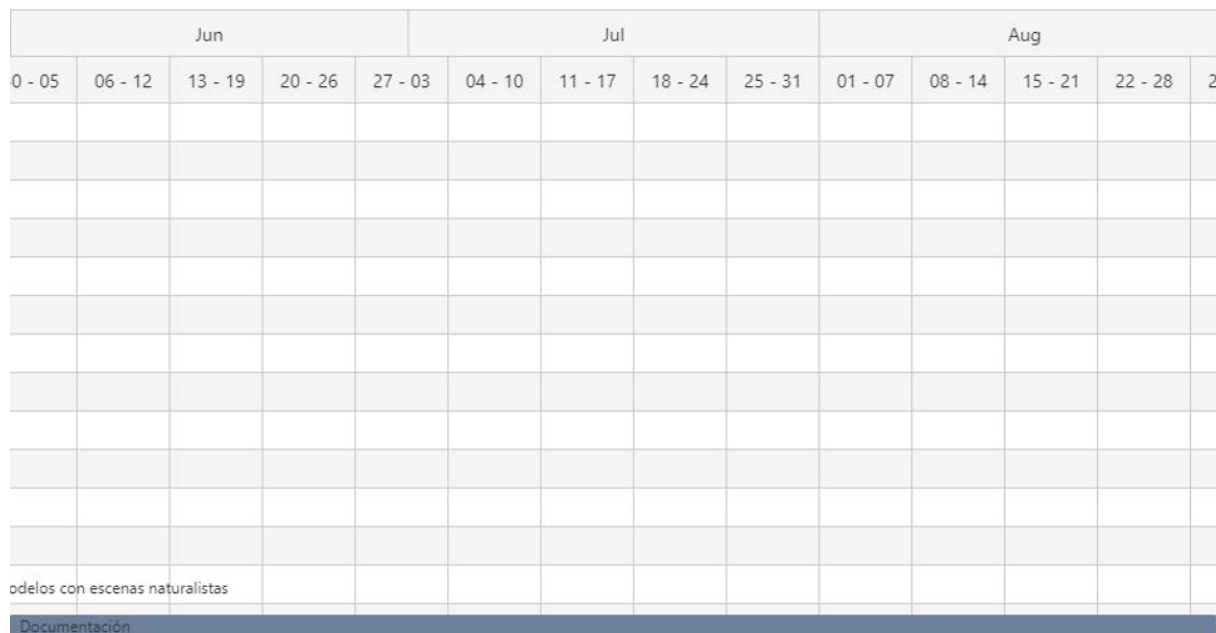


Ilustración 1.3 Diagrama de Gantt de junio a agosto

## 1.6. Presupuesto

En este apartado se realiza una estimación de los costes de este proyecto en caso de haber sido un desarrollo profesional. Para esto tomamos en cuenta las siguientes consideraciones iniciales:



- El proyecto ha sido desarrollado por una persona con un nivel profesional equivalente al de ingeniero junior.
- El tiempo de desarrollo del proyecto es de siete meses, en los que no se ha seguido una jornada de trabajo continua. Es decir, no se ha conseguido por ejemplo una jornada de oficina de 8 horas diarias. Si no que se ha ido realizando según la disponibilidad de tiempo (incluyendo trabajo en sábados, domingos y festivos.). Se han ido apuntando diariamente las horas de trabajo, y gracias a ello podemos hacer el cálculo de las horas totales de trabajo.

Vamos a dividir el presupuesto en dos bloques: recursos humanos y materiales. Pasamos a describirlos.

#### Recursos humanos:

Durante la realización del trabajo fin de grado se han ido anotando, como hemos dicho, las horas de trabajo diarias de forma aproximada. El resultado total se ha alcanzado tras unas 450 horas de trabajo.

Para los costes de los recursos humanos se ha considerado el siguiente coste por hora:

Ingeniero Junior.....12€/h

Teniendo en cuenta lo anterior, los gastos en recursos humanos serían:

Ingeniero Junior (12€/h x 450 h).....5400€

A continuación se muestra una tabla con la información anterior:

Elemento	Coste (€)
Recursos Humanos: Ingeniero Junior	5400
<b>COSTE TOTAL RECURSOS HUMANOS</b>	<b>5400</b>

Tabla 1.1 Coste de los Recursos Humanos.

#### Recursos materiales:

Los recursos materiales utilizados podemos desglosarlo en dos tablas, una contiene el coste del hardware que se ha utilizado y otra el software y consumibles.

A continuación se muestra la tabla que contiene los costes del hardware necesario para este proyecto:

Elemento	Coste (€)
Ordenador Personal	950
Raspberry Pi	46,65
Micrófono	25,77
Tarjeta micro SD	11,99
<b>COSTE TOTAL HARDWARE</b>	<b>1034,41</b>

Tabla 1.2 Coste del Hardware.

Los recursos software necesarios, se ha de destacar que el IDE utilizado, Jupyter notebook, base para el desarrollo del proyecto, está disponible bajo una licencia BSD modificada, por lo que cualquier usuario puede usarlo gratuitamente. Por tanto, el único coste relativo al software es el del sistema operativo usado en nuestro ordenador personal, en este caso Windows 10 home. A continuación se incluye la tabla en la que aparece también el coste de los consumibles

Elemento	Coste (€)
Microsoft Windows 10 Home	145
Electricidad Consumida	80
Conexión a internet ADSL (32€/mes x 7 meses)	224
Material de oficina: papel, bolígrafos, etc.	5
<b>COSTE TOTAL SOFTWARE Y CONSUMIBLES</b>	<b>454</b>

Tabla 1.3 Coste del software y los consumibles.

#### Resumen del presupuesto:

Todo el desglose anterior lo resumimos en la siguiente tabla:

Coste	Coste (€)
Recursos Humanos	5400
Hardware	1034,41
Software y consumibles	454
<b>COSTE TOTAL PROYECTO</b>	<b>6888,41</b>

Tabla 1.4 Coste total del proyecto.

Por tanto, el presupuesto estimado para la ejecución material del proyecto, asciende a la cantidad de 6888,41€.

## 2. Estado del arte

Reconocer eventos en entornos domésticos puede desarrollarse con diferentes técnicas. Para los humanos, algunos de estos métodos de reconocimiento son naturales y espontáneos; por ejemplo, si cierra los ojos y escucha su entorno, probablemente reconocerá la voz de otras personas en la misma habitación, y podrá inferir debido al sonido que produce abrir un grifo, si alguien está lavando los platos, o podría inferir que están limpiando la casa por el sonido de un aspirador. Toda esta información, puede ser utilizada por los sistemas de automatización del hogar para ayudar a los inquilinos del hogar a abstenerse de realizar algunas tareas que con esta información este sistema podría hacer, como por ejemplo apagar luces, cerrar grifos, ayudar a la seguridad del hogar ... [50] En nuestra propuesta, investigamos las posibilidades de usar sensores de audio ambientales conectados a una placa inteligente (RaspBerry) que a través del sonido puede reconocer el contexto de un entorno doméstico o de oficina. Necesitamos saber cómo realizar el reconocimiento de la actividad humana con este sensor, por tanto, esto se hará grabando las señales de audio del entorno, realizando una extracción de características y clasificando a través de un modelo estas características para saber de qué evento doméstico se trata.

Para poder aplicar estas técnicas, es necesario colocar en las habitaciones de la casa este sistema de detección y reconocimiento de sonidos ambientales con lo que introduciremos el tema de internet de las cosas.

Para poder extraer las características de estos sonidos capturados, introducimos también las **redes neuronales convolucionales**.

Por último, también hablaremos del proceso de clasificación de sonido ambiental para que pueda utilizarse dentro de nuestro sistema de predicción.

### 2.1. Internet de las cosas

La próxima ola en la era de la informática estará fuera del ámbito del escritorio tradicional. Con el paradigma de Internet de las cosas (IoT), muchos de los objetos que nos rodean estarán en la red de una forma u otra. La identificación por

radiofrecuencia y las tecnologías de redes de sensores se elevarán para hacer frente a este desafío, en el que los sistemas de información y comunicación están integrados de forma invisible en el entorno que nos rodea. Esto da como resultado la generación de enormes cantidades de datos que deben almacenarse, procesarse y presentarse de forma transparente, eficiente y fácilmente interpretable.

Para que la tecnología desaparezca de la conciencia del usuario, Internet de las cosas exige:

1. Una comprensión compartida de la situación de sus usuarios y sus dispositivos.
2. Arquitecturas de software y redes de comunicación generalizadas para procesar y transmitir la información contextual a donde es relevante.
3. Las herramientas de análisis en el Internet de las cosas que apuntan a un comportamiento autónomo e inteligente.

Con estos tres fundamentos en su lugar, se puede lograr la conectividad inteligente y la computación consciente del contexto. [51].

Para citar algunas de las aplicaciones de IoT se mostrarán algunos ejemplos de su uso:

1. **Alerta:** El sistema de una casa inteligente es capaz de detectar su entorno y, en consecuencia, enviar alertas al usuario en su dispositivo. La alerta consta de información relacionada con datos ambientales. Esta información puede incluir el nivel de diferentes gases en el ambiente, la temperatura, la humedad, la intensidad de la luz, etc. Se puede enviar una alerta al usuario de forma regular en un momento predefinido. La alerta puede enviarse por correo electrónico, mensaje de texto, tweets o cualquier otra red social [52].
2. **Monitor:** Esta es la función más importante de una casa inteligente. Una casa inteligente es capaz de monitorear su entorno con la ayuda de varios sensores y señales de cámara. El monitoreo es una función importante, ya que realiza un seguimiento de cada actividad en una casa inteligente, que es la necesidad principal sobre la base de la cual se pueden tomar otras acciones o tomar decisiones. Por ejemplo, monitorear la temperatura de la

habitación y enviar una alerta al usuario para que encienda el aire acondicionado si la temperatura está por encima del umbral [52].

3. **Control:** Esta función de una casa inteligente permite al usuario controlar diferentes actividades. Las actividades pueden incluir encender / apagar luces, aire acondicionado y electrodomésticos, bloquear / desbloquear puertas, abrir / cerrar ventanas y puertas y muchas más. El usuario puede controlar las cosas desde el mismo lugar o desde una ubicación remota. Esta función incluso permite al usuario automatizar la actividad, como encender / apagar automáticamente el aire acondicionado cuando la temperatura ambiente es alta / baja [52].
4. **Detección de intrusiones:** La detección de intrusiones se utiliza para alertar al usuario a través de correo electrónico y mensaje de texto. La aplicación de detección de intrusos también puede enviar informes detallados con imágenes o clips de audio / video al usuario. El objetivo principal de esta aplicación es monitorear la actividad sospechosa en el hogar inteligente y alertar al usuario y tomar las acciones necesarias con fines de seguridad [52].
5. **E-Health:** Se utiliza para crear, gestionar y mantener datos relevantes para el paciente o una organización sanitaria. Los datos pueden ser para la planificación del tratamiento y la organización de los resultados de los pacientes.
6. **Monitoreo remoto de pacientes (RPM):** Las RPM ya se usan comúnmente para monitorear la presión arterial, la frecuencia del pulso, el nivel de oxígeno y la frecuencia cardíaca de los pacientes. Los datos se utilizan para la detección temprana de enfermedades o problemas de salud o para rastrear dolencias en curso.
7. **Gestión de calidad del aire:** Las ciudades inteligentes también están implementando herramientas que pueden capturar datos de contaminación en tiempo real y pronosticar emisiones. Ser capaz de predecir la contaminación del aire con precisión permite a las ciudades llegar a la raíz de sus problemas de emisiones y pensar en formas estratégicas para limitar la cantidad de contaminación del aire que emiten.
8. **Gestión inteligente de residuos:** Las soluciones de gestión de residuos ayudan a optimizar la eficiencia de la recogida de residuos y a reducir los

costes operativos, al tiempo que abordan mejor todos y cada uno de los problemas medioambientales asociados con la recogida de residuos ineficiente. En estas soluciones, el contenedor de residuos recibe un sensor de nivel; cuando se alcanza un cierto umbral, la plataforma de gestión de un conductor de camión recibe una notificación a través de su teléfono inteligente. El mensaje le ayuda a recoger eficazmente los residuos que hay por la ciudad.

## **2.2. Redes neuronales convolucionales**

Las redes neuronales convolucionales (CNN) han obtenido rendimientos muy notables en problemas de difícil resolución. Las CNNs se han convertido en una de las redes neuronales más representativas en el campo del aprendizaje profundo. La visión por computador basada en redes neuronales convolucionales ha permitido a las personas lograr lo que se había considerado imposible en los últimos siglos, como el reconocimiento facial, los vehículos autónomos, el supermercado de autoservicio y el tratamiento médico inteligente [53].

La red neuronal convolucional es un tipo de red neuronal de retroalimentación que puede extraer características de los datos con estructuras de convolución. A diferencia de los métodos tradicionales de extracción de características CNN no necesita extraer características manualmente. La arquitectura de CNN está inspirada en la percepción visual. Una neurona biológica corresponde a una neurona artificial; Los núcleos de la CNN representan diferentes receptores que pueden responder a diversas características; Las funciones de activación simulan la función de que solo las señales eléctricas neuronales que superan un cierto umbral pueden transmitirse a la siguiente neurona. Las funciones de pérdida y los optimizadores son algo que la gente inventó para enseñarle a todo el sistema CNN a aprender lo que esperábamos. En comparación con las redes neuronales artificiales generales, CNN posee muchas ventajas:

1. Conexiones locales. Cada neurona ya no está conectada a todas las neuronas de la capa anterior, sino solo a un pequeño número de neuronas, lo que es eficaz para reducir los parámetros y acelerar la convergencia.

2. **Peso compartido.** Un grupo de conexiones puede compartir los mismos pesos, lo que reduce aún más los parámetros.
3. **Reducción de dimensionalidad por muestreo descendente.** Una capa de agrupación aprovecha el principio de correlación local de imágenes para muestrear una imagen, lo que puede reducir la cantidad de datos al tiempo que retiene información útil. También puede reducir la cantidad de parámetros al eliminar características triviales.

Estas tres atractivas características hacen que CNN se convierta en uno de los algoritmos más representativos en el campo del aprendizaje profundo.

Algunas de las aplicaciones que tienen estas redes neuronales son las siguientes:

1. **Clasificación de imágenes:** La clasificación de imágenes es la tarea de asignar a una imagen en una categoría. CNN representa un gran avance en este campo. LeNet-5 se considera la primera aplicación utilizada en la clasificación de dígitos escritos a mano.
2. **Detección de objetos:** La detección de objetos es una tarea basada en la clasificación de imágenes. Los sistemas no solo necesitan identificar a qué categoría pertenece la imagen de entrada, sino que también deben marcarla con un cuadro delimitador.
3. **Identificación de señales multidimensionales:** La identificación de la señal se basa en discriminar la señal de entrada de acuerdo con la característica que CNN aprendió de los datos de entrenamiento. Zhang et al. [54] propuso una estructura de red neuronal convolucional unidimensional de resolución múltiple para identificar arritmias y otras enfermedades.

### 2.3. Clasificación de sonidos ambientales

Las mejoras en el campo de la clasificación de imágenes en los últimos años están llevando a los investigadores a empezar a utilizar imágenes a la hora de clasificar sonidos. La diferencia importante entre el habla / música y el sonido ambiental es que los primeros están fuertemente estructurados y claramente delimitados, mientras que los segundos no tienen una estructura común. Esto hace que sea un problema completamente nuevo. Es posible que exista una solución

elegante en el aprendizaje profundo, ya que se ha demostrado que las redes neuronales profundas pueden manejar grandes cantidades de datos y modelar características complejas debido a los avances en la potencia informática, incluido el uso más general de las GPU [55].

El enfoque basado en el aprendizaje profundo más común para la clasificación de sonidos es convertir el archivo de audio en una imagen y luego usar una red neuronal para procesar la imagen. Mostafa et al [56] realiza la clasificación de la música utilizando redes neuronales probabilísticas con resultados satisfactorios. La mayoría de los enfoques de clasificación sólidos utilizan el reconocimiento de patrones supervisado. Sin embargo, Zhang y Schuller [57] expresan el problema de que el etiquetado manual de conjuntos de datos es muy costoso y recomiendan el aprendizaje semi-supervisado como una mejor solución. McLoughlin et al [58] afirma que la clasificación del sonido en entornos ruidosos realistas es un desafío y proponen una red neuronal profunda como una solución viable. Piczak y Zhang et al [59] transmiten la idea de que las redes neuronales convolucionales tienen las mejores tasas de precisión en el análisis de espectrograma.

Según lo resumido por Chachada et al [60], hay tres formas amplias de procesar sonidos ambientales con fines de clasificación:

1. Framing-based donde las señales de audio se separan en frames usando una ventana de Hamming. Luego, las características se extraen de cada frame y se clasifican por separado.
2. Procesamiento basado en sub-framing donde los frames se subdividen aún más y cada frame se clasifica en base a la votación mayoritaria de los sub-frames.
3. Procesamiento secuencial donde las señales de audio se dividen en segmentos de típicamente 30 ms con un 50% de superposición. El clasificador luego clasifica las características extraídas de estos segmentos.



### 3. Proceso ingeniería software

Una de las definiciones que podemos encontrar de ingeniería del software es una colección sistemática de buenas prácticas y técnicas de desarrollo de programas [61].

En el proceso de ingeniería de software de este trabajo fin de grado (Recolección y reconocimiento de eventos domésticos mediante sensores de sonido en dispositivos IoT) contaremos con los siguientes puntos a desarrollar:

- Búsqueda de información sobre integración de sensores IoT de sonido.
- Búsqueda de información sobre clasificación de sonido
- Implementación y ejecución con Python de la recolección y etiquetado de audios.
  - Aplicación desarrollada en Python que nos ayudará a recopilar sonido ambiente en los interiores de una casa.
  - Aplicación desarrollada en Python que sirve para dividir los sonidos recogidos en clips de audio de 3 segundos.
  - Aplicación desarrollada en Python facilita el nombramiento de los clips generados para su posterior etiquetación en ficheros CSV.
  - Aplicación desarrollada en Python que con una serie de parámetros genera los ficheros CSV con el nombre de los clips de audio y su correspondiente etiqueta.
- Diseño y programación de algoritmos de reconocimiento de sonidos (clasificación).
- Evaluación de las métricas de accuracy, f1-score, precisión y recall de la clasificación.
- Documentar el proyecto y redacción de la memoria

#### 3.1. Fases de la ingeniería del software

Las fases con las que nos encontramos en el proceso de ingeniería del software son las siguientes:

- **Especificación de Requerimientos:** Descripción completa del comportamiento del sistema que se va a desarrollar.
- **Análisis del sistema:** Especificación de los objetivos y limitaciones de nuestro sistema.
- **Diseño del sistema:** Definición de arquitectura, módulos, interfaces y datos para alcanzar los requisitos especificados.
- **Implementación del sistema:** Desarrollo del software anteriormente especificado que cumpla todos los requisitos.
- **Pruebas:** Verificar y validar que el sistema cumpla con nuestras perspectivas.

