Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGUIMIENTO PARA EL CONTROL DE AVES

Alumno: **Jesús Soriano Martínez**

Tutores: Damián Martínez Muñoz y
Manuel Ángel Gadeo Martos

Depto.: Ingeniería de Telecomunicación

Febrero, 2023

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGUIMIENTO PARA EL CONTROL DE AVES

Alumno: Jesús Soriano Martínez

Tutor: Damián Martínez Muñoz
Manuel Ángel Gadeo Martos

Depto.: Ingeniería de Telecomunicación

FEBRERO,2023

Firma del autor

SORIANO MARTINEZ JESUS - 45498592S

Firmado digitalmente porSORIANO MARTINEZ JESUS - 45498592S
DN: cn=SORIANO MARTINEZ JESUS - 45498592S gn=JESUS c=ES
Motivo:Soy el autor de este documento
Ubicaci- n:
Fecha:2023-02-16 20:37+01:00

Firma de los tutores

GADEO MARTOS MANUEL ANGEL - 25987883E

Firmado digitalmente por GADEO MARTOS MANUEL ANGEL - 25987883E
Fecha: 2023.02.16 15:11:51 +01'00'

MARTINEZ MUÑOZ DAMIAN - 26202116X

Firmado digitalmente por MARTINEZ MUÑOZ DAMIAN - 26202116X
Fecha: 2023.02.16 13:37:30 +01'00'

ÍNDICE

ÍNDICE DE TABLAS:

ÍNDICE DE ILUSTRACIONES:

1. INTRODUCCIÓN

1.1. Antecedentes

Para conocer el estado de las poblaciones de aves, se llevan a cabo **programas de monitorización,** que proporcionan una gran fuente de información. Los programas efectivos de seguimiento pueden identificar a las especies que están en riesgo, ayudar a comprender como las acciones de gestión afectan a las poblaciones, evaluar las respuestas de la población a modificaciones en los paisajes y el clima, y proporcionar información básica sobre la distribución de las diferentes especies.[3,4]

En la actualidad, existe una gran necesidad de programas de seguimiento y control de aves dado que hay un alarmante descenso en sus poblaciones y un aumento en la degradación y pérdida de sus hábitats. [3,4]

1.2. Resumen

En este proyecto se pretende crear una red de sensores de bajo coste que sean capaces de procesar señales acústicas provenientes de las aves. Hacer una clasificación de las aves para posteriormente ser capaz de identificarlas, poder hacer un seguimiento de las aves en lugares específicos donde se instalen los nodos de nuestra red. Se desea tener una información en tiempo real por lo que las aves identificadas serán almacenadas en una Base de datos (BBDD) con los parámetros, hora, día y nombre de la especie y se podrán visualizar esto datos registrados en una aplicación web en tiempo real.

Para ello, primero se han analizado tramas (secciones de audio de un tamaño reducido de la orden de 30milisegundos) de las que se obtienen las frecuencias a las que emiten sonido las distintas aves, habiéndose seleccionado palomas, gaviotas, lechuzas, gorriones, carboneros y vencejo para el propósito de estudio. Con sus respectivos audios se ha elaborado una biblioteca de muestras que se ha utilizarlo en el diseño de un algoritmo en Matlab. Este algoritmo ha permitido la extracción de los parámetros necesarios para configurar un dispositivo de bajo coste (Arduino DUE) y con él obtener la identificación fidedigna de cada ave. Para ello, se colocarían sensores en las áreas de estudio, los cuales registrarían las señales acústicas procedentes de las aves. Estas señales pasarían a analizarse en un nodo de la red, el cual identificaría el ave detectada. Una vez detectada e identificada el ave necesitaremos enviar dicha identificación a la BBDD por lo que es necesaria una conexión inalámbrica que conseguiremos conectando un módulo Wifi al

Arduino. Una vez establecida la conexión Wifi, mediante un protocolo de comunicación se envía la identificación a una BBDD donde quedara registrada con los atributos de fecha hora y especie. Para visualizar estos registros de las aves identificadas se creará una web con distintos filtros para observar la información deseada por los usuarios

1.3. Objetivos

La biodiversidad es un aspecto clave en la conservación de nuestro planeta y las aves juegan un papel fundamental en este ecosistema. Por esta razón, la identificación y seguimiento de las aves es una tarea importante para estudiar y preservar la fauna de una determinada área geográfica. En este trabajo de fin de grado se propone la implementación de un sistema distribuido de bajo coste que permita la identificación y seguimiento de las aves que habitan en un área geográfica determinada.

Para lograr este objetivo, se realizará un análisis que permita caracterizar el sonido de las aves más comunes en la zona de estudio y se propondrá un algoritmo que permita identificarlas. Es importante que la propuesta de este algoritmo tenga en cuenta las limitaciones computacionales del hardware donde finalmente se instalará.

El sistema final permitirá la consulta vía web de información relativa al registro de las aves identificadas por los diferentes dispositivos de bajo coste desplegados en la zona de estudio. Con esta iniciativa, se espera contribuir a la conservación de las aves en la zona de estudio y aportar información valiosa para futuros estudios y proyectos de conservación en la región.

A continuación, para tener una visión global del proyecto se crean los diagramas con lo explicado en los puntos anteriores y llevar a término los objetivos:

Diagrama genérico del objetivo:

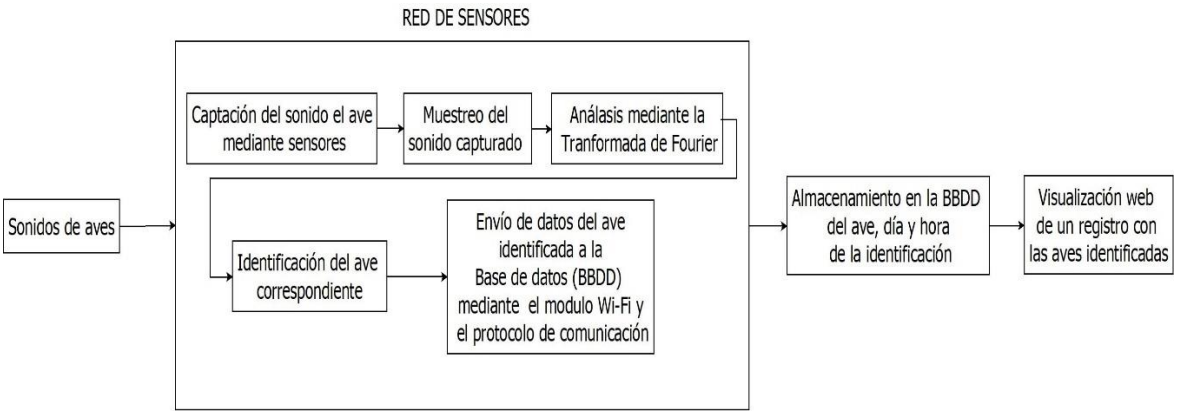


Ilustración 1 Diagrama global de la solución

Diagrama genérico del algoritmo de identificación y envio:

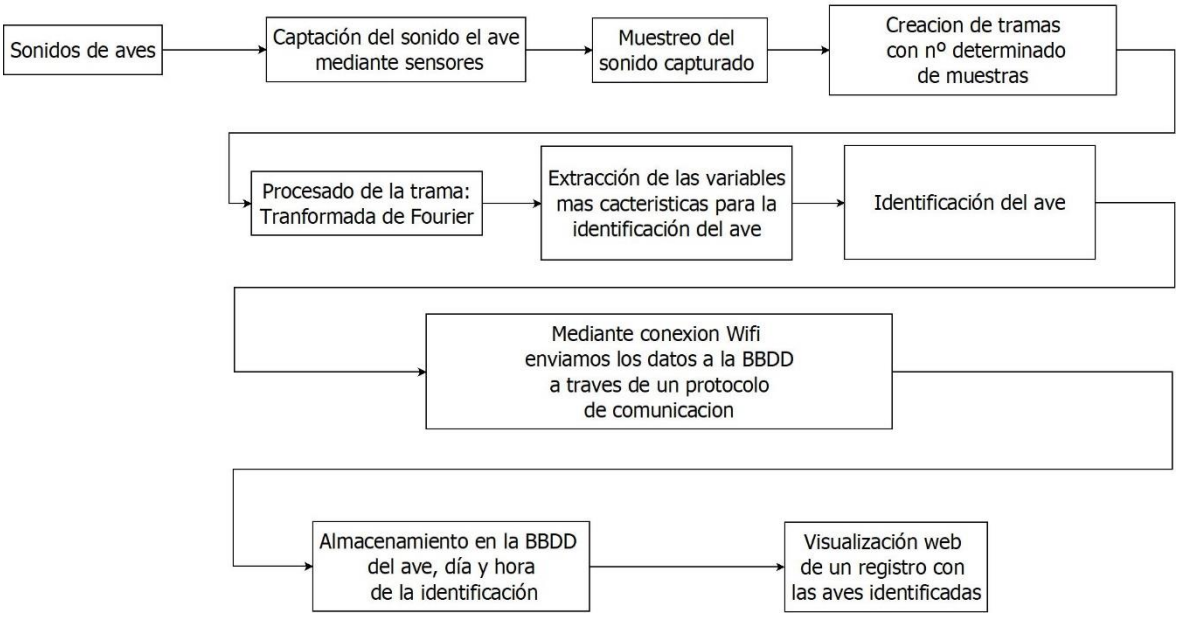


Ilustración 2 Diagrama genérico del algoritmo

2. ESTADO DEL ARTE

Para identificar las mejores opciones a utilizar en este proyecto, se estudian las características de distintas alternativas. Por lo que, en esta sección se detallan:

- Los **dispositivos de hardware** que se han estudiado para la realización de este trabajo, habiéndose seleccionado para la realización del mismo el hardware Arduino DUE por los motivos que se detallan más adelante.
- Los **software y lenguajes** de programación necesarios para este trabajo. Se utilizará MySQL, HTML y PHP.
- La **caracterización** de los sonidos de las aves para su posterior identificación con la solución propuesta.

2.1. Hardware: SBCs y placas con microcontroladores

Para el desarrollo de este proyecto, primero se estudia si se necesita un ordenador de placa única (del inglés single-board computer, SBC) o una placa con microcontrolador:

Un **SBC** es un miniordenador construido en una placa con un solo circuito, que incluye un microprocesador, la memoria, el almacenamiento y las entradas/salidas (E/S). Es capaz de realizar varias tareas y ejecutar varios programas a la vez. Puede utilizar distintos sistemas operativos. Los SBCs se suelen usar para aplicaciones donde el espacio está limitado o un ordenador de tamaño normal sería demasiado caro o innecesario. Uno de los SBCs más utilizados es la Raspberry Pi, que se explica más adelante. [5–10]

Respecto a las **placas con microcontroladores**, estas pueden realizar menos funciones que un SBC, pero se pueden programar para pequeñas aplicaciones, tareas sencillas que no requieran una programación muy compleja, o para realizar una misma tarea simple de manera repetida. Para ello, no necesitan utilizar ningún sistema operativo, por lo que son más rápidas para funciones básicas y consumen menos energía. Además, son más fáciles y rápidas de programar. Relativamente fáciles de usar para principiantes y flexibles para usuarios más avanzados. Un ejemplo de placas con microcontroladores son las de Arduino. [5–10]

Características	Microprocesador	Microcontrolador
Definición	Es la unidad central de procesamiento (CPU), realiza las tareas, cálculos y procesamiento de datos. No dispone de entradas y salidas por lo que requiere de periféricos adicionales para funcionar.	Integra una CPU, memoria y periféricos de entrada y salida en un solo chip. Es un elemento funcional completo, programable, que ejecuta las operaciones grabadas en su memoria.
CPU	Mayor potencia de cálculo	Menor potencia
Velocidad de reloj	Rápido en realizar operaciones digitales (GHz)	Más lento, realiza menos instrucciones por segundo (kHz-Mhz)
Memoria RAM y ROM	Se añaden como dispositivos externos que complementan el funcionamiento del microprocesador por lo que tienen mayor capacidad	Están incluidas en el mismo circuito integrado por lo que sus capacidades son menores
Uso	Tareas que requieren una gran capacidad de computación, como parte de un sistema que controla otros periféricos.	Se utilizan para tareas puntuales. No son potentes ni fácilmente reprogramables.
Consumo de energía	Mayor	Menor
Coste	Mayor	Menor
Ejemplos	Intel o AMD	ATMEGA de Atmel

Tabla 1 Características de los microcontroladores[5–10]

2.1.1. Raspberry Pi

La Raspberry Pi, es un SBCs de bajo coste y formato compacto, que fue creada en 2012 en Reino Unido por la Raspberry Pi Foundation con el objetivo de promover los conocimientos de computación en colegios y universidades. [11,12]

Es capaz de realizar la mayoría de las tareas típicas de un ordenador de escritorio, como navegar en internet, reproducir vídeos, modificar documentos y reproducir juegos.

Su uso se extendió rápidamente por su bajo coste, versatilidad y al poder ejecutar el sistema operativo Linux de software libre. Además, es compatible con muchos programas y lenguajes de programación, incluyendo C++ y Python. [11,12]

Existen diferentes modelos de Raspberry Pi pero son prácticamente versiones actualizadas del mismo hardware. La más popular es la Raspberry Pi 3 B+[11,12]:

La **Raspberry Pi 3 B+,** comercializada en 2018, dispone de un procesador quad-core de 64 bits con 1,4 GHz (Broadcom BCM2837B0), red inalámbrica de banda dual (2.4/5 GHz), Bluetooth (4.2 / BLE) y Ethernet de alta velocidad (300Mbps y capacidad PoE a través de un PoE HAT separado). [11,12]

La LAN inalámbrica de doble banda tiene certificación de cumplimiento modular, permitiendo que las pruebas de cumplimiento de LAN inalámbrica para productos finales sean más cortas, reduciendo el coste y el tiempo de comercialización. [11,12]

También, cuenta con: 40 pines GPIO que funcionan a 3.3V, puertos de comunicación I2C, SPI, UART, 4 puertos USB, Jack de 3.5mm, puerto HDMI, puerto para memoria microSD, conector micro-usb para la alimentación, memoria de1GB LPDDR2 SDRAM, puerto de visualización MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de video compuesto. Sus dimensiones: 85 x 56 x 17 mm con un peso de 50g. [11,12]

Ilustración 3 Raspberry Pi 3B+. Imagen adaptada de raspberrypi.cl[13]

2.1.2. ESP-32

El ESP32 es un microcontrolador de bajo coste y bajo consumo de energía desarrollado por Espressif Systems. Es un dispositivo completo que combina Wi-Fi, Bluetooth y numerosos periféricos en un solo módulo compacto. [14–16]

Está equipado con un procesador dual-core basado en Xtensa LX6. Cada núcleo puede funcionar a frecuencias diferentes, lo que permite aumentar la eficiencia y la flexibilidad del dispositivo. [14–16]

Una de las características más importantes del ESP32 es su capacidad para conectarse a redes Wi-Fi y dispositivos Bluetooth. Esto permite a los desarrolladores crear aplicaciones IoT que se comuniquen con otros dispositivos a través de Internet. [14–16]

Cuenta con hasta 4 MB de memoria flash integrada, lo que permite almacenar datos y programas. Esto hace que el ESP32 sea una solución ideal para proyectos que requieren un almacenamiento interno. [14–16]

Es compatible con la plataforma de desarrollo de IoT de Espressif, lo que permite a los desarrolladores crear proyectos con facilidad. La plataforma incluye una amplia gama de herramientas de desarrollo y bibliotecas para simplificar el proceso de programación. [14–16]

Soporta diversos protocolos de comunicación inalámbricos: Además de Wi-Fi y Bluetooth, el ESP32 también soporta otros protocolos de comunicación inalámbricos, como Zigbee, BLE Mesh y Thread. [14–16]

Es una solución asequible para los desarrolladores que buscan un microcontrolador para proyectos IoT. Su bajo coste lo hace accesible a una amplia gama de usuarios y proyectos. [14–16]

Es un dispositivo de bajo consumo de energía, lo que lo hace adecuado para proyectos que requieren una larga duración de batería. [14–16]

Aunque el ESP32 tiene 4 MB de memoria flash integrada, puede ser insuficiente para proyectos más grandes o aplicaciones que requieren una gran cantidad de almacenamiento. [14–16]

Es compatible con la plataforma de desarrollo de IoT de Espressif, pero no es compatible con otras plataformas de desarrollo como Arduino o Raspberry Pi. [14–16]

Características	Descripción
Procesador	Dual Core Xtensa LX6
Frecuencia	240 MHz
Memoria RAM	520 KB
Memoria Flash	4 MB
Wi-Fi	802.11 b/g/n/e/i
Bluetooth	4.2 BR/EDR & BLE
GPIO	34 pines
ADC	18 bits
UART	3
I2C	2
SPI	2
PWM	16
DAC	2
Interrupciones	19

Tabla 2. Características Modulo ESP-32[14–16]

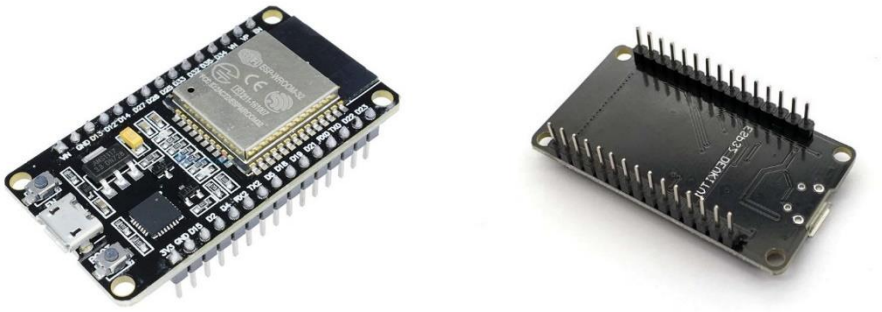


Ilustración 4. Imagen Modulo ESP-32. Imagen adaptada de la web bricogeek.[15]

2.1.3. Arduino

Arduino, creado en el año 2005, es una plataforma electrónica de bajo coste basada en un hardware y un software fácil de usar. Consta de una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra que permiten establecer conexiones fácilmente entre el microcontrolador y diferentes sensores y actuadores.[17,18]

La placa Arduino es una placa de circuito impreso (PCB) capaz de leer entradas (como luz en un sensor o el sonido de el ave en un sensor) y transformarlas en una salida (como activar un motor o enviar un mensaje a una base de datos). Para ello, se envían

instrucciones a su microcontrolador utilizando el lenguaje de programación de Arduino y su software (IDE).[7,18]

A las placas de Arduino se les puede ampliar sus funciones al conectarles a través de sus pines otras PCBs llamadas Shields, como, por ejemplo, Shield para WIFI, Shield para bluetooth, Shield para impresoras 3D, etc. [7,18]

El uso de Arduino está ampliamente extendido por las siguientes razones: es libre y extensible (se puede mejorar y adaptar su diseño, con lo que existen placas no oficiales y librerías de terceros), se puede utilizar en distintos sistemas operativos (como Linux, Windows, Mac y OS), el lenguaje de programación es sencillo pero potente (basado en C++) y es reutilizable al poderse desmontar los componentes externos de la placa para comenzar con un nuevo proyecto. [7,18]

Existen distintos modelos de placas Arduino oficiales (Uno, Due, Mega, Yun, Leonardo, entre otras) con distintas características (tamaño físico, número de pines E/S, modelo del microcontrolador, etc.) para diferentes propósitos. La mayoría utilizan la familia de microcontroladores AVR de la marca Atmel, por lo que comparten características de software, como arquitectura, librerías y documentación. [7,18]

A continuación, se explican las principales características de Arduino UNO, Arduino Mega 2560 y Arduino DUE. [7,18]

Nombre	Arduino UNO R3	Arduino Mega 2560	Arduino DUE
			
Microcontrolador	ATmega328P (AVR 8 bits)	ATmega2560 (AVR 8 bits)	Atsam3x8e (ARM 32 bits)
Voltaje Uso/Entrada	5V/7-12V	5 V / 7-12 V	3.3V/7-12V
Veloc. CPU	16 MHz	16 MHz	84 Mhz
Ent/Sal analógica	6/0	16/0	12/2 (DAC)
E/S digital/PWM	14/6	54/15	54/12
Corriente salida total pines E/S	20mA?	20mA?	130mA
Corriente max. pin 3.3V/5V	50mA/	50mA/	800mA/800mA
EEPROM	1KB	4KB	No tiene
SRAM	2 KB	8 KB	96 KB
Flash	32 KB	256 KB	512 KB
USB	Regular	Regular	2 Micro
USB OTG			1
ICSP	1	1	1
Jack	5,5/2.1mm	5,5/2.1mm	5,5/2.1mm
UART	1	4	4
Shields	Compatibilidad muy buena	Compatibilidad buena	Compatibilidad limitada
Botón Reset/ Borrar	Sí/No	Sí/No	Sí/Sí
Dimensiones	86.6x53.4mm	101.52x53.3mm	101.52x53.5mm
Peso	25g	37g	36g
Coste	24€	42€	42€

Tabla 3 Principales características de Arduino UNO, Mega 2560 y DUE.[7,18]

2.1.3.1. UNO

Arduino UNO es la placa más indicada para comenzar con la electrónica y la programación. Es el modelo de referencia, la placa más utilizada y de la cual hay mayor información disponible. Es muy económica y cuenta con una potencia adecuada para proyectos sencillos. [19]

Véase Tabla 3 para ver sus especificaciones.

2.1.3.2. Mega

Es una versión mejorada y ampliada de Arduino UNO, con mejor microprocesador, mayor número de pines, mayor memoria EEPROM, SRAM y Flash. Sin embargo, es menos potente que Arduino DUE. Véase Tabla 3 para ver sus especificaciones.

2.1.3.3. DUE

Arduino Due es la primera placa construida con un potente microcontrolador Atmel SAM3X8E de 32 bit CortexM3 ARM. Su uso es recomendado para proyectos muy complejos, que requieren muchos cálculos y memoria, y para sistemas de control especializados, con aplicaciones en tiempo real.[20,21]

Se debe tener en cuenta que esta placa y sus pines funcionan a 3.3V, a diferencia de la mayoría de placas de Arduino que funcionan a 5V. [20,21]

Véase Tabla 3 para ver sus especificaciones.

Se determina que Arduino es la placa más idónea para la realización de este trabajo frente a las otras opciones, y más concretamente el Arduino DUE sobre el UNO por los siguientes motivos:

- Microcontrolador de 32 bits en lugar del de 8 bits del Uno, lo que significa que puede manejar cálculos más complejos y procesar datos a mayor velocidad.
- Mayor cantidad de memoria RAM y memoria flash, lo que permite almacenar más datos y programas más grandes.
- Mayor capacidad de entrada/salida (I/O) y puede manejar voltajes más altos, lo que lo hace más adecuado para proyectos que requieren más potencia y precisión.
- Mayor compatibilidad con shields (placas adicionales) y puede ser utilizado en proyectos más complejos que requieren múltiples sensores y actuadores.
- Mayor velocidad de reloj que el Uno, lo que significa que puede procesar datos y ejecutar instrucciones más rápido.

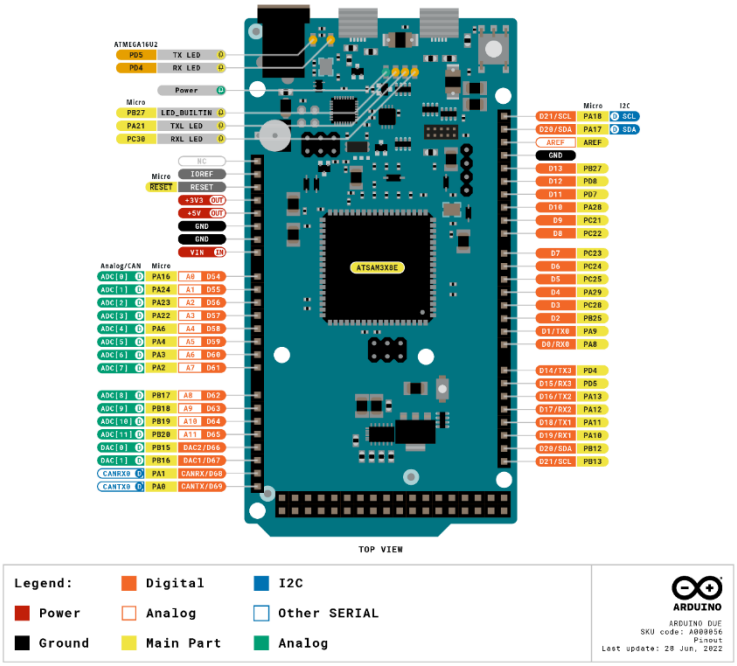


Ilustración 5 Pines de Arduino Due.

Imagen adaptada de: docs.arduino.cc/hardware/due[21]

2.2. Software - Vinculado a la elección del Hardware (Arduino)

2.2.1. Arduino

El lenguaje de programación de Arduino se puede dividir en 3 partes[22,23]:

- **Funciones:** para controlar la placa y llevar a cabo los cálculos.
- **Valores:** constantes y variables (tipos de datos).
- **Estructura:** los elementos del código C++ de Arduino.

La **estructura** incluye dos funciones principales en los sketches de Arduino[22,23]:

- Setup() se ejecuta al iniciar el sketch y se utiliza para la inicialización de los pines y la carga de las librerías, se ejecutará una vez al inicio de la placa o cuando la reseteamos
- Loop() es una función que se ejecuta en un bucle consecutivo, permitiendo al programa adaptarse y responder a las posibles entradas.

Los **valores** incluyen tipos de variables como enteros, caracteres, condicionales y decimales, así como constantes, que no pueden cambiar de valor durante la ejecución del

código. Por último, las **funciones** incluyen tanto funciones creadas por el usuario como funciones propias de Arduino que se ejecutan en la parte del loop()[22,23].

2.2.2. MySQL

MySQL es un sistema de gestión de BBDD relacionales diseñado por Oracle Corporation. Es ampliamente utilizado en aplicaciones web y es considerado uno de los sistemas de gestión de bases de datos más populares en el mundo. [24]

MySQL es un software de código abierto, lo que significa que es gratuito para usar, distribuir y modificar. Además, tiene una amplia variedad de herramientas y aplicaciones disponibles para ayudar a los usuarios a administrar y trabajar con sus bases de datos. La arquitectura de MySQL se basa en un motor de almacenamiento de tablas y una interfaz de usuario, lo que permite a los usuarios crear, modificar y consultar bases de datos de manera sencilla. [24]

Es compatible con una gran variedad de lenguajes de programación, lo que lo hace ideal para aplicaciones web, aplicaciones de escritorio y sistemas empresariales.

Se selecciona MySQL por la facilidad de su implantación gracias a los conocimientos previos que se tenían sobre él y cumple perfectamente las necesidades del proyecto.

2.2.3. Web HTML y PHP

HTML y PHP son dos lenguajes de programación diferentes que se utilizan en conjunto para desarrollar sitios web dinámicos. [24]

HTML (Hypertext Markup Language) es un lenguaje que se utiliza para estructurar el contenido de una página web. Es el lenguaje básico que se utiliza para crear la estructura de una página web, como los encabezados, párrafos, imágenes, enlaces, formularios, etc. [24]

Por otro lado, PHP (PHP: Hypertext Preprocessor) es un lenguaje de programación de servidor que se utiliza para agregar funcionalidad dinámica a un sitio web. Con PHP, se pueden crear scripts que interactúan con bases de datos, realizan cálculos, validan formularios, entre otras funciones. [24]

Un ejemplo de cómo se utilizan HTML y PHP juntos es en la creación de un formulario de contacto. El formulario se crea utilizando HTML para estructurar los campos y botones, y PHP se utiliza para enviar los datos del formulario a una base de datos o enviar un correo electrónico. [24]

En resumen, HTML se utiliza para estructurar el contenido de una página web, mientras que PHP se utiliza para agregar funcionalidad dinámica. Juntos, HTML y PHP son una combinación poderosa para crear sitios web dinámicos y funcionales. [24]

Por la sencillez que demanda la web, con HTML y PHP para la realización de las consultas a la BBDD son idóneos para nuestro proyecto.

2.3. Protocolos de comunicación

2.3.1. HTTP y HTPPS

HTTP (Hypertext Transfer Protocol) es un protocolo de red utilizado para transferir información en la World Wide Web. Es el protocolo básico que permite la comunicación entre un servidor web y un navegador web.[25,26]

Sus principales características son[25,26]:

- Es un protocolo cliente-servidor: HTTP permite que un navegador web (cliente) envíe solicitudes a un servidor web y reciba respuestas de éste.
- Es un protocolo sin conexión: HTTP no mantiene una conexión abierta entre el cliente y el servidor después de que se ha completado una solicitud.
- Es un protocolo en texto plano: HTTP utiliza un formato de texto plano para las solicitudes y las respuestas, lo que lo hace fácilmente legible para humanos y máquinas.
- Utiliza métodos de solicitud: HTTP define varios métodos de solicitud, como GET, POST, PUT y DELETE, que se utilizan para realizar diferentes acciones en el servidor.
- Utiliza códigos de estado: HTTP utiliza códigos de estado para indicar el resultado de una solicitud, como 200 OK para una solicitud exitosa y 404 Not Found para una solicitud fallida.
- Es un protocolo abierto y sin restricciones: HTTP es un protocolo abierto y no requiere licencias ni permisos para su uso.
- Es un protocolo inseguro: HTTP no proporciona seguridad para la información transmitida entre el cliente y el servidor, lo que puede ser vulnerable a ataques de interceptación o suplantación de identidad.

HTTPS (HTTP Secure) es una extensión de HTTP que agrega una capa de seguridad mediante el uso de SSL/TLS (Secure Sockets Layer/Transport Layer Security). Esto

proporciona una conexión encriptada entre el servidor web y el navegador web, lo que protege contra la interceptación de información por parte de terceros. [25,26]

En nuestro caso, como no necesitamos transmitir información segura, hemos decidido realizar las peticiones en HTTP. Esto significa que nuestra comunicación entre el cliente y el servidor no estará protegida por una conexión encriptada, pero ya que no estamos transmitiendo información confidencial, consideramos que no es necesario utilizar HTTPS.

2.4. Sonidos de las aves

Los pájaros producen sonidos vocales y no vocales. [25,26]

Los sonidos vocales se crean en su órgano llamado siringe, que se encuentra en la parte superior de la tráquea y resuena a las vibraciones cuando el pájaro fuerza el paso del aire. Las aves producen sonidos utilizando otras zonas como el pico o las alas.[27,27,28]

En este trabajo se utilizan los audios producidos por sonidos vocales, de los cuales se analizarán la frecuencia fundamental y sus armónicos.

