UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS DE GANDIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Desarrollo de un medidor de intensidad sonora conectado a un servidor web a partir de un controlador M5Stack Core 2

Trabajo Fin de Grado

Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen

AUTOR/A: Romaguera Pérez, Néstor

Tutor/a: Marín-Roig Ramón, José

CURSO ACADÉMICO: 2024/2025

RESUMEN

En este proyecto se desarrolla un sistema de medición de intensidad acústica mediante el uso del microcontrolador M5Stack Core2 y su micrófono integrado MEMS (Micro-Electro-Mechanical System), con el objetivo de mostrar en tiempo real el nivel de decibelios (dB) presentes en el ambiente.

Para el almacenamiento y visualización de los datos recogidos, se han implementado dos soluciones distintas:

· Un servidor web local basado en XAMPP, que conecta una base de datos SQL con una página HTML sencilla para consultar las mediciones realizadas.

· La plataforma ThingSpeak de MathWorks, que permite el envío y visualización de los datos en la nube en tiempo real.

El proyecto busca ofrecer una solución económica, funcional y de fácil uso para la monitorización de niveles sonoros en distintos entornos.

Palabras clave: M5Stack Core2; Arduino; Medición acústica; Visualización de datos; ThingSpeak.

ABSTRACT

This project presents the development of an acoustic intensity measurement system using the M5Stack Core2 microcontroller and its integrated MEMS (Micro-Electro-Mechanical System) microphone, with the aim of displaying in real time the decibel (dB) levels present in the environment.

Two different approaches have been implemented for storing and visualizing the collected data:

· A local web server based on XAMPP, which connects an SQL database with a simple HTML page to display the measurements.

· The ThingSpeak platform by MathWorks, which enables cloud-based data transmission and real-time visualization.

The goal of this project is to provide an affordable, functional, and user-friendly solution for monitoring sound levels in various environments.

Key words: M5Stack Core2; Arduino; Acoustic Measurement; Data visualization; ThingSpeak.

ÍNDICE

Contenido

Tabla de figuras

1. Introducción

1.1 Motivación

Desde que inicié mis estudios en el Grado en Ingeniería de Sistemas de Telecomunicaciones, sentí una gran curiosidad por el funcionamiento de la electrónica digital. Me fascinaba cómo, mediante el uso de diferentes placas, módulos y combinaciones de ceros y unos, era posible construir máquinas complejas y funcionales. Siempre quise comprender cómo operaban estos dispositivos internamente y, durante mis años en la universidad, enfoqué mis esfuerzos en desarrollar los conocimientos y habilidades necesarios para, en el futuro, poder crear mis propios proyectos.

Del mismo modo, siempre he disfrutado del aprendizaje de distintos lenguajes de programación. Cuando descubrí Arduino y su entorno de desarrollo integrado, entendí que tenía ante mí una herramienta muy potente. Usando un lenguaje basado en C++, podía controlar dispositivos electrónicos, experimentar con sensores y actuadores, y desarrollar aplicaciones interactivas, uniendo así la programación y la electrónica en un mismo entorno de aprendizaje práctico.

Por ello, desde un principio supe que quería enfocar mi Trabajo de Fin de Grado en un proyecto relacionado con Arduino. Buscaba trabajar en algo que implicara la medición de datos, y que además me permitiera diseñar una interfaz para visualizarlos, creando así un canal de comunicación entre el dispositivo medidor y el entorno de usuario.

Tras plantear esta idea a mi tutor, Pepe, me propuso un proyecto concreto que encajaba perfectamente con mis intereses: diseñar e implementar un sistema de medición de decibelios utilizando un microcontrolador M5Stack Core2, compatible con el entorno Arduino. Su propuesta me pareció una gran oportunidad para profundizar en mis conocimientos tanto de programación como de hardware embebido, así que decidí asumir el reto y comenzar el desarrollo con ese objetivo en mente.

1.2 Objetivos

Este proyecto trata de diseñar e implementar un medidor de intensidad acústica, que sea capaz de hacer mediciones del ruido ambiente, con el objetivo de poder crear un sistema de monitorización del ruido en horario nocturno. Gracias a este sistema, se podría tener un control del nivel de contaminación acústica en el horario de noche, con el objetivo de informar de aquellas áreas donde el nivel de decibelios percibido sobrepase un umbral que puede llegar a ser considerado molesto o dañino.

Para conseguir dicho objetivo, se pretende, aprovechar el micrófono integrado, modelo SPM1423HM4H-B, del microcontrolador M5Stack core2, para que el propio M5Stack Core2 pueda obtener los decibelios (dBs) del ambiente, aprovecharemos la pantalla programable del microcontrolador para que muestre los valores que va registrando.

Por otra parte, se diseñarán dos sistemas de visualización de datos, uno donde se podrá ver una página HTML local, que estará conectada a una base de datos SQL, gracias a la aplicación XAMPP, una herramienta de software libre que permite crear un servidor web local. Aprovechando que XAMPP permite usar el lenguaje de programación PHP, que permitirá generar contenido HTML dinámico a partir de datos almacenados en una base de datos MYSQL local gestionada también por la propia herramienta XAMPP. Dentro del contenido dinámico generado, se tratará de establecer un umbral de "aviso" donde las muestras que superen cierto umbral, sean marcadas a modo de detección de niveles acústicos elevados. El segundo sistema que implementaremos utilizará la plataforma de análisis para el internet de las cosas (IoT) desarrollada por Mathworks, que nos permitirá conectarnos a su API web (Application Programming Interface) para visualizar los datos en tiempo real, pudiendo generar gráficas.

Con ello, se podrán crear las bases de un sistema de medición y consulta de datos que utilice estos elementos, consiguiendo un esquema económico que funcione bien y pueda competir con otras soluciones propuestas por otras empresas.

1.3 Metodología y planificación

La planificación del proyecto se ha estructurado en varias fases, desarrolladas de forma secuencial según los objetivos técnicos planteados. Cada fase se abordó mediante investigación, pruebas iterativas y validación de los resultados, permitiendo una evolución progresiva del sistema desde su núcleo funcional hasta su integración completa.

Fase 1: Implementación del cálculo de decibelios

La primera etapa consistió en implementar la funcionalidad principal del proyecto: la medición de niveles de presión sonora (dB) mediante el microcontrolador M5Stack Core2 y su micrófono MEMS digital. Para ello, se realizó una investigación inicial consultando documentación técnica oficial, foros y proyectos similares desarrollados con Arduino.

Tras analizar ejemplos que utilizaban la interfaz del micrófono, y realizar diversas pruebas con distintas fórmulas de cálculo, se obtuvo una primera versión funcional del código, capaz de mostrar un nivel de decibelios en pantalla con un rango y precisión adecuados para un sistema de propósito general.

Fase 2: Almacenamiento local y visualización en entorno web

Una vez establecida la lectura y cálculo correcto de los niveles sonoros, se abordó la necesidad de almacenar las mediciones para su posterior análisis.

Se investigaron distintas herramientas de conectividad y bases de datos compatibles con Arduino y se optó finalmente por utilizar XAMPP, por su facilidad de configuración y su capacidad para simular un entorno completo de servidor web (Apache + PHP + MySQL) en red local.

Desde el entorno PHPMyAdmin de XAMPP se diseñó una base de datos MySQL adaptada al tipo de datos que generaba el M5Stack Core2. A continuación, se modificó el código Arduino para enviar los datos de dB mediante peticiones HTTP al servidor local, donde se almacenaban correctamente en la base de datos.

Una vez resuelto el flujo de entrada de datos, se desarrolló una página HTML dinámica mediante PHP que mostraba en una tabla todos los registros almacenados, incluyendo valores de dB y otros parámetros asociados. Esta interfaz podía consultarse desde cualquier navegador conectado a la red local. Esta tabla dinámica, se implementó con la funcionalidad de marcar de forma visual aquellos parámetros que superasen el umbral de "aviso" pudiéndose ver con un recuadro de advertencia aquellos valores que se registren como peligrosos.

Fase 3: Visualización remota en la nube mediante ThingSpeak

Como paso adicional para mejorar la accesibilidad del sistema y permitir la consulta remota desde cualquier dispositivo con conexión a internet, se integró el sistema con la plataforma ThingSpeak, desarrollada por MathWorks. Esta herramienta permite enviar datos mediante API y visualizarlos en tiempo real en un panel web configurable.

Para implementar esta integración, se partió de nuevo del código original de lectura de decibelios, y se adaptó para conectarse directamente a la API de ThingSpeak. Como referencia, se utilizó el código de una práctica previa de la asignatura "Proyecto CDIO", del primer curso del grado en Tecnologías Interactivas, cuyo acceso fue facilitado por el tutor del proyecto. En dicha práctica se realizaba una conexión HTTP a la misma plataforma, por lo que solo fue necesario adaptar los parámetros de entrada y reestructurar las peticiones para transmitir correctamente los datos de decibelios generados por el dispositivo.