### 3.2. Especificación de requisitos

La definición de requisitos es la tarea de recopilar toda la información relevante que se utilizará para comprender una situación de un problema antes del desarrollo del sistema. La documentación de esta información se denomina especificación de requisitos. La forma y el contenido de la especificación de requisitos pueden tener un impacto tremendo en la tarea del software a lo largo de su vida útil [62].

#### 3.2.1. Requerimientos del sistema de recopilación de sonido ambiente.

##### 3.2.1.1. Funcionales

- **Requerimiento funcional 1: Grabar audios ambientales mediante dispositivo de sonido:**  
El programa obtiene por parámetros de entrada el nombre del fichero, índice inicial e índice final, de modo que grabará audios de 3 segundos tantos como la diferencia entre el índice final menos el índice inicial. Estos audios se irán almacenando en nuestra memoria ROM con el nombre obtenido por el parámetro de entrada y el índice correspondiente.

##### 3.2.1.2. No funcionales

- **Requerimientos físicos (RaspBerry):**
  - Procesador: 1.2 GHz

- RAM: 1GB
- HDD: 120GB
- Conexión a la red

### 3.2.2. Requerimientos del sistema de segmentación de audios.

#### 3.2.2.1. Funcionales

##### **Requerimiento funcional 1: Segmentar audios:**

El programa ofrecerá la opción al usuario de guardar con el nombre del fichero que desee y el número de audios en los que quiere dividir una escena anteriormente grabada. Para ello se harán las correspondientes divisiones en audios de tres segundos y se guardarán en la ROM de nuestro computador. Ej: Escena1.wav de 300 segundos pasaría a ser part1.wav, part2.wav, ..., part100.wav).

### 3.2.3. Requerimientos del sistema etiquetado de clips de audios.

#### 3.2.3.1. Funcionales

- **Requerimiento funcional 1: Generador CSV:**  
El sistema deberá etiquetar cada audio con su correspondiente clase, para ello se dispone de un diccionario con todas las posibles clases donde se indica cuántos audios de esa clase existen. El sistema generará un archivo con formato CSV tal que su cabecera contendrá (name, target, category), lo que servirá para su posterior procesamiento a la hora de entrenar nuestro modelo.
- **Requerimiento funcional 2: Generador CSV de escenas:**  
El sistema deberá etiquetar las escenas grabadas para su posterior procesamiento para comprobar las predicciones de nuestro modelo y entrenamiento. Generará un archivo CSV con la siguiente cabecera (name, target, category) que a diferencia del generador CSV aquí se respeta el orden de la secuencia en la que fue grabado cada audio.

#### 3.2.3.2. No funcionales

- **Requerimientos de software:**  
Para poder utilizar este programa se necesita:

- Python 3.7.3

### 3.2.4. Requerimientos del sistema algoritmos para el reconocimiento de sonidos.

#### 3.2.4.1. Funcionales

- **Requerimiento funcional 1: Cargar etiquetas y sonidos:**  
El sistema deberá cargar en memoria principal los sonidos dado un directorio y el fichero CSV que contiene el etiquetado correspondiente a cada sonido anteriormente cargado.
- **Requerimiento funcional 2: Conversión sonido a imagen:**  
El sistema deberá disponer de una conversión de sonido a espectrograma y de este último a imagen ya que nuestra red neuronal es convolucional.
- **Requerimiento funcional 3: Cargar imágenes y etiquetas:**  
El sistema deberá cargar en memoria estas imágenes generadas por los sonidos anteriormente cargados y su correspondiente clasificación para su posterior uso.
- **Requerimiento funcional 4: Generación del modelo:**  
El sistema deberá hacer uso de las imágenes y sus correspondientes clasificaciones previamente procesadas para entrenarse y generar un modelo que se adapte lo mejor posible a la realidad.

#### 3.2.4.2. No funcionales

- **Requerimientos de software:**  
Para poder utilizar este programa se necesita:
  - Python 3.7.3

### 3.2.5. Requerimientos del sistema de evaluación offline del modelo.

#### 3.2.5.1. Funcionales

- **Requerimiento funcional 1: Cross validation:** El sistema deberá dividir los datos de entrenamiento y de evaluación, de modo que la primera vez tendrá un 90% de los datos para entrenar y un 10% para predecir, posteriormente 80% para entrenar y 20% para predecir, y así sucesivamente dependiendo del número de divisiones que le

indiquemos (para el caso indicado el nº de divisiones es 10). Así, se evalúa el modelo en cada iteración y recopila los datos para saber si está prediciendo con una tasa de precisión elevada.

#### **3.2.5.2. No funcionales**

- Para poder utilizar este programa se necesita:
  - Python 3.7.3

### **3.2.6. Requerimientos del sistema de evaluación online del modelo.**

#### **3.2.6.1. Funcionales**

- **Requerimiento funcional 1: Validación del modelo:**  
El sistema deberá evaluar el modelo generado por nuestra red neuronal convolucional con unos datos de entrada que predecirá mientras se graban en tiempo real.

#### **3.2.6.2. No funcionales**

- Para poder utilizar este programa se necesita:
  - Python 3.7.3

### **3.3. Análisis del sistema**

El análisis del sistema es el proceso que descompone un sistema en sus partes componentes con el fin de definir qué tan bien interactúan estos componentes para cumplir con los requisitos establecidos.

Es conveniente que añadas un caso de uso general que contenga todos los que has considerado, además de un pequeño resumen de todos los módulos en conjunto

- **Caso de uso 1: Recopilación de sonido ambiente**

En la siguiente ilustración se muestra el caso de uso para recopilar

fuentes de sonido ambiente:

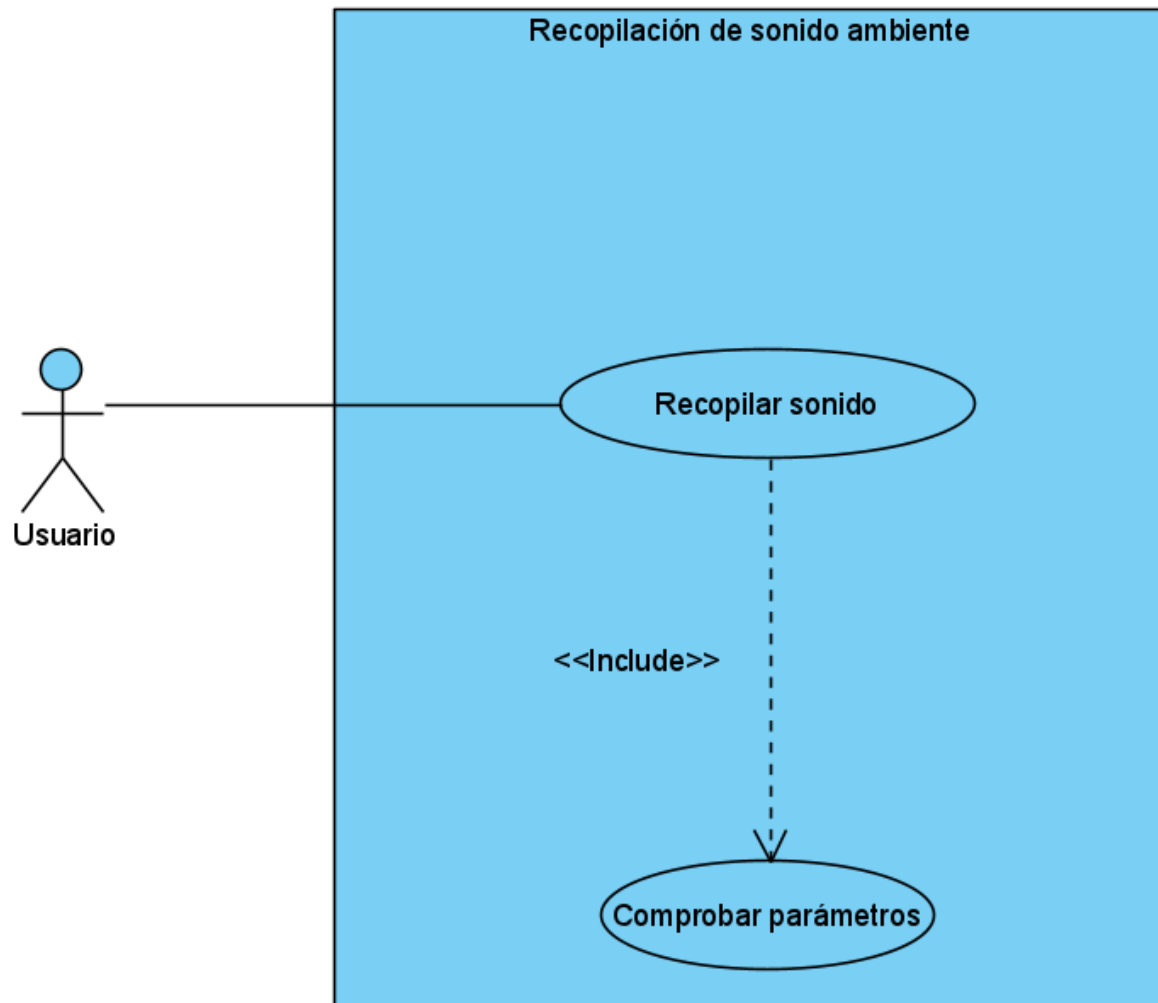


Ilustración 3.1 Caso de uso 1: Recopilación de sonido ambiente

- **Actores:** Usuario
- **Condiciones de entrada:** Debe proporcionarse los parámetros de entrada:
  - Nombre del fichero.
  - Índice inicial.
  - Índice final.
- **Eventos:**
  - El programa inicia con unos parámetros de entrada.
  - El programa empezará a mostrar información sobre los audios que se están generando.

- Los audios generados serán guardados con una duración de 3 segundos con el nombre indicado más un índice que los hace únicos.
- **Excepciones:** Si no se introducen bien los tres parámetros necesarios el programa indicará por consola que ha ocurrido un error.
- **Caso de uso 2: Segmentación de audios.**

En la siguiente ilustración se muestra el caso de uso para segmentar los audios:

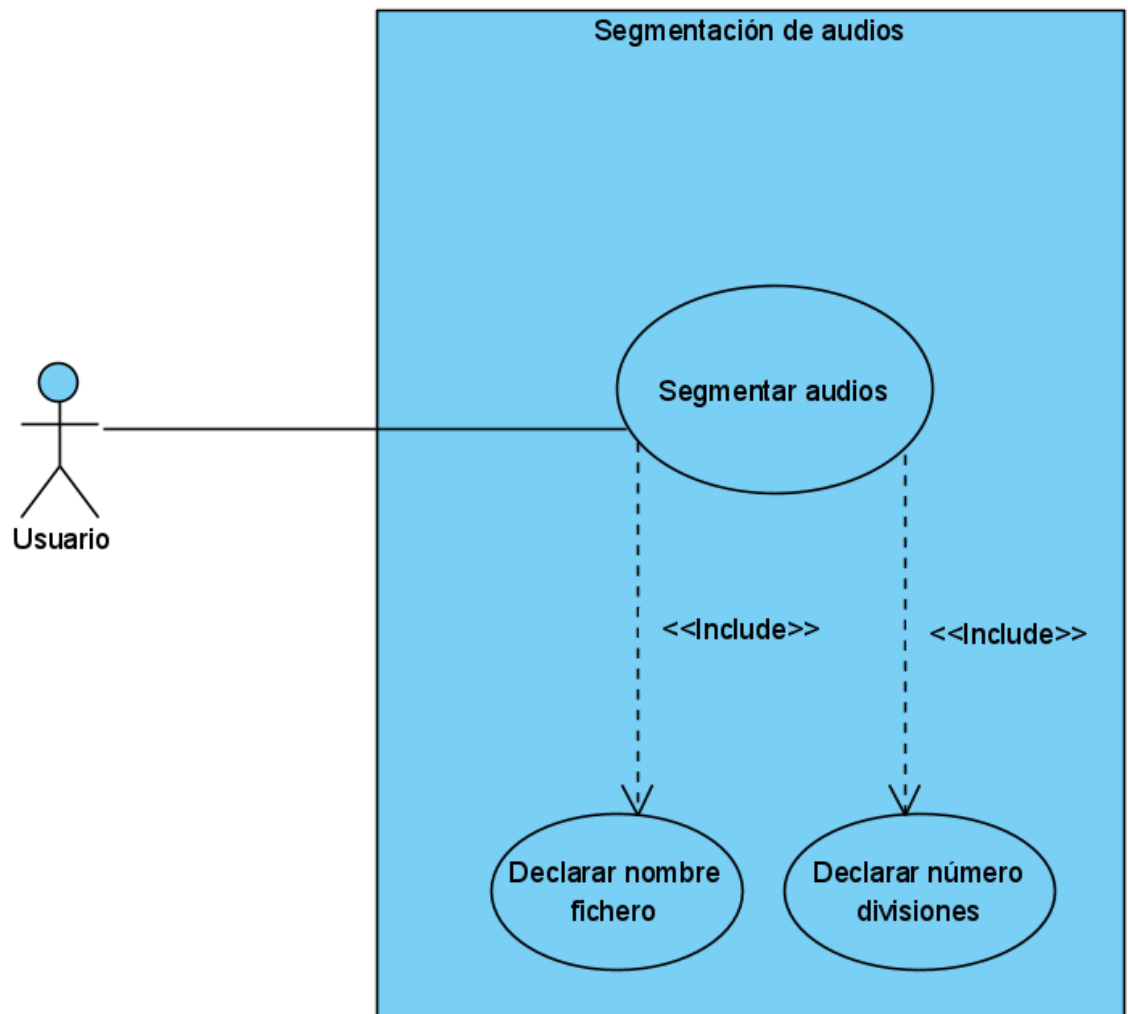


Ilustración 3.2 Caso de uso 2: Segmentación de audios

- **Actores:** Usuario
- **Condiciones de entrada:** Debe proporcionarse los parámetros de entrada:
  - Nombre fichero
  - Número de divisiones

- **Eventos:**
  - El programa inicia con unos parámetros de entrada.
  - El programa empezará a dividir el audio indicado en el número de clips indicados con una duración de 3 segundos.
  - Los clips generados serán guardados con el nombre part1.wav, part2.wav..., part(n).wav hasta llegar al número de divisiones indicadas.
- **Excepciones:** Si no se introducen bien los parámetros necesarios el programa indicará por consola que ha ocurrido un error y no ejecutará nada.
- **Caso de uso 3: Etiquetado de clips de audio**

En la siguiente ilustración se muestra el caso de uso para etiquetar el audio en un fichero CSV:

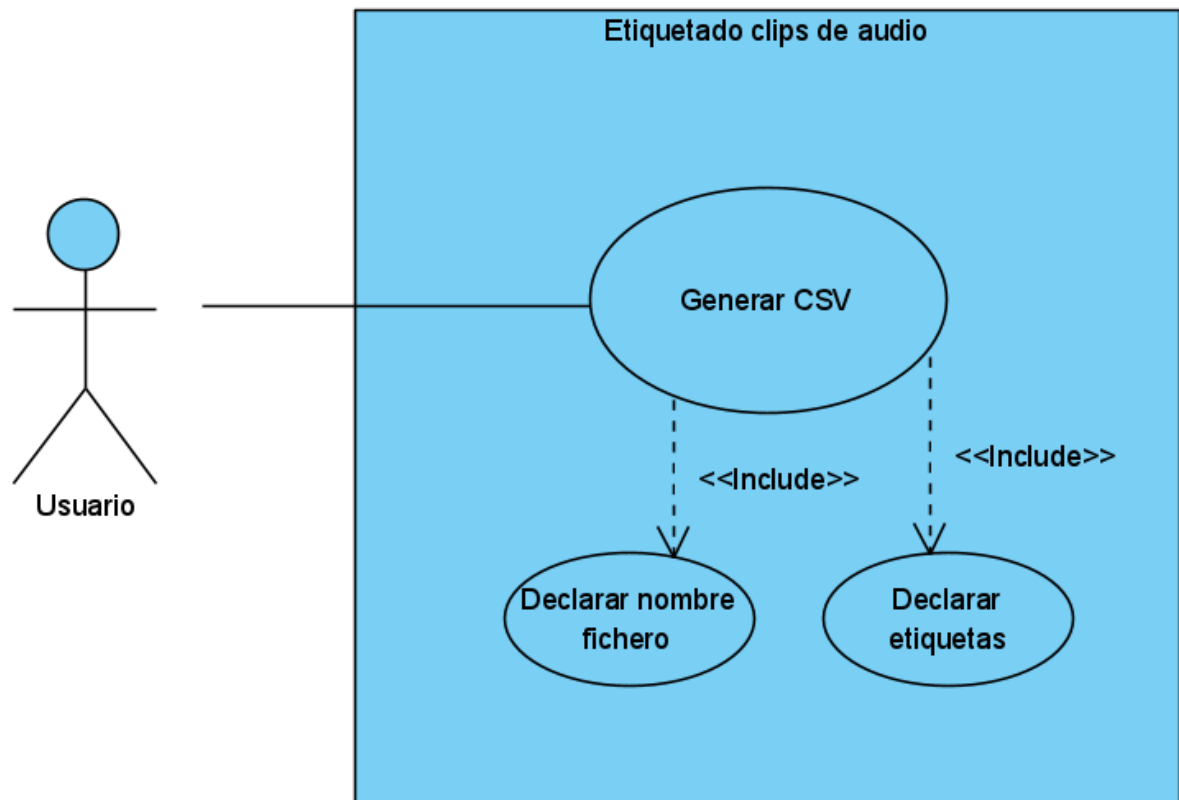


Ilustración 3.3 Caso de uso 3: Etiquetado de clips de audio

- **Actores:** Usuario
- **Condiciones de entrada:**
  - Se debe indicar cuantos audios de cada clase hay.