En la distribución armónica de un sonido, la frecuencia fundamental (f_o) sería la más baja de todas, la primaria, la que hace referencia la nota que estamos escuchado. Esto no significa que sea la frecuencia con más amplitud de toda ya que puede ser que algún armónico tenga más amplitud. [27,27,28]

La f_o juega un papel importante en el canto de las aves. El canto de las aves es una forma de comunicación y es utilizado para atraer a un compañero, marcar territorio y expresar emociones. El canto de las aves suele ser característico de cada especie y consiste en una serie de notas o sonidos emitidos en un patrón específico y aprovecharemos esas características para nuestras aves. Véase ilustración 6 a continuación. [27,27,28]

La f_o es la frecuencia más baja de un sonido y es utilizada para identificar a la especie de ave. Por ejemplo, un pájaro puede cantar en una frecuencia de 1 kHz, pero su canto puede contener armónicos (notas más agudas) en 2 kHz, 3 kHz, etc. Se utiliza la frecuencia

fundamental para identificar a las aves mediante el análisis de su canto. [27,27,28]

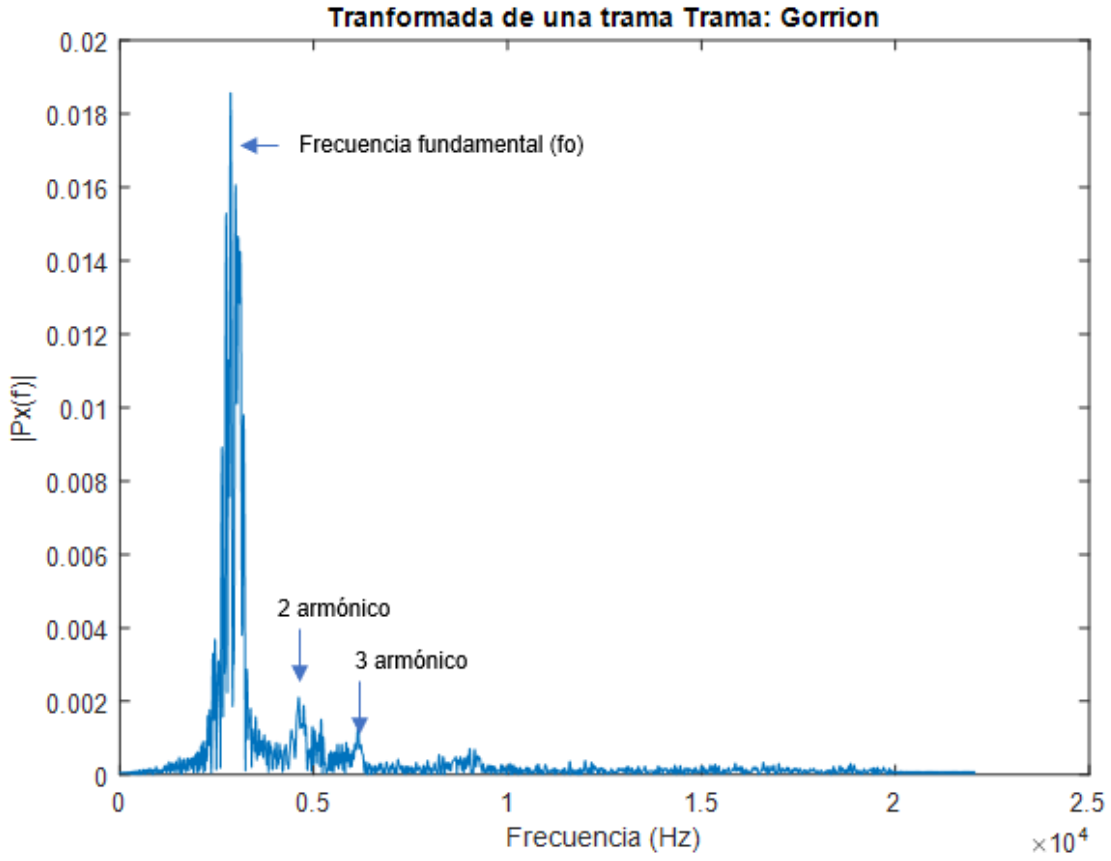


Ilustración 6 Frecuencia fundamental y sus armónicos de una trama de GORRION

Es una característica importante en el canto de las aves, ya que es utilizada para identificar a la especie y como un medio de comunicación entre los individuos de la misma. [27,27,28]

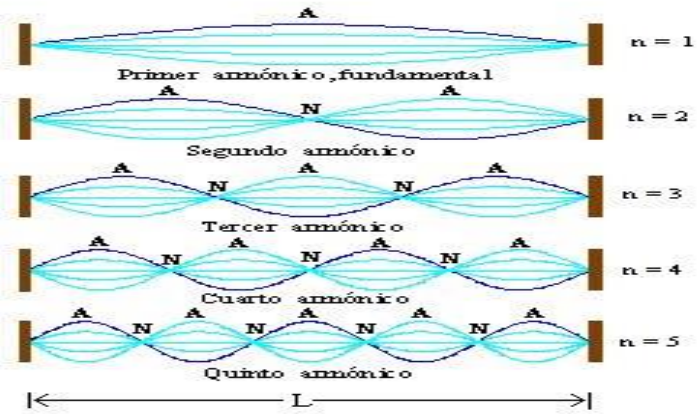


Ilustración 7 Frecuencia fundamental y sus armónicos.[29]

3. SELECCIÓN HARDWARE

3.1. Arduino DUE

Basándonos en el estudio en el apartado anterior 2.1, tras analizar las características de los posibles hardware a utilizar y sus limitaciones, se decide que los hardware más apropiados son el ESP-32 o El Arduino DUE. Finalmente se selecciona Arduino Due puesto que se ajusta perfectamente a las necesidades del proyecto y tenemos la disponibilidad en el laboratorio.

3.2. Sensores (Micrófonos)

Utilizaremos como sensor el módulo micrófono con amplificador MAX4466, este módulo incorpora un micrófono electret de 20KHz.

En la parte posterior tiene un potenciómetro el cual podemos ajustar una ganancia de 25X hasta 125X.

Especificaciones[30]:

- Chip Principal: **MAX4466**.
- Voltaje de alimentación: 2.4V a 5V.
- Dimensiones: 10 mm x 15 mm.
- Ganancia ajustable 25x – 125x.
- **Micrófono** Electrec.
- Salida riel a riel.

Según las características de su rango se sensibilidad de 20hz hasta 20Khz y por el reducido precio del módulo, basándonos en experiencias anterior con buenos resultados tras su utilización, se selecciona el sensor MAX446 para este proyecto.

3.3. Modulo Wifi Fuente de alimentación MB102

ESP8266 ESP-01 es un módulo Wifi de bajo coste y bajo consumo de energía desarrollado por Espressif Systems. Es uno de los módulos ESP8266 más populares debido a su tamaño compacto y su precio accesible. [31–33]

Características[31–33]:

- Conectividad Wifi: El ESP8266 ESP-01 se conecta a redes Wifi existentes mediante el protocolo 802.11 b/g/n.
- Microcontrolador: Incorpora un microcontrolador de 32 bits (L106) con una velocidad de reloj de 80 MHz.
- Memoria: Tiene una memoria flash de 1MB y una memoria RAM de 80 KB.
- Entradas/Salidas: Tiene dos pines de entrada/salida (GPIO) y un pin de reset.
- Tamaño: El tamaño del módulo es de solo 14.5 x 24.7 mm, lo que lo hace ideal para proyectos de espacio reducido.
- Bajo consumo de energía: El ESP8266 ESP-01 tiene un bajo consumo de energía, lo que lo hace ideal para proyectos de batería.

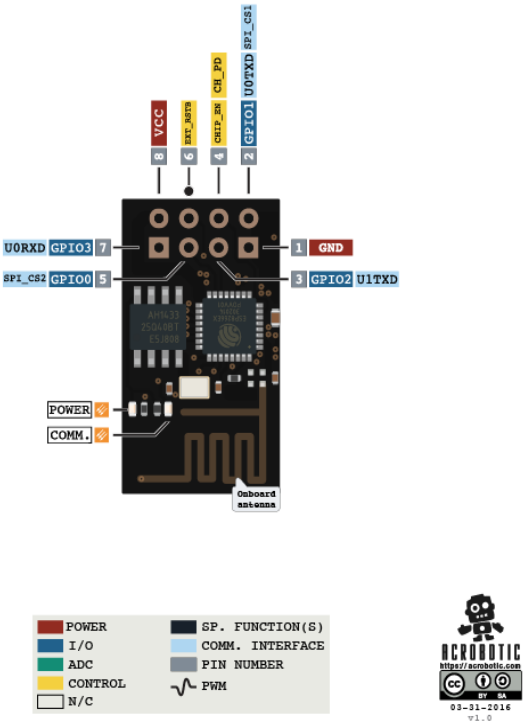


Ilustración 8 Modulo wifi-Pines [33]

El ESP8266 ESP-01 es utilizado en una amplia variedad de proyectos IoT, como sensores de temperatura y humedad, dispositivos de control remoto, automatización del

hogar, dispositivos de seguimiento GPS, entre otros. Es compatible con una gran cantidad de lenguajes de programación como C, Python, Lua y JavaScript.

ESP8266 ESP-01 es un módulo Wifi de bajo costo y bajo consumo de energía, con características adecuadas para proyectos IoT, con conectividad Wifi, microcontrolador, memoria flash y RAM, entradas/salidas e interfaz de programación, lo que lo hace una excelente opción para este proyecto.

Para la alimentación del módulo Wifi se necesita una fuente de alimentación independiente al Arduino y que su demanda de amperaje es superior a la ofrecida únicamente por la placa de Arduino.

Fuente de alimentación MB102 es una fuente de alimentación de bajo voltaje y corriente regulada que se utiliza para alimentar proyectos electrónicos. Es una fuente de alimentación de laboratorio conocida por su fiabilidad y estabilidad en la corriente y el voltaje. [32]

El uso de la fuente de alimentación no obliga al uso de una placa protoboard para su conexionado. Véase ilustración 10

Características[32]:

- Voltaje de salida: El rango de voltaje de salida es de 3.3V a 12V, con un voltaje de salida ajustable mediante un potenciómetro.
- Corriente de salida: La corriente de salida máxima es de 3A.
- Regulación: La fuente de alimentación MB102 tiene una regulación de voltaje y corriente precisa para garantizar una estabilidad en el voltaje y la corriente de salida.
- Indicador LED: Incorpora un indicador LED que indica el estado de encendido/apagado de la fuente de alimentación.
- Conectores: La fuente de alimentación MB102 tiene conectores de entrada y salida para facilitar la conexión con los dispositivos a alimentar.
- Tamaño: El tamaño de la fuente de alimentación MB102 es compacto y fácil de transportar.

La fuente de alimentación MB102 es ampliamente utilizada en proyectos de electrónica, robótica, automatización y otros proyectos en los que se requiere una alimentación estable y regulada. Es compatible con una gran variedad de dispositivos y puede ser utilizada para alimentar desde circuitos integrados hasta motores y otros dispositivos con un consumo de energía moderado[32].

Por estas características y su reducido precio es idónea para este proyecto

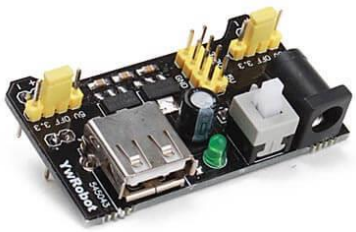


Ilustración 9 Fuente de alimentación MB102.[32]

3.4. Conexionado

Aquí explicaremos las conexiones conexiones fisicas que se han realizado para el montaje de nuesrto dispositivo final.

Lo primero que conectaremos sera la fuente de alimentacion a la protoboard de la siguiente manera:

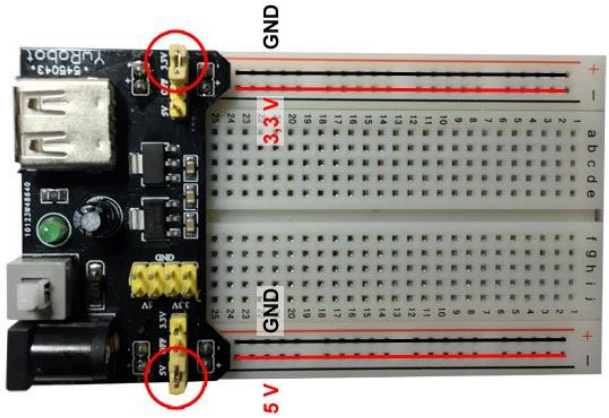


Ilustración 10 Fuente de alimentación MB102. [32]

Una vez tenemos la fuente de alimentacion d instalada, realizaremos las conexión de GND y 3.3v con el arduino.

Tambien conectaremos en la protoboard dos leds informativos con una resistencia de 220 Omios cada una. Los led se conectaran a las salidas digitales 10 y 8 del arduino.

8:Azul // 10:Blanco

Conectaremos La salida del microfono a la entrada analogica A0 del arduino.

- Out: A0-Verde
- VCC:3.3v del Arduino-Rojo
- GND: GND del arduino-Negro

Ilustración 11 Conexionado micrófono

El dispositivo wifi se conectaremos de la siguiente manera[32]:

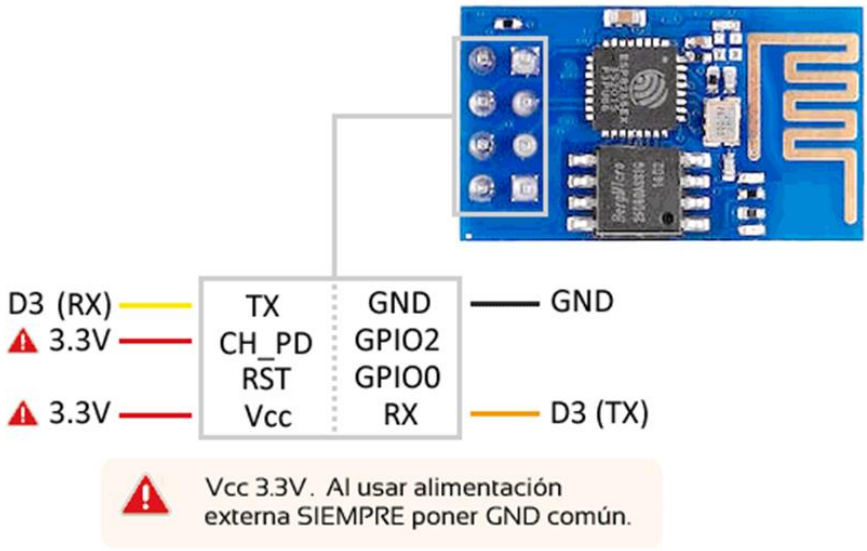


Ilustración 12 Conexionado Wifi [32]

Finalmente tendremos todo conectado de la siguiente manera:

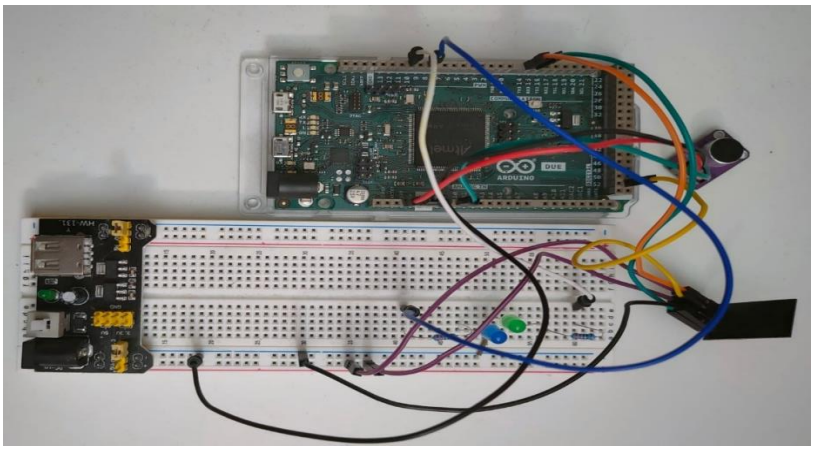


Ilustración 13 Conexionado General 1

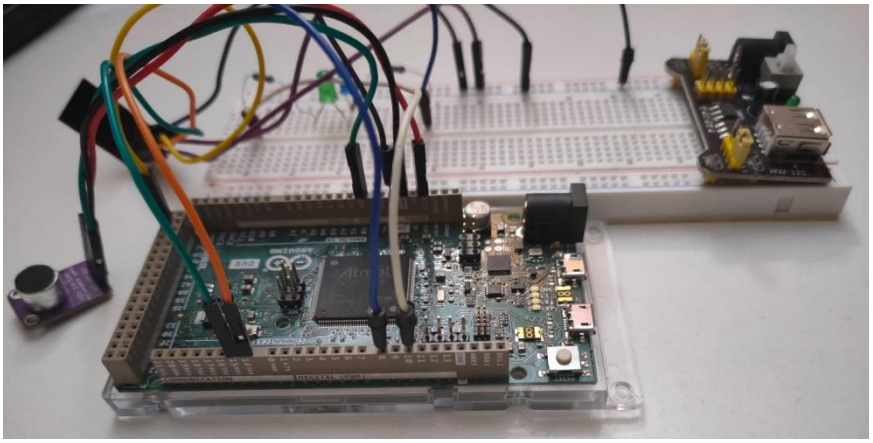


Ilustración 14 Imagen Conexionado general 2

4. SOLUCIÓN

El diseño de la solución propuesta para la identificación de las aves mediante el sonido que emiten, consta de los siguientes puntos:

1. **Creación de una biblioteca de muestras:** contenedora de 20 audios por ave (10 con los que se crea el algoritmo y 10 externos al mismo para pruebas), como se estudian 6 aves diferentes, el total de la biblioteca consta de 120 audios. La selección de aves, se hizo para abarcar todo su espectro:
    - Palomas, gaviotas: graves (frecuencias bajas)
    - Lechuzas, carboneros, gorriones: medios (frecuencias medias)
    - Carbonero, vencejos: agudos (frecuencias altas)
2. **Desarrollo en Matlab:**
    a. **Análisis de los audios de las aves:** se calcula la transformada de Fourier para posteriormente, extraer las frecuencias y los armónicos característicos de cada ave. Para ello, cada audio se divide previamente.
    b. **Caracterización de las aves:** para caracterizarlas se crea un algoritmo de extracción de parámetros utilizando los resultados de los análisis de los audios. En cada trama (fragmento del audio) de cada audio se determina:
        - la frecuencia fundamental (f_o)
        - los armónicos
        - la relación entre la amplitud de la frecuencia fundamental y el segundo armónico
        - la relación entre la amplitud de la frecuencia fundamental y el tercer armónico.

    Estos datos procedentes de todas las tramas analizadas, se almacenan en vectores para, posteriormente, calcular la media sobre 10 tramas consecutivas de la f_o, su relación entre el segundo y tercer armónico, la desviación típica (SD) de la f_o y el coeficiente de variación (CV).

    c. **Identificación de las aves:** con los resultados de la caracterización de las aves se crea un algoritmo de identificación que permita diferenciar cada ave dentro de Matlab. Se realizan pruebas durante todo el proceso hasta conseguir una identificación elevada. Véase punto 4.2.3 tablas 23-24 y 25.
3. **Programación de Arduino DUE:** con lo aprendido del paso anterior se utiliza para la programación del algoritmo de identificación en este dispositivo de bajo coste, al cual se conectan los sensores ubicados en las áreas de estudio. Estos sensores

registran las señales acústicas producidas por las aves que se analizan en Arduino. Se realizan pruebas en Arduino hasta conseguir una identificación óptima. El dato con el ave identificada se envía a una base de datos. Véase punto 4.3.6

4. **Creación de una base de datos (BBDD) y visualización en una aplicación web:** la base de datos recibe los datos de las aves identificadas procedentes de Arduino y los muestra en una aplicación web, que es accesible por cualquier usuario.

Esquema general de la solución propuesta:

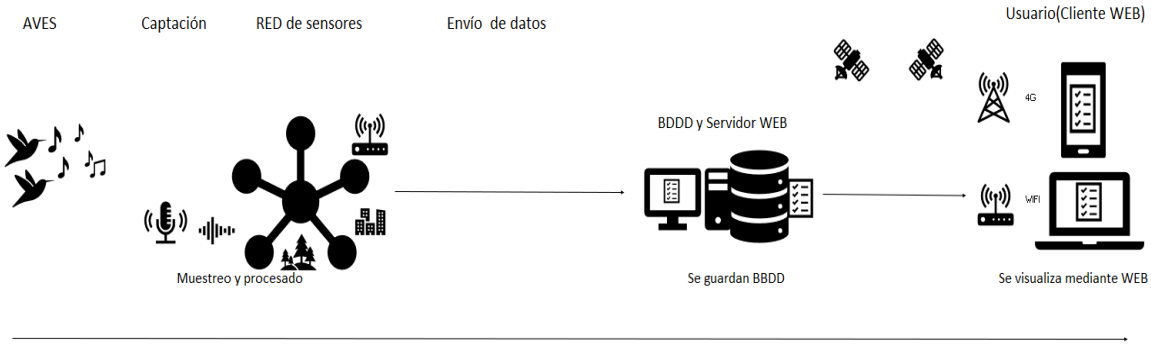


Ilustración 15 Esquema genérico de la solución.

Las aves emiten sonidos, estos los captamos a través de una red de sensores, muestreamos el sonido en tramas, tras analizar dichas tramas identificamos el ave presente.

Una vez se ha identificado correctamente el ave a través de la conexión wifi enviamos el ave identificada a la BBDD. Véase punto 4.3.6 aquí se almacenará con una estructura definida. Véase 4.4 ilustración 58

Se podrá observar todas las aves identificadas en un registro accesible mediante una ampliación web. Véase 4.4 Ilustraciones 60-61 y 62

En los siguientes apartados se explican todos los componentes de la solución en detalle.

4.1. Biblioteca de muestras

Para la realización de este proyecto y estudio de las aves necesitamos hacer uso de ficheros de sonido que fueran representativos de ellos; así que realizamos una pequeña biblioteca de muestras con los sonidos de las aves, para poder realizar las pruebas y perfeccionar el algoritmo.

Como no disponíamos de un equipo profesional para realizar audios de estas aves, nos hemos dispuesto a obtenerlos de manera digital de webs especializadas.

Fuente de los audios: https://www.xeno-canto.org/[34]

Se han seleccionado las siguientes aves para el estudio ya que abarcan casi todo el espectro de frecuencias sobre los cantos de aves, y valdrían como ejemplo para cualquier ave que queramos identificar.

En esta biblioteca de muestras se encuentran: 10 audios de cada ave para la realización del algoritmo y 10 audios de la misma ave externos al algoritmo para la realización de pruebas.

Especie de Ave- Algoritmo	Nombre del archivo de audio
Palomas	Paloma bravía_001-010
Gaviotas	Gaviota argéntea europea _001-010
Gorriones	Gorrión común_001-010
Lechuzas	Lechuza_común_001-010
Carboneros	Carbonero común _001-010
Vencejos	Vencejo_comun_001-010

Tabla 4 Biblioteca de muestras

4.2. Desarrollo en Matlab

En este apartado se detalla la ejecución del código creado para producir la solución. Este código consta de un algoritmo de extracción de parámetros y un algoritmo de identificación. Para crearlos, se realizan pruebas y que eligiendo las variables que aportan más información y en virtud de estas se propone el algoritmo que posteriormente se

aplicara, se ha realizado un proceso de aprendizaje para concluir con el algoritmo que mejor se adapta a la identificación

El desarrollo en Matlab se compone de las siguientes etapas:

- **Análisis de los audios de las aves.**
- **Caracterización de las aves: algoritmo de extracción de parámetros.**
- **Identificación de las aves: algoritmo de identificación y confirmación.**

4.2.1. Análisis de los audios de las aves

El análisis de los audios comprende la carga del fichero del audio, la división en tramas, % el solapamiento (porcentaje de la trama anterior analizada que vuelve analizarse conjuntamente con la nueva trama siguiente**), filtro de potencia** (filtro que descarta la información no útil. Véase punto 4.2.1.3) **y la transformada de Fourier de las tramas.**

- A mayor tiempo de audio menos precisa será la información que aporte la Transformada de Fourier, por lo que cada fichero se divide en tramas. Se hacen pruebas para escoger el tamaño óptimo de las tramas, resultando el valor más apropiado el de 2048 muestras (véase punto 4.2.2.1). Estas tramas se prueban con distintos valores de solapamiento: 0%, 50% y 90% (sin solapamiento, un valor medio, y un valor cercano al máximo para evaluar bien sus efectos). Se escoge el valor del 90% por sus mejores resultados (véase punto 4.2.1.2), para usar en el algoritmo de extracción.
- Se calcula la potencia total del fichero, para aplicar un filtro y desechar todas las secciones de los audios y tramas con potencia inferior al 10% de la potencia del total del fichero, al considerarse información no relevante. Se hicieron pruebas con filtro de potencia de 0%, 1%, 5% y 10%, seleccionándose 10% al ofrecer las tramas de canto de ave óptimas sin desestimar datos importantes como podría pasar con un filtro mayor a 10%. (Véase punto 4.2.1.3)

A continuación, se proporciona más información sobre estas operaciones que se han realizado en paralelo.

4.2.1.1. Carga del fichero de audio

El primer paso es cargar el fichero del que se va a extraer la información. Para esto, se utiliza la **función *audioread*** de Matlab.

Esta función nos permite extraer información del archivo de audio:

- Dos canales de sonido, dado que los audios están en estéreo.
- Una variable con la frecuencia de muestreo (Fs.) del audio. La Fs. de todos los audios analizados siguen el estándar de la codificación MP3[35] de una frecuencia de muestreo de 44100 muestras.

```
[x1,fs1]=audioread(Ruta de la pista');
 x1=x1(:,1);
 ts=1/fs1;
```

Cargamos la pista en estéreo. Lo siguiente que se realiza es pasarla a mono para así facilitar su tratamiento. La introducimos en la varíale “x1” así como su frecuencia de muestreo “fs1”.

Una vez ya obtenida fs podremos obtener periodo fundamental “ts” invirtiendo fs.

4.2.1.2. División en tramas de los audios y solapamiento

- A menor tamaño del fichero mayor precisión de la transformada de Fourier, por lo que cada fichero se divide en tramas. Se hacen pruebas para escoger el tamaño óptimo de las tramas, resultando el valor más apropiado el de 2048 muestras (estas pruebas se pueden ver en el apartado 4.2.2.1 al final del documento y que se terminara de decidir con la efectividad de nuestro algoritmo de detección). En una primera instancia comenzaremos por el valor de 512 muestras para comprobar posteriormente su valor idóneo.

- Estas tramas se prueban con distintos valores de solapamiento: 0%, 50% y 90% (sin solapamiento, un valor medio, y un valor cercano al máximo para evaluar bien sus efectos). Se escoge el valor del 90% tras la conclusión, para usar en el algoritmo de extracción. No obstante, se concluye que el solapamiento no afecta de manera significativa a la obtención de los parámetros

- Tras obtener la información aportada con un solapamiento de 0% 50% o 90% de las tramas observamos que no aporta información nueva dicho solapamiento, solo repite la información que extraíamos con la primera prueba de 0%. Se puede observar en las Ilustraciones 16-17 y 18 como la información no varía, solo se repite. Por lo que concluimos en usar un 90% y así poder distinguir mejor visualmente la información. Esto incrementa el coste computacional al tener que analizar 200 tramas en lugar de 20 que se analizarían sin solapamiento. Al no existir limitaciones en potencia del hardware en este punto, puede analizar sin problema este incremento de tramas.

Obtención de tramas con un 0% de solapamiento:

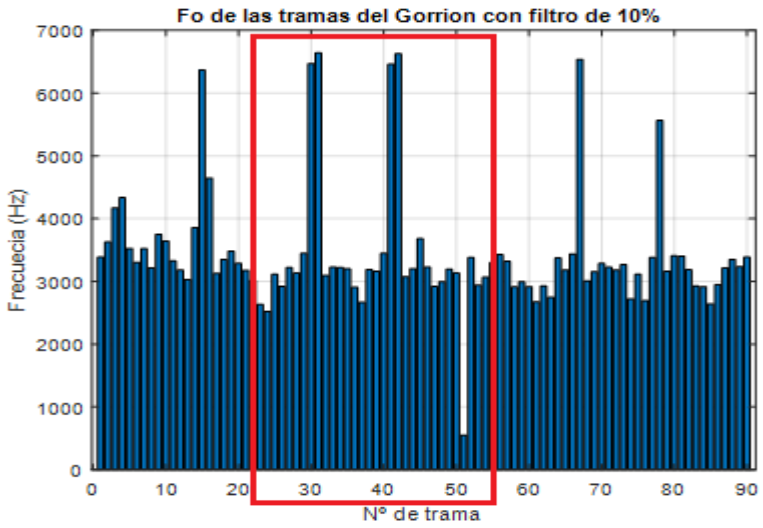


Ilustración 16 Tramas con solapamiento de 0%

Obtención de tramas con un 50% de solapamiento:

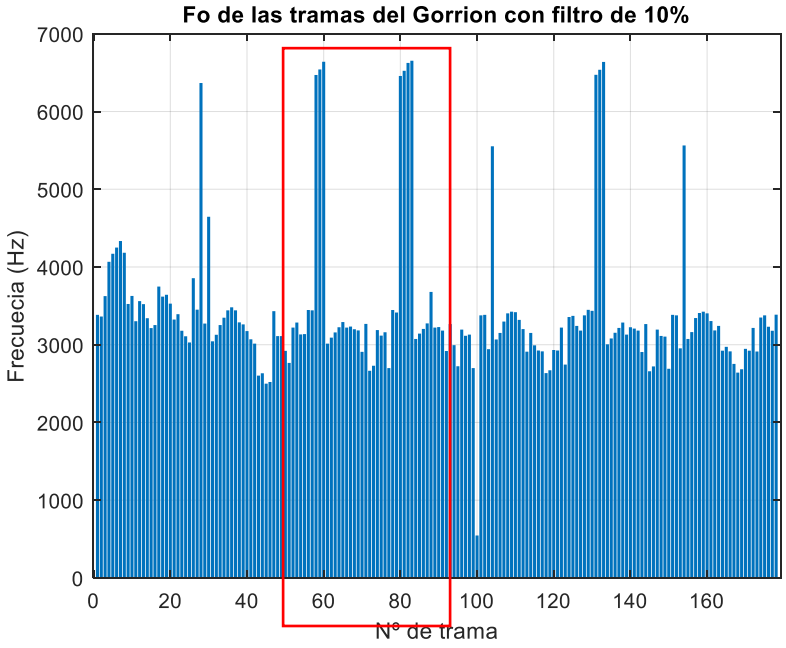


Ilustración 17 Tramas con solapamiento de 50 %

Obtención de tramas con un 90% de solapamiento:

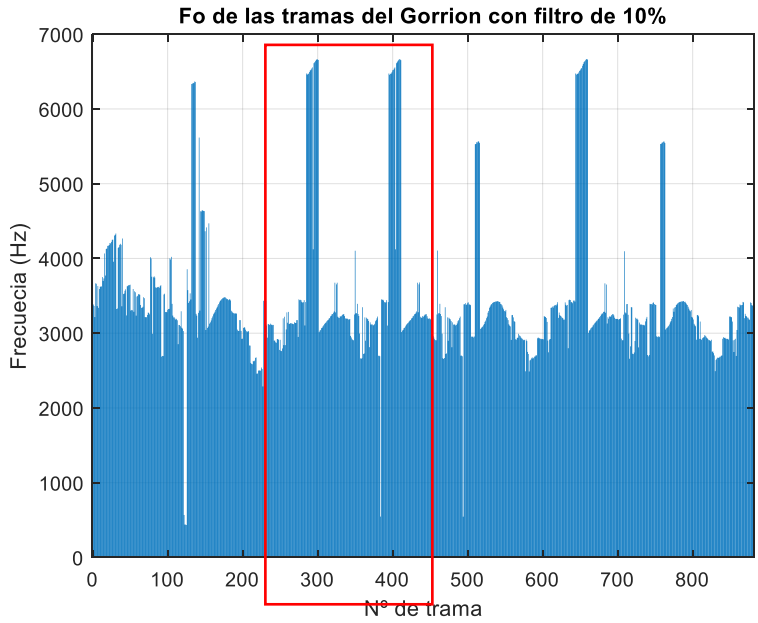


Ilustración 18 Tramas con solapamiento del 90%

Una vez determinados los parámetros del filtro de potencia, así como el tamaño de trama, realizaríamos la separación de estas.

```
solapa=0.1;
Tamaño: tam_trama=2048;
Numero de tramas: numero_de_tramas=round(round(length(Audio{i})-
tam_trama+1)/n);
```

Con estos datos necesarios se construirá un bucle por el que se repetirá, así como en tantas tramas se divida el fichero original. El porcentaje de solapamiento incrementara considerablemente el número de tramas analizar.

