Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

Monitorización de aves mediante redes de sensores inalámbricas orientada al control de plagas de aves.

Alumno: Ignacio Mula López

Tutor: Manuel Ángel Gadeo Martos
Dpto.: Dpto. de Ing. Telecomunicación.

Tutor: Damián Martínez Muñoz
Dpto.: Dpto. de Ing. Telecomunicación.

Septiembre, 2018

- Página en blanco -

Índice.

1. Introducción.

1.1. Antecedentes.

Hoy día, gracias al desarrollo de la tecnología, hemos conseguido mejorar la calidad de vida en la mayoría de sectores en los que esta se ha implementado, como por ejemplo: la automatización de los sistemas, mejora en la calidad y cantidad de producción de bienes, la efectividad y el consumo energético de los equipos, entre otros.

Otra de las aplicaciones de la tecnología actual es aquella que tiene relación con los seres vivos ya sea el control de especies a través de sensores, la monitorización de especies amenazadas y en peligro de extinción, la gestión de ganado, entre otros. Pero uno de los sectores donde todavía no hay mucha actividad es en el de control de plagas.

Pues bien, este último sector es en el que se va a centrar este Trabajo Fin de Grado, de ahora en adelante TFG.

1.2. Investigación Previa.

El desarrollo de este TFG comienza con una investigación sobre cuáles son las especies identificadas como plagas potenciales, dentro del territorio nacional, para el análisis y recogida de las características principales de dichas especies, con las que construir un sistema de identificación.

En lo relacionado con el mundo de las aves, tras la consulta de varias páginas de expertos, se menciona como la identificación de especies puede llevarse a cabo a través del plumaje de los individuos, sus hábitos alimenticios, reproductivos y sociales, pero el factor en el que más se basan estos expertos para la identificación y diferenciación de especies es en el canto de las aves, por lo que para este TFG se utilizarán técnicas de análisis de sonido, siendo más concretos, el estudio del canto de las aves identificadas como plagas potenciales.

Habiendo identificado ya las especies sobre las que se va a desarrollar el sistema de identificación y teniendo en mente los métodos que se van a utilizar para el análisis de las muestras, el último paso de esta etapa previa es la recogida de muestras vocales de las especies para elaborar una pequeña biblioteca sobre la que trabajar.

1.3. Esquema.

A lo largo de este documento se revisan la base teórica de este proyecto, los componentes hardware que se han utilizado, así como el software utilizado. Se representarán también las pruebas realizadas, así como los resultados obtenidos de las mismas.

Estas partes quedan representadas en los siguientes puntos:

- El punto 2 explica el estado del arte, en el que se estudian los tres aspectos fundamentales del trabajo: Caracterización de las aves, Componentes Hardware y Software.

- El punto 3 muestra los componentes de hardware que se han utilizado y cómo se han realizado las conexiones en el mismo.

- El punto 4 se describe las solución que se ha implementado separándola en los distintos escenario sobre los que se ha trabajado, empezando por el desarrollo del sistema de caracterización en Matlab, su traslado a la placa Arduino y la comunicación de este con el servicio web.

- El punto 5 presenta las diferentes pruebas que se realizaron en Matlab para la optimización de los parámetros seleccionados para el sistema de identificación de las aves, así como los resultados obtenidos de cada uno.

- El punto 6 explorará los distintos métodos de ahuyentamiento implementables en Arduino, así como una comparación teórica entre ellos para elegir el que implementará en el sistema final de gestión.

- El punto 7 resume el trabajo realizado en este proyecto, así como un análisis de las oportunidades que este sistema ofrece.

1.4. Objetivos

En este Trabajo Fin de Grado se persiguen por tanto los siguientes objetivos:

- La elaboración de una biblioteca de ficheros de audio, de los que obtener la información necesaria para elaborar un sistema de identificación.

- La propuesta de una metodología con la que caracterizar espectralmente el sonido de los ficheros de la biblioteca.

- La propuesta de un método de identificación a través de los parámetros obtenidos durante el análisis espectral de los ficheros de la biblioteca.

- Llevar a cabo una optimización del sistema de identificación para que pueda ser implementado en una red de sensores inalámbricos.

- La implementación del sistema de detección en los dispositivos que formarán la red de dispositivos inalámbricos.

- El estudio de las diferentes técnicas utilizadas hoy día en relación con la gestión de plaga de aves.

- La implementación de las mejores técnicas para ahuyentar a las aves en los nodos que forman la de sensores inalámbricos.

2. Estado del Arte.

En este apartado se encuentra una breve descripción teórica de los que son los puntos esenciales de este proyecto, que como se han mencionado en la introducción son: el hardware utilizado, en el cual se verá lo que son los microcontroladores Single-Board Computer, su historia, así como una comparación entre los distintos dispositivos existentes, el software implementado, que cubrirá los lenguajes de programación utilizados para la realización del trabajo y por último una caracterización del colectivo que se va a gestionar, donde se hará un breve análisis de las plagas que están presenten en territorio nacional y para terminar se revisarán las técnicas de detección de aves mediante su sonido.

2.1. Hardware.

2.1.1. Microcontroladores.

Un Single-Board Computer (SBC), o lo que sería su traducción al español, un ordenador monoplaca, es un ordenador que se caracteriza por estar construido en una sola placa y que cuenta con microprocesadores (uno o varios), memoria, entradas y salidas y otras características de un ordenador convencional [1]. Estos SBC, se desarrollaron como demostración del desarrollo de los sistemas informáticos, para fines educativos, o para ser utilizados como sistemas integrados.

A diferencia de los ordenadores de sobremesa, los SBC no cuentan con los puertos de expansión típicos de los equipos de mayor tamaño, pero si tienen (en su mayoría), pines de conexión con los que aumentar las tareas que pueden realizar. Gracias a esta peculiaridad de los SBC, acompañado de su bajo coste y tamaño reducido, ha permitido que en los últimos años, muchos usuarios los hayan utilizados para los proyectos más dispares, desde el control de electrodomésticos a través de aplicaciones móvil, hasta la instalación de servidores portátiles para implementar protocolos de comunicación en desarrollo.

Desde el punto de vista histórico, el primer SBC fue desarrollado por la empresa *Radio-Electronics* en Mayo de 1976 y le pusieron el nombre de *Dyna-micro* [2]. Estaba basado en el *Intel-C8080A y* también uso el primero EPROM de *Intel*, el *C1702A* [3], lo cual permitía el almacenamiento de información cuando se apagaba el dispositivo. El *Dyna-micro* fue renombrado más tarde como *MMD-1*(Mini-Micro Designer 1), por la empresa *E&L Instruments* y fue famoso como ejemplo de microordenador en la aclamada serie *BugBook 8080.*

Como se ha mencionado antes, estos SBC cuentan con los componentes necesarios para cumplir con las mismas funciones que un ordenador cualquiera (aunque con sus limitaciones en cuanto a potencia de procesamiento, potencia gráfica, durabilidad, entre otros). Los componentes principales que permiten esto son:

- Procesador o Unidad Central de Procesamiento (CPU en inglés), es el hardware dentro del equipo que interpreta las instrucciones de un programa informático mediante operaciones aritméticas [4].

- Memoria, es el dispositivo que guarda los datos necesarios para el correcto funcionamiento del equipo hardware, así como de los programas instalados en el mismo. Esta memoria puede ser de tres tipos, Flash, RAM y ROM, que se utilizan de igual manera en todos los dispositivos [5].

- Puertos de Entrada y Salida, son aquellos que permiten la conexión de elementos externos que amplían las funcionalidades de los equipos y que además conectan dichos equipos con los usuarios, o con otros dispositivos [6].

Encontramos varios tipos de SBC, que se diferencian en los que permiten la instalación de un sistema operativo (SO) y los que no pueden tenerlo. La diferencia principal entre ambas SBC es que la que permite la instalación del SO tiene integrado un microprocesador y la otro no lo tiene [7]. La elección de un tipo de SBC dependerá principalmente de la aplicación que se quiera realizar con ella y es lo que analizaremos a continuación.

2.1.2. Raspberry Pi.

La *Raspberry Pi,* es una serie de SBCs desarrollados en Reino Unido, por la Raspberry Pi Foundation, para promover la enseñanza de informática básica en las escuelas y en los países en desarrollo. Según datos oficiales de la fundación para Febrero de 2015 ya se habían vendido 5 millones de Raspberries en todo el mundo, convirtiéndolo en el ordenador británico más vendido [8].

Dada su creciente popularidad, la Raspberry Pi, ha sufrido una evolución que ha diferenciado los productos que se desarrollaban para ajustarse a las necesidades de todos los clientes, con lo que la fundación ahora ofrece cuatro modelos distintos al mercado [9].

- Raspberry Pi Model 1 B.

  Fue la primera generación de la fundación, lanzada al mercado en febrero de 2012, aunque dos años más tarde este primer modelo fue sustituido por el B+. Estas placas tienen el tamaño de una tarjeta de crédito y son consideradas la línea estándar de la compañía. La versión disponible hoy día cuenta con un procesador BCM2835, 512MB de memoria RAM, 2 puertos USB, 1 puerto Ethernet y un puerto HDMI [10].

- Raspberry Pi Model 2 B.

  Lanzado al mercado en Febrero de 2015, el modelo 2 fue la evolución natural del modelo 1, al cual le añadieron más memoria RAM, así como un cambio en el procesador con el que ofrecer más potencia a los usuarios. Se cambió el procesador BCM2835 por el BCM2836, lo cual supuso una mejora de 200MHz al chip, pasando de 700MHz a 900MHz. Además se incluyeron otros dos puertos USB, para ampliar el número de periféricos que se podían conectar [11].

- Raspberry Pi Zero.

  Esta placa se lanzó en Noviembre de 2015 y fue la versión reducida de las placas que se habían desarrollado hasta el momento. Se redujo el tamaño de la placa y también se eliminaron puertos de entrada//salida, así como de pines generales de entrada/salida (GPIO en inglés), pero a cambio solo costaban 5$ americanos. Tal fue su éxito que en 2017 se convirtió en la nueva línea principal de ventas de la fundación [12].

- Raspberry Pi Model 3 B. [Imagen 2.1.2.1]

  De nuevo, la evolución del modelo 2 fue el modelo 3. Esta placa se lanzó al mercado en Febrero de 2016 e incluía mejoras significativas en cuanto a su predecesora. Pero no fue hasta el día de Pi (14 de Marzo), que apareció el modelo 3 B+, que tenía un procesador de 1.4GHz, un puerto gigabit Ethernet de 300 Mbit/s, o una tarjeta wifi compatible con las frecuencias de 2'4 y 5 GHz [13].

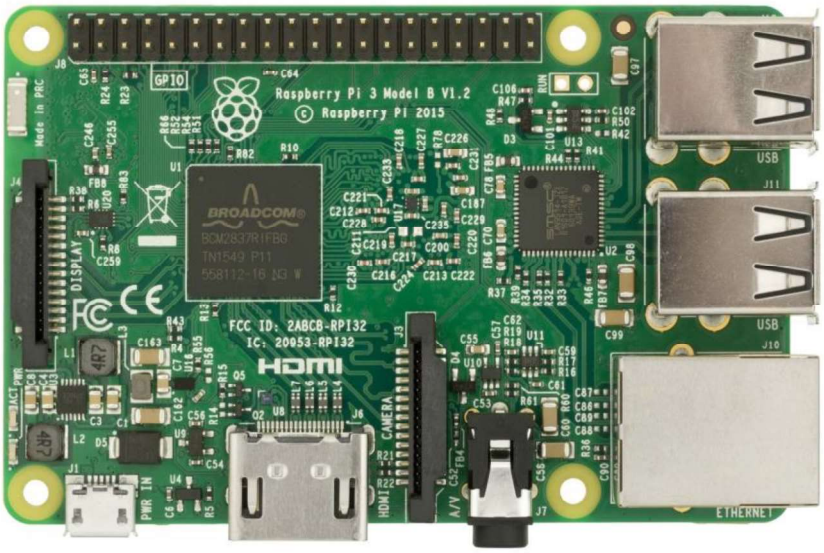


Imagen 2.1.2.1

Placa Raspberry Pi Model 3 B+

Fuente: https://www.raspberrypi.org/

Desde entonces hasta el día de hoy, todas las tarjetas han sido mejoradas y cuentan con el mismo número de puertos entrada/salida, el mismo número de pines GPIO y mismas conexiones (Ethernet, HDMI, Micro SD), pero manteniendo las diferencias de procesadores y velocidades en los puertos, para ofrecer soluciones a todos los usuarios interesados en una especificaciones más o menos potentes.

Las Raspberry Pi, entran dentro del anteriormente citado grupo de SBC que cuentan con microprocesador, por lo que se le puede instalar un SO, que suele ser normalmente Raspbian, una distribución de Ubuntu adaptada especialmente para la Raspberry Pi. Este tipo de SBC son más adecuados para tareas que necesiten de una interfaz gráfica más compleja, el tratamiento de grandes cantidades de datos, la elaboración de protocolos de comunicación, el tratado de archivos multimedia ya sea imagen, audio, video.

Desde el punto de vista de este proyecto, en el que se va a necesitar la recolección de datos en pequeños grupos y el tratamiento inmediato de los mismos, no es necesario el uso de una interfaz gráfica, ni la utilización de puertos de comunicación físicos, por lo que, aunque tenga muchas funcionalidades, no se va a necesitar de un sistema Raspberry Pi.

2.1.3. Arduino.

Arduino es una compañía de open source y de open hardware, así como un proyecto y comunidad internacional, que diseña, fabrica y distribuye placas de desarrollo de hardware. El proyecto conocido hoy día como Arduino, comenzó en el año 2005 como una iniciativa enfocada a estudiantes en el Instituto de Ivrea en Italia. Dicha iniciativa surge debido al elevado coste de los microcontroladores que se estaban utilizando en ese momento (alrededor de unos 100$ americanos). [14]

Hoy día Arduino goza de una alta demanda debido a su gran compatibilidad con un amplio repertorio de módulos de hardware llamados Shields y a que cuenta con un lenguaje de programación propio basado en C [15], que permite la integración de estas shields y de otros objetos. [16]

A diferencia de la Raspberry Pi, Arduino tiene un gran abanico de productos con los que abarcan diferentes sectores, por eso podemos encontrar placas de Arduino, como son Arduino Uno, Arduino Due, Arduino Mega, Arduino Leonardo y varios más; pero también podemos encontrar dispositivos peculiares como Arduino Lilypad, el cual se ha desarrollado para prendas inteligentes. Pero en este apartado sólo analizaremos el Arduino Due ya que es la placa que ha sido la elegida para el desarrollo de este TFG. [Imagen 2.1.3.1]

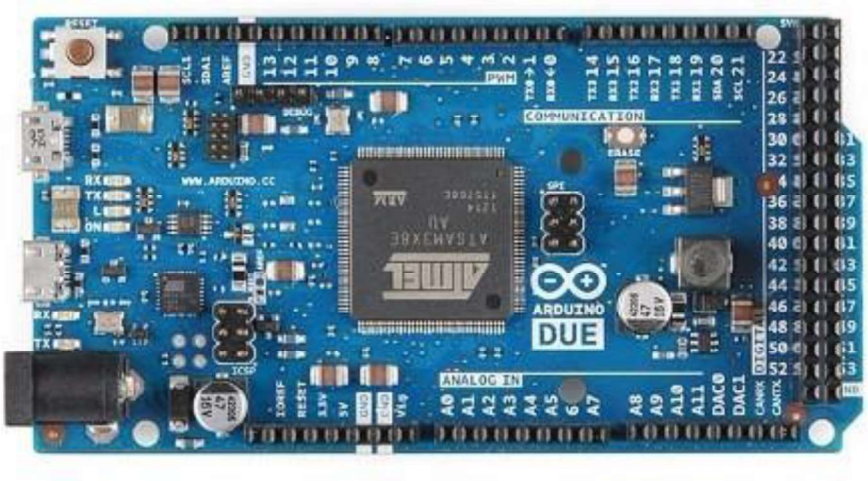


Imagen 2.1.3.1
Placa Arduino Due
Fuente: https://www.arduino.cc/

A continuación se muestra unas tablas con las especificaciones técnicas de la placa, con el objetivo de ilustrar las posibilidades de la misma. [Tabla 2.1.3.1]

Microcontrolador	AT91SAM3X8E
Voltaje Operativo	3.3 V
Voltaje de Entrada (recomendado)	7 - 12 V
Voltaje de Entrada (límites)	6 - 16 V
Pines Entrada/Salida Digitales	54
Pines de Entradas Analógicas	12
Pines de Salida Analógicas	2 (DAC)
Salida continua en los pines entrada/salida	130 mA
Corriente continua del pin 3’3V	800 mA
Corriente continua del pin 5V	800 mA
Memoria Flash	512 KB disponibles para aplicaciones
SRAM	96 KB (dos bancos; 64KB y 32KB)
Velocidad del Reloj	84 MHz
Longitud	101.52 milímetros
Anchura	53.5 milímetros
Peso	36 gramos

Tabla 2.1.3.1

Especificaciones Placa Arduino Due

Fuente: https://www.arduino.cc/

Además, la placa Arduino Due, es la primera en contar con un núcleo ARM de 32-bit, lo que la hace perfecta para proyectos de gran escala [17].

2.2. Lenguajes de Programación.

2.2.1. Arduino.

Todas las placas Arduino mencionadas en el punto anterior se programan con el lenguaje Arduino, el cual nace a partir de C y C++. Según la página oficial: "el lenguaje de programación Arduino es un mero conjunto de funciones en C/C++ que pueden ser llamadas por el controlador. Los sketches (ficheros de código) sufren unas pequeñas variaciones a la hora de llamada de funciones y variables y luego pasan por un compilador de C/C++ como por ejemplo (avr g++)" [18].

El lenguaje de programación de Arduino se puede dividir en tres grandes partes que son: Estructura, Valores y Funciones.

- Estructura. Encontramos dos funciones principales en los sketches de Arduino que son:
    - Setup(); que es la función que se ejecuta al iniciar el sketch y sirva para la inicialización de los pines, la carga de las librerías, etc. El setup() sólo se ejecuta una vez cada vez que se enciende la placa, o cada vez que se produce un reset [19].
    - Loop(); Esta función, como su nombre indica, se ejecuta en bucle consecutivamente, lo que permite al programa ir cambiando y responder a las posibles entradas que se produzcan. Se utiliza para controlar de forma activa la placa Arduino [20].
- Valores. Como es de esperar en esta parte encontramos los tipos de variables que acepta el lenguaje y como es de suponer, al estar basado en C/C++ son las mismas: enteros, caracteres, condicionales, decimales, etc. En Arduino también existen las llamadas constantes, que no pueden cambiar de valor cuando se ejecuta el código.
- Funciones. Aquí encontramos las funciones que se ejecutarán en la parte del loop() y bien pueden ser funciones creadas por el usuario, o algunas propias de Arduino. [21]

2.2.2. PHP y HTML.

PHP, acrónimo recursivo en inglés de PHP Hypertext Preprocessor (preprocesador de hipertexto), es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en un documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera el HTML resultante [22].

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web [23].

Se han elegido estos lenguajes para la elaboración del servicio web debido a su facilidad de implementación y por qué son los más utilizados hoy días, lo cual permite que una gran cantidad de usuarios pueda comprenderlo en el caso futuro de una continuación de este proyecto.

2.2.3. SQL.

SQL (por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada) es un lenguaje específico del dominio utilizado en programación; y diseñado para administrar sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas [24].

Ya que en este proyecto se necesita de una base de datos, se ha utilizado el lenguaje SQL para realizar las consultas, así como las adiciones de las especies identificadas en cada momento.

2.3. Protocolo HTTP.

El Protocolo de transferencia de hipertexto (en inglés: Hypertext Transfer Protocol o HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1 [34].

Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. El cliente realiza una petición enviando un mensaje, con cierto formato al servidor. El servidor le envía un mensaje de respuesta [35].

Para este proyecto se utilizará el protocolo HTTP para el envío de información, en este caso el ave que sea identificada, desde la placa Arduino hasta un servidor web que contiene una base de datos, como se ha mencionado anteriormente.

2.4. Caracterización de las aves.

2.4.1. Estudio de las plagas de aves.

Como se puede apreciar en el título de este TFG y como se ha mencionado anteriormente, el objetivo del proyecto es el de la gestión de las especies de aves consideradas como plagas, así que en primer lugar comenzaremos revisando que es una plaga. Una plaga, desde el punto de vista animal, es un colectivo de individuos que, por sus acciones, repercuten de manera negativa en el estado y en la salud del entorno en el que se encuentran [25].

Tras una investigación sobre cuáles son las principales especies que amenazan el territorio Español encontramos que las aves que se han adaptado a vivir en áreas urbanas son principalmente las palomas, estorninos, gorriones, gaviotas, cuervos, cigüeñas, vencejos y recientemente las cotorras. La falta de depredadores, la abundancia de alimento

y la similitud del medio urbano con el suyo natural, son la principal razón de su proliferación. [26]. Pueden ser un riesgo para la salud pública por varias razones:

- Crean suciedad con sus excrementos, además de contener sustancias ácidas que resultan muy agresivas para los materiales de construcción como la piedra y provocan corrosión en las superficies de metal.

- Las gaviotas pueden traer problemas especiales causando ruidos molestos, produciendo gran cantidad de basuras y excrementos, daños a edificaciones, e incluso ataques a personas y animales de compañía provocando esto efectos adversos contra el turismo.

