UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Diseño de una plataforma web basada en ESP32 para adquisición y procesado de eventos sonoros

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Mención: Sistemas Electrónicos

Autor: Yeremi del Carmen Santana Suárez

Tutor: Miguel Ángel Quintana Suárez

David de la Cruz Sánchez Rodríguez

Fecha: Enero 2019

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Diseño de una plataforma web basada en ESP32 para adquisición y procesado de eventos sonoros

HOJA DE EVALUACIÓN

Calificación:_____

Presidente

Fdo.:

Vocal **Secretario**

Fdo.: Fdo.:

Fecha: Enero, 2019

Agradecimiento

En primer lugar, quiero agradecer a mi mujer, a mis hijos, a mis padres, a mis hermanas y familia, por haber estado siempre conmigo en los momentos difíciles, apoyándome para poder seguir adelante y enseñándome a no rendirme hasta alcanzar mis objetivos. Gracias por saber comprenderme y apoyarme.

En segundo lugar, quiero agradecer a todos los profesores que han contribuido en esta etapa tan importante de mi vida, aportando los distintos conceptos que me han formado como Ingeniero en la titulación de Grado en Tecnologías de las Telecomunicaciones. Destacando en especial, a mis tutores Miguel Ángel Quintana Suárez y David Sánchez Rodríguez, quienes en todo momento me han ayuda y aconsejado de la mejor manera posible, gracias.

Finalmente, y no menos importante, agradecer a mis compañeros de grado quienes han hecho más llevadera esta etapa de mi vida. Pues, nos hemos ayudado y apoyado formando un equipo de compañeros y amigos. En especial, a mi compañero Alejandro Piñan Roescher que ha sabido estar en los buenos y los malos momentos, siempre empujando para subir la cuesta.

Resumen

Los dispositivos electrónicos son cada vez más robustos, en cuanto a: velocidad, capacidad de cómputo y tamaño de almacenamiento. Además, con los avances tecnológicos, podemos encontrar en el mercado dispositivos de bajo coste que pueden cumplir con tareas especificas, dando resultados satisfactorios.

Una parte de las áreas de investigación, llevada a cabo por los ingenieros, centra sus esfuerzos en el reconocimiento de eventos sonoros, siendo de alto interés el reconocimiento del habla humano. De hecho, es posible interactuar con algunos vehículos haciendo uso de la voz, es posible usar software de reconocimiento de voz en nuestros dispositivos móviles para realizar algunas tareas, etc.

Es por lo descrito con anterioridad que, en este Trabajo Fin de Grado se pretende realizar un breve análisis sobre la viabilidad de un dispositivo de bajo coste para la detección de eventos sonoros. Para este caso se estudiará un sistema compuesto por dos subsistemas: el SoC (*System on Chip*) ESP32 y un ordenador de sobremesa. Con el subsistema SoC ESP32, se pretende realizar la adquisición y el procesado de eventos sonoros, en el ordenador de sobremesa se quiere montar un servidor web. Ambos subsistemas se conectarán mediante una red local wifi, esto permitirá que ambos puedan comunicarse, y por tanto, puedan interactuar entre ellos. El ordenador de sobremesa será destinado, entre otros, a almacenar las adquisiciones de los subsistemas ESP32 y los datos correspondientes al procesado de dicha adquisición.

En el subsistema ESP32, será de especial interés analizar los siguientes aspectos: la resolución disponible para la adquisición; qué cantidad de datos se puede adquirir y almacenar para luego trabajar debido a la escasa memoria disponible en el dispositivo; qué tipos de algoritmos para procesado de audio podemos implementar y hasta donde podemos procesar; cuáles son los tiempos de computo para diferentes escenarios.

Índice

Índice de figuras

Índice de códigos

Índice de tablas

Acrónimos

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
AP	Access Point
BBDD	Data Base
BLE	Bluetooth Low Energy
FFT	Fast Fourier Transform
FreeRTOS	Free Real Time Operating System
FTP	File Transfer Protocol
GPIO	General Purpose Input/Output
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
MFCC	Mel Frequency Cepstral Coefficients
MVP	Model View Presenter
PHP	Personal Hypertext processor
PLL	Phase Locked Loop
SoC	System On Chip
STA	Station
TCP	Transmission Control Protocol
TFG	Trabajo Fin de Grado
TIC	Tecnologías de Información y Comunicación
TKIP	Temporal Key Integrity Protocol
UDP	User Datagram Protocol

WEP	Wired Equivalent Privacy
WPA	Wifi Protected Access

Capítulo 1. Introducción

1.1 Antecedentes

Los avances tecnológicos permiten tener en un mismo SoC (SoC, *System On Chip*) varios sensores, y en algunos casos uno o varios actuadores, además de microcontroladores con prestaciones considerables. La capacidad de cómputo de los microcontroladores se ha incrementado, así como las capacidades de almacenamiento, tanto estática como la dinámica.

Las telecomunicaciones presentan un campo donde se realizan grandes avances en redes de datos. Las TIC (Tecnologías de Información y Comunicación) permiten la comunicación entre usuarios, donde un usuario puede ser una persona, una máquina, un sensor, o incluso una red de sensores. Todo esto es conocido como la IoT (*Internet of Thing*), donde cualquier dispositivo electrónico puede ser dotado de capacidad para conectarse a la red e interactuar en ella. La IoT está tomando una gran influencia en la vida cotidiana, y además, en algunos casos como pudiera ser la monitorización de un sistema, ayudan a mejorar la calidad de vida de las personas. Hay muchos dispositivos que se destinan a mejorar la experiencia de los usuarios en áreas como la conducción,

con sistemas que permiten interactuar con el vehículo usando comandos de voz, así como el control de distintos sistemas multimedia. En definitiva, la IoT en su continuo avance, promueve mejoras a la sociedad, permitiendo tener conectados objetos físicos en tiempo real[1] con el fin de comunicarlo todo en una misma red, cuyo cometido es gestionar e interactuar ante la necesidad humana[2].

La ubicación de sensores en lugares de interés que son de difícil acceso, son posible gracias a la evolución de las comunicaciones inalámbricas. Además, en la actualidad se dispone de sistemas que incorporan conectividad inalámbrica junto al microcontrolador, que son de bajo coste, con bajo consumo, y capacidad de cómputo considerable, entre otros, lo cual abarata el precio y fomenta el desarrollo.

La disposición de los sistemas, descritos anteriormente, hacen que los proyectos sean más fácilmente escalables. Como consecuencia del avance encontramos despliegues de sensores cuyo fin puede ser: control inteligente de generadores de energía[3], gestión de cultivos, vigilancia e interpretación de situaciones de peligro o riesgo, ayuda a personas de edad avanzada[4], entre otros.

En este TFG (Trabajo Fin de Grado), se pretende realizar la adquisición de eventos sonoros usando una plataforma de bajo coste y bajo consumo, como es el ESP32. Además, se programará esta plataforma para permitir la gestión de servicios web como servidor y cliente. Aprovechando las características de este SoC se pretende adquirir, procesar y enviar audio o eventos sonoros a un dispositivo destino bien sean un servidor web que haga las solicitudes de forma periódica o a un cliente que los solicite en un determinado momento. Se evaluará el rendimiento del sistema para conocer su capacidad de cómputo, capacidad de almacenamiento, así como otras variables que puedan ser de interés en para nuestro TFG.

1.2 Objetivos

El objetivo principal de este TFG es adquirir y procesar eventos sonoros usando el SoC ESP32, el cual tiene las características de bajo coste, bajo consumo y capacidad de cómputo. Además, se pretende comunicar los datos adquiridos con otros servidores externos para el procesado de los datos si fuese necesario. Para cumplir con este objetivo general es necesario realizar los siguientes objetivos específicos:

- Aprender técnica de procesado de audio.

- Conocer el funcionamiento del ESP32.
- Implementar el sistema para adquisición y procesado de datos en el ESP32.
- Implementar el servidor web tanto en el ESP32 como en una máquina externa.

1.3 Sistema propuesto

El sistema propuestos permite al usuario realizar capturas de audio y obtener informes asociados al procesado de dicho audio. Los subsistemas ESP32 son los encargados de obtener las capturas de audio. El usuario se comunica con el sistema por medio de una interfaz web. La interfaz web permite al usuario configurar los subsistemas ESP32 y el proceso de captura. El sistema debe ser capaz de almacenar las capturas de audio y los datos asociados a dicho proceso. Ver Figura 1

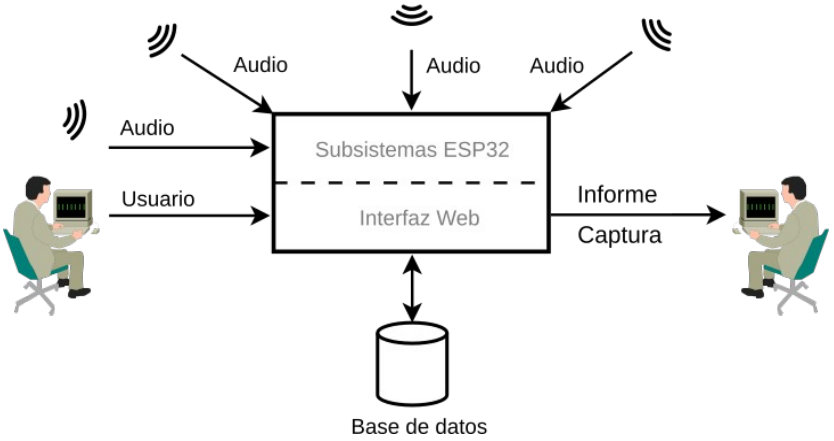


Figura 1: Sistema propuesto

Los subsistemas ESP32 son los encargados de la captura y procesado del audio. Cada uno de estos subsistemas se conectan al sistema global mediante una red inalámbrica Wifi. Estos subsistemas se descomponen en varios bloques o módulos: uno encargado de la captura de audio, otro para el procesado del audio y otro que da soporte a la interfaz web. Este último bloque permite la comunicación tanto con el sistema global, como el usuario si se utiliza de manera autónoma o independiente. Ver Figura 2

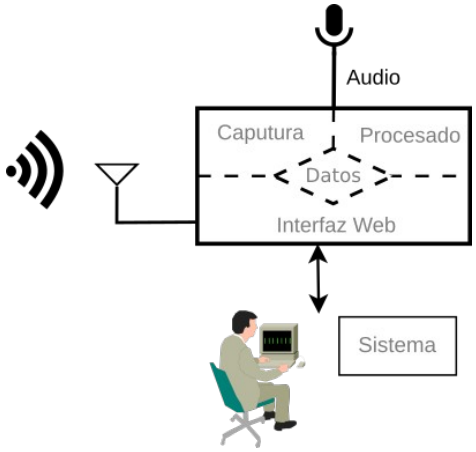


Figura 2: Subsistema ESP32. Diagrama de bloques.

La interfaz web permite, al sistema global o usuario, realizar modificaciones en la configuración del subsistema. El subsistema almacena la configuración en su modelo de datos. La interfaz web habilita al usuario la descarga de los datos capturados, en un fichero con formato wav.

El módulo de captura tiene un micrófono. Es el encargado de la adquisición de audio. Obtiene su configuración del modelo de datos. Almacena el audio en un bloque de memoria.

El módulo de procesado es el encargado del procesamiento. Su configuración la toma del modelo de datos. Este utiliza el bloque de memoria, mencionado anteriormente, para obtener el audio, realizar el procesado y almacena el resultado en otro bloque de memoria.

1.4 Peticionario

Actúa como peticionario del presente TFG la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria como requisito indispensable para la obtención del título de Graduado en Ingeniería en Tecnologías de la Telecomunicación, tras haber superado con éxito las asignaturas especificadas en el Plan de Estudios.

1.5 Estructura del documento

El presente documento está dividido en tres partes diferenciadas: Memoria, Pliego de condiciones, y Presupuesto. A su vez, la Memoria se ha estructurado en seis capítulos, además de las referencias bibliográficas.

La memoria tiene la siguiente estructura:

- **Capítulo 1**, ***Introducción.*** Se presentan los motivos que han dado lugar al planteamiento de este TFG, se explican los conceptos básicos que se tratarán en este y se presentan, además, el sistema propuesto, los objetivos, el peticionario y la estructura del documento.
- **Capítulo 2**, ***Introducción a las tecnologías y entornos utilizadas.*** Se presenta un breve estudio del estado del arte sobre el uso de los SoC en IoT, se estudian algunos de los sensores que hay en el mercado para la captura de audio y de qué librerías dispone. Se estudian las características básicas de procesado de audio, siendo de especial interés la FFT (*Fast Fourier Transform)*, para su posterior evaluación sobre su implementación en el SoC. Se hace un breve estudio de la gestión asociada a los eventos web, y de la forma de interactuar con el protocolo de comunicaciones HTTP (*Hypertext Transfer Protocol*). Para finalizar se realiza un estudio sobre algunos lenguajes de programación que pueden ser usados para el desarrollo del sistema propuesto.
- **Capítulo 3, *Configuración e instalación del entorno de trabajo para el ESP32.*** Se hace un breve estudio de algunos IDEs (*Integrated Development Environment*) que sirvan para la programación del SoC ESP32. Se estudia la configuración básica, así como las librerías necesaria para el desarrollo de este TFG. Se presenta la forma en la que se hará el muestreo, cuál es la estructura

necesaria para el almacenamiento de la información y cómo se realizará la validación. Se hace un estudio sobre las librearías que son necesarias para montar un servidor web sobre el SoC ESP32, se presenta cual es la estructura básica para la construcción del programa y la gestión de los eventos asociados al servicio web.

- **Capítulo 4, *Desarrollo del firmware del ESP32.*** Se da una explicación más detallada sobre el modelo de la solución buscada. Se analizan las especificaciones necesarias para realizar las capturas e implementar el portal web. Se describe el desarrollo e implementación del algoritmo de procesado, se definen las pruebas realizas para la validación del sistema y se presenta la integración dentro del sistema desarrollado.
- **Capítulo 5, *Desarrollo del servidor web para la recopilación de datos.*** Se desarrolla lo necesario para la implementación del servidor web. Se hace un resumen sobre la instalación del sistema base, se mencionan las diferentes librerías necesarias, así como las aplicaciones usadas, para el servidor web. Se presenta los modelos usados para la programación del portal web, se habla de las distintas vistas que hay en la interfaz del usuario, se detalla el proceso que sigue el servidor web para la gestión de los subsistemas así como la gestión de la interfaz del usuario. Además, se muestra la estructura de la base de datos empleada para el almacenamiento de los datos.
- **Capítulo 6, *Conclusiones y trabajos futuros.*** Se hace un resumen sobre las conclusiones extraídas al finalizar el TFG. Se mencionan alguna mejoras que se podrían dar al sistema en nuevas versiones.

La segunda parte del documento contiene el Pliego de condiciones, y la tercera parte el Presupuesto. Además, en el CD se encuentra los anexos sobre la documentación del código implementado en este TFG, así como los fichero usados para la validación del sistema. En dicho anexo, la documentación del código, se ha generado mediante la herramienta Doxygen de manera automatizada a partir de los comentarios incluidos en el código fuente.

Capítulo 2. Introducción a las tecnologías y entornos utilizados

2.1 SoC utilizados en IoT

Arduino[5], fue la primera plataforma open hardware que incluía un microcontrolador y que al salir al mercado supuso un cambio para la comunidad de usuarios, siendo utilizada en un gran número de desarrollos. Muchas empresas han aprovechado las contribuciones aportadas por la comunidad sobre el desarrollo de sistemas para un inmenso abanico de escenarios. Si bien es cierto que no cuenta con una capacidad de procesamiento elevada, hay que tener presente que dicha plataforma fue diseñada para realizar aplicaciones de control que no requieren una elevada precisión. La plataforma ha evolucionado de forma continuada, llegando a disponer de procesadores con doble núcleo. Una de sus ventajas es la facilidad con la que se pueden acoplar desde una simple pantalla para mostrar datos hasta controladores ethernet/WiFi. Sus capacidades varían según el modelo, su versión básica cuenta con: 26 pines de I/O, un microcontrolador ATmega328P de 16 MHz; y su versión más completa cuenta con: 54

pines de I/O, un microcontrolador Atmel SAM3X8E ARM Cortex-M3 que trabaja a 84 MHz.

Figura 3: Arduino Uno

Waspmote[6], de la empresa Libelium, es una solución comercial de la plataforma open source completa. Dispone de compatibilidad con más de 60 sensores y múltiples tecnologías de comunicación. Lleva un microcontrolador ATmega1281 que opera a una frecuencia de 14.74 MHz. Su consumo en operativa es de 17 mA y en modo hibernación es de 7 µA. Lleva incluido: 8 conectores para acoplar una batería o una placa solar, entre otros tiene una memoria flash de 128 kB, dispone de 23 pines de entrada y salida, tiene hardware dedicado para procesos criptográficos, dispone de un acelerómetro.

Figura 4: Waspmote

Particle[7] es un componente hardware diseñado y fabricado especialmente para conectividad wifi, 2G y 3G en espacios reducidos. Dispone de un controlador ethernet Broadcom Wi-Fi. El WiFi tiene soporte para el 802.11 b/g/n. El diseño es open source. Tiene un microcontrolador ARM Cortex M3 que opera a 120 MHz. Lleva 18 pines GPIO (*General Purpose Input/Output*) de entrada y salida. Esta pensado para trabajar con

FreeRTOS (*Free Real Time Operating System*). Ofrece librerías para publicar datos, variables y eventos en la nube. Tiene una SDKs de integración rápida para Web, iOS y Android. Las APIs están protegidas con seguridad basada en API KEY.

Figura 5: Particle Photon

ESP8266 WROOM[8], es un microcontrolador de bajo consumo de 32 bits. Implementa la pila de protocolos TPC/IP, provee conversores ADC de 10 bits, tiene una RAM integrada de 80 kB, incorpora un reloj RTC que permite el control de estados de baja energía y tiene hardware dedicado para la criptografía. El WiFi tiene soporte para el 802.11 b/g/n y soporta seguridad WPA/WPA2 (*Wifi Protected Access*) con encriptación WEP (*Wired Equivalent Privacy*), TKIP (*Temporal Key Integrity Protocol), AES (*Advanced Encryption Standard). Además, una de sus propiedades es que dispone del certificado Wi-Fi de Allience. Soporta diferentes configuraciones de modos de infraestructuras: Cliente Wifi, conexión punto a punto y punto de acceso. Viene con un procesador Tensilica L106 de 32 bits, con un núcleo que trabaja a una velocidad de 80MHz. Opera con tensiones que varían de 2.5 V a 3.6 V. Soporta actualización del firmware por red, IPv4 y los protocolos TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*), HTTP y FTP (*File Transfer Protocol*). Su tamaño es bastante reducido con tan solo 18mm x 20mm.

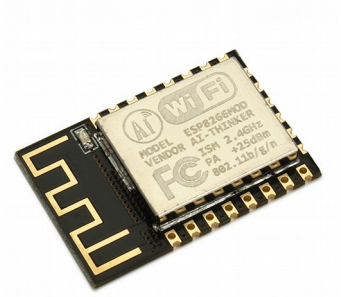


Figura 6: ESP8266

ESP32[9], es nuevo microcontrolador de la compañía Espressif orientado al desarrollo de dispositivos para la IoT. Es el sucesor del ESP8266 ya mencionado

anteriormente. Implementa la pila de protocolos TPC/IP . Utiliza un procesador de Tensilica Xtensa Dual-Core LX6 de 32 bits a 160 ó 240 MHz. La capacidad de los dos núcleos permite que se puede dedicar uno de ellos a la gestión de las comunicaciones y el otro a procesos programados. Tiene una memoria RAM de 520 kB además de poder usar una memoria externa de hasta 8Mb. Soporta memoria flash externa de hasta 16 MB. Tiene una ROM interna de 448kB utilizada para el bootloader y funciones internas. Posee una interfaz Bluetooth 4.2. Viene con un acelerador criptográfico por hardware para las comunicaciones. Soporta los modos de infraestructuras Station, P2P y SoftAP. El WiFi tiene soporte para el 802.11 b/g/n y soporta seguridad WPA/WPA2 con encriptación WEP/TKIP/AES. Dispone de 32 pines GPIO de entrada y salida de las cuales 12 entradas son analógicas las cuales incorporan conversores analógico-digital de 12 bits de resolución. Opera con una tensión de 3.3 V. Integra un chip bluetooth de baja energía o también llamado BLE (B*luetooth Low Energy*).

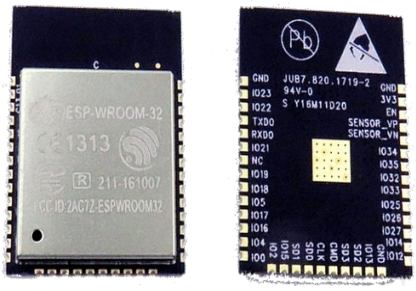


Figura 7: ESP32

En la Tabla 1 aparecen recogidas las características más importantes de estos SoC.

SoC	Procesador	Velocidad	Memoria	Pines I/O	Comunicación	Precio (€)
Arduino	ATmega328P	16 MHz	32 kB	26	Serial	20,00
Waspmote	ATmega1281	14,8 MHz	128 kB	21	Serial 16 tecnologías radio	178,00
Particle	ARM Cotex M3	120 MHz	128 kB	18	Serial, WiFi	15,40
ESP8266	Tensilica L106	160 MHz	50 kB	32	Serial, WiFi	9,00
ESP32	Xtensa LX6	240 MHz	520 kB	38	Serial, WiFi, BLE	12,00

Tabla 1: Resumen de las características más importantes de los SoC

2.2 Sensores de audio

El mercado dispone de una amplia variedad de sensores para poder capturar audio y eventos sonoros. Entre los más utilizados por la comunidad de desarrollo esta el detector de sonidos KY-038[10], este micrófono es apto para la detección de presión sonora. Es poco versátil, y tiene un coste muy bajo que ronda los 0.80€. Es usado en aplicaciones de domótica y automatización. Este dispositivo cuenta con un comparador LM393 que, según el umbral, ajustable por hardware, nos da una salida digital al detectar niveles sonoros. Por ello se usa ampliamente en aplicaciones en las que se requiere actuar frente a eventos sonoros, como puede ser la activación de una lámpara al detectar un golpe de palmada o la orientación de un robot, entro otros. Para calibrar la referencia del comparador dispone de potenciómetro y de un led que se activa con los disparos producidos por el comparador. Dispone de 2 pines, uno da una salida digital cuya salida depende del ajuste del umbral comentado anteriormente y el otro pin nos da valores analógicos correspondientes a la media de la presión sonora obtenida por el micrófono.

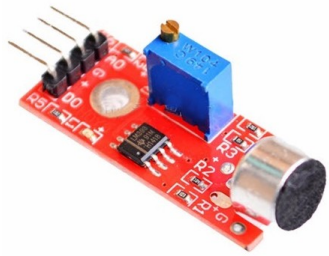


Figura 8: Detector de sonidos KY-038

Otro dispositivo usado para la adquisición de audio es el VS1003[11]. Esta placa, además del micrófono para capturar audio, viene provista de un decodificador de audio que soporta diferentes formatos, entre los que se encuentra el mp3 y el wav. Tiene un valor medio de 18€, tiene: salida analógica y digital. Incorpora un multiplicador de frecuencia para el reloj PLL (*Phase Locked Loop*) interno. Trabaja con un único reloj de 12 MHz. Tiene soporte para streaming de mp3 y wav. El driver para el audífono incluido es capaz de soportar una carga de 30 ohms. Dispone de 5.5 kB para datos de usuario.