Cada trama realizada con el for es analizada por separado. Calculamos su potencia para aplicar un filtro de potencia y descartar las tramas que no llegan a un mínimo y así deshacernos de las tramas en silencio.

Si la trama supera el umbral de potencia establecido será válida para realizar la Transformada de Fourier que explicaremos en el siguiente punto.

```
for t=1:numero_de_tramas
trama=x(1+(tam_trama*P*solapa):(tam_trama*t)-(tam_trama*(1-solapa)*P));
            P=P+1;
Px_trama=sum(trama.^2)*(1/length(trama));
if Px_trama>=filtro
….
```

4.2.1.3. Filtro de potencia

El código para utilizar el filtro de potencia es el siguiente:

```
pot_x1=(1/length(x1))*sum(x1.^2);
filtro=(porcentaje)*pot_x1;

if Px_trama>=filtro
```

Se calcula la potencia total del audio y de las tramas. Este se determina para descartar tramas que no contienen información útil. Las pruebas realizadas con el filtro de potencia de 0%, 1%, 5% y 10%, se muestran a continuación con el ejemplo del gorrión. Con esto se concluye que el filtro de potencia más adecuado es el 10% y que es el idóneo para eliminar las tramas del audio que no aportan información sin llegar a eliminar las tramas que si aportan información, el cual se utilizará para el algoritmo de extracción de parámetros. (Véase ilustraciones 19,20,21,21)

En la primera imagen se observa el canto del gorrión en el tiempo y al lado la información de la f0 de cada una de sus tramas. (Véase ilustraciones 19-18)

Se puede observar como en el tiempo del canto del gorrión es donde se obtiene la información útil de las fo de las tramas.

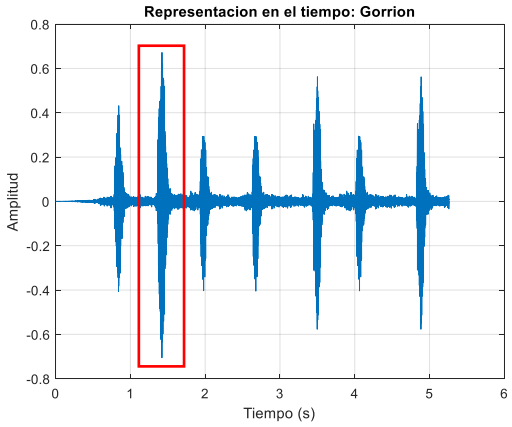


Ilustración 19 Sonido del gorrión en el tiempo

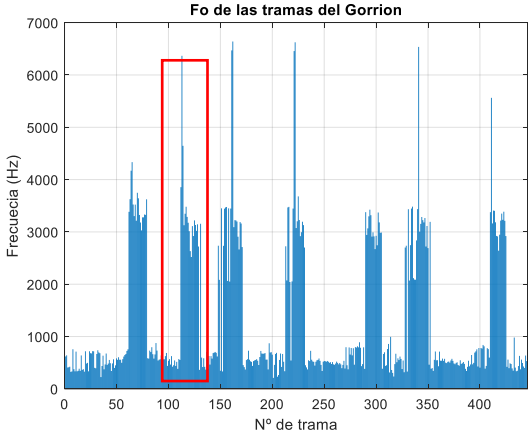


Ilustración 20 F_0 de las tramas del gorrión

En las siguientes imágenes se aplicará un el filtro anterior mencionado de 1% de potencia el 5% de potencia y el 10% de la potencia del archivo original. (Véase ilustraciones 21-22)

Información con un filtro de potencia del 1%. Se ve como seguimos obteniendo mucha información no útil donde no hay canto por lo que se analizarían tramas innecesarias. (Véase ilustraciones 21)

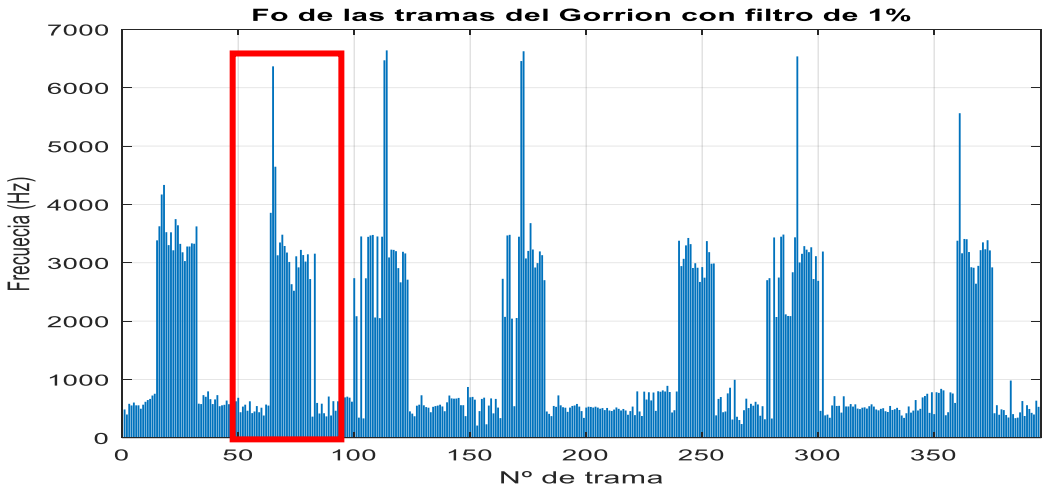


Ilustración 21 F_o de las tramas del gorrión con un filtro de potencia del 1%

Información con un filtro de potencia del 5%. Se observa como disminuye esa información no útil en los vacíos de canto manteniéndose la información cuando hay canto del ave. (Véase ilustraciones 22)

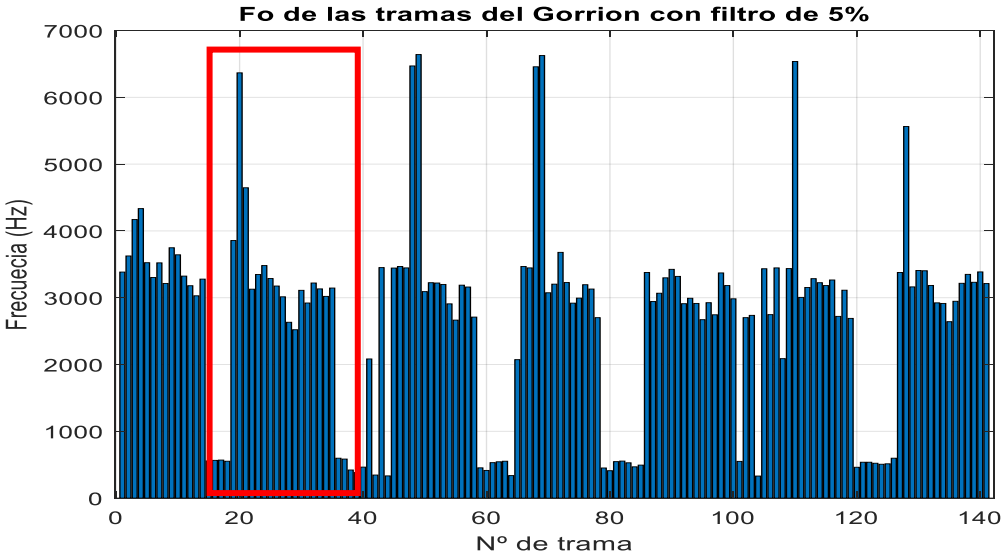


Ilustración 22 F_o de las tramas del gorrión con un filtro de potencia del 5%

Información con un filtro de potencia del 10%. Se observa como disminuye más aun esa información no útil manteniendo la información útil en exclusiva. (Véase ilustraciones 23)

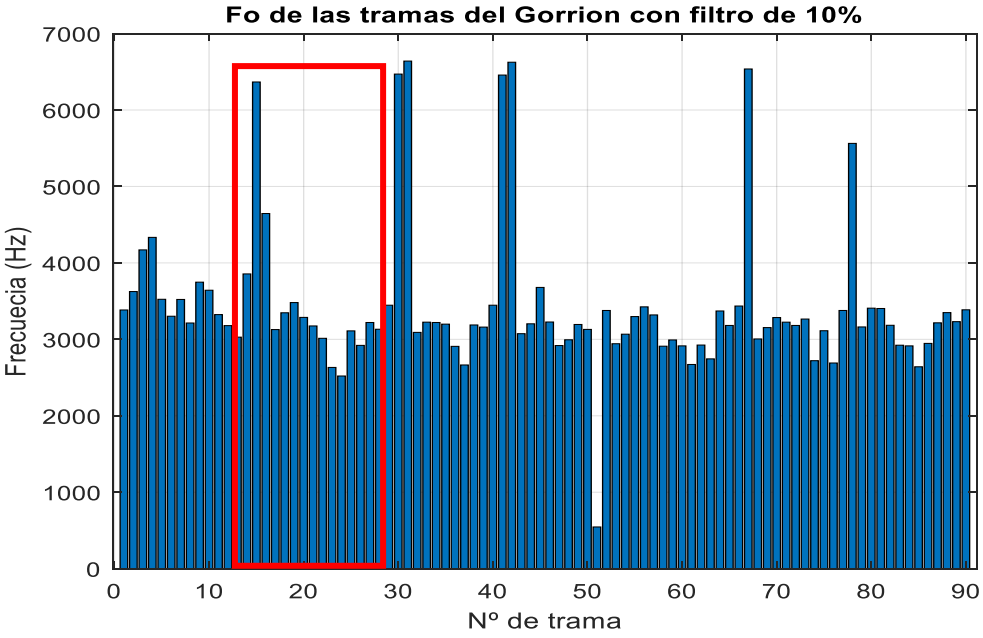


Ilustración 23 F_o de las tramas del gorrión con un filtro de potencia del 10%

Información con un filtro de potencia del 15%. Se observa como ya no hay tramas de información no útil y empiezan a disminuir la información útil del ave. (Véase ilustraciones 24)

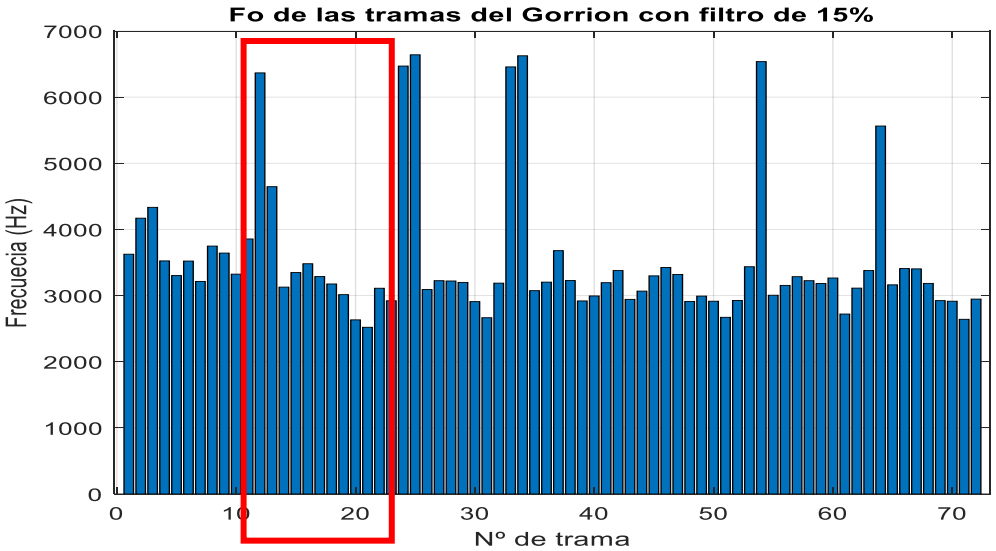


Ilustración 24 F_o de las tramas del gorrión con un filtro de potencia del 15%

Por los motivos anteriores concluimos que el valor para el filtro de potencia donde se decide que tramas son útiles o no, será del 10%

4.2.1.4. Transformada de Fourier de las tramas

Se realiza la transformada de Fourier a todas las tramas que han pasado el filtro de potencia y así obtener las f_o y sus armónicos.

En Matlab la función necesaria para realizar la transformada es ftt(). Esta función devuelve los valores invertidos. Se usa la función fftshift (TF) para ordenar las frecuencias.

De valores obtenidos, solo se utilizarán los positivos y se descartarán las frecuencias negativas (que no existen).

```
TF=fft(trama,40*length(trama))/(length(trama));
TF1=abs(fftshift(TF));
TF2=TF1((length(TF1)/2):length(TF1)-1)

%Para la representación grafica
f=-FS/2:FS/(length(TF1)-1):FS/2;
f1=f((length(f)/2)+1:length(f));

figure()
plot(f1,TF2)
```

Ejemplo de la transformada de Fourier de una trama del gorrión:

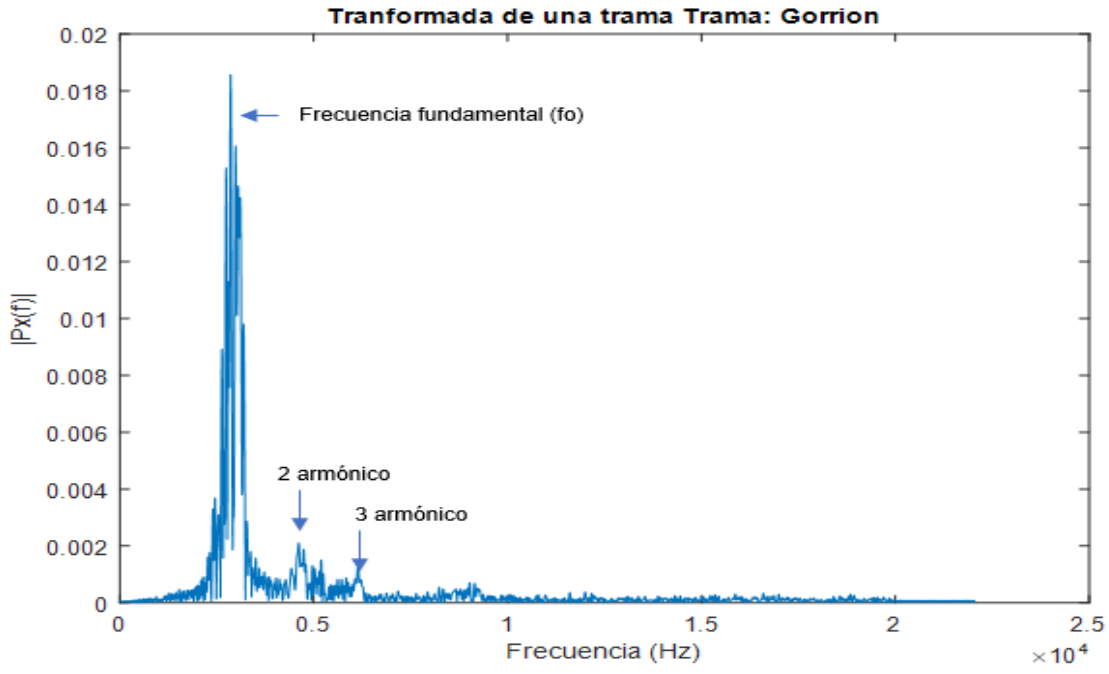


Ilustración 25 Frecuencia fundamental y sus armónicos de una trama de GORRION

4.2.2. Caracterización de las aves: algoritmo de extracción de parámetros

Para caracterizar las aves se estudian los siguientes parámetros:

- la frecuencia fundamental (f_o)
- los armónicos (segundo y tercero)
- la relación entre la amplitud de la frecuencia fundamental y el segundo armónico
- la relación entre la amplitud de la frecuencia fundamental y el tercer armónico.
- Estos datos procedentes de todas las tramas analizadas, se almacenan en vectores para, posteriormente, calcular la media sobre 10 tramas* consecutivas de la f_o, su relación entre el segundo y tercer armónico, la desviación típica (SD) de la f_o y el coeficiente de variación (CV)de la f_o.

*Se escogieron 10 tramas porque se hicieron pruebas con 3, 5, 10 tramas, observándose que con 3 y 5 tramas no podíamos obtener un valor fidedigno de desviación típica ya que apenas había variación en esas muestras, y se obtenía un valor similar en todas las aves. (Véase ilustración 26)

Con más de 10 muestras se obtenían valores demasiado dispares y concluimos que 10 muestras era el tamaño idóneo para este valor.

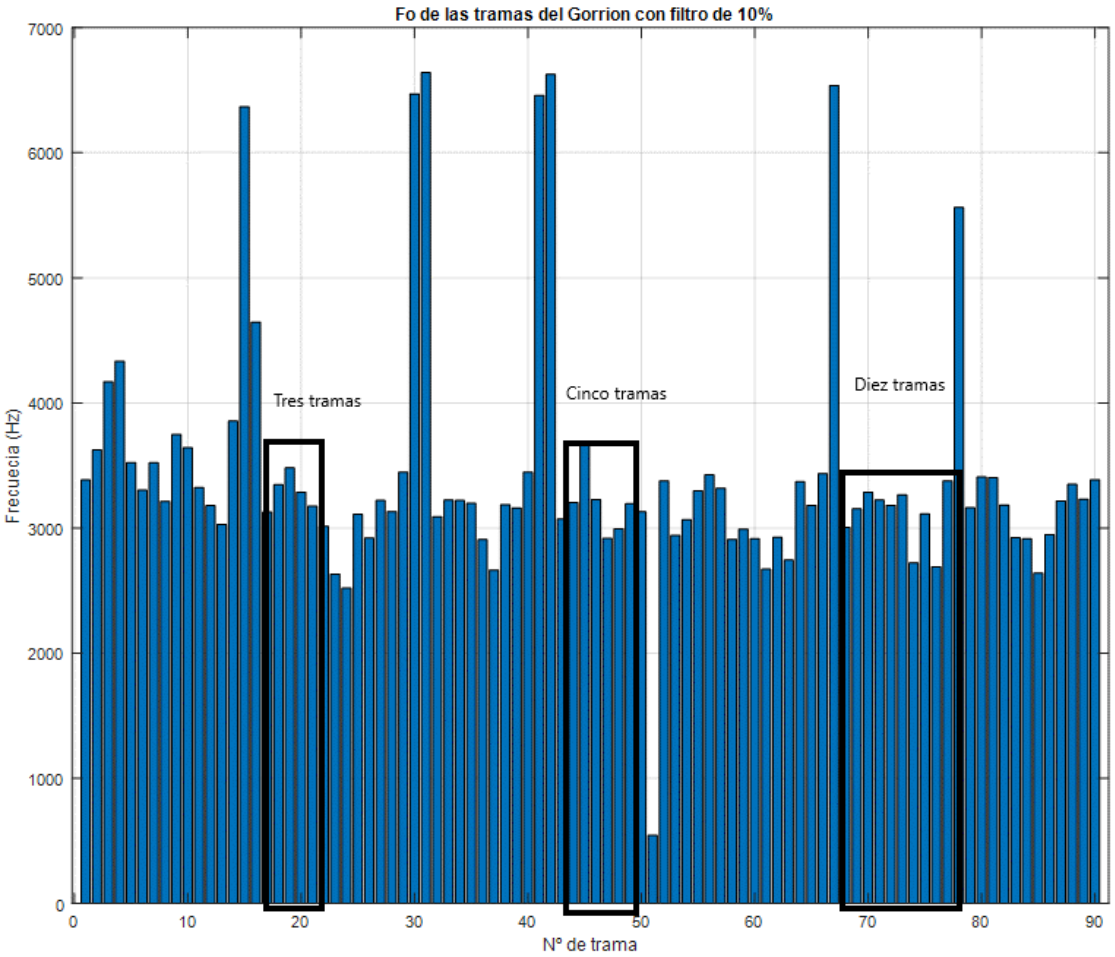


Ilustración 26 Visualización de la obtención del número de muestras para realizar la SD y CV

El diagrama del algoritmo de la extracción de parámetros sería el siguiente:

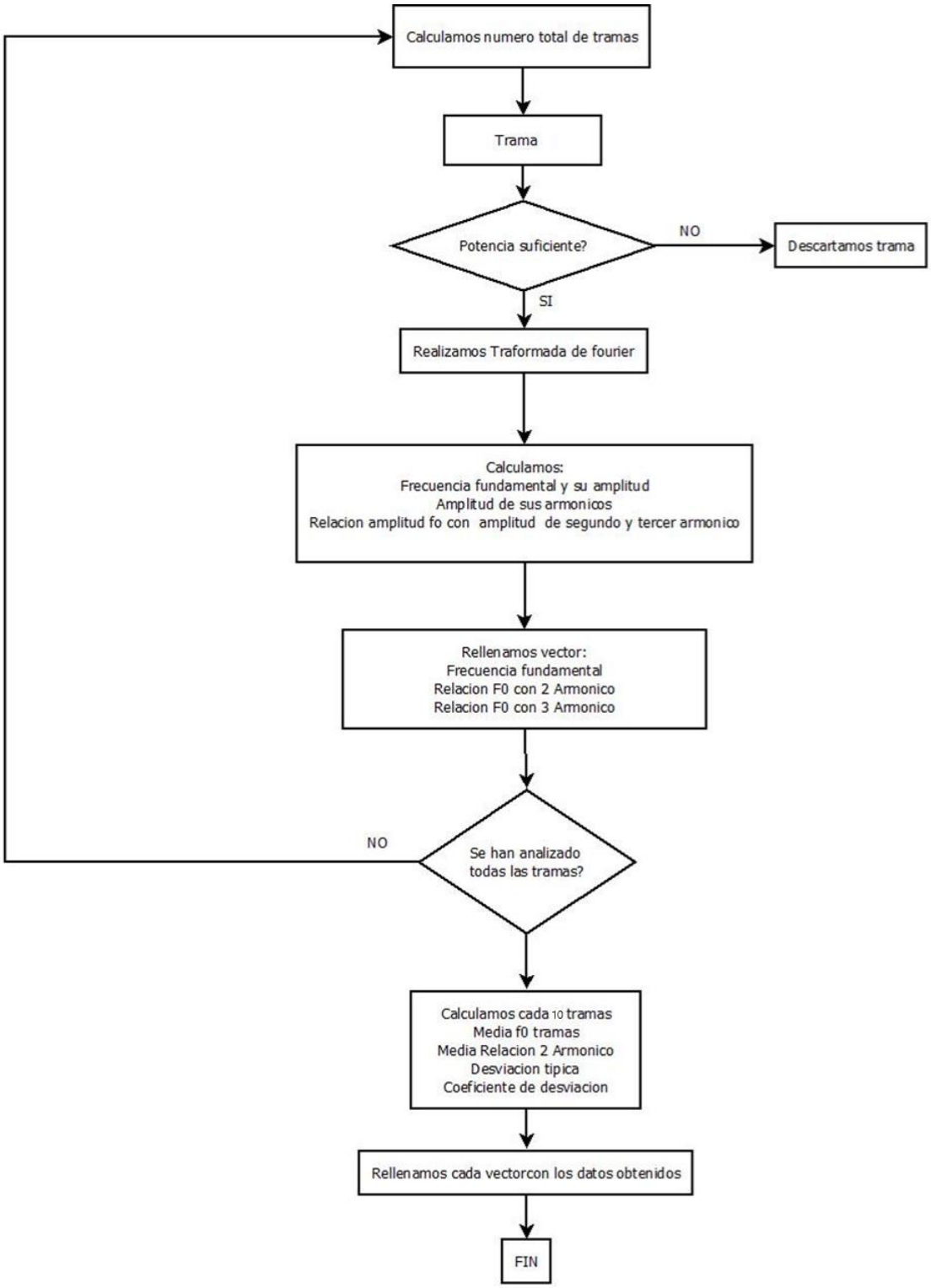


Ilustración 27 Diagrama del algoritmo de extracción de parámetros.

Para extraer la frecuencia fundamental lo realizaremos con el siguiente código. Calculamos el máximo de nuestro vector Transformada de Fourier. Y la posición que ocupa. Así como su amplitud

```
[maximo, pos ]=max(TF2);
fo=f1(pos);
```

```
A_fo=maximo
```

Después nos disponemos a buscar los armónicos que no es más que múltiplos de la frecuencia fundamental con su amplitud. Si el segundo armónico se encuentra por encima de nuestro hacho de banda se igualará a 0. El Ancho de banda se limita a 44100hz/2 por el teorema de Nyquist[36].

Solo considerarnos el segundo y tercer armónico ya que más allá del tercero desbordaríamos nuestro rango de frecuencias, estando fuera de nuestro rango para poder analizar. Como el proceso es igual solo detallaremos el 2 armónico.

```
fa2=fo*2;
if fa2>f1(end)
A_f2=0;
Else

fa3=fo*3;
if fa3>f1(end)
A_f3=0 ;
else
```

Una vez tenemos la frecuencia del segundo y tercer armónico nos disponemos a obtener su amplitud

```
incre=length(TF2)/f1(end);
incre=incre/2;
if incre<1
incre=round(incre*10);
else
incre=round(incre/2
end

Muestra_fa2=find((f1 >= fa2-incre)&(f1<=fa2+incre)); A_f2=TF2(Muestra_fa2);
A_f2=max(A_f2);
```

Rellenamos los vectores con los valores obtenidos, al tener ya estos vectores podremos calcular la relación entre la amplitud de Fo y la del 2º armónico

```
frec(j)=fo;
ampli_fo(j)=A_fo;

frec_a2(j)=fa2;
```

```
ampl_fa2(j)=A_f2;
```

Relación entre Fo y segundo armónico, cuando la amplitud del segundo armónico es demasiado pequeña o próxima a 0, resultado de la relación entre la amplitud de Fo y la amplitud del 2º armónico es Inf, para poder seguir operando y no rompa nuestro código con un error, todos los valores Inf serán modificado por 1000.

```
relacion_fo_f2(j)=ampli_fo(j)/ampl_fa2(j);

if relacion_fo_f2(j)==Inf
relacion_fo_f2(j)=1000;
   else
    end


                  j=j+1;
```

Teniendo ya estos datos, realizamos el cálculo de 10 tramas seguidas de los siguientes parámetros.

- Media de las Fo
- Media de la relación con el segundo armónico
- Media de la relación con el tercer armónico
- Desviación típica de las f_o
- Coeficiente de variación de las f_o.

Estos datos se calculan y almacenan en vectores gracias al siguiente código:

```
Frecuencias2=frec;
relacion=relacion_fo_f2;
          grupo=grupo1;
         conjunto=0;
         conju_rel=0;
         cont=0;


         for q=1:length(Frecuencias2)/grupo
               conjunto=Frecuencias2((1+cont):(grupo*q));
               conju_rel=relacion((1+cont):(grupo*q));
                 cont=cont+grupo;
                DE(q)=std(conjunto);
                media=mean(conjunto);
                media_re=mean(conju_rel);
                media_conj(q)=mean(conjunto);
                media_rel(q)=mean( media_re);
```

```
        Coef(q)=(DE(q)/media)*100;
End
```

Estos parámetros se consideran los claves para la caracterización del ave.

A continuación, con la información explicada anteriormente, se detalla la caracterización de cada ave.

4.2.2.1. Pruebas para seleccionar el tamaño de trama idóneo para nuestro algoritmo de extracción de parámetros

Resultados con un tamaño de trama de 512 1024 y 2048 muestras:

Se puede observar en estas pruebas que la trama de 2048 muestras es la que aporta mejores resultados para la identificación de las aves ya que es con las que se obtiene un % más alto positivo y a la vez es la que menos tramas indeterminadas obtiene.

Pruebas con trama de 512 Muestras:

Tamaño de trama de 512		
Ave	**%Acierto en tramas detectadas como aves**	**%Tramas Indeterminadas**
Palomas	99,3	49,6
Gaviotas	92,5	56,4
Lechuzas	99,1	44,4
Carboneros	63,6	40,7
Gorriones	85,5	33,4
Vencejos	92,7	23,9
Media	86,7	38,3

Tabla 5 Resultados de pruebas con tramas de 512 muestras

Pruebas con trama de 1024 Muestras:

Tamaño de trama de 1024		
Ave	**%Acierto en tramas detectadas como aves**	**%Tramas Indeterminadas**
Palomas	99,7	49,9
Gaviotas	94,2	61,8
Lechuzas	98,3	52,4
Carboneros	77,5	47,2
Gorriones	82,9	30,8
Vencejos	93,6	19,1
Media	90,3	37,2

Tabla 6 Resultados de pruebas con tramas de 1024 muestras

Pruebas con trama de 2048 Muestras:

Tamaño de trama de 2048		
Ave	**%Acierto en tramas detectadas como aves**	**%Tramas Indeterminadas**
Palomas	100,0	36,7
Gaviotas	99,4	61,6
Lechuzas	100,0	43,7
Carboneros	96,8	45,2
Gorriones	97,8	21,7
Vencejos	98,6	7,2
Media	98,7	22,0

Tabla 7 Resultados de pruebas con tramas de 2048 muestras

No se siguen analizando más tamaños de trama ya que los resultados obtenidos con tramas de 2048 muertas son lo suficientemente óptimos.