Gracias a esta última fase, el sistema es ahora capaz de operar tanto en un entorno local como en la nube, con posibilidad de visualizar las mediciones en tiempo real desde cualquier lugar, lo cual amplía significativamente las posibilidades de uso y despliegue del proyecto.

1.4 Estructura de la memoria

El siguiente documento se estructura en varios puntos en los que se tratan los siguientes aspectos:

- Situación actual de los medidores acústicos conectados a servidores web
- Elección de los componentes necesarios a partir de la propuesta planteada
- Plan de desarrollo completo donde se definirán todos los procesos llevados a cabo, tanto en la fase de codificación del medidor de dBs, la fase de creación de un documento HTML dinámico, como la fase de conectividad con la API de Mathworks
- Muestra de pruebas funcionales, y los resultados obtenidos en cada medición
- Conclusiones y propuestas de mejora

2. Estado del arte

En los últimos años, el avance de la tecnología IoT (Internet of Things) ha permitido el desarrollo de dispositivos de medición acústica más compactos, precisos y conectados. Dentro de este contexto, los micrófonos inteligentes conectados a bases de datos representan una herramienta fundamental en aplicaciones que van desde la monitorización ambiental hasta la ingeniería acústica. La posibilidad de registrar, analizar y almacenar datos sonoros en tiempo real ha transformado la forma en que las instituciones, empresas y administraciones gestionan la información sobre el ruido.

Actualmente, en el mercado existen múltiples soluciones que integran sensores acústicos con conectividad inalámbrica para su vinculación directa con plataformas de análisis de datos o bases de datos SQL. Empresas como Libelium, con su solución Waspmote Plug & Sense! Smart Environment, ofrecen kits de sensores que permiten medir niveles de ruido en exteriores y enviarlos mediante WiFi, 4G o LoRaWAN a servidores remotos [1]. De forma similar, la compañía NTi Audio comercializa el NoiseScout [Figura 1], un sistema autónomo de monitorización de ruido que permite el registro continuo en entornos urbanos e industriales, con acceso remoto a través de una interfaz web y almacenamiento en la nube [2]. Otras propuestas, como los dispositivos de Brüel & Kjær, se centran en el sector profesional y ofrecen micrófonos de clase 1 con capacidades de integración en redes de control y bases de datos mediante protocolos como TCP/IP y FTP.

Además de estas soluciones comerciales, plataformas como Arduino y ESP32 han impulsado una gran variedad de proyectos de bajo coste basados en micrófonos MEMS o electret, combinados con conectividad WiFi y bases de datos MySQL. Esto ha dado lugar a una comunidad activa de desarrolladores e investigadores que han logrado implementar soluciones funcionales de monitorización acústica en entornos educativos, urbanos o industriales a precios considerablemente más asequibles que los dispositivos profesionales. Una de las placas más utilizadas en este contexto es la M5Stack Core2, que incluye capacidades WiFi y es compatible con sensores analógicos y digitales. Gracias a su arquitectura abierta, permite el desarrollo de sistemas embebidos personalizados que pueden almacenar mediciones en tiempo real en bases de datos remotas mediante protocolos HTTP y servidores intermedios PHP o Node.js.

El interés por este tipo de tecnología no se limita al ámbito académico o empresarial, sino que también ha encontrado una aplicación directa en la gestión urbana del ruido. Ciudades como Barcelona, París o Nueva York han implementado redes de sonómetros inteligentes como parte de sus políticas de smart city. Estas redes permiten una monitorización continua del ruido ambiental en puntos críticos, como zonas escolares, zonas de ocio nocturno o vías de alta densidad de tráfico. En Barcelona, por ejemplo, el Ayuntamiento desplegó más de 60 sonómetros conectados a la plataforma Sentilo para medir en tiempo real los niveles de ruido y ajustar las políticas municipales en base a datos objetivos [3]. Estos dispositivos suelen cumplir con la normativa de clase 1 según la IEC 61672-1 y están conectados a bases de datos que permiten generar informes, detectar eventos anómalos y activar alertas automáticas.

En resumen, el estado actual de la tecnología en monitorización acústica muestra una evolución clara hacia dispositivos conectados, interoperables y orientados a la recolección masiva de datos en tiempo real. Esta tendencia responde a la necesidad de tomar decisiones fundamentadas en datos objetivos, ya sea en el ámbito de la salud pública, la sostenibilidad urbana o la gestión industrial. La existencia de una amplia gama de soluciones, desde dispositivos de bajo coste hasta sistemas profesionales, permite adaptar estas tecnologías a diferentes contextos presupuestarios y técnicos, ampliando significativamente su aplicabilidad y alcance.

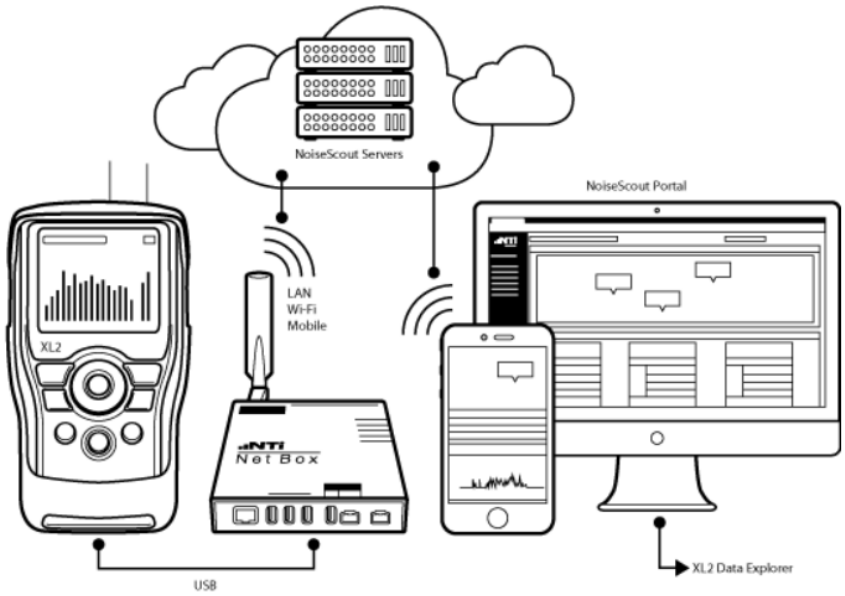


Figura 1. Esquema de la tecnología de NoiseScout, de compañía Nti Audio. Fuente: ntiaudio.cn

2.1 Funcionamiento técnico de un sonómetro profesional

Un sonómetro profesional es un instrumento diseñado para medir con precisión los niveles de presión sonora conforme a estándares internacionales, como la norma IEC 61672-1, que establece las especificaciones para los medidores de nivel de sonido de clase 1 y clase 2. El principio básico de funcionamiento parte de la captación de la presión acústica mediante un micrófono de condensador, que convierte las variaciones de presión del aire en una señal eléctrica proporcional. Este tipo de micrófonos, debido a su alta linealidad y baja distorsión, son especialmente adecuados para aplicaciones de medida.

La señal eléctrica generada por el micrófono es de muy baja amplitud y requiere de una etapa de preamplificación analógica. En esta etapa se adapta la impedancia y se eleva el nivel de la señal hasta un rango adecuado para su posterior procesamiento. A continuación, el sonómetro incorpora un conversor analógico-digital (ADC) de alta resolución —típicamente de 24 bits— que digitaliza la señal con una frecuencia de muestreo suficiente para abarcar el espectro audible (20 Hz – 20 kHz). De esta manera, la información de presión sonora queda representada como una serie de valores discretos en el dominio digital.

Una vez digitalizada, la señal es sometida a distintos procesos de filtrado digital. En primer lugar, se aplican filtros de ponderación frecuencial, como A, C o Z, que simulan la respuesta del oído humano o bien proporcionan una medida lineal. Posteriormente, se aplican filtros temporales (Fast, Slow, Impulse) que determinan cómo se promedian las variaciones rápidas de la señal en el tiempo. Estos parámetros de ponderación y respuesta temporal son configurables en el sonómetro según la normativa y el tipo de medida que se desee realizar.

El cálculo del nivel de presión sonora (SPL) se basa en la relación entre la señal RMS (Root Mean Square) de la presión acústica medida y un valor de referencia estándar de 20 µPa, que corresponde al umbral de audición humana a 1 kHz. Matemáticamente, el nivel de presión sonora se expresa como:

$$Lp = 20 \cdot log10(pRMS/p0) \quad [dB\ SPL]$$

Donde $pRMS$ es la presión eficaz obtenida de la señal medida y $p0 = 20\mu Pa$ es la presión de referencia.

El resultado de este cálculo se representa en pantalla en tiempo real. En los modelos más avanzados, estos cálculos se realizan mediante procesadores digitales de señal (DSP) dedicados, capaces de ejecutar análisis en bandas de octava o tercios de octava, así como transformadas rápidas de Fourier (FFT) para obtener información espectral más detallada.