- Se debe indicar el nombre del fichero que se va a crear.
- **Eventos:**
  - El programa recorrerá una estructura que almacena la información de cuantos elementos de cada clase existen (número de audios de cada etiqueta).
  - Escribirá un fichero con formato CSV donde la cabecera será (name, target, category).
  - Se guardará en memoria ROM con el nombre indicado y la ruta activa a la ejecución del programa.
- **Excepciones:** Si no se introducen bien los parámetros necesarios el programa indicará por consola que ha ocurrido un error y no ejecutará nada.
- **Caso de uso 4: Reconocimiento de sonidos**

En la siguiente ilustración se muestra el caso de uso de reconocimiento de sonido:

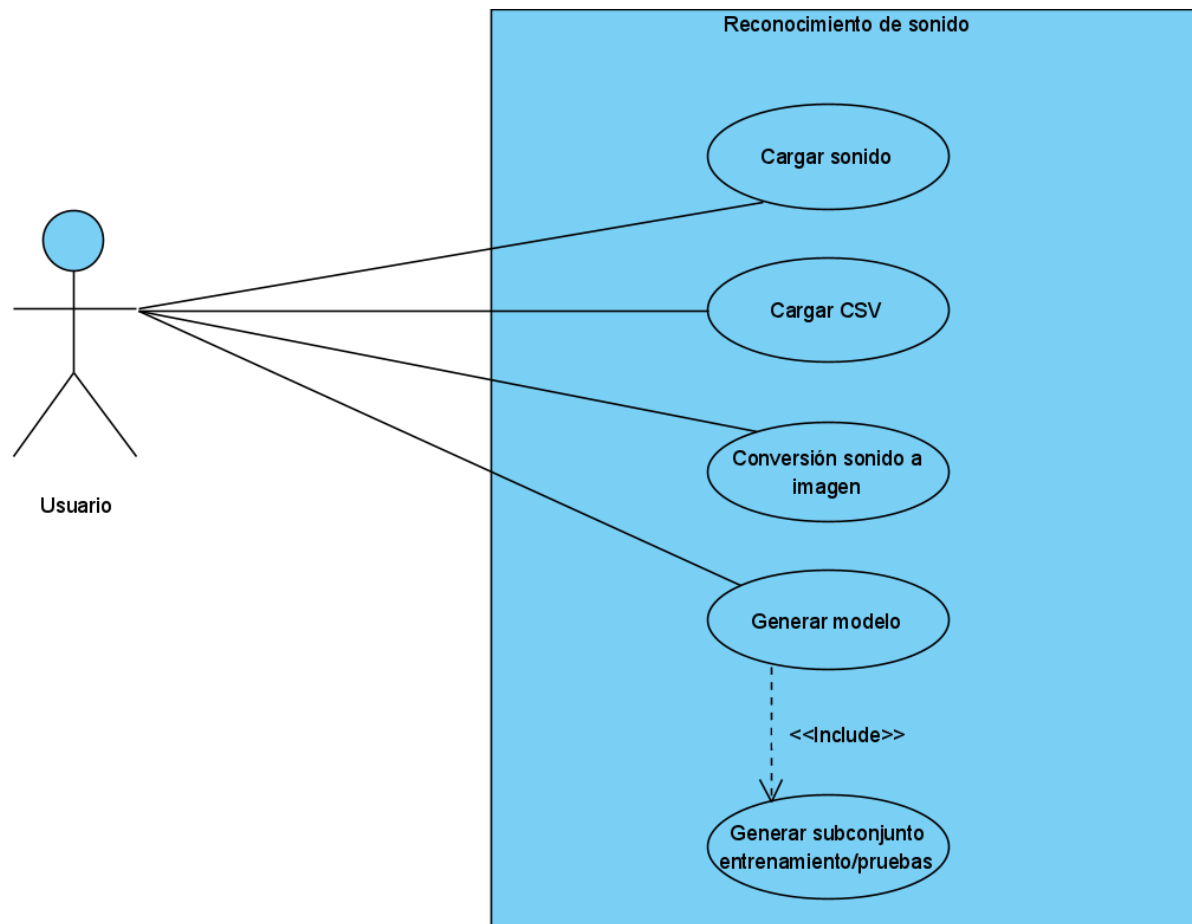


Ilustración 3.4 Caso de uso 4: Reconocimiento de sonidos

- **Actores:** Usuario
- **Condiciones de entrada:**
  - Se debe indicar la ruta de la carpeta que contienen los sonidos
  - Se debe indicar la ruta del fichero CSV
- **Eventos:**
  - El sistema cargará los audios de la ruta indicada
  - El sistema cargará los datos de los audios con sus correspondientes etiquetas a través de un fichero CSV.
  - El sistema convertirá el sonido a espectrograma y este a una imagen, generando un nuevo dataSet de imágenes.
  - El sistema generará un modelo a través de un entrenamiento con un subconjunto de imágenes y su correspondiente etiqueta. Luego con un subconjunto para pruebas no utilizado en el entrenamiento se valorará la tasa de acierto de este.
- **Excepciones:** El sistema no funcionará correctamente si no se introduce las rutas de la carpeta que contiene los sonidos y el fichero CSV que tiene la información de estos.
- **Caso de uso 5: Evaluación del modelo**

En la siguiente ilustración se muestra el caso de uso para evaluar un

modelo:

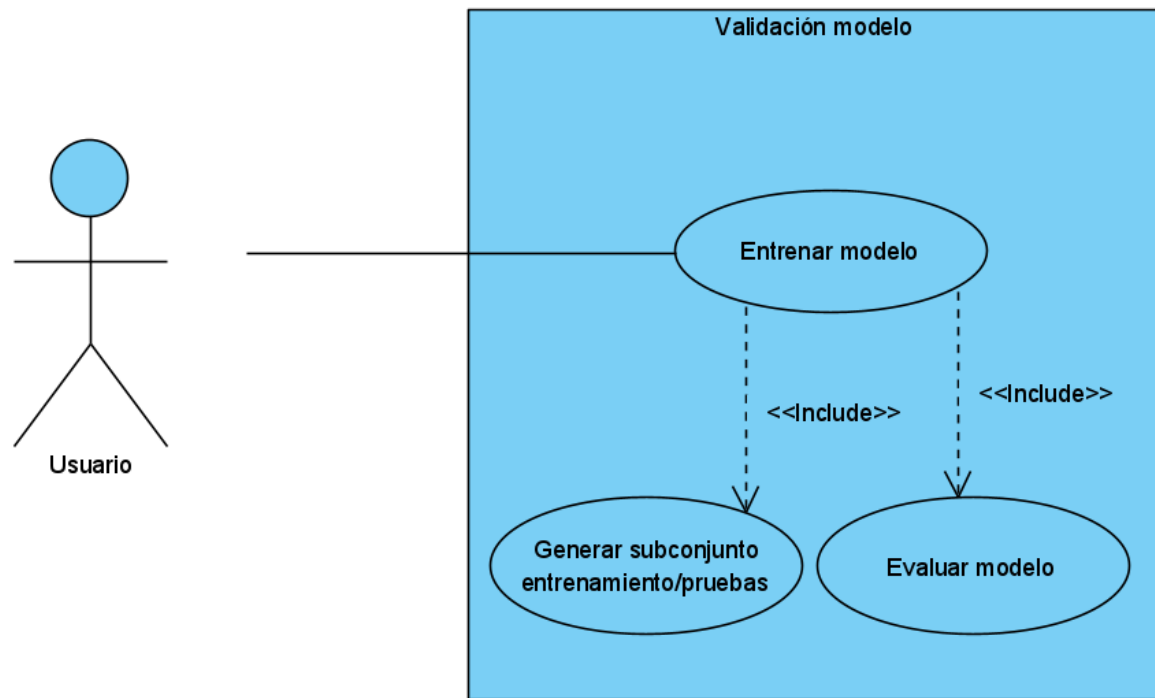


Ilustración 3.5 Caso de uso 5: Evaluación del modelo

- **Actores:** Usuario
- **Condiciones de entrada:**
  - Datos de entrenamiento.
  - Datos de predicción.
- **Eventos:**
  - El sistema evaluará el modelo haciendo uso de un dataSet de imágenes con sus correspondientes etiquetas.
- **Excepciones:** En caso de no ingresar bien el subconjunto de entrenamiento/pruebas el sistema no funcionará.
- **Caso de uso 6: Evaluación del modelo en tiempo real**

En la siguiente ilustración se muestra el caso de uso para evaluar un modelo en tiempo real:

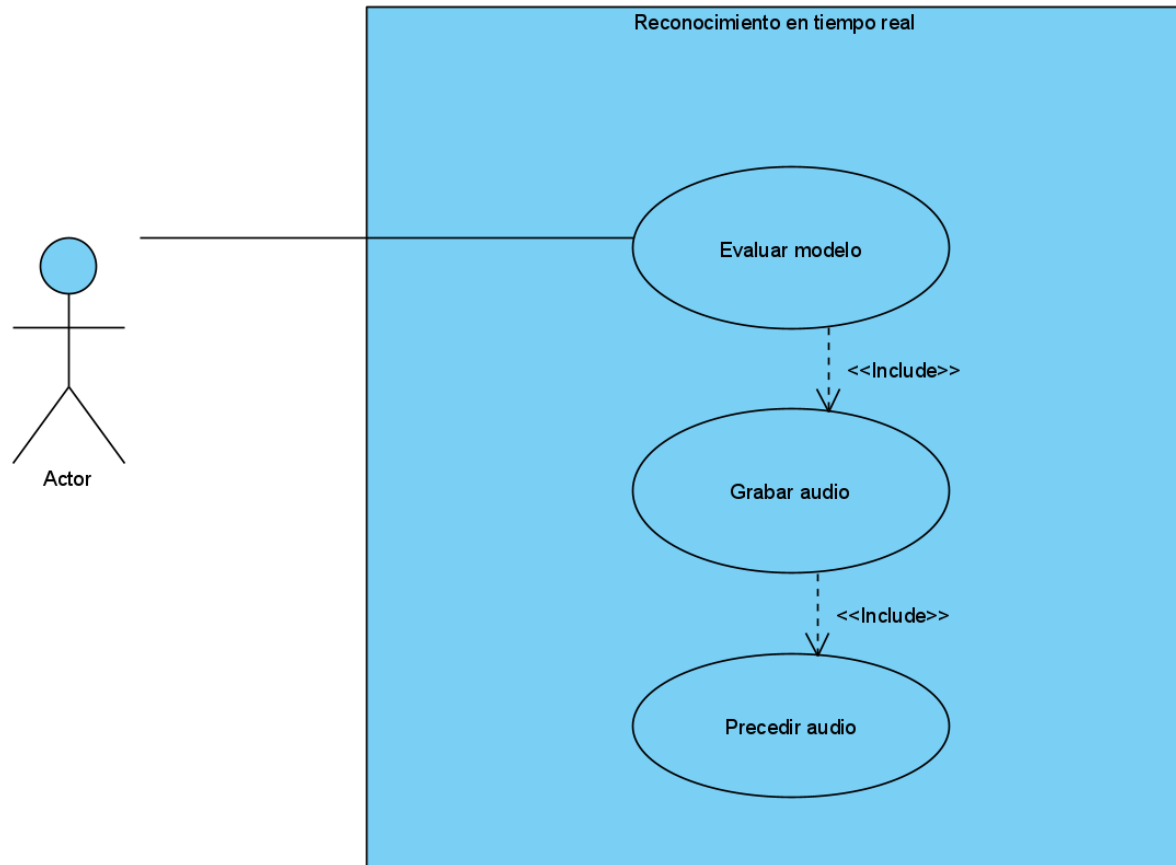


Ilustración 3.6 Caso de uso 6 Evaluación del modelo en tiempo real

- **Actores:** Usuario
- **Condiciones de entrada:**
  - Datos de entrenamiento.
  - Datos de predicción.
- **Eventos:**
  - El sistema evaluará el modelo haciendo uso de los sonidos que graba durante 3 segundos y lo evalúa en tiempo real.
- **Excepciones:** En caso de no ingresar bien el subconjunto de entrenamiento/pruebas el sistema no funcionará

### 3.4. Diseño del sistema

El diseño del sistema es el proceso de diseñar los elementos de un sistema, como la arquitectura, los módulos y componentes, las diferentes interfaces de esos componentes y los datos que pasan por ese sistema.

#### 3.4.1. Análisis de señales de audio

La extracción de características de audio es una de las piedras angulares de la investigación y el desarrollo actuales del procesamiento de señales de audio. Las funciones de audio son información contextual que se puede extraer de una señal de audio. Se pueden aplicar a una variedad de campos de investigación que incluyen [63]:

- Extracción de funciones vinculadas a efectos de audio.
- Síntesis estadística.
- Síntesis basada en funciones.
- Evaluación de técnicas de síntesis.
- Medidas de similitud.
- Clasificación de datos.
- Minería de datos.

En este TFG vamos a utilizarlas para clasificación de datos, ya que nos interesa extraer las características de estos audios para su posterior procesamiento.

Algunas herramientas para trabajar la extracción de características son las siguientes:

- **Aubio:** Una biblioteca de extracción de características de alto nivel que extrae características como detección de inicio, seguimiento de ritmo, tempo, melodía.
- **jAudio:** Aplicación independiente basada en Java con interfaz gráfica de usuario (GUI) y CLI. Diseñado para procesamiento por lotes en formato XML o ARFF para cargar en Weka.

- **Librosa:** API para extracción de características, para procesar datos en Python.
- **Meyda:** Herramienta de extracción de características de bajo nivel basada en Web Audio API, escrita en Javascript. Diseñado para un procesamiento eficiente en tiempo real basado en un navegador web.
- **YAAFE:** Biblioteca de extracción de características de bajo nivel diseñada para la eficiencia computacional y el procesamiento por lotes mediante la utilización de gráficos de flujo de datos, escritos en C ++ con una CLI y enlaces para Python y Matlab.

### 3.4.2. Etiquetado de audio

En este apartado se hablará sobre los tipos de archivos que podemos utilizar para etiquetar los audios que hemos recogido en el ambiente doméstico, de esta manera, podremos utilizarlos para cargarlos en el sistema y tener información sobre la clase de evento que representa cada uno de ellos. Estos datos que se van a almacenar nos resultan útiles para entrenar nuestra CNN y su posterior evaluación.

- **Comma Separated Values (CSV):** Son valores separados por comas que dan lugar a una tabla, donde la primera fila es la cabecera y el resto de filas representan los datos de estas. Nos ayuda a manejar una inmensa cantidad de datos sin que tenga un coste computacional alto.

En la siguiente ilustración se muestra un ejemplo de fichero CSV [37]:



Ilustración 3.7 Ejemplo fichero CSV

- **JavaScript Object Notation (JSON):** Es un formato de datos basado en los tipos de datos del lenguaje de programación JavaScript. En los últimos años, JSON ha ganado una enorme popularidad entre los desarrolladores web y se ha convertido en el formato principal para intercambiar información a través de la web. El esquema JSON puede especificar cualquiera de los seis tipos de documentos JSON válidos: objetos, matrices, cadenas, números, valores booleanos y nulos; y para cada uno de estos tipos hay varias palabras clave que ayudan a dar forma y restringir el conjunto de documentos que especifica un esquema [64].

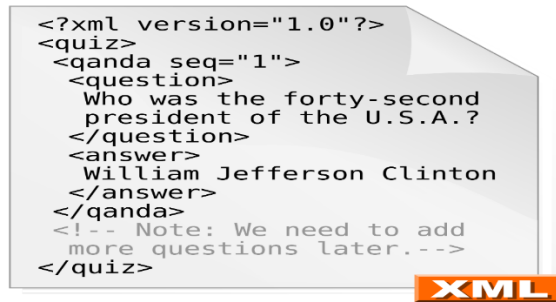
En la siguiente ilustración se muestra un ejemplo de fichero JSON [38]:

```
{
  "orders": [
    {
      "orderno": "748745375",
      "date": "June 30, 2088 1:54:23 AM",
      "trackingno": "TN0039291",
      "custid": "11045",
      "customers": [
        {
          "custid": "11045",
          "fname": "Sue",
          "lname": "Hatfield",
          "address": "1409 Silver Street",
          "city": "Ashland",
          "state": "NE",
          "zip": "68003"
        }
      ]
    }
  ]
}
```

Ilustración 3.8 Ejemplo fichero JSON

- **Extensible Markup Language (XML):** Es un lenguaje de marcas que nos permite almacenar datos de forma legible bajo un conjunto de reglas.