Figura 9: Placa VS1003

Otra de las variantes disponibles para adquisición de audio es el detector de audio MAX4466[12]. Esta placa está provista de un micrófono de condensadores y un amplificador operacional de bajo ruido. Tiene la capacidad de detectar respuestas dentro del rango de frecuencias comprendido entre 20Hz y los 20kHz. El amplificador que incorpora tiene un excelente rechazo al ruido de la fuente de alimentación. Dispone de ganancia variable pudiendo tomar valores en el rango de 25x a 125x. Es apto para proyectos de adquisición de audio. Es utilizado para reconocimiento de voz, y muestreo de audio. Dispone de un único pin de salida, y éste no está diseñado para conectarse directamente a un altavoz. Su rango de tensión de operación va de 2.4 V a 5V. Su precio medio de mercado está entorno a los 6€.

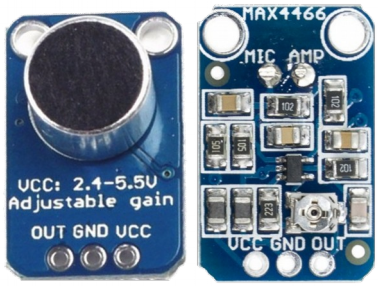


Figura 10: Micrófono con MAX4466

En la Tabla 2 aparecen recogidas características de los micrófonos

Placa	Precio	Salidas	Procesamiento
KY-038	0,80 €	2	No
VS1003	10,50 €	4	Si
MAX4466	2,68 €	1	No

Tabla 2: Resumen de las características de los sensores

Los micrófonos se ven sujetos a muchos factores que afectan a su correcta percepción del sonido. Dependiendo de la aplicación final se debería tener en consideración algunos parámetros del micrófono, como pueden ser su respuesta en

frecuencia y la distorsión. Existe una gran variedad de micrófonos como son los de electreto, dinámicos, de cristal y de carbón.

	Tipo de Micrófono			
Parámetro	***Electreto***	***Dinámico***	***Carbón***	***Cristal***
Respuesta en frecuencia	Excelente	Excelente	Regular	Bien
Distorsión	Muy Bajo	Muy Bajo	Alto	Bajo
Cancelación de ruido	Excelente	Bien	Regular	Regular
Tamaño	Pequeño	Medio	Grande	Grande
Peso	Bajo	Medio	Medio	Bajo
Nivel de salida (Voltaje)	Bajo	Medio	Alto	Alto
Impedancia	Alto	Bajo	Bajo	Alto

Tabla 3: Características principales según el tipo de micrófono

En la Tabla 3 se muestra una comparativa de los micrófonos mencionados anteriormente, y basados en la comparación se recomienda usar micrófonos de electreto o bien dinámico.

2.3 Procesado de audio: algoritmos más utilizados

La herramienta más utilizada para el análisis y procesamiento de audio es la FFT, que nos permite estudiar el audio en el dominio de la frecuencia.

La FFT se aplica a un espacio limitado del audio. El límite viene determinado por su longitud. Esto se conoce como enventanado de la señal.

El enventanamiento del audio en el dominio temporal no es más que una multiplicación del audio con una pequeña ventana de tiempo, dicha ventana hará que toda la señal que esté fuera de sus límites tome un valor cero. El enventanamiento se corresponde con la convolución del espectro del audio con el espectro de la ventana. El enventanado permite analizar señales con longitudes muy grandes con menor exigencia computacional. La resolución del espectro obtenido a la salida del enventanado dependerá en mayor medida de la resolución o número de muestras que haya en él.

Existen varios tipos de ventana que se aplican en el procesado de audio, entre los que cabe mencionar:

- Ventana rectangular
- Ventana de Hamming
- Ventana de Hanning
- Ventana de Blackman

El usar un tipo u otro de ventana dependerá de la aplicación final. En el orden en que han sido citadas en la Figura 11, se diferencian en la distorsión final, donde la ventana rectangular, aun siendo la menos compleja de implementar, es la que mayor distorsión tiene. La ventana más utilizada en el procesamiento de audio es la de Hamming. La ventana de Hamming tiene un buen coeficiente de rechazo.

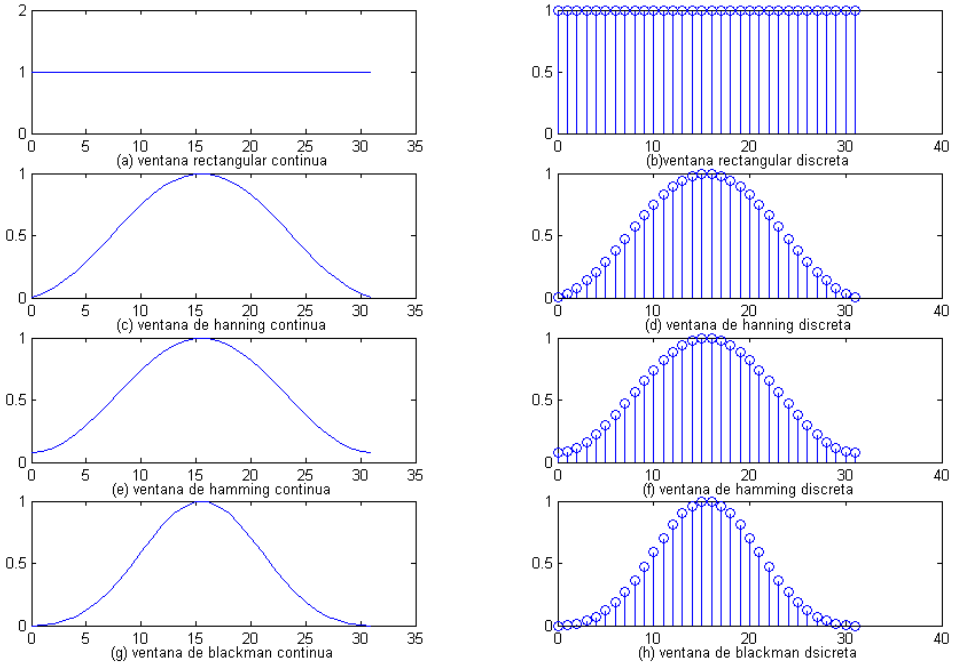


Figura 11: Ventana rectangular. Ventana de Hanning. Ventana de Hamming. Ventana de blackman.

La técnica de enventanamiento puede ser usada con solapamiento, contiguas o incluso separadas, como se muestra en la Figura 12. El problema del enventanado con solapamiento reside en el coste computacional ya que al aumentar el solape aumentará el número de ventanas a procesar para un mismo sonido.

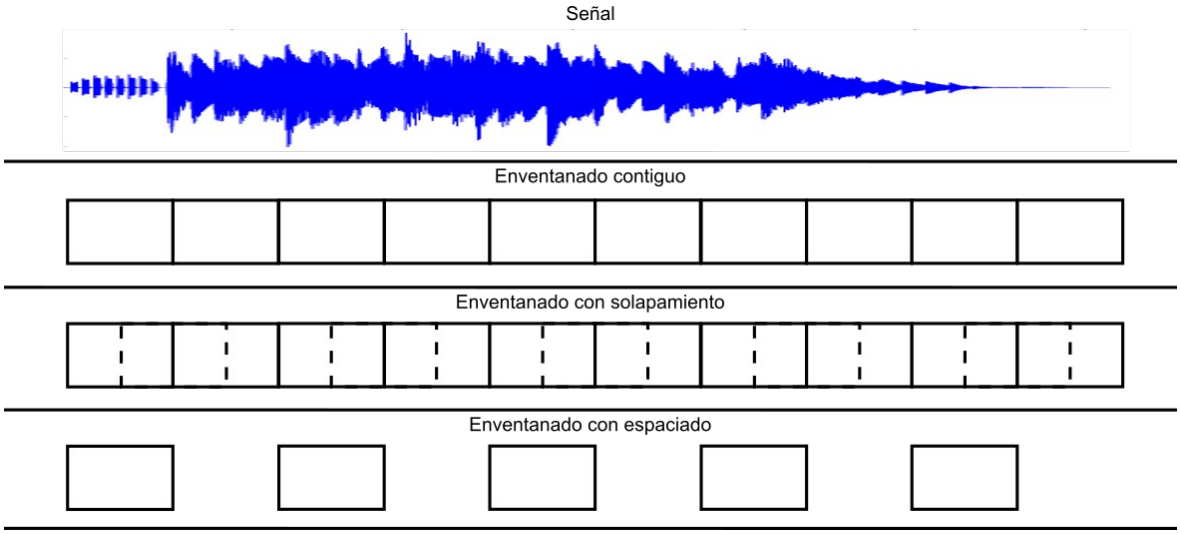


Figura 12: Enventanado contiguo. Enventanado con solapamiento. Enventanado con separación.

Al calcular la FFT en un enventanado tenemos como resultado un valor complejo con su respectiva parte real y parte imaginaria. Lo que se debe tener en consideración es que el movimiento de la ventana producirá cambios en la fase del sonido, mientras que el módulo se mantendrá constante. Es por lo dicho anteriormente que solo será de interés trabajar con el módulo del enventanado.

Para trabajar con una versión muestreada del sonido es necesario cumplir con el Teorema de muestreo de Nyquist-Shannon o Teorema de Nyquist, el cual demuestra que para poder recuperar una señal analógica que ha sido muestreada es necesario realizar dicho muestreo, al menos, al doble la frecuencia de la señal original.

Realizado todo el proceso de enventanado con la FFT sobre el audio, se pueden extraer patrones característicos del mismo. Un método eficiente para la extracción de características del sonido son los MFCC (*Mel Frequency Cepstral Coefficients*). Los MFCC son una representación de una señal enventanada en el tiempo la cual ha sido derivada de una FFT, pero en una escala de frecuencias no lineal.

2.4 Fundamentos sobre servidores web y protocolo http

Un servidor web es una aplicación cliente servidor, cuya función principal es la de almacenar, gestionar y devolver páginas HTML (*HyperText Markup Language*). Existen varias alternativas:

- **Ngnix**, servidor web multiplataforma. Es utilizado por muchos sitios web como Netflix, WordPress, Facebook, entre otros
- **Microsoft** IIS, servidor web que incluye servicios para: FTP, SMTP, HTTP, HTTPS; convierte cualquier ordenador bajo windows en un servidor web, tanto de forma local como remota.
- **Google Server**, conocido como GWS, es el servidor web que Google utiliza en sus infraestructuras.
- **Apache Server**[16], servidor web HTTP de código abierto, y multiplataforma.

En la Figura 13 puede verse una gráfica, obtenida del sitio web netcraft.com[13], del uso de aplicaciones para servidor web. En ésta puede observarse que el más utilizado es el Apache Server.

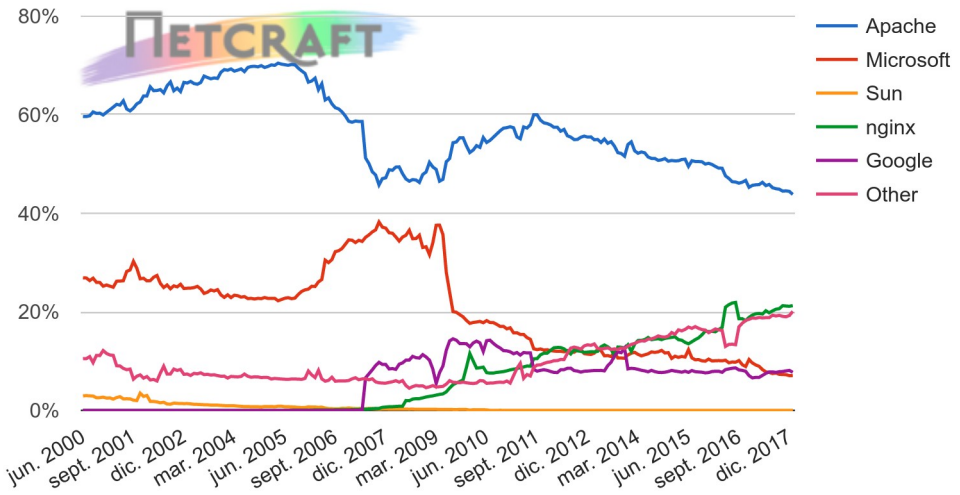


Figura 13: Ranking del uso de servidores web. Imagen obtenida de netcraft.com

Por lo comentado anteriormente, se usará como software de gestión para el servicio web Apache Server. Puede ser instalado, bien compilando el software en la máquina destino o bien descargando el binario que corresponda con la máquina destino desde los repositorios oficiales. En nuestro caso, será necesario activar el módulo rewrite, además se usara la configuración por defecto del servidor con lo que éste gestionará las peticiones dirigidas al puerto TCP 80, y su directorio web estará ubicado en la dirección /var/www/html.

Para la gestión del modelo de datos existen varias alternativas, de las que se destacan por su uso en la web:

- **MySQL.** Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation. Ofrece la facilidad de uso, escalabilidad y un buen rendimiento. Actualmente está siendo utilizada por grandes empresas como: Facebook, Google, Twitter, Uber y Booking.com, entre otras.
- **PostgreSQL.** Es un potente sistema de base de datos relacional de objetos de código abierto, su arquitectura altamente testeada y que aporta una gran confiabilidad, integridad de datos, conjunto de características sólidas, extensibilidad. Destacando la dedicación de la comunidad que da soporte al software, ofreciendo constantemente soluciones innovadoras y de alto rendimiento.

Para el desarrollo de este TFG usaremos como motor de datos el PostgreSQL. En el se debe crear un usuario con el nombre `dryst` y contraseña de acceso `1234`, se creará una base de datos llamada acts cuyo propietario debe ser dryst.

Para la programación web usaremos PHP (*Personal Hypertext Processor*), JavaScript, y para la presentación de información al usuario usaremos el lenguaje de etiquetado HTML.

Para la comunicación con los subsistemas en modo cliente se usará la librería php-curl y para la comunicación de la web con la base de datos se utilizará la librería php-pgsql.

El servidor web recibe las peticiones del cliente, las gestiona y devuelve la respuesta correspondiente, ver Figura 14. El cliente hace las peticiones usando el estándar HTTP. El servidor atiende las peticiones HTTP por el puerto TCP 80.

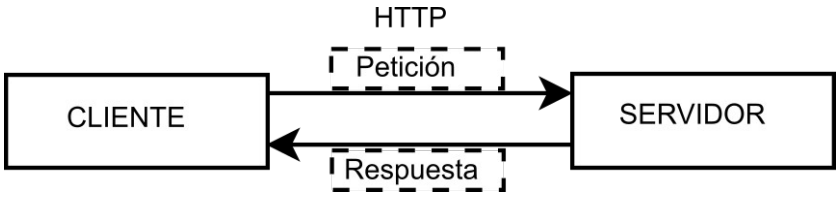


Figura 14: Comunicación mediante el protocolo HTTP

El cliente puede realizar una petición usando un navegador, pero también puede usar cualquier software diseñado para tal fin, como puede ser el telnet.

El servidor web permite trabajar con contenido estático y dinámico, permite la asignación de directivas de seguridad, donde la más usada consiste en la creación de

un fichero llamado '.httacces', el cual se coloca dentro del directorio de interés, éste puede contener información sobre usuarios con acceso, terminales con acceso, etc., esto dependerá de la aplicación final.

El protocolo de comunicación HTTP esta orientado a transacciones y es un protocolo sin estados. La dos formas más usadas para realizar una petición son por GET y el POST. El GET envía los datos de la petición codificados en la url, el POST envía los datos en el cuerpo de la petición.

De cara a este TFG, nos centramos en la gestión de las peticiones de tipo POST realizadas mediante HTTP. Dada las características de nuestra aplicación es de necesario trabajar con contenidos dinámicos. No aplicamos directivas de seguridad por no ser un objetivo definido en las especificaciones de este TFG. La gestión de datos será programada sobre PHP.

2.5 Lenguajes de programación

En este TFG necesitamos trabajar con dos sistemas totalmente distintos tanto en prestaciones, arquitectura, capacidad de almacenamiento, así como en entornos y lenguajes de programación. El primero de ellos es el uso de microcontroladores con software empotrado y el otro la programación de aplicaciones webs sobre servidores.

Gracias a los avances que han sufrido los compiladores, del lado del SoC podemos encontrar variedad en cuanto a los lenguajes de programación. Hace años, los lenguajes más usuales para la programación de chips era por excelencia el ensamblador, y la programación en C. La evolución de los lenguajes fue pasando desde microbasic hasta llegar a micropython. Este último cuenta con una gran aceptación por parte de la comunidad de usuarios, por lo que se está poniendo de moda en la actualidad. Debido a la evolución de los compiladores se hizo posible usar lenguajes como el C++. Este lenguaje es más atractivo para los programadores acostumbrados a desarrollar aplicaciones de escritorio. De cara a este proyecto, podemos hablar de entornos de programación como puede ser el Lua, Python, Nodemcu, etc. que han sido preparados para trabajar sobre estos SoC. EL uso de un lenguaje u otro dependerá en gran medida del desarrollador software y de las exigencias del sistema. Por lo general, todo puede ser programado por lenguajes de alto nivel, siendo en pocas ocasiones necesario recurrir al lenguaje ensamblador para realizar algunas tareas. C++ nos permite la programación orientada a objetos la cual

facilita el desarrollo de grandes proyectos y el desarrollo de aplicaciones. Existen plataformas de programación que usan bloques visuales para orientar al usuario en el desarrollo de aplicaciones. Estas plataformas ponen sus esfuerzos en aplicaciones de control, de uso cotidiano, haciendo un uso básico del SoC. Para el caso de nuestro proyecto se usará la programación C++ para programar el SoC.

Del lado del servidor hay diferentes lenguajes que permiten la interacción entre máquinas, como PHP, Nodejs, Phyton, JavaScript, etc. El Nodejs[14] permite realizar la comunicación cliente-servidor y actualmente está siendo apoyado por la Linux Foundation, cuenta con soporte, trabaja con un modelo de operaciones de entrada-salida sin bloqueo y orientado a eventos. Este es un entorno de ejecución para JavaScript orientado a eventos asíncronos, posee una cantidad considerable de librerías de código abierto, ha sido diseñado para crear aplicaciones en la red que sean fácilmente escalables. Python[15] es un lenguaje de programación interpretado, es multiplataforma y permite programación orientada a objetos y actualmente es usado en el desarrollo de aplicaciones web. El JavaScript es un lenguaje de programación interpretado, permite también una programación orientada a objetos. El PHP es el lenguaje de código abierto para programación y desarrollos web más extendido. Tiene una amplia variedad de librerías de código abierto y permite realizar una programación orientada a objetos. En este proyecto usaremos para la programación de nuestra aplicación web en el servidor el lenguaje PHP. Asimismo, las páginas webs generadas incluirán scripts escritos en javascripts para modelar su comportamiento, verificación y envío de peticiones al servidor.

Capítulo 3. Configuración e instalación del entorno de trabajo para ESP32

3.1 Entornos de desarrollo para la programación del ESP32

Actualmente existen varios entornos que ayudan a la programación con este tipos de SoC. Dependiendo de las necesidades de cada proyecto será de interés uno u otro. La plataforma IDE[17] de Arduino, mostrada en la Figura 15, es una plataforma bastante completa y se mantiene actualizada. Ésta integra una gran cantidad de librerías que son de mucha ayuda en la realización de las tareas más habituales. Dicho entorno está pensado para trabajar desde lo más básico, la configuración del entorno es muy sencilla, siendo ideal para aquellos usuarios con poca experiencia. Además, en

su última versión han desarrollado un entorno de programación vía web, no siendo necesario instalar el entorno en el pc y es independiente al sistema operativo.

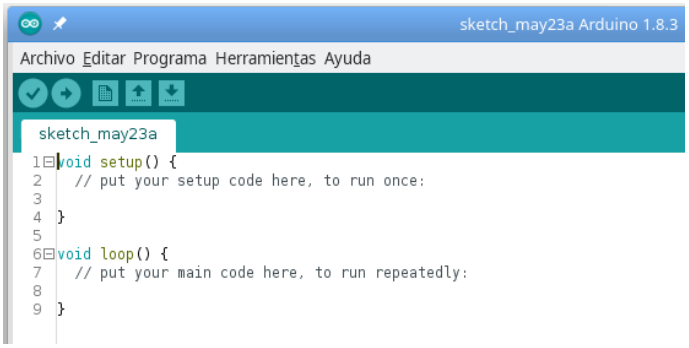


Figura 15: Entorno de desarrollo Arduino

En la Figura 16 se muestra otro entorno de desarrollo, el PlatformIO[17], montado sobre Atom que es open source y diseñado para desarrollo de sistemas para la IoT. La herramienta es algo más compleja de configurar que la descrita anteriormente pero por contrapartida dispone de más flexibilidad a la hora de escribir el firmware para muchos tipos de SoC. Este sistema usa un fichero de configuración que permite incluir librerías que están fuera del propio entorno de trabajo de manera sencilla. Permite organizar el proyecto en diferentes directorios. Integra control de versión, lo cual es muy interesante de cara a mantener un control y copia del proyecto cuando hay más de un usuario trabajando sobre el mismo. Cabe destacar el soporte por parte de la comunidad, lo que le permite tener disponible gran variedad de librerías, y plugins que ayudan en tareas importantes como puede sera la documentación del código.

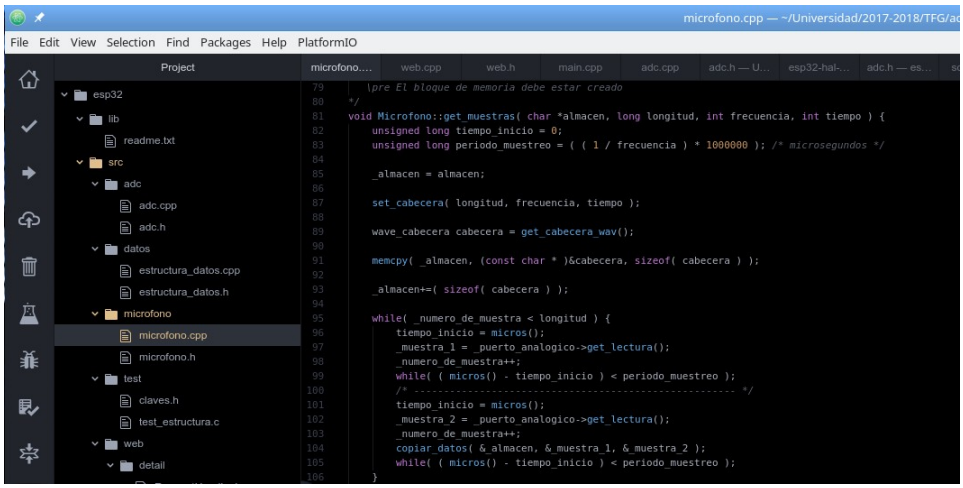


Figura 16: Entorno de desarrollo PlatformIO

Hay otras alternativas para programar el SoC, como puede ser un editor de texto, compilar el proyecto usando el gcc ó g++, según sea el caso, y luego subir el fichero

binario al chip, pero es muy tedioso y no ofrece ninguna ventaja sobre el uso del los entornos de desarrollo descritos.