Finalmente, el sonómetro integra sistemas de almacenamiento interno o transmisión de datos hacia bases de datos externas. Los dispositivos profesionales incluyen puertos USB, conectividad Ethernet o incluso módulos inalámbricos para la sincronización con plataformas de análisis. De esta forma, la información medida puede ser utilizada en estudios ambientales, auditorías acústicas o sistemas de control normativo.

Este esquema de funcionamiento ilustra cómo, a partir de una señal acústica analógica, el sonómetro aplica una cadena de procesos electrónicos y digitales estandarizados hasta representar un valor en dB SPL con la máxima precisión. La comprensión de este flujo resulta especialmente útil al compararlo con sistemas embebidos de bajo coste, como los basados en la M5Stack Core2, en los cuales se reproducen de manera simplificada las mismas etapas: captación acústica mediante micrófonos MEMS, conversión ADC, procesamiento digital (filtrado, ponderación y cálculo RMS) y envío de resultados a bases de datos en tiempo real.

2.2 Percepción humana del nivel de presión sonora

El concepto de decibelios (dB SPL) no solo tiene un fundamento físico basado en la presión acústica, sino también una relación directa con la manera en que el oído humano percibe el sonido. El sistema auditivo no responde de forma lineal a las variaciones de presión sonora, sino de manera logarítmica, lo que justifica el uso de la escala en decibelios para describir la intensidad sonora. Gracias a esta relación logarítmica, un aumento de 20 dB SPL corresponde a una multiplicación por 10 de la presión acústica, pero se percibe únicamente como un incremento subjetivo de “el doble de volumen aproximado”.

La sensibilidad del oído humano varía con la frecuencia. El umbral de audición es más bajo en la zona de 2–5 kHz (zona más sensible) y más alto en frecuencias muy bajas o muy altas. Esta característica fue estudiada por Fletcher y Munson en 1933, quienes desarrollaron las llamadas curvas de igual sonoridad o curvas isofónicas. Dichas curvas muestran el nivel de presión sonora necesario en cada frecuencia para que el oído humano perciba un mismo nivel subjetivo de volumen.

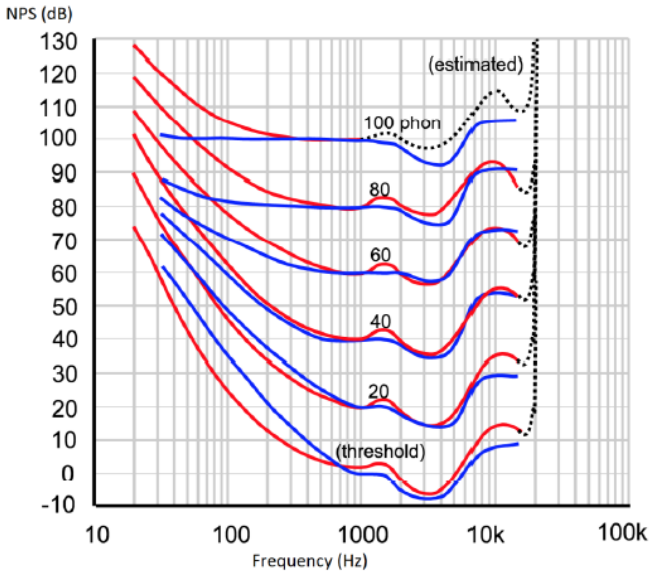


Figura 2: Curvas Isofónicas definidas por Fletcher y Munson en 1933, fuente: Trabajo final de Grado sobre diseño electrónico

En estas curvas se observa, por ejemplo, que un tono de 50 Hz necesita alrededor de 50 dB SPL para percibirse con la misma sonoridad que un tono de 1 kHz a 0 dB SPL. Este comportamiento explica por qué un sonómetro profesional incluye filtros de ponderación frecuencial (A, C o Z): el filtro A se utiliza para simular la percepción humana a niveles moderados de ruido, el filtro C para niveles altos, y el filtro Z para una medida lineal sin ponderación.

En resumen, el uso de decibelios y de filtros de ponderación no responde únicamente a la necesidad de expresar niveles de presión sonora en una escala manejable, sino a la intención de aproximar la medición técnica a la forma real en que los seres humanos percibimos el sonido.

3. Análisis del problema y especificaciones del sistema

El problema que se pretende abordar con este Trabajo de Fin de Grado surge de la necesidad de disponer de un sistema de monitorización acústica en tiempo real, que sea económico, portátil y fácilmente escalable para múltiples escenarios. En entornos urbanos, industriales o educativos, el control del nivel sonoro se ha convertido en una herramienta esencial para la gestión ambiental y la prevención de la contaminación acústica, uno de los principales factores de estrés en las ciudades modernas según la Organización Mundial de la Salud. Sin embargo, la mayoría de las soluciones existentes en el mercado presentan dos limitaciones principales: el alto coste de adquisición y mantenimiento, y la dificultad de integración con sistemas abiertos y personalizados.

Este proyecto parte del uso del M5Stack Core2, un dispositivo IoT compacto basado en ESP32, que permite leer valores analógicos de un micrófono incorporado y procesarlos para obtener el nivel de presión sonora (SPL) en decibelios (dB). Además, este dispositivo cuenta con conectividad WiFi, lo que posibilita la transmisión de los datos a una base de datos MySQL remota, permitiendo su almacenamiento, consulta y visualización mediante una interfaz web. Esta propuesta permite cubrir una necesidad no resuelta: la creación de un sistema low-cost, extensible y conectado, que pueda ser desplegado en escenarios donde los dispositivos profesionales resulten inasumibles por coste o complejidad.

Uno de los principales objetivos de este proyecto es el control del ruido ambiental durante el horario nocturno, con el fin de detectar y prevenir niveles sonoros que superen los umbrales permitidos por la normativa vigente. Para ello, se plantea el desarrollo e implementación de un medidor de decibelios basado en un dispositivo embebido, capaz de capturar muestras acústicas del entorno, procesarlas en tiempo real y alertar cuando se exceda un límite de ruido preestablecido.

La contaminación acústica es un problema que afecta directamente al bienestar y a la salud de las personas, especialmente durante la noche, cuando el descanso puede verse interrumpido por ruidos persistentes o puntuales de alta intensidad. Según establece la Organización Mundial de la Salud (OMS) y se recoge en diversas normativas a nivel estatal y municipal, el nivel de presión sonora recomendado durante la noche no debe superar los 45 dB en el interior de las viviendas y los 55 dB en el exterior (en fachada).

En el caso concreto de España, el Real Decreto 1367/2007, que desarrolla la Ley del Ruido (Ley 37/2003), establece los valores límite de inmisión según zonas de sensibilidad acústica y franjas horarias. Para la franja nocturna (de 23:00 a 7:00 horas), los niveles máximos permitidos en zonas residenciales oscilan generalmente entre los 40 y 50 dB, dependiendo del entorno. Este rango de entre 40 y 50 dBs no sería el que se mide desde la propia fuente de ruido, si no en las propias casas o en las zonas urbanas donde la gente habite.

A partir de estos valores, se determina que el medidor debe ser capaz de:

· Capturar señales acústicas del entorno con suficiente resolución y sensibilidad, para ser capaz de hacer mediciones del ruido ambiental nocturno.

· Calcular el nivel de presión sonora efectivo (RMS) y convertirlo a decibelios.

· Comparar el valor calculado con un umbral ajustable, basado en la normativa de ruido máximo aceptable en horario nocturno.

· Detectar cuándo se excede dicho umbral, activando una alerta o registrando el evento.

En la siguiente imagen [Figura 2], se pretende dar una pequeña idea de cuántos decibelios corresponden con diferentes sonidos.

Figura 3: Distintos sonidos y sus niveles de ruido, fuente ecoacustika.com

Resulta importante recalcar que, estos datos que aparecen en la imagen surgen de hacer medidas de intensidad acústica con el micrófono a escasos centímetros de la fuente emisora, por lo que, el sonido que se aprecian en las zonas urbanas, donde se aplica la normativa, sufre una atenuación debido a la distancia recorrida entre la fuente y las zonas donde se debe reducir el ruido.

4. Diseño de la solución

Una vez definida la planificación por fases y establecidos los objetivos técnicos del sistema, este apartado describe con detalle el proceso de desarrollo llevado a cabo en cada una de ellas. La implementación del proyecto se realizó de forma secuencial, siguiendo un enfoque iterativo y orientado a la validación continua de cada funcionalidad. En primer lugar, se abordó la Fase 1, centrada en la obtención y cálculo del nivel de presión sonora (dB) desde el micrófono del M5Stack Core2. A continuación, en la Fase 2, se integró el sistema con un entorno local de base de datos y servidor web para almacenar y visualizar las mediciones de forma estructurada. Finalmente, en la Fase 3, se amplió la funcionalidad del sistema permitiendo la visualización remota en la nube mediante la plataforma ThingSpeak. A lo largo de este apartado se explicarán con detalle las decisiones técnicas tomadas, las herramientas utilizadas y los retos encontrados en cada etapa del desarrollo.