En la siguiente ilustración se muestra un ejemplo de fichero XML [39]:



```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

Ilustración 3.9 Ejemplo fichero XML

### 3.4.3. Herramientas para construir modelos de predicción

#### 3.4.3.1. TensorFlow:

Antes de saltar a la biblioteca de TensorFlow, vamos a familiarizarnos con la unidad básica de datos en TensorFlow. Un tensor es un objeto matemático y una generalización de escalares, vectores y matrices. Un tensor se puede representar como una matriz multidimensional. Un ejemplo de tensor es:  $[[1., 2., 7.], [3., 5., 4.]]$ : Este es un tensor de rango 2; es una matriz con forma  $[2, 3]$ .

Una imagen es un tensor de tercer orden donde las dimensiones pertenecen a la altura, el ancho y el número de canales (rojo, azul y verde) [65].

En la siguiente ilustración se muestra la arquitectura de una CNN [40]:



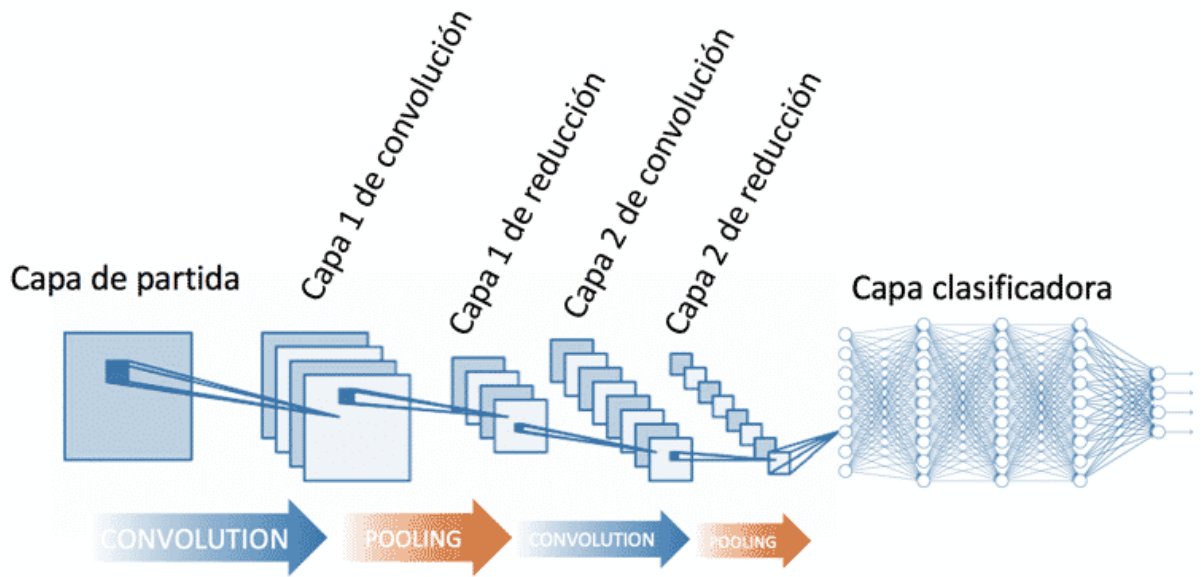


Ilustración 3.10 Arquitectura CNN

En la convolución se realizan operaciones de productos y sumas entre la capa de partida y los  $n$  filtros (neuronas) que genera un mapa de características. Las características extraídas corresponden a cada posible ubicación del filtro en la imagen original. Ilustración que muestra la capa de convolución en una CNN **[40]**:

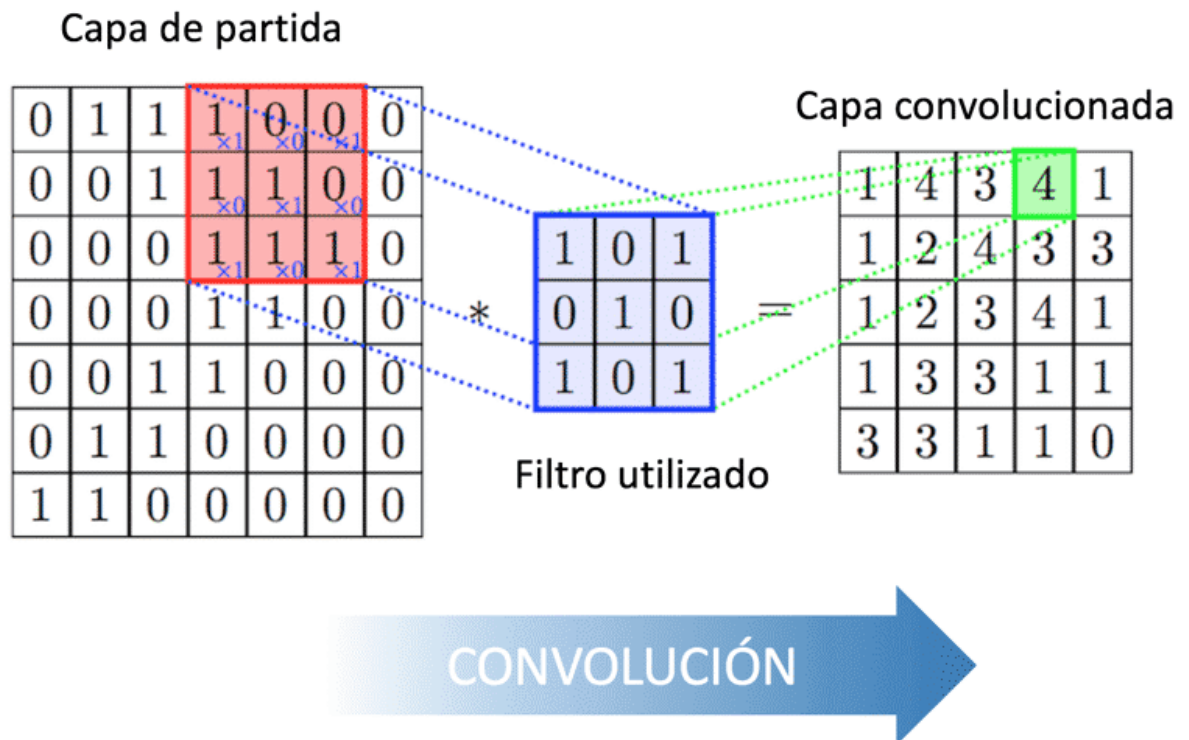


Ilustración 3.11 Capa convolución

Cuando se procesa una imagen, cada capa de convolución debe detectar algún patrón en la imagen y pasarla a la siguiente convolución, como los valores negativos no son importantes en el procesamiento de imágenes se establecen a 0. Los valores positivos son los que pasan a la siguiente capa, aquí es donde entra la función de activación relu.

$$f(x) = \max(0, x) \quad \{0 \text{ for } x < 0, x \text{ for } x \geq 0$$

(Fórmula 1)

En la siguiente ilustración se ejemplifica la función de activación relu [40]:

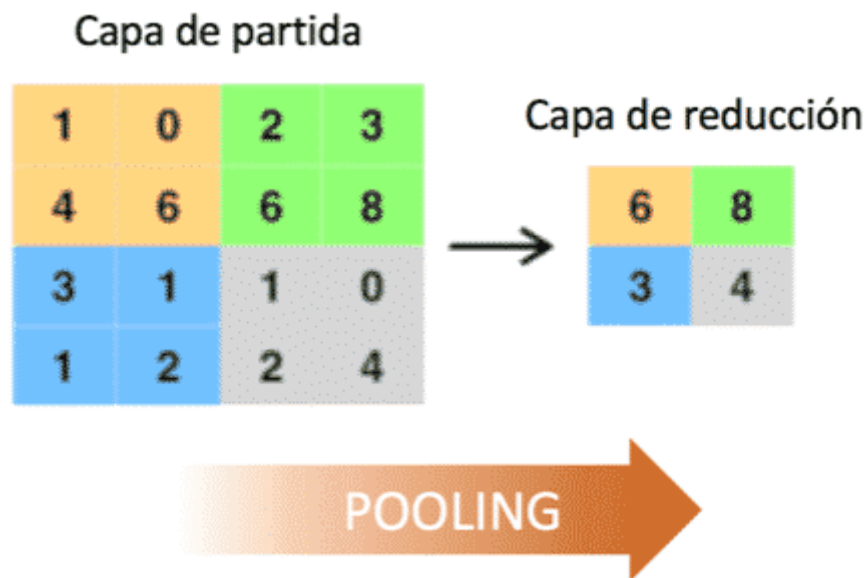


Ilustración 3.12 Función activación relu

De esta manera, garantizamos que las capas de convolución siempre pasen a la siguiente los valores positivos que son los que contienen los patrones.

#### 3.4.4. Evaluación de modelos

##### 3.4.4.1. Matriz de confusión (Matrix confusión)

La matriz de confusión es una de las métricas más intuitivas y sencillas que se utiliza para encontrar la precisión y exactitud del modelo. Se utiliza para el problema de clasificación donde la salida puede ser dos o más tipos de clases.

En la siguiente figura se muestra un esquema de cómo está organizada la matriz de confusión para dos clases, imaginemos que esta matriz contiene la información de si una imagen es un gato o por lo contrario no es un gato. En tal caso cada vez que analicemos una imagen pasar lo siguiente:

- La imagen es de un gato y el modelo predice que es un gato verdadero **positivo**.
- La imagen es de un gato y el modelo predice que **no** es un gato falso **negativo**.
- La imagen **no** es de un gato y el modelo predice que es un gato falso **positivo**.

- La imagen **no** es de un gato y el modelo predice que **no** es un gato verdadero **negativo**.

En la siguiente ilustración se describe una matriz de confusión:

### MATRIZ DE CONFUSIÓN

	Predicción		
		Positivo	Negativo
	Actual		
	Positivo	Verdaderos Positivos	Falsos Negativos
	Negativo	Falsos Positivos	Verdaderos Negativos

**Ilustración 3.13 Descripción matriz de confusión**

El escenario ideal en un modelo es que tenga 0 falsos positivos y 0 falsos negativos, pero esto no es lo que suele ocurrir.

Esta métrica nos servirá para determinar el rendimiento de nuestro modelo, sacando la información y las conclusiones de esta matriz. En la siguiente figura podemos observar de un vistazo el número de aciertos y errores que ha cometido el modelo, donde la diagonal principal será el número de aciertos de cada categoría, y los restantes, el número de fallos y las categorías con las que se suele confundir.

En la siguiente ilustración se muestra un ejemplo de una matriz de confusión [41]:

Matriz de confusión: Modelo SVM

actual	ira	37	0	1	2	5	0
	tristeza	0	59	3	0	0	4
	asco	0	3	47	4	4	0
	felicidad	2	0	3	45	3	0
	sorpresa	9	0	3	7	45	0
	miedo	0	1	1	1	0	48
		ira	tristeza	asco	felicidad	sorpresa	miedo
		predicción					

Ilustración 3.14 Ejemplo matriz confusión

#### 3.4.4.2. Validación cruzada (Cross validation)

Generalmente, un clasificador se induce a partir de datos de entrenamiento utilizando un algoritmo de aprendizaje de clasificador. Cada clasificador tiene un error de predicción asociado, también llamado error verdadero. Pero, por lo general, se desconoce el verdadero error, no se puede calcular y debe estimarse a partir de los datos. Este error se denomina error de predicción estimado. Un estimador del error de un clasificador es una variable aleatoria y su calidad suele medirse mediante su sesgo y varianza. Hay varios estimadores del error de clasificación, probablemente la más popular, es la validación cruzada [66].

Para realizar la validación cruzada se establece un valor para K y el conjunto de datos se divide en K particiones de igual tamaño. K- 1 se utilizan para entrenamiento, mientras que el restante se utiliza para pruebas. Este proceso se repite K veces, con una partición diferente utilizada para probar cada vez.

Por ejemplo, si nuestro conjunto de datos lo dividimos con un  $k = 5$  (donde en cada división se utiliza el 80% para entrenar y el 20% para pruebas):

En la siguiente ilustración se muestra un ejemplo de la división de datos para entrenar y validar un modelo usando la técnica Cross Validation:

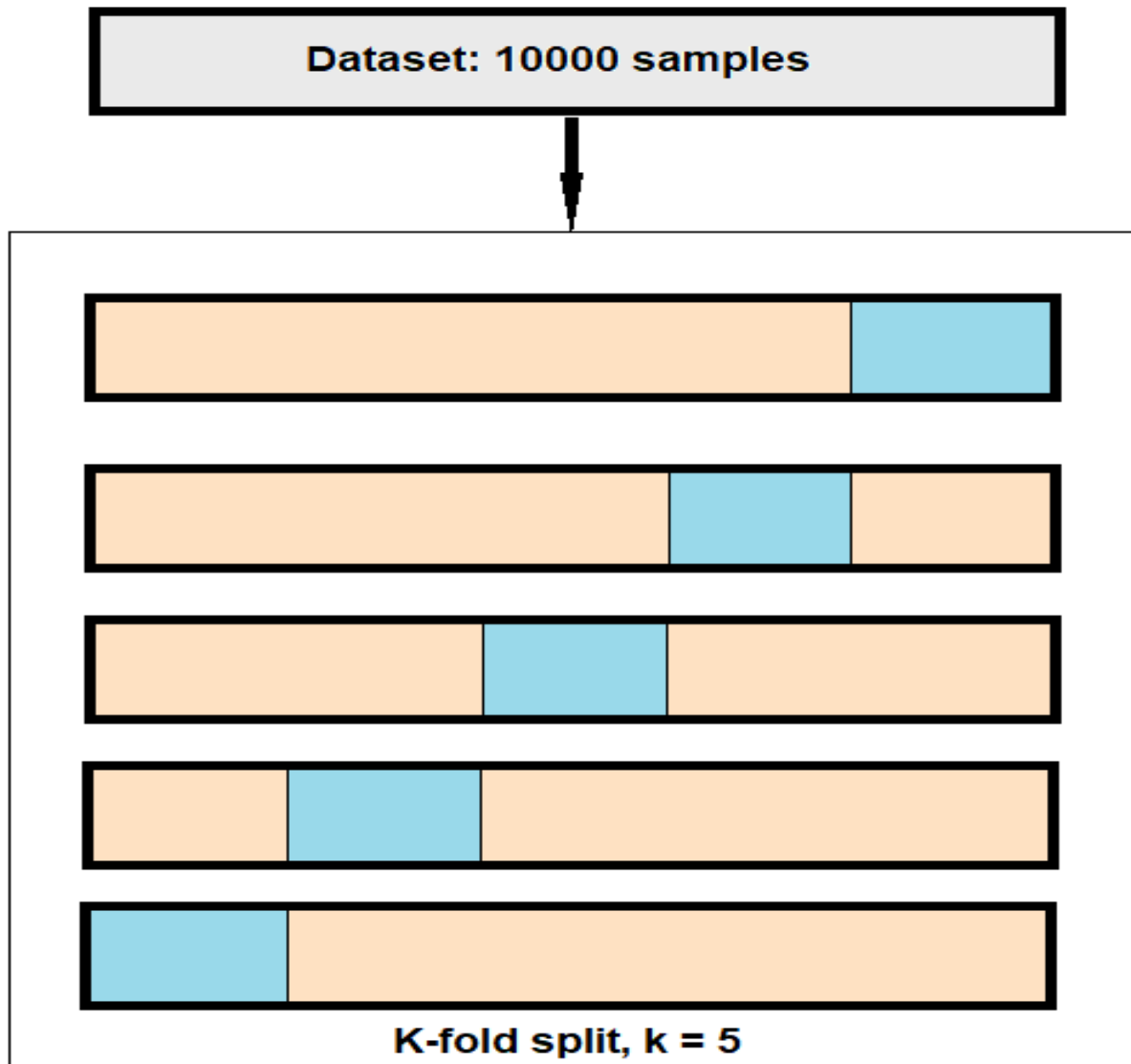


Ilustración 3.15 Divisiones en cross validation (k-fold)

Para cada división, se entrena el mismo modelo y el rendimiento se muestra por pliegue. Para fines de evaluación también se puede promediar en todos los pliegues. Si bien esto produce mejores estimaciones, la validación cruzada de K-fold también aumenta el costo de entrenamiento, en el  $K = 5$  del escenario anterior, el modelo debe entrenarse 5 veces. Las métricas calculadas con la validación cruzada

nos dan una información más cercana a la realidad si la comparamos con una evaluación única.

#### 3.4.4.3. Paso de sonido a MEL

La escala Mel es una transformación logarítmica de la frecuencia de una señal. La idea central de esta transformación es que los sonidos a la misma distancia en la escala Mel se perciben como a la misma distancia que los humanos.

En realidad, es mucho más difícil para los humanos poder diferenciar entre frecuencias más altas y más fácil para las frecuencias más bajas. Entonces, aunque la distancia entre los dos conjuntos de sonidos es la misma, nuestra percepción de la distancia no lo es. Esto es lo que hace que Mel Scale sea fundamental en las aplicaciones de aprendizaje automático para audio, ya que imita nuestra propia percepción del sonido.

$$m = 1127 * \log (1 + f/700)$$

*(Fórmula 2, transformación de la escala Hertz a escala de Mel)*

Con ayuda de la librería librosa pasaremos el sonido al dominio de frecuencia.

En la siguiente ilustración se muestra el sonido en el dominio de frecuencia:

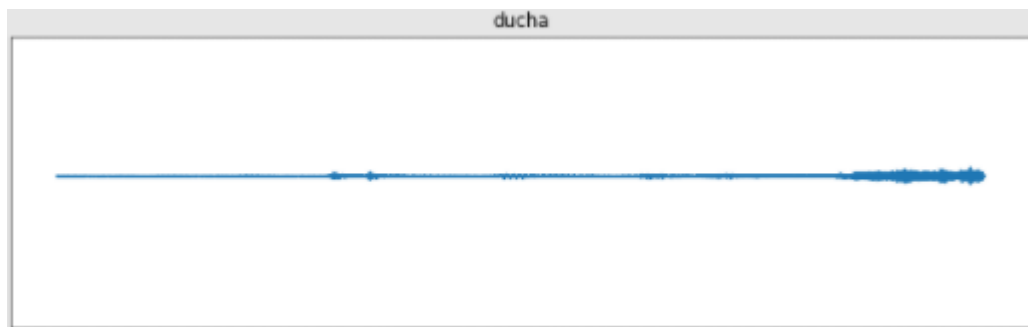


Ilustración 3.16 Dominio de frecuencia del evento ducha

Pasaremos el dominio de frecuencia a él espectrograma de mel, lo que ayudará a eliminar ruido vacío innecesario.

En la siguiente ilustración se muestra un espectrograma de mel:

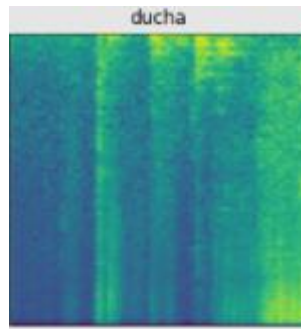


Ilustración 3.17 Espectrograma de mel del evento ducha

Por último, utilizaremos MFCC, originalmente se usaron en varias técnicas de procesamiento de voz, sin embargo, a medida que el campo de la recuperación de información musical (MIR) comenzó a desarrollarse como complemento del aprendizaje automático, se descubrió que los MFCC podían representar el timbre bastante bien.

El procedimiento básico para desarrollar MFCC es el siguiente:

- Convertir de Hertz a Mel Scale
- Usar el logaritmo de la representación Mel de audio
- Tomar la magnitud logarítmica y usar la transformación de coseno discreta
- Este resultado crea un espectro sobre las frecuencias Mel en lugar del tiempo, creando así MFCC

En la siguiente ilustración se muestra un audio convertido a MFCC:



Ilustración 3.18 MFCC del evento ducha

### 3.5. Implementación del sistema

En este apartado se hará una especificación técnica con detalle de cada uno de los componentes que se han desarrollado para conseguir el objetivo marcado, tales como, algoritmos, entornos de desarrollo, dispositivos necesarios, configuraciones y scripts ejecutables.



### 3.5.1. Lenguaje de programación escogido

Python es un lenguaje de programación interpretado de propósito general, multiplataforma y multiparadigma (imperativo, orientado a objetos y funcional) [43]. Los motivos de la elección de este lenguaje es una sintaxis clara e intuitiva que facilita la legibilidad del código, también la creciente adopción del aprendizaje automático en todo el mundo es un factor importante que contribuye a su creciente popularidad de python, se ha convertido en la opción favorita para el análisis de datos, el aprendizaje automático y la inteligencia artificial, todo gracias a su vasto ecosistema de bibliotecas que permite a los profesionales del aprendizaje automático acceder, manejar, transformar y procesar datos con facilidad.

En la siguiente ilustración se muestra una gráfica hasta el año 2020 sobre el uso de los lenguajes de programación en el aprendizaje automático [42]:

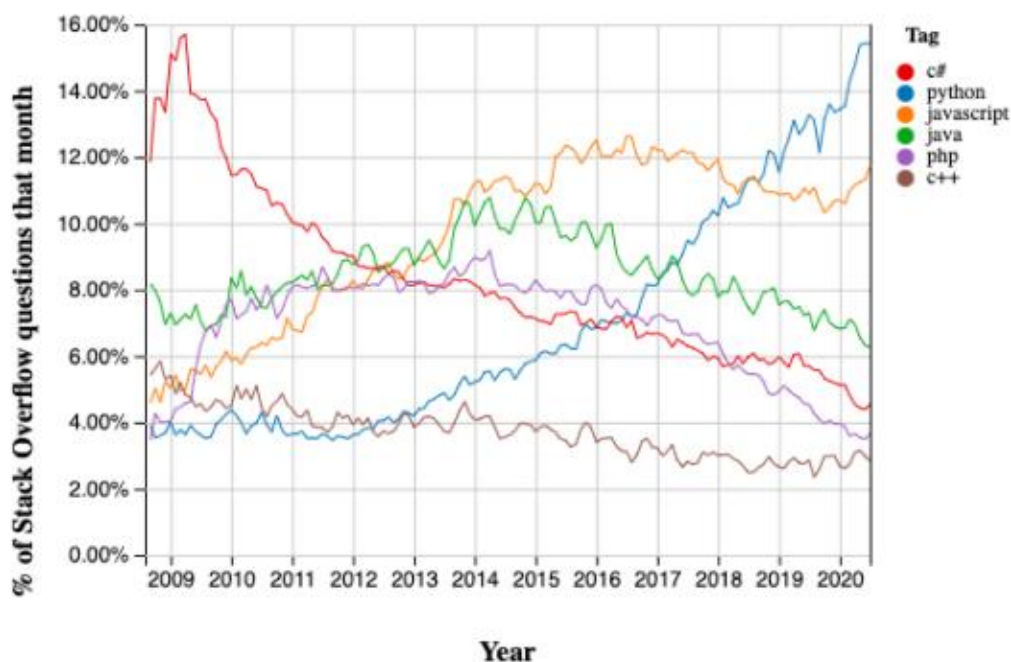


Ilustración 3.19 Uso de lenguajes de programación para aprendizaje automático

Es por esto por lo que la elección de este lenguaje ha sido clave para el desarrollo de este proyecto, gracias a la gran comunidad que hay detrás y la facilidad con la que podemos procesar datos y encontrar bibliotecas altamente especializadas para la generación del modelo de predicción de sonido ambiental con redes neuronales convolucionales.

### 3.5.2. Preparación de la Raspberry pi

En esta sección hablaremos sobre el dispositivo IoT que hemos utilizado, sus características y las configuraciones necesarias para que el sistema desarrollado funcione correctamente cuando se haga un despliegue en este.

#### 3.5.2.1. Características Raspberry

- **Serie:** Raspberry PI 3 Model B
- **Procesador:** Core 2 Quad 1.2 GHz.
- **RAM:** 1GB DDR3
- **Disco:** 64 GB microSD
- **Conectividad:** Wi-Fi
- **Puertos USB 2.0:** 4
- **Dimensiones:** 12.19 x 7.59 x 3.4 cm; 45 gramos

A continuación se muestra una Ilustración de la Raspberry PI utilizada para este trabajo fin de grado:



Ilustración 3.20 RaspBerry PI

#### 3.5.2.2. Instalación sistema operativo

En la instalación del sistema operativo debemos introducir la imagen ISO del sistema elegido en la tarjeta de memoria para posteriormente iniciar la instalación del sistema, este proceso lo hemos llevado a cabo con la herramienta BalenaEtcher.

- **BalenaEtcher:** Es una herramienta que permite hacer una memoria booteable para arrancar la instalación del sistema operativo, el uso es

fácil e intuitivo, se debe seleccionar la imagen ISO, la memoria deseada y por último ejecutar el programa.

A continuación, se muestra una Ilustración sobre la interfaz de balenaEtcher para bootear una ISO:

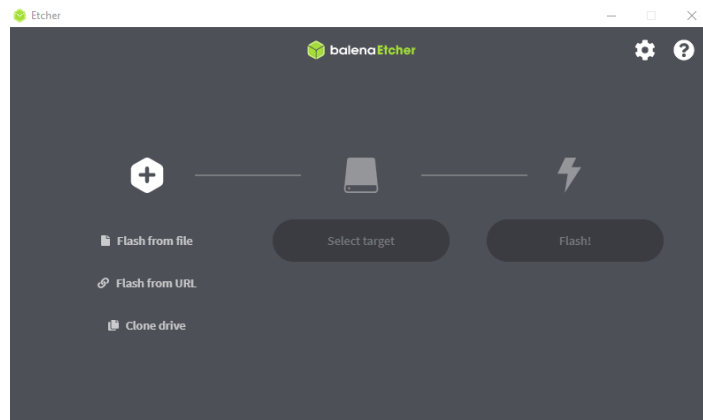


Ilustración 3.21 Interfaz aplicación balenaEtcher

- **Imagen ISO:** En este caso para mayor facilidad de uso y compatibilidad con RaspBerry, se ha elegido una imagen del sistema operativo de Raspbian, quederiva de Debian y es un sistema en el que es cómodo integrar todas las librerías de Python, además, el entorno es bastante familiar, por lo que simplifica las posibles dificultades que se presenten.
- **Instalación sistema operativo:** Una vez elegido el sistema operativo e incorporado en nuestra tarjeta de memoria microSD de manera correcta, este paso es sencillo, tan solo es necesario introducir la tarjeta en la RaspBerry pi y proceder a encenderla. La instalación se puede hacer usando teclado, ratón y pantalla, de una manera muy intuitiva sin dar lugar a mayores problemas.

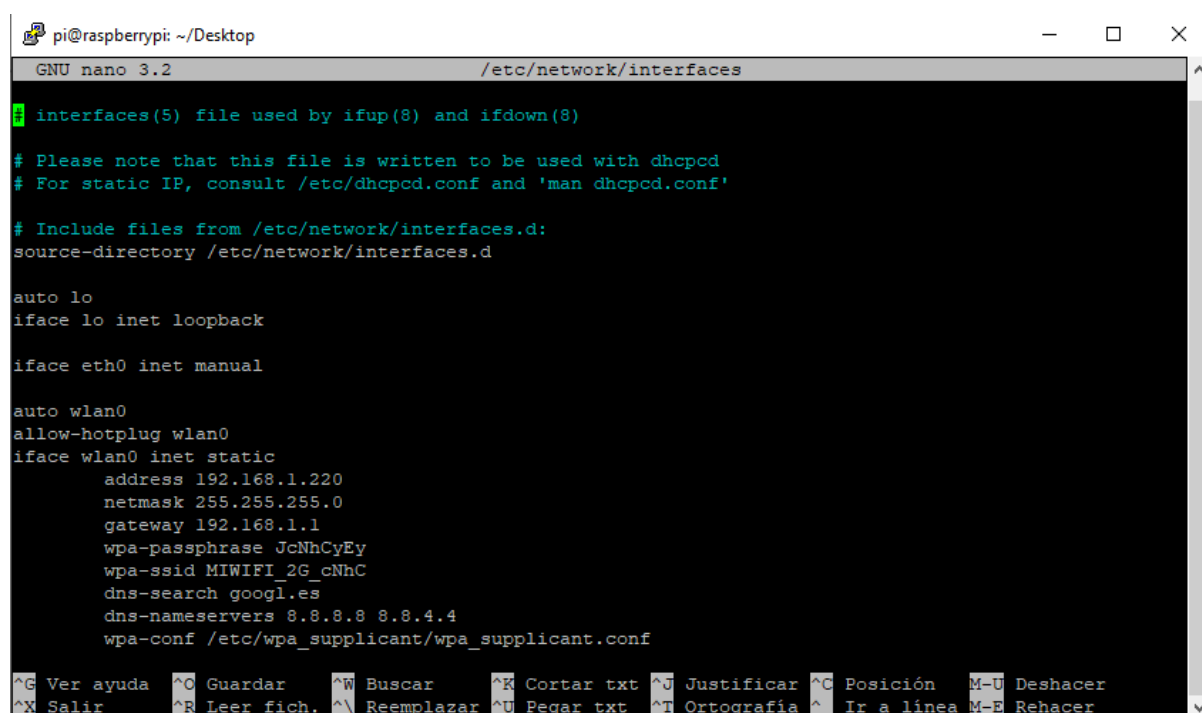
### 3.5.2.3. Configuración conexión wifi

Este subapartado es importante, ya que necesitamos acceder remotamente al dispositivo para hacer una serie de grabaciones y empezar a generar nuestro DataSet. Por tanto, es necesario introducir los parámetros de la Wifi adecuadamente

para que el dispositivo se encuentre en la misma subred y se puedan realizar las conexiones pertinentes.

En la siguiente figura, podemos observar que accedemos al fichero `/etc/network/interfaces` y añadimos la configuración de nuestro router poniendo una IP estática para que siempre sepamos identificar el dispositivo IoT.

A continuación, se muestra una ilustración de la configuración del interfaz wifi para conectar la Raspberry a la red:



```
pi@raspberrypi: ~/Desktop
GNU nano 3.2 /etc/network/interfaces
interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpd
# For static IP, consult /etc/dhcpd.conf and 'man dhcpd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.1.220
    netmask 255.255.255.0
    gateway 192.168.1.1
    wpa-passphrase JcNhCyEy
    wpa-ssid MIWIFI_2G_cNhC
    dns-search googl.es
    dns-nameservers 8.8.8.8 8.8.4.4
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

^G Ver ayuda  ^O Guardar  ^W Buscar  ^K Cortar txt  ^J Justificar  ^C Posición  M-U Deshacer
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar txt  ^I Ortografía ^_ Ir a línea M-E Rehacer
```

Ilustración 3.22 Fichero configuración para la interfaz de red

Una vez puesta en marcha esta configuración y después de reiniciar el sistema habrá quedado listo para el siguiente paso.

#### 3.5.2.4. Habilitar servicio SSH

En este proyecto uno de los objetivos es la recolección de muestras de eventos diarios. Para ello, se ha de poder trasladar la RaspBerry a determinadas habitaciones del hogar para realizar las grabaciones de los eventos, los cuales son típicos de cada habitáculo. Por esto, es importante que podamos acceder al

dispositivo de forma remota para enviar las correspondientes órdenes y empezar a capturar los eventos grabados con un sensor de audio.

### **Solución 1:** Crear un archivo ssh en el directorio de arranque de la tarjeta SD

Si no disponemos de un ratón, teclado y pantalla podemos activar el servicio SSH en nuestra Raspberry accediendo de manera externa a la tarjeta microSD en la que está instalada el sistema operativo de Raspbian, en el directorio de arranque crearemos un fichero llamado ssh sin ninguna extensión, después reiniciamos la Raspberry y el acceso a SSH lo encontraremos habilitado.

### **Solución 2:** Activar el servidor SSH en el escritorio

Si tenemos la posibilidad de conectar un ratón, teclado y pantalla podemos activar el servicio SSH en nuestra RaspBerry desde el “menú de inicio” → preferencias → “configuración raspberry pi” y en la pestaña interfaces aparecerá SSH deshabilitado de manera predeterminada, podemos hacer el cambio para habilitarla y con esto estaría listo para funcionar.

#### **3.5.2.5. Instalación de python y librerías requeridas**

Se ha de tener en cuenta que determinadas librerías solo funcionan con algunas versiones de Python, es por ello que se ha seleccionado la versión 3.7.3, compatible con todos los módulos necesarios para el desarrollo del proyecto.

Para proceder a instalar Python en nuestro dispositivo ejecutamos las siguientes órdenes:

- `sudo apt update`  
`sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev curl libbz2-dev`
- `curl -O https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tar.xz`
- `tar -xf Python-3.7.3.tar.xz`

- `cd Python-3.7.3`  
`./configure --enable-optimizations`
- `make -j 8`
- `sudo make altinstall`
- `python3.7 --version`

En resumen, lo que hacemos es actualizar el sistema operativo, instalar los paquetes necesarios para compilar la fuente Python, descargar el código fuente de la versión deseada en este caso la 3.7.3, proceder a descomprimir e instalar la versión descargada y, por último, comprobar que la versión instalada sea la correcta.

Por otro lado, las librerías necesarias para que todo el proyecto funcione correctamente y podamos grabar audios y generar el modelo de predicción, se encuentra en el fichero `requirements.txt`, adjunto en la carpeta del proyecto en la entrega del trabajo fin de grado. Por comentar algunas de las librerías más importantes, nos encontramos con:

- **Pyaudio:** Para iniciar la grabación de audio recogida por el micrófono.
- **Numpy:** Para operar con vectores y matrices de una manera más fácil.
- **Tensorflow:** Librería que nos sirve para crear modelos de deep learning.
- **AudioSegment:** Se utiliza en este proyecto para particionar las escenas de varios minutos de duración en audios de 3 segundos.
- **Librosa:** Es un paquete de Python para análisis de audio y música. Proporciona los componentes básicos necesarios para crear sistemas de recuperación de información musical.

### 3.5.3. Entorno de programación usado

Jupyter notebook es el entorno de programación seleccionado para desarrollar el proyecto encomendado, algunos de sus puntos fuertes son:

- Permite establecer bloques de código que se pueden ejecutar independientemente.

- Tiene un lenguaje de marcas que permite establecer imágenes, títulos, texto de forma que se puede explicar cada bloque de una manera más vistosa y natural.
- Gracias a su ejecución en bloques no es necesario volver a ejecutar todo para hacer cambios en el código y ver los resultados, solo basta con volver a ejecutar el trozo de código que hemos actualizado.
- Permite exportar el código en varios formatos de archivos.

Para su instalación se puede hacer de una forma muy sencilla, solo hay que instalarlo con el comando 'pip install jupyterlab' y el comando 'pip install notebook' con esto sería más que suficiente para empezar a usarlo.

Para abrir el entorno de trabajo lo haremos introduciendo la orden 'jupyter notebook' en una terminal. En la siguiente figura se muestra el entorno con una división de bloques de código que se ejecutan por separado, se puede observar cómo detecta la versión de python instalada y algunas de las herramientas que nos ofrece.

En la siguiente ilustración se muestra el entorno de desarrollo jupyter notebook:

The screenshot shows a Jupyter Notebook window titled 'water-music-jose' with a last checkpoint of '26/04/2021 (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and output. The code in the first cell imports various libraries including pandas, matplotlib, librosa, sklearn, tensorflow, and keras. It also sets up paths for a CSV file and a folder. The second cell runs the code to load the CSV file into a DataFrame. The output shows a preview of the DataFrame with columns 'name', 'target', and 'category'.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import librosa
import librosa.display
from tqdm import tqdm
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
import tensorflow.keras.models as models
import tensorflow.keras.layers as layers
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from tensorflow.keras.models import Sequential

from sklearn.model_selection import KFold
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

CSV_FILE_PATH = "../dataSet.csv" # path of csv file
DATA_PATH = "../sonidos/" # path to folde
```

```
In [2]: df = pd.read_csv(CSV_FILE_PATH)
df
```

```
Out[2]:
```

	name	target	category
0	aspirador1.wav	1	aspirador
1	aspirador2.wav	1	aspirador
2	aspirador3.wav	1	aspirador
3	aspirador4.wav	1	aspirador
4	aspirador5.wav	1	aspirador
...	...	...	...
1495	timbre96.wav	15	timbre
1496	timbre97.wav	15	timbre
1497	timbre98.wav	15	timbre

Ilustración 3.23 Interfaz de desarrollo de jupyter notebook

En definitiva, el entorno nos acelera el proceso de desarrollo ya que no tener que cargar varias veces un dataSet grande o el simple hecho de no tener que ejecutar todo para ver los resultados simplifica bastante el trabajo.

### 3.5.4. Recolección de datos y etiquetado

Es de vital importancia tener un DataSet extenso con muchos ejemplos de lo que queremos clasificar, en nuestro caso, la recopilación de audios se ha realizado de los eventos diarios/comunes que suceden en un hogar. Es por ello que nuestra Raspberry ya configurada y lista para usarse, hace uso de un script desarrollado en el lenguaje de programación Python. Dicho script se encarga de grabar el sonido ambiente donde se encuentra el dispositivo IoT.



Vamos a hablar a continuación del funcionamiento de este programa, utilizando la librería Pyaudio de Python y con unos parámetros de entrada tal y como se explica en el caso de uso 1. Grabamos tantos audios de 3 segundos como la diferencia entre el índice final e inicial. El código del script esta adjunto en el anexo en el apartado de [Scripts](#):

Una vez grabado el evento deseado y cogido las muestras necesarias (en este caso de estudio se han cogido 100 muestras por cada evento) procederemos a enviarlas al ordenador donde estamos desarrollando el proyecto a través de SSH.

Es ahora cuando procedemos a etiquetar cada audio recopilado en un fichero CSV con los atributos:

- Name: Nombre del audio (Ej.: aspirador1.wav)
- Target: Número entero asociado a la categoría en nuestro proyecto estudiamos 16 categorías diferentes por tanto un número entre el 1 y el 16, para el ejemplo anterior sería el (1).
- Category: Nombre de la categoría para el ejemplo anterior sería (aspirador).

Para hacer este trabajo se ha desarrollado el script del caso de uso 4 que se muestra a continuación, donde podemos ver que el diccionario nombrado clasificación contiene como clave la categoría y como valor un par (target, número de audios) por tanto como los audios los tenemos ordenados por categoría tal que (aspirador1.wav, aspirador2.wav ... aspirador(n).wav) generamos un CSV correctamente en un breve instante de tiempo. El código del script esta adjunto en el anexo en el apartado de [Scripts](#):

Este es el método que hemos utilizado para generar un DataSet de audio de los eventos domésticos diarios.

Aún tenemos que seguir explicando cómo hemos resuelto el objetivo marcado de estudiar un caso real, es por este motivo que también hemos desarrollado unos scripts que nos permiten grabar escenas, lo que significa que son audios que duran minutos donde suceden varios eventos a lo largo del tiempo, y otros dos scripts que

nos permiten generar el CSV de la escena grabada y dividir audios de varios minutos en audios de 3 segundos.

Para la división de las escenas tenemos el caso de uso 2, donde se recibe como parámetro el nombre del audio que queremos dividir y el número de divisiones, estos scripts simplifican y automatizan todo el proceso de recolección, división y etiquetado de audio. El código del script esta adjunto en el anexo en el apartado de [Scripts](#):

Es necesario una vez divida la escena en  $n$  audios, escucharlos uno a uno e identificar de qué evento se está tratando ya que posteriormente se realizará una predicción de dicha escena y es necesario haber etiquetado correctamente los audios para ver la tasa de acierto que tiene el modelo generado. También realizamos el etiquetado de estos audios en un fichero CSV, con la diferencia que ahora los audios vienen ordenados con el nombre (part1.wav, part2.wav, ... part(n).wav) en el orden en el que se grabó, por tanto nos ayudaremos de un script que nos reduce la carga de trabajo al poder hacer de manera automática el fichero CSV escribiendo los parámetros adecuadamente tal y como se muestra en el código del script que esta adjunto en el anexo en el apartado de [Scripts](#):

Básicamente estamos añadiendo la secuencia de eventos que nos encontramos a lo largo de la escena, indicando: category, target, número de audios a continuación del mismo evento. Si sumamos todos los audios que aparecen en el código son 160, y como su duración es de 3 segundos, podemos determinar que la escena grabada fué de 8 minutos.

Con esto ya tenemos nuestro DataSet con su correspondiente etiquetado en formato CSV y las escenas reales que vamos a evaluar con su correspondiente etiquetado en formato CSV.

### **3.5.5. Conversión de audio a imagen (tratamiento de los datos)**

En este proyecto se han capturado las señales de audio de cada evento a 44,1kHz, como la entrada de nuestra CNN es una imagen y no un sonido, se ha

pasado todos los audios de las diferentes categorías a imágenes haciendo uso de la librería 'Librosa' de Python.

Los coeficientes cepstrales en las frecuencias de mel (MFCC) resultante de las frecuencias positivas desarrollaron imágenes de tamaño 13 x 130.