En este proyecto se usará el PlatformIO por la flexibilidad y adecuación de librerías disponibles. Para realizar una correcta configuración del sistema se usará de guía la web de ESP-IDF [18], en la cual se exponen, una buena variedad de ejemplo de uso, los esquemáticos y guías de configuración.

3.2 Configuración básica del ESP32

Para el proyecto que se lleva acabo es necesario usar el ADC(*Analog-to-Digital Converter*) para poder tomar los datos de la salida del micrófono. El SoC ESP32 tiene dos canales para los ADC, ADC1 y ADC2, en este TFG haremos uso del ADC1, y no se hará uso del ADC2 pues está destinado a la gestión de las comunicaciones WiFi, y podría interferir en la toma de datos durante el proceso de muestreo o provocar errores cuya consecuencia se manifieste con el reset del SoC ESP32. Tal como se puede observar en la Figura 17, el canal ADC1 está multiplexado con los GPIO: 36,39,34,35,32 y 33.

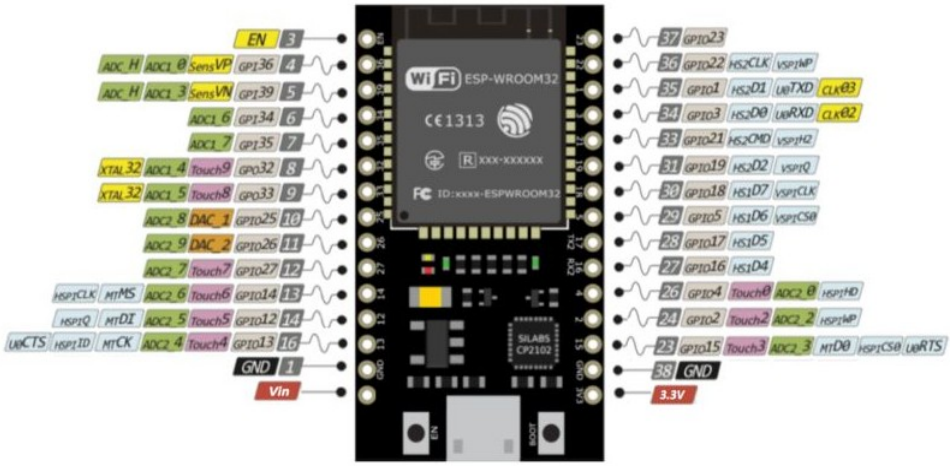


Figura 17: ESP32, pinout

Los pines etiquetados como SVP y SVN, situados en la GPIO36 y GPIO39, son utilizados por el SoC ESP32 para la gestión del sensor Hall que tiene incorporado, según el manual facilitado por el fabricante , en la página 56. Estos pines no aceptan entradas, sólo lecturas. Por lo comentado anteriormente, será usado el ADC1_6 que se encuentra situado en la GPIO34 o pin 10. Para la configuración del ADC usaremos la librería adc facilitada por el fabricante. Dicha librería es obtenida del sistema FreeTOS, sistema usado en aplicaciones de tiempo real. La librería nos facilita la tarea de

configuración de los parámetros principales necesarios para el correcto funcionamiento del ADC.

Para la gestión de las comunicaciones por WiFi podemos usar la librería `esp_wifi` que es facilitada por el fabricante, ó `wifi` la cual es facilitada por la comunidad de Arduino. Para el desarrollo de nuestro TFG haremos uso de la librería wifi.h debido a que ya hemos trabajo con dicha librería con anterioridad, lo que nos ayuda a usarla en nuestro proyecto.

Una parte importante del proyecto reside en la gestión del muestreo a frecuencia constante para lo cual haremos uso de las interrupciones. Para poder trabajar con las interrupciones tenemos varias alternativas, de las cuales se destacan la librerías ofrecidas por el fabricante y las ofrecidas por la comunidad de Arduino. En nuestro proyecto haremos uso de la librería ofrecida por el fabricante, llamada `timer`, que se ajusta al hardware utilizado y tiene suficiente documentación con respecto a su uso. La librería facilita la configuración y gestión de las interrupciones del SoC ESP32.

Para poder comunicar con el SoC ESP32 vía web usando el protocolo HTTP haremos uso de la librería `ESP32WebServer`[19], esta librería facilita la captura y configuración de la comunicación HTTP, también será utilizada para presentar y gestionar un pequeño portal web sobre el SoC ESP32. De cara al usuario, el portal web le permite trabajar con el subsistema y de cara al servidor, el control y gestión del protocolo HTTP permite la centralización y manejo del subsistemas de forma remota. Con el uso de esta librería realizaremos la comunicación y control entre el servidor y el subsistema.

Para realizar la FFT se toman en cuenta dos alternativas. La primera alternativa es usar la librería desarrollada por Robin Scheibler. Dicha librería se encuentra disponible en Github y es de código libre. La segunda alternativa es usar la librería que ha sido desarrollada por la comunidad de Arduino. La primera alternativa utiliza variable de tipo float en el cálculo de la FFT, mientras que la segunda alternativa usa variables de tipo double. Usar variables de tipo float es de interés dado el poco almacenamiento del que disponemos, la segunda alternativa requiere el doble de almacenamiento más que la primera, por lo que la descartamos.

Durante la validación, la librería desarrolla por Robin Scheibler, presentó errores de: gestión de memoria, desbordamiento de la pila y errores en el cálculo de la FFT. Por lo cual, no permite alcanzar el objetivo del TFG.

Debido a esto, para poder cumplir con los objetivos marcados en el presente TFG se tomó la decisión de realizar la implementación propia del algoritmo necesario para llevar a cabo el cálculo de la FFT.

3.3 Prototipo básico para la adquisición de audio

La adquisición se hace tomando muestras por el canal del ADC1 que corresponda, y en el intervalo de tiempo preestablecido. Si se quiere mantener o establecer una frecuencia constante de muestreo existen dos aproximaciones. La primera aproximación se realiza en un bucle de n muestras tal como se muestra a continuación en el Código 1.

```
mientras ( _numero_de_muestras < _total_de_muestras ) {
      obtener los microsegundos;
      esperar a que transcurra el periodo de muestreo;
      tomar el valor del ADC;
      almacenar el valor obtenido;
      incrementar el número de muestras;
}
```

Código 1: Algoritmo de muestreo por bucle

La segunda forma es usando las interrupciones tal como se muestra a continuación en el Código 2.

```
rutina_de_atendimiento () {
      si ( _numero_de_muestras >= _total_de_muestras ) {
            parar_interrupcion();
      } sino {
            tomar el valor del ADC;
            almacenar el valor obtenido;
            incrementar el número de muestras;
      }
}
```

Código 2: Algoritmo de muestreo por interrupción

Dependiendo de lo que se necesite hacer durante el proceso de muestreo, hay que tener en cuenta que la forma mostrada en el primer pseudocódigo, Código 1, es un algoritmo bloqueante, lo que implicará tener al sistema ocupado sin posibilidad de hacer otra cosa hasta terminar de realizar la operación por completo, además, si durante el proceso de adquisición se produjese una interrupción esto podría provocar

una pérdida de sincronización del muestreo. Al usar el muestreo por interrupciones es menos probable que ocurra el problema comentado anteriormente. Hay que tener en cuenta las prioridades de la interrupción a la hora de programar para evitar que la adquisición de las muestras sea interrumpida innecesariamente.

La adquisición del audio se realiza bajo demanda, ya sea por parte de un usuario que esté conectado directamente al SoC ESP32, como pudiera ser el servidor quien solicite dichos datos. En la Figura 18 mostrada a continuación, se presenta el diagrama que seguirá el sistema para iniciar el proceso de adquisición y almacenamiento de audio.

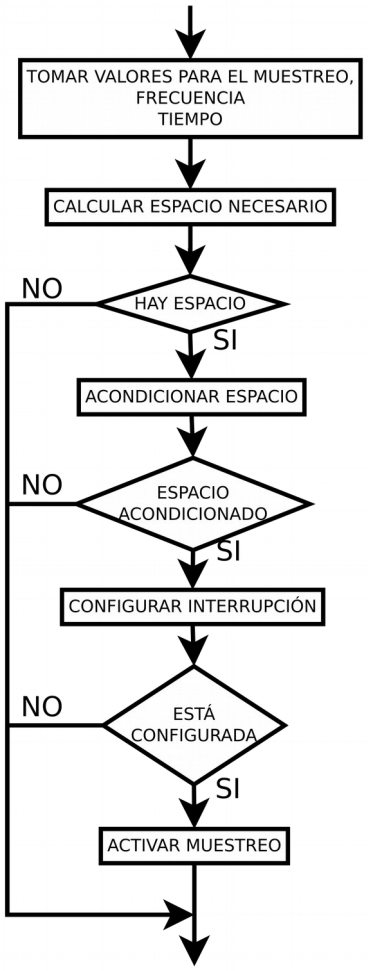


Figura 18: Diagrama de la adquisición y almacenamiento de audio

El sistema está continuamente analizando si hay alguna solicitud de muestreo, mientras no exista el sistema no hará nada. Al recibir una petición de muestreo se ejecuta un bloque de código encargado de asegurar la existencia de las variables

necesarias para realizar el muestreo y en su caso se encargará de acondicionar los datos para que puedan ser tratados en la siguiente etapa.

En esta etapa se procede al cálculo del tamaño de almacenamiento necesario el cual depende de: la frecuencia de muestreo, del tiempo de muestreo y de la resolución de la muestra; para nuestro caso se fija a 12 bits que es la mayor resolución del ADC; y del tamaño de la cabecera. Debido al empleo del formato wav la longitud de la cabecera tiene una tamaño de 44 bytes. Una vez calculado el tamaño de almacenamiento se comprueba que dicho espacio esté disponible. Si no hay suficiente espacio se aborta la operación y se regresa a la etapa inicial.

Conocido el tamaño y que hay espacio disponible, en esta etapa se hace la reserva de dicho espacio para el almacenamiento y se pasa a la siguiente etapa. El espacio reservado es un array de tipo char.

Se procede con la configuración de la interrupción que se encargará de realizar el muestreo. Para ello, hay que tener en cuenta la velocidad de muestreo solicitada, inicialmente el sistema tendrá el timer de la interrupción configurado con: un preescaler de 8 para generar una interrupción a cada micro segundo; estará activado en modo contador ascendente con reload. Con los datos anteriores se procede a calcular el número de cuentas que debe hacer el timer para llamar a la rutina de atendimiento para el muestreo. Es importante tener en cuenta que hemos limitados el tiempo mínimo entre interrupciones a un micro segundo, de dar un valor por de bajo de este debemos abortar la operación. Según el valor obtenido anteriormente se configura el valor del conteo del timer, quedando esta configurada para comenzar la adquisición.

El la siguiente etapa se realiza el almacenamiento de la cabecera correspondiente al protocolo wav y se activa el timer. Con la activación del timer comienza el proceso de adquisición.

Durante el proceso de adquisición se realizar el almacenamiento de las muestras. Para ello, se debe tener en cuenta que el espacio reservado es un array de tipo char y la adquisición se realiza sobre 12 bits de resolución. Para almacenar 12 bits necesitamos 1.5 byte con lo que será necesario almacenar cada muestras en 2 bytes.

El proceso de adquisición se ejecutará hasta que el espacio de almacenamiento reservado anteriormente esté lleno, esto indicará que ya tenemos la muestra solicitada con lo que pasamos al bloque encargado de desactivar la interrupción y activar los

flags correspondientes para avisar de que ya está la muestra y así poder regresar a la etapa inicial.

Para comprobar el correcto funcionamiento de sistema de adquisición serán realizadas varias pruebas haciendo uso de tonos conocidos. Para usar dichos tonos se hace uso de la app Generador de Frecuencias[20] desde un dispositivo móvil Android. De esta forma se podrán contrastar los resultado y estudiar la viabilidad del sistema.

3.4 Diseño de un portal web sobre el ESP32

El subsistema ESP32 incorpora un módulo para la gestión wifi. Este módulo implementa la pila de protocolos TCP y permite al subsistema conectarse a una red de forma inalámbrica. Además, permite al subsistema trabajar en modo cliente-servidor.

Para poder trabajar con el módulo de comunicaciones del subsistema, contamos con una serie de librerías. Estas librerías nos permiten, indicando parámetros como: ssid de la red, tipo de conexión, password, etc; conectar a una red inalámbrica. Las librerías nos permiten poner al subsistema bien en modo AP (AP, *Access Point*) o bien en modo STA (STA, *Station*). Haciendo uso de estas librerías podemos implementar la funcionalidad de un servidor web sencillo, que nos permita interactuar con funciones básicas del propio subsistema. Las librerías permiten realizar la comunicación por TCP y por UDP. El UDP es un protocolo no orientado a conexión, siendo la comunicación con el extremo unidireccional y no lleva confirmación de ningún tipo, no es fiable y no garantiza la recepción de datos. El TCP está orientado a conexión, la comunicación con el extremo está sujeta a un control de calidad, es fiable y asegura la integridad en la recepción de los datos.

De las librerías usadas podemos destacar:

- **ESP32WebServer.** Gestiona el protocolo HTTP, ofrece una serie de métodos con los cuales podemos atender las peticiones externas, ya sean de un terminal cliente o desde un servidor externo.
- **Wifi.** Gestiona la pila del protocolo de comunicaciones TCP, nos facilita la tarea de realizar la conexión a una red.
- **WifiCliente.** Gestiona la comunicación en modo cliente, y será usada para comunicar el subsistema con el servidor destino.

El sistema se inicia en tres fases, tal como se muestra en la Figura 19. En primer lugar el subsistema realiza el intento de conectar con la red wifi, para ello hace uso de los parámetros de configuración previamente establecidos. Una vez conectado a la red, la segunda fase consiste en la activación del servicio web. Para finalizar, a partir de la dirección ip, obtenida durante la conexión, se determina cual es la dirección de red, para realizar la identificación con el servidor externo, el cual se espera que haya sido configurado como el host 100.

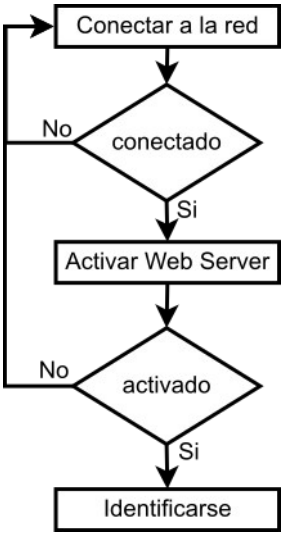


Figura 19: Inicio del subsistemas ESP32

El servidor destino es una máquina donde se centraliza la recopilación de los datos desde todos los subsistemas que hayan en la misma red. Esto permite a un usuario, conectado al servidor destino, trabajar con todos los subsistemas que haya en la red de forma centralizada.

Capítulo 4. Desarrollo del firmware del ESP32

4.1 Análisis y diseño de bloques

Como se describe en el apartado 1.3, los subsistemas ESP32 se dividen en 4 bloques que serán implementados por medio de clases y librerías:

- Bloque de Captura. Se implementa con dos clases:
  - **Adc**, configura y gestiona las entradas analógicas del subsistema.
  - **Microfono**, implementa la gestión de las interrupciones, adquisición de las muestras y adecuación de los datos.
- Bloque de Interfaz Web. Se implementa con dos clases:
  - **Wifi**, gestiona la conexión a la red.
  - **Web**, gestiona lo referente al protocolo HTTP.
- Bloque de Procesado. Se implementa por medio de cuatro clases:

- **Enventanado**, gestiona la creación y aplicación de los diferentes enventanados.
- **Complejo**, maneja lo referente a los número complejos.
- **Mariposa**, implementa el algoritmo para el cálculo de la FFT.
- **Tono**, esta clase gestiona el cálculo del tono de la FFT.

- Bloque de Datos. Se implementa con una clase:
  - **TablaDeDatos**, gestiona el espacio de memoria.

4.2 Estudio y Diseño del bloque de captura

La clase **Adc**, que permite la configuración y uso del las entradas analógica y su conversor analógico-digital, tiene la siguiente estructura:

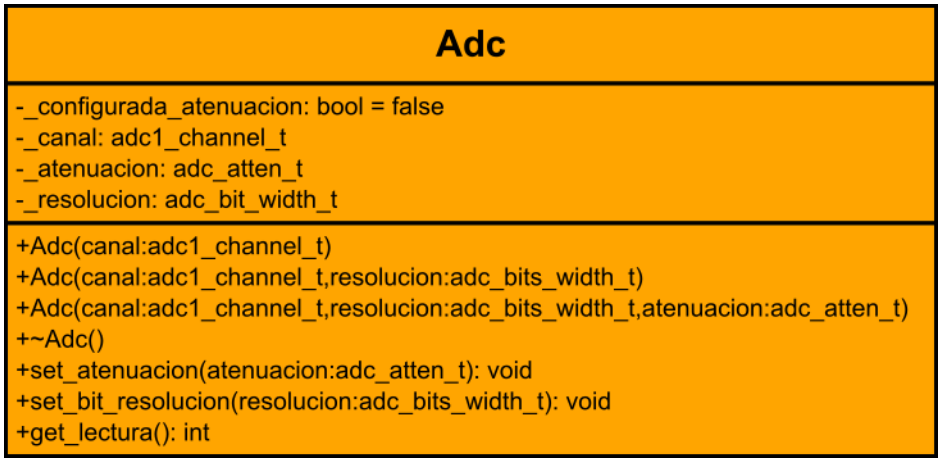


Figura 20: Diagrama UML de la clase Adc

Sus atributos son:

- `_configurada_atenuacion`, indica si la atenuación ha sido configurada.
- `_canal`, canal o pin de entrada utilizados.
- `_atenuacion`, valor de la atenuación configurada.
- `_resolucion`, valor de la resolución configurada.

Sus métodos son:

- `Adc`, constructor, inicializa el objeto con los valores predeterminado, y tiene los siguiente parámetros: canal o el pin de entrada.

- `Adc`, constructor, inicializa el objeto con los valores predeterminado, y tiene los siguiente parámetros: canal o el pin de entrada; resolución o la cantidad de bits que se usaran para tomar los valores.
- `Adc`, constructor, inicializa el objeto con los valores predeterminado, y tiene los siguiente parámetros: canal o el pin de entrada; resolución o la cantidad de bits que se usaran para tomar los valores; y atenuación o valor de atenuación para la referencia en la toma de valores.
- `set_atenuacion`, coloca la atenuación para la referencia para la toma de valores al valor pasado por parámetros.
- `set_bit_resolucion`, establece la resolución del convertidor analógico-digital a la resolución o cantidad de bits que se usaran para tomar los valores.
- `get_lectura`, devuelve un entero correspondiente a la última conversión realizada de la entrada analógica.

La clase **Microfono**, que permite la configuración de las interrupciones, realiza el proceso de muestreo y crea la estructura de los archivos de audio y tiene la siguiente estructura.

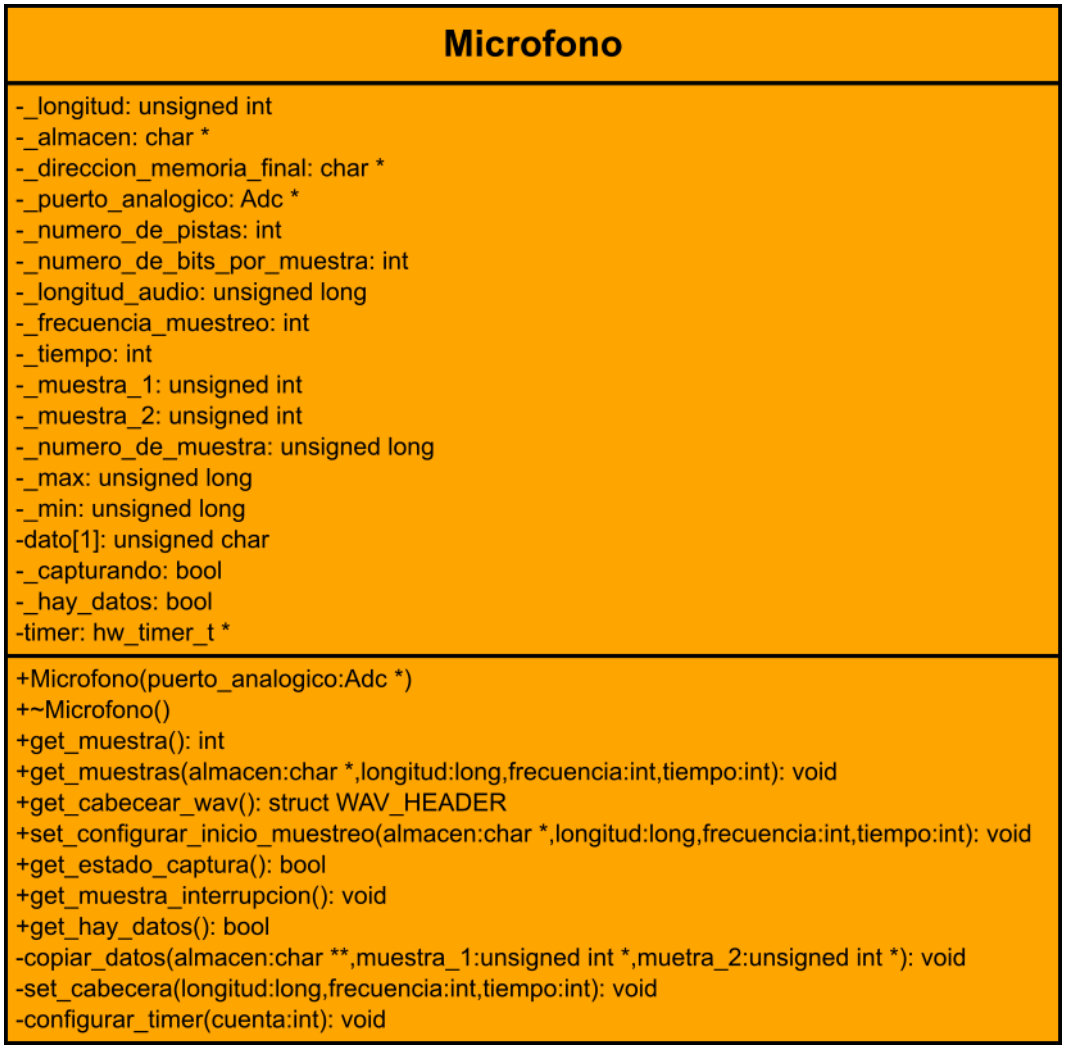


Figura 21: Diagrama UML de la clase Micrófono

Sus atributos son:

- `_longitud`, longitud del bloque de memoria destinado al almacenamiento de las muestras.
- `_almacen`, dirección al bloque de memoria donde se alojan las muestras.
- `direccion_memoria_final`, dirección al final del bloque de memoria donde se alojarán las muestras.
- `_puerto_analogico`, puntero al objeto Adc.
- `_numero_de_pistas`, número de pistas para generar la cabecera wav.
- `_numero_de_bits_por_muestra`, cantidad de bit por muestra.
- `_longitud_audio`, longitud del audio muestreado.