4.1 Implementación del medidor de decibelios con el M5 Stack Core 2

El primer paso que se llevó a cabo en el desarrollo del proyecto fue investigar las distintas formas de programar el dispositivo M5Stack Core2. Existen varias alternativas disponibles para este propósito, como MicroPython, UIFlow o incluso Visual Studio Code con PlatformIO. Sin embargo, tras valorar las diferentes opciones, se decidió por optar por el entorno de desarrollo Arduino IDE, debido a su amplia comunidad, su simplicidad y la gran cantidad de recursos y documentación disponibles para esta placa en particular.

Gracias a la documentación oficial de los propios desarrolladores el m5core 2 [4] donde se indican los aspectos más relevantes del m5core 2 [Figura 4], además de que se indican los distintos entornos donde se puede programar su microcontrolador, se pudo hacer una muy buena estimación inicial, en la que finalmente se tomó la decisión de utilizar Arduino para este proyecto.

Una vez establecido el entorno de trabajo, comenzó el estudio del funcionamiento general del M5Stack Core2. Se descubrió que este dispositivo se basa en un microcontrolador ESP32, concretamente el ESP32-D0WDQ6-V3 [Figura 3], un chip ampliamente utilizado en aplicaciones IoT por su capacidad de procesamiento, conectividad Wi-Fi y bajo consumo energético. A diferencia de un ESP32 genérico, el M5Core2 integra de fábrica varios módulos como una pantalla táctil, micrófono, acelerómetro, batería recargable y altavoces, todo dentro de un único encapsulado, lo cual simplifica enormemente el trabajo del desarrollador.

Este diseño evita tener que lidiar con aspectos electrónicos adicionales como la conexión de cables, la utilización de una protoboard o la definición manual de pines de entrada y salida, que serían necesarios si se estuviera trabajando con un ESP32 sin encapsulado y con módulos externos.

Figura 4: ESP32-D0WDQ6-V3. Fuente: Oceancomponents.es

Figura 5: M5Stack Core2. Fuente: DigiKey.es

Para facilitar aún más la integración con estos módulos internos, M5Stack proporciona la librería M5Unified, de la cual se hablará más adelante, que abstrae las conexiones internas y permite utilizar los periféricos del dispositivo sin preocuparse por la gestión directa de los pines. Esta librería permite, por ejemplo, inicializar la pantalla o acceder al micrófono con unas pocas líneas de código. Para entender mejor cómo estaban organizadas estas conexiones internas, realicé un estudio del esquema de hardware del dispositivo. En la parte trasera del M5Core2 se encuentra serigrafiado un diagrama del llamado M-BUS [Figura 5], que actúa como una especie de "bus interno" de interconexión entre el microcontrolador ESP32 y los distintos periféricos integrados. Este M-BUS define cómo están conectados los distintos pines del ESP32 a los módulos del sistema, permitiendo al desarrollador conocer, por ejemplo, qué pines están reservados para el micrófono, el puerto I2C o la pantalla táctil. Este diseño modular interno proporciona una arquitectura organizada y eficiente para el desarrollo de proyectos embebidos, especialmente en prototipos que requieren rapidez y facilidad de implementación.

Figura 6: M-BUS del m5core2, y el módulo MPU6886.

Fuente: docs.M5Stack.com

En esta última imagen, se puede ver, por un lado, el M-BUS con todo el sistema de conexiones. A su vez, se puede apreciar el módulo MPU6886, que contiene el micrófono SPM1423, elemento muy importante en el proyecto.

4.1.1 Instalación del entorno de trabajo

Como primer paso, se instalrá el software de Arduino, siguiendo la guía en la documentación oficial del M5Stack. El primer paso será instalar Arduino IDE a partir de la página oficial de Arduino [5]. El siguiente paso será instalar los controladores de las placas de la familia m5Stack. Estos componentes no vienen por defecto en la plataforma de m5stack, pero la herramienta de Arduino permite incluir librerías externas si se introduce la url de donde se pueden descargar, en la siguiente imagen [Figura 6] se muestra el procedimiento desde la propia herramienta Arduino IDE:

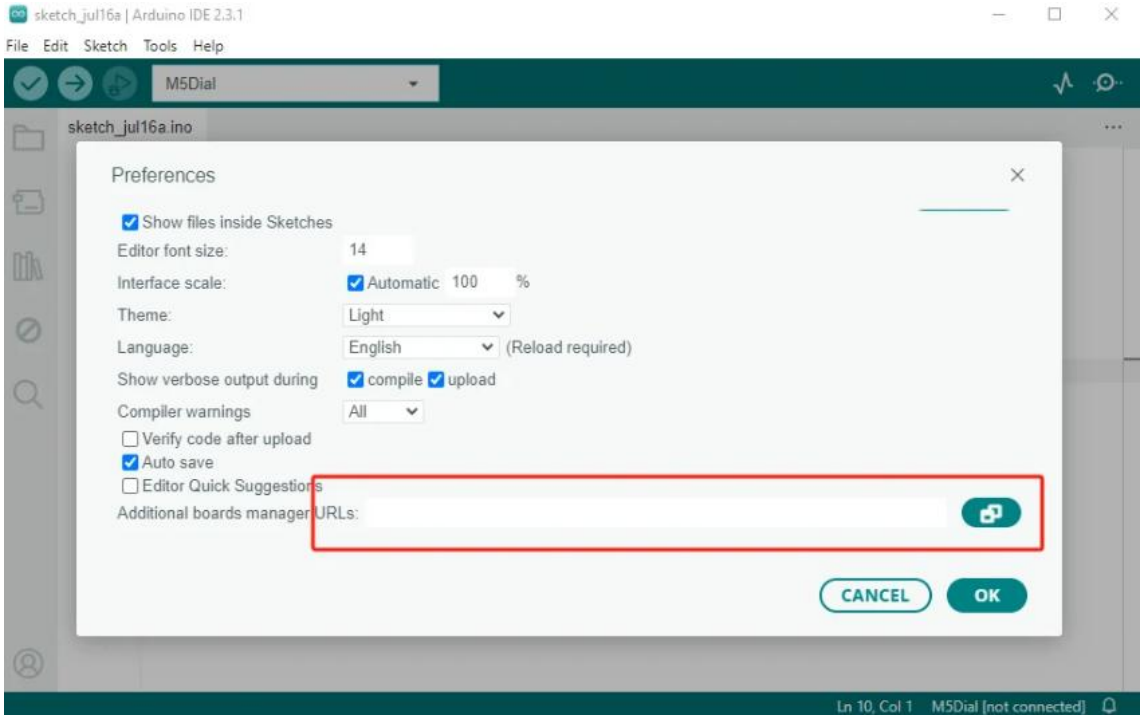


Figura 7: Pestaña de importación de librerías externas en Arduino IDE

Seguido a esto, se instalará la librería de M5Unified, esta librería dará acceso a una gran variedad de funciones para programar el microcontrolador.

4.1.2 Estudio de las librerías

El desarrollo del presente proyecto comenzó con la exploración de la librería M5Unified, que proporciona un conjunto de funciones específicas para el manejo de los periféricos integrados en la placa M5Stack Core2. Entre los diferentes ejemplos disponibles, se trabajó especialmente con el proyecto Microphone.ino, donde se encontró la función M5.Mic.record().

Esta función resulta fundamental, ya que permite acceder directamente a las muestras digitales obtenidas por el micrófono integrado, almacenándolas en un buffer de memoria para su posterior procesamiento. Para comprender en detalle el funcionamiento de dicha función, es necesario analizar los procesos internos que tienen lugar en la captura de la señal acústica mediante el micrófono digital del dispositivo.

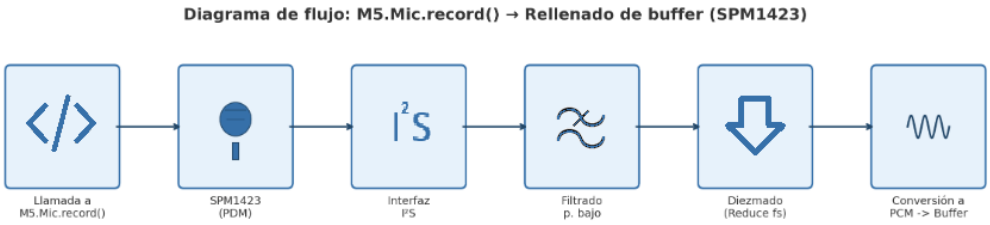
El M5Core2 incorpora de fábrica un micrófono MEMS digital modelo SPM1423 de la marca Knowles, cuyo principio de funcionamiento se basa en la detección de variaciones de presión sonora que hacen vibrar una membrana sensible. Estas variaciones se traducen en cambios de capacitancia dentro del transductor MEMS, generando una señal eléctrica proporcional a la onda acústica incidente. A diferencia de un micrófono analógico, donde la señal se entrega en forma de voltaje continuo, el SPM1423 dispone de un conversor interno que modula esta señal analógica en una forma digital, concretamente utilizando la técnica PDM (Pulse Density Modulation). En esta representación, la información de amplitud del sonido no se codifica en valores binarios discretos como en un PCM convencional, sino en la densidad relativa de unos y ceros en un flujo de pulsos de alta frecuencia. Cuando la presión acústica es positiva respecto al nivel de referencia, predominan los unos en el tren digital, mientras que en los valores negativos aumenta la densidad de ceros.