La siguiente ilustración muestra la señal de audio sin procesar de cada uno de los eventos que vamos a clasificar:

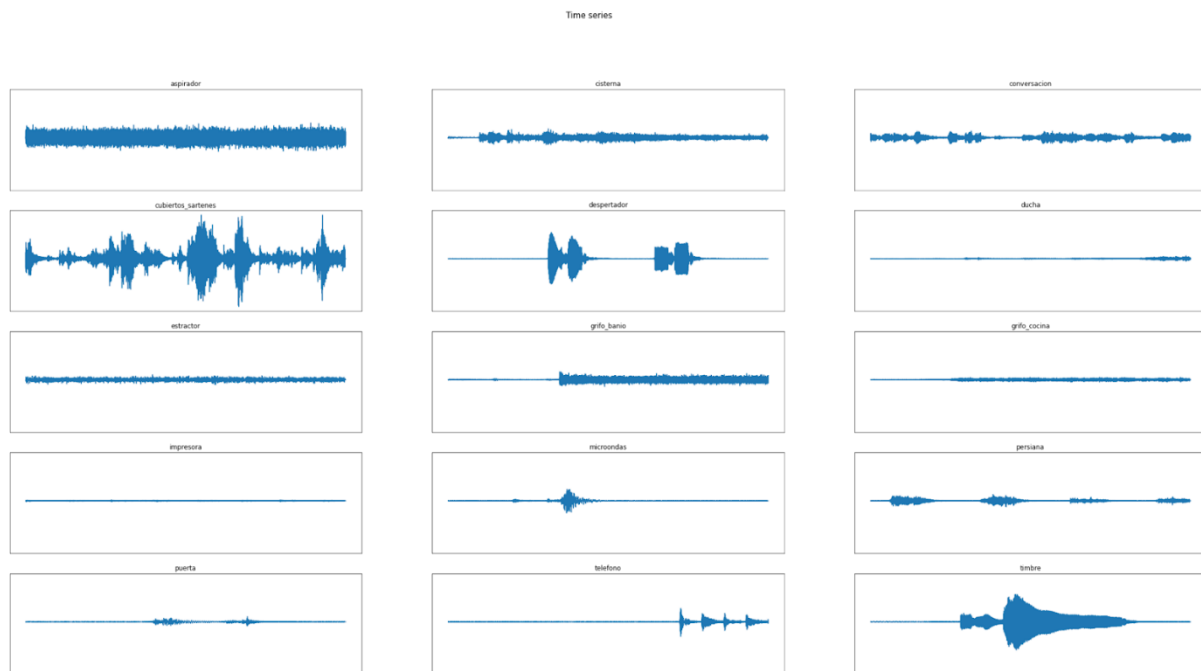


Ilustración 3.24 Señal de audio de los eventos a clasificar

En la siguiente ilustración se muestra el espectrograma log-Mel (LM):

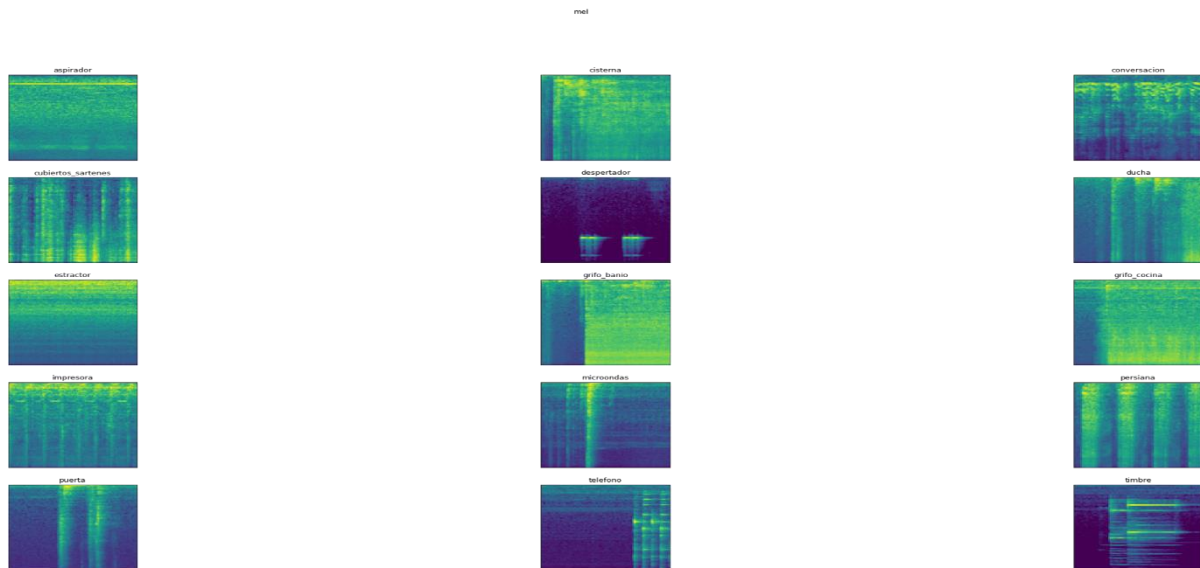


Ilustración 3.25 Espectrograma de mel de los eventos a clasificar

Por último y con lo que vamos a trabajar como datos de entrada en nuestra CNN tenemos en la siguiente Ilustración los MFCC:

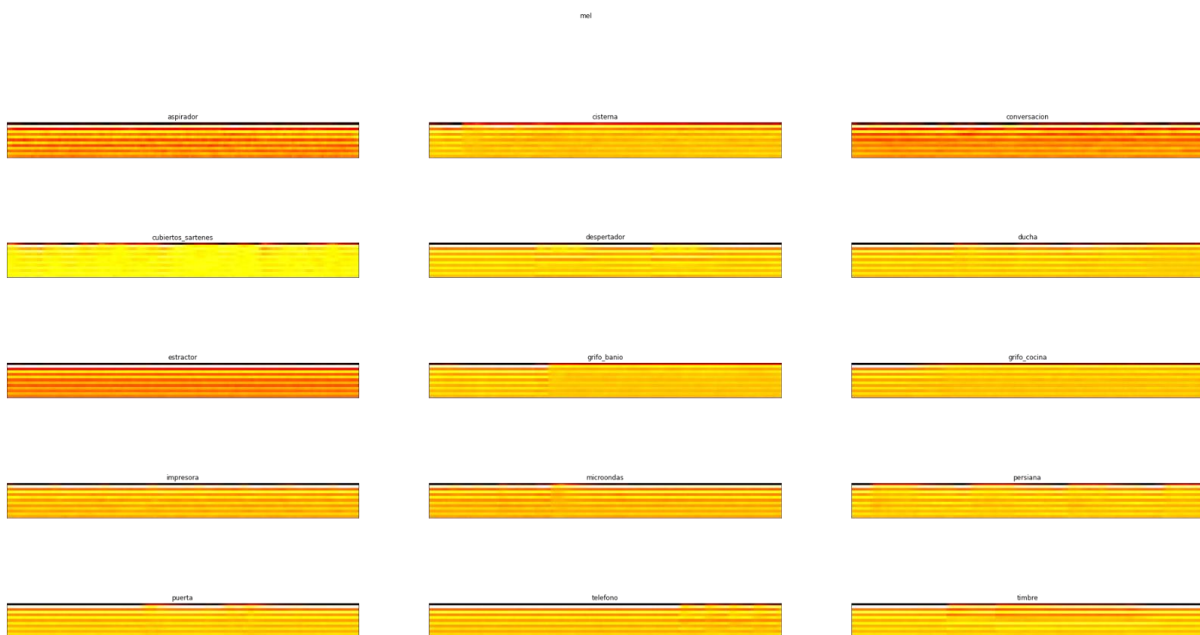


Ilustración 3.26 MFCC de los eventos a clasificar

### 3.5.6. Arquitectura red neuronal

Las CNN como hemos visto en anteriormente son muy buenas extrayendo y clasificando características. En este trabajo, se evalúa un modelo CNN con cinco capas convolucionales para el procesamiento MFCC, donde se incluye una

agrupación promedio única después de las convoluciones debido al espacio de entrada reducido  $13 \times 130 \times 1$

Tabla con los datos de la arquitectura de red del modelo CNN +MFCC:

Arquitectura de red del modelo CNN +MFCC	
Input	$13 \times 130 \times 1$
Conv(3 × 3)	$11 \times 128 \times 16$
Conv(3 × 3)	$9 \times 126 \times 16$
Conv(3 × 3)	$7 \times 124 \times 32$
Conv(3 × 3)	$5 \times 122 \times 64$
Conv(3 × 3)	$3 \times 120 \times 128$
Conv(3 × 3)	$1 \times 118 \times 256$
GlobalAvgPool2D	256
Dense	1024
Dense	15

Tabla 3.1 Arquitectura de red del modelo CNN + MFCC

Los parámetros a resaltar que se utilizan en esta arquitectura son los siguientes:

- **Relu:** Función de activación relu, es muy buena para CNN y solo permite que los valores positivos pasen a la siguiente capa, los negativos se actualizan a cero.
- **Softmax:** Función de activación softmax se suele utilizar para comprimir un vector K-dimensional a valores reales entre  $[0,1]$  de esta manera sabemos a qué clase pertenece el input a predecir cogiendo el que mayor peso tenga.

Los parámetros que se ha utilizado para la compilación del modelo son:

- **Adam:** La optimización de Adam es un método de descenso de gradiente estocástico que se basa en la estimación adaptativa de momentos de primer y segundo orden.
- **Accuracy:** Esta métrica crea dos variables locales, **total** y **count** que se utilizan para calcular la frecuencia con la que  $y\_pred$  coincide con  $y\_true$ . Esta frecuencia se devuelve finalmente como binary accuracy: una operación idempotente que simplemente divide total por count.

El entrenamiento del modelo generado por nuestra red neuronal convolucional lo llevamos a cabo utilizando el 80% de nuestro dataSet como entrenamiento y el 20% restante para realizar test que, a priori, nos dará un resultado de cómo de bueno es nuestro modelo, aunque esta evaluación no es suficiente para decidir si un modelo es suficientemente robusto o no. Por ello, en el siguiente apartado se explicará un método mucho más robusto para decidir si un modelo generado es lo suficientemente bueno para el problema que estamos resolviendo.

Uno de los parámetros que tenemos que tener en cuenta a la hora de utilizar la función 'fit' del modelo es el llamado 'epoch', que se le asigna un número entero positivo que indica el número de vueltas que se le dá al 80% de los datos elegidos aleatoriamente del dataSet. Si elegimos un número muy alto de veces puede sobreentrenarse y dar resultados muy buenos con los datos de nuestro dataSet con el que se entrenó pero muy malas predicciones con datos nuevos con los que no han entrenado, es importante tener esto en cuenta para que no exista el llamado "overfitting".

## 4. Evaluación

Para la evaluación de este trabajo se ha implementado dos evaluaciones:

1. La validación cruzada **k-fold**, que nos permite tener unos resultados muy robustos sobre las predicciones del modelo. Eso es así por que como anteriormente se explicó esta validación trabaja sobre el conjunto de datos haciendo 10 divisiones de la siguiente manera:
  - a. Se divide el conjunto de datos el 90% para entrenamiento y el 10% para realizar tests.
  - b. Empieza el entrenamiento del modelo.
  - c. Se evalúa el modelo con el 10% de datos restante.
  - d. Es en este punto cuando hemos completado la primera evaluación de las 10 en la que hemos dividido este test. Por tanto, volvemos a coger un 90% de los datos para entrenar y un 10% para el test (Este 10% será distinto a los anteriormente seleccionados).

e. Se repite el proceso hasta que se hagan las  $k = 10$  evaluaciones.

2. La validación de un modelo en tiempo real sobre cuatro escenarios, en los que se recolectaron muestras de audio de micrófonos ambientales mientras el habitante realizaba actividades de la vida diaria en condiciones naturalistas.

Los DataSet utilizados para el entrenamiento / evaluación de los modelos generados en este estudio constaron de los siguientes audios:

	N.º sonidos (k-fold)	N.º sonidos (Escena1)	N.º sonidos (Escena2)	N.º sonidos (Escena3)	N.º sonidos (Escena4)
aspirador	100	100	100	100	100
cisterna	50	100	100	100	100
conversacion	100	100	100	100	100
cubiertos_sartenes	100	100	100	100	100
despertador	100	100	100	100	100
ducha	100	100	100	100	100
extractor	100	100	100	100	100
grifo_baño	100	100	100	100	100
grifo_cocina	100	100	100	100	100
impresora	100	100	100	100	100
microondas	100	100	100	100	100
persiana	100	100	100	100	100
puerta	100	100	100	100	100
telefono	100	100	100	100	100
timbre	100	100	100	100	100
idle	0	201	213	227	238
<b>TOTAL:</b>	1450	1701	1713	1727	1738

Tabla 4.1 Tabla de parámetros del DataSet utilizado para generar y evaluar el modelo

Indicar que los parámetros utilizados para las escenas en el evento idle varían de una escena a otra ya que se hace de la siguiente forma:

- Se graba la escena y segmenta en audios de 3 segundos.
- Los audios que pertenecen a la categoría idle son usados en el resto de escenas. De esta manera para el entrenamiento del modelo de la escena 1 se usan los audios que pertenecen a la categoría idle del resto de escenas (2,3,4), para la escena 2 se usaran los audios de dicha categoría de las escenas (1,3,4) y así respectivamente con el resto de escenas.

- Haciéndolo de esta forma, podemos entrenar los modelos con un mayor número de audios en esta categoría, haciendo que estos sonidos al ser grabados en diferentes habitaciones nuestro modelo aprenda a diferenciar esta categoría idle con mayor precisión.

#### 4.1. Pruebas

Una vez se ha llegado a este punto del trabajo de fin de grado, solo queda plasmar los datos extraídos de la realización de las pruebas para la comprobación de la viabilidad y robustez de los modelos generados, así como los tiempos de ejecución en la raspberry pi. Para saber a los eventos sonoros que nos referimos en el contexto de las actividades diarias, se ha confeccionado la siguiente tabla:

Tabla de eventos sonoros y descripciones desarrolladas en el contexto de las actividades diarias:

Clase	Descripción
<b>Aspiradora</b>	Muestra de audio de pasar la aspiradora
<b>Cisterna</b>	Muestra de audio de inodoro con descarga de agua
<b>Conversación</b>	Muestra de audio de personas conversando
<b>Cubiertos + sartenes</b>	Muestra de audio de cubiertos y sartenes
<b>Despertador</b>	Muestra de audio del sonido del despertador
<b>Ducha</b>	Muestra de audio de la ducha
<b>Extractor</b>	Muestra de audio de un extractor
<b>Grifo de la cocina</b>	Muestra de audio de un grifo de cocina
<b>Grifo del baño</b>	Muestra de audio de un grifo de baño
<b>Impresora</b>	Muestra de audio de una impresora en funcionamiento
<b>Microondas</b>	Muestra de audio de un microondas en funcionamiento
<b>Persiana</b>	Muestra de audio de una persiana moviéndose
<b>Puerta</b>	Muestra de audio de puerta que se abre o se cierra
<b>Teléfono</b>	Muestra de audio de un teléfono sonando
<b>Timbre de la puerta</b>	Muestra de audio de un timbre

Tabla 4.2 Descripción eventos clasificados

También indicar como y donde se ha desplegado el sistema para desarrollar las pruebas, en primer lugar, para la grabación de las escenas siempre se ha contado con la Raspberry Pi, un micrófono, una batería portátil para poder movernos de una habitación a otra cuando así lo requiere el siguiente escenario a grabar con la Raspberry siempre encendida. Y en segundo lugar las habitaciones donde se colocaron el sensor de audio para capturar estas escenas naturalistas.



A continuación, se muestra una ilustración de la colocación del dispositivo en la escena 1 (Cocina):



Ilustración 4.1 Despliegue en escena 1

A continuación se muestra una ilustración de la colocación del dispositivo en la escena 2 (Salón):



Ilustración 4.2 Despliegue en escena 2

A continuación se muestra una ilustración de la colocación del dispositivo en la escena 1 (Dormitorio):



**Ilustración 4.3 Despliegue en escena 3**

A continuación se muestra una ilustración de la colocación del dispositivo en la escena 1 (Baño):



**Ilustración 4.4 Despliegue en escena 4**

## 4.2. Casos

### 4.2.1. Modelo evaluado con la técnica de validación cruzada

Se tiene un dataSet con 1500 audios de eventos diarios, 100 por cada categoría. Estos datos se encuentran reflejados en un fichero CSV con sus correspondientes nombres, categorías y etiquetas.

La prueba consiste en introducir las imágenes generadas por cada uno de estos audios (MFCC) como entrada en nuestro modelo generado con la arquitectura anteriormente vista en el punto **3.5.6** con el método de evaluación validación cruzada explicado en el punto **3.5.7** de esta forma podremos sacar unas conclusiones que nos servirá para ver la viabilidad del reconocimiento de eventos diarios a través de una CNN.

### 4.2.2. Modelos para las escenas naturalistas

Uno de los objetivos de este trabajo fin de grado es evaluar la posibilidad de integrar el modelo en tiempo real incluyendo un caso de estudio de una escena real. En este estudio se han estudiado cuatro escenas reales que, a continuación, vamos a desglosar. Nos interesan los resultados, ya que, de ser favorables, puede tener esta investigación líneas de trabajo futuras muy interesantes.

**Escena 1)** El habitante llega a casa, se dirige hacia la cocina y comienza a hablar, después empieza a usar cubiertos, más tarde enciende el extractor durante un tiempo prolongado, después abre el grifo y enciende el microondas. El habitante interactúa con los siguientes objetos para hacer estos eventos citados anteriormente:

Tabla que recoge los eventos y objetos que intervienen en la escena 1:

Evento	Objeto/s
Llegar a casa.	Timbre.
Se dirige a la cocina.	Puerta.
Comienza a hablar.	Conversación.
Usar cubiertos.	Tenedores, cucharas, cuchillos.
Encender el extractor.	Extractor.
Abrir el grifo.	Grifo de la cocina.
Encender el microondas.	Microondas.

Tabla 4.3 Objetos con los que se interactúa en la escena 1

Duración de la escena 12 minutos.

**Escena 2)** El habitante llega a casa, se dirige hacia el salón y comienza a hablar, después empieza a pasar la aspiradora, más tarde abre y cierra persianas hasta que le llaman al teléfono. El habitante interactúa con los siguientes objetos para hacer estos eventos citados anteriormente:

Tabla que recoge los eventos y objetos que intervienen en la escena 2:

Evento	Objeto/s
<b>Llegar a casa.</b>	Timbre.
<b>Se dirige hacia el salón.</b>	Puerta.
<b>Comienza a hablar.</b>	Conversación.
<b>Pasar la aspiradora.</b>	Aspirador.
<b>Abrir y cerrar persianas.</b>	Persiana
<b>Llamada de teléfono.</b>	Teléfono

Tabla 4.4 Objetos con los que se interactúa en la escena 2

Duración de la escena 9 minutos.

**Escena 3)** El habitante llega a casa, se dirige hacia el dormitorio y comienza a hablar, después suena el despertador durante un tiempo prolongado, abre la persiana del dormitorio y después se pone a imprimir unos documentos. El habitante interactúa con los siguientes objetos para hacer estos eventos citados anteriormente:

Tabla que recoge los eventos y objetos que intervienen en la escena 3:

Evento	Objeto/s
<b>Llegar a casa.</b>	Timbre.
<b>Se dirige hacia el dormitorio.</b>	Puerta.



<b>Comienza a hablar.</b>	Conversación.
<b>Suena el despertador.</b>	Despertador.
<b>Abrir y cerrar persianas.</b>	Persiana.
<b>Imprime documentos.</b>	Impresora.

Tabla 4.5 Objetos con los que se interactúa en la escena 3

Duración de la escena 8 minutos.