- Las aves están llenas de organismos patógenos perjudiciales para el ser humano y para los animales domésticos. Tienen efectos sobre la salud ya que llevan asociados más de 60 enfermedades ya que pueden actuar como reserva de microorganismos patógenos que afecten a las personas y animales domésticos.

- Son portadores de ectoparásitos como garrapatas, pulgas, piojos, etc. [27]

2.4.2. Métodos de Identificación.

Una vez que ya conocemos cuales son las principales especies causantes de los problemas más graves, pasamos a la parte de cómo realizar la identificación del ave. A través de varios manuales encontramos las siguientes formas que tienen los expertos de identificarlas:

- Uno de los factores que se tienen en cuenta para la diferenciación de especies son las adaptaciones que estas han sufrido, para acomodarse al entorno en el que se desarrollaron. Desde las fuertes patas de las rapaces, hasta los pequeñas y ágiles patas de los semilleros, o las patas con membrana de las aves que viven en entornos acuáticos, estos son aspectos que pueden ayudar a distinguir las especies. Lamentablemente este factor no es lo suficientemente significativo como para

poder utilizarlo en un sistema con Arduino, además de que necesitamos una cámara con una resolución lo suficientemente amplia como para poder distinguir estas partes de los animales [28].

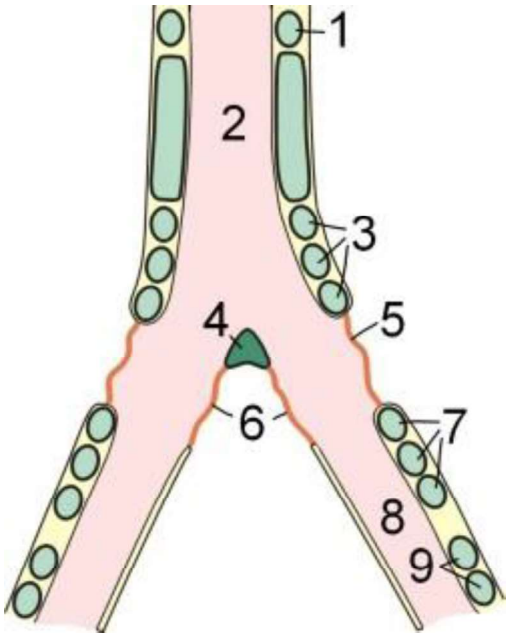
- Conductas, migraciones y anidado. De nuevo, la manera en la que viven las especies puede aportar mucha información de las mismas, pero estos factores, al igual que el anterior, no son sencillos ni eficaces a la hora de implementarlo en un sistema portátil como es Arduino, por lo que estos aspectos también quedan descartados [28].

- Llegamos por tanto al factor que es único para cada especie y con el que es más sencillo distinguir a individuos de familias diferentes, que es el canto. Según un manual de identificación de aves, en el apartado acerca de las vocalizaciones dice: "Cada especie posee su propio canto, llegando en algunos casos a emitir más de una docena de sonidos diferentes. Por lo general existen dos tipos de vocalizaciones: el canto y la llamada. El canto está relacionado principalmente con la defensa de un territorio o con la atracción de la hembra, en especial en los Paseriformes (aves que se perchan). El canto está constituido por sonidos complejos, melodiosos y de gran alcance. "[28] Este aspecto si podemos utilizarlo para su implementación en Arduino ya que con un micrófono se puede recoger muestras de sonidos que tras ser analizadas pueden servir para diferenciar unas especies de otras.

A continuación se hará un breve estudio del órgano vocal de las aves, a fin de entender la complejidad de sonidos que pueden producir, para luego implementar un sistema que permita ajustarse a distintas especies que tengan localidades similares.

La siringe es el órgano vocal de las aves [Imagen 2.3.1]. Se ubica en la base de la tráquea y produce sonidos complejos sin las cuerdas vocales que tienen los mamíferos. Los sonidos se producen por vibraciones en las paredes de la siringe o por la vibración del aire que pasa por la siringe. Puesto que la siringe se ubica donde la tráquea se bifurca para formar los bronquios, más profundo que la laringe mamífero, muchos pájaros cantores pueden producir dos sonidos o más simultáneamente [29].

A través de la apertura o cierre de distintos anillos ubicados en la siringe, las aves son capaces de estirar o contraer las membranas de este órgano, lo que le permite el tono de la palabra que producen, llegando así a reproducir docenas de palabras por individuo.



1. Último anillo cartilaginoso traqueal libre
2. Tímpano
3. Primer grupo de anillos siringales
4. Pessulus
5. Membrana timpaniforme lateral
6. Membrana timpaniforme medial
7. Segundo grupo de anillos siringales
8. Bronquio principal
9. Cartílago bronquial

Imagen 2.3.1

Siringe

Fuente: https://es.wikipedia.org/wiki/Siringe

2.4.3. Métodos de gestión.

Tras la consulta de varios portales sobre el control de plagas, así como de manuales propuestos por asociaciones protectoras de las aves, se pueden implementar varios métodos para el control de estas especies que se diferencian en dos grupos:

- Métodos de gestión previa a la presencia de las aves, entre las que encontramos la instalación de barreras físicas en los lugares que son foco de instalación de estos animales. Se consideran barreras físicas todas aquellas que impiden el acceso, posado y la anidación de las aves, en las zonas habitables.

- Métodos de gestión ante la presencia de las aves. En este apartado encontramos dos soluciones posibles, que son la captura de las aves, con el objetivo de reducir el número de ejemplares existentes en los bandos de las poblaciones de las aves; y métodos de ahuyentamiento, los cuales están diseñados para desplazar las bandadas generando espacios libres de aves. [39]

Dada la implementación final que va a tener este proyecto, los métodos de gestión previa son imposibles de realizar a través de un Arduino, ocurriendo lo mismo con el método de captura, ya que se precisarían de jaulas donde capturas a los individuos, así como de personal que se encagara de gestionar las aves capturadas. Eso nos deja con los métodos de ahuyentamiento.

En el punto 6 se realizará un estudio y comparación entre los distintos métodos de ahuyentamiento que se puedan implementar en el Arduino, y se elegirá el que se vea que es más factible a la hora de obtener resultados.

3. Materiales Utilizados.

A continuación se presentarán los componentes hardware utilizados para la elaboración del sistema de detección de este TFG.

3.1. Arduino Due

Como ya se ha visto en el punto 2.1.3 se ha hecho un estudio previo de los productos Arduino, así como un estudio de las especificaciones técnicas del Arduino Due, que ha sido el seleccionado para este proyecto. Así para no repetir contenidos, se puede encontrar toda la información relacionada con la placa en ese punto.

3.2. Wifi Shield para Arduino Due

La Wifi Shield [Imagen 3.2.1] para Arduino permite conectar la placa Arduino a la que se acople a la red de manera inalámbrica. Para realizar la conexión a la red solo hay que incluir en el sketch la librería de Wifi de Arduino y luego crear la función necesaria para suministrar los credenciales necesarios para que la conexión con la red sea exitosa. [30]

Las características de este módulo son: [Tabla 3.2.1]

Voltaje Operativo	5V
Estándares Soportados	802.11b/g
Encriptación	WEP y WPA2 Personal
Conexión con Arduino	A través de puertos SPI

Tabla 3.2.1

Especificaciones de la Wifi Shield para Arduino

Fuente: https://store.arduino.cc/arduino-wifi-shield

Imagen 3.2.1

Wifi Shield para Arduino

Fuente: https://store.arduino.cc/arduino-wifi-shield

3.3. Micrófono MAX4466

El micrófono que se va a utilizar para este TFG es el Electret MAX4466 [Imagen 3.3.1]. Es un micrófono que cuenta con un amplificador incorporado que cuenta con un potenciómetro para poder ajustar la ganancia del dispositivo, con un rango operativo de x25 hasta x125. [31]

A continuación se presentan las especificaciones del dispositivo. [Tabla 3.3.1]

Voltaje Operativo	2.4 - 5 V
Frecuencias de Trabajo	20Hz - 20 KHz
Voltaje de Salida	hasta 5Vp-p
Ganancia	x25 - x125

Tabla 3.3.1

Especificaciones Micrófono Electret MAX4466

Fuente: https://www.digikey.es

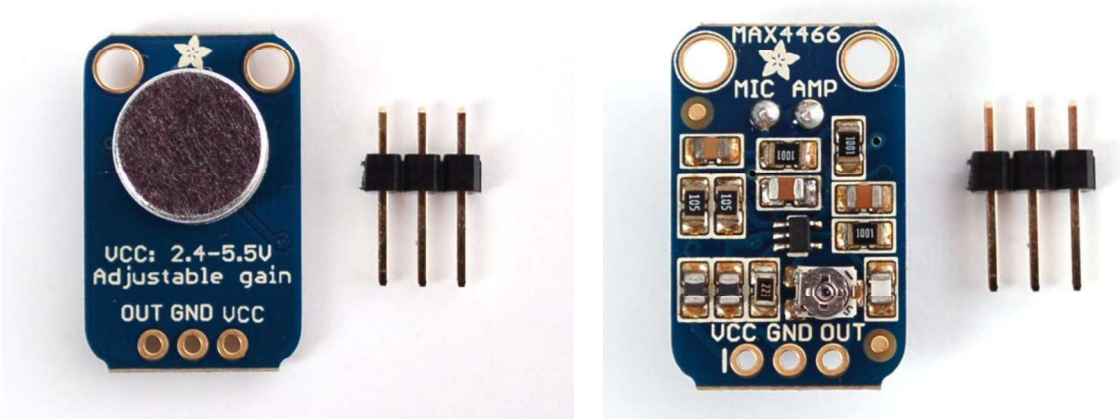


Imagen 3.3.1

Micrófono Electret MAX4466

Fuente: https://www.digikey.es

3.4. Montaje del Hardware

Para que el futuro sistema de identificación funcione correctamente, primero es necesario conectar las partes analizadas en los puntos anteriores entre ellas; por suerte otra de las características de Arduino es que todos sus componentes se conecta fácilmente entre ellos.

El primer paso será conectar el shield a la placa Arduino, para ello, prestamos atención a la numeración de los pines en ambas partes para conectarlos en la misma posición. Dicha numeración se encuentra en el lateral de cada componente, con lo que no deberíamos de tener ningún problema para conectar ambas partes. Una vez estén unidas, deberíamos ver lo siguiente [Imagen 3.4.1]

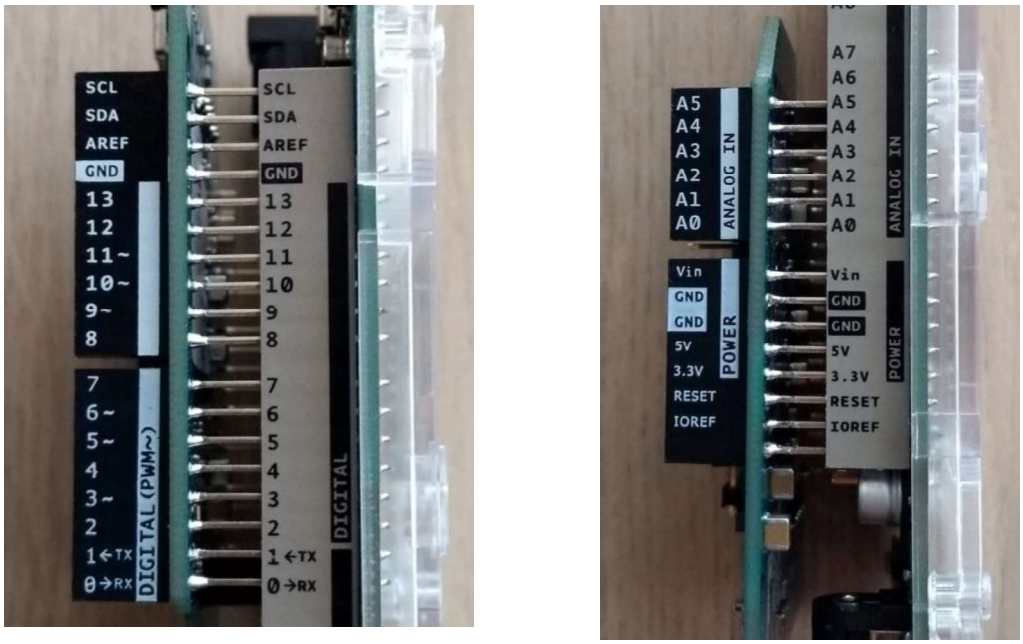


Imagen 3.4.1
Conexión Placa Arduino y Wifi Shield
Fuente: Propia

Por último tendremos que conectar el micrófono al Shield; como se ha podido observar en la Imagen 3.3.1, el micrófono tiene 3 conexiones: una conexión a Tierra, una conexión de alimentación y una conexión de salida de datos. Para que el micrófono funcione correctamente se conectará de la siguiente manera: [Imagen 3.4.2]

- Conectamos la tierra del micrófono a la tierra del shield.
- Conectamos la alimentación del micrófono a la salida de 3’3V del shield.
- Conectamos la salida del micrófono al pin analógico “A0” del shield.

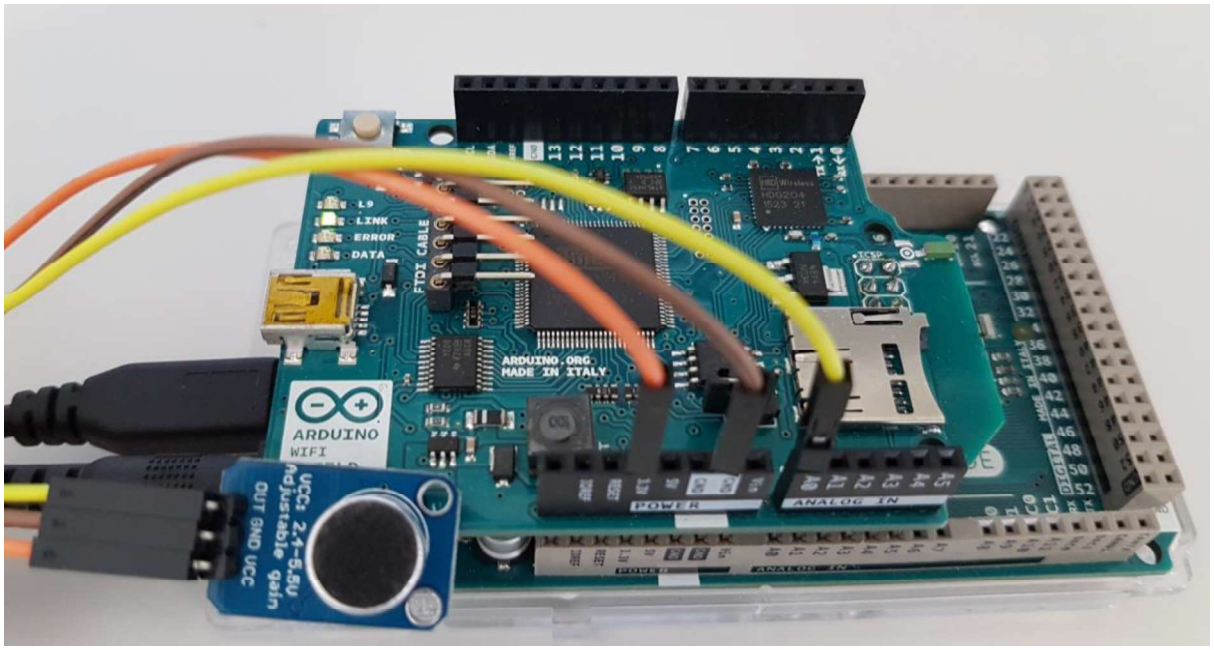


Imagen 3.4.2

Hardware Completo.

Fuente: Propia.

4. Solución.

En este apartado ya se desarrolla la solución por la que se ha optado para este TFG, así como su implementación en las distintas partes que conforman este trabajo. Antes de entrar en cada implementación se dará una imagen global de la solución de forma teórica.

Después se presentará la solución en los escenarios de Matlab y Arduino respectivamente y para finalizar se mostrará cómo se establece la comunicación entre la placa Arduino y el servicio web, así como la estructura de este último y de la base de datos integrada para la recogida de datos.

4.1. Definición de la solución.

4.1.1. Tratamiento del Audio.

Tras haber realizado la investigación previa al desarrollo de la investigación, se comenzó delimitando el número de especies para las que se iba a implementar el sistema de identificación y se ha trabajado en base al análisis de las siguientes: Palomas, Cuervos, Gaviotas, Gorriones y Estorninos.

Como se ha elegido la identificación por audios, en este proyecto se hará uso de la Transformada de Fourier (de ahora en adelante FT) ya que una de sus utilidades más popular es la de descomposición de señales acústicas en sus tonos fundamentales.

Con la FT lo que haremos será trasladar las señales que queramos analizar, en este caso el canto de las aves, desde el dominio del tiempo, al dominio de la frecuencia [32], donde podremos extraer información mucho más útil a la hora de extraer parámetros significativos que nos ayuden a distinguir unas aves de otras.

Una vez se han obtenido los parámetros característicos de cada ave, se elaboró el sistema de identificación de las mismas a través de los valores obtenidos.

4.1.2. Elaboración de biblioteca de audios.

Para el desarrollo del trabajo y del entorno de estudio de las aves se necesitan de ficheros de sonido que fueran representativos de las mismas, así que para ello se elaboró una pequeña biblioteca que recoge varias muestras de cada especie con las que poder elaborar un sistema de identificación lo suficientemente fiable. En esta biblioteca encontraremos para cada especie 19 muestras de audio, de las cuales el 50% (10 en este caso) son utilizadas para la extracción de los parámetros con los que construir el filtro de identificación y el otro 50% para realizar pruebas de eficacia.

Como se han elegido tantas muestras de audio y dado que no se contaba con el equipo para realizar grabaciones exentas de ruido, se ha recurrido a un portal virtual en el que aficionados a la ornitología suben sus grabaciones [33].

4.1.3. Comunicación con el Servidor.

Como el objetivo de este proyecto, es el de gestionar la identificación de las plagas de aves, para la parte de gestión se ha configurado un servicio web, el cual recoge en una base de datos todas las identificaciones que el sistema vaya realizando. Este servicio web constará de una visualización de la base de datos, la cual recogerá el ave que se ha identificado, la fecha y la hora de la identificación y además se le añadirán algunas funciones extra como la búsqueda de un espécimen concreto en la base de datos, o una función de búsqueda por fecha u hora.

De esta manera podremos mantener un registro sobre qué aves se identifican, así como la frecuencia con la que se realizan identificaciones exitosas. Almacenando la fecha y la hora se podrán tomar medidas para gestionar la presencia de dichos animales preparando medidas para los momentos en los que la presencia de estas aves sea más densa.

4.2. Implementación en Matlab.

Tras la introducción teórica a la solución, a continuación se desarrolla la implementación de la misma a través de Matlab. En este apartado se mostrará el código implementado separado en partes, para facilitar tanto la explicación como su comprensión.

4.2.1. Análisis Previo del Audio.

La primera parte en la que podemos dividir la solución es el tratamiento previo que se le da a los ficheros para la extracción de información, así que nuestro código comienza con la carga del fichero que queremos analizar. Como la mayoría de estos ficheros han sido grabados en estéreo, vamos a realizar un paso previo en el que nos quedaremos con el sonido de uno de los altavoces, para poder trabajar como si tuviéramos un archivo grabado en mono:

```
pista = input('Introduce el audio que quieres analizar: ');
[x,fs]=audioread(pista);
x = x(:,1);
L = length(x);
```

De esta manera ya tendremos almacenado en la variable “x” la información en el dominio del tiempo del fichero, en la variable “fs” la frecuencia de muestreo del fichero de audio y en “L” el número de muestras del fichero total.

El siguiente paso es calcular la potencia del fichero total ya que es necesaria para poder descartar tramas con silencio o con ruido en los pasos siguientes, así que procedemos a calcularla con la siguiente instrucción:

```
potencia = sum(x.*x)/length(x);
```

4.2.2. Separación en Tramas.

Una vez que hemos cargado el fichero que queremos analizar y hemos obtenido unos parámetros iniciales del mismo vamos a proceder a la separación en tramas del mismo. Aquí aparecen los que serán los parámetros de configuración que se verán en el punto 5, para afinar el sistema de identificación. Con eso, encontramos las siguientes instrucciones:

```
porcentaje = 0.1;
tiempotrama = 0.05;
tamanotrama = fs*tiempotrama;
s = 0.9;
ntramas = fix((L-tamanotrama)/((1-s)*tamanotrama))+1;
```

Así establecemos que:

- La potencia de la trama ha de ser superior al 10% de la potencia del audio para poder analizarse.
- El tiempo que van a durar las tramas es de 50 milisegundos.
- Que el tamaño de la trama será la duración de estas por la frecuencia de muestreo del fichero original, de esta manera nos aseguramos que aunque dos ficheros tengan frecuencias de muestreo diferentes, sus tramas tendrán el mismo número de muestras.
- El solapamiento entre las tramas será del 90%.
- El número de tramas en el que se va a separar el fichero original.