La función M5.Mic.record() se encarga de gestionar todo el proceso de adquisición de datos a partir de este flujo PDM. El primer paso que ejecuta el sistema es activar la interfaz digital que comunica el micrófono con el microcontrolador, la cual en este caso está basada en el protocolo I²S (Inter-IC Sound). Este estándar resulta especialmente adecuado para transportar señales de audio digital de alta velocidad, y en el caso del PDM se utiliza en un modo específico que permite leer directamente el flujo de pulsos generado por el micrófono. Sin embargo, la señal PDM bruta no es aún adecuada para su uso en aplicaciones de análisis, ya que contiene una elevada tasa de muestreo (del orden de varios megahercios) y requiere ser transformada a un formato PCM estándar que pueda representar con precisión la amplitud de la señal acústica en el dominio temporal.

Para llevar a cabo esta conversión, el firmware del M5Core2 ejecuta un proceso que combina filtrado y diezmado. En primer lugar, el flujo de pulsos PDM se somete a un filtro paso bajo digital, cuya finalidad es eliminar las componentes de alta frecuencia que se encuentran por encima del rango de interés audible (20 Hz – 20 kHz). Este filtrado es necesario debido a que el PDM codifica la información útil en el espectro de baja frecuencia, mientras que la mayor parte de la energía digital asociada al patrón de pulsos se sitúa en frecuencias muy superiores. Tras aplicar este filtrado, se procede a la etapa de diezmado, en la cual se reduce drásticamente la frecuencia de muestreo, pasando de varios megahercios a una tasa en el orden de decenas de kilohercios (típicamente 16 kHz, 22,05 kHz o 44,1 kHz, dependiendo de la configuración). Este proceso no solo comprime la cantidad de datos generados, sino que además transforma la señal en un formato PCM convencional de 16 bits con signo, mucho más manejable para cálculos matemáticos posteriores.

El resultado de esta cadena de procesamiento es una serie de valores enteros que representan las amplitudes instantáneas de la onda sonora en el dominio digital. Estos valores son los que la función M5.Mic.record() copia directamente en un buffer de memoria RAM previamente definido por el usuario. Una vez disponible este buffer, el programador puede aplicar sobre él diversos algoritmos de análisis, tales como el cálculo de la RMS (Root Mean Square) para obtener niveles de presión sonora relativos, la transformación de Fourier para análisis espectral, o la integración con rutinas de envío de datos a servidores remotos. De esta forma, el uso del micrófono digital SPM1423 junto con la función M5.Mic.record() constituye la base sobre la cual se construye la lógica de este proyecto, permitiendo que un dispositivo compacto como el M5Core2 pueda capturar y procesar sonido en tiempo real.

En la siguiente imagen, se explica este proceso de forma sencilla mediante un diagrama de flujo:



Aquí se puede ver cómo hacer la llamada desde el entorno Arduino a la función descrita:

M5.Mic.record(data, record_length, record_samplerate)

De donde se puede hacer una definición de cada input que necesita dicha función:

Data = buffer de datos donde se realiza la escritura

Record_length = Número de muestras que se toma por captura

Record_samplerate = Frecuencia de muestreo a la que se capturan las muestras

Para la adquisición de señal en este proyecto se ha optado por una frecuencia de muestreo de (Fs) =16 kHz y por bloques de N=1024 muestras en cada llamada a M5.Mic.record(). Esta elección responde a un compromiso entre cobertura de banda útil, latencia, coste computacional y uso de memoria en el M5Core2. Con Fs=16 kHz el límite de Nyquist queda en fNyquist= Fs/2 =8 kHz, lo cual permite capturar con fidelidad la mayor parte de la energía relevante en entornos urbanos (voz, tráfico, sirenas y eventos transitorios) sin necesidad de procesar tasas de muestreo mucho más altas que incrementarían notablemente el volumen de datos y la carga sobre la CPU y la comunicación Wi-Fi. Por otro lado, un bloque de N=1024 muestras implica una ventana temporal T = N / Fs = 1024/16000 = 0,064s (12,5 ms), proporcionando una latencia muy baja adecuada para la detección rápida de eventos y para actualizaciones frecuentes hacia la base de datos.

4.1.3 Cálculo de la estimación de dB SPL

Una vez obtenidas las muestras digitales mediante la función M5.Mic.record(), el siguiente paso consiste en realizar un tratamiento matemático de dichos datos con el objetivo de estimar el nivel de presión sonora (SPL, *Sound Pressure Level*) expresado en decibelios. Para ello, se ha implementado un procedimiento dividido en varias etapas, que se detallan a continuación.

En primer lugar, se procede a calcular el valor cuadrático medio (*Root Mean Square*, RMS) de la señal, eliminando previamente cualquier posible componente de continua (DC offset) que pueda sesgar el resultado. El cálculo del RMS sobre un bloque de N muestras digitalizadas x[n] se realiza mediante la expresión:

$$Xrms = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n] - \mu)^2}$$

donde μ representa la media de todas las muestras del bloque, utilizada para centrar la señal en torno a cero.

Posteriormente, este valor RMS se normaliza respecto al valor máximo admisible por el conversor digital del micrófono (16 bits con rango [−32768,32767]), obteniendo así una magnitud en unidades relativas de la escala digital, denominada dBFS (*decibels relative to Full Scale*). El cálculo se expresa mediante:

$$L_{dBFS} = 20 * log_{10}\left(\frac{Xrms}{X_{FS}}\right)$$

donde X_{FS}=32768 corresponde al valor de referencia de escala completa en el dominio digital. Una vez obtenido el nivel en dBFS, se lleva a cabo la conversión a unidades de presión sonora (dB SPL). Para ello, se utiliza la sensibilidad especificada por el fabricante en el *datasheet* del micrófono SPM1423. Dicha sensibilidad se proporciona como la relación entre un nivel de presión sonora de referencia, normalmente 94 dB SPL a 1 kHz, y la correspondiente salida en dBFS registrada por el dispositivo. En este caso, la relación indicada es de:

$$Lref = 94\ dB\ SPL \leftrightarrow Lmic = -22dBFS$$

2. ACOUSTIC & ELECTRICAL SPECIFICATIONS

TEST CONDITIONS: 23 ±2°C, 55±20% R.H., V_{DD}=1.8 V, F_{CLOCK}=2.4 MHz, SELECT pin grounded, no load, unless otherwise indicated

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Supply Voltage[1]	V_{DD}		1.6	-	3.6	V
Supply Current[1,2,3]	I_{DD}		-	500	600	µA
Sleep Current[3]	I_{SLEEP}	F_{CLOCK} < 1 kHz	-	25	50	µA
Sensitivity[1]	S	94 dB SPL @ 1 kHz	-25	-22	-19	dBFS
Signal to Noise Ratio	SNR	94 dB SPL @ 1 kHz, A-weighted	60	61.5	-	dB(A)
Total Harmonic Distortion	THD	94 dB SPL @ 1 kHz, S = Typ	-	0.25	0.35	%
Acoustic Overload Point	AOP	10% THD @ 1 kHz, S = Typ	112	115	-	dB SPL
Power Supply Rejection Ratio	PSRR	200 mVpp sinewave @ 1 kHz	-	48	-	dBV /FS
Power Supply Rejection	PSR	100 mVpp square wave @ 217 Hz, A-weighted	-	-80	-	dBFS (A)
DC Output		Fullscale = ±100	-	4	-	% FS
Directivity			Omnidirectional			
Polarity		Increasing sound pressure	Decreasing density of 1's			
Data Format			½ Cycle PDM			
Logic Input High	V_{IH}		0.65xV_{DD}	-	3.6	V

www.knowles.com

Figura 8: Especificaciones del datasheet del SPM1423, fuente: knowles.com

A partir de esta información, se define un factor de calibración K que permite convertir cualquier medida de dBFS en su correspondiente valor aproximado en dB SPL. Dicho factor se obtiene como:

$$K = L_{ref} - L_{mic}$$

En este caso:

$$K = 94 - (-22) = 116$$

Finalmente, la estimación del nivel de presión sonora se obtiene aplicando la siguiente relación:

$$L_{dB\ SPL} = L_{dBFS} + K$$

De esta manera, cada bloque de muestras capturado por el micrófono es transformado en un valor de nivel sonoro expresado en dB SPL, aproximando así el comportamiento de un sonómetro básico. Es importante señalar que esta calibración es de carácter teórico, ya que se ha realizado directamente a partir de la información proporcionada por el fabricante y no mediante un proceso experimental con una fuente de presión sonora de referencia. Por este motivo, los valores obtenidos constituyen una aproximación que, aunque útil para el propósito del proyecto, no debe ser considerada como una medida certificada de nivel de presión sonora.