- `_frecuencia_muestreo`, la frecuencia de muestreo.
- `_tiempo`, el tiempo de muestreo.
- `_muestra_1`, valor de la muestra impar.
- `_muestra_2`, valor de la muestra par.
- `_numero_de_muestra`, número de muestras a tomar.
- `_max`, valor máximo de todas las muestras.
- `_min`, valor mínimo de todas las muestras.
- `dato[1]`, espacio usado para concatenar el valor de las dos muestras antes de almacenarlo en el bloque de memoria.
- `_capturando`, indica que aun no ha finalizado el proceso de captura.
- `_hay_datos`, indica si hay datos en el bloque de memoria actual.
- `timer`, objeto de control de la interrupción.

Sus métodos son:

- `Microfono`, constructor, inicializa el objeto con los valores predeterminado, y el objeto de control Adc con el pasado por parámetro.
- `get_muestra`, solicita al objeto Adc una muestra de la entrada analógica y devuelve el valor obtenido.
- `get_muestras`, realiza el muestreo de forma bloqueante y tiene los siguiente parámetros: dirección al bloque de memoria destinado a las muestras; longitud del bloque de memoria; frecuencia de muestreo; tiempo de muestreo.
- `get_cabecera_wav`, devuelva la estructura correspondiente a la cabecera del fichero wav.
- `set_configurar_inicio_muestreo`, configura la interrupción y los datos iniciales para realizar el muestreo, tiene los siguientes parámetros: dirección al bloque de memoria para alojar las muestras; longitud del bloque de memoria; frecuencia de muestreo; tiempo de muestreo.
- `get_estado_captura`, devuelve el estado del muestreo.
- `get_muestra_interrupcion`, rutina de interrupción del servicio de muestreo.

- `get_hay_datos`, devuelve si hay datos almacenados.
- `copiar_datos`, almacena dos muestras en 3 bytes, tiene los siguientes parámetros: dirección al bloque de memoria para alojar las muestras; valor de la primera muestra; valor de la segunda muestra.
- `set_cabecera`, crea la cabecera del fichero wav, con los valore de longitud, frecuencia y tiempo, pasados por parámetros.
- `configurar_timer`, configura e inicializa la interrupción encargada del muestreo, tiene como parámetro el valor del conteo para el temporizador.

4.3 Estudio y Diseño del bloque de interfaz web

Para el estudio y diseño del bloque de interfaz web son necesarias las clases: **Wifi** y **Web**.

La clase **Wifi**, que permite la configuración y uso de la conexión de red soportada en la clase **WiFiClass** de la empresa Espressif, cuenta con la siguiente estructura.

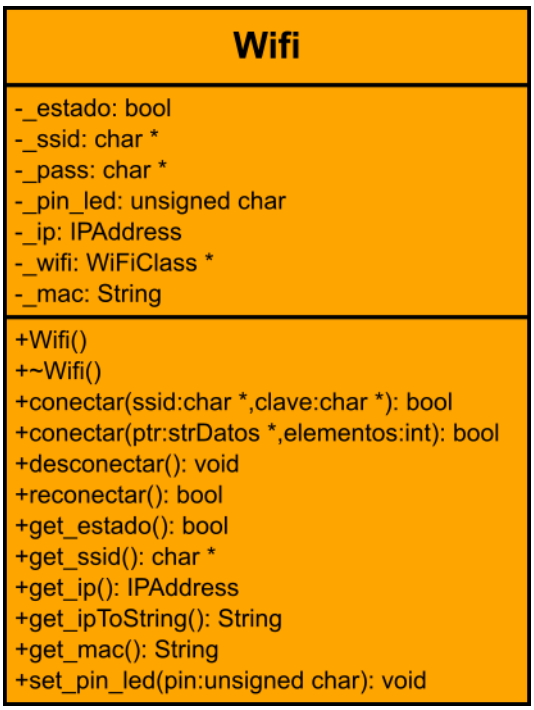


Figura 22: Diagrama UML de la clase Wifi

Sus atributos son:

- `_estado`, almacena el estado de la conexión con la red.
- `_ssid`, almacena el ssid de la conexión actual.
- `_pass`, almacena la clave de la conexión actual.
- `_pin_led`, almacena el pin usado para indicar el estado de la conexión.
- `_ip`, almacena la dirección ip obtenida en la conexión.
- `_wifi`, objeto de control de tipo WiFiClass.
- `_mac`, almacena el valor de la mac.

Su métodos son:

- `Wifi`, constructor, inicializa el objeto con los valores predeterminado, crea una instancia del objeto WiFiClass.
- `conectar`, intenta la conexión wifi contra el punto de acceso cuyo ssid y clave son pasados por parámetros y devuelve el estado de la conexión.
- `conectar`, intenta la conexión wifi contra alguno de los puntos de acceso pasados por parámetro en una estructura del tipo strDatos que contiene parejas de valores de ssid-clave, y devuelve el estado de la conexión.
- `desconectar`, cierra la conexión wifi.
- `reconectar`, vuelve a conectar con la última wifi conocida.
- `get_estado`, devuelve el estado de la conexión.
- `get_ssid`, devuelve el ssid de la conexión actual.
- `get_ip`, devuelve la dirección ip en formato IPAddress.
- `get_ipToString`, devuelve la dirección ip en formato string.
- `get_mac`, devuelve la dirección mac en un formato string.
- `set_pin_led`, establece el pin de salida que indicará el estado de la conexión, y que puede ser usado para alimentar un led de indicación.

La clase **Web**, que permite la gestión del protocolo HTTP, está implementada en la librería ESP32WebServer de Ivan Grokhotkov. La clase ESP32WebServer original fue modificada debido a errores en su implementación, cuenta con la siguiente estructura:

Figura 23: Diagrama UML de la clase Web

Sus atributos son:

- `_puerto`, puerto usado para el servicio web.
- `_wifi`, objeto para la gestión de la conexión con la red.
- `_microfono`, objeto para la gestión y adquisición del muestreo.
- `_pin_led_procesando`, pin usado para indicar el estado del proceso.
- `_pin_led_enviando`, pin usado para indicar el estado del envío.
- `_servidor`, objeto para la gestión y control del servicio web.
- `_cliente`, objeto para hacer la comunicación, usando el protocolo HTTP, en modo cliente.
- `_respuesta`, última respuesta recibida.
- `_delimitador`, carácter usado para delimitar los campos del JSon.
- `_esperando_captura`; indica el estado del proceso de muestreo.
- `_realizando_proceso`, indica el estado del procesado.
- `_hay_datos_para_descargar`, indica si hay datos listos para descargar.
- `_refrescando`, estado del servicio web
- `bloque_de_memoria`, bloque de memoria destinado a almacenar las muestras obtenidas en el proceso de muestreo.
- `_longitud_audio`, longitud del audio almacenado.
- `datos`, objeto de control y gestión de la memoria.
- `_solicitud`, estructura donde se almacenan los datos de la solicitud de procesado.
- `_almacen`, estructura con la dirección de memoria de los bloque usados en el cálculo de la FFT.
- `_tono`, objeto para el cálculo del tono.
- `_mariposa`, objeto para el cálculo de la FFT.

Sus métodos son:

- `Web`, constructor, realiza la configuración básica para la gestión web, crea una instancia de los objeto: ESP32WebServer; WiFiClient. Tiene los siguiente parámetros: puerto utilizado para el servicio web; objeto para el control y gestión de la comunicación con la red; objeto para el control y gestión del proceso de muestreo.
- `available`, comprueba la existencia de una petición o si hay algún proceso pendiente.
- `identificarse`, envía la trama de identificación al destino para se registrado y tiene los siguiente parámetros: dirección del destino; puerto del destino.
- `set_pin_led_procesando`, establece el pin de salida que indicará el estado del proceso, y que puede ser usado para alimentar un led de indicación.
- `set_pin_led_enviando`, establece el pin de salida que indicará el estado del envio, y que puede ser usado para alimentar un led de indicación.
- `home`, procesa las peticiones.
- `keep_alive`, envía al destino una confirmación para indicar que sigue activo.
- `enviar_datos_api`, envía los datos indicados al destino, tiene los siguientes parámetros: dirección del destino; puerto del destino; servicio solicitado; método a ejecutar en el servicio; los datos a enviar.
- `gestiona_respuesta_json_contar_campos`, devuelve la cantidad de parejas campo-valor hay en la data pasada por parámetros.
- `gestiona_respuesta_json`, devuelve un array de string, en el destino pasado por parámetros, con el contenido de cada pareja campo-valor obtenidos de la data pasada por parámetros.
- `error`, envía al destino un cadena JSon informando del error.
- `lectura_variable_post`, inicializa los valores según los datos recibidos en la petición por post.
- `favicon`, envía el favicon cuando el navegador los solicita.
- `foto`, envía la imagen de la ULPGC cuando el navegador la solicita.
- `muestra_audio`, realiza el muestreo y procesado según lo indicado en la petición.

- `calcular_fft`, calcula la FFT según lo indicado en la petición.
- `calcular_tono`, calcula el tono que corresponde en la FFT actual.
- `descargar_datos_memoria`, envía al destino los datos almacenados en memoria en formato wav.
- `liberar_memoria`, libera la memoria usada por la FFT.
- `sample`, informa al destino que se ha iniciado el proceso de muestreo.
- `es_potencia_de_dos`, devuelve si el valor pasado por parámetros es potencia de dos.
- `enviar_json`, envía la cadena, con formato JSon, pasa por parámetros al destino.
- `enviar_muestras_muestreo`, abre conexión en modo cliente con el destino y envía el resultado del muestreo.
- `enviar_muestras_fft`, abre conexión en modo cliente con el destino y envía el resultado de la FFT.
- `enviar_muestras_tono`, abre conexión en modo cliente con el destino y envía el resultado del tono.
- `indicar_procesando`, establece el valor pasado por parámetros sobe el pin configurado en `set_pin_led_procesando`.
- `indicar_enviando`, establece el valor pasado por parámetros sobre el pin configurado en `set_pin_led_enviando`.

4.4 Validación del bloque de muestreo

Como se comentó en el apartado 3.3, para la validación del subsistemas se hará uso de tonos conocidos, estos tonos serán generados con una aplicación de Android llamada Generador de Frecuencias de Hoel Boedec. Para la validación se generan los siguientes tonos: 500 Hz; 1000 Hz; 2000 Hz.

Tal como se dijo en el apartado 1.3, se hará uso de la interfaz web del subsistema para realizar las pruebas. La interfaz se muestra en la Figura 24, permite seleccionar una frecuencia para el muestreo y el tiempo de muestreo.

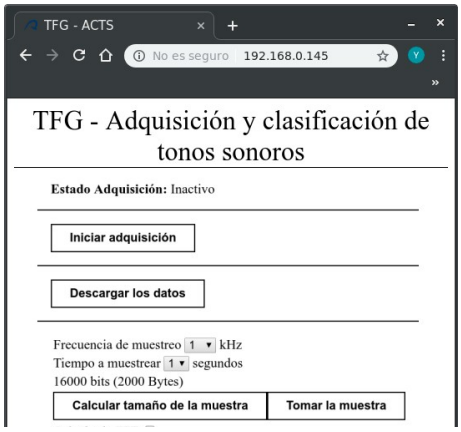


Figura 24: Interfaz web del subsistema ESP32

El subsistema, mostrado en la Figura 2, debe acondicionar los datos de la petición para poder realizar la captura durante el tiempo y frecuencia solicitados, y enviar los datos al destino.

Al descargar los datos, estos pueden ser reproducidos, o utilizar el programa Audacity, que nos permite ver la señal en el tiempo y obtener su espectro, se realizará la validación del archivo recibido.

Se realiza el muestreo y descarga de los archivo de audio generados por el subsistema y se comprueba su correcto funcionamiento.

4.5 Desarrollo de algoritmos básico para el procesamiento de audio sobre el ESP32

Tal como se comentó en el apartado 4.3, la clase **Web** es la que gestiona, además de toda la comunicación usando el protocolo HTTP, el muestreo así como los cálculos de la FFT, el enventanado y el cálculo del tono principal. Para el cálculo de la FFT se hace uso de la clase **Mariposa** que a su vez hace uso de la clase **Complejo**, además para realizar los diferentes enventanados se hace uso de la clase **Enventanado**, destacando que han sido implementadas por nosotros.

Las clase **Web** cuenta con un objeto de la clase Enventanado. tal como se comentó en el apartado 4.1, esta clase se encarga de crear y aplicar los diferentes enventanados, y cuenta con la siguiente estructura:

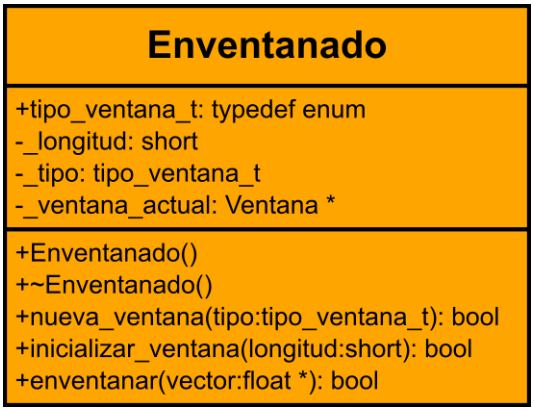


Figura 25: Diagrama UML de la clase Enventanado

Sus atributos son:

- `tipo_ventana_t`, enumerado con los diferentes tipos de ventana disponibles. Tiene los siguientes valores: RECTANGULAR, HAMMING, HANNIG, BLACKMAN, BLACKMAN_HARRIS y BLACKMAN_NUTTALL.
- `_longitud`, almacena la longitud del último enventanado.
- `_tipo`, almacena el último tipo de ventana que se uso.
- `_ventana_actual`, almacena la ventana que se está usando.

Sus métodos son:

- `Enventanado`, constructor de la clase, inicializa las variables para un correcto funcionamiento.
- ~`Enventanado`, destructor, libera la memoria usada por la clase.
- `nueva_ventana`, crea un objeto del tipo indicado por parámetros, devuelve si la ventana fue creada o no.
- `inicializar_ventana`, se inicializan los valores de la ventana según la longitud indicada por parámetros.
- `enventanar`, toma el vector pasado por parámetros y aplica el enventanado con la ventana actual.

Para asegurar el correcto funcionamiento de las diferentes clases que definen una ventana, se ha implementado la interfaz **Ventana** la cual tiene la siguiente estructura:

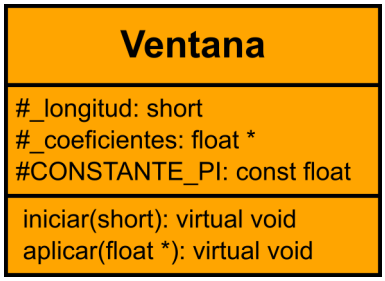


Figura 26: Diagrama UML de la interfaz Ventana

Sus atributos son:

- `_longitud`, almacena la longitud de la ventana.
- `_coeficientes`, puntero al bloque de memoria que contiene los coeficientes de la ventana.
- `CONSTANTE_PI`, constante con el valor de pi.

Sus métodos son:

- `iniciar`, inicializa el vector de coeficientes con los valores correspondientes según la longitud pasada por parámetros y la/s fórmula/s que definan dicha ventana.
- `aplicar`, aplica el enventanado al vector pasado por parámetros.

La clase **Mariposa** cuenta con un objeto de la clase **Complejo**. Tal como se comentó en el apartado 4.1, dicha clase se encarga del manejo de los número complejos, y cuenta con la siguiente estructura:

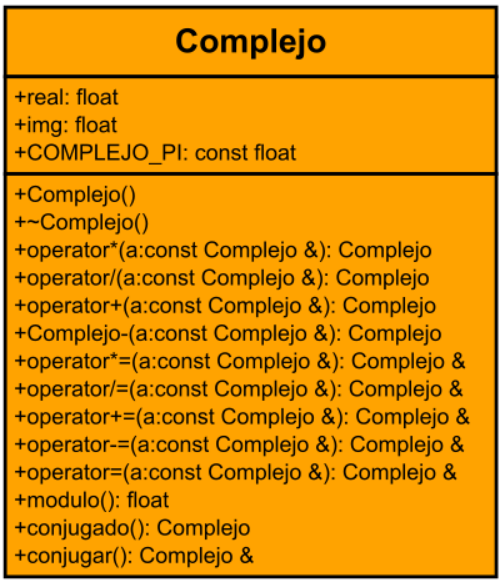


Figura 27: Diagrama UML de la clase Complejo

Sus atributos son:

- `real`, almacena la parte real del número complejo.
- `img`, almacena la parte imaginaria del número complejo.
- `COMPLEJO_PI`, constante con el valor de pi.

Sus métodos son:

- `Complejo`, constructor.
- `~Complejo`, destructor.
- `operator`*, devuelve el producto de los dos complejos.
- `operator`/, devuelve la división de los dos complejos.
- `operator`+, devuelve la suma de los dos complejos.
- `operator`-, devuelve la resta de los dos complejos.
- `operator`*=, modifica el valor del complejo principal con el valor resultante del producto entre los dos complejos.
- `operator`/=, modifica el valor del complejo principal con el valor resultante de la división entre los dos complejos
- `operator`+=, modifica el valor del complejo principal con el valor resultante de la suma de los dos complejos

- `operator-=`, modifica el valor del complejo principal con el valor resultante de la resta de los dos complejos
- `modulo`, devuelve el módulo del complejo.
- `conjugado`, devuelve el conjugado del complejo.
- `conjugar`, modifica el valor del complejo por su conjugado.

Las clase **Web** cuenta con un objeto de la clase **Mariposa**. Tal como se comentó en el apartado 4.1, dicha clase implementa el algoritmo para el cálculo de la FFT, y cuenta con la siguiente estructura:

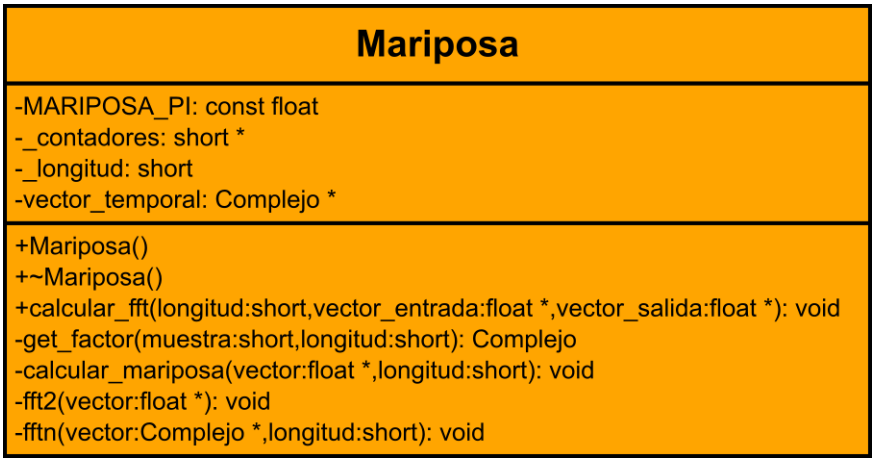


Figura 28: Diagrama UML de la clase Mariposa

Sus atributos son:

- `MARIPOSA_PI`, constante con el valor de pi.
- `_contadores`, vector de contadores para la adecuación del cálculo en la mariposa.
- `_longitud`, almacena la longitud de la mariposa.
- `vector_temporal`, puntero al buffer destinado al almacenamiento de los valores obtenidos durante el proceso de cálculo.

Sus métodos son:

- `Mariposa`, constructor, inicializar los valores por defecto.
- `~Mariposa`, destructor.
- `calcular_fft`, inicializa el espacio de trabajo para el cálculo de la FFT.

- `get_factor`, calcula y devuelve el valor del factor de giro que corresponda. Tiene los siguientes parámetros: número de la muestra que se está procesando; longitud de la mariposa en proceso de cálculo.
- `calcular_mariposa`, realizar el proceso de adecuación de los datos para las distintas ramas de la mariposa, tiene los siguiente parámetros: puntero al buffer con los datos a procesar; longitud de la FFT.
- `fft2`, inicia el cálculo de la FFT. Este toma las parejas formadas por la unidad mínima de cálculo (la unidad mínima está compuesta por 2 valores), aplica la formulación correspondiente y ubica los resultados en el buffer temporal.
- `fftn`, realizar el cálculo de la FFT usando los datos actuales almacenados en el buffer temporal con el vector de complejos y longitud recibidos por parámetros. El buffer temporal es reutilizado para ubicar, nuevamente, los resultados.

La clase **Web** cuenta con un objeto de la clase **TablaDeDatos**, como se comentó en el apartado 4.1, la clase **TablaDeDatos** se encarga de la gestión de memoria para el muestreo y cuenta con la siguiente estructura:

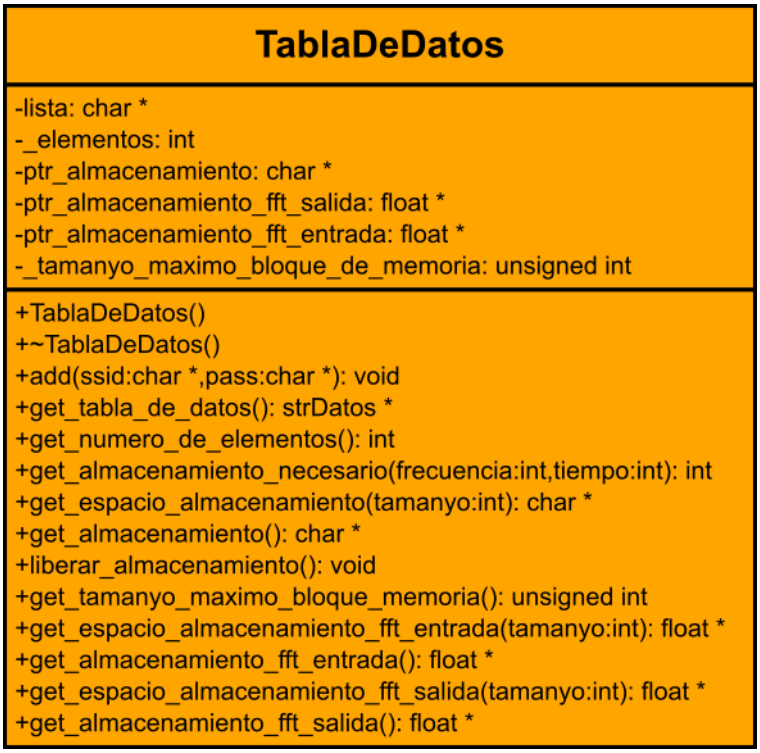


Figura 29: Diagrama UML de la clase TablaDeDatos

Sus atributos son:

- `lista`, almacena el puntero al bloque donde se almacenan las parejas ssid-pass de la lista de las posibles conexiones.
- `_elementos`, indica el número de parejas ssid-pass hay almacenadas.
- `ptr_almacenamiento`, almacena el puntero al bloque de memoria destinado al muestreo.
- `ptr_almacenamiento_fft_salida`, almacena el puntero al bloque de memoria destinado a los resultados de la FFT.
- `ptr_almacenamiento_fft_entrada`, almacena el puntero al bloque de memoria destinado a las muestras a procesar por la FFT.
- `_tamanyo_maximo_bloque_de_memoria`, indica el tamaño máximo con el cual podemos obtener un bloque de memoria.