4.2.2.2. Resultados del algoritmo de extracción de parámetros tras las pruebas

La caracterización de cada ave se calcula con los siguientes datos previamente seleccionados:

- Tamaño de trama 2048 muestras
- Filtro de potencia del 10%

Palomas

Habiendo analizado los ficheros de las palomas, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Paloma bravía - Columba livia_001	320	420	10	60	30	70	5	70	2	10
2	Paloma bravía - Columba livia_002	340	460	10	50	20	100	5	60	2	20
3	Paloma bravía - Columba livia_003	300	450	2	54	20	80	2	40	2	15
4	Paloma bravía - Columba livia_004	300	400	5	40	10	80	3	100	1	20
5	Paloma bravía - Columba livia_005	320	420	10	70	10	70	6	30	2	10
6	Paloma bravía - Columba livia_006	300	450	10	50	20	80	5	50	2	15
7	Paloma bravía - Columba livia_007	290	400	2	40	10	80	3	40	1	10
8	Paloma bravía - Columba livia_008	250	500	5	40	5	50	2	60	1	30
9	Paloma bravía - Columba livia_009	350	450	10	40	20	80	10	50	2	15
10	Paloma bravía - Columba livia_010	300	450	4	20	5	40	5	60	1	15

Tabla 8 Parámetros de las palomas

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

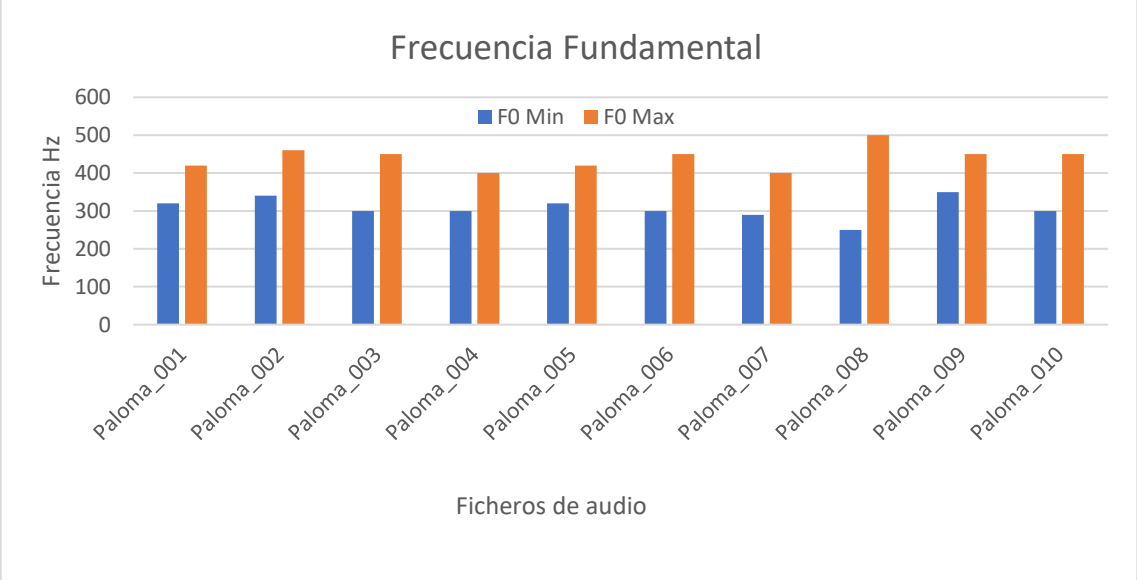


Ilustración 28 Frecuencias Fundamentales de la Paloma

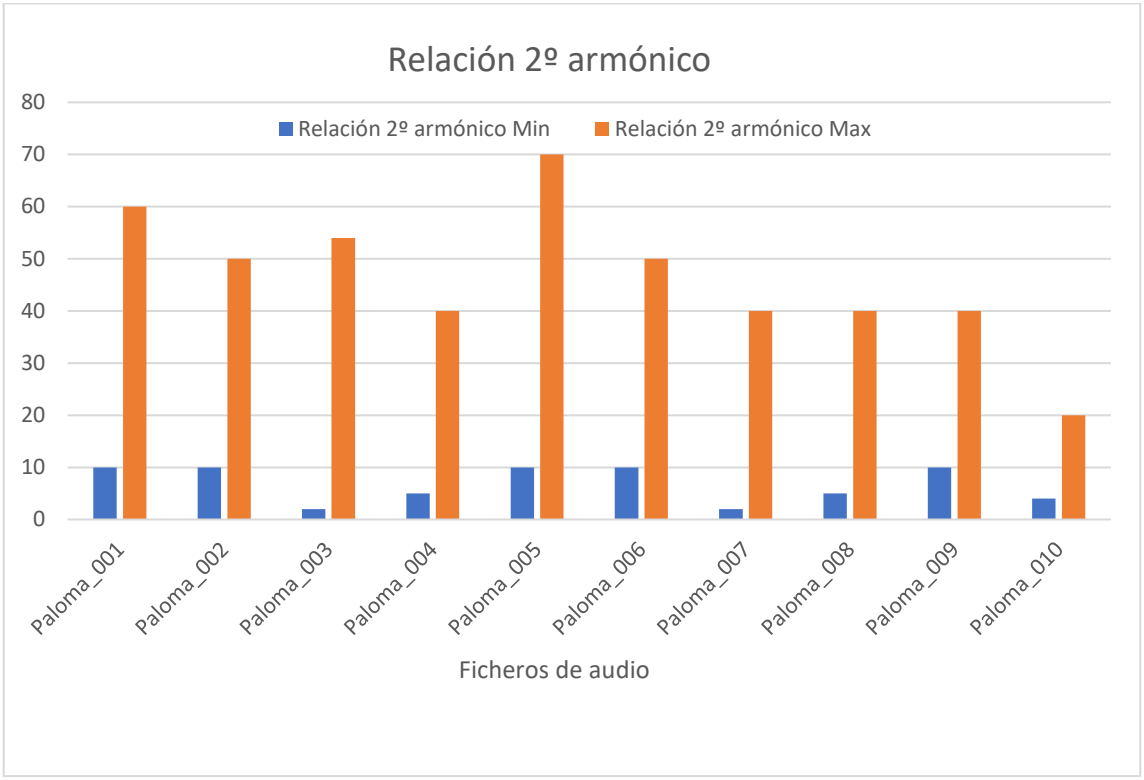


Ilustración 29 Relación del 2º armónico con la frecuencia fundamental de la Paloma

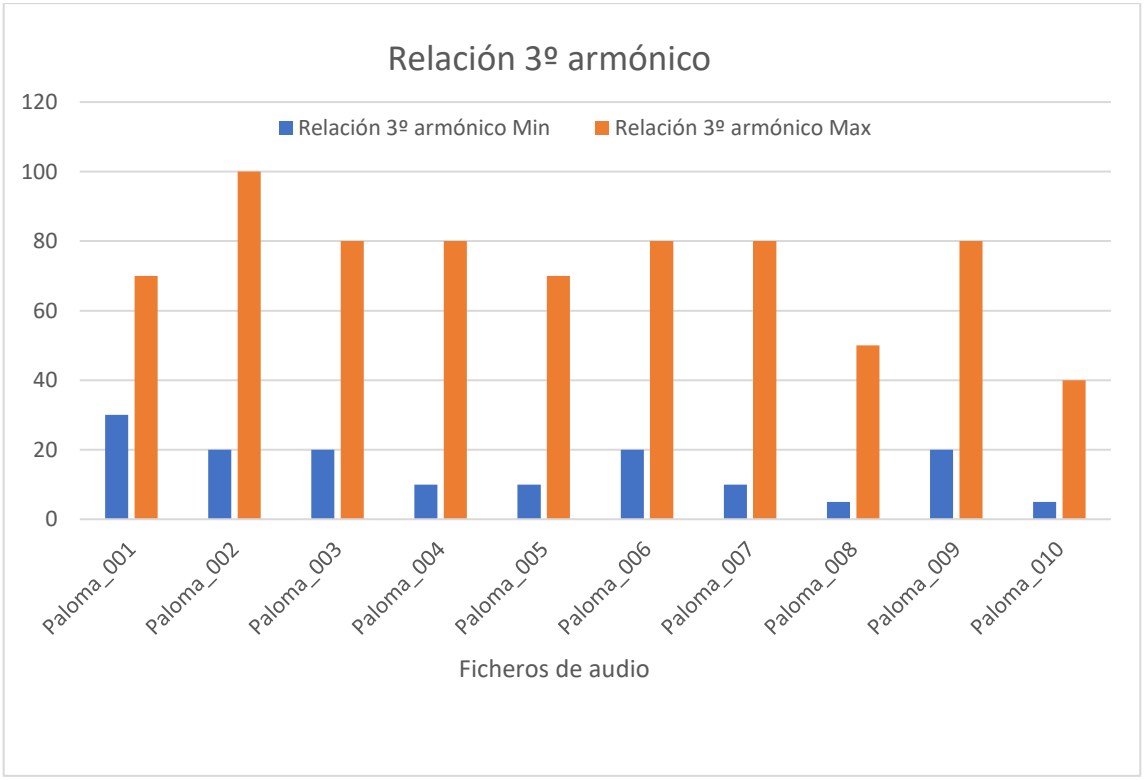


Ilustración 30 Relación del 3º armónico con la frecuencia fundamental de la Paloma

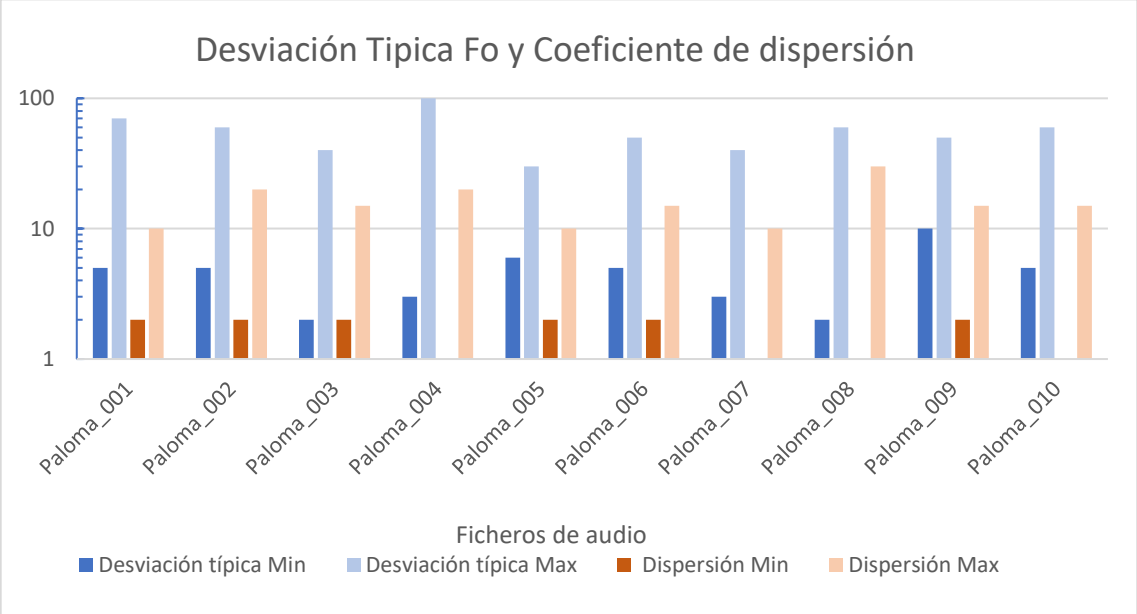


Ilustración 31 Desviación típica y coeficiente de dispersión de la Paloma

Conclusiones de las palomas:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las palomas con una frecuencia fundamental que estaría entre los 300Hz y 500Hz.

De igual manera nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 2 y 70 y de 3er armónico entre 5 y 80.

Una desviación típica de 3-100 y un Coeficiente de variación desde un 1% a un 35%

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

La paloma se caracteriza por ser el ave con más baja frecuencia fundamental de las aves analizadas.

Resumen de los parámetros sobre las palomas:

Media Frecuencias	**300Hz y 450Hz**
Media segundo armónico	3 y 60
Media tercer armónico	4 y 80
Desviación típica	3-100
Coeficiente de variación	0% a un 35%

Ilustración 32 Resumen de los parámetros de la Paloma

Gaviotas

Habiendo analizado los ficheros de las Gaviotas, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Gaviota _001	900	1500	2	25	3	100	2	130	1	20
2	Gaviota _002	900	1600	1	30	2	70	10	120	3	20
3	Gaviota _003	1100	1600	2	20	3	50	3	100	2	20
4	Gaviota _004	1000	1600	2	22	10	100	10	100	1	20
5	Gaviota _005	1000	1700	1	50	3	130	3	120	1	10
6	Gaviota _006	1000	1800	1	60	10	100	1	120	1	11
7	Gaviota _007	1000	1500	2	10	2	20	1	75	1	8
8	Gaviota _008	1000	1600	1	20	1	120	1	100	1	10
9	Gaviota _009	1200	1700	1	50	4	100	1	100	1	10
10	Gaviota _010	900	1400	3	15	1	20	1	60	1	5

Ilustración 33 Tabla sobre los parámetros de las Gaviotas

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

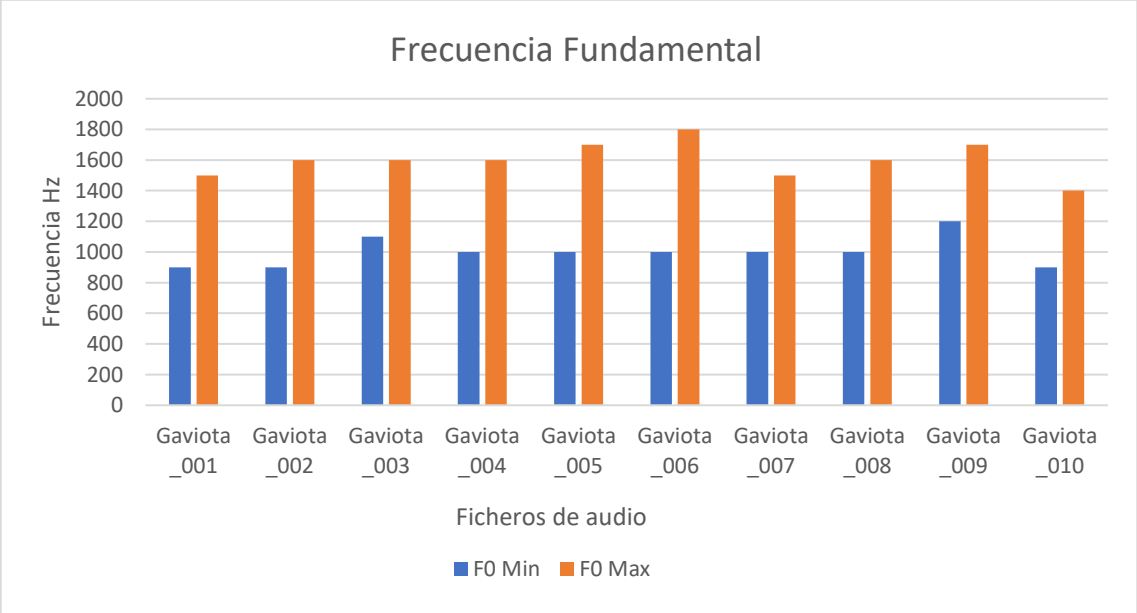


Ilustración 34 Frecuencias Fundamentales de la Gaviota

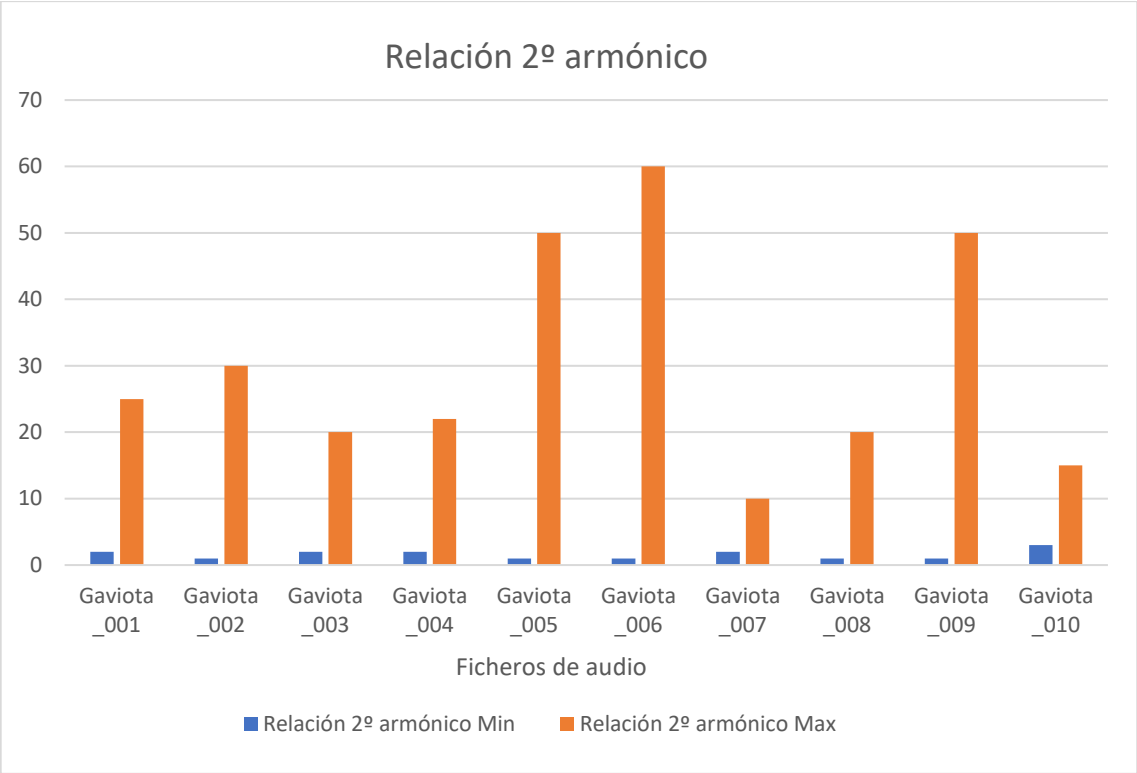


Ilustración 35 Relación del 2º armónico con la frecuencia fundamental de la Gaviota

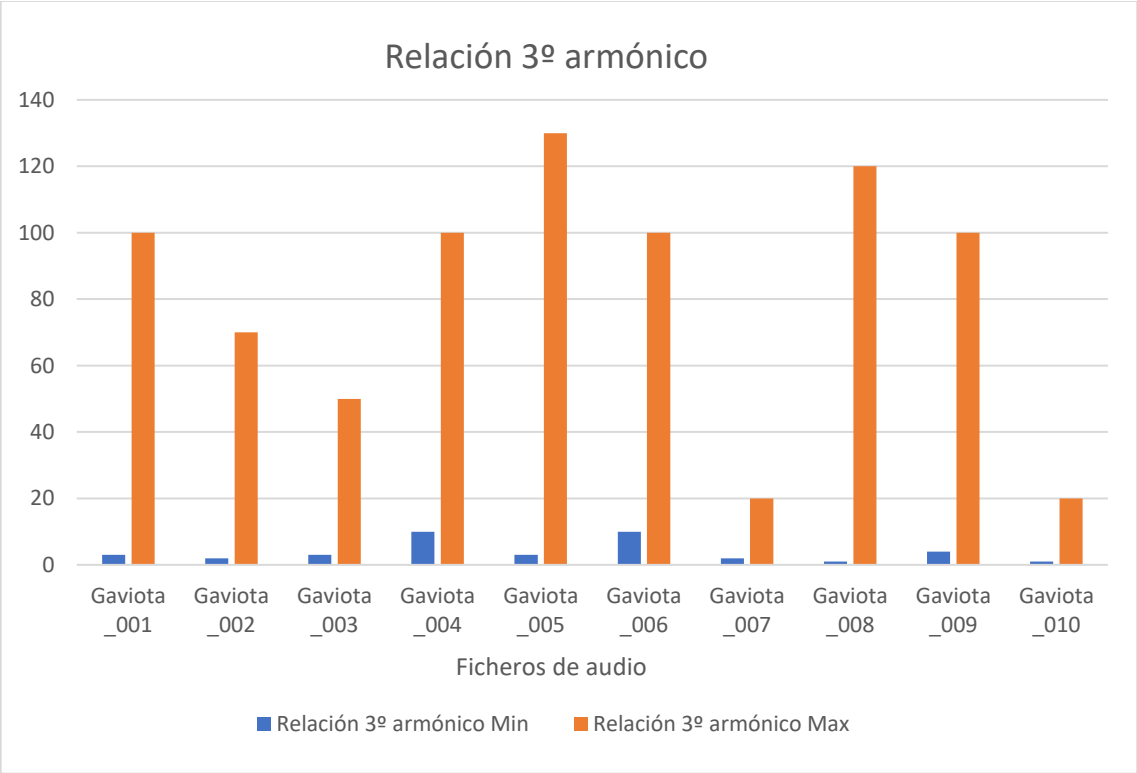


Ilustración 36 Relación del 3º armónico con la frecuencia fundamental de la Gaviota

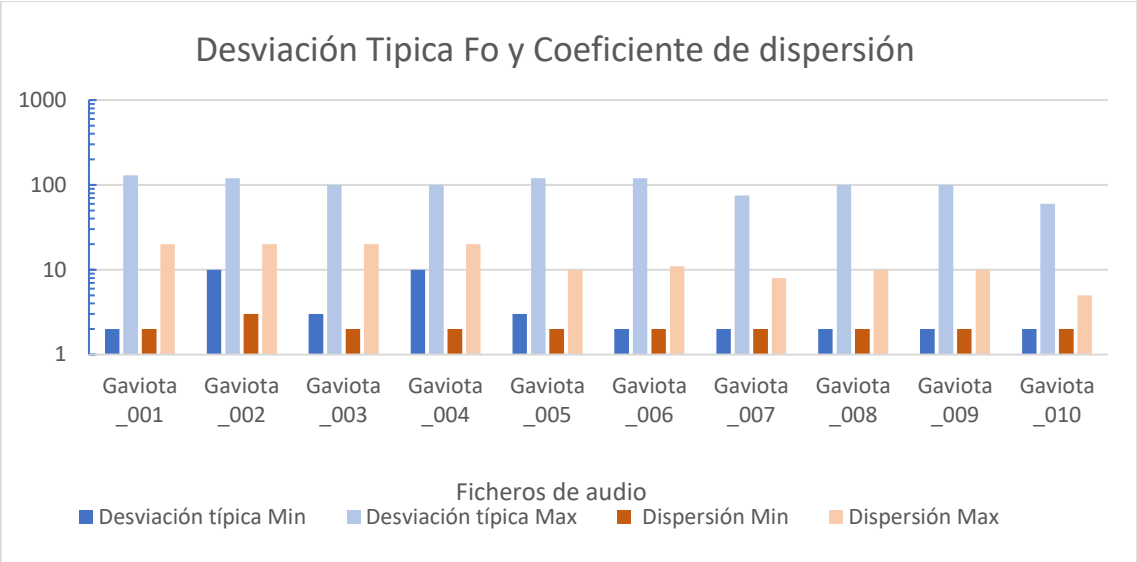


Ilustración 37 Desviación típica y coeficiente de dispersión de la Gaviota

Conclusiones de los Gaviotas:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las Gaviotas con una frecuencia fundamental que estaría entre los 1000Hz y 1700.

Esta ave se ve que comparte rango de frecuencias con las lechuzas, pero en una cantidad de tramas apenas significante para su identificación, por ello se decide ignorar estas frecuencias para la Gaviota, ya que en su mayoría las tramas están fuera de ese rango.

De igual manera nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 2 y 60 y de 3er armónico entre 3 y 120.

Una desviación típica de 2-100 y un Coeficiente de variación desde un 1% a un 20%

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

Resumen tras un estudio minucioso de las tramas irrelevantes para tener una identificación optima del ave. Los parámetros sobre las gaviotas:

Media Frecuencias	**1000Hz y 1500Hz**
Media segundo armónico	2 y 30
Media tercer armónico	10 y 100
Desviación típica	10-100
Coeficiente de variación	1% a un 25%

Tabla 9 Resumen de los parámetros de la Gaviota

Lechuzas

Habiendo analizado los ficheros de las Lechuzas, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Lechuza_común_001	1800	2100	20	80	10	100	10	150	1	10
2	Lechuza_común_002	1500	2500	5	60	10	150	1	200	1	20
3	Lechuza_común_003	1600	2300	5	40	10	70	1	100	1	10
4	Lechuza_común_004	1800	2400	7	50	30	150	1	80	1	5
5	Lechuza_común_005	1800	2900	0	50	10	150	2	200	1	10
6	Lechuza_común_006	2000	2500	30	60	20	150	10	115	1	10
7	Lechuza_común_007	1900	2500	5	70	10	150	5	150	1	15
8	Lechuza_común_008	1800	2300	20	60	5	100	5	100	1	5
9	Lechuza_común_009	1600	2300	10	50	10	60	3	200	1	8
10	Lechuza_común_010	2000	2300	30	90	80	150	30	200	1	8

Tabla 10 Parámetros de las Lechuzas

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

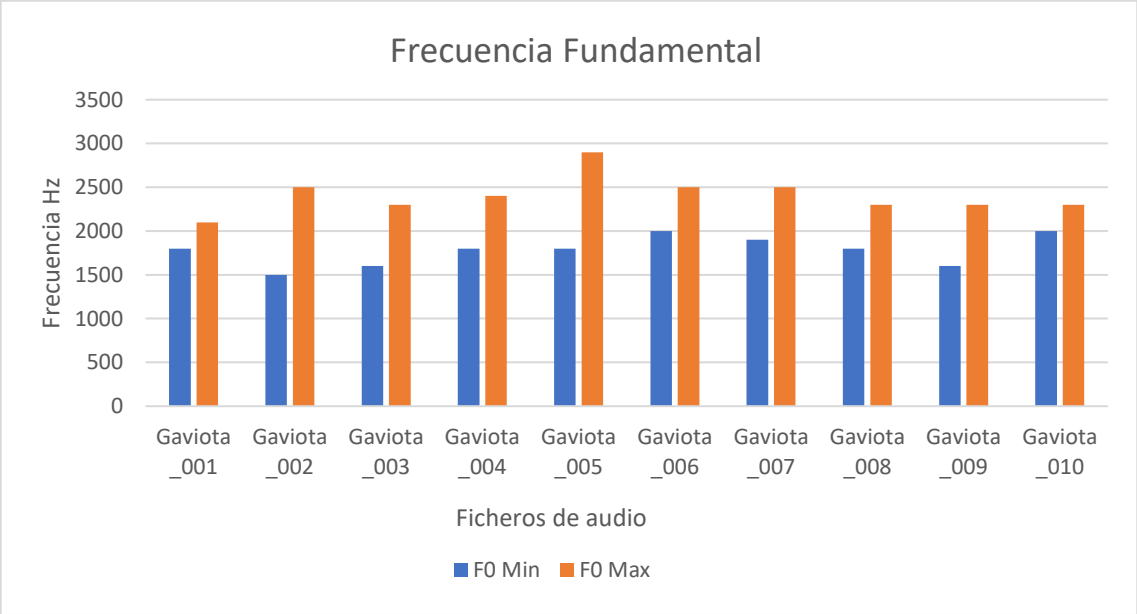


Ilustración 38 Frecuencias Fundamentales de la Lechuzas

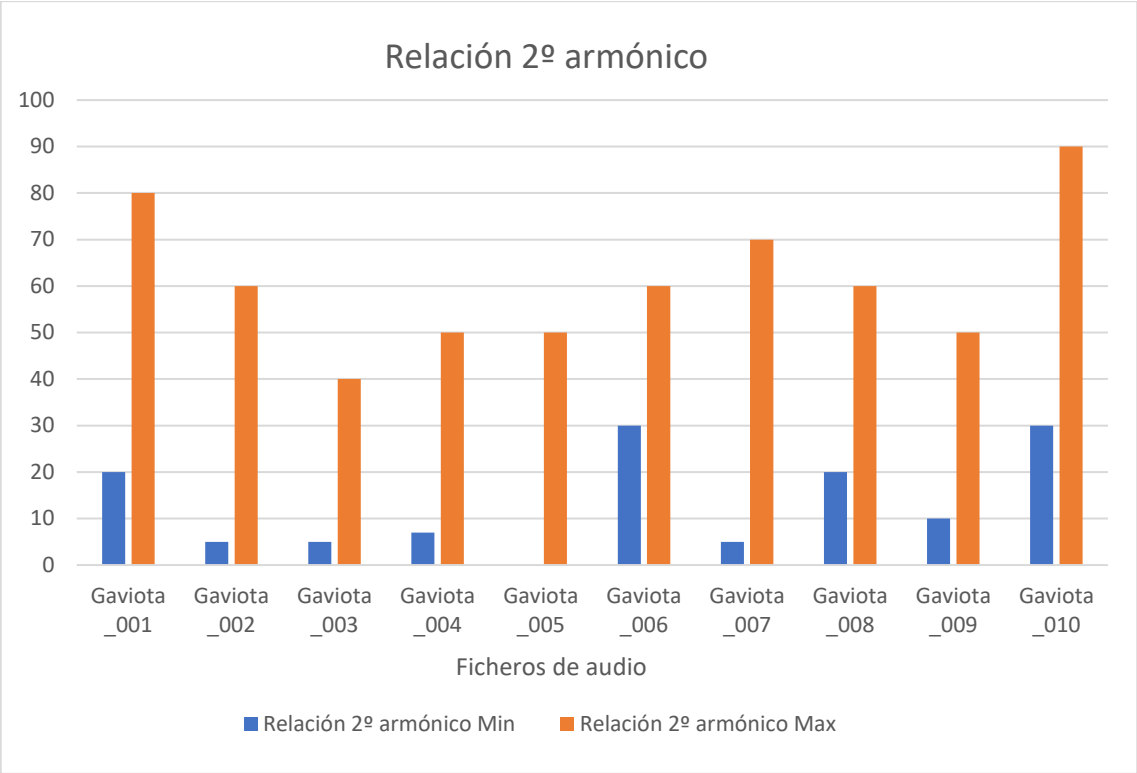


Ilustración 39 Relación del 2º armónico con la frecuencia fundamental de la Lechuza

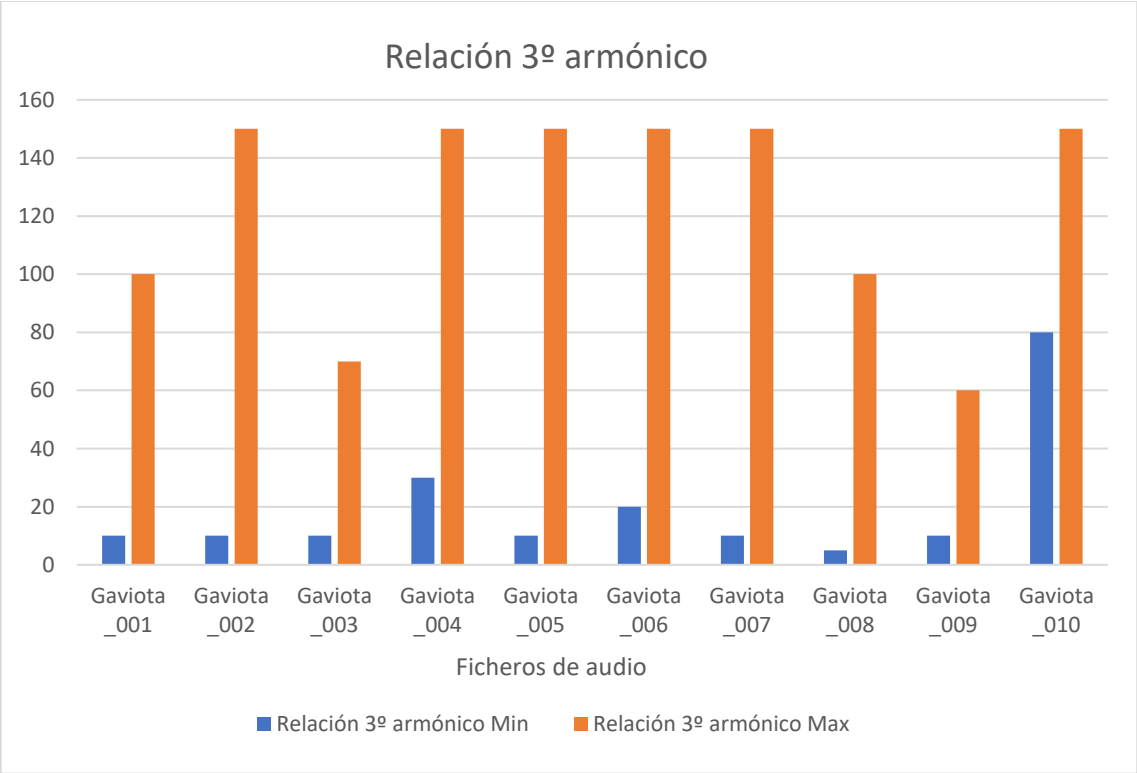


Ilustración 40 Relación del 3º armónico con la frecuencia fundamental de la Lechuza

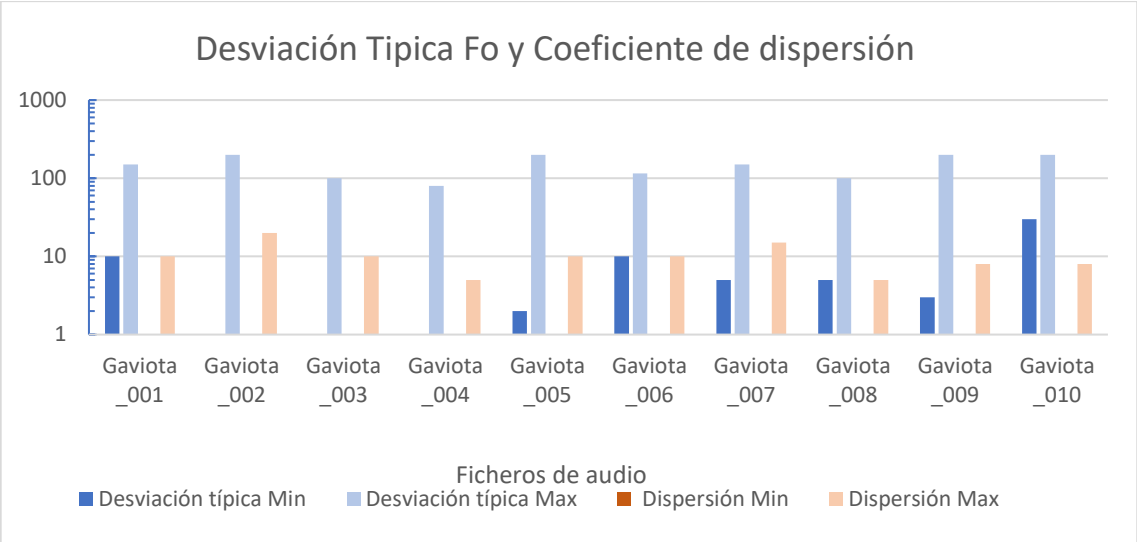


Ilustración 41 Desviación típica y coeficiente de dispersión de la lechuza

Conclusiones de las lechuzas:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las Gaviotas con una frecuencia fundamental que estaría entre los 1500Hz y 2900.