Una vez tenemos que estos parámetros han sido calculados se procede a la separación del fichero original en las tramas resultantes. Para ello construiremos el siguiente bucle:

```
for t=1:ntramas
xtrama = x((t-1)*fix(tamanotrama*(1-s))+1:(t-1)*fix(tamanotrama*(1-s))+tamanotrama);
L_2 = length(xtrama);
pot_xtrama=sum(xtrama.*xtrama)/length(xtrama);
 if pot_xtrama>potencia*porcentaje
...
```

El bucle recorrerá el fichero original hasta haber rellenado el número de tramas calculadas anteriormente y lo primero que hace es obtener las muestras correspondientes a la trama que está rellenando. Una vez está la trama rellena, se procede a calcular la potencia de la misma, para después comparar esta potencia de trama con la potencia original y decidir si se analiza o no (En el caso de que la trama solo tenga muestras de silencio, se descarta).

4.2.3. Transformada de Fourier.

El siguiente paso, una vez que la trama pasa el filtro de potencia es realizar la FT para poder obtener la información en frecuencia de la misma. Esto lo realizamos con la función implementada en Matlab llamada "fft" que viene de "Fast Fourier Transform". Una vez que realicemos la "fft" como Matlab nos devuelve un espectro en el que podemos observar tanto las frecuencias positivas como las negativas que existen meramente como propiedades matemáticas de la Transformada de Fourier, así que lo que haremos será quedarnos solo con las frecuencias positivas para facilitar la representación del espectro. Para terminar almacenamos en una variable llamada "aux_trama" las frecuencias correspondientes a las muestras de la trama. Esto queda representado en el siguiente código.

```
Xtrama = fft(xtrama);
Xtrama_2 = abs(Xtrama/L_2);
Xtrama_1 = Xtrama_2(1:L_2/2+1);
Xtrama_1(2:end-1) = 2*Xtrama_1(2:end-1);
aux_trama = fs*(0:(L_2/2))/L_2;
```

Si representamos la información que hemos obtenido después de realizar la transformada nos encontramos con lo siguiente [Imagen 4.2.3.1].

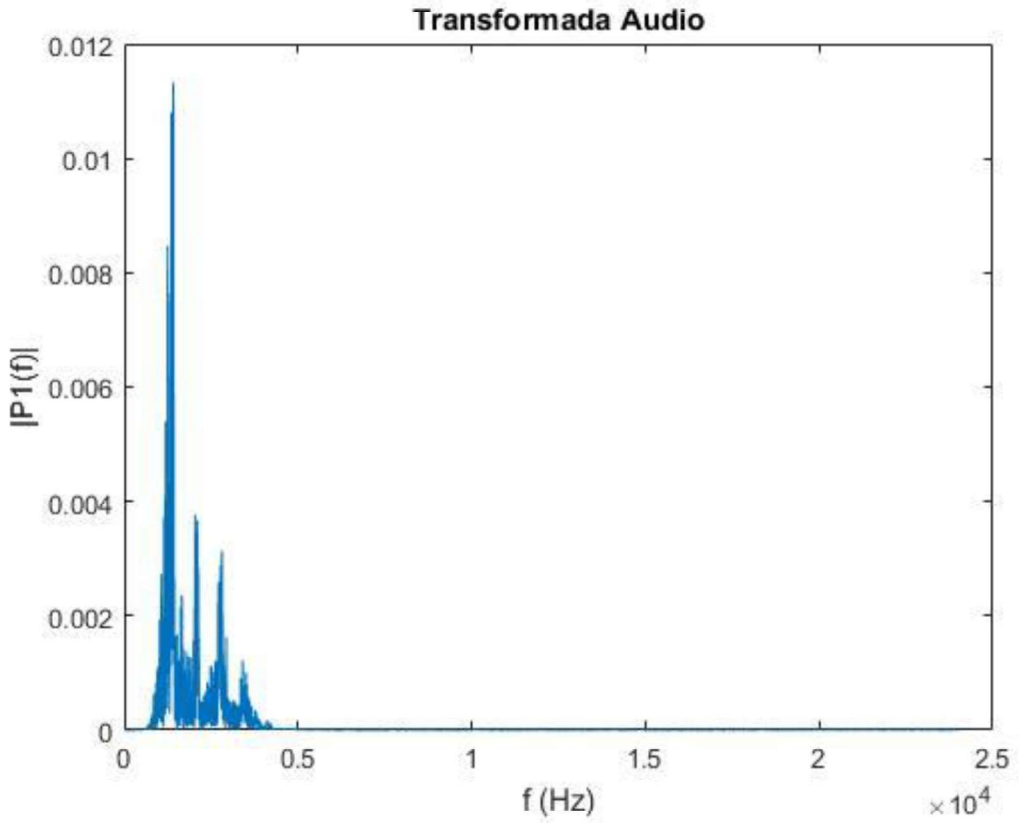


Imagen 4.2.3.1

Transformada de Fourier de una trama de un fichero de audio.

Fuente: Propia.

Para poder generar esta gráfica se ha utilizado el siguiente código:

```
figure;
plot(aux_trama,Xtrama_1);
title('Transformada Audio');
xlabel('f (Hz)');
ylabel('|P1(f)|');
```

4.2.4. Extracción de Parámetros.

Antes de empezar con el comentario del código se va a incluir un diagrama de bloques del procedimiento seguidos hasta la ejecución completa del código. [Imagen 4.2.4.1]

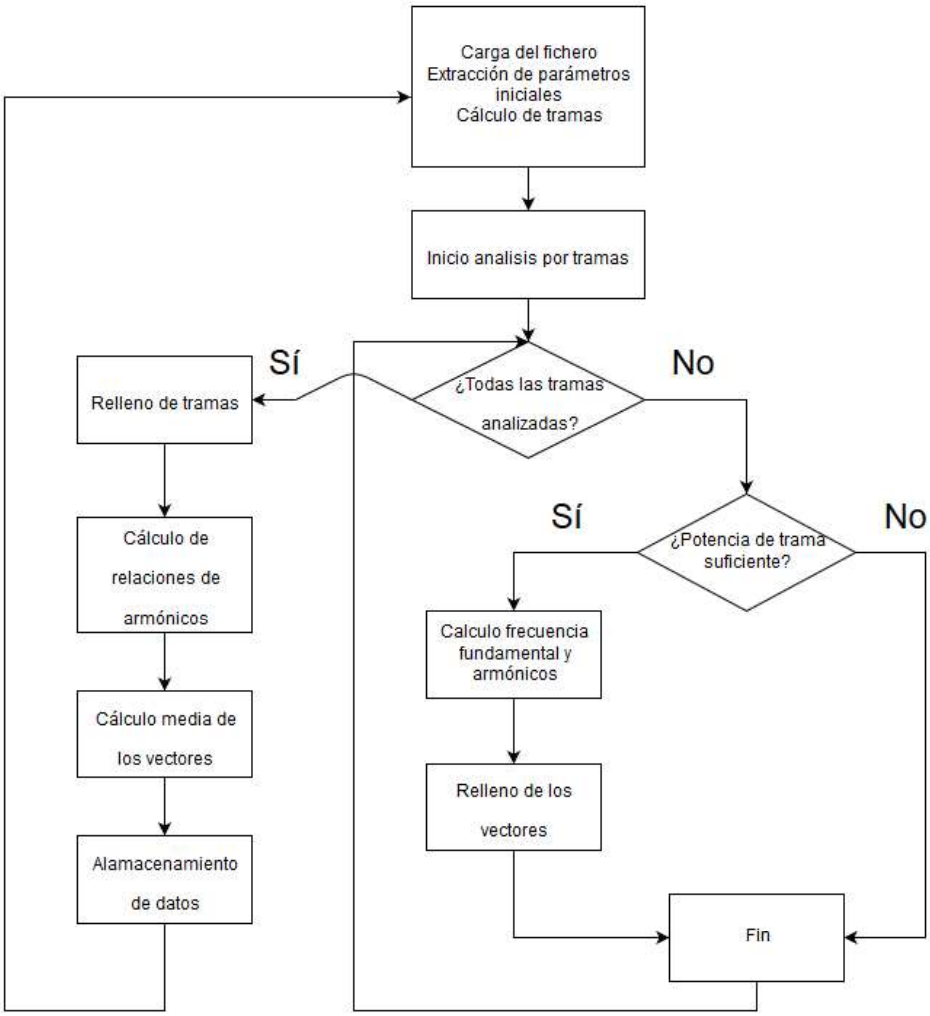


Imagen 4.2.4.1

Diagrama de bloque de la función de extracción de parámetros.

Fuente: Propia.

Una vez que hemos realizado la “fft” ya tenemos almacenada la información en frecuencia de la trama y esa es la información que necesitamos para la elaboración del sistema de decisión. Como puede verse en la Imagen 4.2.3.1, observamos unos picos significativos en comparación con el resto de valores, esos picos son la frecuencia fundamental y los armónicos de esa trama en concreto por lo que los almacenaremos en un vector para así tener la información de todas las tramas.

Lo primero que necesitamos averiguar es el valor de la frecuencia fundamental, y para ello lo que hacemos es buscar el valor máximo de la trama y almacenarlo junto con su posición. Una vez que tenemos estos valores, buscamos en la frecuencia en la que se encuentra y así obtendremos la frecuencia fundamental de la trama. Esto se realiza de la siguiente manera:

```
[m_1, p_1]=max(Xtrama_1);
freq=aux_trama(p_1);
```

El siguiente paso es localizar los armónicos que pueda haber tras realizar la “fft”, para ello, como ya tenemos identificada la frecuencia fundamental, buscamos en los múltiplos de la misma y establecemos un rango en el que buscar de nuevo los valores máximos para obtener así las frecuencias de los armónicos. En este proyecto solo se han considerado el primer y el segundo armónico ya que la consideración de más componentes no aportaba resultados mejores que con solo dos armónicos. A continuación se muestra como se realiza la búsqueda de los armónicos:

```
a_1 = find ((aux_trama >= (freq*2)-rang) & (aux_trama <= (freq*2)+rang));
a_2 = find ((aux_trama >= (freq*3)-rang) & (aux_trama <= (freq*3)+rang));
L_a1 = length(a_1);
L_a2 = length(a_2);
 [m_11, p_11] = max(Xtrama_1(a_1));
[m_21, p_21] = max(Xtrama_1(a_2));
```

Como solo considerar un valor para la amplitud de las frecuencias encontradas puede conducir a diferencias significativas entre valores encontrados en diferentes tramas, se calcula la media de las amplitudes para la frecuencia encontrada y las inmediatas a ella. Se ha tenido en cuenta que la frecuencia estuviera en el primer lugar de la trama y en el último lugar de la trama, a fin de evitar posibles errores si caían en estas posiciones. El código que realiza la media de estas amplitudes es el siguiente.

Cálculo de la media para las amplitudes de la frecuencia fundamental:

```
if p_1 == 1
        m_1 = (m_1+Xtrama_1(p_1+1))/2;
else
        m_1 = (Xtrama_1(p_1-1)+m_1+Xtrama_1(p_1+1))/3;
end
```

Cálculo de la media para las amplitudes del primer armónico:

```
if p_11 == 1
        m_11 = (m_11+Xtrama_1(a_1(p_11+1)))/2;
elseif p_11 == L_a1
        m_11 = (Xtrama_1(a_1(p_11-1))+m_11)/2;
else
        m_11 = (Xtrama_1(a_1(p_11-))+m_11+Xtrama_1(a_1(p_11+1)))/3;
end
```

Cálculo de la media para las amplitudes del segundo armónico:

```
if p_21 == 1
        m_21 = (m_21+Xtrama_1(a_1(p_21+1)))/2;
elseif p_21 == L_a2
        m_21 = (Xtrama_1(a_1(p_21-1))+m_21)/2;
else
        m_21 = (Xtrama_1(a_1(p_21-))+m_21+Xtrama_1(a_1(p_21+1)))/3;
end
```

Lo último que se hace antes de que se repita el bucle for para el análisis de las tramas, es el almacenamiento de los valores obtenidos en vectores, para la futura extracción de parámetros que serán utilizados en la construcción del filtro de selección. Se presenta por tanto como se realiza el relleno de los vectores donde se almacenan todas las variables:

```
 valor_f=[valor_f freq];          % Se almacena frecuencia fundamental.
valor_max=[valor_max m_1];% Se almacena valor de la freq fundamental.
valor_f_1=[valor_f_1 a_12];  % Se almacena la freq del primer armónico.
valor_max_1=[valor_max_1 m_11]; % Se almacena el valor del primer armónico.
valor_f_2=[valor_f_2 a_22];  % Se almacena la freq del segundo armónico.
valor_max_2=[valor_max_2 m_21]; % Se almacena el valor del segundo armónico.
```

Con eso el bucle “for” se irá ejecutando hasta que se rellenen todas las tramas y se obtenga la información de todo el fichero de audio en pequeñas partes. Una vez que termina la ejecución del bucle se procede a rellenar los vectores de los armónicos ya que en algunas tramas, la amplitud de estos es tan baja, o no están presenten que estos vectores quedan desfasados de el de la frecuencia fundamental. Es necesario el relleno ya que uno de los parámetros que necesitamos es la relación entre la amplitud de la frecuencia fundamental y la de los armónicos, en el caso de que los hubiera. A continuación se presenta por tanto la parte del código que realiza el relleno de las tramas, así como el cálculo de las relaciones entre las amplitudes mencionadas:

```
% Relleno de las tramas
        s1 = length(valor_f);
        s2 = length(valor_f_1);
        s3 = length(valor_f_2);

        s_21 = s1 - s2;
        s_31 = s1- s3;

        for i = 1:s_21
                valor_f_1 = [valor_f_1 0];
                valor_max_1 = [valor_max_1 0];
        end

        for i= 1:s_31
                valor_f_2 = [valor_f_2 0];
                valor_max_2 = [valor_max_2 0];
        end

% Cálculo relaciones armónicos
        rela_1 = valor_max./valor_max_1;
        rela_2 = valor_max./valor_max_2;
```

El último paso de este apartado es la extracción de los valores característicos de cada ave, con los que fabricar el sistema de identificación. Para ello se calcula la media de todas las frecuencias fundamentales obtenidas en cada trama, así como la relación que hay entre la amplitud de dicha frecuencia fundamental y de los armónicos primero y segundo, en el caso de que estuvieran presentes. Como algunas de las especies no tienen componentes de frecuencia en los armónicos, la relación que se obtiene para esos caso es "Inf" infinita, pero como con ese valor no se puede trabajar a la hora de elaborar el filtro, para los casos en los que la relación sea infinita, esta se iguala a "999". Por último, se presenta el código de este apartado:

```
% Cálculo del promedio de los valores obtenidos

    promf = mean(valor_f);
    prom_rela1 = mean(rela_1);
    prom_rela2 = mean(rela_2);

    if prom_rela1 == Inf
            prom_rela1 = 999;
    end

    if prom_rela2 == Inf
            prom_rela2 = 999;
    end
```

4.2.4.1. Parámetros característicos de las palomas.

Tras realizar el análisis de los distintos ficheros de la especie de las palomas se han obtenido los siguientes parámetros [Imagen 4.2.4.1.1] [Imagen 4.2.4.1.2]

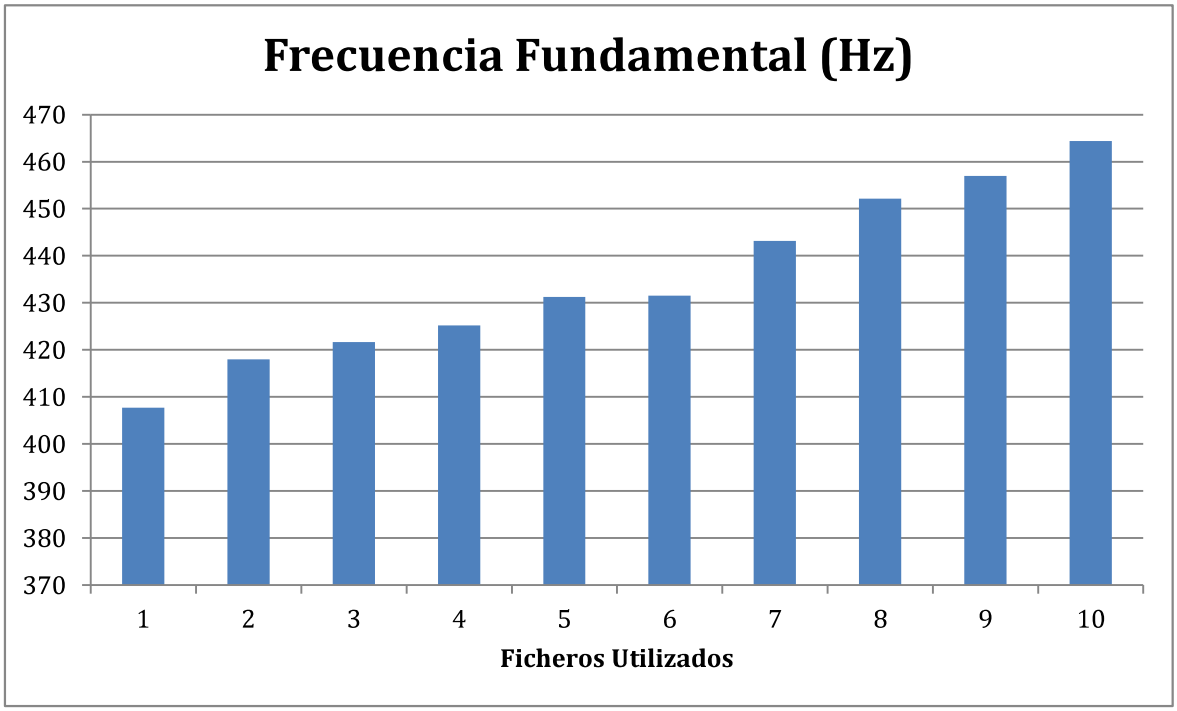


Imagen 4.2.4.1.1
Frecuencias Fundamentales de las Palomas.
Fuente Propia.

Como se puede apreciar en la gráfica, (y como se verá en comparación con el resto de aves analizadas) las frecuencias fundamentales que se han obtenido para esta especie son relativamente bajas, y ocupan un ancho de banda muy reducido. Esto nos permitirá separar a las palomas de manera muy eficiente del resto de aves, debido a estos dos factores.

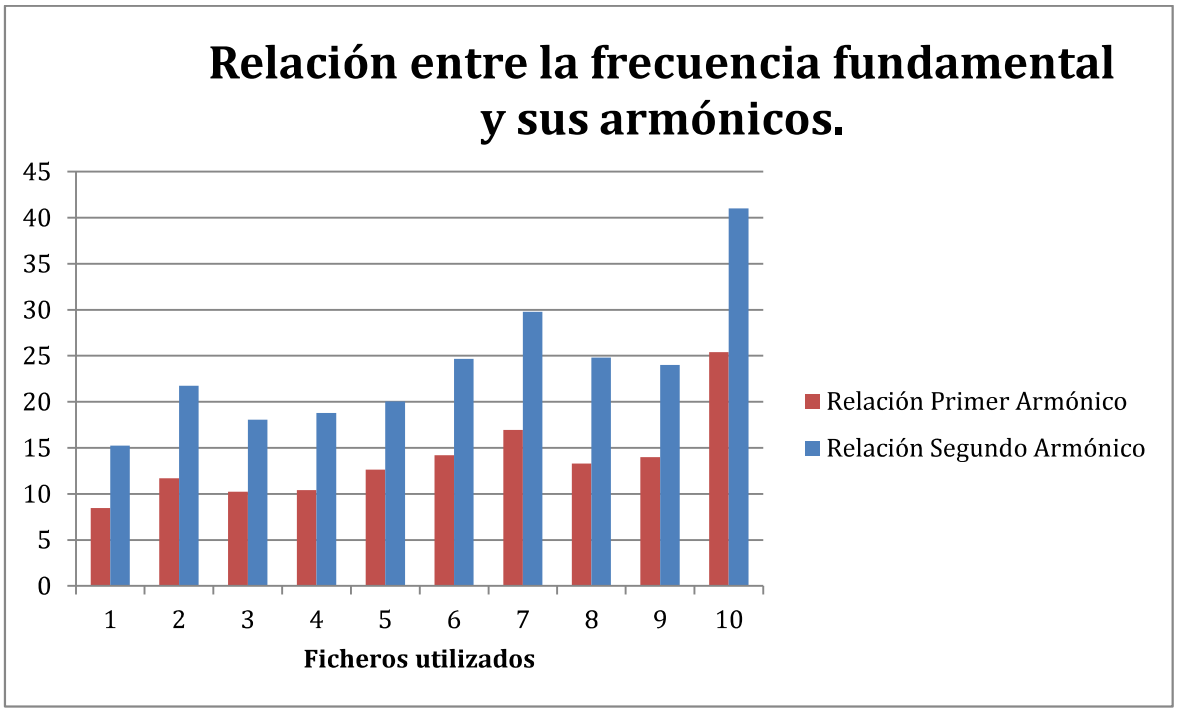


Imagen 4.2.4.1.2
Relación entre frecuencia fundamental y sus armónicos de las Palomas.
Fuente Propia.

En cuanto a las relaciones obtenidas entre los armónicos y las frecuencias fundamentales mostradas arriba, lo primero que observamos es que la diferencia entre el primer y el segundo armónico con respecto de su frecuencia fundamental es notable. Esto acompañado de que la diferencia entre los armónicos y la frecuencia fundamental es creciente en cuanto avanzamos en el eje de frecuencias nos permite indicar que la carga de información se encuentra en la zona de la frecuencia fundamental.