Sus métodos son:

- `TablaDeDatos`, constructor, inicializa el objeto con los valores predeterminados.
- `add`, añade a la lista de posibles conexiones el ssid y el pass pasados por parámetros.
- `get_tabla_de_datos`, devuelve un puntero al bloque de memoria donde se alojan las parejas ssid-pass de la lista de posibles conexiones.
- `get_numero_de_elementos`, devuelve el número de parejas ssid-pass hay almacenadas.
- `get_almacenamiento_necesario`, calcula el tamaño del bloque de memoria necesario para el muestreo, tiene los siguiente parámetros: frecuencia de muestreo; tiempo de muestreo.
- `get_espacio_almacenamiento`, toma el bloque de memoria indicado por parámetros y devuelve la dirección de memoria de este.
- `get_almacenamiento`, devuelve el puntero al bloque de memoria destinado al muestreo.
- `liberar_almacenamiento`, libera la memoria que se ha usado.
- `get_tamanyo_maximo_bloque_memoria`, almacena el valor de tamaño máximo con el que se puede obtener un bloque de memoria.

- `get_espacio_almacenamiento_fft_entrada`, toma el tamaño de memoria indicado por parámetros y devuelve el puntero a dicho bloque.
- `get_almacenamiento_fft_entrada`, devuelve la dirección al bloque de memoria destinado a almacenar las muestras a procesar por la FFT.
- `get_espacio_almacenamiento_fft_salida`, toma el tamaño de memoria indicado por parámetros y devuelve el puntero a dicho bloque.
- `get_almacenamiento_fft_salida`, devuelve la dirección al bloque de memoria destinado a almacenar los resultados de la FFT.

La clase **Tono** se encarga del calculo del tono de la FFT actual y cuenta con la siguiente estructura:

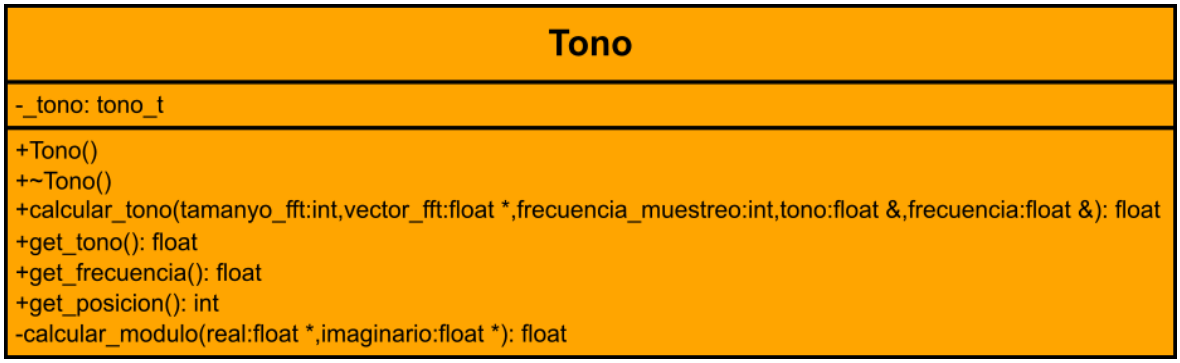


Figura 30: Diagrama UML de la clase Tono

Sus atributos son:

- `_tono`, modelo de datos para almacenar la información del resultado sobre el cálculo.

Sus métodos son:

- `Tono`, constructor, inicializa el objeto con los valores predeterminados.
- `calcular_tono`, busca el tono fundamental, calcula la frecuencia a la que se corresponde, almacena los valores y devuelve el tono y la frecuencia. Tiene los siguientes parámetros: tamaño del bloque de memoria con los resultados de la FFT; puntero al bloque de memoria con los resultados de la FFT; frecuencia a la que se realizado el muestreo; dirección de memoria para almacenar el tono calculado; dirección de memoria para almacenar la frecuencia correspondiente al tono calculado.
- `get_tono`, devuelve el valor del último tono calculado.

- `get_frecuencia`, devuelve el valor de la frecuencia correspondiente al último tono calculado.
- `get_posicion`, devuelve la posición en la que se encontró el tono.
- `calcular_modulo`, devuelve el módulo correspondiente a la parte real e imaginaria pasadas por parámetros.

Dado los recursos escasos de nuestro subsistema en la Tabla 4 se relacionan tamaño de la fft con sus requerimientos de espacio de memoria.

Tamaño FFT	Entrada	Salida	Vector Temporal	**Total (byte)**
32	128	256	256	**640**
64	256	512	512	**1280**
128	512	1024	1024	**2560**
256	1024	2048	2048	**5120**
512	2048	4096	4096	**10240**
1024	4096	8192	8192	**20480**

Tabla 4: Relación del tamaño de la FFT con la memoria necesaria.

En la Tabla 5 se muestran algunas relaciones según el tamaño de la ventana FFT y el tamaño de la muestra con el espacio necesario de memoria, con un enventanado contiguo.

Tamaño FFT	Numero de muestras	Total FFT (byte)	Nº ventanas	**Total Memoria (byte)**
32	500	640	16	**10320**
32	1000	640	31	**20000**
64	500	1280	8	**10448**
64	1000	1280	16	**20640**
128	500	2560	4	**10704**
128	1000	2560	8	**20896**
256	500	5120	2	**11216**
256	1000	5120	4	**21408**
512	500	10240	1	**12240**
512	1000	10240	2	**22432**
.	.	.	.	**.**
256	10000	5120	39	**200768**
512	10000	10240	20	**205888**

Tabla 5: Memoria necesaria sin solapamiento.

De la tabla anterior se deduce que de hacer un solapamiento en las ventanas para el cálculo de las FFT el espacio de memoria necesario será mayor. En la Tabla 6 se muestra la relación según el tamaño de la ventana FFT y el tamaño de la muestra con el espacio de memoria necesario, para un enventanado con solapamiento del 50%.

Tamaño FFT	Numero de muestras	Total FFT (byte)	Nº ventanas	**Total Memoria (byte)**
32	500	640	31	**18000**
32	1000	640	63	**36384**
64	500	1280	16	**18640**
64	1000	1280	31	**36000**
128	500	2560	8	**18896**
128	1000	2560	16	**37280**
256	500	5120	4	**19408**
256	1000	5120	8	**37792**
512	500	10240	2	**20432**
512	1000	10240	4	**38816**

Tabla 6: Memoria necesaria con solapamiento del 50%.

De las tablas anteriores, podemos deducir que no se podrá almacenar la totalidad de las FFTs, junto al vector de muestras capturadas.

Suponiendo que se quiere hacer un muestreo a 8000 Hz, teniendo en cuenta que el conversor analógico digital es de 12 bits necesitamos dos bytes por muestra, entonces necesitamos 16000 bytes para almacenar las muestras, para poder añadir la cabecera del formato wav hay que aumentar 44 bytes para el almacenamiento. Hasta aquí tenemos la memoria necesaria para almacenar el fichero wav con el contenido del proceso de muestreo. Supongamos que queremos realizar una FFT de 1024, como a la entrada solo tenemos valores reales necesitamos un buffer de tipo float de 1024 posiciones, esto requiere una capacidad de memoria de 4096 bytes. El cálculo de la FFT da como resultado número complejos, tiene parte real y parte imaginaria, eso implica que necesitamos un buffer de tipo float de 2048 posiciones, esto requiere una capacidad de memoria de 8192 bytes. Los cálculos de requerimiento de memoria en la FFT son para una única muestra, para el caso de 8000 muestras tendremos, aplicando enventanado contiguo, 7 ventanas distintas lo que implica que el requerimiento de capacidad de memoria para las muestras de entrada es de 28672 bytes y para la salida es de 57344 bytes. Si sumamos las capacidades de memoria requerida tenemos una necesidad de almacenamiento de al menos 102060 bytes. La suposición ha sido calculada para una muestra de 1 segundo, si tomamos 2 segundo de muestra

necesitaríamos una capacidad de almacenamiento de al menos 216364 bytes, capacidad que no tenemos.

Es por lo comentado anteriormente que después de cada FFT calculada deberá ser enviado al servidor el resultado de esta antes de comenzar a realizar una nueva FFT con los valores de la siguiente ventana sobre el vector de muestras. El procesado de la FFT y el tono se realiza de manera iterativa enviando los resultados al servidor. El algoritmo básico seguido es:

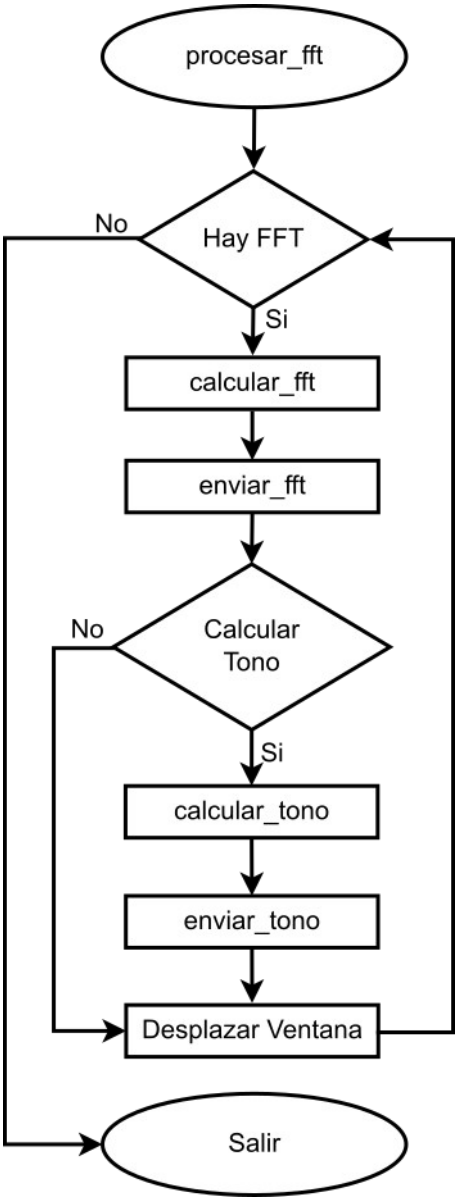


Figura 31: Procesado de la FFT y el Tono

4.5.1 Implementación de algoritmos

Para realizar una FFT básica es necesario dividir el conjunto de muestras obtenidos en el proceso de adquisición por nuestras limitaciones. El conjunto de muestras quedará definido por la longitud indicada para la FFT. Además, antes de realizar el calculo de la FFT es necesario aplicar algún tipo de ventana que ayude a reducir los efectos de las discontinuidades que se pueden dar al realizar el enventanado sobre una señal continua, infinita y no periódica. Finalmente se analiza el resultado obtenido en busca de la frecuencia correspondiente al tono principal, el cual queda definido por ser el tono de mayor amplitud.

Para proceder con el enventanado se cuenta con la clase **Enventanado**, la cuál cuenta con las siguiente funciones:

- `nueva_ventana`, crear la ventana según el tipo pasado por parámetros.
- `inicializar_ventana`, toma la ventana creada anteriormente y la inicializa con los coeficientes que correspondan. Los coeficientes dependerán del tipo de ventana indicada en su construcción.
- `enventanar`, toma el vector de muestras pasado por parámetros y le aplica la ventana.

Para proceder con el cálculo de la FFT se aplica el algoritmo de la mariposa el cuál se implementa, tal como se comento en el apartado 4.5, en la clase **Mariposa**. Esta clase cuenta con la siguiente función:

- `calcular_fft`, inicializa el espacio de trabajo, ejecuta el algoritmo de la mariposa y almacena el resultado en la salida. Tiene los siguientes parámetros: longitud de la FFT; puntero al buffer que contiene los datos a procesar; puntero al buffer destinado al almacenamiento de la salida.

Para obtener el tono principal de las muestras procesadas tenemos la clase **Tono**. Esta clase cuenta con la siguiente función:

- `calcular_tono`, toma el vector de los valores resultantes del procesado, calcula el módulo correspondiente a cada pareja de valores que forman un vector (parte real, parte imaginaria) y lo almacena en el mismo buffer. Durante el proceso de cálculo toma el valor de mayor amplitud y la posición que

corresponde a dicho valor para calcular la frecuencia a la que se corresponde. El valor del tono principal así como la frecuencia a la que se corresponde se devuelve por referencia. Tiene los siguientes parámetros: longitud de la FFT; puntero al buffer que contiene los datos procesados; frecuencia a la que se realizó el muestreo; referencia a la variable para almacenar el valor del tono; referencia a la variable para almacenar la frecuencia del tono.

4.5.2 Definición de pruebas

Para realizar las pruebas se hará uso de los tonos descritos en el apartado 4.4. Se realizará un muestreo a 8000 bps con un tiempo de muestreo de 1 segundo para cada tono. A partir del resultado obtenido en el muestreo se calculará la FFT, para las siguientes longitudes: 32; 64; 128, 256, 512 y 1024, con el correspondiente cálculo del tono. Por cada cálculo de FFT y tono enviaremos los resultados por consola para poder analizarlos.

4.5.3 Implementación y medidas

Siguiendo el escenario comentado en el apartado anterior, las pruebas se realizarán siguiendo el esquema mostrado en la Figura 32. Se inicia el proceso de muestreo, luego por cada valor de longitud para la FFT, se realiza el cálculo de la FFT mostrando el resultado por consola y se realiza el cálculo del tono que corresponde a la FFT calculada mostrando el resultado por la consola.

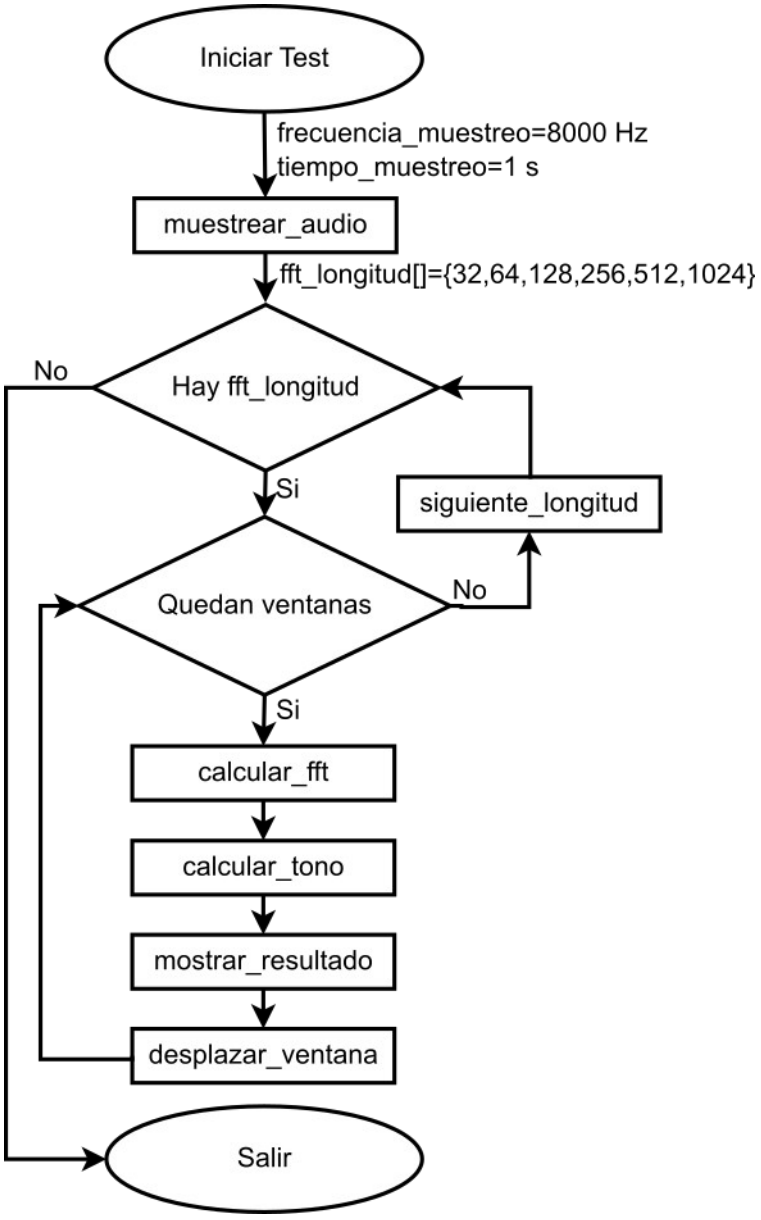


Figura 32: Proceso para las pruebas

A continuación se muestran algunos de los resultados obtenidos durante las pruebas. Se mostrarán los resultados correspondientes a la FFT de 32 para las frecuencias descritas anteriormente. El resto de las FFT correspondientes a 64, 128, 256, 512 y 1024 se adjuntan en el anexo debido a que los resultados son muy extensos para mostrarlos a continuación.

Para una FFT de 32

- Para una frecuencia de 500Hz obtenemos los resultados de la FFT mostrados en la Figura 33. En la Figura 34, en la imagen A se muestra el desarrollo para el cálculo del tono principal donde se puede ver que la última linea indica el tono principal en 500Hz, en la imagen B se presentar el espectrograma realizado, del fichero de audio obtenido, usando el programa Audacity.

```
x0: -2.039926528931+0.000000000000j
x1: -7.834179401398+4.384320259094j
x2: 21.159729003906-8.017170906067j
x3: -9.657142639160+2.808838844299j
x4: 0.744012832642-0.587876081467j
x5: -0.269937247038+0.214877381921j
x6: -0.139848634601-0.025271922350j
x7: 0.099851913750+0.230989798903j
x8: -0.084345720708-0.516999006271j
x9: -0.061383247375-0.024532720447j
x10: 0.134453400970+0.608349084854j
x11: -0.110773518682-0.350651651621j
x12: 0.211009979248+0.198456317186j
x13: -0.252192974091-0.178134083748j
x14: 0.374807357788+0.256078958511j
x15: -0.286441326141-0.102149724960j
x16: 0.298572182655+0.000000000000j
x17: -0.286441326141+0.102149486542j
x18: 0.374808311462-0.256076097488j
x19: -0.252192497253+0.178133487701j
x20: 0.211009919643-0.198456317186j
x21: -0.110772885382+0.350651502609j
x22: 0.134452745318-0.608348846436j
x23: -0.061382599175+0.024532690644j
x24: -0.084345676005+0.516999006271j
x25: 0.099852085114-0.230990484357j
x26: -0.139849290252+0.025271654129j
x27: -0.269937098026-0.214877456427j
x28: 0.744012892246+0.587876081467j
x29: -9.657142639160-2.808837890625j
x30: 21.159730911255+8.017168045044j
x31: -7.834180355072-4.384319305420j
```

Figura 33: Resultado de la FFT de 32 muestras sobre un tono de 500Hz.

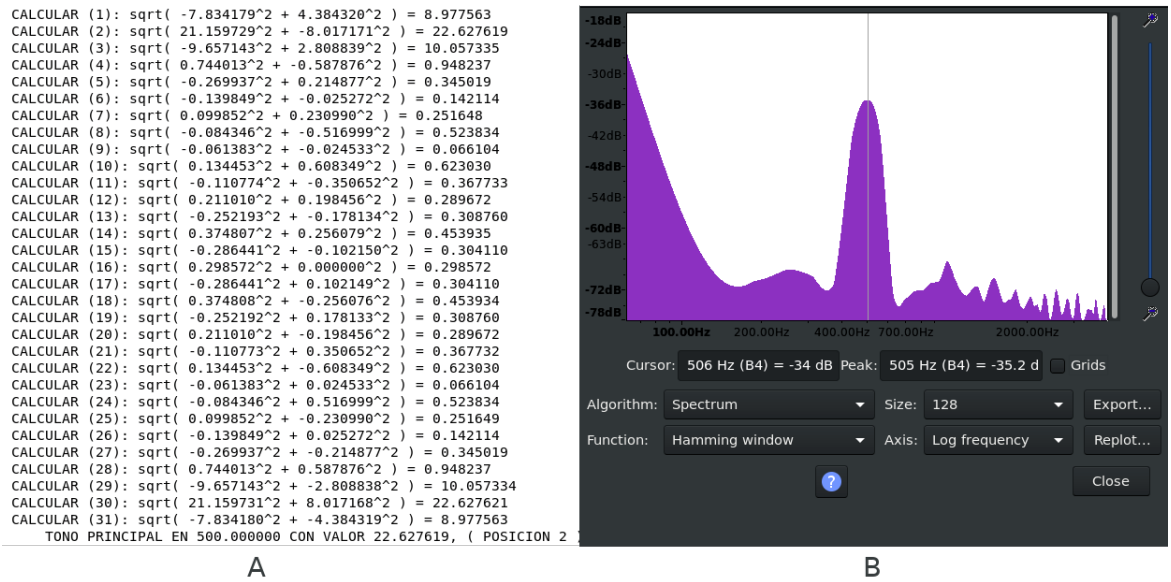


Figura 34: Resultados del proceso de cálculo para obtener el tono principal.