Esta ave con encontramos que comparte rango de frecuencias con las Gaviotas y los gorriones, pero en una cantidad de tramas apenas significante para su identificación, por ello se decide ignorar estas frecuencias para la lechuza, ya que en su mayoría las tramas están fuera de ese rango.

De igual manera nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 1 y 90 y de 3er armónico entre 5 y 150.

Una desviación típica de 5-200 y un Coeficiente de variación desde un 1% a un 20%

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

Resumen tras un estudio minucioso de las tramas irrelevantes para tener una identificación optima del ave. Los parámetros sobre las lechuzas:

Media Frecuencias	**1500Hz y 2900Hz**
Media segundo armónico	5 y 40
Media tercer armónico	10 y 150
Desviación típica	15-250
Coeficiente de variación	1% a un 20%

Tabla 11 Resumen de los parámetros de las lechuzas.

Gorriones

Habiendo analizado los ficheros de los Gorriones, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Gorrión_común_001	3000	5000	0	100	50	200	100	1000	2	30
2	Gorrión_común_002	2500	4500	10	50	10	150	2	500	0	60
3	Gorrión_común_003	2800	4500	20	80	40	300	10	400	0	30
4	Gorrión_común_004	2800	4600	10	90	40	300	5	2000	0	120
5	Gorrión_común_005	3000	5000	5	100	10	300	6	2000	0	60
6	Gorrión_común_006	3000	5000	5	70	40	200	100	1200	5	50
7	Gorrión_común_007	2500	5000	20	150	60	300	10	1500	1	40
8	Gorrión_común_008	3000	5000	20	120	100	500	100	2000	1	20
9	Gorrión_común_009	3200	4800	20	110	40	250	100	1500	3	40
10	Gorrión_común_010	2500	4500	8	100	50	400	200	1500	2	50

Tabla 12 Parámetros del gorrión

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

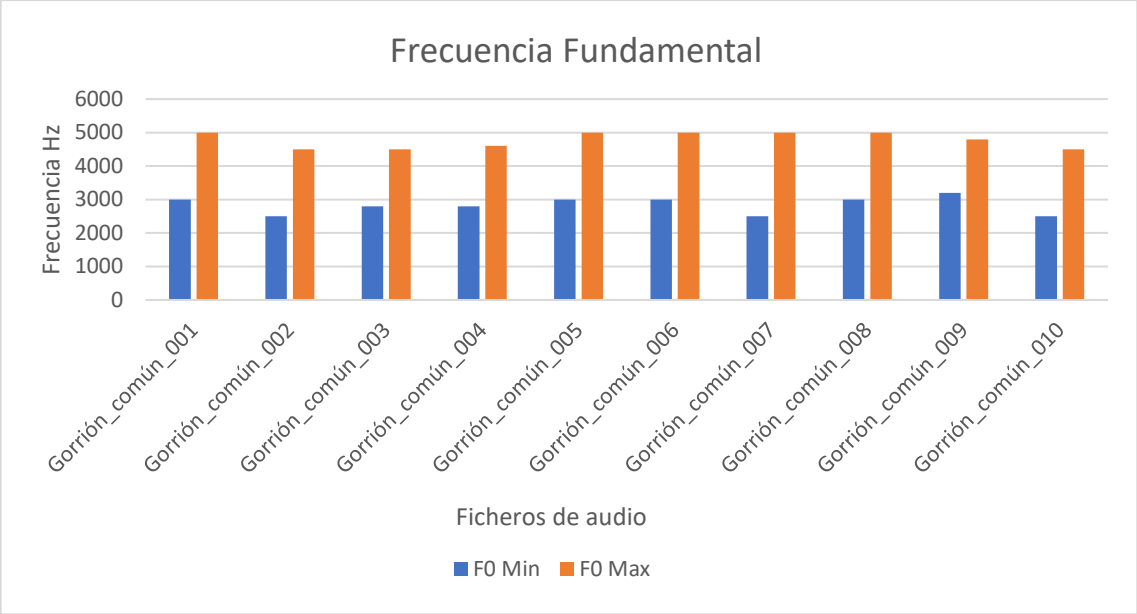


Ilustración 42 Frecuencias Fundamentales del Gorrión

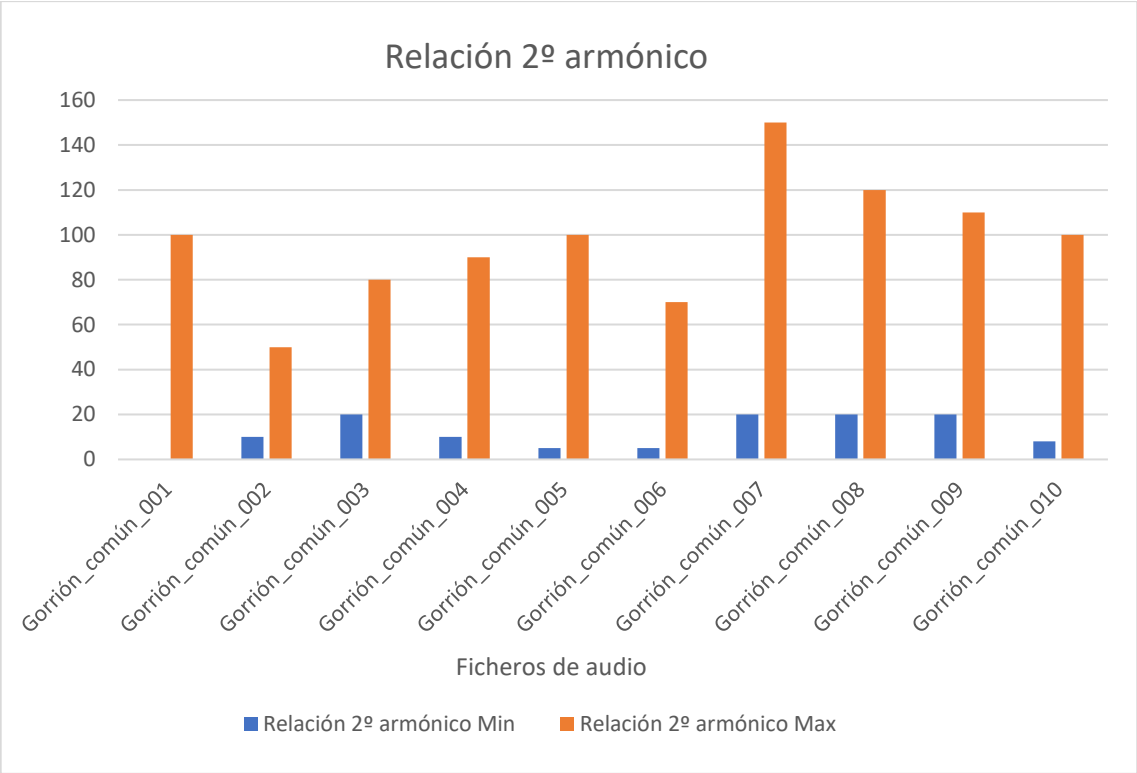


Ilustración 43 Relación del 2º armónico con la frecuencia fundamental del Gorrión

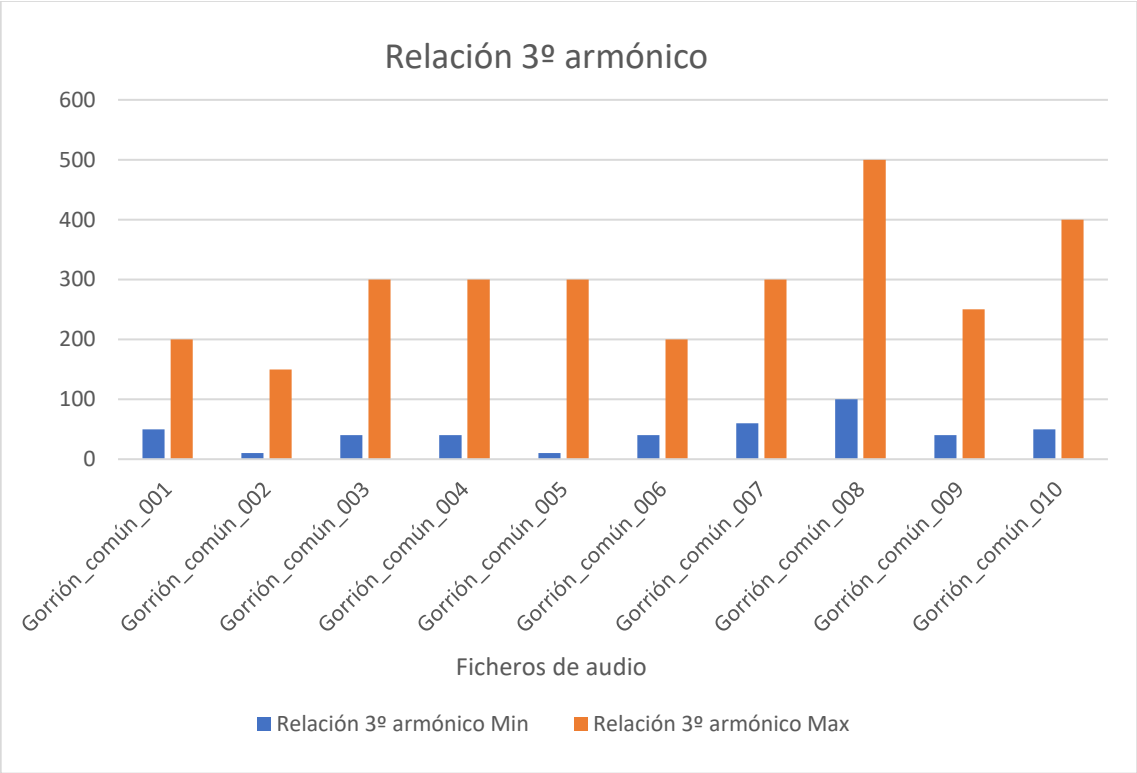


Ilustración 44 Relación del 3º armónico con la frecuencia fundamental del Gorrión

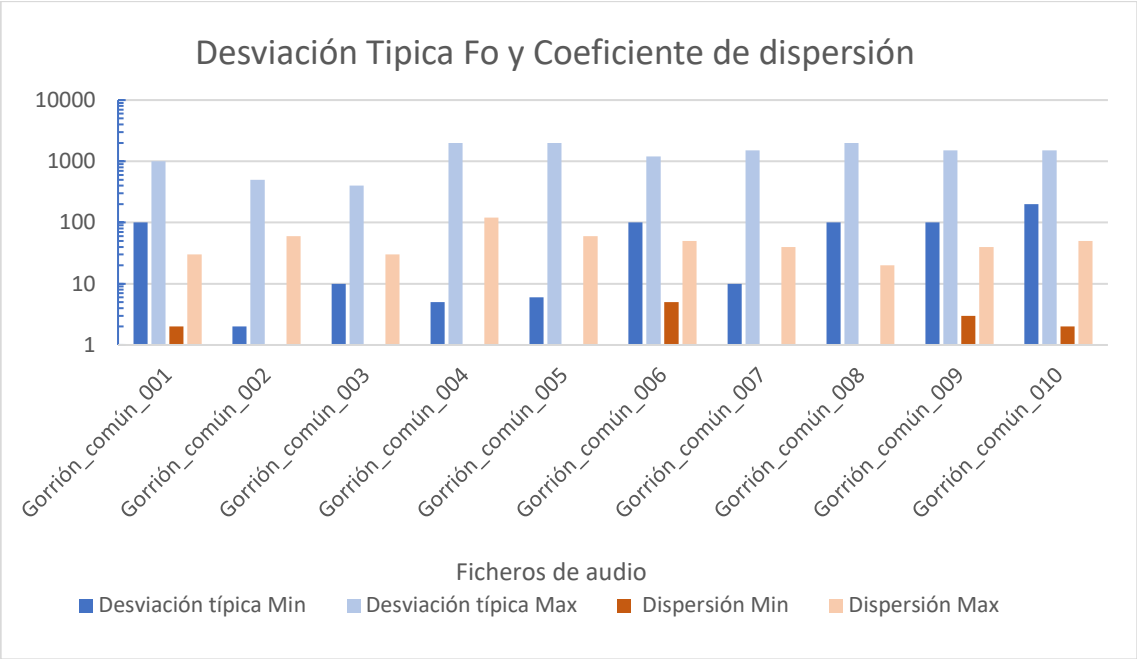


Ilustración 45 Desviación típica y coeficiente de dispersión del Gorrión

Conclusiones de los Gorriones:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las Gaviotas con una frecuencia fundamental que estaría entre los 2500Hz y 5000.

Este ave con encontramos que comparte rango de frecuencias con las Lechuzas y los Carboneros, pero en una cantidad de tramas apenas significante para su identificación, por ello se decide ignorar estas frecuencias para la lechuza, de lo contrario con el Carboneros nos encontramos que no es posible ignorar ese conflicto en la frecuencia fundamental y que sea viable la identificación de las aves, por lo que utilizaremos la diferencia de la relación con el segundo y tercer armónico para afinar más la identificación así como las desviación típica y el coeficiente de dispersión.

Nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 10 y 100 y del tercer armónico entre 20 y 500.

Una desviación típica de 10-2000 y un Coeficiente de variación desde un 1% a un 120%, observando en estos unos altos valores, diferenciados respecto a las aves analizadas.

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

Resumen tras un estudio minucioso de las tramas irrelevantes para tener una identificación optima del ave. Los parámetros sobre los Gorriones:

Media Frecuencias	3000Hz y 5000Hz
Media segundo armónico	10 - 100
Media tercer armónico	20 - 50
Desviación típica	10 - 2000
Coeficiente de variación	1% a un 120%

Tabla 13 Resumen de los parámetros de los gorriones

Carbonero

Habiendo analizado los ficheros de los Carboneros, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Carbonero_001	3000	5000	90	250	200	600	10	200	0	5
2	Carbonero_002	3000	4500	100	300	200	600	10	200	0	5
3	Carbonero_003	3800	4800	90	300	100	500	10	100	0	5
4	Carbonero_004	4000	5000	100	400	200	600	10	100	0	5
5	Carbonero_005	3000	6000	150	300	100	1000	10	100	0	5
6	Carbonero_006	3500	6000	90	300	100	500	10	150	0	5
7	Carbonero_007	3500	5000	100	350	200	800	5	100	0	4
8	Carbonero_008	3000	5000	100	300	100	600	10	100	0	5
9	Carbonero_009	3500	6000	90	350	100	1000	10	100	0	5
10	Carbonero_010	3500	6000	100	400	100	800	10	150	0	5

DS		CV	
Min	Max	Min	Max
800	1500	15	30
400	1000	10	25
400	1000	10	15
400	800	10	20
500	1200	10	35
400	100	10	35
500	1500	10	35
400	1200	10	30
400	800	10	30

Tabla 14 Parámetros del carbonero

En esta ave nos encontramos dos zonas diferenciadas significativamente por si desviación típica, una marcada en gris y la otra opción marcada en azul, por eso la agrandamos la tabla dependiendo de ella.

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

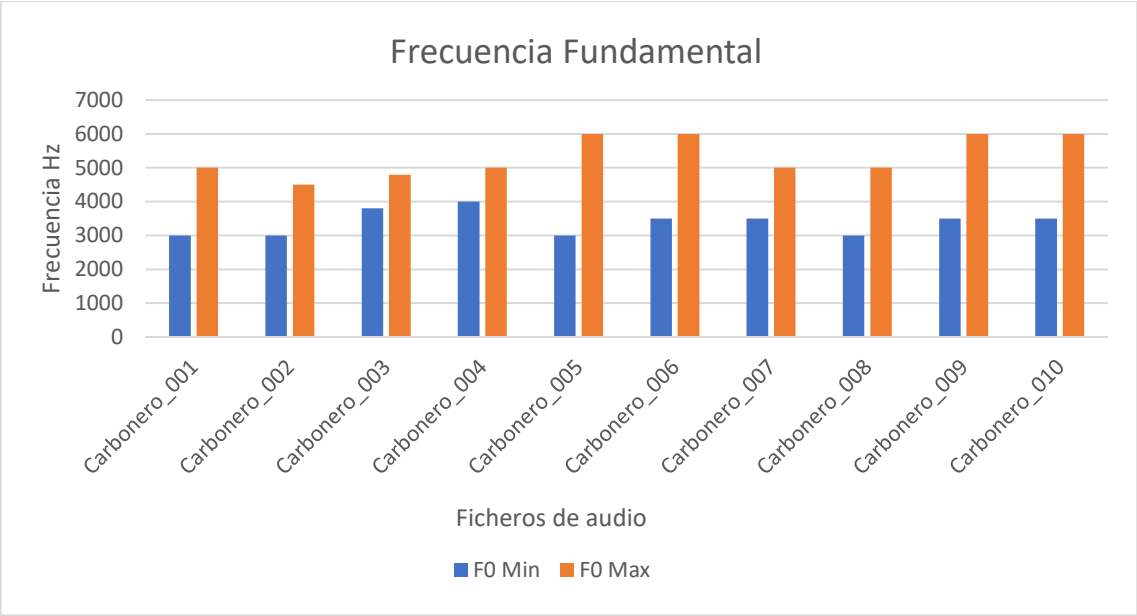


Ilustración 46 Frecuencias Fundamentales del carbonero

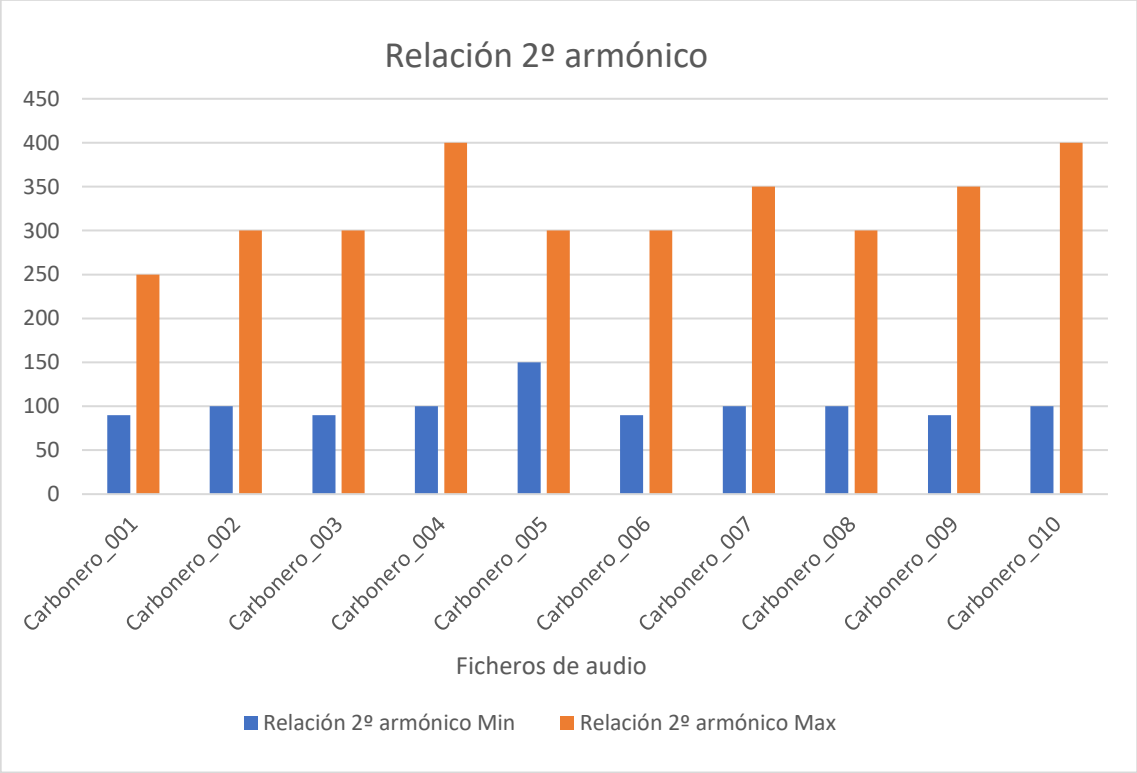


Ilustración 47 Relación del 2º armónico con la frecuencia fundamental del Carbonero

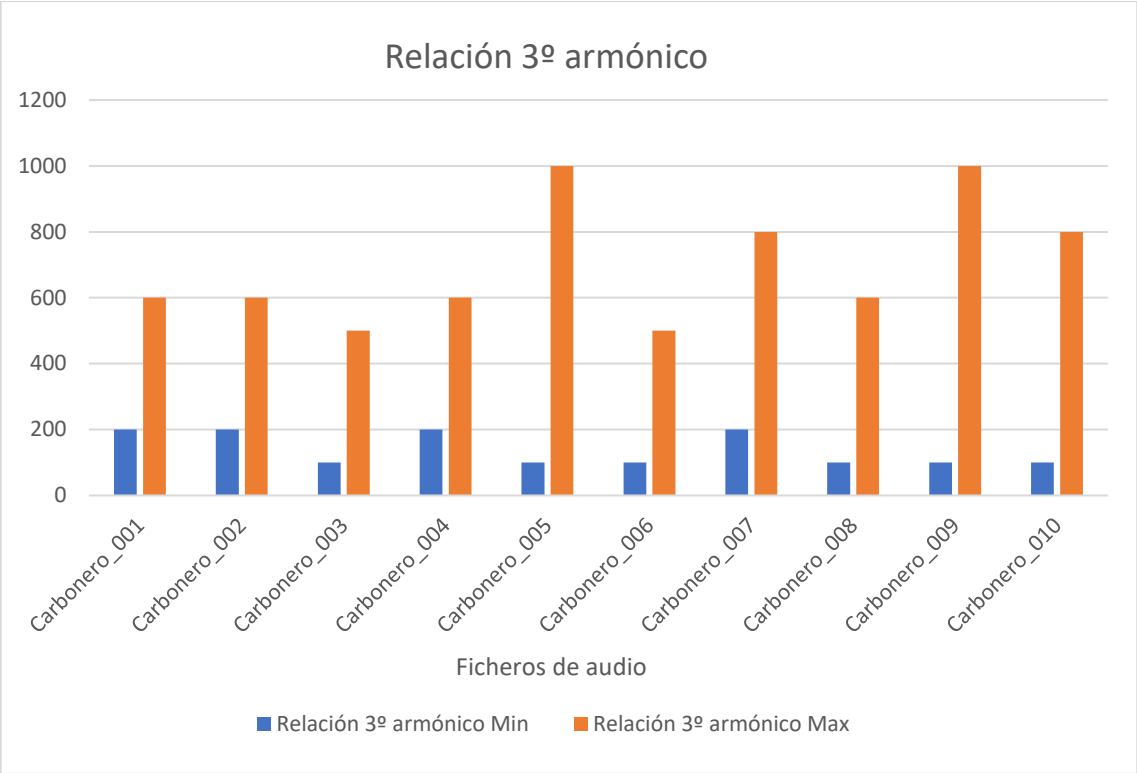


Ilustración 48 Relación del 3º armónico con la frecuencia fundamental del Carbonero

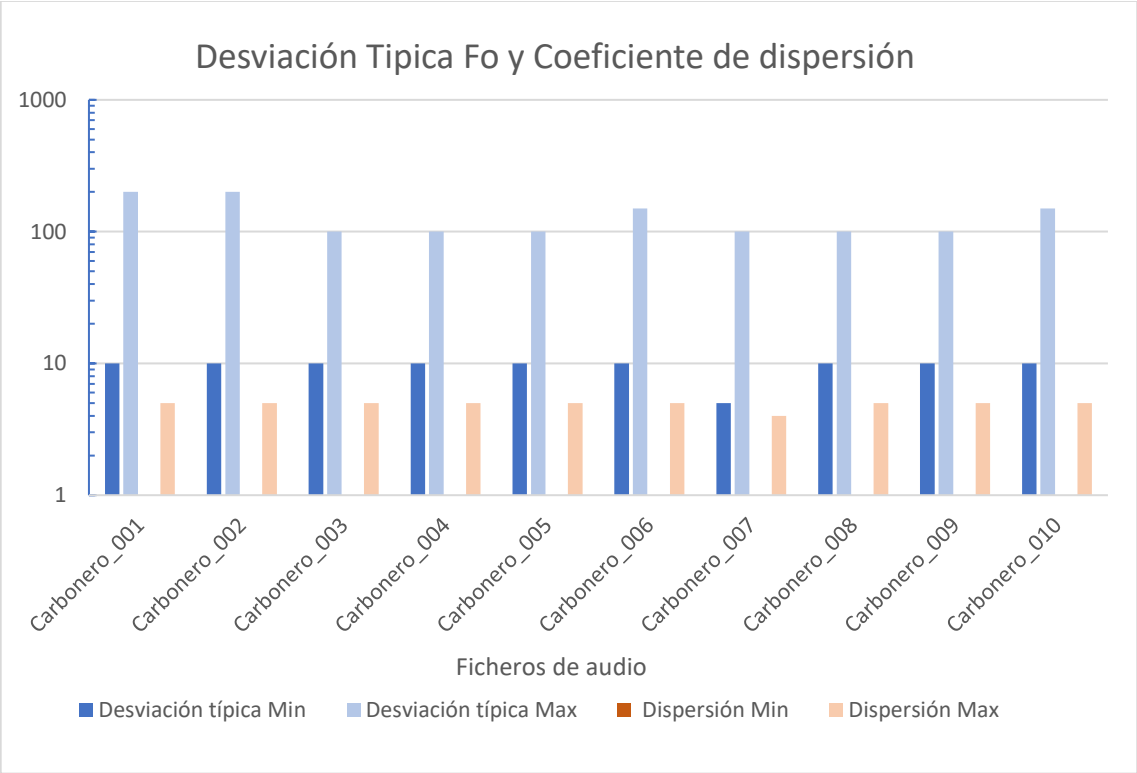


Ilustración 49 Desviación típica y coeficiente de dispersión del Carbonero

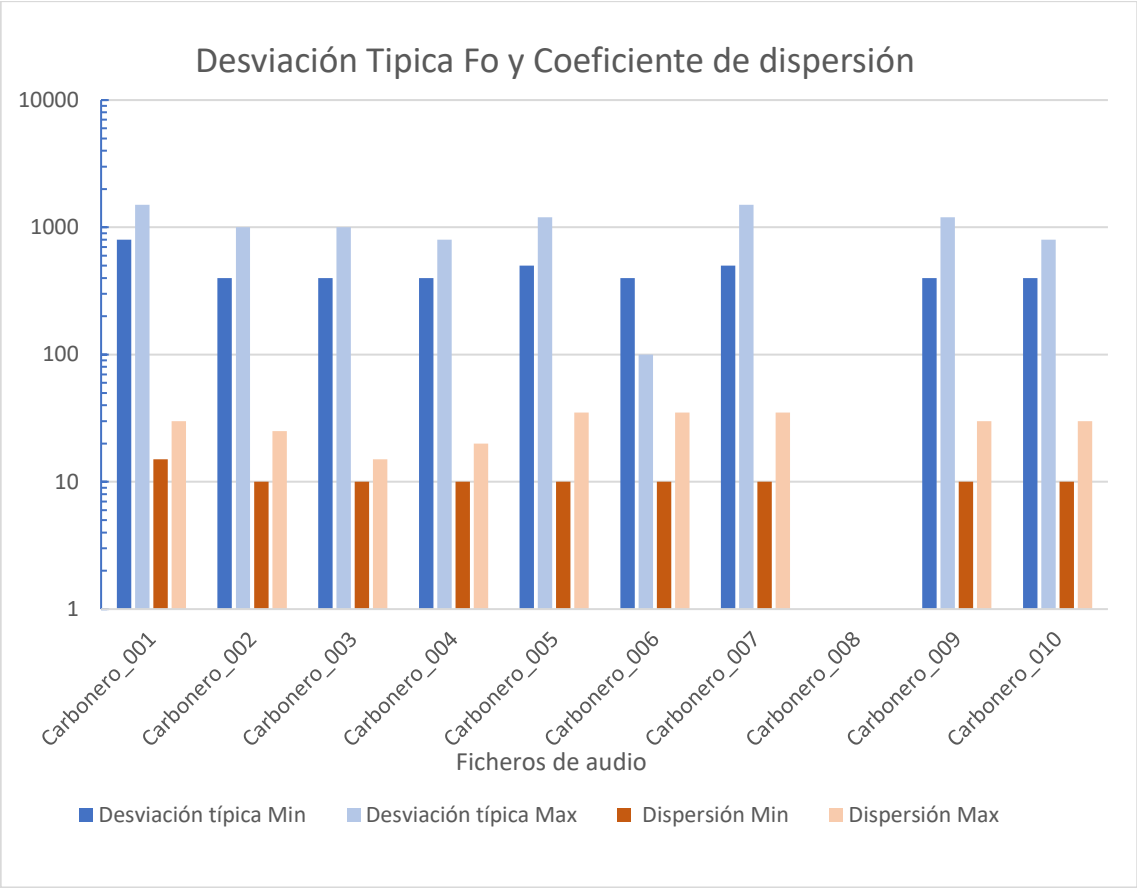


Ilustración 50 Desviación típica y coeficiente de dispersión del carbonero

Conclusiones de los carboneros:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las gaviotas con una frecuencia fundamental que estaría entre los 3000Hz y 6000.

Esta ave con encontramos que comparte rango de frecuencias con las Los Gorriones y Vencejos, no es posible ignorar ese conflicto en la frecuencia fundamental y que sea viable la identificación de las aves, por lo que utilizaremos la diferencia de la relación con el segundo y tercer armónico para afinar más la identificación, así como la desviación típica y el coeficiente de dispersión.

Nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 90 y 300 y del tercer armónico entre 100 y 1000.

Una desviación típica que se puede diferenciar en dos rangos, de 10-120 y con un coeficiente de dispersión del 0 - 5 % y una desviación típica de 400 a 1500 con un Coeficiente de variación desde un 10% a un 30%.