4.2.4.2. Parámetros característicos de las gaviotas.

Tras realizar el análisis de los distintos ficheros de la especie de las gaviotas se han obtenido los siguientes parámetros [Imagen 4.2.4.2.1] [Imagen 4.2.4.2.2]

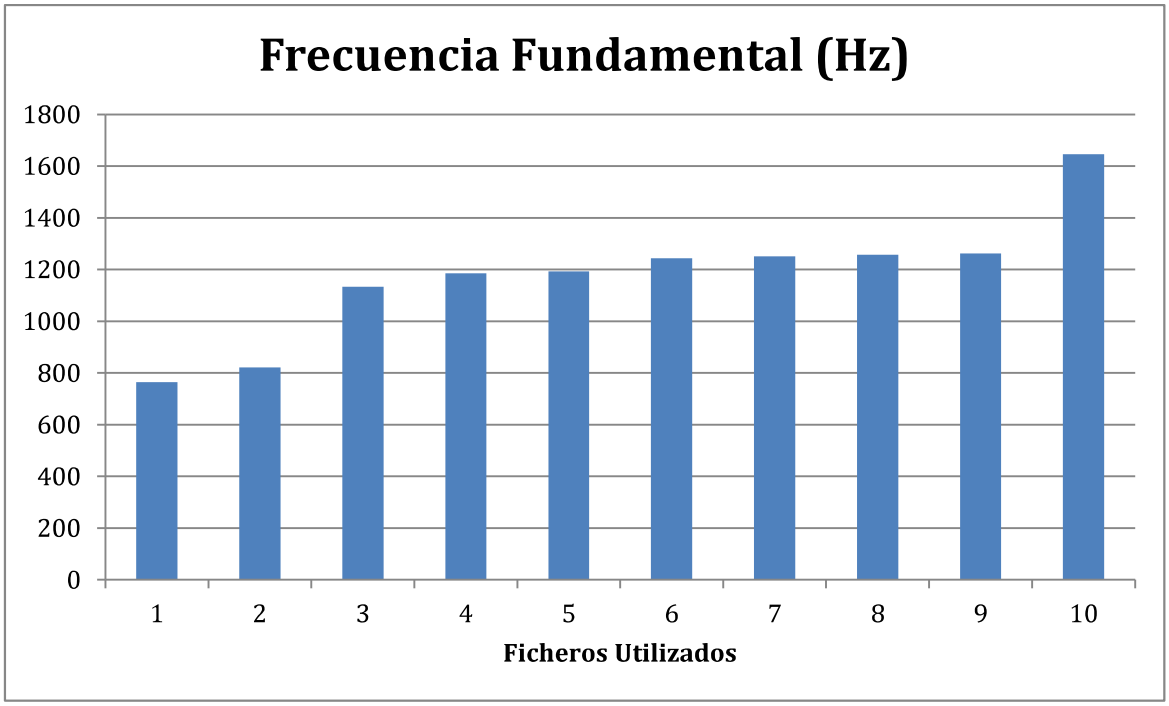


Imagen 4.2.4.2.1
Frecuencias Fundamentales de las gaviotas.
Fuente: Propia

Como podemos observar en las gráficas, la frecuencia fundamental del canto de las gaviotas está entre los 790 Hz (valor mínimo obtenido durante la extracción de datos), y los 1640 Hz (valor máximo obtenido durante la extracción de datos). Estos valores como se verán a continuación ocupan la misma banda de frecuencia que la de los cuervos, por lo que diferencia a ambas especies es el otro factor que se ha tenido en consideración, la relación entre estas frecuencias fundamentales, y sus armónicos.

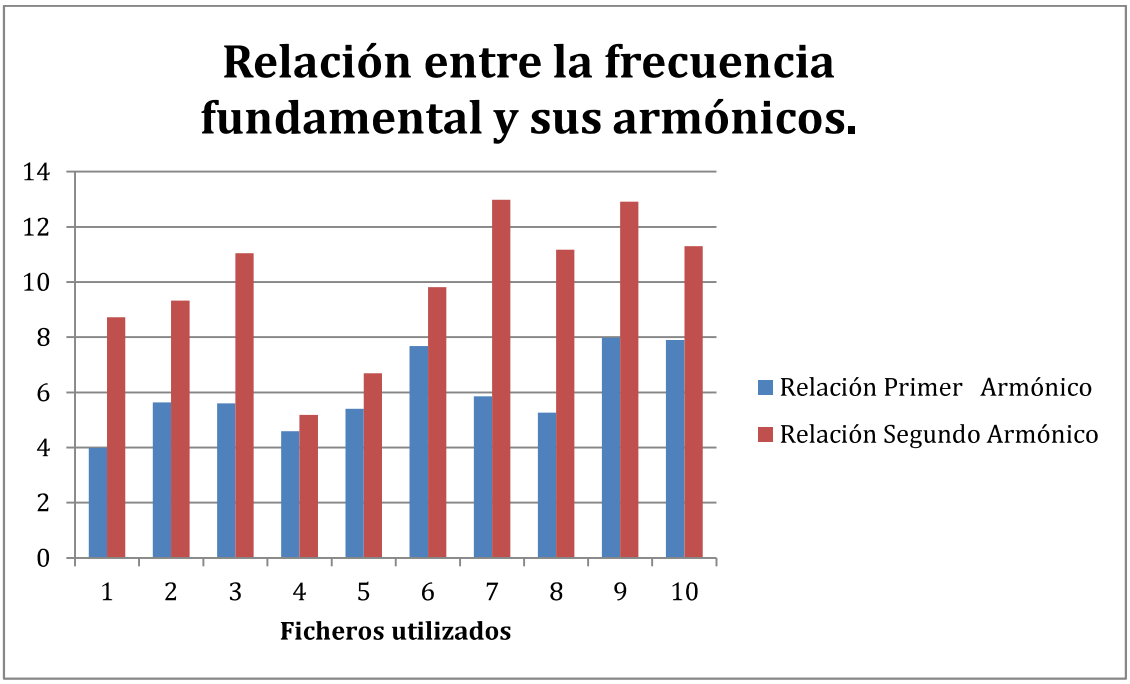


Imagen 4.2.4.2.2
Relación entre frecuencia fundamental y sus armónicos de las gaviotas.
Fuente: Propia

Tal y como se observa en su gráfica la relaciones obtenidas son muy reducidas, lo cual indica que la caracterización de esta especie tendrá más importancia por este factor, que por el de su frecuencia fundamental. Tras analizar los datos, la relación más reducida que encontramos entre armónico y su fundamental se encuentra entre 4 veces menor (en amplitud), mientras que el caso donde más diferencia hay es de 8 veces menor.

Para el caso de las relaciones entre el segundo armónico y su frecuencia fundamental, no encontramos valores tan separados de los anteriores, siendo la menor de las relaciones encontradas unas 4’5 veces menor, mientras que la mayor diferencia para estos ficheros supera ligeramente las 12 veces.

4.2.4.3. Parámetros característicos de los cuervos.

Tras realizar el análisis de los distintos ficheros de la especie de los cuervos se han obtenido los siguientes parámetros [Imagen 4.2.4.3.1] [Imagen 4.2.4.3.2]

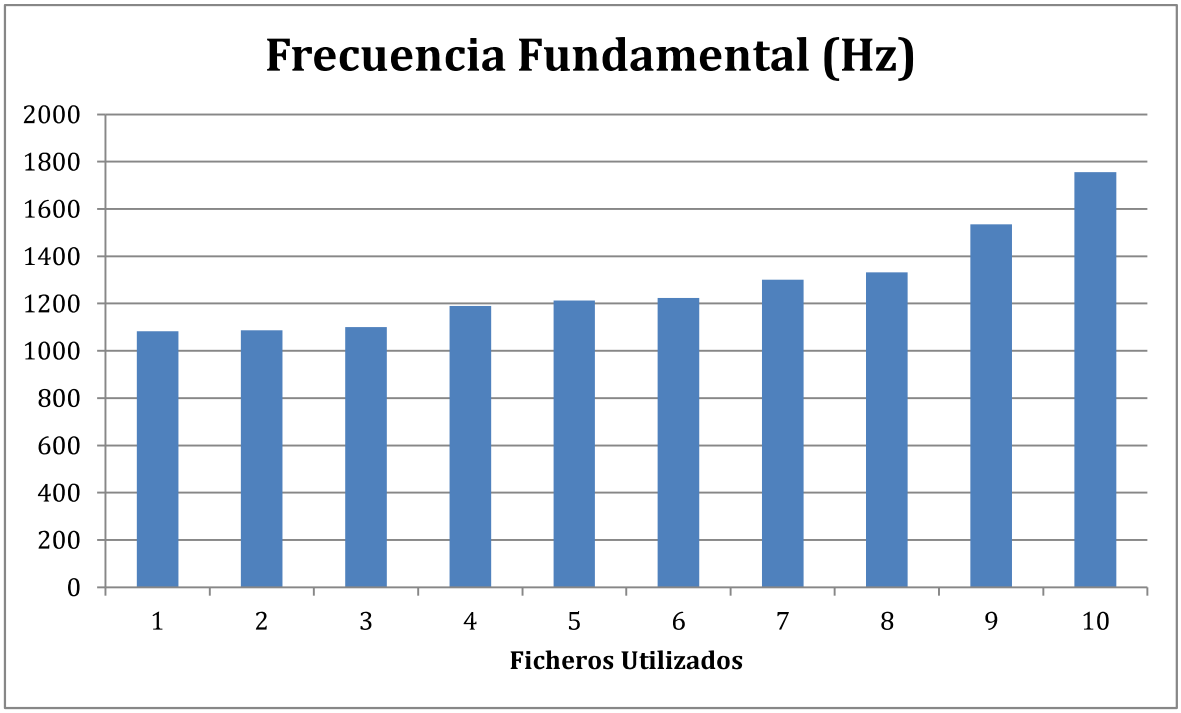


Imagen 4.2.4.3.1
Frecuencias Fundamentales de los cuervos.
Fuente: Propia

Como se puede observar en la gráfica, y como se ha mencionado anteriormente, las frecuencias fundamentales pertenecientes a los cuervos, ocupan el mismo ancho de banda que el de las gaviotas, por lo que si nuestro sistema solo contara con este parámetro como único factor de decisión, sería incapaz de diferenciar a una de la otra.

Además podemos observar como en los dos últimos ficheros la frecuencia fundamental tiene un valor significativamente más alto que los primeros, y tal y como hemos visto en el punto donde se estudiaba la morfología del órgano vocal de las aves, esto puede deberse a cualquier variación del sonido producido, ya sea por alarma, llamada de crías u otra situación.

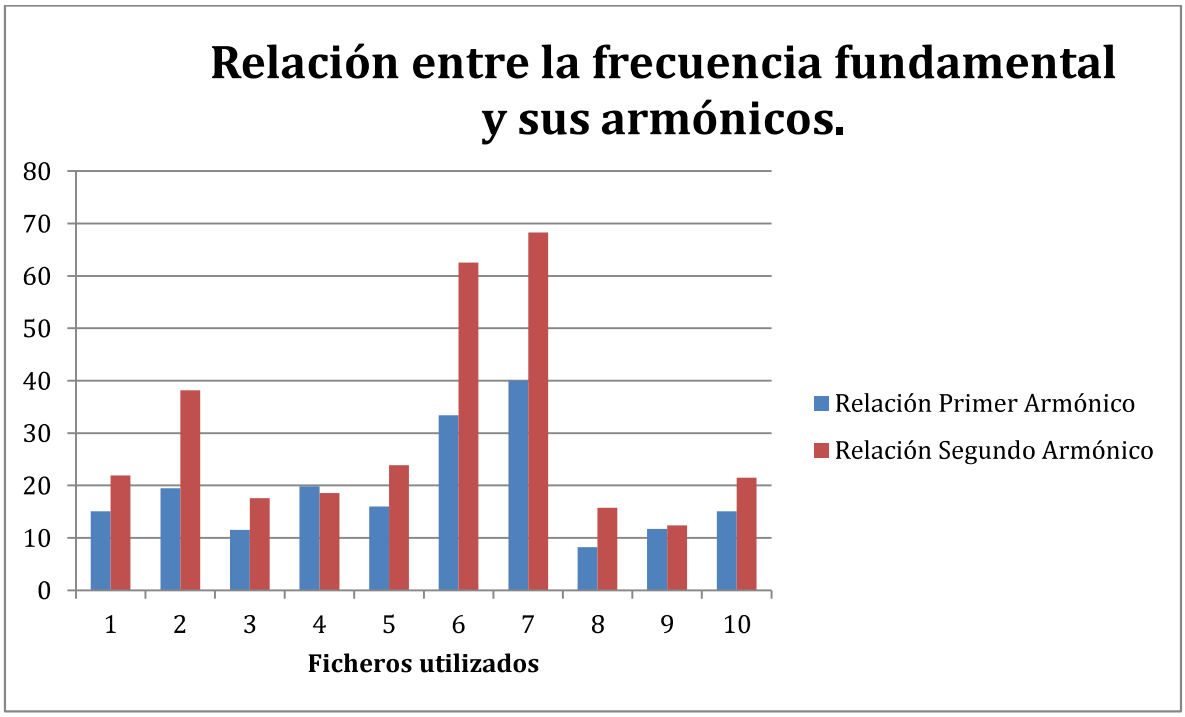


Imagen 4.2.4.3.2
Relación entre frecuencia fundamental y sus armónicos de los cuervos.
Fuente: Propia

Tras la extracción de los armónicos y calcular la relación que estos tenían con su frecuencia fundamental, para el caso de los cuervos, la mayoría de la información recae en la frecuencia fundamental, lo cual nos permite darle más importancia a dicho factor en contra de a estas relaciones.

Esto puede apreciarse en la gráfica incluido arriba, donde se pueden observar que se obtienen valores mucho más elevados que para la especie anterior, con algunas excepciones como son el archivo 8 y 9, aunque estos archivos no influyeron en la detección de las especies, como se verá en el apartado 5.

4.2.4.4. Parámetros característicos de los estorninos.

Tras realizar el análisis de los distintos ficheros de la especie de los estorninos se han obtenido los siguientes parámetros [Imagen 4.2.4.4.1] [Imagen 4.2.4.4.2]

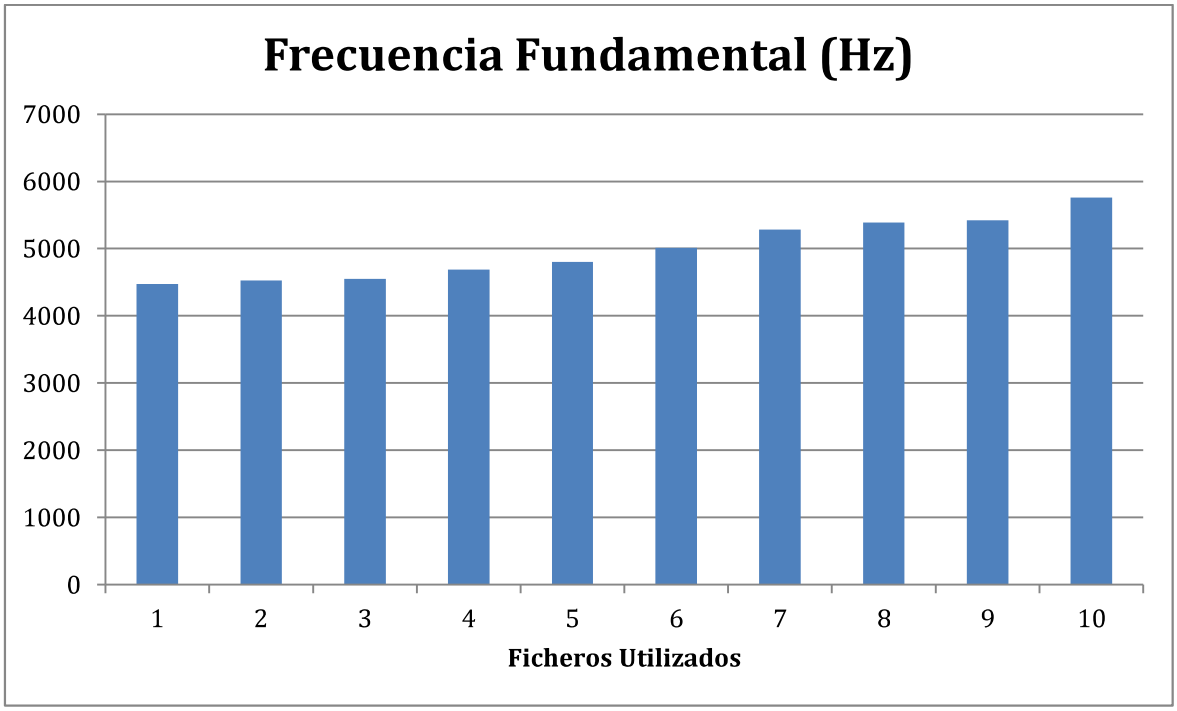


Imagen 4.2.4.4.1
Frecuencias Fundamentales de los estorninos.
Fuente: Propia

Tras el análisis de los estorninos, de nuevo observamos que el ancho de banda que ocupa esta especie, aunque es un poco más elevado que en el resto de las consideradas, se mantiene estable para los archivos analizados. Estas aves por tener un tamaño más reducido que las anteriores necesitan de un tono con frecuencias más altas que las demás para que se pueda identificar bien a los individuos en casos de que haya otras especies presentes. Una de las ventajas que podemos extraer tras el análisis de esta especie, y como veremos en la tabla siguiente, es que son aquellos que tienen la frecuencia fundamental más elevada de todas las especies analizadas, lo cual facilita la labor de reconocimiento ya que son los únicos presentes en la banda de los 4 KHz a los 6 KHz.

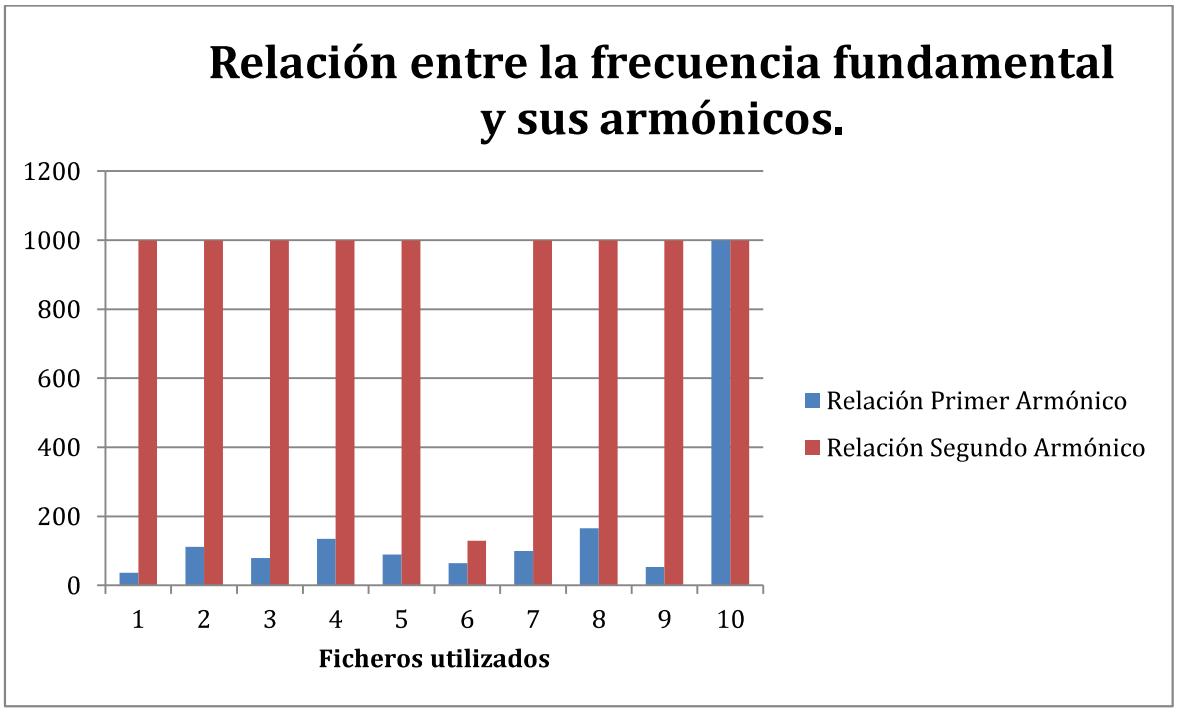


Imagen 4.2.4.4.2
Relación entre frecuencia fundamental y sus armónicos de los estorninos.
Fuente: Propia

Tal y como se había mencionado en el apartado anterior de esta especie, la relación existente entre los armónicos (sobre todo el segundo), y la frecuencia fundamental, ha tenido unas características únicas en el entorno de estudio. Como se puede observar en la gráfica, la relación entre el primer armónico y la frecuencia fundamental para los estorninos, aunque sea muy elevada está presente para todos los casos, a excepción del último, lo cual podría intervenir con el de otras especies, pero para el caso del segundo armónico encontramos que la relación entre ambos es nula; para todos los casos a excepción del sexto, se ha obtenido que la relación que guardan estos parámetros es infinita, lo cual a la hora de implementar el sistema de decisión nos va a facilitar mucho identificar a esta especie, ya que es la única de las analizadas que tiene este comportamiento.