4.1.4 Codificación final en Arduino

El código desarrollado en el dispositivo M5Stack Core2 con el micrófono digital SPM1423 tiene como finalidad capturar bloques de muestras de audio y calcular, a partir de ellas, una estimación del nivel de presión sonora en dB SPL. A continuación, se explica detalladamente cada conjunto de instrucciones implementadas:

1. Configuración inicial

En la cabecera del código se definen los parámetros principales: la frecuencia de muestreo (SAMPLERATE = 16000 Hz), el número de muestras por bloque (RECORD_LEN = 1024), así como las constantes necesarias para la conversión entre valores digitales y niveles de presión sonora:

```
const float MIC_SENS_dBFS = -22.0f;   // Sensibilidad del micrófono en dBFS
const float REF_dBSPL     = 94.0f;    // Nivel de referencia acústico
const double FS_REF       = 32768.0;  // Valor máximo representable en int16
```

La sensibilidad del micrófono (extraída del datasheet) establece la relación entre una señal acústica conocida (94 dB SPL a 1 kHz) y el nivel digital que genera el micrófono en dBFS.

Con FS_REF se define el valor máximo absoluto del ADC digitalizado (±32768 en int16_t).

2. Captura de muestras de audio

El M5Core2 utiliza su micrófono interno para registrar bloques de audio de 1024 muestras a 16 kHz, almacenados en un búfer:

```
M5.Mic.record(buffer, RECORD_LEN, SAMPLERATE);
```

Cada muestra corresponde a un valor entero (int16_t) proporcional a la presión sonora captada en ese instante.

3. Eliminación de componente DC (centrado de señal)

El micrófono puede introducir un desplazamiento respecto al cero (offset o componente DC), lo que alteraría el cálculo del RMS. Para corregirlo, en la función calculateRMS primero se calcula la media aritmética de todas las muestras:

```
double sum = 0.0;
for (size_t i = 0; i < len; ++i) sum += (double)samples[i];
double mean = sum / (double)len;
```

Luego, a cada muestra se le resta esta media antes de acumular los cuadrados:

```
double v = (double)samples[i] - mean;
acc += v * v;
```

De este modo, la señal queda centrada en torno a cero, eliminando la influencia de posibles sesgos eléctricos o de captura.

4. Cálculo del valor cuadrático medio (RMS)

Una vez corregida la señal, se calcula la potencia media efectiva mediante el RMS:

```
double mean_sq = acc / (double)len;
return sqrt(mean_sq);
```

El RMS refleja la magnitud promedio de la onda sonora y constituye la base para la conversión a decibelios.

5. Conversión a dBFS

Con el RMS ya obtenido, se normaliza respecto al valor máximo posible (FS_REF) y se pasa a escala logarítmica:

```
double dBFS = 20.0 * log10(rms / FS_REF);
```

Esto proporciona el nivel en decibelios relativos al máximo digital posible (dBFS).

6. Conversión a dB SPL reales

Para obtener un valor acústico real, se aplica la referencia proporcionada por el fabricante del micrófono:

```
double K = REF_dBSPL - MIC_SENS_dBFS;
dBSPL = dBFS + K;
```

El datasheet indica que –22 dBFS corresponden a 94 dB SPL. El dato de -22 dBs se introduce en el código como un parámetro, y la siguiente sentencia calcula la constante de calibración K. Con ello, se transforma el nivel relativo (dBFS) en un nivel absoluto de presión sonora en dB SPL.

7. Visualización del resultado

Finalmente, el valor calculado se muestra en pantalla:

```
M5.Lcd.printf("Nivel de ruido:\n %.2f dB SPL", dBSPL);
```

4.2 Almacenamiento local y visualización web

Con el objetivo de almacenar y visualizar de forma estructurada los datos recogidos por el sistema de medición acústica, se desarrolló una solución de almacenamiento local y visualización web utilizando herramientas de servidor local.

Esta fase del proyecto consistió en implementar una arquitectura básica cliente-servidor que permitiera guardar los valores registrados en una base de datos local y, posteriormente, mostrarlos en una página web dinámica.

En primera instancia, se hizo una pequeña investigación en internet sobre diferentes herramientas donde se pudiera crear una base de datos sencilla, y se pudiera conectar directamente con una página HTML en local, y finalmente, se optó por utilizar XAMPP [Figura 9], como entorno de desarrollo, el cual ofrece una instalación integrada de Apache, MySQL y PHP, facilitando el despliegue de servicios web en el entorno local del equipo. Este conjunto de herramientas no solo permitió validar el correcto envío de datos desde el dispositivo M5Stack Core2 hacia una base de datos MySQL, sino también permitió programar una interfaz de consulta visual que se puede abrir desde el propio navegador, y permite visualizar los datos medidos al momento.

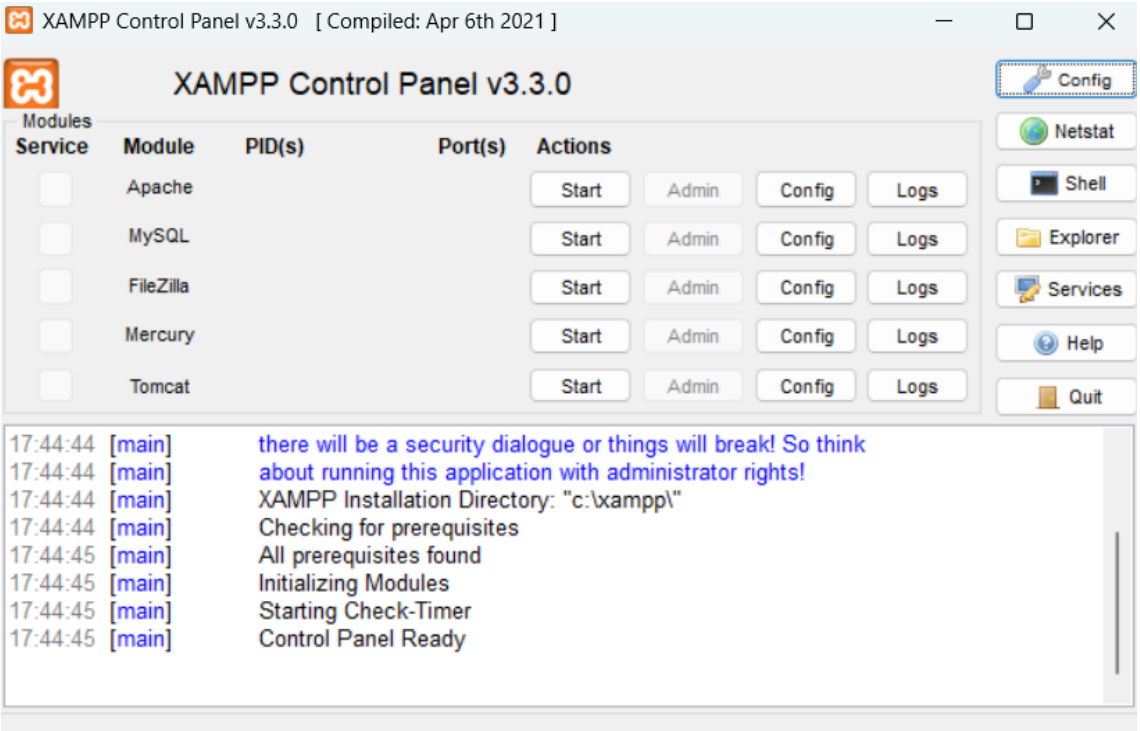


Figura 9: Panel de control de XAMPP

Existe mucha documentación en la plataforma de YouTube [6] donde explican varios de los aspectos más básicos y relevantes para aprender desde cero las mecánicas de XAMPP.

4.2.1 Configuración de la base de datos en XAMPP

Para gestionar y almacenar los datos recibidos por el dispositivo de medición, se optó por utilizar el entorno de desarrollo XAMPP, que integra el servidor web Apache y el sistema gestor de bases de datos MySQL, entre otros componentes. A través de la interfaz phpMyAdmin incluida en XAMPP, se creó una base de datos local específica para este proyecto.