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

Resumen tras un estudio minucioso de las tramas irrelevantes para tener una identificación optima del ave. Los parámetros sobre los Carboneros:

Media Frecuencias	**3000Hz y 6000Hz**
Media segundo armónico	100 - 300
Media tercer armónico	100 - 700
Desviación típica	10-120 // 400-1500
Coeficiente de variación	0 -5 % // 10 – 30 %

Tabla 15 Resumen de los parámetros de los carboneros

Vencejo

Habiendo analizado los ficheros de los vencejos, se obtienen los siguientes parámetros.

Nº	Audio	Fo		Relación 2ºArmónico con Fo		Relación 3er Armónico con Fo		DS		CV	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
1	Vencejo_comun_001	5000	7000	10	250	100	1000	20	300	1	5
2	Vencejo_comun_002	5000	6500	10	200	100	1000	10	300	1	6
3	Vencejo_comun_003	5000	7000	10	150	100	1000	2	300	0	8
4	Vencejo_comun_004	5000	6500	7	150	100	1000	2	400	0	8
5	Vencejo_comun_005	4800	6700	10	100	50	600	2	400	0	8
6	Vencejo_comun_006	5000	6800	10	100	90	1000	5	300	2	8
7	Vencejo_comun_007	4800	7000	10	100	100	1000	3	400	0	8
8	Vencejo_comun_008	5000	7000	10	150	100	1000	4	400	0	9
9	Vencejo_comun_009	4500	7000	10	100	100	800	2	250	0	8
10	Vencejo_comun_010	5000	6500	20	200	100	500	4	250	0	6

Tabla 16 Parámetros del vencejo

A continuación, mostraremos de forma visual los parámetros obtenidos en la tabla.

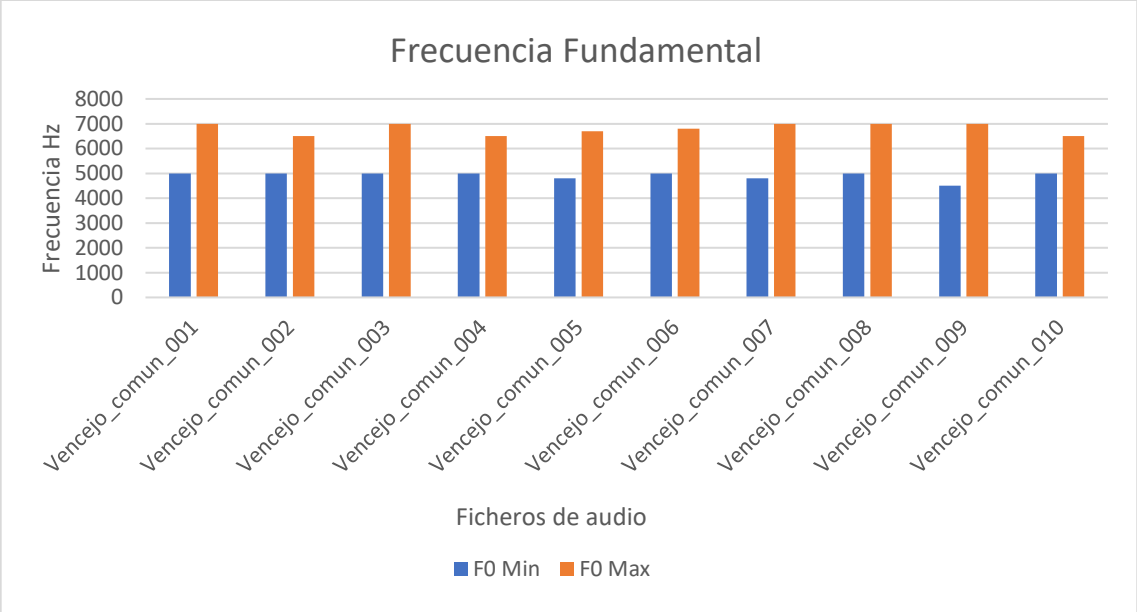


Ilustración 51 Frecuencias Fundamentales del vencejo

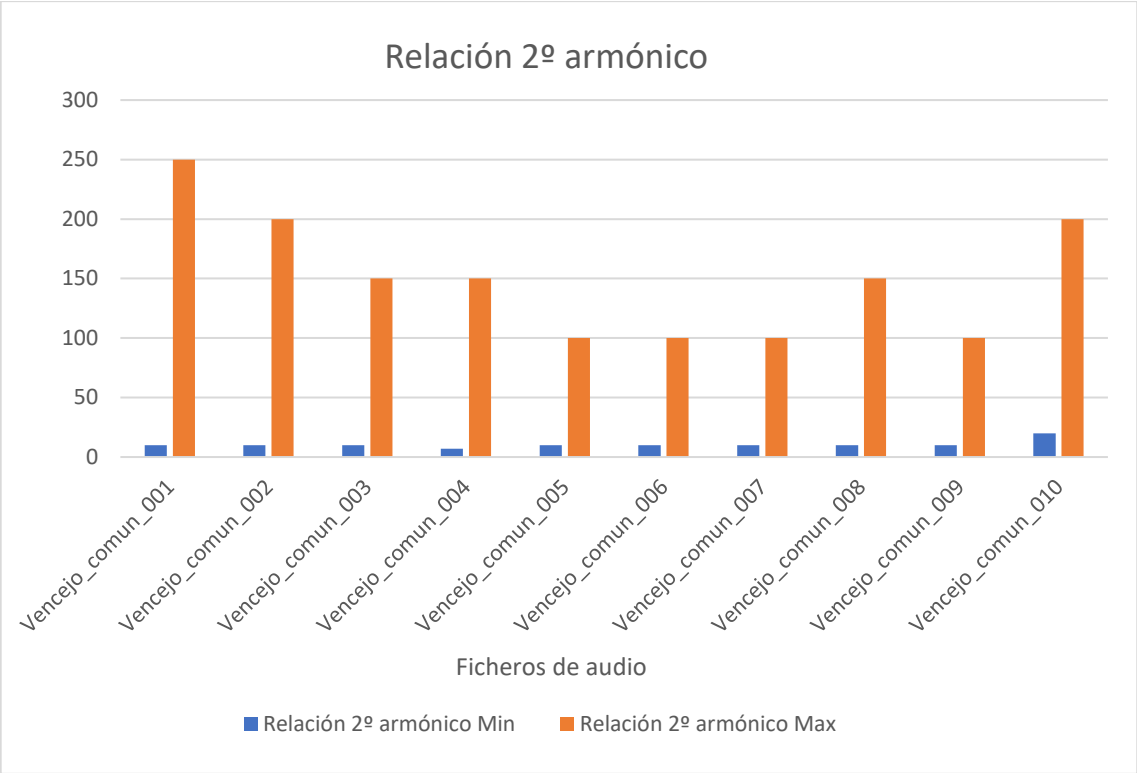


Ilustración 52 Relación del 2º armónico con la frecuencia fundamental del vencejo

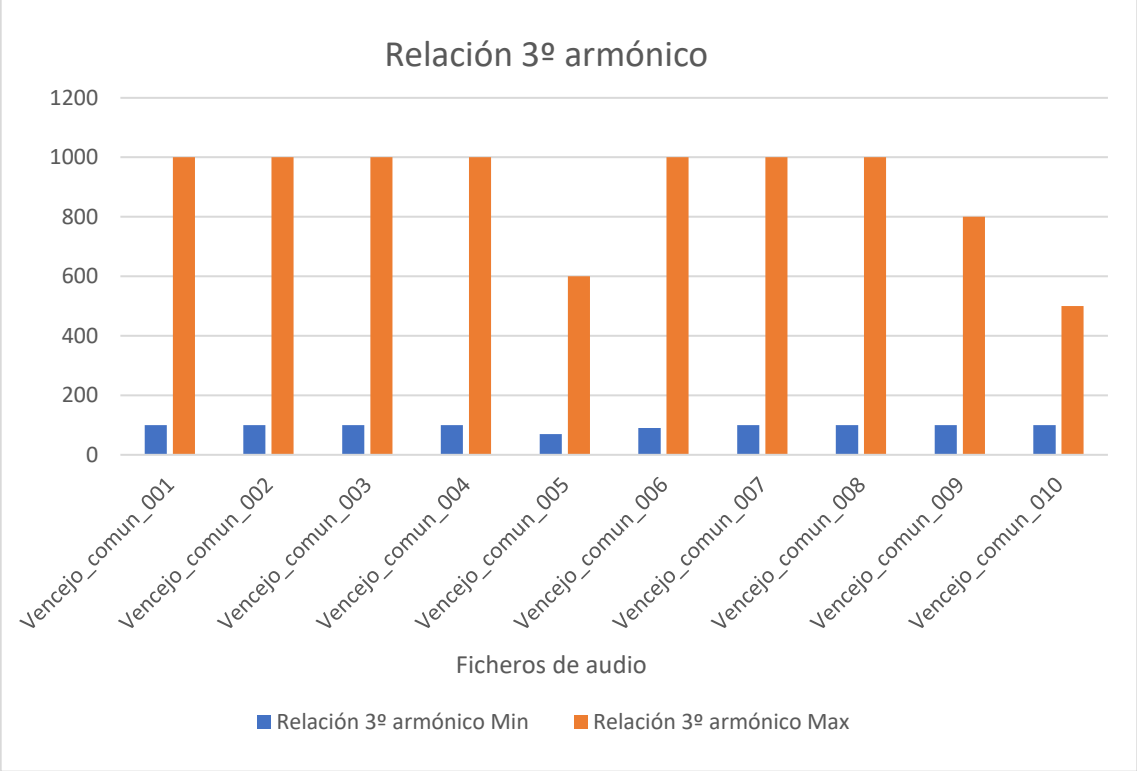


Ilustración 53 Relación del 3º armónico con la frecuencia fundamental del vencejo

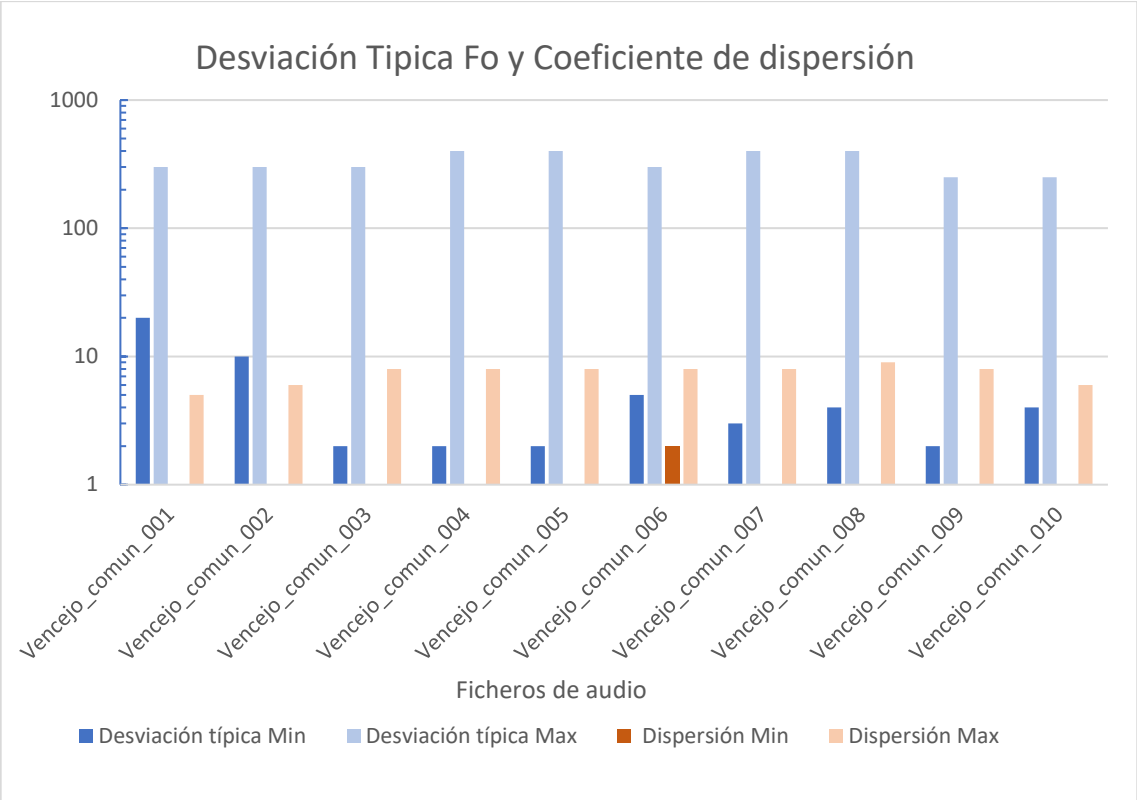


Ilustración 54 Desviación típica y coeficiente de dispersión del vencejo

Conclusiones de los vencejos:

Basándonos en la tabla y gráficos mostrados, podemos caracterizar el sonido de las gaviotas con una frecuencia fundamental que estaría entre los 4800Hz y 7000.

Este ave con encontramos que comparte rango de frecuencias con los gorriones y carboneros, con los gorriones el número de tramas es insignificante pudiendo obviar estas frecuencias, pero no es posible ignorar ese conflicto en la frecuencia fundamental a lo que el carbonero se refiere y que sea viable la identificación de las aves, por lo que utilizaremos la diferencia de la relación con el segundo y tercer armónico para afinar más la identificación así como las desviación típica y el coeficiente de dispersión.

Nos encontramos con una relación de entre la frecuencia fundamental y el segundo armónico que varía entre 10 y 250 y del tercer armónico entre 100 y 1000.

Una desviación típica de 2 y 500 con un Coeficiente de variación desde un 0% a un 8%.

Dejaremos fuera del algoritmo de manera excepcional algunos valores anormales que su eliminación no repercute en falsos positivos.

Resumen tras un estudio minucioso de las tramas irrelevantes para tener una identificación óptima del ave. Los parámetros sobre los vencejos:

Media Frecuencias	**5000Hz y 7000Hz**
Media segundo armónico	10 - 250
Media tercer armónico	100 - 1000
Desviación típica	2-400
Coeficiente de variación	0 -10 %

Tabla 17 Resumen de los parámetros de los vencejos

Campana de Gauss sobre la f_o de todas las aves

En líneas generales, la caracterización de todas las aves la podemos observar en la siguiente figura, que muestra la campana de Gauss, con las frecuencias fundamentales y desviaciones de todos los audios utilizados.

Se puede ver como las palomas (amarilla) son las aves con la f_o más baja estando entre 300Hz y 600Hz.

En las frecuencias superiores a 1000Hz hasta 3000 Hz se ve como coinciden ciertas aves como la gaviota (azul), la lechuza (rojo) y el gorrión (rosa). Sin embargo, hay una diferencia en el ancho de la campana de las aves, aunque compartan mismo rango de frecuencias.

Respecto a las frecuencias de 3000Hz a 7000Hz sucede con el vencejo (azul), el gorrión (rosa) y el carbonero (verde)

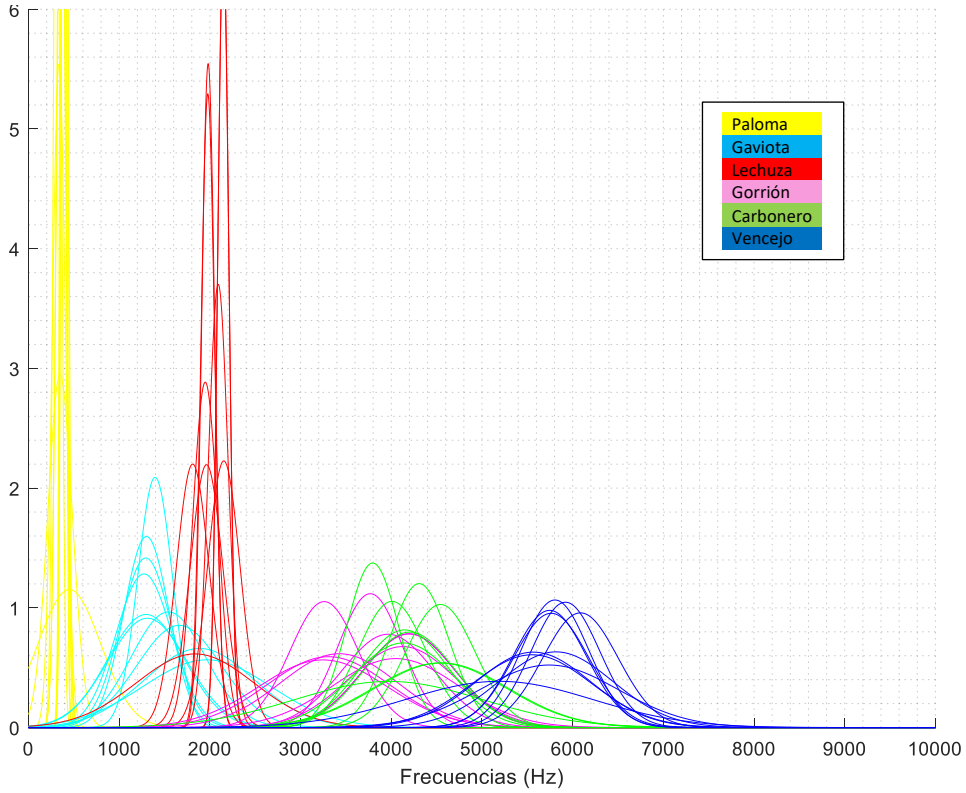


Ilustración 55 Campana de Gauss sobre la f_0 de todas las aves

4.2.3. Algoritmo final de identificación y confirmación

Con los resultados de la caracterización de las aves se crea un algoritmo de identificación que permita diferenciar cada ave dentro de Matlab. Se realizan pruebas durante todo el proceso hasta conseguir una identificación elevada.

Revisando los valores y ajustando el algoritmo de identificación para que tenga el mejor % de eficiencia posible, los criterios obtenidos serían los siguientes.

	Frecuencia Fundamental	**Relación segundo armónico**	**Relación tercer armónico**	**Desviación típica**	**Coeficiente de variación**
Paloma	300-400Hz	3 a 60	4 a 80	10 a 130	5 a 35%
Gaviota	1000-1500Hz	2 a 30	10 a 100	10-100	1 a 25%
Lechuza	1500-2900Hz	5 a 40	10 a 150	26 a 250	1 a 20%
Gorrión	3000-5000Hz	10-100	20 a 50	10 a 2000	1 a 120%
Carbonero	3000-6000Hz	100-300	100-700	400-1500	10-30%
Vencejo	5000-7000Hz	10-250	100-1000	2-400	1-10%

Tabla 18 Tabla sobre los parámetros generales de todas las aves

A continuación, expondremos un diagrama de flujo de como realizamos la identificación de las aves y posteriormente su confirmación.

En el diagrama utilizaremos los siguientes acrónimos para facilitar el tamaño.

- Frecuencia Fundamental → Fo
- Relación frecuencia fundamental con segundo armónico→ R2a
- Relación frecuencia fundamental con tercer armónico→ R3a
- Desviación típica de la Fo→ DS
- Coeficiente de dispersión de la Fo→ %

En este diagrama, se observa 7 divisiones de la frecuencia, si cumple los requisitos identificación del ave correspondiente, si no los cumple no realiza ninguna identificación y se contabiliza como Indeterminada.

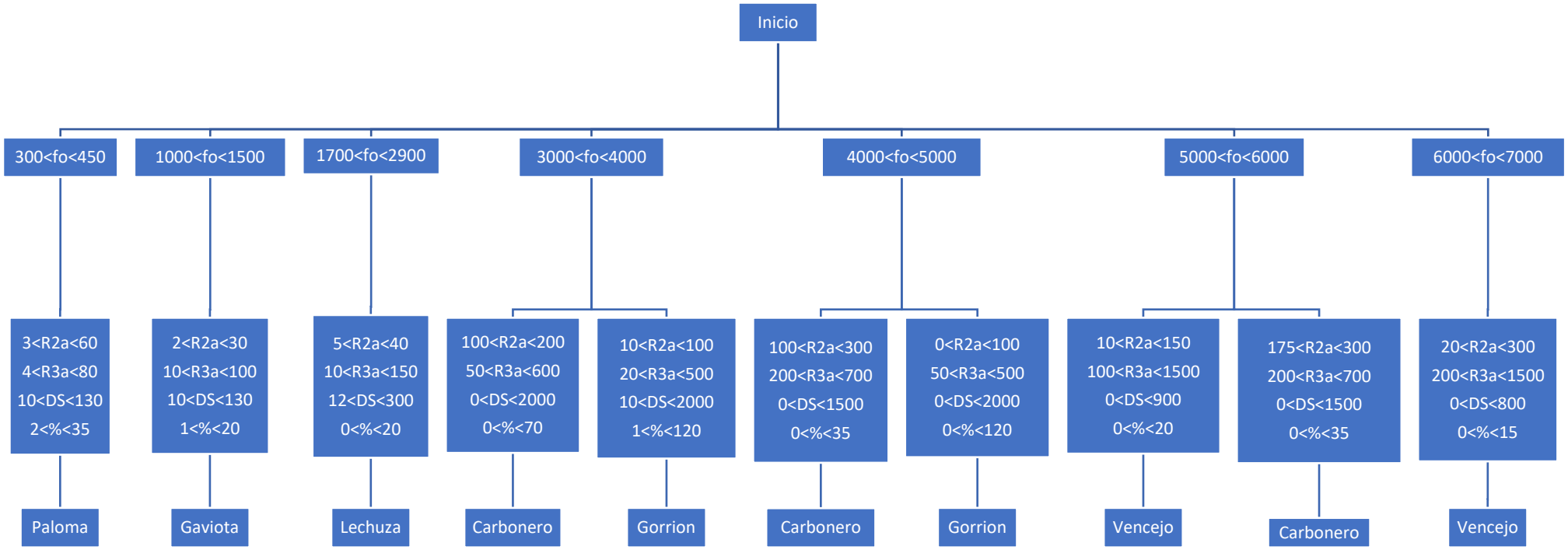


Ilustración 56 Diagrama del algoritmo de identificación.

Confirmación del ave identificada.

Tras haber identificado las tramas pertenecientes a cada ave, se detectó que en ciertas ocasiones el algoritmo confundía una trama con otra.

Para reducir al máximo estos fallos, se añade al algoritmo de identificación un factor de confirmación de 10, que consiste en observar las 10 últimas aves identificadas, y validar la que más se repita de todas ellas.

Se ha seleccionado el valor de 10 tramas como factor de confirmación por las conclusiones obtenidas en las siguientes pruebas:

Factor de confirmación, Pruebas hechas con los valores 1,5 y 10.

Pruebas con un factor de confirmación de 1: la propia trama identificada es confirmada. No puede haber un valor más bajo.

Se observa como confirmamos aves erróneas en 4 de las 6 aves, por culpa de identificaciones erróneas aleatorias, estas pueden eliminarse aumentando el factor de confirmación.

Factor de confirmación de 1		
Ave	**Acierto en tramas detectadas como aves**	**% De confirmaciones correctas**
Palomas	100,0	100,0
Gaviotas	97,4	97,4
Lechuzas	100,0	100,0
Carboneros	96,0	96,0
Gorriones	91,6	91,6
Vencejos	96,3	96,3
Media	96,8	96,8

Tabla 19 Resultados con un factor de confirmación =1

Pruebas con un factor de confirmación de 5: Se observan las 5 ultimas tramas para asegurar el ave identificada.

Podemos observar cómo aumenta el valor de las confirmaciones con respecto a las tramas detectadas como ave. Esto se debe a la eliminación de tramas aleatorias erróneas que pudiera ocasionar el algoritmo.

Factor de confirmación de 5		
Ave	**Acierto en tramas detectadas como aves**	**% De confirmaciones correctas**
Palomas	100,0	100,0
Gaviotas	97,4	98,3
Lechuzas	100,0	100,0
Carboneros	96,0	97,2
Gorriones	91,6	95,1
Vencejos	96,3	98,8
	96,8	98,2

Tabla 20 Resultados con un factor de confirmación =5

Pruebas con un factor de confirmación de 10: Se observan las 10 ultimas tramas para asegurar el ave identificada.

Ya podemos observar cómo todas esas tramas que el algoritmo identificaba erróneamente como otra ave, son comparadas con la mayoría de tramas siguientes para confirmar que ciertamente se trataban de errores salteados.

Con un factor de confirmación de 10 logramos una identificación positiva del 100%.

Factor de confirmación de 10		
Ave	**Acierto en tramas detectadas como aves**	**% De confirmaciones correctas**
Palomas	100,0	100,0
Gaviotas	97,4	100,0
Lechuzas	100,0	100,0
Carboneros	96,0	100,0
Gorriones	91,6	100,0
Vencejos	96,3	100,0
	96,8	100,0

Tabla 21 Resultados con un factor de confirmación =10

Con cada identificación positiva de un ave, se incrementan varios contadores. Cada ave tiene su propio contador, que se incrementara en uno el valor de este, cada vez que se detecte su ave.

Aparte, se dispone de otro contador que se incrementa siempre que se detecte cualquier ave.

Cuando se detecten 10 aves positivamente, se revisará cual ha sido el ave más detectada, y ese será el ave final identificada y confirmada.

De esta manera reducimos considerablemente la opción de detectar erróneamente un ave por una trama suelta y hacemos más robusto el algoritmo frente a errores ocasionales en sus identificaciones.

Ejemplo de una confirmación:

Paloma	Paloma	Paloma	Paloma	Paloma	Paloma	Gaviota	Paloma	Paloma	Paloma

En esta detección, se observaría como tenemos 9 identificaciones de paloma y una de gaviota, así que confirmaríamos que el ave detectada es una Paloma

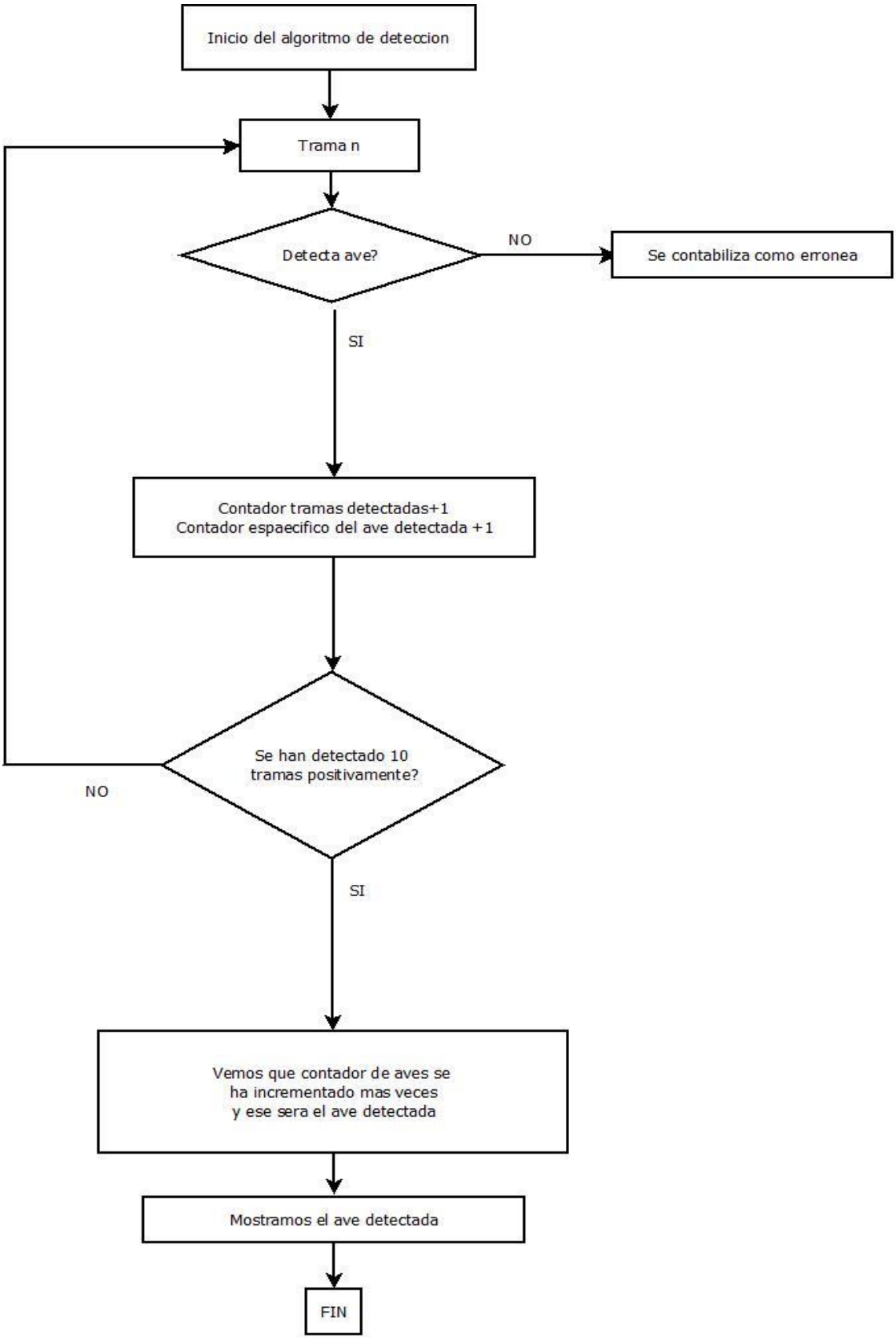


Ilustración 57 Diagrama del todo el algoritmo de identificación incluyendo la confirmación

En las siguientes tablas se presentan las pruebas que demuestran la eficiencia del algoritmo de identificación y confirmación elaborado. Se debe diferenciar entre las tramas identificadas como aves y las confirmaciones correctas (que son del conjunto de 10 aves identificadas). Aunque él % de acierto de las tramas identificadas sea algo menor del 100%, gracias al factor de confirmación se pueden descartar los fallos aislados.

Ave (audios 1-10)	% Acierto en tramas identificadas como aves	% confirmaciones correctas	% Tramas Indeterminadas*
Palomas	100,0	100,0	30,0
Gaviotas	97,4	100,0	62,5
Lechuzas	100,0	100,0	46,7
Gorriones	91,6	100,0	28,7
Carboneros	96,0	100,0	39,2
Vencejos	96,3	100,0	9,0
Media	96,8	100,0	24,8

Tabla 22 Aciertos, confirmaciones y tramas indeterminadas

*El % de tramas indeterminadas se refiere a cuando existiendo un ave, no se detecta nada.

Como se puede observar en las tablas, todas las aves tienen un % de acierto en la identificación > 90%, con un porcentaje de confirmaciones correctas el 100%. Esto demuestra que el algoritmo de identificación y confirmación es efectivo. Sin embargo, hay situaciones en las que el algoritmo no detecta el ave, representado por el % de tramas indeterminadas, como en el caso de la gaviota que es del 62.5%.

Datos con los que se ha calculado la tabla anterior:

	TRAMAS identificadas					
Ave (audios 1-10)	**Palomas**	**Gaviotas**	**Lechuza**	**Gorrión**	**Carbonero**	**Vencejo**
Palomas	**519**	0	0	0	0	0
Gaviotas	0	**590**	8	1	7	0
Lechuzas	0	0	**628**	0	0	0
Gorriones	0	0	1	**667**	13	14
Carboneros	0	0	0	31	**336**	0
Vencejos	0	0	0	11	29	**1053**

Tabla 23 Tramas identificadas

	CONFIRMACIONES					
Ave (audios 1-10)	**Palomas**	**Gaviotas**	**Lechuza**	**Gorrión**	**Carbonero**	**Vencejo**
Palomas	**51**	0	0	0	0	0
Gaviotas	0	**12**	0	0	1	0
Lechuzas	0	0	**62**	0	0	0
Gorriones	0	0	0	**69**	0	0
Carboneros	0	0	0	0	**36**	0
Vencejos	0	0	0	0	0	**109**

Tabla 24 Confirmaciones

4.2.4. Comprobación de la robustez del algoritmo de identificación

Una vez realizada todas las pruebas anteriores con un alto % favorable incluimos el ruido para ver la robustez de nuestro algoritmo.