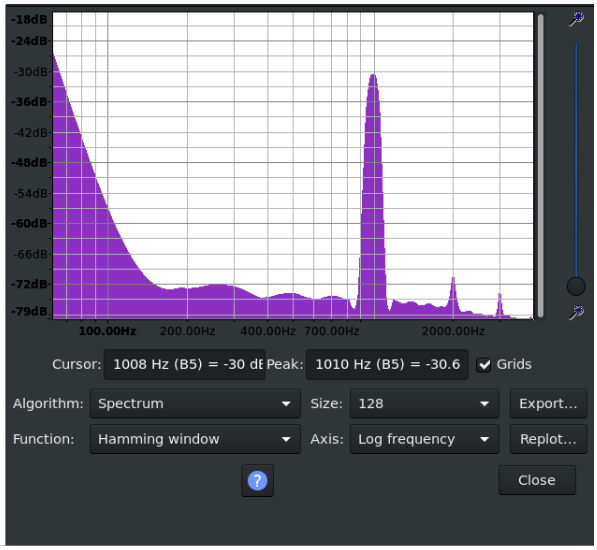
- Para una frecuencia de 1000Hz obtenemos los resultados de la FFT mostrados en la Figura 35. En la Figura 36, en la imagen A se muestra el desarrollo para el cálculo del tono principal donde se puede ver que la última linea indica el tono principal en 1000Hz, en la imagen B se presentar el espectrograma realizado, del fichero de audio obtenido, usando el programa Audacity.

```
x0: 1.834642887115+0.000000000000j
x1: -0.674727797508-0.052446261048j
x2: 0.084046423435-0.025580102578j
x3: 5.931284427643+2.602446556091j
x4: -12.970251083374-7.386242866516j
x5: 5.330876350403+3.670118570328j
x6: 0.058418180794-0.014031818137j
x7: -0.044875785708-0.205351144075j
x8: 0.256835252047+0.471687078476j
x9: -0.012782407925-0.221852153540j
x10: 0.057718455791-0.012395597063j
x11: -0.046023845673-0.039533495903j
x12: 0.251115322113+0.046918153763j
x13: -0.036049842834-0.042096138000j
x14: 0.065873950720+0.000982783735j
x15: 0.056402474642-0.003769606352j
x16: -0.047723770142+0.000000000000j
x17: 0.056402593851+0.003769615665j
x18: 0.065873950720-0.000982781872j
x19: -0.036049842834+0.042096138000j
x20: 0.251115798950-0.046918630600j
x21: -0.046024322510+0.039534568787j
x22: 0.057718459517+0.012395596132j
x23: -0.012782409787+0.221852213144j
x24: 0.256835192442-0.471687078476j
x25: -0.044875845313+0.205351293087j
x26: 0.058418184519+0.014031819068j
x27: 5.330875396729-3.670119285583j
x28: -12.970251083374+7.386243820190j
x29: 5.931283950806-2.602447271347j
x30: 0.084046423435+0.025580100715j
x31: -0.674727678299+0.052446268499j
```

Figura 35: Resultado de la FFT de 32 muestras sobre un tono de 1000Hz.

```
CALCULAR (1): sqrt( -0.674728^2 + -0.052446^2 ) = 0.676763
CALCULAR (2): sqrt( 0.084046^2 + -0.025580^2 ) = 0.087853
CALCULAR (3): sqrt( 5.931284^2 + 2.602447^2 ) = 6.477103
CALCULAR (4): sqrt( -12.970251^2 + -7.386243^2 ) = 14.925950
CALCULAR (5): sqrt( 5.330876^2 + 3.670119^2 ) = 6.472095
CALCULAR (6): sqrt( 0.058418^2 + -0.014032^2 ) = 0.060080
CALCULAR (7): sqrt( -0.044876^2 + -0.205351^2 ) = 0.210197
CALCULAR (8): sqrt( 0.256835^2 + 0.471687^2 ) = 0.537078
CALCULAR (9): sqrt( -0.012782^2 + -0.221852^2 ) = 0.222220
CALCULAR (10): sqrt( 0.057718^2 + -0.012396^2 ) = 0.059034
CALCULAR (11): sqrt( -0.046024^2 + -0.039533^2 ) = 0.060672
CALCULAR (12): sqrt( 0.251115^2 + 0.046918^2 ) = 0.255461
CALCULAR (13): sqrt( -0.036050^2 + -0.042096^2 ) = 0.055423
CALCULAR (14): sqrt( 0.065874^2 + 0.000983^2 ) = 0.065881
CALCULAR (15): sqrt( 0.056402^2 + -0.003770^2 ) = 0.056528
CALCULAR (16): sqrt( -0.047724^2 + 0.000000^2 ) = 0.047724
CALCULAR (17): sqrt( 0.056403^2 + 0.003770^2 ) = 0.056528
CALCULAR (18): sqrt( 0.065874^2 + -0.000983^2 ) = 0.065881
CALCULAR (19): sqrt( -0.036050^2 + 0.042096^2 ) = 0.055423
CALCULAR (20): sqrt( 0.251116^2 + -0.046919^2 ) = 0.255461
CALCULAR (21): sqrt( -0.046024^2 + 0.039535^2 ) = 0.060673
CALCULAR (22): sqrt( 0.057718^2 + 0.012396^2 ) = 0.059034
CALCULAR (23): sqrt( -0.012782^2 + 0.221852^2 ) = 0.222220
CALCULAR (24): sqrt( 0.256835^2 + -0.471687^2 ) = 0.537078
CALCULAR (25): sqrt( -0.044876^2 + 0.205351^2 ) = 0.210198
CALCULAR (26): sqrt( 0.058418^2 + 0.014032^2 ) = 0.060080
CALCULAR (27): sqrt( 5.330875^2 + -3.670119^2 ) = 6.472095
CALCULAR (28): sqrt( -12.970251^2 + 7.386244^2 ) = 14.925951
CALCULAR (29): sqrt( 5.931284^2 + -2.602447^2 ) = 6.477103
CALCULAR (30): sqrt( 0.084046^2 + 0.025580^2 ) = 0.087853
CALCULAR (31): sqrt( -0.674728^2 + 0.052446^2 ) = 0.676763
    TONO PRINCIPAL EN 1000.000000 CON VALOR 14.925950, ( POSICION 4
```

A



B

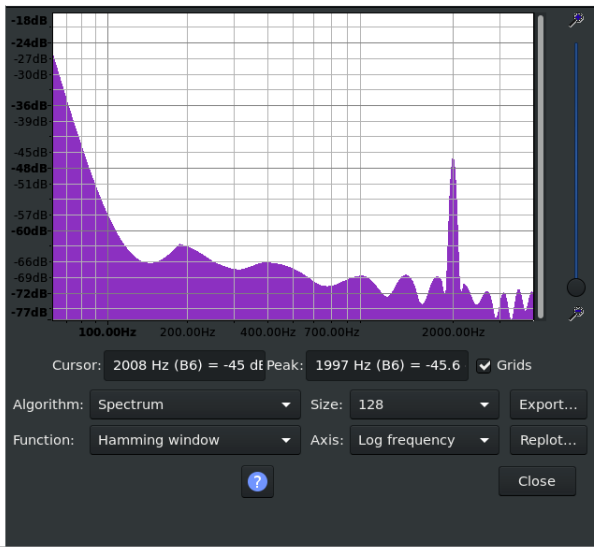
Figura 36: Resultados del proceso de cálculo para obtener el tono principal.

- Para una frecuencia de 2000Hz obtenemos los resultados de la FFT mostrados en la Figura 37. En la Figura 38, en la imagen A se muestra el desarrollo para el cálculo del tono principal donde se puede ver que la última linea indica el tono principal en 2000Hz, en la imagen B se presentar el espectrograma realizado, del fichero de audio obtenido, usando el programa Audacity.

```
x0: 7.333814144135+0.000000000000j
x1: -3.068974256516-0.326920479536j
x2: 0.032935302705+0.017257759348j
x3: 0.041096635163+0.017569649965j
x4: 0.045536171645-0.010412483476j
x5: 0.043124645948+0.017092641443j
x6: 0.044199097902+0.008363722824j
x7: -2.076751708984+4.541837215424j
x8: 6.038034915924-10.012619018555j
x9: -2.930931091309+4.041217327118j
x10: 0.055276639760+0.010317702778j
x11: 0.062399342656+0.010671744123j
x12: 0.057840690017-0.005119472742j
x13: 0.059005029500+0.008474636823j
x14: 0.070229977369+0.013441052288j
x15: 0.139580130577-0.007612049580j
x16: -0.133895874023+0.000000000000j
x17: 0.139579772949+0.007612198591j
x18: 0.070229977369-0.013441056944j
x19: 0.059005036950-0.008474708535j
x20: 0.057840693742+0.005119471811j
x21: 0.062399469316-0.010671660304j
x22: 0.055276643485-0.010317702778j
x23: -2.930931568146-4.041217327118j
x24: 6.038035869598+10.012619018555j
x25: -2.076752662659-4.541837215424j
x26: 0.044199094176-0.008363722824j
x27: 0.043124519289-0.017092909664j
x28: 0.045536175370+0.010412484407j
x29: 0.041096866131-0.017569750547j
x30: 0.032935291529-0.017257755622j
x31: -3.068974494934+0.326920270920j
```

Figura 37: Resultado de la FFT de 32 muestras sobre un tono de 2000Hz

```
CALCULAR (1): sqrt( -3.068974^2 + -0.326920^2 ) = 3.086338
CALCULAR (2): sqrt( 0.032935^2 + 0.017258^2 ) = 0.037183
CALCULAR (3): sqrt( 0.041097^2 + 0.017570^2 ) = 0.044695
CALCULAR (4): sqrt( 0.045536^2 + -0.010412^2 ) = 0.046711
CALCULAR (5): sqrt( 0.043125^2 + 0.017093^2 ) = 0.046389
CALCULAR (6): sqrt( 0.044199^2 + 0.008364^2 ) = 0.044983
CALCULAR (7): sqrt( -2.076752^2 + 4.541837^2 ) = 4.994115
CALCULAR (8): sqrt( 6.038035^2 + -10.012619^2 ) = 11.692323
CALCULAR (9): sqrt( -2.930931^2 + 4.041217^2 ) = 4.992173
CALCULAR (10): sqrt( 0.055277^2 + 0.010318^2 ) = 0.056231
CALCULAR (11): sqrt( 0.062399^2 + 0.010672^2 ) = 0.063305
CALCULAR (12): sqrt( 0.057841^2 + -0.005119^2 ) = 0.058067
CALCULAR (13): sqrt( 0.059005^2 + 0.008475^2 ) = 0.059611
CALCULAR (14): sqrt( 0.070230^2 + 0.013441^2 ) = 0.071505
CALCULAR (15): sqrt( 0.139580^2 + -0.007612^2 ) = 0.139788
CALCULAR (16): sqrt( -0.133896^2 + 0.000000^2 ) = 0.133896
CALCULAR (17): sqrt( 0.139580^2 + 0.007612^2 ) = 0.139787
CALCULAR (18): sqrt( 0.070230^2 + -0.013441^2 ) = 0.071505
CALCULAR (19): sqrt( 0.059005^2 + -0.008475^2 ) = 0.059611
CALCULAR (20): sqrt( 0.057841^2 + 0.005119^2 ) = 0.058067
CALCULAR (21): sqrt( 0.062399^2 + -0.010672^2 ) = 0.063305
CALCULAR (22): sqrt( 0.055277^2 + -0.010318^2 ) = 0.056231
CALCULAR (23): sqrt( -2.930932^2 + -4.041217^2 ) = 4.992174
CALCULAR (24): sqrt( 6.038036^2 + 10.012619^2 ) = 11.692323
CALCULAR (25): sqrt( -2.076753^2 + -4.541837^2 ) = 4.994115
CALCULAR (26): sqrt( 0.044199^2 + -0.008364^2 ) = 0.044983
CALCULAR (27): sqrt( 0.043125^2 + -0.017093^2 ) = 0.046388
CALCULAR (28): sqrt( 0.045536^2 + 0.010412^2 ) = 0.046711
CALCULAR (29): sqrt( 0.041097^2 + -0.017570^2 ) = 0.044695
CALCULAR (30): sqrt( 0.032935^2 + -0.017258^2 ) = 0.037183
CALCULAR (31): sqrt( -3.068974^2 + 0.326920^2 ) = 3.086338
    TONO PRINCIPAL EN 2000.000000 CON VALOR 11.692323, ( POSICION 8
```



A B

Figura 38: Resultados del proceso de cálculo para obtener el tono principal.

Para continuar con la validación del sistema se toman las muestras de entrada, del archivo de captura, que proceso el subsistema y las copiamos en el Matlab. Usando el Matlab aplicamos a las muestras de entrada la ventana de Hamming y realizamos la FFT de 32. Esta prueba la hemos realizado para los procesos mostrados anteriormente. En la Figura 39 se muestran los plots correspondientes al procesado de las muestras en Matlab, se comprueba que la salida se corresponde con los valores obtenidos por el subsistema. La Figura 39 A, corresponde a un tono de 500Hz, la Figura 39 B corresponde a un tono de 1000Hz y la Figura 39 C corresponde a un tono de 2000Hz.

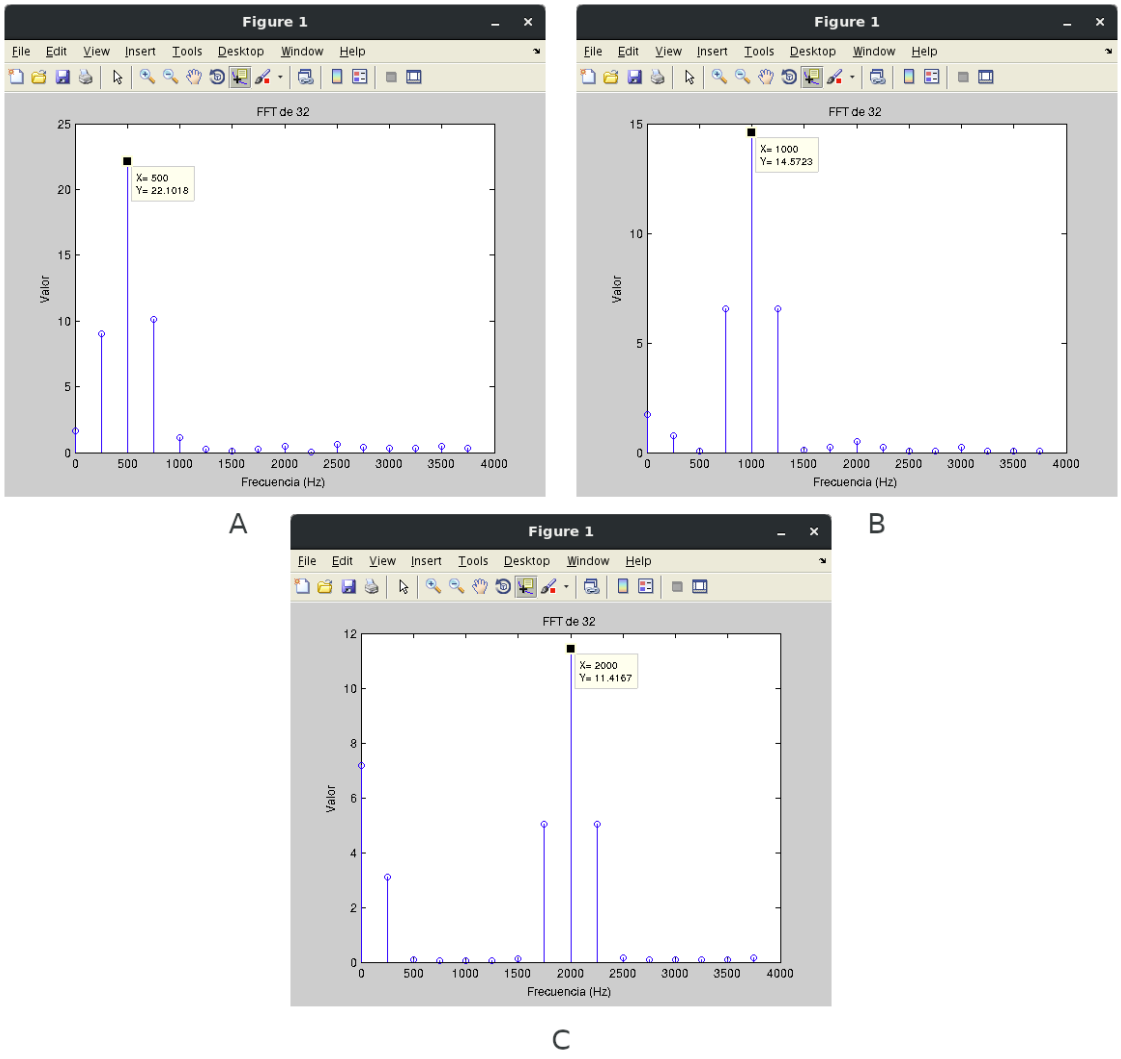


Figura 39: Gráficas, de la FFT de 32 muestras, realizadas con Matlab.

Con las medidas realizadas se da por validado el subsistema.

Cabe destacar que la resolución en frecuencia, para el caso descrito, es de 250Hz, y los tonos usados en las pruebas son múltiplo de dicho valor, por este motivo los resultado son exactos. De usar un tono que no sea múltiplo de dicho valor, la exactitud en el cálculo de la frecuencia no sería tan precisa. Con el aumento del tamaño de la FFT la resolución de frecuencia aumenta siendo posible identificar el tono principal con mayor precisión.

4.5.4 Integración dentro del sistema desarrollado

En el proceso de inicio los subsistemas realizan una identificación contra el servidor destino, tal como se muestra en la Figura 40. El servidor recibe la identificación, almacena los datos de interés y responde con la configuración inicial que tenga almacenada para el subsistema en cuestión.

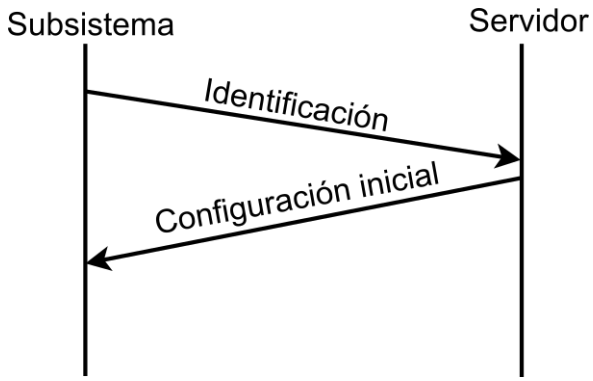


Figura 40: Diagrama de flujo de la identificación del subsistema con el servidor.

Una vez realizada la identificación el subsistema queda registrado en el servidor.

Durante la conexión el servidor solicita, de forma periódica, el keep-alive de todos los subsistemas que estén conectados y no estén realizando algún tipo de procesado. Tal como se muestra en la Figura 41, el servidor envía una trama solicitando una señal de actividad y el subsistema responde enviando su configuración actual.

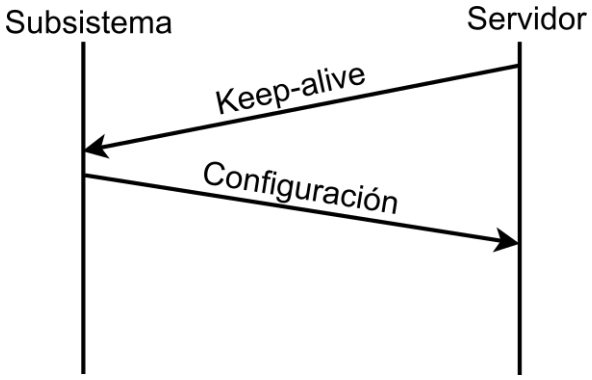


Figura 41: Diagrama de flujo de la solicitud de keep-alive.

Para realizar la solicitud de muestreo a un subsistema, el servidor envía los datos de configuración para la adquisición de muestras y el subsistema responde enviando las muestras adquiridas en formato wav, junto con la información de configuración realizada para la adquisición y datos de identificación, ver Figura 42.

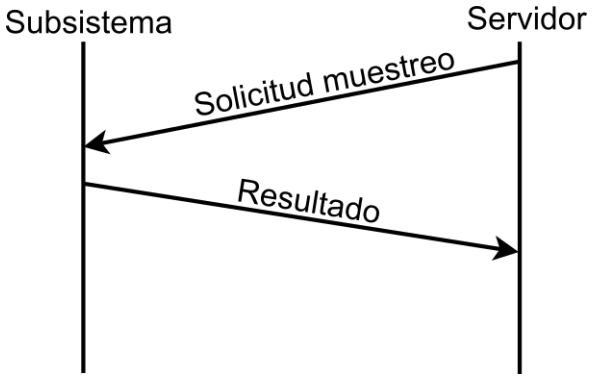


Figura 42: Diagrama de flujo de la solicitud de muestreo.

En la solicitud de procesado de la FFT y calculo del tono principal, el servidor envía los datos de configuración para la adquisición de muestras así como los datos para la configuración necesaria para realizar el procesado de la FFT, el subsistema responderá con el resultado de la adquisición de muestras donde enviará el resultado del muestreo en formato wav, y seguidamente, enviará cada procesado de la FFT realizado junto al cálculo del tono hasta realizarlas todas, en cada envío el subsistema enviará datos de identificación, ver Figura 43.

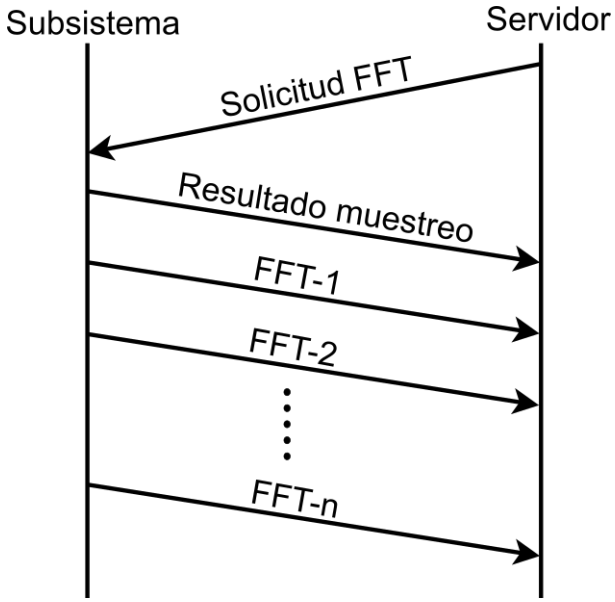


Figura 43: Diagrama de flujo de la solicitud de procesado de la FFT.

Capítulo 5. Desarrollo del servidor web para la recopilación de datos

Como se comentó en el apartado 2.4, utilizaremos Apache Server como sistema para el desarrollo de nuestra aplicación web, que implementará el portal web ofrecido al usuario y a los subsistemas. Para el desarrollo de esta aplicación web se hará uso del lenguaje PHP.