**Escena 4)** El habitante se dirige hacia el cuarto baño, después abre el grifo, más tarde se da una ducha durante un tiempo prolongado, después cierra la puerta del baño, estira de la cisterna del váter y luego se marcha del cuarto de baño. El habitante interactúa con los siguientes objetos para hacer estos eventos citados anteriormente:

Tabla que recoge los eventos y objetos que intervienen en la escena 4:

<b>Evento</b>	<b>Objeto/s</b>
<b>Se dirige al cuarto de baño.</b>	Puerta.
<b>Abre el grifo del baño.</b>	Grifo baño.
<b>Se da un baño.</b>	Ducha.
<b>Cierra la puerta del baño.</b>	Puerta.
<b>Estira de la cisterna.</b>	Váter.
<b>Sale del cuarto de baño.</b>	Puerta.

Tabla 4.6 Objetos con los que se interactúa en la escena 4

Duración de la escena 8 minutos.

#### 4.2.3. Tiempo de ejecución en Raspberry

Esta prueba sirve para ver la viabilidad de predecir sonidos en tiempo real, ya que los audios generados son de 3 segundos nos gustaría saber que no se produce ningún delay cuando lo ponemos en marcha en la raspberry pi. También se quiere saber el tiempo que tarda la placa inteligente en generar un modelo.

### 4.3. Resultados

#### 4.3.1. Modelo evaluado con la técnica de validación cruzada

Los resultados mostrados a continuación son extraídos de la evaluación del modelo que utiliza solamente el dataSet para entrenarse y evaluarse.

Tabla de resultados de la evaluación cruzada:

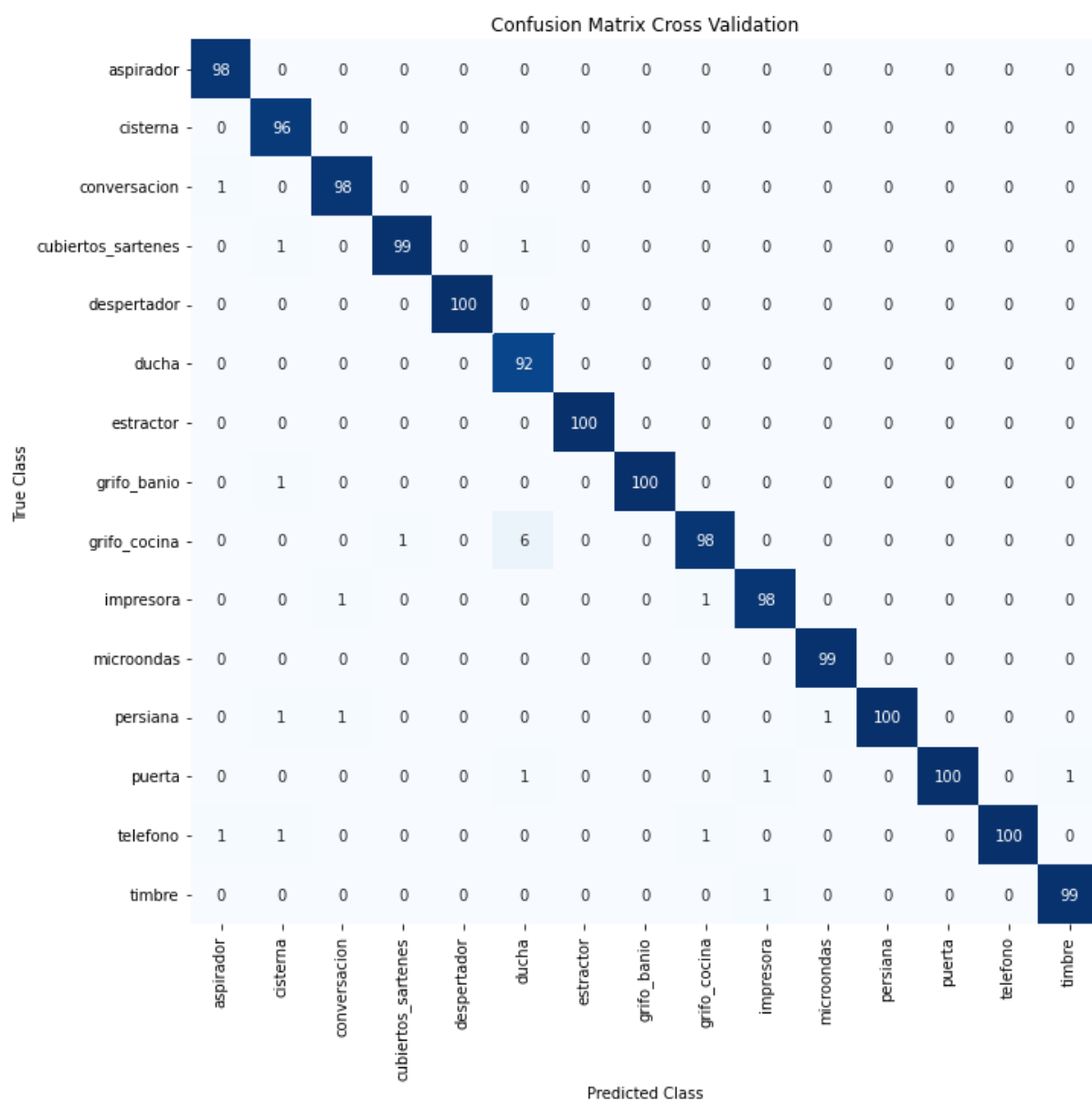
División	Loss	Accuracy
División 1	0.0046	100.00%
División 2	0.0499	99.33%
División 3	0.6603	89.99%
División 4	0.0034	100.00%
División 5	0.1058	98.00%
División 6	0.0929	98.66%
División 7	0.0313	99.33%
División 8	0.0013	100.00%
División 9	0.0584	99.33%
División 10	0.0003	100.00%

Tabla 4.7 Resultados evluación cruzada

Puntajes promedio para todas las divisiones:

- Accuracy: 98.46 (+- 2.89)
- Loss: 0.1008

En la siguiente ilustración vemos la matriz de confusión obtenida es la siguiente:



**Ilustración 4.5 Matriz confusión con la técnica validación cruzada**

Como se ve reflejado en los datos pasando los sonidos a imágenes MFCC conseguimos que el modelo generado con CNN consiga una tasa de acierto bastante elevada, por lo que el siguiente paso es comprobar si en escenas naturalistas sigue dando tan buenos resultados.

### 4.3.2. Modelos para las escenas naturalistas

#### 4.3.2.1. Resultados para la escena 1

Matriz de confusión de la escena 1, se desarrolla la actividad humana en torno a la cocina, tenemos unos resultados alentadores, teniendo una matriz de confusión bastante ideal.

En la siguiente ilustración vemos la matriz de confusión obtenida en la escena 1:

Confusion Matrix Cross Validation																	
True Class	aspirador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	cisterna	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	conversacion	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0
	cubiertos_sartenes	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
	despertador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ducha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	extractor	0	0	0	0	0	79	0	0	0	0	0	0	0	0	0	0
	grifo_banio	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	grifo_cocina	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
	impresora	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	1
	microondas	0	0	0	0	0	0	0	1	0	9	0	0	0	0	0	0
	persiana	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	puerta	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
	telefono	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	timbre	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
	idle	0	0	0	2	0	0	0	0	0	2	0	2	0	0	0	91
		aspirador	cisterna	conversacion	cubiertos_sartenes	despertador	ducha	extractor	grifo_banio	grifo_cocina	impresora	microondas	persiana	puerta	telefono	timbre	idle
Predicted Class																	

Ilustración 4.6 Matriz de confusión de la escena 1



Se adjunta una tabla de resultados por categorías donde se puede ver con más detalle cuáles de los eventos producidos en la escena tienen mejor y peor resultados, donde podemos darnos cuenta que la puerta es un sonido que se tiene mayor dificultad para predecir, a continuación, los cubiertos y sartenes también tiene nuestro modelo dificultad para predecirlo. Si queremos ver los eventos que predicen mal en cada una de las categorías podemos ir a la matriz de confusión.

En estos resultados y mirando la matriz anterior podemos ver que los cubiertos y sartenes suelen confundirse con el evento “idle”.

Tabla de resultados de la evaluación escena 1:

N.º Categorías	Categoría	Total categoría	Total Aciertos	Porcentaje Acierto
1	timbre	3	3	100,00 %
2	puerta	5	3	60,00 %
3	conversación	20	20	100,00 %
4	cubiertos_sartenes	6	4	66,67 %
5	extractor	81	79	97,53 %
6	grifo_cocina	21	20	95,24 %
7	microondas	12	9	75,00 %
8	idle	92	91	98,91 %
TOTAL:		240	229	95,42 %

Tabla 4.8 Resultados evaluación escena 1

#### 4.3.2.2. Resultados para la escena 2

Por lo general estamos viendo que tenemos muy buenos resultados también en la escena 2, esta es desarrollada en el salón.

En la siguiente ilustración vemos la matriz de confusión obtenida en la escena 2:

		Confusion Matrix Cross Validation															
True Class	aspirador	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	cisterna	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	conversacion	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0
	cubiertos_sartenes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	despertador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ducha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	extractor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	grifo_banio	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	grifo_cocina	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	impresora	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
	microondas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	persiana	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0
	puerta	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
	telefono	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0
	timbre	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
	idle	0	0	0	0	0	0	0	0	0	0	0	2	0	0	80	0
	aspirador	cisterna	conversacion	cubiertos_sartenes	despertador	ducha	extractor	grifo_banio	grifo_cocina	impresora	microondas	persiana	puerta	telefono	timbre	idle	
		Predicted Class															

Ilustración 4.7 Matriz de confusión de la escena 2

En la tabla de resultados por categorías de la escena número dos se ha observado que tiene una predicción excelente, se puede resaltar que los falsos positivos del aspirador han ocurrido con los eventos cisterna e impresora, que pueden tener algún parecido en ventanas de audio de 3 segundos, ya que al estar acabando o empezando el evento puede haberse confundido. Aun así, se tiene una tasa de verdaderos positivos ideal. De los eventos estudiados por ahora se puede

observar que los modelos generados por la CNN son bastante buenos trabajando con imágenes.

Tabla de resultados de la evaluación escena 2:

N.º Categorías	Categoría	Total categoría	Total Aciertos	Porcentaje Acierto
1	timbre	4	4	100,00 %
2	puerta	5	3	60,00 %
3	conversación	20	20	100,00 %
4	aspirador	51	49	96,08 %
5	persiana	10	8	80,00 %
6	teléfono	10	10	100,00 %
7	idle	80	80	100,00 %
TOTAL:		180	174	96,67 %

Tabla 4.9 Resultados evaluación escena 2

#### 4.3.2.3. Resultados para la escena 3

Esta matriz de confusión muestra un dato interesante, la escena es desarrollada en un dormitorio, sin embargo, se cometen unos falsos positivos que tienen que ver con el evento conversación, se está confundiendo a veces con el evento de la ducha, aunque es un error muy pequeño. Esto podría evitarse si se aplicase algún algoritmo que filtre los falsos positivos, ya que en el dormitorio no debería dispararse nunca la predicción de ducha.

En la siguiente ilustración vemos la matriz de confusión obtenida en la escena 3:

Confusion Matrix Cross Validation																	
True Class	aspirador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	cisterna	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	conversacion	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	
	cubiertos_sartenes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	despertador	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	
	ducha	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	
	extractor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	grifo_banio	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	grifo_cocina	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	impresora	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	
	microondas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	persiana	0	0	1	0	0	0	0	0	0	0	11	0	0	0	0	
	puerta	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	
	telefono	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	timbre	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	
	idle	0	0	0	0	0	0	0	0	1	0	0	0	0	0	66	
		aspirador	cisterna	conversacion	cubiertos_sartenes	despertador	ducha	extractor	grifo_banio	grifo_cocina	impresora	microondas	persiana	puerta	telefono	timbre	idle
Predicted Class																	

Ilustración 4.8 Matriz de confusión de la escena 3

Por lo general sobre estos datos obtenidos se puede observar que los modelos en cada una de las escenas estudiadas por ahora, están resultando muy eficaces, suelen tener un error muy pequeño y la tasa de acierto afianza que el método seleccionado para recoger sonido ambiental y reconocer eventos está siendo bastante robusto.

Tabla de resultados de la evaluación escena 3:

N.º Categorías	Categoría	Total categoría	Total Aciertos	Porcentaje Acierto
1	timbre	3	3	100,00 %
2	puerta	4	4	100,00 %
3	conversación	20	16	80,00 %
4	despertador	41	41	100,00 %
5	persiana	11	11	100,00 %
6	impresora	15	14	93,33 %
7	idle	66	66	100,00 %
TOTAL:		160	155	96,88 %

Tabla 4.10 Resultados evaluación escena 3

#### 4.3.2.4. Resultados para la escena 4

En la siguiente ilustración vemos la matriz de confusión obtenida en la escena 4:

Confusion Matrix Cross Validation																
True Class	aspirador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	cisterna	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	conversacion	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	cubiertos_sartenes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	despertador	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ducha	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0
	extractor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	grifo_banio	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
	grifo_cocina	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	impresora	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	microondas	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
	persiana	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	puerta	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0
	telefono	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	timbre	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	idle	0	0	0	0	0	1	0	1	0	0	0	1	0	0	55
	aspirador	cisterna	conversacion	cubiertos_sartenes	despertador	ducha	extractor	grifo_banio	grifo_cocina	impresora	microondas	persiana	puerta	telefono	timbre	idle
	Predicted Class															

Ilustración 4.9 Matriz de confusión de la escena 4

En esta escena se estudia la actividad humana dentro de un cuarto de baño, es interesante ver cómo el modelo puede distinguir de donde proviene el agua (baño, grifo del baño y cisterna del inodoro) es muy interesante observar esta escena y la escena que ocurre la actividad en la cocina, ya que se diferencia perfectamente el grifo del baño con respecto el grifo de la cocina.

Tabla de resultados de la evaluación escena 4:

N.º Categorías	Categoría	Total categoría	Total Aciertos	Porcentaje Acierto
1	puerta	13	12	92,31 %
2	grifo_baño	21	20	95,24 %
3	ducha	63	59	93,65 %
4	cisterna	8	7	87,50 %
5	idle	55	55	100,00 %
TOTAL:		160	153	95,63 %

Tabla 4.11 Resultados evaluación escena 4

#### 4.3.3. Tiempo de ejecución en Raspberry

Tabla de Parámetros entrenables, tiempo de aprendizaje, millones de instrucciones (MI) y tiempo de evaluación:

	Parámetros entrenables	Tiempo de aprender	Millones de instrucciones (MI)	Tiempo de evaluación
<b>Modelo CNN + MFCC</b>	1,7M	96 min	230,4 x 103 MI	2,53 segundos

Tabla 4.12 Tiempo evaluación de validación cruzada en RaspBerry

## 5. Conclusiones y líneas de trabajo futuras

### 5.1. Conclusiones

A medida que el trabajo fin de grado desarrollándose se ha desarrollado, he tomado consciencia de que es una oportunidad increíble para aprender cosas nuevas. Después de desarrollar y obtener conocimiento de cómo afrontar el problema que se plantea he descubierto en base a los resultados obtenidos que las redes neuronales convolucionales son un método útil para clasificar audios desde dispositivos ambientales. El hecho de plantear que los sonidos se podrían convertir en imágenes y hacer las pruebas con esta idea representa una técnica interesante e

ingeniosa que combina diferentes aspectos de la ingeniería y de la inteligencia artificial. El aprendizaje de estos métodos ha sido posible gracias a los tutores que tengo asignados a este TFG, que me han asesorado muy bien para abordar este problema.

Se puede concluir que los sonidos transformados a MFCC con una CNN generan modelos de predicción muy buenos con sonidos al menos en el enfoque doméstico que es donde se desarrolla la recolección de audios y grabación de escenas naturalistas.

También me gustaría sacar como conclusión que gracias a tener ciertos sensores conectados a una placa inteligente como una raspberry pi se puede predecir ciertos eventos que si no fuera porque hemos generado un modelo MFCC + CNN hubiéramos necesitado muchos sensores para detectar cada uno de los eventos que aspiramos a reconocer. Por tanto, estamos ahorrando un coste económico y simplificando la tarea de reconocimiento de sonido ambiental.

Después de haber desarrollado este estudio he comprobado el difícil proceso de recolección de audio, ya que debe establecerse una ventana de tiempo en la que se captura el evento de interés para nosotros, estos audios que capturamos deben ser claros y que no empiecen y terminen de la misma forma dando una diversidad mayor del evento que queremos clasificar.

Por último, el etiquetado de audio es una tarea que en este estudio se ha automatizado en lo posible, aun así, se encuentra algún problema en este proceso. Por ej. Cuando se recolecta los audios de un evento en concreto, el problema del etiquetado es más fácil ya que podemos nombrar a los ficheros con la categoría a la que pertenece, de esta manera se puede automatizar con un script la generación de un fichero CSV. Pero cuando los audios provienen de una escena con una duración mayor donde intervienen varios eventos, en este caso hay que escuchar los audios una vez segmentada la escena y el proceso de etiquetado se convierte en una tarea más tediosa y difícil de automatizar.



## 5.2. Líneas futuras

En este apartado me gustaría indicar muchas de las aplicaciones que tiene este proyecto que se pueden abordar en el futuro, pero por tiempo y los objetivos que se han marcado no se han desarrollado y se quedan en el aire por si alguien quiere retomarlos.

Se puede desarrollar una aplicación móvil o web donde la casa está monitorizada en tiempo real. Y se me ocurren varias vertientes que puede tomar esta idea:

- **Domótica:** tales como ahorrar costes, complementar los datos obtenidos por el modelo desarrollado con otros sensores para disparar eventos que hagan la vida más fácil del habitante.
- **Seguridad:** Es interesante poder utilizar este proyecto en el ámbito de la seguridad, imaginemos que salimos de casa o nos vamos de vacaciones y tenemos una aplicación que nos notifique de si ha entrado un intruso, detectándolo porque se escucha alguna conversación, se abren o cierran puertas...
- **Monitorización:** Es también un tema de interés el poder monitorizar la vida diaria de una persona, pues si sigue ampliando este proyecto podremos mantener a nuestros ancianos con cierta dependencia controlados en caso de que pase algo grave y poder mandar una respuesta temprana para ayudarlos.

Otro de los enfoques podría ser utilizar el dataSet existente para investigar alguna técnica que mejore los modelos generados en este estudio, permitiendo mejorar los resultados, se podría incorporar al proyecto lógica difusa para que los valores que oscilen entre una predicción u otra se resuelva efectivamente, aumentando así la tasa de acierto de nuestros modelos.

Investigar acerca de la generación de datos y etiquetado automático sería un buen punto a incorporar a este proyecto, anteriormente dije en las conclusiones que el intentar automatizar en lo mayor posible este proceso es un tema interesante que

ayudaría a que el proyecto pueda ampliarse con un coste humano de menor implicación.










Como he dicho anteriormente y para concluir este apartado, este proyecto puede ampliarse en varias vertientes, incluso fuera del hogar y destinar el conocimiento aquí plasmado para desarrollar otro tipo de proyectos que utilicen esta mecánica para resolver problemas complejos. Una opción puede ser ayudarse de más sensores para enriquecer el conocimiento obtenido del entorno, otra sin embargo puede ser ampliando el dataSet e incluir más categorías al proyecto para que siga detectando muchas más interacciones que suceden dentro o fuera del hogar.