Se introduce una SNR de desde 30 que sería el valor máximo (indica la mejor calidad de la señal) hasta 5 (ya que con una SNR1 indicaría que solo obtenemos ruido sin ninguna señal valida).

Los resultados de las pruebas realizas (SNR de 30,20,10,5) se encuentran a continuación:

4.2.4.1. SNR de 30 y 20

	SNR30			SNR20		
Ave	**% Acierto en tramas identificadas correctamente**	**% De confirmaciones correctas**	**%Tramas Indeterminadas**	**% Acierto en tramas identificadas correctamente**	**% De confirmaciones correctas**	**%Tramas Indeterminadas**
Palomas	100,0	100,0	30,0	100,0	100,0	28,7
Gaviotas	97,4	100,0	62,5	97,4	100,0	62,5
Lechuzas	100,0	100,0	46,7	100,0	100,0	41,3
Carboneros	96,0	100,0	39,2	72,3	87,0	41,0
Gorriones	91,6	100,0	28,7	97,7	100,0	36,6
Vencejos	96,3	100,0	9,0	95,8	98,4	48,0
Media	96,8	100,0	31,0	93,3	97,4	41,8

Tabla 25 Resultados de añadir ruido SNR 30 y 20

En una SNR de 30 se analiza en su versión más optima y no se obtiene variación con los resultados finales de nuestro algoritmo, se puede concluir que no hay ningún cambio ni afectación, sin embargo, con una SNR de 20 se ve un leve cambio en el % de tramas correctas y confirmaciones correctas, afectando a los carboneros y vencejos que son las aves con las frecuencias más altas a identificar en la biblioteca de audios. Se concluye que el ruido añadido a empezado afectar antes a las aves donde sus frecuencias fundamentas se encuentra más alta en el espectro.

Se aprecia también un leve aumento del porcentaje de tramas indeterminadas. El algoritmo empieza a descartar tramas de canto válidas para su análisis como si no formaran parte del canto de las aves.

4.2.4.2. SNR de 10 y 5

	SNR10			SNR5		
Ave	**% Acierto en tramas identificadas correctamente**	**% De confirmaciones correctas**	**%Tramas Indeterminadas**	**% Acierto en tramas identificadas correctamente**	**% De confirmaciones correctas**	**%Tramas Indeterminadas**
Palomas	100,0	100,0	37,7	100,0	100,0	46,9
Gaviotas	98,1	97,0	75,9	94,2	100,0	93,5
Lechuzas	99,5	100,0	48,6	99,5	100,0	53,0
Carboneros	1,6	2,1	63,6	0,4	1,8	81,4
Gorriones	88,2	93,3	91,0	92,8	91,7	92,8
Vencejos	96,3	100,0	9,0	6,7	100,0	97,6
Medias	48,9	51,5	44,0	24,3	50,5	74,5

Tabla 26 Resultados de añadir ruido SNR 10 y 5

Al analizar los audios con una SNR 10 y 5 se puede ver como el ruido empieza a afectar significativamente al valor de identificaciones correctas de manera general, teniendo un gran impacto en los carboneros, el cual casi se pierden el 100% de su identificación.

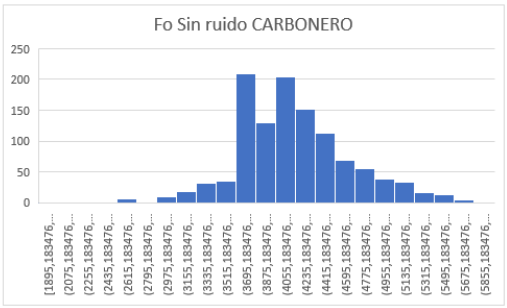
También se puede creer falsamente que según los resultados óptimos de ciertas aves como las gaviotas o los gorriones los cuales se observan con un % de confirmación próximo al 100%, que el ruido no afecta para la identificación de estas aves, no siendo correcta esta afirmación. Se puede observar cómo hay un aumento más que considerable en la cantidad de trama que son capaces de identificar como ave llegando en la SNR de 5 casi a un 75% de tramas ignoradas siendo útiles. La mayoría del tiempo ignora a las aves.

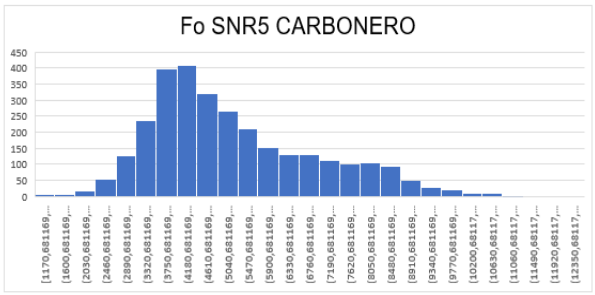
Se concluye que el ruido añadido con una SNR5 consigue identificar muy pocas tramas como aves, aunque las pocas que identifica lo hace en un alto % de acierto en todas las aves menos los carboneros.

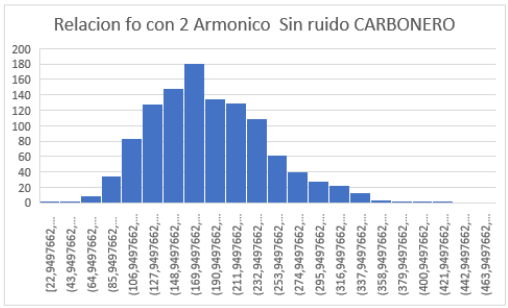
Esto se debe por lo siguiente. Al carbonero afecta considerablemente a la relación de la f_o con el segundo armónico el ruido y este es un valor decisivo en esta especie para poder distinguirlos de otras aves independientemente de la frecuencia:

En las siguientes imágenes se observa la distribución de la frecuencia y su relación con el segundo armónico del carbonero sin ruido y con la afectación de una SNR de 5.

Se ve claramente como afecta a las frecuencias, pero sigue teniendo el pico de la distribución en el mismo punto, pero a la relación de la f_o con el segundo armónico se descontrola por completo, haciendo imposible la identificación del ave.







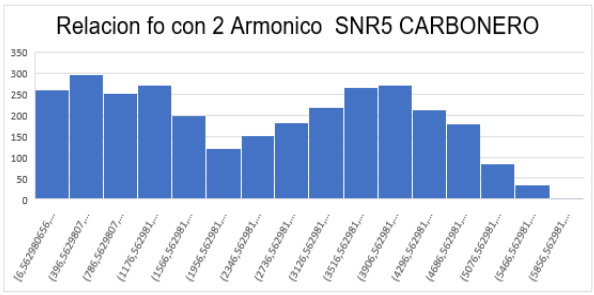


Ilustración 58 Distribución de las Fo y relación de esta con el segundo armónico del carbonero

Ilustración 59 Distribución de las Fo y relación de esta con el segundo armónico del carbonero con SNR5

4.2.4.3. Audios desconocidos para el algoritmo

Cuando ya se han concretado los parámetros idóneos para la identificación y comprobado también su robuste frente al ruido, se comprueba con batería de 100 audios ajenos al algoritmo su porcentaje de efectividad.

En la siguiente tabla se puede observar los % de identificaciones y confirmaciones correctas, así como el % de ave incorrecta confirmada siendo otra el ave analizada.

Se ve en la tabla como se obtiene un alto % de confirmaciones llegando al 90% de las muestras analizadas de forma correcta.

				% de confirmaciones correctas e incorrectas con un ave especifica analizada					
Ave	**%Acierto en tramas detectadas como aves**	**% De confirmaciones correctas**	**%Tramas Indeterminadas**	**Palomas**	**Gaviotas**	**Lechuza**	**Carbonero**	**Gorrión**	**Vencejo**
Palomas	100,0	100,0	28,0	100	0,0	0,0	0,0	0,0	0,0
Gaviotas	94,2	100,0	73,2	0,0	100	0,0	0,0	0,0	0,0
Lechuzas	64,3	79,3	50,3	0,0	0,0	79,3	4,3	17,4	0,0
Carboneros	96,9	100,0	46,8	0,0	0,0	0,0	100	0,0	0,0
Gorriones	75,6	78,8	19,0	0,0	0,0	0,0	21,2	78,8	0,0
Vencejos	81,2	89,7	15,6	0,0	0,0	0,0	6,9	3,4	89,7
Media	**83,3**	**90,1**	**28,9**						

Tabla 27 Resultados positivos con ficheros ajenos al algoritmo

Como era de esperar el filtro con los audios externos no es tan efectivo como han sido los resultados con respecto a los audios utilizados para la elaboración del algoritmo llegando este casi al 100%.

Donde más se ha notado el fallo ha sido en el gorrión, que es el ave con las frecuencias fundamental más extendida en el ancho de banda desde 3000Hz hasta 5000Hz.Haciendo también que disminuya el porcentaje de las lechuzas, ya que estas en su mayoría de errores es detectando el gorrión como lechuza en un 17%.

En cambio, el gorrión al ser detectado se confunde en su mayoría de errores (un 21%) con el Vencejo ya que comparte un amplio rango de frecuencias.

No obstante, se puede concluir que siendo muestras externas al algoritmo seguimos teniendo un alto % de identificaciones y confirmaciones positivas.

4.3. Configuración de Arduino DUE

En este apartado se explican la programación y componentes de Arduino, teniendo en cuenta la información extraída de Matlab.

Esta implementación consta de varias partes:

- Arduino DUE
- Captación del audio y muestreo
- Transformada de Fourier y cálculo de parámetros necesarios para su identificación (con las variables obtenidas en Matlab)
- Identificación y confirmación de las aves detectadas (con el filtro obtenido en Matlab).
- Conexión Wifi
- Envió de la información a nuestra Base de Datos

En los siguientes apartados iremos explicando detalladamente cada una de los puntos mencionados

4.3.1. Arduino DUE

Vamos a mostrar un esquema general de nuestro proceso, desde la captación del audio hasta el envío a la BBDD de nuestra ave confirmada.

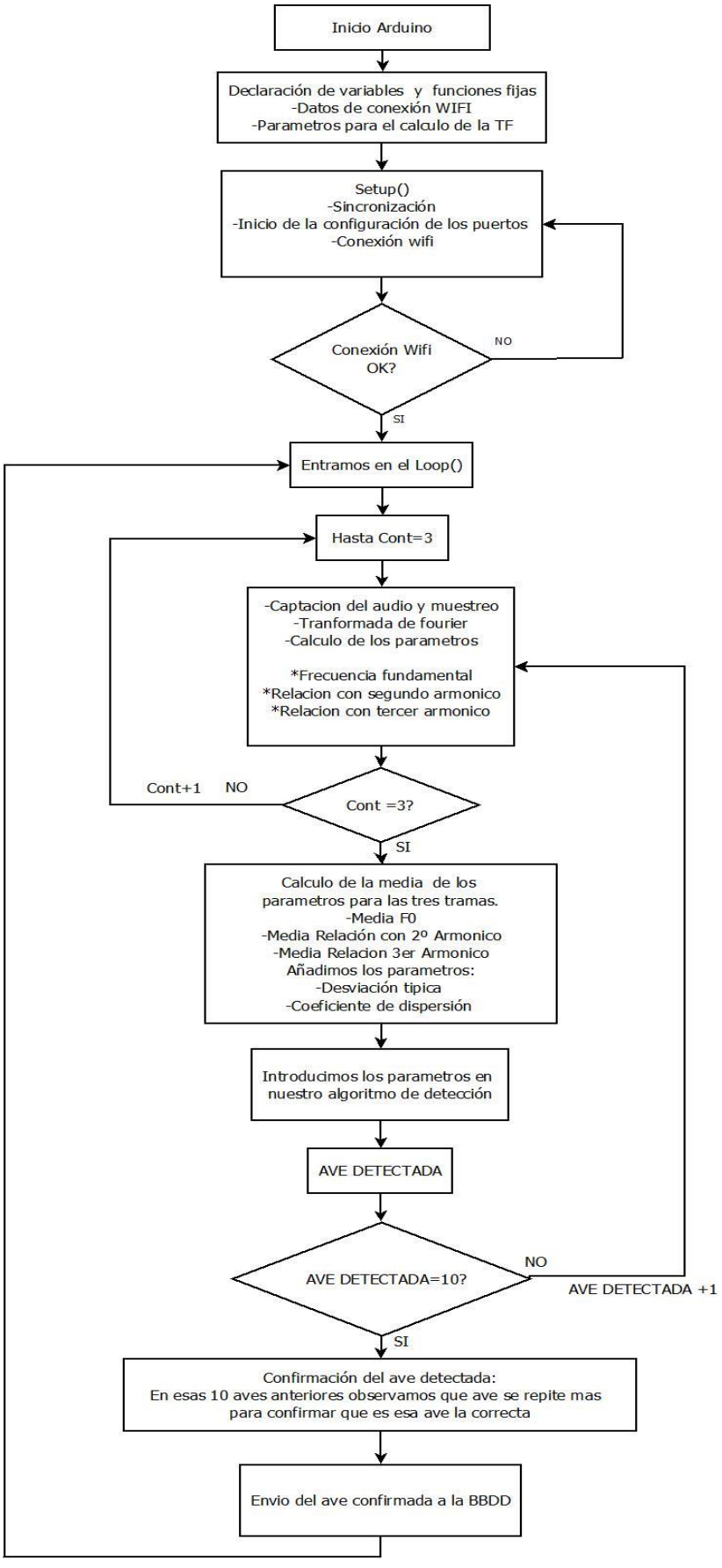


Ilustración 60 Diagrama de Confirmación

Como se observa en el diagrama, todo empieza al encender el Arduino.

Este empieza la configuración de los parámetros y datos necesarios para su funcionamiento,

- Librerías
- Parámetros De la TF, WIFI
- Declaración de los pines necesarios.

Después entraríamos dentro del Setup() donde establecemos la sincronización y la conexión wifi. Una vez la wifi establecida pasaríamos al Loop().

El Loop() es donde ocurre todo nuestro algoritmo.

Desde la captación del sonido, su procesado en tramas, la realización de la TF necesaria para la extracción de los parámetros.

Esto se realiza en tres ocasiones, una vez ya se ha realizado tres veces nos disponemos a calcular los valores medios de las tramas obtenidas.

En este punto ya tenemos todos los parámetros necesarios para la identificación del ave.

- Media de la frecuencia fundamental.
- Media de la relación con el 2º armónico.
- Media de la relación con el 3er armónico.
- Desviación típica de la $f_{o.}$
- Coeficiente de dispersión de la f_o.

Estos parámetros lo introducimos en el algoritmo de identificación, y nos indicara que ave a identificado.

Este valor lo guardaremos y realizaremos el proceso 10 veces seguidas, después observaremos estas 10 identificaciones y confirmaremos el ave que más se repite en esas 10 identificaciones. Este proceso es exactamente igual que el de Matlab.

Esa ave se enviará la Base de datos para que quede registrada.

4.3.2. Captación del audio y muestreo

Habiendo definido en el setup() los parámetros para la sincronización con el puerto serie y la resolución que podrá utilizar para capturar sonido, realizamos el muestreo.

La resolución estándar que ofrece la función analogRead es de 10 bits.

analogRead()=10 bits valores de 0 a 1023, para tener una mayor resolución, lo conseguiremos una resolución de 12 bits insertando la línea en el setup de (analogReadResolution(12);) lo que nos proporcionara valores de 0 a 4095. Siendo así más exacto el valor captado.

Después tomaremos muestras hasta rellenar un vector de 512 muestras y las guardaremos en el vector vReal su posterior uso. (Este valor de trama se modificó con respecto al Matlab ya que al realizar las pruebas de identificación con 2048 muestras era demasiado lento para concluir un ave y se decide ir bajando el tamaño de trama hasta obtener un tiempo aceptable de la identificación. Ver tabla de pruebas en el punto 4.3.5

```
#define SAMPLES 2048

for(int i=0; i<SAMPLES; i++)
 {
   useconds_sampling = micros();

   vReal[i] = analogRead(analogpin);
   vImag[i] = 0;

   while(micros() < (useconds_sampling + sampling_period_us)){
     //wait...
   }
```

4.3.3. Transformada de Fourier y cálculo de parámetros necesarios para la identificación

Para la realización de la transformada hemos usado la librería de Arduino (arduinoFFT.h) y lo siguientes parámetros.

La primera línea, FFT.Windowing, aplica una ventana de tipo Hamming a la señal para reducir las distorsiones de bordes. Esto se hace para mejorar la precisión del análisis espectral.

La segunda línea, FFT.Compute, realiza una transformada de Fourier rápida (FFT) en la señal para calcular la representación de frecuencia de la señal.

La tercera línea, FFT.ComplexToMagnitude, convierte la representación compleja de frecuencia de la señal en una representación de magnitud para facilitar su visualización y análisis.

Estas tres funciones se utilizan juntas para analizar el espectro de la señal y obtener información sobre la distribución de intensidad de frecuencia en la señal.

```
#include <arduinoFFT.h>
#define SAMPLES 512              //Must be a power of 2
#define SAMPLING_FREQUENCY 28000  //Hz
#define REFRESH_RATE 10          //Hz
#define ARDUINO_IDE_PLOTTER_SIZE 512
arduinoFFT FFT = arduinoFFT();

voido loop()

 FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
      FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
      FFT.ComplexToMagnitude(vReal, vImag, SAMPLES); //
```

En este punto tendremos un vector vReal con los valores de la transformada de Fourier tanto positivos como negativas, solo nos quedaremos con las frecuencias positivas ya que las negativas no tienen sentido. No están divido en frecuencias si no en bins. Al usar una frecuencia de muestreo de 28 KHz ya que nuestro ave con más alta frecuencia es el vencejo con 7000Hz y teniendo en cuenta que por el teorema de Nyquist[36] podremos obtener frecuencias hasta 14Khz, esto no será suficiente rango de frecuencias para poder obtener 20 armónicos de todas nuestras aves para realizar la identificación..

Sabiendo eso podremos calcular los parámetros necesarios para la identificación de igual manera que hacíamos en Matlab.

Cálculo de la F_o:

```
for(int i=(200/(SAMPLING_FREQUENCY/2/(SAMPLES/2))); i<(SAMPLES/2); i++)
{ //Calculo la posición donde se encuentra la amplitud máxima
                if (vReal[i]>aux1){
                aux1=vReal[i];
                poss=i;
                frec=(float((i+1)*(float(SAMPLING_FREQUENCY/2)/float(SAMPLES/2))))
                 }//fin del if
 }//fin
```

Cálculo del 2º y tercer armónico, si este se encuentra fuera de nuestra frecuencia máxima lo indicaremos como 1000

```
if ((poss*2)<SAMPLES/2)
          {
            ampl_2arm=vReal[poss_ar];
            R_arm2= float(vReal[poss]/vReal[poss_ar]);
            }
       Else
           {
             R_arm2=1000;
             ampl_2arm=0;
            }
```

Este proceso se repetirá como indicamos en el diagrama, 3 veces y guardaremos los valores en sus vectores correspondientes. Al tener los tres valores calcularemos la media de cada uno de ellos y añadiremos el cálculo de la desviación típica y coeficiente de dispersión.

```
float media_frec=0;
float media_R2arm=0;
float media_R3arm=0;
float Varianza_frec=0;
float Std_frec=0;
float Coe_frec=0;
float tramitas=3;
//---------------------------------------media frecuencias
for(int p=0;p<tramitas;p++){
aux_r=vfrec[p];
delay(1)
aux_ra=aux_ra+aux_r;
}
media_frec=aux_ra/tramitas;
//---------------------------------------Calculo de la media de 2º Armónico
```

```
media_R2arm=0;
aux_r=0;
aux_ra=0;
for(int p=0;p<tramitas;p++){
aux_r=vRarm[p];
delay(1);// Pondo el delay porque no le daba tiempo a sumar tres solo sumaba 2 datos y divida entre 3
aux_ra=aux_ra+aux_r;
}
media_R2arm=aux_ra/tramitas;
//---------------------------------------Cálculo de la media de 3er armónico
media_R3arm=0;
aux_r=0;
aux_ra=0;
for(int p=0;p<tramitas;p++){
aux_r=vRarm3[p];
delay(1);// Pondo el delay porque no le daba tiempo a sumar tres solo sumaba 2 datos y divida entre 3
aux_ra=aux_ra+aux_r;
}
media_R3arm=aux_ra/tramitas;
//---------------------------------------Calculo de la varianza
aux_r=0;
aux_ra=0;
float algo=0;
for(int p=0;p<tramitas;p++){//
algo=(vfrec[p]-media_frec);
algo=pow(algo,2);
delay(1);
aux_ra=aux_ra+algo;
}
Varianza_frec=aux_ra/tramitas;

//---------------------------------Calculo de la desviación típica y coeficiente de variacion
Std_frec=sqrt(Varianza_frec);
Coe_frec=(Std_frec/media_frec)*100;
```

Finalizando este proceso tendremos 5 variables necesarias para la identificación del Ave.

- media_frec: Media de la frecuencia fundamental
- media_R2arm: Media de la relación de la frecuencia fundamental con el segundo armónico
- media_R3arm: Media de la relación de la frecuencia fundamental con el tercer armónico
- Std_frec: Desviación típica de la Frecuencia fundamental
- Coe_frec: Coeficiente de dispersión de la Frecuencia fundamental

4.3.4. Identificación y confirmación de las aves detectadas.

Como ya tenemos los parámetros necesarios para la identificación nos disponemos a explicar el proceso de identificación y confirmación de las aves.

Primero creamos los posibles escenarios que nos encontraremos según las frecuencias. Estos escenarios nos basamos en los obtenidos en Matlab.

```
ave1= identificar( media_frec,media_R2arm,media_R3arm,Std_frec,Coe_frec);

if (media_frec >= 350 && media_frec <= 500) {
        caso = 1;
      } else if (media_frec > 1000 && media_frec <= 1250) {
        caso = 2;
      } else if (media_frec > 1250 && media_frec <= 1700) {
        caso = 3;
      }else if (media_frec > 1700 && media_frec <= 2900) {
        caso = 4;
      } else if (media_frec > 3000 && media_frec <= 5500) {
        caso = 5;
      }  else if (media_frec > 5500 && media_frec <= 7000) {
        caso = 6;
      }
      else {
        caso = 100;
      }
```

Una vez decidido en que escenario nos encontramos identificamos el ave. Que tras pasar por el algoritmo nos devolverá el ave detectada.

```
 switch (caso)
        {
          case 1:
          if     (media_R2arm > 15 && media_R2arm < 50 && media_R3arm > 3 && media_R3arm < 50
&& Std_frec >= 20 && Std_frec <= 130 && Coe_frec >=10 && Coe_frec <= 60)//MI VOZ 14
           {
                    ave=paloma;
           }
           break;
           case 2:
            if     (media_R2arm > 15 && media_R2arm < 50 && Std_frec >= 400 && Std_frec <= 700 &&
Coe_frec >= 30 && Coe_frec <= 60)
             {
                      ave =gaviota;
             }

              if     (media_R2arm > 2 && media_R2arm < 15 && Std_frec >= 300 && Std_frec <= 700 &&
Coe_frec >= 30 && Coe_frec <= 70)
             {
```

```
                    ave =gorrion;
            }

            if     (media_R2arm > 2 && media_R2arm < 15 && Std_frec >= 800 && Std_frec <= 1700 &&
Coe_frec >= 50 && Coe_frec <= 120)
            {
                    ave =gorrion;
            }
           break;
           case 3:
            if     (media_R2arm > 10 && media_R2arm < 70 && Std_frec >= 20 && Std_frec <= 200 &&
Coe_frec >= 2 && Coe_frec <= 16)
            {
                    ave =gaviota;
            }
            if    (media_R2arm > 10 && media_R2arm < 70 && Std_frec >= 900 && Std_frec <= 1200 &&
Coe_frec >= 50 && Coe_frec <= 70)
            {
                    ave =gaviota;
            }
            if     (media_R2arm > 2 && media_R2arm < 10 && Std_frec >= 500 && Std_frec <=
200 && Coe_frec >= 50 && Coe_frec <= 120)
            {
                    ave =gorrion;
            }
             if     (media_R2arm > 2 && media_R2arm < 50 && Std_frec >= 1200 && Std_frec <= 2000
&& Coe_frec >= 70 && Coe_frec <= 120)
            {
                    ave =gorrion;
            }
           break;
           case 4:

             if     (media_R2arm > 15 && media_R2arm < 400 && Std_frec >= 5 && Std_frec <= 300 &&
Coe_frec >= 0 && Coe_frec <= 15)
            {
                    ave =lechuza;
            }
             if     (media_R2arm > 50 && media_R2arm < 100 && Std_frec >= 400 && Std_frec <= 2500
&& Coe_frec >= 30 && Coe_frec <= 120)
            {
                   ave =gorrion;
            }
           break;
           case 5:
              if     (media_R2arm > 30 && media_R2arm < 2000 && Std_frec >= 0 && Std_frec <= 1000
&& Coe_frec >= 0 && Coe_frec <= 20)
            {
                    ave =carbonero;
            }
           if     (media_R2arm > 20 && media_R2arm < 150 && Std_frec >= 1200 && Std_frec <= 2500
&& Coe_frec >= 50 && Coe_frec <= 80)
            {
```

```
                    ave =gorrion;
                }
                break;
              case 6:
                if    (media_R2arm > 10 && media_R2arm < 800 && Std_frec >= 0 && Std_frec <= 900 &&
Coe_frec >= 0 && Coe_frec <= 20)
                {
                    ave =vencejo;
                }
                break;
            default:
             // Serial.println("No detectado");
              break;
          }
  return ave1;
}
```

Llegados a este punto nos dispondremos a realizar la confirmación de las aves detectadas de la misma manera que en Matlab.

Observando las 10 identificaciones últimas y viendo cual es el ave que más se repite.

```
if (ave1 == "PALOMA") {
      vdetec[0]++;
   } else if (ave1 == "GAVIOTA") {
      vdetec[1]++;
   } else if (ave1 == "GORRION") {
      vdetec[2]++;
   } else if (ave1 == "LECHUZA") {
      vdetec[3]++;
   }else if (ave1 == "CARBONERO") {
      vdetec[4]++;
   } else if (ave1 == "VENCEJO") {
      vdetec[5]++;
   }
   ave_detec++;

if (ave_detec == 10) {  // if para determinar cual es el maximo y seleccionar el ave

            int auxmax = 0;
            int posmax = 0;
            for (int i = 0; i < 6; i++) {  // para calcular el maximo
              if (vdetec[i] > auxmax) {
                auxmax = vdetec[i];
                posmax = i;
              }
            }  //fin for para ver el maximo

            if (posmax == 0) {
```

```
  ave_enviar = "PALOMA";
 }
 if (posmax == 1) {
  ave_enviar = "GAVIOTA";
 }
 if (posmax == 2) {
  ave_enviar = "GORRION";
 }
 if (posmax == 3) {
  ave_enviar = "LECHUZA";
 }
 if (posmax == 4) {
  ave_enviar = "CARBONERO";
 }
 if (posmax == 5) {
  ave_enviar = "VENCEJO";
 }
 Serial.print("EL AVE a enviara la  BASE DE DATOS : ");
 Serial.println(ave_enviar);
Serial.println("-----------------------------------------------");
```

4.3.5. Resultados: pruebas Arduino DUE Falta completar

Todas las pruebas de Arduino han sido realizadas en un entorno de interior. Con altavoces del ordenador portátil MSI GF63 Thin 11UD-271XES. Con el sensor ESP8266 ESP-01 a un metro de distancia.

4.3.5.1. Audios utilizados para la realización del algoritmo

Las primeras pruebas que realizamos en el algoritmo de Arduino están basadas en los variables obtenidas de Matlab.

	Paloma	Gaviota	Lechuza	Gorrión	Carbonero	Vencejo
Tiempo medio en detectar trama como ave	0:00:04s	0:00:05s	0:00:03s	0:00:04s	0:00:03s	0:00:04s
Tiempo medio en Mandar dato a la BBDD	0:00:40s	0:00:51s	0:00:34s	0:00:43 s	0:00:33s	0:00:42s
Tiempo en alcanzar la 50 identificación es 5 envíos a la BBDD	0:08:25s	0:07:58s	0:06:16s	0:07:34 s	0:06:45s	0:07:56s
Tramas identificadas correctamente	97,6%	63,4%	73,5%	91,7%	52,5%	89,7%
Tramas identificadas erróneamente	2,4%	36,6%	26,5%	8,3%	47,5%	10,3%
Envíos correctos a la BBDD (Confirmaciones)	99%	81%	97%	99%	59%	100%

Tabla 28 Resultados de pruebas con los valores del Matlab de partida.

Se puede observar cómo los parámetros son positivos con alto % de datos correctos enviados a la BBDD, pero se consideran hacer más pruebas para reducir el tiempo de estos envíos.

Nos disponemos a modificar el tamaño de trama a la mitad (estando ahora en 2048) para comprobar si esta reducción ayuda aumentar los resultados de identificación, pero sobre todo el tiempo de análisis para reducir así de esta manera el tiempo en de envió de los datos a la BBDD.

Pruebas con el tamaño de trama de 1024 muestras. Tras realizar las pruebas con un tamaño de trama de 1024 podemos observar cómo se reduce el tiempo en que al Arduino analiza las tramas y toma una decisión de enviar un dato válido a la BBDD. Pasamos de una media de 7 minutos con 2048 tramas a una media de 2 minutos con 1024.

Se reduce casi a la mitad de tiempo, y además se observa una mejora en el porcentaje de envío de algunas aves como es la gaviota o el carbonero el cual tenía un porcentaje muy bajo de 59% con 2048 muestras. Ahora se obtiene un % superior al 95% en todas las aves.

Aun así, vamos a observar los resultados con una trama de 512 muestras a ver si mejorarán la velocidad y los resultados aún más.

	Paloma	Gaviota	Lechuza	Gorrión	Carbonero	Vencejo
Tiempo medio en detectar trama como ave	0:00:03	0:00:01	0:00:01	0:00:02	0:00:01	0:00:01
Tiempo medio en Mandar dato a la BBDD	0:00:42	0:00:07	0:00:06	0:00:17	0:00:11	0:00:05
Tiempo en alcanzar las 50 identificaciones 5 envíos a la BBDD	0:04:05	0:00:44	0:00:35	0:01:31	0:01:00	0:00:28
Tramas identificadas correctamente	93,6%	89,6%	97,2	86,0%	85,0%	97,4%
Tramas identificadas erróneamente	6,4%	10,4%	2,8%	14,0%	15,0%	2,6%
Envíos correctos a la BBDD (Confirmaciones)	96%	96%	100%	96%	96%	100%

Tabla 29 Resultados de pruebas con trama de 1024 muestras.

Pruebas con el tamaño de trama de 512 muestras. Tras realizar las pruebas con un tamaño de trama de 512 podemos observar cómo se reduce el tiempo en que tarda el Arduino en analiza las tramas y tomar una decisión de enviar un dato valido a la BBDD en ciertas aves.

Obteniendo también casi un 100% de tramas correctas enviadas en todas las aves.

Por este motivo el parámetro final de nuestro algoritmo de identificación será con tramas de 512 muestras.