A continuación se detallan los pasos a seguir para la configuración del Apache Server, necesaria para nuestro proyecto:

- **Activar el módulo rewrite**. Se crea un enlace simbólico en el directorio /etc/apache2/mod-enable/ que apunte al modulo que está en /etc/apache2/mod-available/rewrite.load.
- **Generar el fichero de configuración**. Para indicar al servidor Apache que debe hacer con la peticiones recibidas por el puerto TCP 80, se modifica el fichero /etc/apache2/site-available/000-default.conf con la configuración siguiente:

```
<VirtualHost *:80>
      ServerAdmin webmaster@localhost
      DocumentRoot /var/www/html

      <Directory /var/www/html
            Options Indexes FollowSymLinks
            AllowOverride All
            Require all granted
      </Directory>

      ErrorLog ${APACHE_LOG_DIR}/error.log
      CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

En la configuración mostrada le indicamos al servidor Apache que todas las peticiones recibidas por el puerto 80 deben ir al directorio /var/www/html (DocumentRoot), la configuración para dicho directorio se muestra entre las etiquetas Directory y en la parte final de la configuración se le indica donde debe alojar el registro de errores y de accesos.

- **Crear el fichero llamado .htaccess**. Debe estar en el directorio indicado en el fichero de configuración, con el siguiente contenido:

```
<IfModule mod_rewrite.c>
      RewriteEngine On
      RewriteBase /
      RewriteRule ^index\.php$ - [L]
      RewriteCond %{REQUEST_FILENAME} !-f
      RewriteCond %{REQUEST_FILENAME} !-d
      RewriteRule . /index.php [L]
</IfModule>

php_value post_max_size 300M
php_value upload_max_filesize 300M
php_value max_file_uploads 2000
```

Esta configuración indica al servidor Apache las reglas para nuestro alojamiento web.

Como se comento en el apartado 2.4, el sistema para la gestión de la base de datos que se utilizará es el PostgreSQL.

Será necesario instalar las librerías indicadas a continuación para usar el lenguaje de programación PHP:

- **php.** Conjunto de librerías necesarias para la ejecución de programas PHP.

- **php-pgsql.** Librería utilizada para la comunicación con el sistema gestor de la base de datos.
- **php-curl.** Librearía utilizada para la comunicación usando el protocolo HTTP en modo cliente.

5.1 Análisis y diseño de bloques

El diseño de la web se estructura sobre el patrón MVP (MVP, *Model View Presenter*). El desarrollo de la aplicación se ha dividido en dos bloques, una para la vista-presentador de la web y otra para gestionar los datos.

La vista permite al usuario: ver los distintos subsistemas que están conectados; gestionar la configuración por defecto de cada subsistema; realizar peticiones a los distintos subsistemas; y ver el historial de los procesos de cada subsistema.

El módulo de control se encarga de: la gestión de la base de datos; de almacenar, actualizar, eliminar y obtener los diferentes registros; de la comunicación con los subsistemas; y del mantenimiento de los directorios para el almacenamiento de la información recibida de los distintos subsistemas.

5.2 Estudio y Diseño del bloque de almacenamiento y gestión

Para la gestión de la base de datos, el sistema usa llamadas a través del protocolo HTTP, lo que permite que los datos estén almacenados en un servidor distinto al que gestiona la aplicación web. Para acceder al módulo de gestión es necesario construir una dirección como la siguiente: url/api/nombre_del_servicio, donde la url es la dirección del servidor donde se almacenarán los datos, api es la ruta para indicar al servidor web lo que se solicita y el nombre_del_servicio hace referencia al proceso o clase que se desea cargar para la gestión de datos.

En el servidor se almacenan los datos obtenidos de cada subsistema. Cada proceso solicitado es almacenado en la base de datos, por cada registro se genera con un identificador único, el cual es enviado al subsistema junto con los parámetros de configuración de la petición realizada, el identificador sirve para enlazar los datos recibidos de los distintos subsistemas con su solicitud correspondiente, dicho

identificador es usado para generar los ficheros de almacenamiento que se distribuyen en: uno para el audio, uno para la información de la FFT y los tonos.

En este proyecto solo se ha diseñado una clase llamada **dispositivo** para la gestión de los datos referentes a los subsistemas. Dicha clase gestiona la base de datos mostrada en la Figura 44.

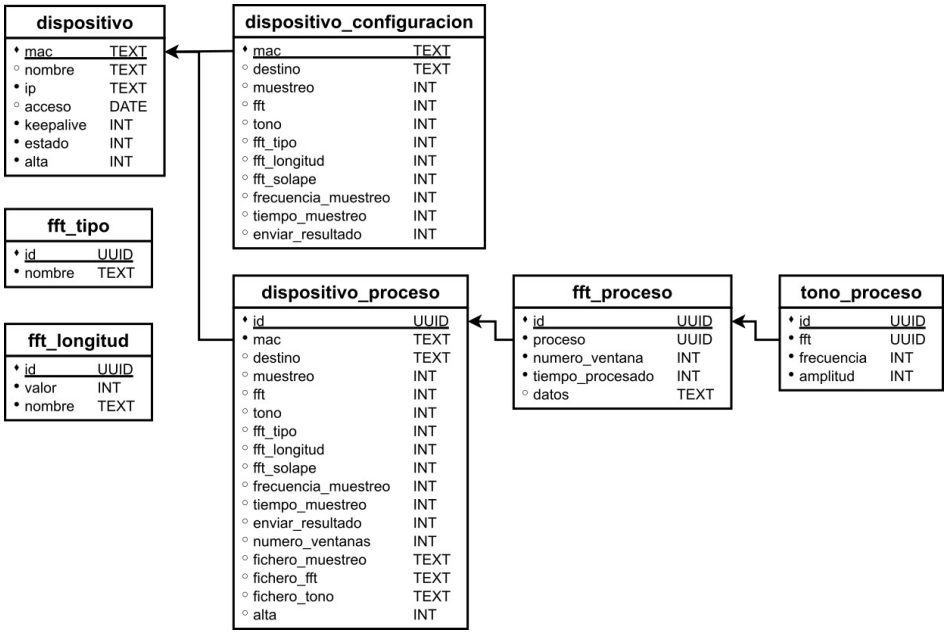


Figura 44: Esquema relacional del modelo de datos

El contenido de las tablas es el que sigue:

- **dispositivo**, almacena los datos de básicos de cada subsistema. Contiene los siguientes campos: *mac*, dirección mac del subsistema; *nombre*, nombre del subsistema; *ip*, dirección de red; *acceso*, fecha del último acceso; *keepalive*, última vez que dio señal de operatividad; *estado*, estado actual del subsistema; *alta*, fecha en la que fue dado de alta.

- **dispositivo_configuracion**, almacena la configuración por defecto de cada subsistema. Cuenta con los siguiente campos: *mac*, dirección mac del subsistema; *destino*, dirección de red del destino para enviar los resultados del procesado; *muestreo*, indica si se debe realizar el muestreo; *fft*, indica si se desea realizar la FFT; *tono*, indica si se desea calcular el tono principal de la FFT; *fft_tipo*, tipo de FFT; *fft_longitud*, tamaño de la FFT; *fft_solape*, cantidad de solapamiento entre FFT; *frecuencia_muestreo*, frecuencia de muestreo; *tiempo_muestreo*, tiempo de la muestra; *enviar_resultado*, indica si los resultados deben ser enviados al destino.

- **dispositivo_proceso**, almacena todas las peticiones enviadas a los distintos subsistemas. Cuenta con los siguientes campos: *id*, identificador del proceso; *mac*, dirección mac del subsistemas; *destino*, dirección de red del destino de los resultados del procesado; *muestreo*, indica si hay que realizar el muestreo; *fft*, indica si hay que realizar la FFT; *tono*, indica si hay que calcular el tono de cada FFT; *fft_tipo*, tipo de FFT; *fft_longitud*, tamaño de la FFT; *fft_solape*, solapamiento entre cada FFT; *frecuencia_muestreo*, frecuencia para realizar el muestreo; *tiempo_muestreo*, tiempo de la muestra; *enviar_resultado*, indica si el resultado del procesado debe ser enviado; *fichero_muestreo*, dirección del fichero con el contenido del muestreo; *fichero_fft*, dirección del fichero con la información de cada FFT; *fichero_tono*, dirección del fichero con la información de cada tono.
- **fft_tipo**, almacena los posibles tipos de FFT que pueden ser solicitados y cuenta con los siguiente campos: *id*, identificador del registro; *nombre*, nombre de la FFT.
- **fft_longitud**, almacena los valores posibles para el tamaño de la FFT y cuenta con los siguientes campos: valor, indica la longitud de la FFT; nombre, relaciona el valor con una pequeña descripción.
- **fft_proceso**, almacena los datos del procesado de cada FFT y cuenta con los siguientes campos: *id*, identificador del registro; *proceso*, identificador del proceso padre; *numero_ventanas*, número de ventanas a calcular; *numero_ventana*, número de la ventana a la que pertenecen los datos; *frecuencia*, frecuencia del tono principal; *tono*, amplitud del tono principal; *tiempo_procesado*, indica el tiempo que llevo al subsistema realizar la FFT; *datos*, resultado de la FFT de la ventana indicada.
- **tono_proceso**, almacena los datos del procesado cuando se ha solicitado solamente el tono, cuenta con los siguiente campo: *id*, identificador del registro; *proceso*, identificador del proceso padre; *tiempo_procesado*, indica el tiempo que llevo al subsistema realizar el cálculo del tono; *frecuencia*, frecuencia del tono principal; *tono*, amplitud del tono principal.

Además, se asegura la existencia de los directorios necesarios para almacenar los datos recibidos en los ficheros correspondientes. Debe existir un directorio por cada uno de los subsistemas, cuyo nombre es su dirección mac. En él se almacenarán los

datos obtenidos de cada subsistema en el formato que corresponda, tal como se muestra en la Figura 45.

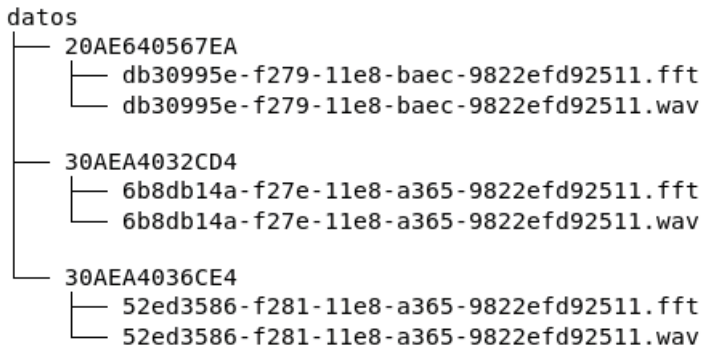


Figura 45: Estructura del directorio destino de los datos recibidos.

5.3 Estudio y Diseño del bloque interfaz web.

La interfaz web contiene dos clases para gestionar la vista: home y subsistema. Tal como se especifica en la configuración del fichero *.htaccess*, comentado anteriormente, todas las llamadas a la web pasan por el directorio raíz (DocumentRoot). En el raíz se aloja el fichero *index.php* donde se gestiona el destino de la petición recibida. Cuando en la uri se hace mención a **api,** el sistemas dirige la petición al módulo de gestión de la base de datos y cuando no hace dicha mención dirige la petición hacia la interfaz web.

La interfaz de nuestra web está almacenada un directorio llamado app. Dentro de éste hay un directorio por cada clase: home y subsistema; junto a varios fichero necesarios para la presentación del resultado final de la interfaz web. La interfaz se construye atendiendo a la uri y a las variables recibidas en la petición. En este TFG se han diseñado tres vistas: home, vista general de los dispositivos que hay registrados en el sistema, ver figura Figura 46. Está pantalla informa al usuario de todos los subsistemas gestionados y sus datos de carácter general. Por cada subsistema hay tres botones que permiten interactuar con cada uno de ellos.

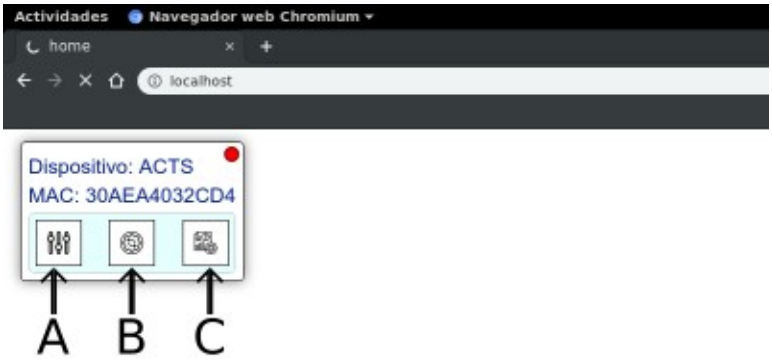


Figura 46: Interfaz Web - Vista inicial, home

Por cada subsistemas registrado aparece una caja, en la parte superior derecha se indica el estado del subsistema, el cual puede ser:

- **rojo**, inactivo
- **verde**, activo y listo
- **amarillo**, activo y procesando alguna petición.

Cada caja dispone de los siguiente botones:

- **Botón A**. Muestra al usuario la vista que corresponde a la configuración básica del subsistema, ver figura Figura 47. Desde esta vista el usuario puede definir los parámetros iniciales de cada subsistema y estos parámetros son almacenados por el módulo de gestión de la base de datos y enviados al subsistemas correspondiente cuando este se identifica por primera vez.

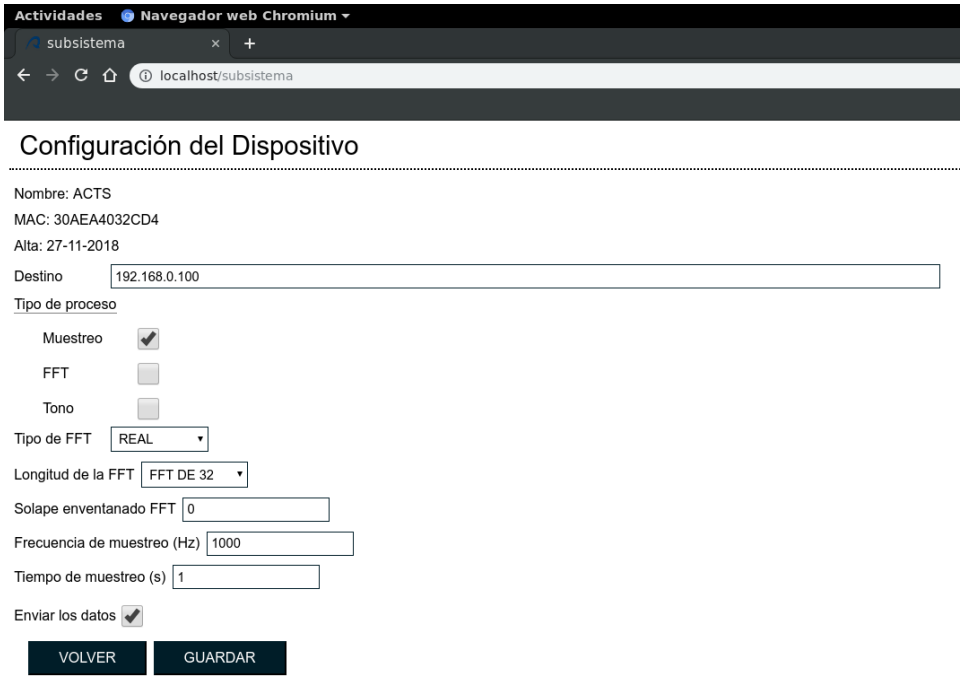


Figura 47: Interfaz Web - Vista para la configuración básica del subsistemas

- **Botón B**. Muestra al usuario la vista que corresponde con la parte de gestión de procesos, ver figura Figura 48. Desde esta vista el usuario puede solicitar al subsistema que realice el/los proceso/s seleccionado/s. Está vista es igual que la vista anterior con la salvedad de que permite al usuario solicitar el resultado de un proceso.

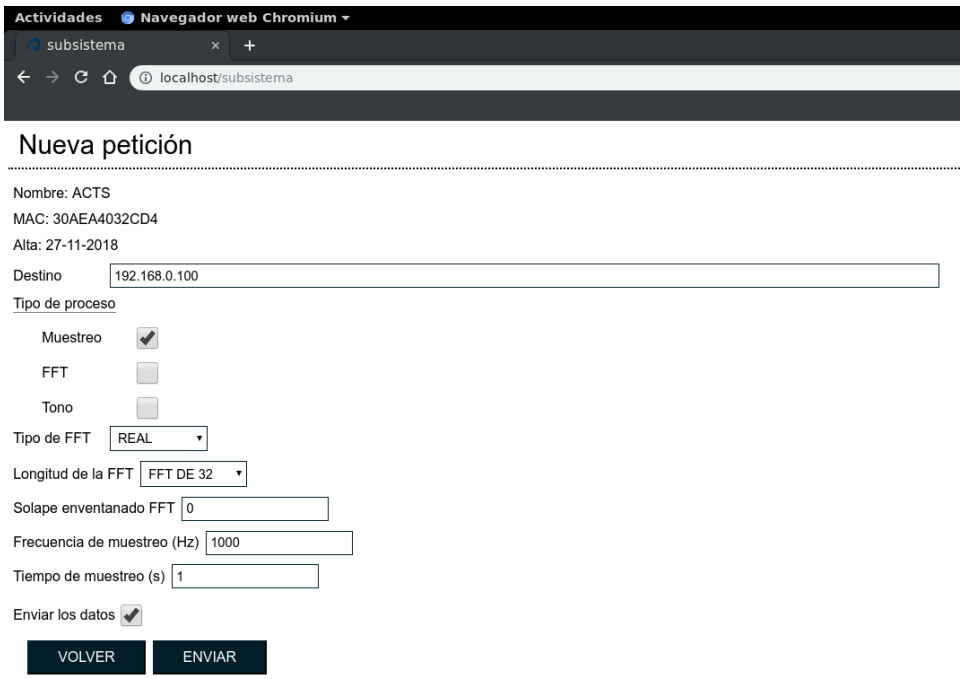
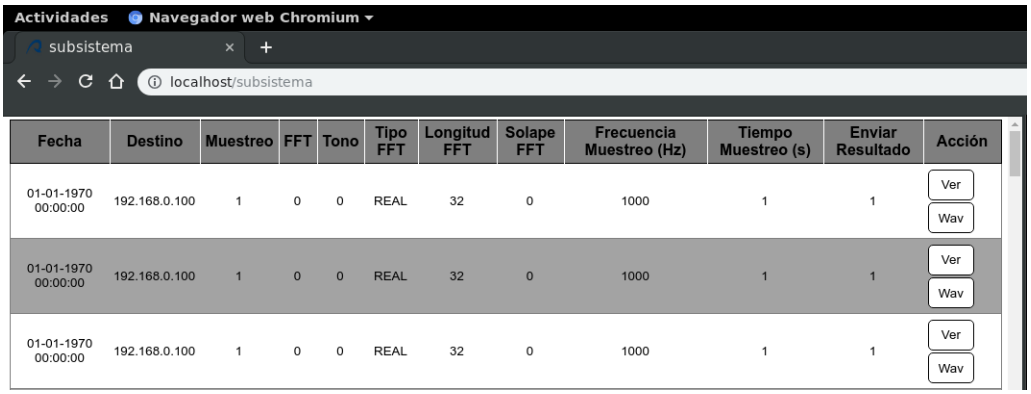


Figura 48: Interfaz Web - Vista para la captura-proceso de audio.

- **Botón C**. Muestra la relación de procesos que ha realizado el subsistema, ver figura Figura 49. Desde la vista del historial se puede descargar el fichero wav que corresponde con el muestreo, ver el detalle de la/s FFT/s realizada/s. También se puede imprimir un informe con el contenido del registro mostrado.



Fecha	Destino	Muestreo	FFT	Tono	Tipo FFT	Longitud FFT	Solape FFT	Frecuencia Muestreo (Hz)	Tiempo Muestreo (s)	Enviar Resultado	Acción
01-01-1970 00:00:00	192.168.0.100	1	0	0	REAL	32	0	1000	1	1	Ver Wav
01-01-1970 00:00:00	192.168.0.100	1	0	0	REAL	32	0	1000	1	1	Ver Wav
01-01-1970 00:00:00	192.168.0.100	1	0	0	REAL	32	0	1000	1	1	Ver Wav

Figura 49: Interfaz Web - Vista del historial de procesos del subsistema

5.4 Estudio y Diseño del bloque de procesado de datos obtenidos/informes

Cuando el usuario solicita el historial de un subsistema se le presenta en pantalla el listado de los procesos almacenados, tal como se muestra en la figura Figura 49. El usuario puede acceder al contenido de un registro o descargar el contenido de la captura en formato wav. El contenido del registro es un listado con el proceso del cálculo de la FFT y, de haberse solicitado, del tono, tal como se muestra en la figura Figura 50. En el listado de procesos se detalla la cantidad de ventanas a calcular, en cada registro se indica: a que ventana corresponden el cálculo; tiempo que se tomó para realizar la FFT; frecuencia a la que se encontró el tono principal, de haberse solicitado; amplitud del tono principal, de haberse solicitado.

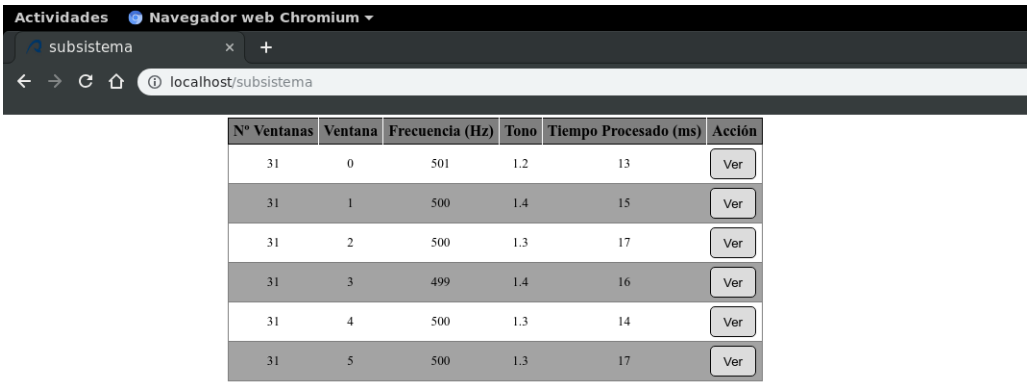


Figura 50: Interfaz Web - Listado del proceso de la FFT

5.5 Validación de sistema

Para la validación del sistema seguimos el mismo procedimiento que el comentado en el apartado 4.4 pero en esta ocasión se hará uso del portal web alojado en el servidor Apache. En las primeras pruebas solo solicitamos el muestreo, posteriormente se ha de comprobar que la muestra almacenada en el servidor se corresponde con el sonido muestreado, para la verificación usaremos la herramienta Audacity, tal como se hizo en el apartado 4.4.

En la Figura 51 se muestra una prueba de recepción de un tono a 600Hz. Se puede observar que la forma de onda es una senoide. En la Figura 52 se muestra el espectrograma de la audio recibido, en ella se observa que el tono principal esta en 600Hz.

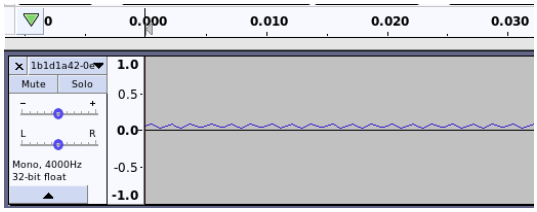


Figura 51: Lectura del fichero recibido con Audacity

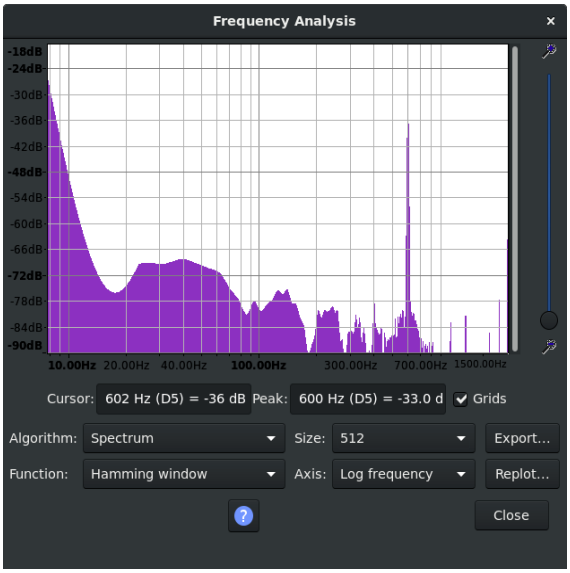


Figura 52: Spectograma del fichero recibido con Audacity

Tras la verificación de la correcta recepción de las muestras, se realizarán varias pruebas solicitando la FFT con dos tamaños de ventana, posteriormente se ha de comprobar que los datos de los enventanados han sido almacenados correctamente.

+8.27133 +3.27617 +0.28215 +0.12376 +0.11742 +0.05128 +0.11503 +38.91094
+90.65134 +38.64381 +0.18658 +0.14756 +0.12645 +0.07949 +0.18317 +1.61531
+3.99587 +1.68111 +0.10054 +0.15774 +0.12537 +0.08536 +0.19148 +0.15984
+0.47648 +0.09867 +0.11197 +0.09301 +0.04237 +0.03073 +0.09173 +0.63233
+1.81856 +0.64048 +0.09733 +0.09515 +0.13486 +0.03566 +0.08926 +0.97918
+2.33085 +0.89824 +0.09711 +0.03898 +0.06067 +0.03052 +0.03283 +0.60161
+1.33134 +0.64587 +0.06767 +0.06700 +0.03476 +0.03589 +0.16150 +0.94810
+1.98498 +0.85962 +0.08914 +0.02356 +0.06798 +0.09162 +0.12250 +0.35468
+1.13163 +0.35468 +0.12250 +0.09162 +0.06798 +0.02356 +0.08914 +0.85962
+1.98499 +0.94810 +0.16150 +0.03589 +0.03476 +0.06700 +0.06767 +0.64587
+1.33134 +0.60161 +0.03283 +0.03052 +0.06067 +0.03898 +0.09711 +0.89824
+2.33085 +0.97918 +0.08926 +0.03566 +0.13486 +0.09515 +0.09733 +0.64048
+1.81856 +0.63233 +0.09173 +0.03073 +0.04237 +0.09301 +0.11197 +0.09867
+0.47648 +0.15984 +0.19148 +0.08536 +0.12537 +0.15774 +0.10054 +1.68111
+3.99587 +1.61531 +0.18317 +0.07949 +0.12645 +0.14755 +0.18658 +38.64381
+90.65134 +38.91095 +0.11503 +0.05128 +0.11742 +0.12376 +0.28215 +3.27617

Figura 53: Datos del procesado de la FFT de 128 muestras a un tono de 500Hz

En la Figura 53 se muestra el contenido de la FFT para un enventanado de 128 muestras.

Se toman los datos, y se pasan al Matlab para su validación. En dicho proceso se realiza un plot de los datos obteniendo la gráfica mostrada en la Figura 54.

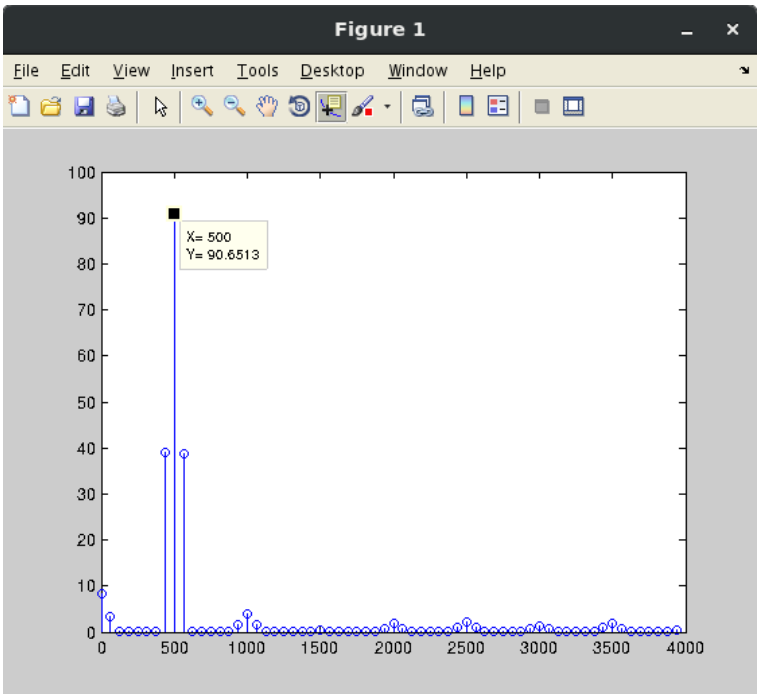


Figura 54: Plot de los datos obtenidos, del tono a 500Hz, usando la herramienta Matlab

Solo resta hacer una verificación completa, se debe solicitar al subsistema que haga el muestreo, la FFT y calcule el tono correspondiente, posteriormente se debe comprobar que las frecuencias de los tono obtenidas se correspondan con el tono del sonido emitido durante la prueba.

Tras realizar la petición del proceso de muestreo a 8000 bps durante 1 segundo, con una FFT 256 y cálculo del tono, obtenemos el resultado mostrado en la Figura 55.