### Publicación en MDPI

Me gustaría mencionar que parte del estudio que se ha hecho en este trabajo fin de grado ha sido utilizado para el desarrollo de un artículo y publicado en Multidisciplinary Digital Publishing Institute en el apartado de ciencias aplicadas el 29 de Julio de 2021, en el cual se ha proporcionado modelos, DataSets de eventos domésticos, escenas etiquetas y datos de evaluación. Ha sido un placer contar con los compañeros con los que se ha contado para la realización del artículo y con su experiencia en el campo.

Open Access
Article

## Ambient Sound Recognition of Daily Events by Means of Convolutional Neural Networks and Fuzzy Temporal Restrictions

by  Aurora Polo-Rodriguez <sup>1,\*</sup> ,  Jose Manuel Vilchez Chiachio <sup>1</sup> ,  Cristiano Paggetti <sup>2</sup>  and  Javier Medina-Quero <sup>1</sup>  

<sup>1</sup> Department of Computer Science, Campus Las Lagunillas, 23071 Jaén, Spain  
<sup>2</sup> I + Srl, Piazza G.Puccini, 26, 50144 Firenze, Italy  
\* Author to whom correspondence should be addressed.

Academic Editors: Sang-Hyun Lee and Seongsoo Cho

*Appl. Sci.* **2021**, *11*(15), 6978; <https://doi.org/10.3390/app11156978>

Received: 30 April 2021 / Revised: 22 July 2021 / Accepted: 23 July 2021 / Published: 29 July 2021

(This article belongs to the Special Issue [The Development and Application of Fuzzy Logic](#))

View Full-Text
Download PDF
Browse Figures
Citation Export

Ilustración 5.1 Publicación de artículo sobre este TFG

## Anexos

### Scripts

#### Recolección de audio

Este script desarrollado en python nos sirve para grabar el sonido ambiente donde se encuentra el dispositivo IoT

```
#!/usr/bin/python3
import sys
import sounddevice as sd #librería para poder trabajar con audio
import time #librería para el tiempo
import pandas as pd
import numpy as np
import scipy.io.wavfile as waves

fs=44100 #frecuencia de muestreo
tiempoGrabacion=3

def grabar():
    grabacion = sd.rec(int(tiempoGrabacion * fs),samplerate=fs, channels=1)#grabacion
    return grabacion

def reproducir(senal):
    sd.play(senal, fs)

def guardar(nombre, senial):
    titulo=nombre+'.wav'
    waves.write("sonidos/"+titulo, fs, senial)

def cargar(nombre):
    titulo=nombre+'.wav'
    muestreo, sonido = waves.read(titulo)
    return sonido

if len(sys.argv)>3:
    indiceFinal= int(sys.argv[3]) #indiceFinal
    indiceInicio = int(sys.argv[2]) #indiceInicio
    while indiceInicio <=indiceFinal:
        print("sonido: ",indiceInicio)
        print("Start record..")
        sonido=grabar()
        time.sleep(tiempoGrabacion)
        print("Finish record")
        guardar(str(sys.argv[1])+str(indiceInicio),sonido) #Nombre del fichero.wav
        print("Fichero guardado")
        indiceInicio+=1
    else:
```

```
print("Faltan argumentos.. (nombre, indiceInicio, indiceFin)")
```

## Etiquetado

Este script nos permite etiquetar los audios recopilados de nuestro DataSet principal y generar un fichero CSV:

```
#valor compuesto de índice y número de ficheros grabados de esa clase.
clasificacion = {
    "aspirador": [1, 100],
    "cisterna": [2, 100],
    "conversacion": [3, 100],
    "cubiertos_sartenes": [4, 100],
    "despertador": [5, 100],
    "ducha": [6, 100],
    "extractor": [7, 100],
    "grifo_banio": [8, 100],
    "grifo_cocina": [9, 100],
    "impresora": [10, 100],
    "microondas": [11, 100],
    "persiana": [12, 100],
    "puerta": [13, 100],
    "telefono": [14, 100],
    "timbre": [15, 100],
    "idle": [16, 238]
}
ficheroCSV = "name,target,category"
for name in clasificacion:
    for i in range(clasificacion[name][1]):
        ficheroCSV += "\n" + name + str(i+1) + ".wav", '+' + str(clasificacion[name][0]) + ',' + name + ""
#print(ficheroCSV)

with open('dataSetEscena4.csv', 'w') as f:
    f.write(ficheroCSV)

print("fichero CSV generado exitosamente.")
```

También en este apartado adjuntaré el código fuente del etiquetado de las escenas, a diferencia del otro este nos permite reconstruir la secuencia tal como fue grabada:

```
secuencia = [("puerta", 13, 4), ("idle", 16, 12), ("grifo_banio", 13, 21), ("idle", 16, 12), ("ducha", 13, 63),
              ("idle", 16, 7), ("puerta", 13, 5), ("idle", 16, 11), ("cisterna", 13, 8), ("idle", 16, 9), ("puerta", 13, 4), ("idle", 16, 4)]
ficheroCSV = "name,target,category"
contador = 1
for i in range(len(secuencia)):
    for j in range(secuencia[i][2]):
        ficheroCSV += "\n" + secuencia[i][0] + str(contador) + ".wav", '+' + str(secuencia[i][1]) + ',' + secuencia[i][0] + ""
        contador = contador + 1
```

```
#print(ficheroCSV)
with open('escena4_etiquetada.csv', 'w') as f:
    f.write(ficheroCSV)

print("fichero CSV generado exitosamente.")
```

## Segmentación de audio

Este script nos permite segmentar las escenas que hemos grabado en este trabajo fin de grado para obtener los audios de 3 segundos de duración:

```
from pydub import AudioSegment

file_name = r"escena1.wav"
sound = AudioSegment.from_wav(file_name)
for i in range(100):
    start_time = (i*3)*1000
    stop_time = ((i*3)+3)*1000

    print("time:", start_time, "~", stop_time)

    print("ms:", start_time, "~", stop_time)

    word = sound[start_time:stop_time]
    # guardar ruta
    save_name = r"part"+ str(i+1) + file_name[-4:]
    print(save_name)

    word.export(save_name, format="wav", tags={'artist': 'jmvc0010', 'album': save_name[:-4]})
```

## Despliegue del sistema

En esta sección se indicará los pasos a seguir para que cualquier persona que tenga los dispositivos que se usan en este estudio puedan desplegar el sistema y ver su funcionamiento, ya sea para generar modelos nuevos o para hacer una predicción en tiempo real.

1. Generar modelos: En primer lugar habría que generar o ampliar un dataSet de audios que tenemos que capturar con el micrófono conectado en la RaspBerry, esto es posible usando el script de grabarMicrofono.py, donde su funcionamiento viene explicado en el caso de uso de recolección de audio.

2. Debemos de etiquetar dichos audios para que el sistema pueda entrenar con ellos, por tanto, haremos uso de la script generarCSV.py, donde su funcionamiento viene explicado en el caso de uso de etiquetado de audios.
3. Una vez tenemos los datos etiquetados haremos uso del entorno de desarrollo jupyter notebook, que usaremos para abrir los ficheros con extensión “.ipynb” en concreto el fichero llamado “EscenasDomesticas.ipynb” ejecutaremos sección por sección hasta llegar a la número 7 en la que encontramos el código de entrenamiento y generación del modelo para guardarlo en el dispositivo local (*model.save(name\_model.h5)*).

Es en este punto donde ya hemos generado un modelo de predicción con el DataSet generado nuevo.

### Predicción en tiempo real

En caso de querer usar los modelos generados en este estudio o en caso de haber generado alguno nuevo, disponemos de la opción de usarlo para predecir en tiempo real los sonidos recolectados por el micrófono.

Con una duración de grabación de 3 segundos se va enviando en tiempo real al modelo los audios generados, es este el que se encarga de predecir a que categoría pertenece estos. Con una media de predicción de 2,53 segundos, no se genera ningún delay, por tanto, la predicción se va generando a la par que se graba la siguiente.

A continuación se muestra el código de la aplicación encargado de esto que hemos expuesto anteriormente:

```
import sounddevice as sd
from scipy.io.wavfile import write
from IPython.display import clear_output
fs = 44100 # Sample rate
seconds = 3 # Duration of recording
new_model = models.load_model('my_model.h5')
def getLabel(i):
```

```

    if i==0:
        return "aspirador"
    if i==1:
        return "cisterna"
    if i==2:
        return "conversacion"
    if i==3:
        return "cubiertos_sartenes"
    if i==4:
        return "despertador"
    if i==5:
        return "ducha"
    if i==6:
        return "extractor"
    if i==7:
        return "grifo_baño"
    if i==8:
        return "grifo_cocina"
    if i==9:
        return "impresora"
    if i==10:
        return "microondas"
    if i==11:
        return "persiana"
    if i==12:
        return "puerta"
    if i==13:
        return "telefono"
    return "timbre"

for x in range(40):
    myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=2)
    sd.wait() # Wait until recording is finished
    write('temp.wav', fs, myrecording) # Save as WAV file

    sig_temp , sr = librosa.load('temp.wav')
    mfcc_temp = librosa.feature.mfcc(sig_temp , sr=sr, n_mfcc=13)
    X_temp=[]
    X_temp.append(mfcc_temp)
    X_temp = np.array(X_temp)
    X_temp = X_temp.reshape(X_temp.shape[0], X_temp.shape[1], X_temp.shape[2], 1)
    Y_temp=new_model.predict(X_temp)
    getLabel(np.argmax(Y_temp))

```

## Bibliografía

- [1] Mondragón, F. J., Pérez-Meana, H. M., Calderón, G., & Jiménez, J. (2021). Clasificación de sonidos ambientales usando la transformada wavelet continua y redes neuronales convolucionales. *Información tecnológica*, 32(2), 61-78.
- [2] Rossi, M., Feese, S., Amft, O., Braune, N., Martis, S., & Tröster, G. (2013, March). AmbientSense: A real-time ambient sound recognition system for smartphones. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)* (pp. 230-235). IEEE.
- [3] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). Ieee.
- [4] Owens, A., Wu, J., McDermott, J. H., Freeman, W. T., & Torralba, A. (2016, October). Ambient sound provides supervision for visual learning. In *European conference on computer vision* (pp. 801-816). Springer, Cham.
- [5] Rashidi, P.; Mihailidis, A. *A survey on ambient-assisted living tools for older adults. IEEE 232 journal of biomedical and health informatics* 2012, 17, 579–590
- [6] Madakam, S., Lake, V., Lake, V., & Lake, V. (2015). Internet of Things (IoT): A literature review. *Journal of Computer and Communications*, 3(05), 164.
- [7] Purohit, H., Tanabe, R., Ichige, K., Endo, T., Nikaido, Y., Suefusa, K., & Kawaguchi, Y. (2019). MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. *arXiv preprint arXiv:1909.09347*.
- [8] Cruz-Sandoval, D.; Beltran-Marquez, J.; Garcia-Constantino, M.; Gonzalez-Jasso, L.A.; Favela, J.; Lopez-Nava, I.H.; Cleland, I.; Ennis, A.; Hernandez-Cruz, N.; Rafferty, J.; others. *Semi automated data labeling for activity recognition in pervasive healthcare. Sensors* 2019, 19, 3035
- [9] Medina-Quero, J.; Zhang, S.; Nugent, C.; Espinilla, M. *Ensemble classifier of long short-term memory with fuzzy temporal windows on binary sensors for activity recognition. Expert Systems with Applications* 2018, 114, 441–453.
- [13] Weiser, M. *The Computer for the Twenty-First Century Scientific American*. September Elsevier Ltd 1991.
- [14] Van Kasteren, T.; Englebienne, G.; Kröse, B.J. *An activity monitoring system for elderly care using generative and discriminative models. Personal and ubiquitous computing* 2010, 14, 489–498.
- [15] Ordóñez, F.J.; de Toledo, P.; Sanchis, A. *Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors* 2013, 13, 5460–5477.
- [16] Ann, O.C.; Theng, L.B. *Human activity recognition: a review. 2014 IEEE international conference on control system, computing and engineering (ICCSCE 2014). IEEE, 2014, pp. 389–393.*
- [17] Laput, G.; Zhang, Y.; Harrison, C. *Synthetic sensors: Towards general-purpose sensing. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, pp. 3986–3999*
- [18] Shi, W.; Dustdar, S. *The promise of edge computing. Computer* 2016, 49, 78–81.
- [19] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. *Fog computing and its role in the internet of things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13–16.*



- [20] Kortuem, G.; Kawsar, F.; Sundramoorthy, V.; Fitton, D. Smart objects as building blocks for the internet of things. *IEEE Internet Computing* 2009, 14, 44–51.
- [21] Ordóñez, F.; De Toledo, P.; Sanchis, A.; others. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors* 2013, 13, 5460–5477
- [22] Lopez Medina, M.A.; Espinilla, M.; Paggeti, C.; Medina Quero, J. Activity recognition for iot devices using fuzzy spatio-temporal features as environmental sensor fusion. *Sensors* 2019, 19, 3512.
- [23] Wang, J.; Chen, Y.; Hao, S.; Peng, X.; Hu, L. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 2019, 119, 3–11
- [24] Quero, J.M.; Burns, M.; Razzaq, M.A.; Nugent, C.; Espinilla, M. Detection of falls from non-invasive thermal vision sensors using convolutional neural networks. *Multidisciplinary Digital Publishing Institute Proceedings*, 2018, Vol. 2, p. 1236.
- [25] Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.
- [26] Wyse, L. Audio spectrogram representations for processing with convolutional neural networks. *arXiv preprint arXiv:1706.09559* 2017.
- [27] Salamon, J.; Bello, J.P. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters* 2017, 24, 279–283.
- [28] Kim, J. URBAN SOUND TAGGING USING MULTI-CHANNEL AUDIO FEATURE WITH CONVOLUTIONAL NEURAL NETWORKS. *Detection and Classification of Acoustic Scenes and Events* 2019.
- [29] Lasseck, M. Acoustic bird detection with deep convolutional neural networks. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, 2018, pp. 143–147.
- [30] Choi, K.; Fazekas, G.; Sandler, M. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298* 2016.
- [31] Pons, J.; Slizovskaia, O.; Gong, R.; Gómez, E.; Serra, X. Timbre analysis of music audio signals with convolutional neural networks. *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 2744–2748.
- [32] Beltrán, J.; Chávez, E.; Favela, J. Scalable identification of mixed environmental sounds, recorded from heterogeneous sources. *Pattern Recognition Letters* 2015, 68, 153–160.
- [33] Beltrán, J.; Navarro, R.; Chávez, E.; Favela, J.; Soto-Mendoza, V.; Ibarra, C. Recognition of audible disruptive behavior from people with dementia. *Personal and Ubiquitous Computing* 2019, 23, 145–157.
- [34] Laput, G.; Ahuja, K.; Goel, M.; Harrison, C. Ubicoustics: Plug-and-play acoustic activity recognition. *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 213–224.
- [35] Xie, C., Cao, X., & He, L. (2012). Algorithm of abnormal audio recognition based on improved MFCC. *Procedia Engineering*, 29, 731-737.
- [36] Chu, S., Narayanan, S., & Kuo, C. C. J. (2009). Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6), 1142-1158.
- [37] Valores separados por comas. (2021, 3 de mayo). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 20:17, septiembre 7, 2021

desde [https://es.wikipedia.org/w/index.php?title=Valores\\_separados\\_por\\_comas&oldid=135266794](https://es.wikipedia.org/w/index.php?title=Valores_separados_por_comas&oldid=135266794).

[38] (2021). Recuperado 11 de marzo de 2021, de <https://www.goanywhere.com/es/mft/mas/tutoriales-de-goanywhere/como-consultar-base-de-datos-y-escribir-resultado-en-json>

[39] Extensible Markup Language. (2021, 10 de junio). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 20:30, septiembre 7, 2021 desde [https://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&oldid=136233446](https://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&oldid=136233446).

[40] Calvo, D. (2018, 8 diciembre). Red Neuronal Convolucional CNN. Diego Calvo. <https://www.diegocalvo.es/red-neuronal-convolucional/>

[41] Morán, N., Pérez, J., & Rodríguez, W. (2018). Reconocimiento de estados emocionales de personas mediante la voz utilizando algoritmos de aprendizaje de máquina. *Revista Venezolana de Computación*, 5(2), 41-52.

[42] Machuca, F. (2021, 21 mayo). ¿Qué es Python? El lenguaje de programación más popular para aprender en 2021. <https://www.crehana.com>. <https://www.crehana.com/es/blog/desarrollo-web/que-es-python/>

[43] Siddique, F., Sakib, S., & Siddique, M. A. B. (2019, September). Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers. In 2019 5th International Conference on Advances in Electrical Engineering (ICAEE) (pp. 541-546). IEEE.

[50] Salomons, E. L., Havinga, P. J., & Van Leeuwen, H. (2016). Inferring human activity recognition with ambient sound on wireless sensor nodes. *Sensors*, 16(10), 1586.

[51] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7), 1645-1660.

[52] Malche, T., & Maheshwary, P. (2017, February). Internet of Things (IoT) for building smart home system. In 2017 International conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC) (pp. 65-70). IEEE.

[53] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*.

[54] Q. Zhang, D. Zhou, and X. Zeng, "HeartID: A Multiresolution Convolutional Neural Network for ECG-based Biometric Human Identification in Smart Health Applications," IEEE Access, pp. 1-1.

[55] Boddapati, V., Petef, A., Rasmusson, J., & Lundberg, L. (2017). Classifying environmental sounds using image recognition networks. *Procedia computer science*, 112, 2048-2056.

[56] M. M. Mostafa and N. Billor, "Recognition of Western style musical genres using machine learning techniques," in Expert Systems with Applications, vol. 36, no. 8, pp. 11378–11389, Oct. 2009.

[57] Z. Zhang and B. Schuller, "Semi-supervised learning helps in sound event classification," in Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, 2012, pp. 333–336

- [58] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, "Robust Sound Event Classification Using Deep Neural Networks," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 540–552, Mar. 2015.
- [59] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 559–563.
- [60] S. Chachada and C.-C. Jay Kuo, "Environmental Sound Recognition: A Survey," in *Proceedings of the 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2013.
- [61] Mall, R. (2018). *Fundamentals of software engineering*. PHI Learning Pvt. Ltd.
- [62] Greenspan, S. J., Mylopoulos, J., & Borgida, A. (1982, September). Capturing more world knowledge in the requirements specification. In *Proceedings of the 6th international conference on Software engineering* (pp. 225-234).
- [63] Moffat, D., Ronan, D., & Reiss, J. D. (2015). An evaluation of audio feature extraction toolboxes.
- [64] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., & Vrgoč, D. (2016, April). Foundations of JSON schema. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 263-273).
- [65] Manaswi, N. K. (2018). *Deep learning with applications using python: chatbots and face, object, and speech recognition with tensorflow and keras*. Apress.
- [66] Rodriguez, J. D., Perez, A., & Lozano, J. A. (2009). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence*, 32(3), 569-575.