4.2.4.5. Parámetros característicos de los gorriones.

Tras realizar el análisis de los distintos ficheros de la especie de los gorriones se han obtenido los siguientes parámetros [Imagen 4.2.4.5.1] [Imagen 4.2.4.5.2]

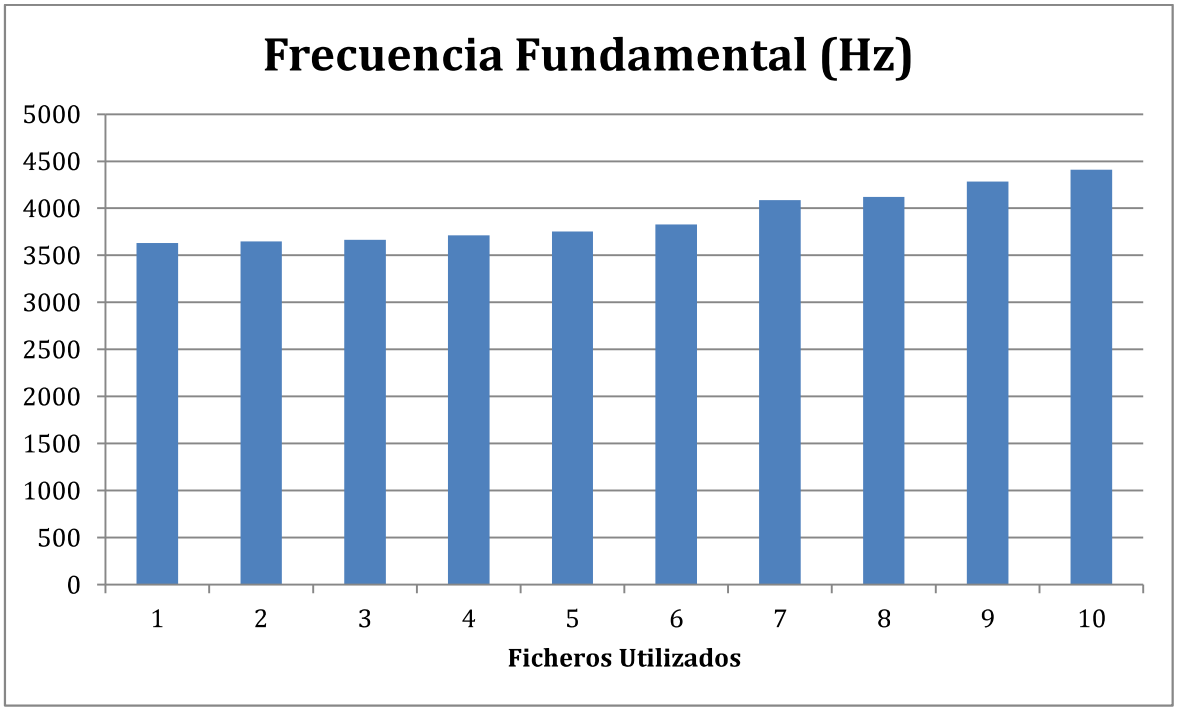


Imagen 4.2.4.5.1
Frecuencias Fundamentales de los gorriones.
Fuente: Propia

Por último, se presentan los datos extraídos de los gorriones. De nuevo volvemos a encontrar una relación entre el tamaño del ave y el tono que esta produce, ya que tanto el estornino visto en el apartado anterior, como los gorriones, son las especies cuyo tamaño es más reducido del grupo de estudio, y son las dos que presentan las frecuencias fundamentales más altas.

Para el caso de los gorriones, las frecuencias extraídas se encuentran un poco por debajo en frecuencia de la de los estorninos, ocupando las banda de los 3'5 KHz, hasta los 4'5 KHz sin llegar a alcanzar este último. Aunque coincidan en frecuencia con los estorninos en algunos casos, como se ha explicado antes, no debería de haber problemas con la identificación por la situación particular de los estorninos y sus armónicos.

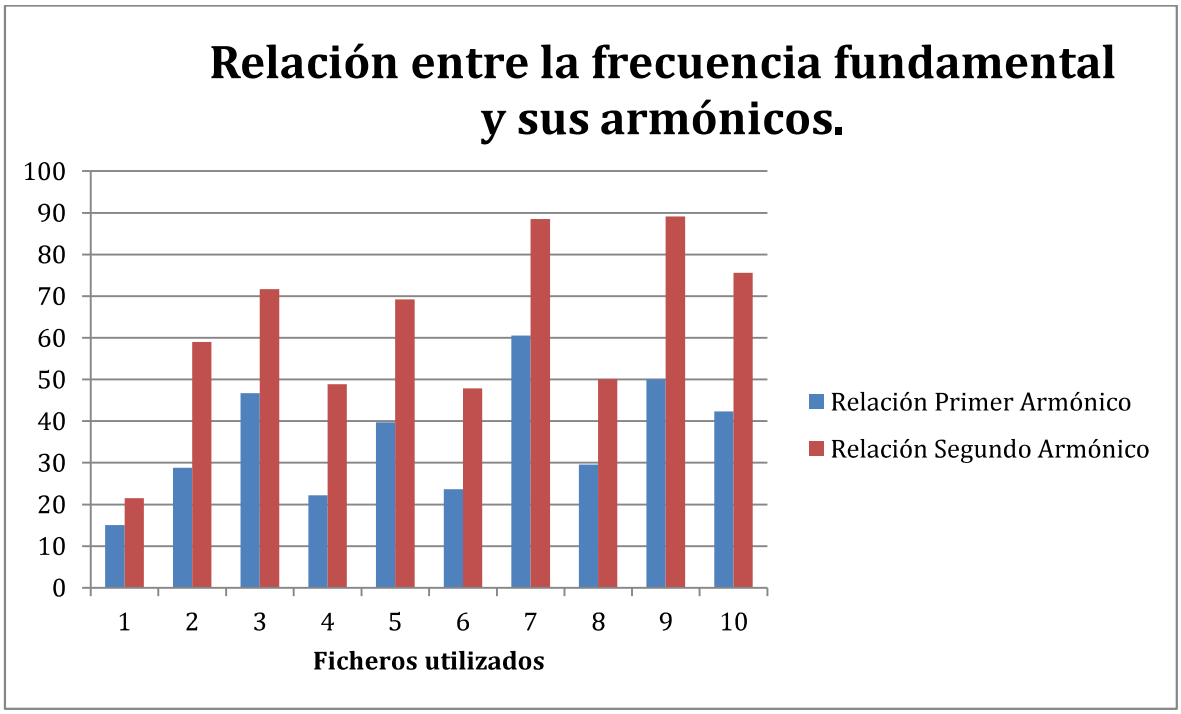


Imagen 4.2.4.5.2
Relación entre frecuencia fundamental y sus armónicos de los gorriones.
Fuente: Propia

Para el caso de los estorninos volvemos a encontrar una distribución más normal en cuanto a la relación de los armónicos y la frecuencia fundamental obtenida. Aunque si es cierto que para los segundos armónicos las relaciones ya comienzan a ser un poco excesivas, que se mantengan dentro del rango de 100 veces menor, nos va a permitir poder diferenciarlas de las aves con las que comparten banda de frecuencia, como se ha explicado anteriormente.

4.2.5. Filtro de Identificación.

Una vez se han realizado los apartados anteriores con los ficheros seleccionados para la configuración del sistema de identificación acabamos con los siguientes datos. Tenemos las tablas de cada especie, en la que encontramos los valores explicados en el apartado 4.2.4. Para cada uno de los ficheros tenemos una frecuencia fundamental, la relación entre dicha frecuencia fundamental con su primer y segundo armónico.

Con estas tablas podemos elaborar los criterios para el filtro de selección, el cual tiene la siguiente configuración [Tabla 4.2.5.2]

	Frecuencia fundamental	Relación Primer Armónico	Relación Segundo Armónico
Paloma	300-600 Hz	1 - 65	1 - 93
Gaviota	1000-1700 Hz	2 - 8	4 - 12
Cuervo	1000-2200 Hz	8 - 110	12 - 155
Gorrión	3000-4000 Hz	12 - 70	24 – 100
Estornino	4000-6500 Hz	> 110	> 187

Tabla 4.2.5.2
Criterios para el filtro de selección de Ave
Fuente: Propia.

De nuevo, antes de volver con el código se incluye un diagrama de bloques de la función utilizada para el sistema de identificación. [Imagen 4.2.5.2.1]

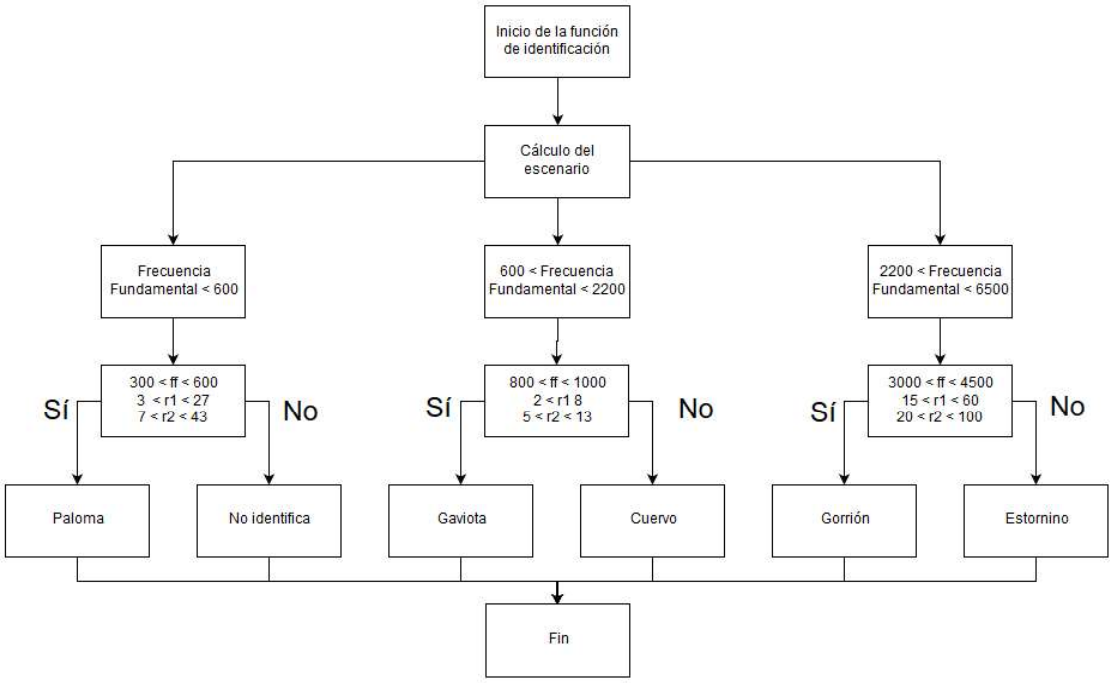


Imagen 4.2.5.2.1

Diagrama de bloques de la función del sistema de identificación.

Fuente: Propia

El filtro se ha implementado en tres partes; la primera que sirve para seleccionar en qué escenario nos encontramos, habiendo un total de tres escenarios: uno para frecuencias bajas, otro para frecuencias medias y otro para frecuencias altas. La parte de la identificación de escenario se ha realizado con condiciones “if” de las que se obtiene el escenario en cuestión:

```
if promf < 600
        escenario = 1;
else if promf < 2200
        escenario = 2;
        else if promf < 6500
                escenario = 3;
        else escenario = 4;
                end
```

```
            end
        end
```

La segunda parte es un “switch case”, en el que dependiendo del escenario en el que nos encontremos se procede a la identificación del ave a través de los parámetros presentados en al Tabla 2.4.5.2. Una vez se cumplen las condiciones para identificar a un ave, se asigna un valor a un parámetro, que servirá para tomar la decisión final; ya que nuestro filtro depende de tres variables, frecuencia fundamental, relación con el primer armónico y relación con el segundo armónico. El código que realiza esta parte es el siguiente:

```
switch(escenario)
    case 1
        if promf > 300 && promf < 600
            resul_1 = 1;
        end
        if prom_rela1 > 3 && prom_rela1 < 27
            resul_2 = 1;
        else
        resul_2 = 0;
        end
        if prom_rela2 > 7 && prom_rela2 < 43
            resul_3 = 1;
        else
            resul_3 = 0;
        end
```

En el escenario 1, tenemos las bajas frecuencias en las que sólo encontramos a las Palomas, cuya frecuencia fundamental está por debajo de los 600 Hz. En el caso de que se cumplan las condiciones necesarias resul 1, 2 y 3 se les asignarán el valor “1”, que es el correspondiente a las palomas; y en el caso de que no se cumplan, se asigna el valor “0”, que corresponde a que no se ha podido identificar el ave, o que no se ha identificado ninguna.

```
    case 2
        if promf > 800 && promf < 1000
```

```
        resul_1 = 3;
else
        resul_1 = 2;
end

if prom_rela1 > 2 && prom_rela1 < 8
        resul_2 = 3;
else
        resul_2 = 2;
end

if prom_rela2 > 5 && prom_rela2 < 13
        resul_3 = 3;
else
        resul_3 = 2;
end
```

En el escenario 2 nos encontramos con las frecuencias medias, que son aquellas que corresponden a las Gaviotas y los Cuervos, cuyas frecuencias fundamentales oscilan entre los 800 Hz y los 2’2 KHz. El factor clave de decisión para este caso es la presencia o no de los armónicos ya que para el caso de las Gaviotas si los encontramos con unas relaciones suficientemente relevantes y para el caso de los cuervos no. En el caso de que se cumplan las condiciones para la identificación de las Gaviotas, a resul se le asignará el valor 3 y para los Cuervos, se le asignará el valor 2.

```
case 3
        if promf > 3000 && promf < 4450
                resul_1 = 4;
        else
                resul_1 = 5;
        end

        if prom_rela1 > 15 && prom_rela1 < 60
                resul_2 = 4;
        else
                resul_2 = 5;
        end

        if prom_rela2 > 20 && prom_rela2 < 100
                resul_3 = 4;
        else
                resul_3 = 5;
        end
```

En el escenario 3 nos encontramos con las altas frecuencias, que son aquellas que corresponden a las Gorriones y los Estorninos, cuyas frecuencias fundamentales oscilan entre los 2’5 KHz y los 7 KHz. En este caso el factor de decisión se comparte entre los tres valores de forma homogénea. En el caso de que se cumplan las condiciones para la identificación de las Gorriones, a resul se le asignará el valor 4 y para los Estorninos, se le asignará el valor 5.

```
case 4
        resul_1 = 0;
        resul_2 = 0;
        resul_3 = 0;
end
```

El último caso corresponde a que no se esté en ninguno de los escenarios, que puede corresponder a un caso de silencio, donde no hay muestras que analizar, o que se registre un ave que no se pueda identificar.

La última parte del sistema de decisión, es la toma de decisión en sí y para ello se realiza la media de los tres valores que devuelve el "switch case" anterior. En el caso de que dos o más de los resultados correspondan a una de las aves, el sistema determina, que ha identificado a dicha ave. El código que realiza esta decisión es el siguiente:

```
final = (resul_1 + resul_2 + resul_3)/3;
if final == 0
        disp('No se puede reconocer el ave.')
end

if final >0.5 && final < 1.5
        disp(['El ave reconocida es: ', aves(1)]);
end

if final > 1.5 && final < 2.5
        disp(['El ave reconocida es: ', aves(2)]);
end

if final > 2.5 && final < 3.5
        disp(['El ave reconocida es: ', aves(3)]);
end

if final > 3.5 && final < 4.5
        disp(['El ave reconocida es: ', aves(4)]);
```

```
end

if final > 4.5
        disp(['El ave reconocida es: ', aves(5)]);
end
```

4.2.6. Proceso de Identificación.

En este último apartado, se explica cómo se realiza el proceso en Matlab desde que se carga un fichero, hasta que se identifica al ave del mismo. Para resumir se citarán las partes del código que ya se hayan explicado en puntos anteriores.

Los tres primeros pasos coinciden con los puntos 4.2.1, 4.2.2 y 4.2.3, es decir, que se realiza la carga del fichero de audio, se calcula el número de tramas en el que se va a dividir, se empieza la separación de las tramas y se realiza la "fft" a las tramas que se van generando. Pero a diferencia del punto 4.2.4, es este caso todavía no se realiza la extracción de parámetros ya que antes de eso se realiza una selección de tramas que contienen la información suficiente para la identificación del ave. Esto se realiza así para aliviar la carga de trabajo en Arduino y que no esté analizando constantemente muestras, sin llegar a tomar una decisión.

Para la selección de tramas válidas, cada vez que se genera una trama nueva, se almacena la información en frecuencia de esta y se compara con la siguiente trama que se analiza. De esta manera vamos contando tramas, hasta que llegamos a un número de tramas consecutivas, que tienen la misma información y así nos aseguramos de que estamos escuchando un ave y no cualquier sonido puntual que se pueda producir. Para el escenario en Matlab, una vez que se tengan diez tramas consecutivas con la misma información en frecuencia, se almacenan las cincuenta siguientes para la extracción de datos, consiguiendo así que solo se analicen en el mejor de los caso sesenta tramas, para poder realizar una identificación.

Antes de analizar el código se presenta el diagrama de bloques del conteo de tramas [Imagen 4.2.6.1]

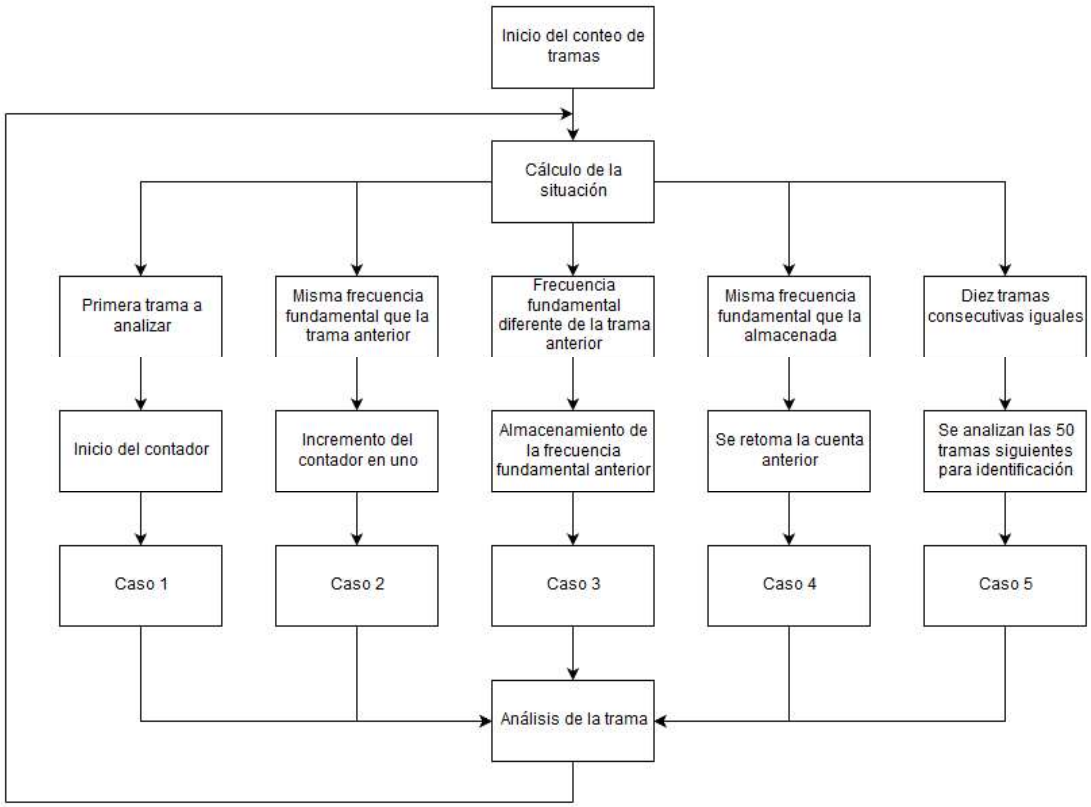


Imagen 4.2.6.1

Diagrama de bloques del conteo de tramas.