	Paloma	Gaviota	Lechuza	Gorrión	Carbonero	Vencejo
Tiempo medio en detectar trama como ave	0:00:02	0:00:01	0:00:01	0:00:02	0:00:01	0:00:01
Tiempo medio en Mandar dato a la BBDD	0:00:14	0:00:07	0:00:05	0:00:17	0:00:04	0:00:05
Tiempo en alcanzar las 50 identificaciones 5 envíos a la BBDD	0:01:21	0:00:47	0:00:31	0:01:31	0:00:25	0:00:28
Tramas identificadas correctamente	96,6	88,8	96,2	86,2	98,4	97,4%
Tramas identificadas erróneamente	1,7	5,6	3,8	13,8	1,6	2,6%
Envíos correctos a la BBDD (Confirmaciones)	100%	100%	100%	100%	96%	100%

Tabla 30 Resultados de pruebas con trama de 512 muestras.

Con estos resultados óptimos nos disponemos comprobar si en audios ajenos al algoritmo seguimos manteniendo este porcentaje de identificación alto.

4.3.5.2. Audios Ajenos al algoritmo

Una vez se consigue una identificación en un tiempo asequible y en un porcentaje alto superior al 95 % en todas las aves con las que creamos el algoritmo, se pone a prueba el algoritmo enfrentándolo a una biblioteca de audios de 60 muestras ajenas al algoritmo.

	Paloma	Gaviota	Lechuza	Gorrión	Carbonero	Vencejo
Tiempo medio en detectar trama como ave	0:00:02	0:00:01	0:00:01	0:00:02	0:00:02	0:00:01
Tiempo medio en Mandar dato a la BBDD	0:00:18	0:00:12	0:00:10	0:00:14	0:00:13	0:00:07
Tiempo en alcanzar las 50 identificaciones 5 envíos a la BBDD	0:01:42	0:01:09	0:00:57	0:01:20	0:01:16	0:00:43
Tramas identificadas correctamente	95,2	75,8	96,8	84,0	77,8	93,8
Tramas identificadas erróneamente	4,8	24,2	3,2	16,0	22,2	6,2
Envíos correctos a la BBDD (Confirmaciones)	100%	68%	100%	76%	96%	96%

Tabla 31Resultados de pruebas con audios ajenos al algoritmo.

Tras pasar las pruebas por las muestras ajenas al algoritmo podemos concluir que seguimos teniendo valores positivos de identificación cerca del 80% de media incluso llegando al 100% en la paloma y la lechuza.

4.3.6. Conexión wifi y envío de la información a la BBDD

Para poder realizar el envío de nuestra ave confirmada a la Base de Datos, tendremos que instalar un módulo wifi, ya que el Arduino no dispone de él.

Es lo primero que se configurara al encender el Arduino. No necesitaremos ninguna librería especial para su configuración. Ya que se realizará mediante comandos AT.[37,38]

Los comandos AT son un conjunto de órdenes utilizadas para controlar y configurar módulos Wi-Fi basados en el chip ESP8266, como el ESP-01. Estos comandos permiten realizar tareas como conectar el módulo a una red Wi-Fi, obtener la dirección IP asignada, configurar el modo de transmisión de datos, entre otras funciones sin necesidad de

librerías. Los comandos AT se envían desde Arduino a través de un puerto serie y el módulo Wi-Fi responde con una respuesta definida para cada comando. [37,38]

Lo primero es establecer las variables fijas de la wifi:

```
#define ESP8266 Serial3
String SSID = "Durden";
String PASSWORD = "JESUS1986";

boolean FAIL_8266 = false;
int LED = 8;
#define BUFFER_SIZE 1024
char buffer[BUFFER_SIZE];
```

Estas variables las utilizaremos para establecer y comprobar la conexión wifi.

```
//-----------------------------------------------------
// Config Wifi ----------------------------------------
//-----------------------------------------------------
 pinMode(LED, OUTPUT);
  // Led Azul indica el inicio de la configuración

  delay(100);
  digitalWrite(LED, HIGH);

  do{
    Serial.begin(115200);
    ESP8266.begin(115200);

    //Iniciamos la conexion, esperamos a que este OK
    while(!Serial);
    Serial.println("--- Start ---");
    ESP8266.println("AT+RST");
    delay(1000);
    if(ESP8266.find("OK"))
    {
      Serial.println("El modulo esta listo");

      ESP8266.println("AT+GMR");
      delay(1000);
      clearESP8266SerialBuffer();

      ESP8266.println("AT+CWMODE=1");

      delay(2000);

      //Quitamos el punto de acceso anterior
      Serial.println("Quit AP");
      ESP8266.println("AT+CWQAP");
```

```
    delay(1000);

    clearESP8266SerialBuffer();
    if(cwJoinAP())
    {
      Serial.println("CWJAP OK");
      FAIL_8266 = false;

      delay(3000);
      clearESP8266SerialBuffer();
      //Mostramos la IP
      sendESP8266Cmdln("AT+CIFSR", 1000);
      //Seleccionamos conexiones multiples
      sendESP8266Cmdln("AT+CIPMUX=1", 1000);
      Serial.println("Setup finish");
      digitalWrite(LED, LOW);//Fin de la configuración
    }else{
      Serial.println("CWJAP Fail");
      delay(500);
      FAIL_8266 = true;
    }
  }else{
    Serial.println("Modulo no responde."); // si la luz Azul parpadea el módulo no responde
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
    digitalWrite(LED, HIGH);
    delay(500);
    FAIL_8266 = true;
  }
}
while(FAIL_8266);
//Tiempo de espera
ESP8266.setTimeout(1000);
```

Una vez finalizada la instalación y la conexión con nuestro wifi, ya podríamos mandar la información que deseemos a nuestro BBDD.

Al no enviar claves de usuario o contendió confidencial, haremos el envío mediante una petición GET.

Para formar y enviar la petición GET lo haremos de la siguiente manera.

```
String TARGET_ID="0";
String TARGET_TYPE="TCP";
String TARGET_ADDR="s939286598.mialojamiento.es";
String TARGET_PORT="80";
String cmd="AT+CIPSTART=" + TARGET_ID;
cmd += ",\"" + TARGET_TYPE + "\",\"" + TARGET_ADDR + "\"";
cmd += ","+ TARGET_PORT;
Serial.println(cmd);
ESP8266.println(cmd);
delay(1000);

clearESP8266SerialBuffer();

String HTTP_RQS = "GET /dato.php?valor=" + ave_enviar + " HTTP/1.1\r\n";
HTTP_RQS += "Host: " + TARGET_ADDR;
HTTP_RQS += ":" + TARGET_PORT + "\r\n\r\n";

String cmdSEND_length = "AT+CIPSEND=";
cmdSEND_length += TARGET_ID + "," + HTTP_RQS.length() +"\r\n";

ESP8266.print(cmdSEND_length);
Serial.println(cmdSEND_ length);
```

Una vez realizado el envío esperamos una respuesta de servidor con SEND OK y si es satisfactoria parpadeara un led verde como señal de envió satisfactorio.

```
//Esperamos a que aparezca SEND OK
while(!OK_FOUND){
  if(ESP8266.readBytesUntil('\n', buffer, BUFFER_SIZE)>0){
    Serial.println("...");
    Serial.println(buffer);

    if(strncmp(buffer, "SEND OK", 7)==0){
      OK_FOUND = true;
      Serial.println("SEND OK found");
            for (int i = 0; i < 10; i++) {
                  digitalWrite(LEDBBDD, HIGH);
                   delay(100);
                  digitalWrite(LEDBBDD, LOW);
                  delay(100);
             }
    }else{
      Serial.println("Not SEND OK...");
```

4.4. Creación de una base de datos y visualización en una aplicación web.

Se crea una BBDD en el servidor phpMyAdminya para almacenar las aves detectadas por el Arduino DUE.

Para recibir el dato e insertarlo en la BBDD se utiliza el archivo dato.php donde se realizará la conexión a la BBDD y realizaremos la instrucción inserta en nuestra BBDD.

Solo realizara la inserción cuando por GET reciba uno de los 6 aves implementados.

```
<?php
// Conectamos a la base de datos
        $host = "db5010670333.hosting-data.io";
        $user = "dbu2146807";
        $password = "Jesus_1986";
        $dbname = "dbs9029820";
                $conn = mysqli_connect($host, $user, $password, $dbname);
                // Comprueba la conexión
                if (!$conn) {
                        die("Connection failed: " . mysqli_connect_error());
                }
                // Realizamos la consulta a la base de datos para insertar un dato en ella
        $fechaActual = date('Y-m-d');
        $horaActual = date('H:i:s');
        $valor=$_GET["valor"];
        echo $valor;

                if( $valor=="PALOMA" || $valor=="GAVIOTA" || $valor=="LECHUZA" || $valor=="CARBONERO" || $valor=="VENCEJO"|| $valor=="GORRION")
                  {
                // Instrucción para insertar los valores en la B.D
                $sql = "INSERT INTO identificador (Id, Hora, Fecha, Insecto) VALUES (NULL, '$horaActual', '$fechaActual', '$valor')";
                 if (mysqli_query($conn, $sql)) {
                  echo "Inserción realizada correctamente";
                } else {
                  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
                }

        }
                 else {
                  echo "Error: Valor desconocido" ;
                }
?>
```

Nuestra base de datos constará con una estructura dividida en 4 parámetros de los cuales solo uno será enviado por nuestro Arduino. El tipo de ave.

La fecha y Hora será implementado por el archivo dato.php al enviar el insert a la BBDD y el identificador lo genera la propia BBDD al inserta una línea nueva.

Ave	El **Ave** reconocida.
Hora	La **hora** de la identificación
Fecha	**Fecha** de la identificación
Identificativo	Un **Identificativo** automático que será único para cada detección. Se incrementará su valor en uno cada vez que haga una nueva detección.

Ilustración 61 Estructura BBDD

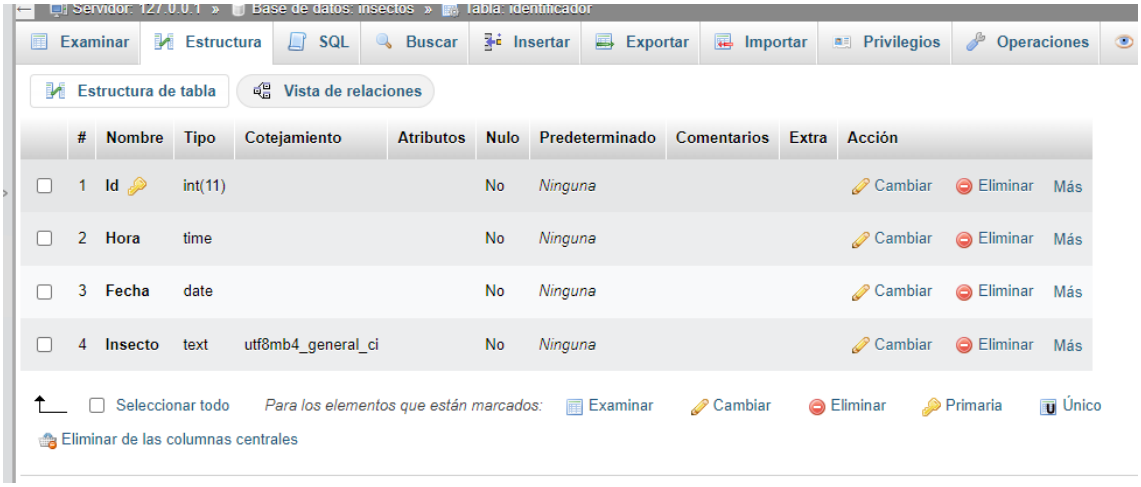


Ilustración 62 Estructura BBDD

En este punto del proyecto vamos a necesitar una web en la que mostraremos las identificaciones en tiempo real que va detectando nuestro nodo.

La web se realizará a base de HTML y PHP.

HTML para la parte visual y PHP para las consultas a la Base de datos.

Consta de un índex donde nos encontraremos las Comunidades autónomas donde tendremos nuestros sensores.

Al entrar en la web veremos una tabla comunidades donde están nuestros sensores.

Actualmente solo se encuentra operativo el apartado de Andalucía. Las otras comunidades son para futuras ampliaciones.

WEB: http://s939286598.mialojamiento.es/

Ilustración 63 Web principal

Al entrar en Andalucía nos encontramos con la lista de aves detectadas.

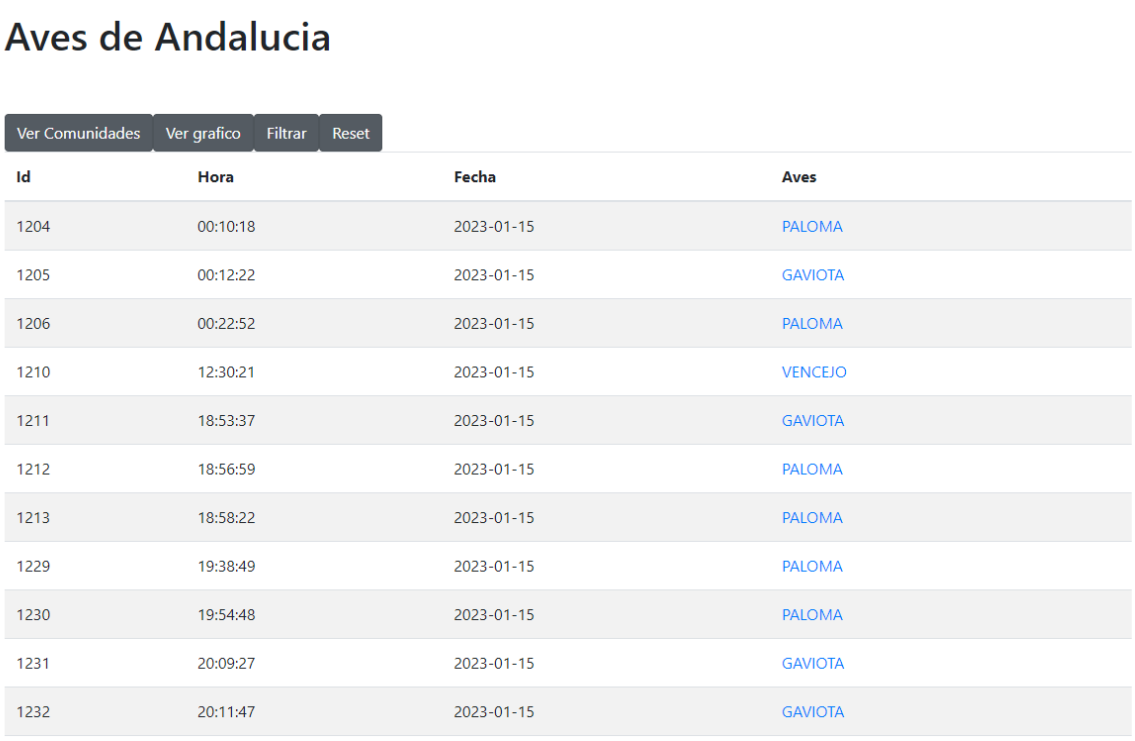


Ilustración 64 Web listado de aves.

Podemos observar un botón de Ver Gráfico en el cual podremos de un vistazo rápido ver todas las aves identificas en nuestra comunidad autónoma seleccionada.

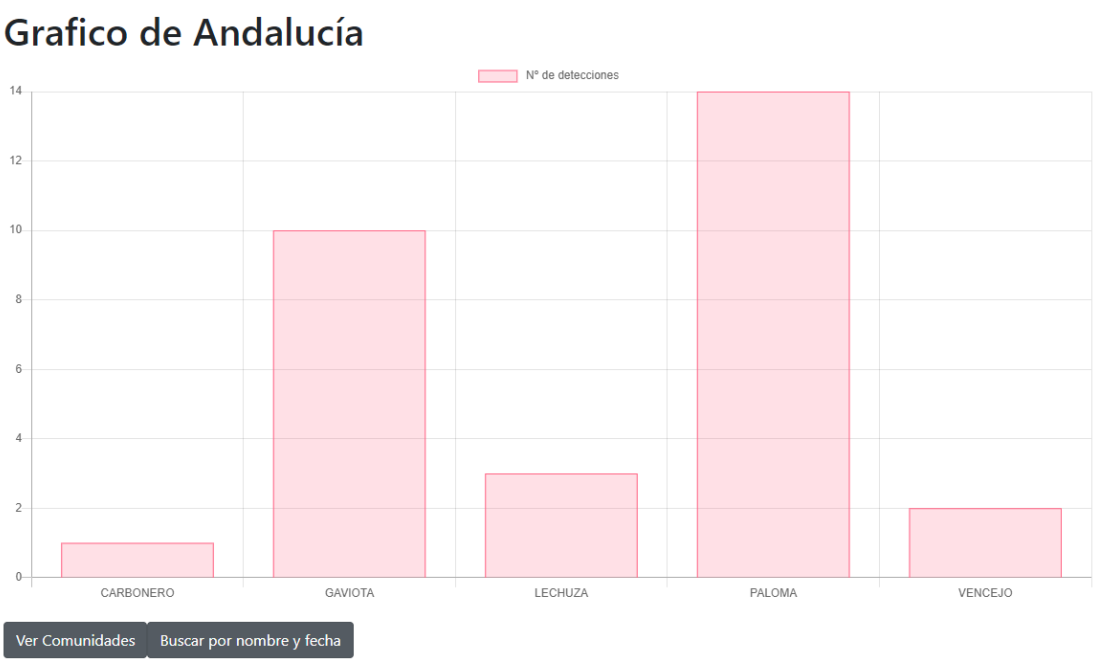


Ilustración 65 Web - Gráfico de una comunidad autónoma.

Como vemos las aves son enlaces que si clicamos en ellas filtraremos por ese tipo de ave concreta. De igual manera si hacemos click en el botón Filtrar nos encontraremos un formulario que nos permitirá filtrar el ave deseada por una fecha concreta, o todas las aves en una fecha elegida o simplemente filtrar un ave en todo el rango de fechas. Se nos informara de cuantas aves se encuentran en nuestro filtro deseado.

Id	Hora	Fecha	Aves
1204	00:10:18	2023-01-15	PALOMA
1205	00:12:22	2023-01-15	GAVIOTA
1206	00:22:52	2023-01-15	PALOMA
1210	12:30:21	2023-01-15	VENCEJO
1211	18:53:37	2023-01-15	GAVIOTA
1212	18:56:59	2023-01-15	PALOMA
1213	18:58:22	2023-01-15	PALOMA
1229	19:38:49	2023-01-15	PALOMA
1230	19:54:48	2023-01-15	PALOMA
1231	20:09:27	2023-01-15	GAVIOTA
1232	20:11:47	2023-01-15	GAVIOTA

Ilustración 66 Web - Formulario

Filtramos por ejemplo las PALOMAS detectadas entre el día 4 y 5 de febrero.

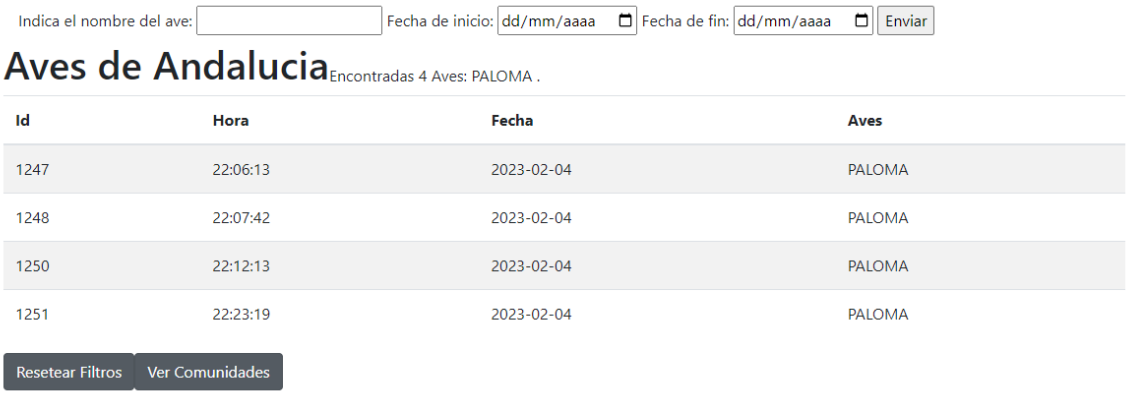


Ilustración 67 Web - Formulario filtrado.

A continuación, mostraremos la implementación del formulario y las consultas relacionadas para mostrar y contar las aves:

```
<form action="Andalucia_nombre_1.php" method="post">
 <label for="nombre">Indica el nombre del ave:</label>
  <input type="text" id="nombre" name="nombre">
 <label for="fecha_inicio">Fecha de inicio:</label>
  <input type="date" id="fecha_inicio" name="fecha_inicio">
  <label for="fecha_fin">Fecha de fin:</label>
  <input type="date" id="fecha_fin" name="fecha_fin">
 <input type="submit" value="Enviar">
</form>


 $sql = "SELECT * FROM identificador WHERE (insecto = '$nombre' OR '$nombre' = '') AND (Fecha BETWEEN '$fecha_inicio' AND '$fecha_fin' OR '$fecha_inicio' = '' OR '$fecha_fin' = '')";



      $sql1 ="SELECT COUNT(*) FROM  identificador WHERE (insecto = '$nombre' OR '$nombre' = '') AND (Fecha BETWEEN '$fecha_inicio' AND '$fecha_fin' OR '$fecha_inicio' = '' OR '$fecha_fin' = '')";
```

5. CONCLUSIONES Y LÍNEAS FUTURAS

Conclusiones:

En el presente proyecto se ha propuesto una solución de bajo coste para la identificación y monitorización de aves en tiempo real, gracias a la detección del sonido que emiten, mediante redes de sensores inalámbricas.

Ave (audios 1-10)	% Acierto en tramas identificadas como aves	% confirmaciones correctas	% Tramas Indeterminadas*
Palomas	100,0	100,0	30,0
Gaviotas	97,4	100,0	62,5
Lechuzas	100,0	100,0	46,7
Gorriones	91,6	100,0	28,7
Carboneros	96,0	100,0	39,2
Vencejos	96,3	100,0	9,0
Media	96,8	100,0	24,8

Tabla 32 Altos % de identificación demostrada

Se ha demostrado la ejecución del dispositivo con altos porcentajes de acierto, del 99% de media.

Se ha probado añadiendo ruido mostrando unos resultados aceptables hasta cierta cantidad de ruido.

Este sistema es de bajo coste, fácil montaje, fácil mantenimiento, consta de materiales accesibles, no requiere de mano de obra de alta especialización para replicarlo, se podría utilizar en áreas de extenso tamaño y también en zonas pequeñas.

También presenta algunas limitaciones, como el ser un sistema cualitativo no cuantitativo, podrían presentarse interferencias por fenómenos meteorológicos y no sería adecuado para zonas marítimas.

Este sistema se podría implementar en:

- Zonas medioambientales protegidas, campanarios, playas, zonas de cultivos pequeños y extensos, zonas urbanas, campos de golf, jardines y bosques.
- Regiones de bajos ingresos.
- La protección de especies en peligro de extinción.
- Sistemas de alerta temprana y de seguimiento de cambios poblacionales
- El intercambio de datos entre distintas regiones
- La evaluación del éxito de programas de controles biológicos

Líneas Futuras:

Además, esta solución se podría seguir mejorando en el futuro para:

- Adaptarlo a otras especies de animales o insectos.
- Ligarlo a un sistema para espantar pájaros y otros animales
- Unirlo a un sistema de alimentación por reconocimiento de sonido para aves.
- Adaptarlo para zonas marítimas.
- Implementar inteligencia artificial y que se mejore automáticamente, entre otros posibles usos.

6. REFERENCIAS

1. admin. Las aves en el medioambiente [Internet]. RH Corporative Internacional. 2020 [citado 12 de febrero de 2023]. Disponible en: https://cirhe.com/las-aves-en-el-medioambiente/

2. SEO/BirdLife P. Servicios ecosistémicos que nos ofrecen las aves y la naturaleza [Internet]. SEO/BirdLife. 2020 [citado 12 de febrero de 2023]. Disponible en: https://seo.org/2020/05/20/servicios-ecosistemicos-que-nos-ofrecen-las-aves-y-la-naturaleza/

3. Monitoring [Internet]. Connecting People, Birds and Land for a Healthy World. [citado 12 de febrero de 2023]. Disponible en: https://www.birdconservancy.org/what-we-do/science/monitoring/

4. Interview — Vital role of bird monitors — European Environment Agency [Internet]. [citado 12 de febrero de 2023]. Disponible en: https://www.eea.europa.eu/signals/signals-2021/articles/interview-vital-role-of-bird-monitors

5. Arduino vs Raspberry Pi [Internet]. OpenWebinars.net. 2019 [citado 12 de febrero de 2023]. Disponible en: https://openwebinars.net/blog/arduino-vs-raspberry-pi/

6. Arduino vs Raspberry Pi: The Differences [Internet]. All3DP. 2020 [citado 12 de febrero de 2023]. Disponible en: https://all3dp.com/2/arduino-vs-raspberry-pi/

7. 330ohms. Diferencias entre un microprocesador y un microcontrolador [Internet]. 330ohms. 2020 [citado 12 de febrero de 2023]. Disponible en: https://blog.330ohms.com/2020/11/19/diferencias-microcontrolador-microprocesador/

8. jecrespom. Microcontrolador vs Microprocesador [Internet]. Aprendiendo Arduino. 2017 [citado 12 de febrero de 2023]. Disponible en: https://aprendiendoarduino.wordpress.com/2017/08/11/microcontrolador-vs-microprocesador-3/

9. Dear R. Single Board Computers – What are they and how are they used? [Internet]. 2022 [citado 12 de febrero de 2023]. Disponible en: https://www.dsl-ltd.co.uk/what-are-single-board-computers-and-how-are-they-used/

10. techslang. What is a Single-Board Computer? — Definition by Techslang [Internet]. Techslang — Tech Explained in Simple Terms. 2022 [citado 12 de febrero de 2023]. Disponible en: https://www.techslang.com/definition/what-is-a-single-board-computer/

11. ¿Que es Raspberry Pi? - Raspberry Pi [Internet]. 2019 [citado 13 de febrero de 2023]. Disponible en: https://raspberrypi.cl/que-es-raspberry/

12. Raspberry Pi 3B+ - Raspberry Pi [Internet]. 2019 [citado 13 de febrero de 2023]. Disponible en: https://raspberrypi.cl/raspberry-pi-3b-2/

13. Inicio - Raspberry Pi [Internet]. 2019 [citado 13 de febrero de 2023]. Disponible en: https://raspberrypi.cl/

14. ESP32. En: Wikipedia, la enciclopedia libre [Internet]. 2023 [citado 13 de febrero de 2023]. Disponible en: https://es.wikipedia.org/w/index.php?title=ESP32&oldid=148961355

15. ESP32 Wroom WIFI + Bluetooth BricoGeek | BricoGeek.com [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://tienda.bricogeek.com/arduino-compatibles/1274-esp32-wroom-wifi-bluetooth.html

16. The Internet of Things with ESP32 [Internet]. [citado 13 de febrero de 2023]. Disponible en: http://esp32.net/

17. ¿Qué es Arduino? | Arduino.cl - Compra tu Arduino en Línea [Internet]. 2014 [citado 13 de febrero de 2023]. Disponible en: https://arduino.cl/que-es-arduino/

18. What is Arduino? [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://www.arduino.cc/en/Guide/Introduction

19. Arduino Uno Rev3 [Internet]. Arduino Official Store. [citado 13 de febrero de 2023]. Disponible en: https://store.arduino.cc/products/arduino-uno-rev3

20. Arduino DUE [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://www.fantasystudios.es/arduino/pages/Arduinos/due.html

21. Due | Arduino Documentation [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://docs.arduino.cc/hardware/due

22. Arduino Reference - Arduino Reference [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://www.arduino.cc/reference/en/

23. jecrespom. Lenguaje de programación de Arduino, estructura de un programa [Internet]. Aprendiendo Arduino. 2015 [citado 13 de febrero de 2023]. Disponible en: https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-de-arduino-estructura-de-un-programa/

24. Qué es MySQL: Características y ventajas [Internet]. OpenWebinars.net. 2019 [citado 13 de febrero de 2023]. Disponible en: https://openwebinars.net/blog/que-es-mysql/

25. Acibeiro M. ¿Cuál es la diferencia entre HTTP y HTTPS? [Internet]. Blog. 2021 [citado 13 de febrero de 2023]. Disponible en: https://es.godaddy.com/blog/diferencia-entre-http-y-https/

26. HTTP - Concepto, para qué sirve y cómo funciona [Internet]. Concepto. [citado 13 de febrero de 2023]. Disponible en: https://concepto.de/http/

27. Vocalización de las aves. En: Wikipedia, la enciclopedia libre [Internet]. 2021 [citado 13 de febrero de 2023]. Disponible en: https://es.wikipedia.org/w/index.php?title=Vocalizaci%C3%B3n_de_las_aves&oldid=139204720

28. ¿Cómo cantan los pájaros? [Internet]. Celebrate Urban Birds. [citado 13 de febrero de 2023]. Disponible en: https://celebrateurbanbirds.org/es/faq/como-cantan-los-pajaros/

29. Armónicos [Internet]. Mgmdenia’s Blog. 2011 [citado 13 de febrero de 2023]. Disponible en: https://mgmdenia.wordpress.com/2011/11/27/armonicos/

30. MAX4466 Low-Cost, Micropower, SC70/SOT23-8, Microphone Preamplifiers with Complete Shutdown | Analog Devices [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://www.analog.com/en/products/max4466.html#product-overview

31. ESP8266 Wi-Fi Modules I Espressif [Internet]. [citado 13 de febrero de 2023]. Disponible en: https://www.espressif.com/en/products/modules/esp8266

32. Guía para configurar un ESP-01, el módulo WiFi basado en ESP8266 [Internet]. 2017 [citado 13 de febrero de 2023]. Disponible en: https://programarfacil.com/podcast/como-configurar-esp01-wifi-esp8266/

33. Módulo ESP-01 ESP8266 WiFi-Serial [Internet]. Naylamp Mechatronics - Perú. [citado 13 de febrero de 2023]. Disponible en: https://naylampmechatronics.com/espressif-esp/48-modulo-esp-01-esp8266-wifi-serial.html

34. Vellinga WP. Xeno-canto - Wildlife sounds from around the world [Internet]. Xeno-canto Foundation for Nature Sounds; 2023 [citado 13 de febrero de 2023]. Disponible en: https://www.gbif.org/dataset/b1047888-ae52-4179-9dd5-5448ea342a24

35. Frecuencia de muestreo. En: Wikipedia, la enciclopedia libre [Internet]. 2022 [citado 13 de febrero de 2023]. Disponible en: https://es.wikipedia.org/w/index.php?title=Frecuencia_de_muestreo&oldid=142505681

36. Teorema de muestreo de Nyquist-Shannon. En: Wikipedia, la enciclopedia libre [Internet]. 2022 [citado 13 de febrero de 2023]. Disponible en: https://es.wikipedia.org/w/index.php?title=Teorema_de_muestreo_de_Nyquist-Shannon&oldid=147155497

37. jecrespom. Comandos AT [Internet]. Aprendiendo Arduino. 2017 [citado 13 de febrero de 2023]. Disponible en: https://aprendiendoarduino.wordpress.com/tag/comandos-at/

38. Configuración del módulo bluetooth HC-06 usando comandos AT [Internet]. Naylamp Mechatronics - Perú. [citado 13 de febrero de 2023]. Disponible en: https://naylampmechatronics.com/blog/15_configuracion-del-modulo-bluetooth-hc-06-usando-comandos-at.html