Dentro de esta base de datos, que se le definió en nombre de mediciones_db, se diseñó una tabla que permitiera almacenar los valores enviados por el M5Stack Core2, junto con la fecha y hora de cada registro. La estructura de esta tabla fue pensada para ser sencilla pero funcional, permitiendo almacenar tanto datos de magnitud sonora como información adicional asociada al registro.

Para poder crear esta base de datos, se entró en la herramienta phpMyAdmin, que se accede introduciendo en el explorador la siguiente url: http://localhost/phpmyadmin.

Para acceder a la base de datos desde fuera de phpmyadmin (por ejemplo, desde el código) se ha definido un usuario de nombre "root" y sin contraseña, tal y como viene por defecto en la configuración de XAMPP

A través de esta página se podrá acceder al gestor de la base de datos MYSQL del servidor APACHE, pero es importante recalcar que estos dos servicios deben estar activos en el panel de control que se ha visto en la Figura 9.

El diseño de la base de datos que se ha diseñado tiene la siguiente estructura [Figura 10]:

Estructura de tabla | Vista de relaciones

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
☐	2	ubicacion	varchar(16)	utf8mb4_general_ci		Sí	*NULL*		
☐	3	decibelios	decimal(5,2)			Sí	*NULL*		
☐	4	fecha	timestamp			No	current_timestamp()		

Figura 10: Estructura de la base de datos en XAMPP

En esta tabla se puede ver cuatro tipos de datos, cada uno de ellos con una función muy específica:

· ID: Es un valor que añade la base de datos a cada registro que se recibe y se guarda, se trata de una clave primaria que no se puede repetir, es un valor numérico "int" que sirve como identificador único, y sirve para poder darle una identidad única a cada muestra que se escribe. Es un elemento muy usado en MYSQL, y es muy sencillo de utilizar, ya que lo asigna la propia base de datos.

·Ubicación: Se trata de un valor de texto "String" se utiliza para definir la ubicación donde se hará la medida, este valor se introducirá por el código Arduino, y será constante durante toda la ejecución de este. De esta forma, una vez se vaya a visualizar los datos, se podrá identificar dónde se hizo la medida.

·Decibelios: El elemento más importante de la tabla, es un valor tipo decimal de 5 dígitos, siendo los dos últimos los decimales. Se usará este tipo de dato para almacenar los decibelios registrados por el M5Core2, más adelante se profundizará en cómo enviar la petición de inserción de datos desde el m5Core2 a la base de datos.

·Fecha: Otro dato de suma importancia, se aprovecha la función current_timestamp() de MYSQL para añadir un campo donde se introduce la fecha y la hora en cada muestra que se registra . Este campo, al ser la propia base de datos quien lo añade, simplifica las cosas en el código del M5Core2.

En esta tabla se puede apreciar mejor la estructura de la base de datos.

Nombre del dato	Formato	De donde proviene	Ejemplo
id	int(11) Primary key	MYSQL	00000000114
ubicacion	varchar(16)	M5Core2	Casa de Néstor
decibelios	decimal(5,2)	M5Core2	34.12
fecha	timestamp	MYSQL	2001-11-06 12:27:45

4.2.2 Codificación en Arduino y PHP

Para esta fase, se inicia con el código previamente desarrollado para la medición de decibelios en tiempo real mediante el módulo M5Stack Core2. A dicho código se le incorporó nuevas funcionalidades orientadas al envío de datos a una base de datos local, alojada en un servidor configurado con XAMPP. Esto implicó la integración de conectividad WiFi y la generación de solicitudes HTTP tipo POST hacia un script PHP ubicado en el servidor. Este script PHP se encargaría de recibir los datos, crear la conexión con la MYSQL, y realizar la petición de escritura por vía POST.

El primer paso fue configurar la conexión WiFi [Figura 11], definiendo el SSID y la contraseña de la red local a la que el dispositivo debía conectarse. Una vez establecida esta conexión, el dispositivo queda preparado para iniciar la recogida de datos del micrófono integrado. El muestreo se realiza de forma continua, acumulando valores absolutos para calcular posteriormente una media representativa del nivel de señal registrada.

```
#include <M5Unified.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <cmath>

// Configuración WiFi
const char* ssid = "Redmi Note 12";
const char* password = "romaguera";
String serverUrl = "http://192.168.58.177/guardar_medicion2.php";
```

Figura 11: Fragmento de código donde se configura conexión a la red WIFI

En este fragmento del código, se puede ver cómo se define el ssid y la contraseña de la red local, en este caso se utiliza un punto de acceso inalámbrico Wifi donde un teléfono actúa con router,, pero también se podría utilizar la red Wifi convencional.

A su vez, se puede ver cómo se guarda el valor serverUrl1 con la dirección IP del PC, y el archivo PHP, cuyo nombre es guardar_medicion2.php. Esto es así porque, la IP del servidor local es la que aparece en la imagen, y es, a su vez, la IP del PC, que es la máquina que está hosteando el servidor Apache gestionado en XAMPP. A su vez, dentro de la carpeta de archivos de XAMPP, existe un directorio llamado “htdocs” donde se pueden guardar los scripts php que se quedarán activos mientras el servidor esté encendido. De esta forma, si se introduce la url con el formato de la IP + nombre del archivo que se ha guardado en htdocs, el programa puede hacer uso de ese script.

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  M5.Display.print(".");
}
```

Figura 12: Fragmento de código Arduino donde se crea un bucle que tratará de conectarse a la red wifi

En este bucle while [Figura 12], se realizará la conexión al WiFi, y se mantendrá intentando hacer la conexión hasta que la condición de haber establecido conexión (WL_CONNECTED) se cumpla.

Una vez se haya establecido la conexión wifi, el programa se encargará de tomar muestras y calcular la estimación de dBs que se ha establecido en el programa original, conservando el valor calculado. Este valor resultante se envía al servidor local mediante una solicitud HTTP POST, que incluye también un identificador de ubicación. En el servidor, un script PHP se encarga de almacenar los datos recibidos en la base de datos previamente configurada.

Cuando ya se tenga el valor de decibelios calculado, se envía fácilmente con la función sendtoserver(), función que previamente se ha podido configurar indicando los datos del servidor Apache.

El script PHP se codificó para que se encargara de recibir los datos que le envía el dispositivo M5Stack Core2 a través de una petición HTTP de tipo POST. En concreto, espera recibir dos datos: el nivel de decibelios medido y la ubicación del dispositivo. Una vez recibidos, los guarda en una base de datos local en XAMPP.

En primer lugar, este script se conecta a la base de datos de nombre mediciones_db que está en el servidor local (localhost) usando el usuario root sin contraseña, tal y como viene por defecto en la configuración del usuario. A continuación, se hace una comprobación de que los datos han sido recibidos correctamente: decibelios y ubicación.

Seguidamente a esto, se usa una consulta preparada para insertar los datos en la tabla mediciones, después se cierra la consulta y la conexión con la base de datos. También se ha programado la situación donde, si no se han recibido todo el conjunto de datos para la inserción, el script muestra un mensaje indicando que los datos están incompletos.

Explicado de forma sencilla, este script, que funciona dentro del servidor, funciona como un intermediario entre el M5Core2 y la base de datos.

```
17  if (isset($_POST['decibelios']) && isset($_POST['ubicacion'])) {
18      $decibelios = floatval($_POST['decibelios']);
19      $ubicacion = substr($_POST['ubicacion'], 0, 16); // Limita a 16 caracteres por seguridad
20
21      // Preparar sentencia SQL segura
22      $stmt = $conn->prepare("INSERT INTO mediciones (ubicacion, decibelios) VALUES (?, ?)");
23      $stmt->bind_param("sd", $ubicacion, $decibelios);  // s = string, d = double (float)
```

Figura 13: Peticiones POST dentro del código PHP

En estas líneas [Figura 13] del script PHP, se muestran las peticiones POST para enviar la información a la base de datos.

Ahora, ya se tiene, por una parte, un código que registra decibelios, y por otra, que es capaz de enviarlos al script PHP que se encuentra dentro del servidor. También se tiene codificado el script PHP que se encarga de hacer las inserciones de datos en MYSQL. Con esta parte del desarrollo completada, quedaría pendiente de implementar la última parte, a partir de los datos que ya se tienen guardados en la base de datos, crear una pantalla de visualización sencilla y manejable para poder acceder a los datos con facilidad. Para ello, se va a emplear una página HTML generada a partir del contenido de las tablas MYSQL.

4.2.3 Visualización de los datos mediante HTML

Para poder visualizar las mediciones almacenadas en la base de datos de forma accesible y ordenada, se ha desarrollado un script en PHP que genera dinámicamente una página web con los datos extraídos de la tabla mediciones. Esta página muestra una tabla con los niveles de decibelios registrados, su fecha de captura y la ubicación correspondiente. A continuación, se explica el funcionamiento del código:

· Conexión a la base de datos

En la primera parte del script[Figura 14] se establece la conexión con la base de datos MySQL mediante la extensión mysqli. Se configuran los parámetros necesarios: servidor, usuario, contraseña y nombre de la base de datos.

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mediciones_db";
$conn = new mysqli($servername, $username, $password, $dbname);
```

Figura 14: Fragmento del código del Script generador del HTML

Además, se incluye una verificación de conexión para detener el script si ocurre algún error.