Fuente: Propia

El procedimiento con las cincuenta tramas almacenadas, es el mismo que el explicado en los puntos 4.2.4 para la extracción de datos y en el punto 4.2.5 para la identificación del ave, con la diferencia de que la carga de información con la que trabajar es mucho menor que con todas las tramas. A continuación se incluye el código de la parte que cuenta las tramas con información similar en frecuencia:

```
if caso ~= 5
        if freq ~= aux && freq ~= aux2
        caso = 1; % Caso inicial
end
```

```
if freq == aux && freq == aux2
        caso = 2; % Guarda número
end

if freq == aux && freq ~= aux2
        caso = 3; % Nuevo número
end

if freq ~= aux && freq == aux2
        caso = 4; % Recupera cuenta
end
else
        caso = 5; %Analiza tramas
end
```

De nuevo se hace uso de las sentencias “if” para identificar en qué caso estamos y así en un futuro “switch case” seguir almacenando tramas, descartarlas, o empezar a tratar las que serán analizadas. Se ha implementado además un caso en el que almacenar la información analizada en casos de silencios.

```
switch(caso)
    case 1
        if ~isempty(repes)
        repes_aux = repes;
        repes = [];
        end
        count = 1;
        aux = freq;
    case 2
        count = count + 1;
        repes = [freq count];
        if count == 10
        caso = 5;
        valor_f = valor_f(end-9+1:end);
        valor_max = valor_max(end-9+1:end);
        valor_f_1 = valor_f_1(end-9+1:end);
        valor_max_1 = valor_max_1(end-9+1:end);
        valor_f_2 = valor_f_2(end-9+1:end);
        valor_max_2 = valor_max_2(end-9+1:end);
        valor_rela1 = valor_rela1(end-9+1:end);
        valor_rela2 = valor_rela2(end-9+1:end);
        end
    case 3
        if count < 5
        count = count + 1;
        else
        count = count + 1;
        aux2 = freq;
        repes_final = [repes_final ; repes_aux];
        repes_aux = [];
        end

    case 4
        if ~isempty(repes_aux)
```

```
            count = repes_aux (:,2);
            end
            count = count + 1;
            aux = freq;
            repes_aux = [];
case 5
            analizando = analizando + 1;
end
```

Una vez que entremos en el caso cinco, se empiezan a almacenar las cincuenta tramas mencionadas anteriormente para que sean analizadas. Al final del bucle "for" encontramos una instrucción que terminará el bucle cuando la variable "analizando" llegue a cincuenta. Dicha sentencia es la siguiente:

```
 if analizando == 50
            break
  end
```

Si se representan las tramas que se han considerado para la identificación del ave, obtendremos lo siguiente [Imagen 4.2.6.1]

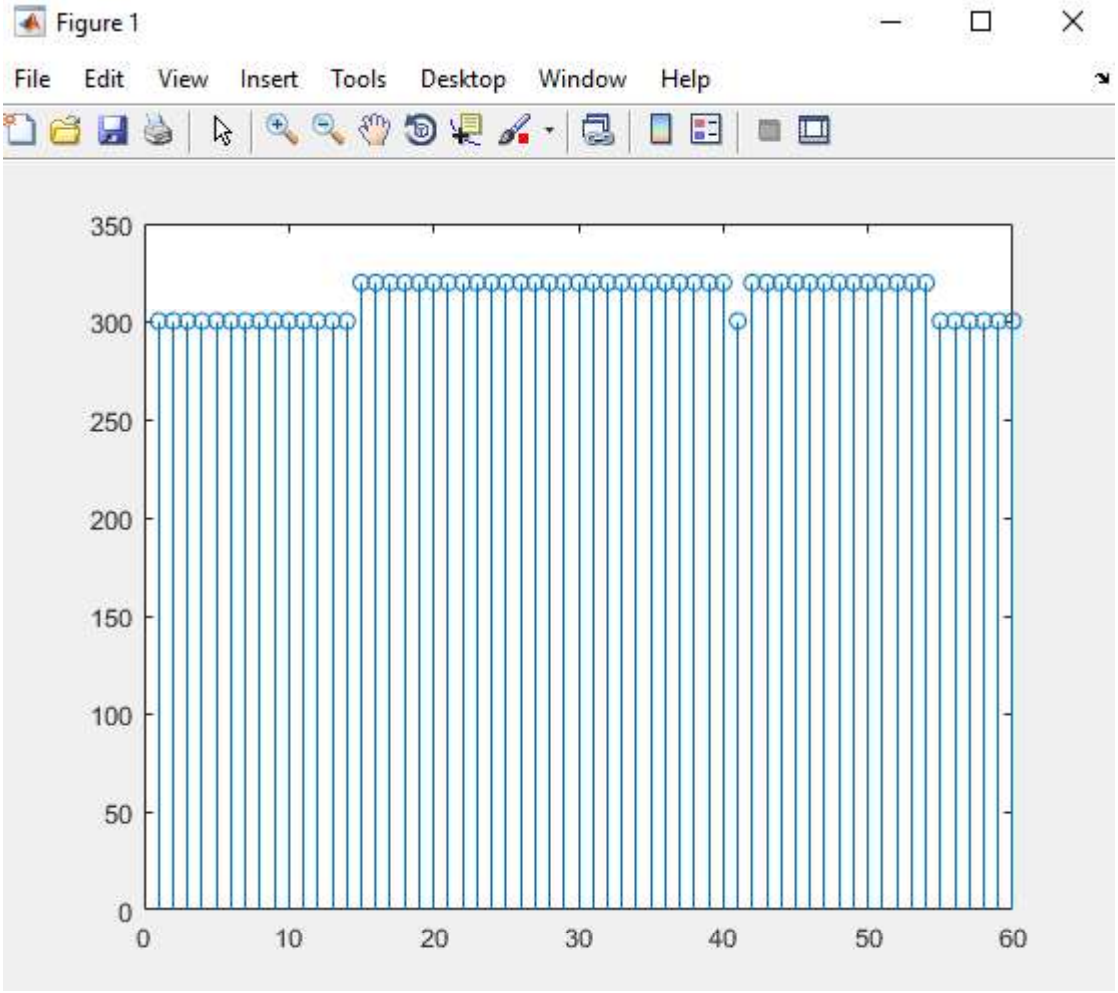


Imagen 4.2.6.1

Representación de las tramas analizadas

Fuente: Propia.

4.3. Implementación en Arduino.

A continuación, y al igual que en el punto anterior se va a presentar el código elaborado en Arduino, separado en distintas partes para poder explicar cada una de ellas de manera independiente. En primer lugar se revisará la librería utilizada para el cálculo de la "fft", así como la estructura general del sketch. A continuación se revisará cómo se trata el sonido, así como la separación en tramas de este para su análisis; de forma similar al apartado anterior se verá cómo se lleva a cabo la decisión del sistema de identificación y para terminar se presentará como se establece la comunicación con el servidor, y como se realiza el envío de datos al mismo.

4.3.1. Librerías y Estructura.

Nuestro sketch comienza con la inclusión de las librerías necesarias para su correcto funcionamiento, tal y como se haría en C/C++; en este caso las que se han implementado han sido:

```
#include <Wifi.h>
#include <SPI.h>
#include <SplitRadixRealP.h>
```

La primera de ellas, <Wifi.h> es la encargada de que el Shield Wifi funcione correctamente y es propia del IDE de Arduino. Lo mismo ocurre con la segunda librería, <SPI.h>, gracias a la cual se habilitan las conexiones a través de los pines GPIO, a través de los cuales se conectan el Wifi Shield a la placa de Arduino, y el micrófono a la Wifi Shield.

La última, <SplitRadixRealP.h>, de ahora en adelante "Radix", es una librería para Arduino Due, que permite la opción de realizar la "fft". Está será el eje principal para el tratamiento de las muestras de audio recogidas por el micrófono. Dentro de esta librería encontramos diferentes funciones, que nos permiten realizar diferentes operaciones con los datos que se obtienen de la "fft" y en este se comenta la tarea principal que ejecuta cada una, y cómo se utiliza para este proyecto.

Empezamos con las funciones de control de la librería:

- “radix.rev_bin”. Con esta función obtenemos las “posiciones” que ocupan las frecuencias en la placa Arduino. Con “posiciones” nos referimos a las posiciones de memoria que se le asocian a cada grupo de frecuencias cuando se realice la “fft”. Por ejemplo, si obtenemos una frecuencia fundamental en 1000 Hz, dicha frecuencia se corresponde con la posición 182.

- “radix.fft_split_radix_real” Con esta función se calcula la “fft” propiamente dicha. Como entrada recibe los parámetros en el dominio del tiempo, y devuelve los parámetros en el dominio de la frecuencia, ya ubicados en sus correspondientes posiciones.

Después de tener almacenados los valores en el dominio de la frecuencia podemos empezar a trabajar con ellos para poder extraer un indicador que nos permita identificar al ave en cuestión. Las funciones que nos van a permitir tratar con las muestras son:

- “get_Magnit1”. Con esta función encontraremos la frecuencia fundamental de la trama que se está analizando, con lo que obtendremos de vuelta la posición de memoria que ocupa. Gracias a esta posición seremos capaces de establecer el parámetro de identificación que nos permita distinguir a unas aves de otras.

- “radix.gain_Reset”. una vez que hemos obtenido la posición de la frecuencia fundamental, con la función de Reset lo que haremos será limpiar los valores almacenados para que se pueda analizar la siguiente trama, y de esta manera no sobrecargar la memoria de la placa Arduino, hasta que se produzca una identificación.

Una vez tenemos una idea de cómo trabaja la librería podemos estudiar la estructura general del sketch:

- En el setup(), lo primero que se ejecuta es la configuración del reloj, así como de la conversión analógico digital (necesaria para la correcta recogida de datos del micrófono). Una vez hecho esto se realiza la conexión con la red Wifi establecida en la declaración de variables, en el caso de tener una conexión exitosa se mostrará por consola la información relacionada con la conexión Wifi establecida.
- En el apartado del loop(), encontramos la ejecución de las funciones de la "fft" y las llamada a la función de detección de aves. En el caso de que se produzca una identificación exitosa, desde la función del filtro se envían directamente los datos al servidor.

4.3.2. Conexión Wifi.

Una vez que encendemos la placa Arduino, lo primero que se va a realizar es la conexión con la red Wifi de la que se hayan proporcionado los credenciales, para eso al comienzo del sketch, en la zona de declaración de variables estáticas, añadimos lo siguiente:

```
char ssid[] = "Nombre de la red";      //  SSID de la red
char pass[] = "Contraseña";            // Contraseña de la red
int status = WL_IDLE_STATUS;     // Estado de la señal de Wifi
```

Con los credenciales de acceso ya definidos, se ejecuta la función llamada "conectar" que buscará entre las redes disponibles aquella cuyos datos de acceso coincidan con los aportados. Dicha función contiene las siguientes instrucciones:

```
// attempt to connect using WPA2 encryption:
Serial.println("Attempting to connect to WPA network...");
Serial.println(ssid);
status = WiFi.begin(ssid, pass);

// if you're not connected, stop here:
if ( status != WL_CONNECTED) {
        Serial.println("Couldn't get a wifi connection");
        while (true);
}
// if you are connected, print out info about the connection:
else {
        Serial.println("Connected to network");
        // print your WiFi shield's IP address:
}
```

En el caso de que la conexión se realice con éxito como podemos ver en el código, se mostrará por consola un aviso. La siguiente función que encontramos en el sketch es aquella que muestra por consola los datos de la conexión que se ha establecido, entre los que podemos encontrar la dirección IP asignada a el shield de Wifi, así como su dirección MAC. La impresión por pantalla de dichos valores queda recogida en el siguiente código:

```
// print your WiFi shield's IP address:
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);
Serial.println(ip);

// print your MAC address:
byte mac[6];
WiFi.macAddress(mac);
Serial.print("MAC address: ");
Serial.print(mac[5], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.println(mac[0], HEX);
```

4.3.3. Conteo de Tramas.

Una vez que nuestra sketch se inicializa y empieza la recogida de datos, es decir, que se habiliten los pines GPIO y se establezca a cuál de ellos se ha conectado nuestro micrófono para la extracción de datos, comienza el relleno de las tramas. gracias a una variable que declaramos al comienzo del sketch, definimos el tamaño que van a tener las tramas, para que una vez que se rellene una de ellas se realicen los análisis necesarios antes de rellenar la siguiente. Para este escenario se ha declarado que el tamaño de las tramas sea de 2048 muestras, con la que tendremos resolución suficiente para un correcto análisis .

Cuando se han recogida las 2048 muestras correspondientes, empieza el tratamiento de las mismas, y después de pasar por la librería "Radix" obtendremos la posición en memoria en la que se encuentra nuestra frecuencia fundamental y esta posición se almacena en una variable y se espera a la siguiente posición para realizar la comparación. Llegados a este punto pueden suceder varias situaciones:

- Que ambas posiciones coincidan dentro del rango de reconocimiento de la misma ave, lo cual incrementa el contador en uno.
- Que las posiciones no coincidan dentro del rango de reconocimiento del mismo ave, lo cual decrementa el contador en uno.

Una vez que el contador llegue a cinco, (las tramas consideradas para comenzar con la identificación), se realiza el llamamiento a la función de toma de decisiones con las posiciones recibidas y que coinciden en el rango de identificación del mismo ave. Esto se puede ver reflejado en el siguiente extracto de código:

```
for ( uint32_t i = 0; i < dlina; i++) {  /
        if (array[i] > mayor) {
        mayor = array[i];
        pos = i;
        }
}
```

Se van almacenando las posiciones que devuelven las funciones de "Radix", en la variable "pos".

```
if (l == 0) {
        auxi[l] = pos;
        buscar (pos);
        l++;
}
```

Este es el caso inicial, en el que no se ha contado ninguna trama (por lo que l = 0), se almacena en el vector “auxi” la posición recibida, y se incrementa en uno el valor de “l” (tramas contadas).

```
if (l > 0 && pos > 50) {
        if (pos > valor_inf && pos < valor_sup) {
                auxi[l] = pos;
                l++;
        } else  {
                l--;
        }
}
```

Aquí es donde dependiendo de si la trama analizada coincide con la anterior en el mismo rango de detección de un ave se incrementa o decrementa el valor de “l” dependiendo de la condición que se cumpla.

```
if (l == 5) {
        l = 0;
        Serial.println(pajaro);
        subir(pajaro);
        mostrar();
        delay(10000);
}
```

Cuando el número de tramas que coincidan sea igual a cinco, se inicializa la cuenta de tramas a cero, se muestra el ave que se ha identificado y se procede a la subida del resultado al servidor.

4.3.4. Identificación del Ave.

En esta parte, y de forma similar a como se ha realizado la toma de decisiones en Matlab, dependiendo de las posiciones que se vayan recibiendo de las funciones de "Radix", y de si estas son consecutivas para que puedan considerarse las tramas necesarias para la toma de decisiones, encontramos un sistema condicional que se centra en la comparación de las posiciones recibidas, con las establecidas en el filtro.

Antes de mostrar el código del sistema, se van a presentar los rangos de decisión correspondientes a cada ave, acompañados de los valores que han sido trasladados desde Matlab. Estos valores están expresados de la siguiente manera en el sketch:

```
// Aves en orden
char* aves[] = {"Paloma", "Gaviota", "Cuervo", "Gorrión", "Estornino"};
// Matriz de frecuencias
double matriz [5][2] = {{300,600}, {700,1200}, {1200,2100}, {2800,4000}, {4000,7000}};
// Matriz de Armónicos 1
double matriz2 [5][2] = {{1, 65},{2, 8},{8, 110},{12, 220},{220, 999}};
// Matriz de Armónicos 2
double matriz3 [5][2] = {{1, 93},{4, 12},{12, 155},{44, 317},{317, 999}};
// Matriz de posiciones
double matriz4 [5][2] = {{50, 120}, {120, 260}, {260, 380}, {450, 680}, {700, 1270}};
```

Si se observan los rangos de las tres primeras matrices, estos coinciden con los valores que se obtuvieron de la elaboración del filtro en Matlab, como se ha explicado anteriormente. Y la última matriz, es aquella que se ha construido a través de la realización de pruebas con dichos parámetros, para almacenar las posiciones que corresponden a cada ave. Con esto presentado, el código correspondiente al filtro de decisión es el siguiente:

```
if (numero > 120) {                    // Donde número es la posición obtenida de la “fft”
        Serial.println("Mayor que 120");
        if (numero > 400) {
                Serial.println("Mayor que 400");
                if (numero > 450 && numero < 680 ) {
                        fila = 3;
                        valor_inf = matriz4[fila][0];
                        valor_sup = matriz4[fila][1];
                        pajaro = aves[fila];
                        }
                else if(numero > 700 && numero < 1270){
                        fila = 4;
                        valor_inf = matriz4[fila][0];
                        valor_sup = matriz4[fila][1];
                        pajaro = aves[fila];
                        }
                }
        else {
                if (numero > 120 && numero < 260 ) {
                        fila = 1;
                        valor_inf = matriz4[fila][0];
                        valor_sup = matriz4[fila][1];
                        pajaro = aves[fila];
                        }
                else if (numero > 260 && numero < 380){
                        fila = 2;
                        valor_inf = matriz4[fila][0];
                        valor_sup = matriz4[fila][1];
                        pajaro = aves[fila];
                }
        }
}
else {
        Serial.println("El número es menor que 120");
```

```
        if (numero > 50 && numero < 120) {
                fila = 0;
                valor_inf = matriz4[fila][0];
                valor_sup = matriz4[fila][1];
                pajaro = aves[fila];
                }
        }
}
```

El procedimiento es muy sencillo, se compara si el valor de posición obtenido se corresponde con algunos de los rangos de posiciones a cada ave, y en el caso de que así sea se almacenan la posición que se está comparando junto con los límites superior e inferior del rango del posible ave a identificar, para después poder realizar el conteo de tramas que coincidan dentro de ese rango hasta cinco, como se ha visto anteriormente.

4.3.5. Comunicación con el Servidor.

Cuando se ha identificado a un ave, el último paso que se realiza en el sketch antes de volver al análisis de datos es enviar al servidor que ave se ha identificado. Para ello al comienzo del sketch se ha indicado la dirección del servidor web a la que nos queremos conectar, la cual se expresa de la siguiente manera:

```
//Datos para wifi shield Base de Datos
char server [] = "iobirds.000webhostapp.com/"; // Dirección IP del servidor
WiFiClient client;
```

Una vez que se le ha proporcionado la dirección a la placa, cuando se produzca la identificación, se realiza una petición HTTP en la que se solicita la conexión con el servidor para poder enviar el ave identificada. Esta petición es de tipo GET, ya que con una sola instrucción podemos enviar al servidor toda la información necesaria para que el envío se realice correctamente. La parte de envío de datos la podemos encontrar a continuación:

```
void subir(char* elemento) {
        char cadena[50];
        sprintf(cadena, "GET /iot.php?valor=%s HTTP/1.1", elemento);
        Serial.print(cadena);
        Serial.println("\nStarting connection to server...");
        // if you get a connection, report back via serial:
        if (client.connect(server, 80)) {
                Serial.println("connected to server");
                // Make a HTTP request:
                client.println(cadena);
                client.println("Host: iobirds.000webhostapp.com/");
                client.println("Connection: close");
                client.println();
        }
}
```

4.4. Servicio Web y Base de Datos.

La última parte de este proyecto consiste en la elaboración de un servicio web que reciba información de la placa Arduino cuando se identifique a un ave. Como se ha mencionado en puntos anteriores, este servicio web ha de contar con una base de datos que recoja la información que vaya siendo enviada. Además se han añadido funciones de búsqueda para facilitar la visualización de los datos almacenados.

4.4.1. Estructura Base de Datos.

Se ha diseñado la base de datos para que recoja la siguiente información:

- Un Identificador para cada entrada que se generará de manera automática cada vez que una entrada sea añadida.
- El nombre de la especie identificada, que se almacena bajo el nombre de "valor". Este "valor" es de tipo texto, ya que desde la placa se envía una cadena de caracteres que informan sobre el individuo reconocido.
- La fecha en la que se produjo la identificación. Esta tiene el formato indicado en la norma ISO 8601, que especifica "AAAA-MM-DD" [36].
- La hora a la que se produjo la identificación. De la misma manera, el formato de la fecha sigue la norma ISO 8601, la cual indica "hh:mm:ss" [37].

Por tanto tenemos una base de datos que contiene una tabla con 4 columnas, una para cada uno de los valores citados. Una vez visualizada en la web con algunos datos tiene la siguiente apariencia [Imagen 4.4.1.1]

Registro

ID	Especie	Hora	Fecha
1090	Paloma	12:22:43	2018-08-29
1089	Cuervo	12:21:06	2018-08-29
1088	Gorrion	12:19:03	2018-08-29
1087	Gaviota	12:17:47	2018-08-29

Imagen 4.4.1.1

Datos almacenados

Fuente: Propia.

4.4.2. Estructura de la Web.

Una vez presentada la base de datos, se construye la web desde la que acceder a la información. La página principal “index.php” va a ser desde la que se acceda a la base de datos para consultar la información almacenada en ella de diferentes formas. La primera será acceder a consultar todo el contenido donde obtendremos una tabla como la representada en la Imagen 4.4.1.1 con todo el histórico de identificaciones.

La segunda opción será acceder a un formulario desde el cual se podrá acceder a la información almacenada de manera selectiva. En este formulario podremos seleccionar si queremos buscar una especie en concreto a través de su nombre, o si queremos consultar las identificaciones que se produjeron en una fecha concreta o como última opción establecer un rango horario en el que buscar cuantos reconocimientos se produjeron para el mismo.

Este formulario se ha escrito en HTML, y su código es el siguiente:

```
<html>
  <head>
      <meta charset="utf-8">
      <title>Buscador</title>
  </head>
  <body>
      <p>Buscador por fecha</p>
      <form action="buscador1.php" method="post">
          <input type="date" name="fecha" placeholder="Fecha a buscar" >
          <input type="submit" name="name" value="Buscar">
      </form>
      <p>Buscador por nombre</p>
      <form action="buscador1.php" method="post">
          <input type="text" name="nombre" placeholder="Nombre de la especie" >
          <input type="submit" name="name" value="Buscar">
      </form>
      <p>Buscador por franja horaria</p>
      <form action="buscador1.php" method="post">
          <input type="time" name="hora1" placeholder="Hora inicial" >
          <input type="time" name="hora2" placeholder="Hora final" >
          <input type="submit" name="name" value="Buscar">
      </form>
  </body>
</html>
```

4.4.3. Funcionalidades.

En este apartado se presentarán los códigos que permiten el funcionamiento del servicio web, entre los cuales encontraremos la conexión con la base de datos, la extracción de información de la misma, y la presentación de estos datos.