```
CALCULAR   (1): sqrt( 3.272928^2 + -0.145672^2 ) = 3.276168
CALCULAR   (2): sqrt( 0.263766^2 + 0.100184^2 ) = 0.282151
CALCULAR   (3): sqrt( 0.112535^2 + -0.051491^2 ) = 0.123755
CALCULAR   (4): sqrt( -0.077690^2 + 0.088050^2 ) = 0.117424
CALCULAR   (5): sqrt( 0.031944^2 + -0.040112^2 ) = 0.051277
CALCULAR   (6): sqrt( 0.113539^2 + 0.018433^2 ) = 0.115025
CALCULAR   (7): sqrt( -35.060120^2 + 16.877483^2 ) = 38.910942
CALCULAR   (8): sqrt( 82.558685^2 + -37.439682^2 ) = 90.651344
CALCULAR   (9): sqrt( -35.639256^2 + 14.939463^2 ) = 38.643810
CALCULAR  (10): sqrt( 0.168039^2 + 0.081080^2 ) = 0.186577
                  •
                  •
                  •
                  •
CALCULAR (117): sqrt( -0.090040^2 + -0.116897^2 ) = 0.147554
CALCULAR (118): sqrt( 0.168039^2 + -0.081080^2 ) = 0.186577
CALCULAR (119): sqrt( -35.639259^2 + -14.939458^2 ) = 38.643810
CALCULAR (120): sqrt( 82.558685^2 + 37.439674^2 ) = 90.651344
CALCULAR (121): sqrt( -35.060127^2 + -16.877476^2 ) = 38.910946
CALCULAR (122): sqrt( 0.113539^2 + -0.018433^2 ) = 0.115025
CALCULAR (123): sqrt( 0.031943^2 + 0.040111^2 ) = 0.051276
CALCULAR (124): sqrt( -0.077690^2 + -0.088050^2 ) = 0.117424
CALCULAR (125): sqrt( 0.112535^2 + 0.051490^2 ) = 0.123756
CALCULAR (126): sqrt( 0.263766^2 + -0.100184^2 ) = 0.282151
CALCULAR (127): sqrt( 3.272930^2 + 0.145668^2 ) = 3.276170
    TONO PRINCIPAL EN 500.000000 CON VALOR 90.651344, ( POSICION 8 )
```

Figura 55: Resultado del cálculo del tono con una FFT de 128 muestras aplicada a una muestrea tomada a 8000 bps

Además, en el CD tenemos varios vídeos adjuntos en el directorio anexo/vídeo, donde se muestra el funcionamiento del sistema.

6 Conclusiones y trabajos futuros

6.1 Conclusiones

Tras la validación realizada en el apartado 5.5 podemos afirmar que los objetivos establecidos para el presente Trabajo Fin de Grado han sido cumplidos. Entre estos se encuentran: estudio y caracterización de los diferentes dispositivos usados; desarrollo de la aplicación en el subsistema para proporciona una pequeña interfaz web que posibilita la interacción directa con el subsistema, desde la cual se puede adquirir y procesar audio así como realizar la descarga de las muestras tomadas en un fichero con formato wav; desarrollado un portal web en una máquina exterior que permite la centralización e interacción con los distintos subsistemas.

En resumen, al haber cumplido con los objetivos establecidos es posible interactuar con los distintos subsistemas de la red desde un mismo terminal, esto permite la ejecución de procesos en los distintos subsistemas de forma paralela, pudiendo analizar los resultados de cada uno de ellos haciendo uso del portal web o descargando los ficheros adjuntos a cada proceso para un análisis personalizado. Con el sistema propuesto es posible ubicar dos subsistemas en un mismo sitio, solicitar

distintos tipos de procesos en cada subsistema y contrastar los resultado para obtener mejores conclusiones.

El impacto económico de la solución propuesta es pequeño en comparación a las funciones que este ofrece.

Una de las características importantes del sistema propuesto es su versatilidad. Permite al usuario realizar gran variedad de configuraciones, cada subsistema puede ser configurado de manera diferente, esto permite al usuario solicitar distintos procesos para un posterior estudio.

El sistema diseñado en este TFG es fácilmente escalable, el servidor externo permite multitud de conexiones. Al ser el subsistema el que se identifica al iniciarse, el servidor externo tendrá una lista de todos los subsistemas de la red, el servidor externo tiene rutinas que permiten la interrogación del estado de cada subsistema para presentarlo al usuario.

A continuación, se exponen las conclusiones obtenidas al terminar la implementación del presente TFG, desde una perspectiva hacia la experiencia adquirida en su desarrollo:

- El estudio y caracterización de los distintos elementos que conforman el subsistema es clave. La calidad del micrófono utilizado es una parte importante, así como la capacidad de almacenamiento del SoC.
- La disposición de librerías ayudan al desarrollo de la solución planteada, aún habiendo ocasiones en las que es necesario realizar alguna modificación por requerir un uso para el cual no está pensada.
- El uso del protocolo HTTP facilita la interacción del sistema en la red, es un protocolo estandarizado y de uso extendido. Hay herramientas, como el wireshark, que permiten monitorizar las comunicaciones con este protocolo, esto nos permite, de una manera fácil y amigable, verificar el funcionamiento de la comunicación en el sistema.
- El subsistemas ESP32 cuenta con gran cantidad de documentación ofrecida desde la página del fabricante, Espressif, además de una faq con la solución a los problemas más conocidos, y un foro con bastante actividad.

6.2 Trabajos futuros

La solución propuesta en el presenta TFG no deja de ser un prototipo basado en una serie de conceptos teóricos. Se pueden plantear varias mejoras, como:

- Se podría implementar un sistema para reconocimiento de voz, para ello se podría realizar el cálculo de MFCC para el cuál se podría utilizar la librería libmfcc[21].
- Se podría usar los 4MB de memoria flash del subsistema para almacenar el modelo resultante de un entrenamiento con una red neuronal pequeña y usar dicho modelo para realizar una clasificación. Con esta mejora el dispositivo podría ser usado para clasificar eventos en el propio dispositivo, como por ejemplo: la rotura de un cristal; el grito de una persona; el llanto de un niño; etc.
- Se podría diseñar un sistema que permita acoplar al subsistema una unidad de alimentación formada por una placa solar alimentando una batería, consiguiendo mayor flexibilidad a la hora de posicionar los subsistemas.
- Se podría implementar una centralización de los servidores, esto permitiría obtener la información almacenada en todos los servidores para un posterior estudio.

Biliografía

[1] M. Universidad de Costa Rica. Escuela de Bibliotecología y Ciencias de la Información, A. Segura, M. Alvarado, and M. Loaiza, *E-ciencias de la información.*, vol. 8, no. 1. Universidad de Costa Rica, 2018.

[2] C.-Y. Chang, C.-H. Kuo, J.-C. Chen, and T.-C. Wang, “Design and Implementation of an IoT Access Point for Smart Home,” *Appl. Sci.*, 2015.

[3] M. Peruzzini, M. Germani, A. Papetti, and A. Capitanelli, “Smart home information management system for energy-efficient networks,” in *IFIP Advances in Information and Communication Technology*, 2013.

[4] M. Quintana-Suárez, D. Sánchez-Rodríguez, I. Alonso-González, and J. Alonso-Hernández, “A Low Cost Wireless Acoustic Sensor for Ambient Assisted Living Systems,” *Appl. Sci.*, vol. 7, no. 9, p. 877, Aug. 2017.

[5] Arduino, “Arduino,” 2018. [Online]. Available: https://www.arduino.cc/.

[6] Libelium, “Waspmote,” 2018. [Online]. Available: http://www.libelium.com/products/waspmote/.

[7] Particle, “Particle,” 2018. [Online]. Available: https://www.particle.io/. [Accedido: 02-Ene-2019].

[8] Espressif, “Espressif,” 2018. [Online]. Available: https://www.espressif.com/.

[9] Espressif, “ESP32 Resources | Espressif Systems,” 2018. [Online]. Available: https://www.espressif.com/en/products/hardware/esp32/resources. [Accedido: 02-Ene-2019].

[10] L. Llamas, “KY-038,” 2016. [Online]. Available: https://www.luisllamas.es/detectar-sonido-con-arduino-y-microfono-ky-038/. [Accedido: 02-Ene-2019].

[11] V. Solution, “VS1003,” 2018. [Online]. Available: http://pdf1.alldatasheet.com/datasheet-pdf/view/106198/ETC/VS1003.html. [Accedido: 02-Ene-2019].

[12] Adrafruit, “MAX4466,” 2018. [Online]. Available: https://www.adafruit.com/product/1063.[Accedido: 02-Ene-2019].

[13] Netcraft, 2018. [Online]. Available: https://www.netcraft.com. [Accedido: 02-Ene-2019].

[14] Joyent, “NodeJS.” [Online]. Available: https://nodejs.org/es/. [Accedido: 02-Ene-2019].

[15] Python, 2018. [Online]. Available: https://www.python.org/. [Accedido: 02-Ene-2019].

[16] T. A. Fundation, “Apache Server.” [Online]. Available: https://www.apache.org/. [Accedido: 15-Mar-2018].

[17] Arduino, “Arduino,” 2018. [Online]. Available: https://www.arduino.cc/en/Main/Software. [Accedido: 02-Ene-2019].

[18] Platformio, “PlatformIO,” 2018. [Online]. Available: https://platformio.org/. [Accedido: 02-Ene-2019].

[19] ESP-IDF, “ESP-IDF Programming Guide,” 2018. [Online]. Available: http://esp-idf.readthedocs.io/en/latest/. [Accedido: 14-Mar-2018].

[18] Espressif, “ESP32WEbServer,” 2018. [Online]. Available: https://github.com/espressif/arduino-esp32/issues/425. [Accedido: 14-Mar-2018].

[19] FFTW, “FFTW,” 2018. [Online]. Available: https://github.com/FFTW/fftw3. [Accedido: 02-Ene-2019].

[20] Generador de Frecuencias, 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.boedec.hoel.frequencygenerator. [Accedido: 02-Ene-2019].

[21] R. K. D, “LIBMFCC,” 2010. [Online]. Available: https://github.com/rohithkd/libmfcc. [Accedido: 14-Mar-2018].

Pliego de condiciones

El conjunto de herramientas hardware, software así como firmware usados para el desarrollo de este TFG, son expuestas a continuación.

PC.1 Elementos hardware

En la siguiente tabla se presenta el listado de elementos hardware utilizados.

Equipo	Modelo	Fabricante/Comenciante
Portátil	Lenovo Yoga 910	Lenovo
Ordenador	Gigabyte H110M-S2H	Gigabyte
ESP32	ESP.WROMM.32	Espressif
Micrófono	MAX4466	Adafruit
Teléfono móvil	Samsung Note 8	Samsung

Tabla 7: Elementos hardware utilizados

PC.2 Elementos software

En la siguiente tabla se presenta el listado de las aplicaciones software utilizadas.

Aplicación	Versión
Sistema operativo	Linux - Debian 9
LibreOffice	6.1.3.2
PlatformIO	3.6.2rc1
Geany	1.33
Minicom	2.7.1
Dia	0.97
Inkscape	0.92.3
Planner	0.14.16
Git	2.20.1

Tabla 8: Aplicaciones software utilizadas

Licencia del prototipo

Normas generales

De conformidad con lo dispuesto en la normativa reguladora en materia de Propiedad Intelectual, el TFG es considerado una obra en colaboración del estudiante y el/los tutor/tutores. Para la explotación industrial del TFG será de aplicación lo establecido por los Estatutos de la Universidad de Las Palmas de Gran Canaria.

El uso de prototipo desarrollado ha de ser bajo la expresa autorización del autor o tutor/tutores del TFG o de la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria.

Derecho de autor

El prototipo y la documentación están protegidos por la ley de Propiedad Intelectual aplicable, así como por las disposiciones de los tratados internacionales. En consecuencia, el usuario podrá usar una copia de esté prototipo, así como del código fuente y documentación, siempre bajo la autorización del autor o tutor/tutores del TFG

o de la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria.

Garantía

Se asegura un correcto funcionamiento del sistema si la instalación se lleva a cabo según las especificaciones vistas con anterioridad.

De haber algún cambio en la versión de alguna de las librería usadas, no se podrá asegurar el corrector funcionamiento.

Presupuesto

Se presentan los gastos generados en la realización de este TFG. Los gastos son desglosados de la siguiente manera:

- Trabajo tarifado por el tiempo empleado.
- Amortización del inmovilizado material.
  - Amortización del material hardware.
  - Amortización del material software.
- Redacción del documento
- Derechos de visado del Colegio Oficial de Ingenieros Técnicos de Telecomunicación (de aquí en adelante COITT).
- Gastos de tramitación y envío.
- Material fungible.

Analizados todos los puntos anteriormente presentados, se aplicarán los impuestos vigentes y se calculará el coste total del TFG.

P.1 Trabajo tarifado por el tiempo empleado

Se contabilizan los gastos referentes a la mano de obra, atendiendo al salario correspondiente a la hora de trabajo de un Ingeniero Técnico de Telecomunicación. Para el cálculo de los honorarios totales se hace uso de la siguiente fórmula:

$$H = C \cdot (74{,}88 \cdot H_n + 96{,}72 \cdot H_e) \quad \text{(P1)}$$

Donde:

- H_n, número de horas trabajadas dentro de la jornada laboral.
- H_e, número de horas trabajadas fuera de la jornada laboral.
- C, factor de corrección en función de las horas trabajadas.

Al desarrollo de este TFG se ha invertido un total de 300 horas. Todas ellas han sido realizada durante la jornada laboral. En la siguiente tabla se muestra los factores de corrección, impuestos por el COITT, que se han de aplicar según el número de horas trabajadas.

Horas	Factor de corrección
Hasta 36	1,00
Exceso de 36 a 72	0,90
Exceso de 72 a 108	0,80
Exceso de 108 a 144	0,70
Exceso de 144 a 180	0,65
Exceso de 180 a 360	0,60

Tabla 9: Factor de corrección por horas trabajadas según el COITT

De acuerdo con lo establecido por el COITT, según se indica en la tabla anterior, el factor de corrección C que se debe aplicar es de 0,60. Se sustituyen los datos en la fórmula P1:

$$H = 0{,}60 \cdot (74{,}88 \cdot 300 + 96{,}72 \cdot 0) = 13.478{,}40\,€$$

El trabajo tarifado por tiempo empleado asciende a la cantidad de *TRECE MIL CUATROCIENTOS SETENTA Y OCHO EUROS CON CUARENTA CÉNTIMOS*.

P.2 Amortización del inmovilizado material

Se tendrán en consideración los recursos, tanto hardware como software, empleados para el desarrollo de este TFG.

Para calcular el coste de amortización en un periodo de 3 años se utiliza un sistema de amortización lineal, en el que se supone que el inmovilizado material se deprecia de forma constante a lo largo de su vida útil. El cálculo de la cuota de amortización anual se obtiene de la siguiente fórmula:

$$Cuota\ Anulla = \frac{Valor\ de\ adquisición - Valor\ residual}{Años\ de\ vida\ útil} \quad \text{(P2)}$$

El valor residual es un valor teórico el cuál se supone que tendrá el elemento en cuestión después de su vida útil.

P.2.1 Amortización del material hardware

El periodo de desarrollo de este TFG ha sido de 4 meses, periodo muy inferior a los 3 años que se estipulan para el coste de amortización. Por dicha razón se calculan los costes sobre la base de los derivados de los primeros 4 meses.

En la siguiente tabla se listan los elementos hardware amortizables necesarios para el desarrollo de este TFG. En dicha lista se indican el valor de adquisición del elemento así como su amortización para el tiempo de 4 meses indicado.

Elemento	Valor de adquisición (€)	Amortización (€)
Portatil	1800,00	200,00
Ordenador	500,00	55,56
ESP32	12,00	12,00
Micrófono	2,68	2,68
	Total	**270,24**

Tabla 10: Amortización del material hardware

El coste total del material hardware es de *DOSCIENTOS SETENTA EUROS CON VEINTICUATRO CÉNTIMOS.*

P.2.2 Amortización del material software

Tal como se comento en el apartado 6.3.2.1, sólo se tendrá en cuenta los costes derivados de los 4 primeros meses.

En la siguiente tabla se listan las aplicaciones necesarias para el desarrollo de este TFG. En dicha lista se indican el valor de adquisición así como el valor de amortización.

Aplicación	Valor de adquisición (€)	Amortización (€)
Sistema operativo	0,00	0,00
LibreOffice	0,00	0,00
PlatforIO	0,00	0,00
Geany	0,00	0,00
Minicom	0,00	0,00
Dia	0,00	0,00
Inkscape	0,00	0,00
	Total	**0,00**

Tabla 11: Amortización del material software

El coste total de las aplicaciones necesarias es de CERO EUROS. Esto se debe a que todo el software usado es libre o de código abierto.

P.3 Redacción del documento

Para determinar el coste de la redacción del documento se usa la siguiente fórmula:

$$R = 0{,}07 \cdot P \cdot C_n \qquad \text{(P3)}$$

Donde:

- P, es el presupuesto.
- C_n, es el coeficiente de ponderación en función del presupuesto.

El valor del presupuesto equivale a la suma de los costes de trabajo tarifados por tiempo con las amortizaciones del inmovilizado material. En la siguiente tabla se muestras el cálculo del presupuesto.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	270,24
Total	**13.748,64**

Tabla 12: Presupuesto según la tarificación por tiempo y la amortización de material.

Según el COITT, el factor de ponderación, para presupuestos cuyo valor es inferior a 30.050,00€, es de 1,00, por lo que los costes derivados de la redacción de documento se obtienen sustituyendo los valores en la ecuación P3.

$$R = 0{,}07 \cdot 13.748{,}64 \cdot 1{,}00 = 962{,}40\,€$$

El coste de la redacción del documento tiene un coste de *NOVECIENTOS SESENTA Y DOS EUROS CON CUARENTA CÉNTIMOS*.

P.4 Derechos de visado del COITT

Para proyectos de carácter general, los derechos de visado para el año 2018 se calculan de acuerdo a la siguiente fórmula.

$$V = 0{,}006 \cdot P_1 \cdot C_1 + 0{,}003 \cdot P_2 \cdot C_2 \qquad \text{(P4)}$$

Donde:

- P_1, es el valor del presupuesto para ester proyecto.
- C_1, es el coeficiente reductor en función del presupuesto.
- P_2, es el presupuesto de ejecución material que corresponde a la obra civil.
- C_2, es el coeficiente reductor en función del presupuesto correspondiente a obra civil.

Tal como se comentó en el apartado 6.3.3, el coeficiente C_1 está fijado a 1,00. En el desarrollo de este TFG no se ha requerido obra civil por lo cual el valor de P_2 es 0,00€. En la siguiente tabla se muestra el valor del presupuesto hasta el momento.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	270,24
Redacción del documento	962,40
Total	**14.711,04**

Tabla 13: Presupuesto según la tarificación por tiempo, la amortización de material y la redacción del documento.

Se sustituyen los valores en la ecuación P4.

$$V = 0{,}006 \cdot 14.711{,}04 \cdot 1{,}00 + 0{,}003 \cdot 0{,}00 \cdot C_2 = 88{,}27\,€$$

Los costes por derecho de visado del presente TFG son de *OCHENTA Y OCHO EUROS CON VEINTISIETE CÉNTIMOS*.

P.5 Gastos de tramitación y envío

Cada documento visado por vía telemática tiene un coste de 6,00€.

P.6 Material fungible

Durante el desarrollo de este TFG se han usado otros materiales a parte de los recursos hardware y software comentados anteriormente. El la siguiente tabla se listan los costes derivados de estos recursos.

Concepto	Coste (€)
Folios	10,00
Tóner para la impresora	30,00
Encuadernado	5,00
CDs	6,00
Total	**51,00**

Tabla 14: Material fungible

El coste del material fungible es de *CINCUENTA Y UN EUROS*.

P.7 Aplicación de impuestos y coste total

El desarrollo del presenta TFG esta por el Impuesto General Indirecto Canario con un valor del 6.5%. Teniendo en cuenta la aplicación de dicho impuesto, en la siguiente tabla se realiza el cálculo total del proyecto.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	270,24
Redacción del documento	962,40
Derechos de visado del COITT	88,27
Gastos de tramitación y envío	6,00
Costes de material fungible	51,00
Subtotal	**14.856,31**
I.G.I.C.	**965,66**
Total	**15.821,97**

Tabla 15: Coste total del proyecto

El coste total para el desarrollo del TFG "Diseño de una plataforma web basada en ESP32 para adquisición y procesado de eventos sonoros" tiene una valor de *QUINCE MIL OCHOCIENTOS VEINTIÚN EUROS CON NOVENTA Y SIETE CÉNTIMOS*.