Comenzando con el fichero “conexion.php”, como su nombre indica es el que se utiliza para conectarse con la base de datos y en él encontramos la dirección del servidor donde está alojada, el usuario y la contraseña para acceder a la base de datos, y por último el nombre de la misma. En caso de error se añade un comando echo para informar de que no se ha podido realizar la conexión:

```
<?php

$mysqli = new  mysqli ( "mysql.hostinger.es" , "u984995782_ardui" , "arduino" , "u984995782_insec" ) ;

   if(mysqli_connect_errno()){
         echo 'Conexion Fallida : ', mysqli_connect_error();
         exit();
   }
?>
```

Por otra parte cuando realizamos una búsqueda a través del formulario presentado en el punto anterior, una vez se haya asociado uno de los parámetros de búsqueda a la entrada del usuario esta se introduce en una sentencia “Select * From” para escoger las entradas que cumplan la condición. si consultamos el código relacionado con esta parte tenemos:

```
<?php
require('conexion.php');
$hora1 = $_POST['hora1'];
$hora2 = $_POST['hora2'];
$nombre = $_POST['nombre'];
$codigo = $_POST['fecha'];

        $query="SELECT *  FROM especie WHERE fecha = '$codigo' or valor like '$nombre' or (tiempo > '$hora1' and tiempo < '$hora2')";

        $resultado=$mysqli->query($query);
?>
```

Una vez que los datos han sido seleccionados, bien por el formulario de búsqueda, o por que el usuario solicite mostrar todo el contenido de la tabla, para mostrarlos utilizaremos una sentencia while, hasta mostrar todas las entradas. El código de esta función es el siguiente:

```
<?php while($row=$resultado->fetch_assoc()){ ?>
        <tr>
                <td><?php echo $row['ID'];?> </td>
                <td> <?php echo $row['valor'];?> </td>
                <td> <?php echo $row['tiempo'];?> </td>
                <td> <?php echo $row['fecha'];?> </td>
        </tr>
  <?php } ?>
```

5. Resultados.

En este punto se hará un análisis de los diferentes parámetros que han sido modificados a lo largo de diferentes pruebas, así como la justificación de los valores finales elegidos para los mismos, en base a los resultados obtenidos tras la modificación de los mismos. Esto se aplica a las dos partes de práctica del trabajo, es decir, para el escenario de Matlab, y para el escenario en Arduino.

Una vez terminada la parte teórica de los parámetros se mostrarán las distintas pruebas que se han realizado en cada escenario para los distintos casos que se han considerado. A cada prueba le acompañará un breve comentario sobre los resultados obtenidos y la valoración de estos.

5.1. Análisis de los Parámetros.

En los siguientes puntos se presentan los parámetros configurables del sistema de identificación y de tratamiento de los datos. Estos parámetros son el porcentaje de potencia con el que se discriminan tramas, la relación señal ruido, el solapamiento de las tramas que se analizan, y por último el número de tramas consideradas para la toma de decisiones. Los tres primeros parámetros son exclusivos de Matlab, aunque su configuración y estudio ha intervenido en la comprensión de cómo podría comportarse el sistema final (Arduino), en distintas situaciones, mientras que el último parámetro está presente tanto en la parte de Matlab, como en la de Arduino.

5.1.1. Porcentaje de Potencia.

Este es el primer parámetro que encontramos para poder configurar en la parte de testeo de Matlab. Con él, se puede configurar las tramas que van a ser consideradas en el filtro de extracción de parámetros, y en el sistema de identificación.

Para ello se comparan las potencias de la trama que se desea analizar, con la potencia total del fichero de audio, de esta manera las tramas que no contenga información, tramas de silencio, o aquellas cuya potencia sea reducida, bien porque no contengan información relevante, o porque tengan ruido, no serán analizadas por el sistema.

Para las pruebas que se han realizado los valores obtenidos de este porcentaje han oscilado entre el 5% para las pruebas realizadas sin ruido, para eliminar las tramas de

silencio, y un 15% para los escenarios con ruido, que alteraban las tramas con potencias más bajas, haciéndolas pasar por el filtro cuando no debían.

5.1.2. Relación Señal Ruido.

La relación señal / ruido (SNR abreviado) es una medida utilizada en ciencia e ingeniería que compara el nivel de una señal deseada con el nivel de ruido de fondo. La SNR se define como la relación entre la potencia de la señal y la potencia del ruido, a menudo expresada en decibelios [38].

En las pruebas que se han realizado dos escenarios, el primero de ellos la relación señal ruido con la que se ha trabajado era de 10dB, mientras que en el segundo de ellos se han realizado pruebas con una SNR de 7dB.

5.1.3. Solapamiento.

Modificando el solapamiento que se produce entre las tramas se buscaba mejorar la resolución de la información obtenida cuando se realizaba el análisis de estas, a más solapamiento más pequeños era los cambios entre trama y trama lo cual permite una mejor visualización de la evolución de los parámetros que se extraían de las tramas.

Para las pruebas que se han realizado el valor de solapamiento se estableció en primer lugar en un 90%, para conseguir la mejor resolución posible sin sobrecargar mucho el coste computacional, y a medida que se reducía, la dispersión entre los valores variaba dependiendo de la especia que se analizaba, por lo que finalmente se dejó con su valor inicial.

5.1.4. Número de Tramas Consideradas.

Este parámetro ha tenido resultados distintos en los escenarios en los que se ha tratado, así que lo analizaremos desde cada uno de ellos:

- En Matlab, dado que la duración de las tramas era de cincuenta milisegundos, dependiendo de la frecuencia de muestreo del fichero de entrada, podía haber más o menos muestras por trama. Por eso se optó a que todas los casos, independiente de la frecuencia de muestreo, todos los ficheros tuvieran las mismas muestras, una 220. Teniendo tan pocas muestras por trama se necesitan analizar más tramas para poder obtener un resultado fiable, y tras varias pruebas se estableció que se analizaran cincuenta tramas.

- Por otro lado, en Arduino las tramas tienen 2048 muestras, diez veces más que en Matlab, por lo que en primer lugar se pensó en analizar diez veces menos tramas, es decir analizar 5 tramas para tomar la decisión. Tras varias pruebas modificando dicho valor, se vio que más tramas ralentizan mucho el tiempo de identificación, ya que hasta que coincidan más de cinco tramas con tantas muestras el sistema de conteo incrementa y decrementa demasiadas veces.

5.2. Pruebas Realizadas en Matlab.

5.2.1. Ficheros del Filtro.

En primer lugar, para comprobar que el filtro se había configurado de manera correcta se le evaluó con los mismos ficheros de los que se habían extraído los parámetros para su configuración. Con estas pruebas se decidieron los valores finales para los parámetros configurables, ya que correctamente configurado el sistema de identificación, para los mismo archivos de entrada siempre tendría que devolver resultados del 100% de aciertos.

A continuación se incluye una tabla con los resultados obtenidos de configurar estos parámetros hasta obtener el 100% de aciertos. [Tablas 5.2.1.1]

	Porcentaje Potencia			Solapamiento			Número de tramas		
Valor utilizado	5%	15%	30%	90 %	75%	50%	10	25	50
Porcentaje de Aciertos para 50 ficheros	100	100	92	100	94	88	86	90	100

Tabla 5.2.1.1

Resultados pruebas de los parámetros en ficheros del filtro

Fuente: Propia

Para estos ficheros se realizaron también pruebas modificando la cantidad de ruido que se añadía, y se obtuvieron los siguientes resultados. [Tabla 5.2.1.2]

Ruido Añadido	10 %	20 %
Porcentaje de Aciertos para 50 ficheros	100	90

Tabla 5.2.1.2

Resultados pruebas con Ruido en ficheros del filtro.

Fuente: Propia

5.2.2. Ficheros ajenos al filtro.

Una vez configurados los parámetros con los que se obtuvieron los resultados del 100% de aciertos para los ficheros con los que se fabricó el filtro, este se puso a prueba con ficheros ajenos al mismo. En primer lugar se realizaron las pruebas de identificación sin ruido, y con la modificación del porcentaje de potencia, que había arrojado dos configuraciones con aciertos del 100%.

Se obtuvieron los siguientes resultados [Tabla 5.2.2.1]

	Porcentaje de Potencia	
Valor utilizado	5 %	15 %
Porcentaje de Aciertos para 95 ficheros	90	82

Tabla 5.2.2.1

Resultados configuración potencia con todos los ficheros.

Fuente: Propia

Como era de esperar, al incluir nuevos ficheros el filtro no sería capaz de identificarlos todos, pero teniendo en cuenta que el filtro solo se elaboró con diez ficheros, una vez que doblas el número de ficheros a analizar solo se haya visto afectado un 10% del porcentaje de acierto, indica que un filtro como este sería reproducible en un escenario ideal.

Analizando un poco más en detalle los resultados obtenidos para cada ave, se puede observar que las tasas de errores más altas se corresponden con la de las aves cuyas frecuencias fundamentales están superpuestas, siendo la Gaviota con una frecuencia fundamental media de 1200 Hz, y el Cuervo que es de 1400 HZ. [Tabla 5.2.2.2]

Ave	Aciertos	Fallos	Porcentaje
Paloma	19	0	100 %
Cuervo	16	3	84 %
Gaviota	15	4	79 %
Estornino	18	1	95 %
Gorrión	18	1	95 %

Tabla 5.2.2.2

Porcentaje de aciertos para cada Ave.

Fuente: Propia.

Por último, cabe mencionar que los ficheros en los que el sistema no ha sido capaz de identificar al ave correctamente, ha fallado a favor del ave que estaba en ese rango de frecuencias, es decir, los fallos de las gaviotas fueron a favor de los cuervos y viceversa, del mismo modo paso para los estorninos y los gorriones.

5.2.3. Pruebas con Ruido.

Dado que los resultados obtenidos en la prueba anterior han seguido teniendo un alto índice de acierto, el siguiente paso es el de empeorar la calidad de los ficheros utilizados añadiendo ruido a los mismos. A través de Matlab se añade ruido blanco gaussiano a los ficheros cada vez que se realiza un análisis, se obtienen resultados diferentes aunque estos estén próximos, por esta razón para cada fichero de audio se han realizado cinco pruebas, de las cuales se ha hecho una estimación de lo probable que es obtener un resultado positivo.

Dado que en este trabajo se ha trabajado con noventa y cinco audios, a cinco pruebas por cada uno, eso supone un total de cuatrocientas setenta y cinco pruebas, así que a fin de no sobrecargar el contenido de la memoria, a continuación se mostrarán solo los resultados finales obtenidos, al igual que en los apartados anteriores, y al final del documento se añade un anexo de pruebas donde se podrán consultar los resultados de cada una de las pruebas de manera más detallada

5.2.3.1. SNR 10dB.

En primer lugar se añade un 10 % de ruido a los ficheros con los que se ha realizado la prueba anterior, con el objetivo de ver la respuesta del filtro a esta nueva situación. Los resultados obtenidos quedan representados en la siguiente tabla [Tabla 5.2.3.1.1]

Ave	Aciertos	Fallos	Porcentaje de Aciertos
Palomas	19	0	100 %
Cuervo	14	5	73 %
Gaviota	13	6	68 %
Estornino	17	2	89 %
Gorrión	18	1	95 %

Tabla 5.2.3.1.1

Tabla de aciertos para ficheros con una SNR de 10 dB.

Fuente: Propia

Como era de esperar la adición de ruido a los ficheros de audio empeora el porcentaje de aciertos en un 15% comparados con los ficheros del filtro, lo que nos deja con una eficacia del 84%.

5.2.3.2. SNR 7 dB.

En esta ocasión se añade un 10 % más de ruido, por lo que el sistema de identificación trabajará con ficheros que tienen un 20 % de ruido añadido. Los resultados deberían de seguir empeorando en comparación con los experimentos realizados anteriormente. En la siguiente tabla se muestran los resultados de este escenario. [Tabla 5.2.3.2.1]

Ave	Aciertos	Fallos	Porcentaje de Aciertos
Palomas	13	6	68 %
Cuervo	13	6	68 %
Gaviota	7	12	37 %
Estornino	17	2	90 %
Gorrión	0	19	0 %

Tabla 5.2.3.2.1

Tabla de aciertos para ficheros con una SNR de 7 dB.

Fuente: Propia

Tras la adición de más ruido a los ficheros de audio, como se había supuesto en un primer momento. Los ficheros más perjudicados por la distorsión son los de las Gaviotas y evidentemente los de los Gorriones, para el primer caso a diferencia de los casos anteriores no todos los ficheros que han dado fallo han sido a favor de los Cuervos, ya que algunos de los fallos han sido Estorninos. Para el caso del gorriones ninguno de los ficheros ha sido identificado, ya que el factor que los diferencia de los estorninos es a partes iguales entre la frecuencia fundamental y los armónicos, y la adición de tanto ruido hace que la información de los armónicos pase a no ser redundante, fallando en todas los intentos.

Por primera vez en todos los experimentos se han encontrado fallos en la identificación de las Palomas, aunque estas se encuentren en su propia zona de frecuencia en el filtro. Teniendo esto en cuenta, el realizar más pruebas, añadiendo más ruido se han considerado innecesarias, ya que los resultados no aportarán resultados relevantes para este proyecto.

5.3. Pruebas Realizadas en Arduino.

En este apartado al igual que en el anterior se presentan los resultados obtenidos tras realizar distintas pruebas con la placa de Arduino. Como se mencionó al principio de este capítulo, en este escenario solo contamos con la modificación de un parámetro, lo cual no ofrece tanta posibilidad de mejorar el sistema como en el caso de Matlab que teníamos 3, además de la posibilidad de añadir ruido a los archivos.

5.3.1. Número de Tramas Consideradas

Para nuestro escenario en Arduino, el único parámetro con el que podemos trabajar es el número de tramas que se necesitan para realizar la toma de decisiones. Para ello se realizaron tres pruebas, una en la que se utilizaban menos tramas (3 tramas) de las calculadas a partir del número de muestras, una con el número de tramas calculadas (5 tramas) en base al número de muestras y el último caso que es con más tramas (7 tramas) que las calculadas.

Para el primer escenario se obtuvieron los siguientes resultados [Tabla 5.3.1.1]

Ave	Aciertos	Fallos	Porcentajes
Paloma	19	0	100 %
Cuervo	7	12	37 %
Gaviota	10	9	52 %
Estornino	13	6	68 %
Gorrión	12	7	63 %
Total	61	34	62 %

Tabla 5.3.1.1

Porcentaje de aciertos Arduino 3 tramas.

Fuente: Propia.

Con este experimento solo se llega a conseguir un 64 % de probabilidad de acierto, la cual es considerablemente menor que el 90% que se obtuvo en Matlab.

Para el segundo escenario se consideraron 5 tramas, y los resultados obtenidos fueron los siguientes [Tabla 5.3.1.2]

Ave	Aciertos	Fallos	Porcentaje
Paloma	19	0	100 %
Cuervo	11	8	58 %
Gaviota	18	1	95 %
Estornino	13	6	68 %
Gorrión	16	3	84 %
Total	77	18	81 %

Tabla 5.3.1.2

Porcentaje de aciertos Arduino 5 tramas.

Fuente: Propia.

En este caso se ha conseguido un 81% de aciertos, que es más que el caso anterior, lo cual indica que los cálculos realizados apuntaban hacia la solución más adecuada. De todas maneras el porcentaje de aciertos que se consigue sigue sin acercarse al 90% que se obtenía en Matlab, y teniendo en cuenta que estamos trabajando con 95 archivos, es una diferencia porcentual considerable.

Por otro lado, teniendo en cuenta las limitaciones del hardware utilizado no podríamos afirmar que este proyecto no es realizable en un dispositivo con estas características o mejores, aunque antes de dar por finalizado el desarrollo de las pruebas todavía queda la prueba con más tramas de las calculadas.

En primer lugar, un incremento de las tramas necesarias para tomar una decisión debería traducirse en identificaciones más veraces, pero desafortunadamente no es el caso, como se puede observar en la siguiente tabla. [Tabla 5.3.1.3]

Ave	Aciertos	Fallos	Porcentaje
Paloma	14	5	74 %
Cuervo	5	14	26 %
Gaviota	11	8	58 %
Estornino	7	12	37 %
Gorrión	9	10	47 %
Total	46	49	48 %

Tabla 5.3.1.3

Porcentaje de aciertos Arduino 7 tramas.

Fuente: Propia.

Incrementando el número de tramas necesaria para tomar una decisión el sistema en la mayoría de las ocasiones no es capaz de tomar una decisión, ya que el canto de las aves no tiene una duración tan extensa como para cubrir espacios de tiempo tan amplios, lo cual ha supuesto la caída del porcentaje de aciertos hasta el 48 %.

Observando la placa Arduino mientras analizaba las tramas, se aprecia como para la mayoría de las aves, durante los periodos de canto, va almacenando las posiciones correctamente e incrementando el contador de detección, pero en el momento en el que el canto cesa, el contador comienza a decrecer ya que para no sobrecargar la memoria del Arduino, se van reciclando los datos que se reciben.

6. Métodos de Ahuyentamiento para las aves.

En este punto se realizará un estudio teórico de los distintos métodos que se utilizan para espantar a las aves en los métodos más tradicionales, y ver si es posible la implementación de alguno de estos métodos en nuestra placa Arduino.

Entre todos los posibles métodos que se puedan instalar, se realizará una comparativa en la que se considerarán aspectos como la eficacia del método, la sencillez de la implementación, el coste del equipo, para poder elegir aquel que cumpla mejor el objetivo, dentro de las posibilidades del proyecto.

6.1. Estudio de los métodos instalables en Arduino.

Como se ha mencionado en el apartado 2.4.3 entre los distintos métodos que existen para la gestión de aves, para este proyecto vamos a investigar aquellas relacionadas con el ahuyentamiento de las especies.

Tras un análisis de las distintas maneras en las que hoy día se lidia con la situación de las plagas de aves, varias empresas dedicadas al control de estas situaciones nos explican que, a parte de los métodos previamente citados, las maneras más eficaces para ahuyentar a las aves se reducen a dos, que son:

- Utilización de repelentes visuales.
- Utilización de repelentes sonoros. [40]

Ambos métodos tienen sus ventajas e inconvenientes, y se van a presentar a continuación.

Comenzando con los métodos que utilizan repelentes visuales, encontramos varias maneras de conseguir que las aves no se acerquen a los lugares donde suelen instalar sus bandadas, entre los que encontramos:

- La generación de una luz de alta intensidad, que al detectar a un ave consiga asustar al animal a través de un fogonazo de luz. Este método, aunque a priori pueda parecer el más efectivo, trae consigo una serie de desventajas, como son la molestia que pueda producir a otros individuos que vivan en las zonas en las que se instale, como pueden ser otros animales, o incluso las personas en el caso de la instalación en un entorno urbano. Aunque su implementación sería sencilla en el sistema Arduino, no se optará por este método.

- La instalación de un espantapájaros. A diferencia de la imagen que se tiene de un espantapájaros habitual, para el caso de estas aves, se han desarrollado réplicas de las aves que son las depredadoras de las especies más pequeñas, que además caen dentro del colectivo de plaga. Esta réplica pretende simular la presencia de un ave rapaz, un búho, halcón, cetrero, que impida que las otras aves se acerquen al lugar donde está instalado. El problema de este tipo de sistema, es que aunque pueda tener resultados positivos, las otras aves se acaban acostumbrando a la presencia de la réplica y acaban poblando las mismas zonas, lo cual a largo plazo no es una solución muy efectiva.

Por otro lado tenemos las técnicas de generación de repelentes sonoros. Estos consisten en generar un sonido con el que se consiga espantar a los individuos que se acerquen a una zona controlada. En este caso se han realizado ya pruebas para ver cuáles eran los sonidos que más efecto tenían sobre las aves, donde los experimentos incluían la generación de tonos a altas frecuencias que consiguieran molestar a los individuos para que abandonaran el lugar, pero al igual que en el caso de la generación de luces de alta intesidad, estos tonos interferían con la comodidad de otras especies, así como de las personas.

Otro de las pruebas que se realizó fue, siguiendo el camino de la utilización de un espantapájaros, la reproducción del sonido que producían las aves depredadoras de estas especies. Se consiguieron resultados tan positivos que se comercializaron sistemas independientes que se encargaban de realizar esta tarea. [Imagen 6.1.1]

Imagen 6.1.1

Ahuyentador de aves acústico.

Fuente: https://diotronic.com/ahuyentador-de-aves-acustico_27684/

Dado que con este método se consigue espantar a las aves de manera efectiva, y no interfiere en la rutina y comodidad de otras especies, ni de las personas, se utilizará la placa Arduino para la generación de un reclamo de ave rapaz con el que espantar a las aves que se detecten a través del mismo.

6.2. Reclamo de rapaz.

Como se ha indicado en el punto anterior, para intentar poner solución al problema de la gestión de las plagas de aves, a través de nuestro sistema arduino se reproducirá un archivo de audio previamente seleccionado que simule la presencia de un ave depredadora de las consideradas para este proyecto.

Para llevar a cabo este paso necesitaremos de un componente hardware adicional que es un altavoz con el que reproducir el sonido que hayamos seleccionado. Al igual que nuestro micrófono conectaremos el altavoz a uno de los pines libres que tenemos en nuestra placa.

Para poder reproducir el fichero que queramos necesitaremos utilizar una librería adicional, esta librería tiene el nombre "TMRpcm" donde encontramos la función "play". Esta función necesita como parámetros de entrada el nombre del archivo que se quiere reproducir si este está almacenado en una SD conectada al arduino. De forma previa a la reproducción del sonido, en el sketch se deberá de incluir el pin al que se ha conectado el altavoz a través de la variable:

```
tmrpcm.speakerPin = #; //Donde # es el número del pin.
```

El siguiente paso será, en la estructura del sketch(), incluir un control de error para la inicialización de la tarjeta SD, que contenga el fichero de audio a reproducir. Este control tendrá el siguiente aspecto:

```
if (¡SD.begin(pinSD)){
        Serial.println("Fallo en la tarjeta SD");
        return
}
```

Con esta configuración inicial realizada, como se ha indicado, se reproducirá el reclamo de la rapaz para espantar así a las aves que se detecten.

7. Conclusión y líneas futuras.

A lo largo del desarrollo de este documento se han ido explicando los distintos procedimientos que se han seguido para establecer un sistema de gestión de aves, orientado a las plagas de aves, a la vez que se han ido presentando los resultados asociados a las pruebas realizadas para cada escenario y en base a dichos resultados se demuestra que es posible desarrollar un software capaz de detectar distintas especies.

Los resultados obtenidos en el sistema Arduino, aunque no llegan a satisfacer un porcentaje lo suficientemente fiable como para dar por finalizado el desarrollo de esta tecnología, sí que han ofrecido un rendimiento más que positivo teniendo en cuenta las limitaciones del equipo. Estas limitaciones pueden ser de diferente índole como son la baja potencia del procesador de la placa, el uso de un micrófono que no es de membrana, o la adición de ruido adicional a través de las conexiones de los distintos componentes hardware. Con la mejora de dichas partes que entorpecen la eficacia del sistema, podrían alcanzarse resultados a la altura de los obtenidos en el entorno de simulación con Matlab.

Como futura línea de trabajo, este sistema podría intentar implementarse en otros equipos SBC como podrían ser una Raspberry Pi, para realizar un estudio comparativo entre los distintos sistemas y ver cuál de los dos arroja mejores resultados. Otras de las líneas de trabajo podría ser el desarrollo de una librería original, con la que ser capaz de tratar el audio recogido del micrófono de manera más enfocada al entorno del trabajo del proyecto, en lugar de utilizar una librería general. Si estas mejores en un futuro entorno del desarrollo mejoran los resultados obtenidos en este proyecto, se podrían incluir más aves en la biblioteca de sonidos para ir perfeccionando el sistema de detección, haciéndolo capaz de identificar también a especies que no son consideradas plagas, con vistas a una posible producción del sistema para instalarlo en zonas urbanas y realizar pruebas reales de gestión de las aves.

8. Referencias.

8.1. Referencias Bibliográficas.

[1] https://en.wikipedia.org/wiki/Single-board_computer
[2] https://en.wikipedia.org/wiki/Radio-Electronics
[3] https://en.wikipedia.org/wiki/EPROM
[4] https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento
[5] https://en.wikipedia.org/wiki/Computer_data_storage
[6] https://en.wikipedia.org/wiki/Input/output
[7] https://en.wikipedia.org/wiki/Microprocessor
[8] https://www.raspberrypi.org/about/
[9] https://www.raspberrypi.org/products/
[10] https://www.raspberrypi.org/products/raspberry-pi-1-model-b-plus/
[11] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[12] https://www.raspberrypi.org/products/raspberry-pi-zero/
[13] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[14] https://en.wikipedia.org/wiki/Arduino
[15]https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-on-various-platforms-4b3e4a
[16] https://www.arduino.cc/en/Main/ArduinoShields
[17] https://store.arduino.cc/arduino-due
[18] https://www.arduino.cc/en/Main/FAQ#toc11
[19] https://www.arduino.cc/reference/en/language/structure/sketch/setup/
[20] https://www.arduino.cc/reference/en/language/structure/sketch/loop/
[21] https://www.arduino.cc/reference/en/
[22] https://es.wikipedia.org/wiki/PHP
[23] https://es.wikipedia.org/wiki/HTML
[24] https://es.wikipedia.org/wiki/SQL
[25] https://es.wikipedia.org/wiki/Plaga
[26] https://sergal.es/gestion-y-control-de-plagas/aves/
[27] https://www.anticimex.com/es-ES/empresa-control-de-plagas/control-de-aves/
[28]http://gamma.catie.ac.cr/pma/es/publicaciones/manual_de_identificacion_aves_silvestres.pdf
[29] https://es.wikipedia.org/wiki/Siringe
[30]https://store.arduino.cc/arduino-wifi-shield
[31]https://www.digikey.es
[32]https://en.wikipedia.org/wiki/Fourier_transform
[33] https://www.xeno-canto.org/
[34] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[35] https://tools.ietf.org/html/rfc2616

[36] https://www.iso.org/iso-8601-date-and-time-format.html
[37] https://en.wikipedia.org/wiki/ISO_8601
[38] https://en.wikipedia.org/wiki/Signal-to-noise_ratio
[39] http://www.anecpla.com/documentos/GDA.pdf
[40] https://www.comercturro.com/blog/mantenimientos/como-evitar-los-pajaros.html

8.2. Referencias de Imágenes.

8.3. Referencias de Tablas.

– Página en blanco –

Anexo I - Pruebas Realizadas en Matlab con Ruido.

En este anexo se van a incluir las tablas que contienen los resultados de las distintas pruebas realizadas al añadir ruido a los ficheros, tal y como se ha explicado en el punto 5.2.3 del documento principal.

Se van a separar en las distintas especies evaluadas y en los escenarios simulados lo que supone cinco tablas por experimento. Se añade a continuación un pequeño índice para facilitar la búsqueda de alguna tabla en concreto.

1 Tablas del escenario con SNR 10 dB.

1.1 Tablas de las Palomas [Tabla Anexo 1 1.1.1]

10% Ruido	1	2	3	4	5	TOTAL
p1	Paloma	Paloma	Paloma	Paloma	Paloma	
p2	Paloma	Paloma	Paloma	Paloma	Paloma	
p3	Paloma	Paloma	Paloma	Paloma	Paloma	
p4	Paloma	Paloma	Paloma	Paloma	Paloma	
p5	Paloma	Paloma	Paloma	Paloma	Paloma	
p6	Paloma	Paloma	Paloma	Paloma	Paloma	
p7	Paloma	Paloma	Paloma	Paloma	Paloma	
p8	Paloma	Paloma	Paloma	Paloma	Paloma	
p9	Paloma	Paloma	Paloma	Paloma	Paloma	
p10	Paloma	Paloma	Paloma	Paloma	Paloma	
p11	Paloma	Paloma	Paloma	Paloma	Paloma	
p12	Paloma	Paloma	Paloma	Paloma	Paloma	
p13	Paloma	Paloma	Paloma	Paloma	Paloma	
p14	Paloma	Paloma	Paloma	Paloma	Paloma	
p15	Paloma	Paloma	Paloma	Paloma	Paloma	
p16	Paloma	Paloma	Paloma	Paloma	Paloma	
p17	Paloma	Paloma	Paloma	Paloma	Paloma	
p18	Paloma	Paloma	Paloma	Paloma	Paloma	
p19	Paloma	Paloma	Paloma	Paloma	Paloma	
TOTAL	19	19	19	19	19	100%

Tabla Anexo 1 1.1.1

Tabla de resultado para las palomas con SNR 10 dB.

Fuente Propia

1.2 Tabla de las Gaviotas [Tabla Anexo 1.2.1]

10%	1	2	3	4	5	TOTAL
g1	Gaviota	Estornino	Gaviota	Gaviota	Error	
g2	Cuervo	Gaviota	Gaviota	Cuervo	Gaviota	
g3	Estornino	Gaviota	Gaviota	Gaviota	Error	
g4	Gaviota	Error	Gaviota	Estornino	Gaviota	
g5	Gaviota	Gaviota	Gaviota	Gaviota	Gaviota	
g6	Estornino	Gaviota	Error	Gaviota	Gaviota	
g7	Gaviota	Estornino	Gaviota	Gaviota	Error	
g8	Gaviota	Estornino	Gaviota	Error	Gaviota	
g9	Gaviota	Gaviota	Gaviota	Gaviota	Gaviota	
g10	Gaviota	Error	Gaviota	Error	Gaviota	
g11	Error	Gaviota	Estornino	Gaviota	Gaviota	
g12	Gaviota	Gaviota	Gaviota	Gaviota	Gaviota	
g13	Estornino	Gaviota	Error	Gaviota	Error	
g14	Error	Gaviota	Gaviota	Gaviota	Gaviota	
g15	Error	Gaviota	Estornino	Gaviota	Gaviota	
g16	Gaviota	Gaviota	Gaviota	Error	Gaviota	
g17	Gaviota	Error	Estornino	Gaviota	Estornino	
g18	Gaviota	Gaviota	Gaviota	Gaviota	Gaviota	
g19	Gaviota	Error	Gaviota	Error	Estornino	
TOTAL	12	12	14	13	13	67%

Tabla Anexo 1 1.2.1

Tabla de resultado para las gaviotas con SNR 10 dB.

Fuente Propia

1.3 Tabla de los Cuervos [Tabla Anexo 1.3.1]

10%	1	2	3	4	5	TOTAL
c1	Cuervo	Estornino	Cuervo	Cuervo	Estornino	
c2	Cuervo	Error	Cuervo	Estornino	Cuervo	
c3	Cuervo	Cuervo	Cuervo	Cuervo	Cuervo	
c4	Gaviota	Cuervo	Cuervo	Cuervo	Cuervo	
c5	Error	Cuervo	Cuervo	Cuervo	Gaviota	
c6	Cuervo	Cuervo	Cuervo	Cuervo	Cuervo	
c7	Gaviota	Cuervo	Cuervo	Cuervo	Gaviota	
c8	Cuervo	Cuervo	Cuervo	Estornino	Cuervo	
c9	Cuervo	Cuervo	Error	Cuervo	Cuervo	
c10	Cuervo	Cuervo	Cuervo	Error	Cuervo	
c11	Cuervo	Estornino	Cuervo	Cuervo	Cuervo	
c12	Cuervo	Gaviota	Error	Cuervo	Error	
c13	Gaviota	Cuervo	Cuervo	Cuervo	Cuervo	
c14	Cuervo	Error	Cuervo	Error	Cuervo	
c15	Cuervo	Cuervo	Cuervo	Cuervo	Cuervo	
c16	Gaviota	Cuervo	Cuervo	Estornino	Cuervo	
17	Cuervo	Error	Cuervo	Cuervo	Cuervo	
c18	Cuervo	Cuervo	Estornino	Cuervo	Cuervo	
c19	Error	Error	Cuervo	Cuervo	Cuervo	
TOTAL	13	12	16	14	15	73%

Tabla Anexo 1 1.3.1

Tabla de resultado para los cuervos con SNR 10 dB.

Fuente Propia

1.4 Tabla de los Estorninos [Tabla Anexo 1.4.1]

10%	1	2	3	4	5	TOTAL
e1	Estornino	Estornino	Error	Estornino	Estornino	
e2	Estornino	Estornino	Estornino	Estornino	Estornino	
e3	Estornino	Estornino	Estornino	Estornino	Estornino	
e4	Estornino	Estornino	Estornino	Estornino	Estornino	
e5	Error	Estornino	Estornino	Estornino	Estornino	
e6	Estornino	Estornino	Estornino	Estornino	Error	
e7	Estornino	Estornino	Estornino	Estornino	Estornino	
e8	Estornino	Estornino	Estornino	Estornino	Estornino	
e9	Estornino	Estornino	Estornino	Cuervo	Estornino	
e10	Estornino	Estornino	Estornino	Estornino	Estornino	
e11	Estornino	Estornino	Estornino	Estornino	Estornino	
e12	Estornino	Estornino	Estornino	Estornino	Estornino	
e13	Error	Estornino	Error	Estornino	Estornino	
e14	Estornino	Estornino	Estornino	Estornino	Estornino	
e15	Estornino	Estornino	Estornino	Error	Estornino	
e16	Estornino	Error	Estornino	Estornino	Estornino	
e17	Estornino	Estornino	Estornino	Estornino	Estornino	
e18	Estornino	Error	Estornino	Estornino	Error	
e19	Estornino	Estornino	Estornino	Estornino	Estornino	
TOTAL	17	17	17	17	17	90%

Tabla Anexo 1 1.4.1

Tabla de resultado para los estorninos con SNR 10 dB.

Fuente Propia

1.5 Tabla de los Gorriones [Tabla Anexo 1.5.1]

10%	1	2	3	4	5	TOTAL
s1	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s2	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s3	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s4	Gorrión	Estornino	Gorrión	Gorrión	Gorrión	
s5	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s6	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s7	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s8	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s9	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s10	Gorrión	Gorrión	Error	Gorrión	Gorrión	
s11	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s12	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s13	Gorrión	Gorrión	Gorrión	Estornino	Gorrión	
s14	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s15	Estornino	Gorrión	Gorrión	Gorrión	Gorrión	
s16	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s17	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
s18	Gorrión	Gorrión	Estornino	Gorrión	Gorrión	
s19	Gorrión	Gorrión	Gorrión	Gorrión	Gorrión	
TOTAL	18	18	17	18	19	94%

Tabla Anexo 1 1.5.1

Tabla de resultado para los gorriones con SNR 10 dB.

Fuente Propia

2 Tablas del escenario con SNR 7 dB.

2.1 Tabla de las Palomas. [Tabla Anexo 2.1.1]

20% Ruido	1	2	3	4	5	TOTAL
p1	Paloma	Cuervo	Paloma	Error	Cuervo	
p2	Paloma	Paloma	Estornino	Paloma	Paloma	
p3	Paloma	Error	Error	Paloma	Paloma	
p4	Paloma	Paloma	Paloma	Error	Paloma	
p5	Paloma	Cuervo	Estornino	Paloma	Error	
p6	Paloma	Paloma	Error	Error	Paloma	
p7	Paloma	Paloma	Paloma	Paloma	Estornino	
p8	Estornino	Paloma	Paloma	Estornino	Paloma	
p9	Paloma	Paloma	Paloma	Paloma	Paloma	
p10	Estornino	Paloma	Paloma	Paloma	Error	
p11	Paloma	Estornino	Paloma	Estornino	Paloma	
p12	Cuervo	Paloma	Paloma	Paloma	Paloma	
p13	Error	Paloma	Error	Paloma	Estornino	
p14	Paloma	Paloma	Paloma	Paloma	Paloma	
p15	Error	Paloma	Estornino	Paloma	Paloma	
p16	Error	Paloma	Paloma	Error	Paloma	
p17	Paloma	Paloma	Paloma	Paloma	Paloma	
p18	Paloma	Paloma	Cuervo	Cuervo	Paloma	
p19	Paloma	Estornino	Paloma	Paloma	Error	
	13	14	12	12	13	67%

Tabla Anexo 1 2.1.1

Tabla de resultado para las palomas con SNR 7 dB.

Fuente Propia

2.2 Tabla de las Gaviotas [Tabla Anexo 2.2.1]

20%	1	2	3	4	5	TOTAL
g1	Estornino	Gaviota	Gaviota	Gaviota	Gaviota	
g2	Estornino	Gaviota	Gaviota	Estornino	Cuervo	
g3	Gaviota	Estornino	Error	Gaviota	Gaviota	
g4	Gaviota	Gaviota	Estornino	Estornino	Gaviota	
g5	Gaviota	Error	Estornino	Gaviota	Cuervo	
g6	Estornino	Gaviota	Gaviota	Gaviota	Gaviota	
g7	Estornino	Error	Estornino	Gaviota	Estornino	
g8	Gaviota	Gaviota	Cuervo	Error	Gaviota	
g9	Gaviota	Estornino	Error	Gaviota	Gaviota	
g10	Gaviota	Gaviota	Estornino	Gaviota	Cuervo	
g11	Estornino	Error	Gaviota	Estornino	Gaviota	
g12	Gaviota	Gaviota	Gaviota	Gaviota	Gaviota	
g13	Error	Gaviota	Error	Estornino	Estornino	
g14	Error	Gaviota	Gaviota	Gaviota	Gaviota	
g15	Cuervo	Estornino	Gaviota	Cuervo	Gaviota	
g16	Gaviota	Gaviota	Gaviota	Gaviota	Estornino	
g17	Error	Cuervo	Estornino	Gaviota	Gaviota	
g18	Gaviota	Error	Gaviota	Gaviota	Estornino	
g19	Gaviota	Gaviota	Error	Gaviota	Gaviota	
	10	11	9	13	12	58%

Tabla Anexo 1 2.2.1

Tabla de resultado para las gaviotas con SNR 7 dB.

Fuente Propia

2.3 Tabla de los Cuervos [Tabla Anexo 2.3.1]

20%	1	2	3	4	5	TOTAL
c1	Gaviota	Error	Cuervo	Cuervo	Error	
c2	Cuervo	Cuervo	Estornino	Cuervo	Cuervo	
c3	Cuervo	Cuervo	Cuervo	Cuervo	Gaviota	
c4	Error	Cuervo	Cuervo	Error	Cuervo	
c5	Cuervo	Estornino	Cuervo	Cuervo	Cuervo	
c6	Gaviota	Cuervo	Error	Estornino	Cuervo	
c7	Cuervo	Cuervo	Cuervo	Cuervo	Gaviota	
c8	Cuervo	Error	Cuervo	Cuervo	Cuervo	
c9	Cuervo	Cuervo	Estornino	Error	Error	
c10	Gaviota	Cuervo	Gaviota	Cuervo	Cuervo	
c11	Cuervo	Gaviota	Error	Cuervo	Estornino	
c12	Error	Cuervo	Cuervo	Estornino	Cuervo	
c13	Cuervo	Cuervo	Cuervo	Cuervo	Cuervo	
c14	Cuervo	Gaviota	Error	Cuervo	Error	
c15	Cuervo	Cuervo	Cuervo	Cuervo	Cuervo	
c16	Cuervo	Cuervo	Cuervo	Gaviota	Cuervo	
c17	Cuervo	Error	Cuervo	Cuervo	Cuervo	
c18	Error	Cuervo	Cuervo	Cuervo	Cuervo	
c19	Gaviota	Cuervo	Estornino	Cuervo	Cuervo	
	12	13	12	14	13	67%

Tabla Anexo 1 2.3.1

Tabla de resultado para los cuervos con SNR 7 dB.

Fuente Propia

2.4 Tabla de los Estorninos [Tabla Anexo 2.4.1]

20%	1	2	3	4	5	TOTAL
e1	Estornino	Estornino	Error	Estornino	Estornino	
e2	Estornino	Estornino	Estornino	Estornino	Estornino	
e3	Estornino	Estornino	Estornino	Estornino	Estornino	
e4	Error	Estornino	Estornino	Estornino	Estornino	
e5	Estornino	Error	Estornino	Estornino	Error	
e6	Estornino	Estornino	Estornino	Cuervo	Estornino	
e7	Estornino	Estornino	Estornino	Estornino	Estornino	
e8	Estornino	Estornino	Estornino	Estornino	Estornino	
e9	Estornino	Estornino	Estornino	Estornino	Estornino	
e10	Estornino	Estornino	Estornino	Error	Estornino	
e11	Cuervo	Estornino	Estornino	Estornino	Estornino	
e12	Estornino	Error	Estornino	Estornino	Estornino	
e13	Estornino	Estornino	Estornino	Estornino	Error	
e14	Estornino	Estornino	Estornino	Estornino	Estornino	
e15	Error	Estornino	Cuervo	Estornino	Estornino	
e16	Estornino	Estornino	Error	Estornino	Estornino	
e17	Estornino	Estornino	Error	Estornino	Estornino	
e18	Estornino	Estornino	Estornino	Error	Estornino	
e19	Error	Estornino	Estornino	Estornino	Estornino	
	15	17	15	16	17	84%

Tabla Anexo 1 2.4.1

Tabla de resultado para los estorninos con SNR 7 dB.

Fuente Propia

2.5 Tabla de los Gorriones [Tabla Anexo 2.5.1]

20%	1	2	3	4	5	TOTAL
s1	Error	Error	Error	Error	Error	
s2	Estornino	Error	Estornino	Error	Error	
s3	Error	Error	Error	Error	Estornino	
s4	Error	Error	Error	Error	Error	
s5	Error	Error	Error	Error	Error	
s6	Estornino	Error	Error	Estornino	Error	
s7	Error	Error	Error	Estornino	Error	
s8	Error	Error	Error	Error	Error	
s9	Error	Error	Error	Error	Error	
s10	Estornino	Error	Error	Error	Error	
s11	Error	Error	Estornino	Error	Error	
s12	Error	Error	Error	Error	Error	
s13	Error	Error	Error	Error	Error	
s14	Error	Error	Error	Error	Estornino	
s15	Error	Error	Error	Error	Error	
s16	Error	Estornino	Error	Error	Error	
s17	Error	Error	Error	Error	Error	
s18	Estornino	Error	Error	Error	Error	
s19	Error	Error	Error	Error	Error	
	0	0	0	0	0	0%

Tabla Anexo 1 2.5.1

Tabla de resultado para los gorriones con SNR 7 dB.

Fuente Propia

3. Referencias Tablas Anexo

– Página en blanco –