· Consulta SQL

A continuación, se ejecuta una consulta SQL [Figura 15] que selecciona los campos decibelios, fecha y ubicacion de la tabla mediciones, ordenando los resultados por fecha en orden descendente (de más reciente a más antiguo).

```php
$sql = "SELECT decibelios, fecha, ubicacion FROM mediciones ORDER BY fecha DESC";
```

Figura 15: Fragmento del código del Script generador del HTML

·Generación del contenido HTML

El cuerpo del documento HTML [Figura 16] contiene un título principal con el nombre Historial de Mediciones, y una tabla con tres columnas, la primera con el dato ubicación, otra con el dato decibelios, y la última con la fecha y la hora.

Siguiendo con las especificaciones definidas en el apartado 3, se ha establecido un umbral de 70dBs donde, toda medición que supere dicho valor será marcada como peligrosa. Por lo tanto, se aprovechará de

Adicionalmente a eso, se ha añadido un efecto visual sencillo, aquellas columnas que superen el umbral de 70dBs, tendrán un resaltado de color rojo, además de un pequeño icono de advertencia.

Finalmente, se ha añadido un pequeño botón en la parte superior de la página web, donde se puede descargar toda la tabla generada en formato CSV, por si se desea tratar los datos desde Excel o Matlab, cualquier herramienta externa.

Historial de Mediciones

Descargar CSV

Decibelios	Ubicación	Fecha
⚠ 62.09	Casa de Nestor	20-06-25 17:28
⚠ 60.13	San Sebastian	12-12-99 12:12
⚠ 64.51	Estación Espacial	20-12-55 15:30
40	Casa de Iván	29-11-00 08:00
35	Casa de Juan	21-12-98 12:00
57.55	Casa de Antoni	04-11-00 14:00

Figura 16: Tabla resultante que se genera con el código HTML

Aquí se observa cómo se vería la tabla si se abre el HTML generado con el explorador. Cabe recalcar que se han cargado valores de prueba que no corresponden con medidas reales, pero con esta imagen se puede ver de forma gráfica lo que hace e código una vez se ejecuta.

Más adelante, en el apartado 5 de esta memoria donde se hace un conjunto de pruebas funcionales y comento los resultados, se puede ver la tabla completa con datos de mediciones reales, realizadas con el M5Core2.

4.2.4 Arquitectura final

Ahora que se han montado todas las partes de este flujo, se adjunta en la siguiente imagen [Figura 17], un diagrama donde se puede ver cómo se conectan todas las herramientas que se han diseñado entre sí:

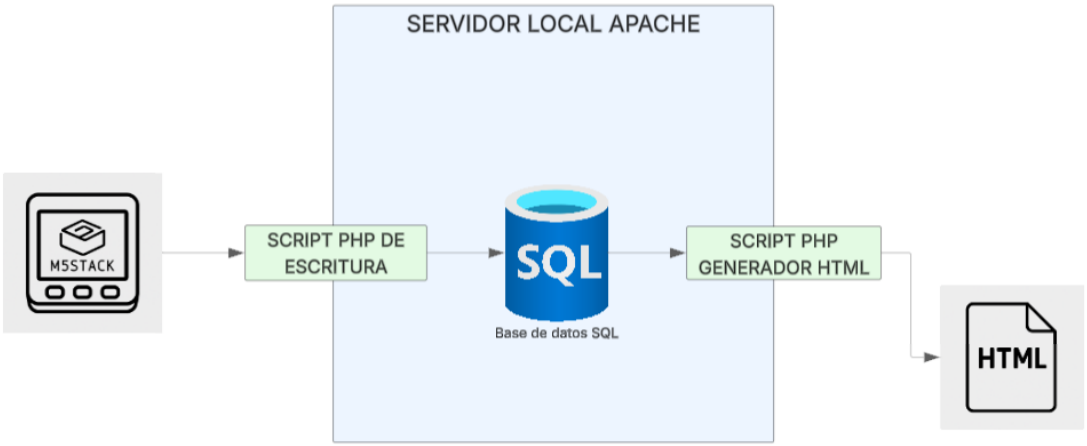


Figura 17: Diagrama de flujo del m5core2 a el doc HTML

Se puede apreciar cómo los scripts PHP se encargan de tomar los datos de fuera del servidor, registrarlos en la base de datos, y generar un documento HTML en el PC que hostea el servidor Apache. Cabe recalcar que, tanto el M5core2, como el PC que hostea el servidor, deben estar conectados a la misma red local, y se necesitará la dirección IP del PC, que será la misma que usará el servidor, para que el M5Core2 pueda conectarse al script PHP de escritura.

4.3 Visualización de mediciones mediante API web

Con el objetivo descrito anteriormente cumplido, el objetivo ahora es crear un flujo que lleve las mediciones del M5core2 a un entorno web, para ello, se dejará de usar la propia red local, y se utilizará un entorno en la nube, donde se deberá establecer una conexión para poder enviar los datos. Desde este propio entorno en la nube, también se tendrá la capacidad de visualizar los resultados.

El entorno que se ha escogido para diseñar esta funcionalidad se llama Thingspeak. Es una plataforma en línea de IoT (Internet de las Cosas) que permite recopilar, almacenar, analizar y visualizar datos en tiempo real procedentes de sensores o dispositivos conectados. Es muy utilizada para proyectos con microcontroladores como Arduino o ESP32, ya que facilita el envío de datos mediante HTTP y ofrece gráficos automáticos para su interpretación. También permite procesar la información con MATLAB y programar alertas o acciones automáticas según condiciones definidas.

El procedimiento general para conectar Arduino con ThingSpeak se puede resumir en los siguientes pasos:

1. Creación de una cuenta en ThingSpeak y configuración del canal:

El usuario debe registrarse en la plataforma y crear un nuevo canal, en el cual se definen los campos que contendrán los datos a registrar.

2. Obtención de la clave de escritura (Write API Key):

Esta clave única se genera al crear el canal y es necesaria para autorizar el envío de datos desde el microcontrolador hacia la plataforma.

3. Conexión desde el código Arduino a la red WiFi y a ThingSpeak:

En el entorno de desarrollo de Arduino se implementa un programa que conecta el dispositivo a una red WiFi, inicializa la comunicación con ThingSpeak mediante un cliente TCP, y envía los valores deseados utilizando su API. Para facilitar esta tarea, se emplea la librería oficial ThingSpeak.h.

4. Envío y visualización de datos:

Una vez conectados, los datos se transmiten periódicamente a ThingSpeak, donde quedan almacenados y disponibles para ser representados en gráficos dentro del propio panel del canal. También pueden exportarse o analizarse mediante MATLAB integrado en la plataforma.

4.3.1 Creación del canal en ThingSpeak

Lo primero de todo, se accederá a la página web de ThingSpeak [7] donde se iniciará sesión con una cuenta de MathWorks. A continuación, en el menú principal [Figura 18] se creará un nuevo canal:

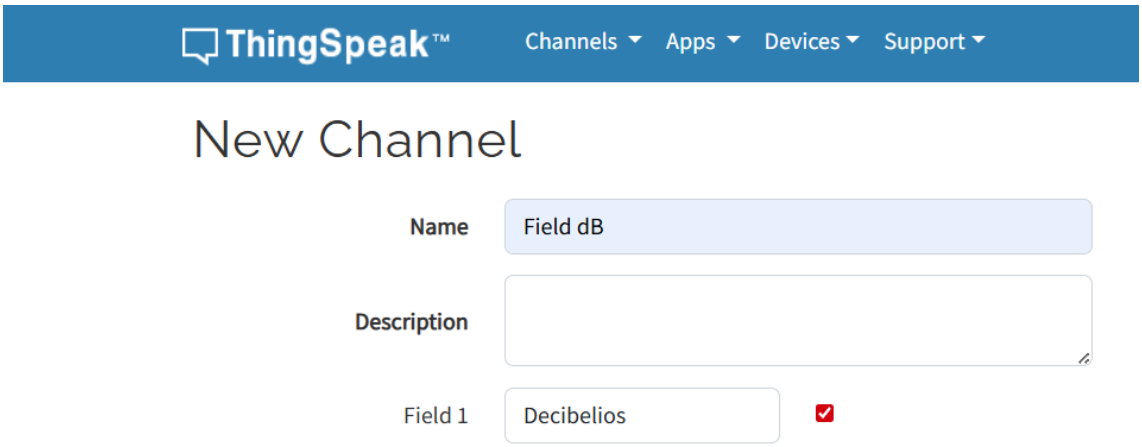


Figura 18: Creación de nuevo canal en Thingspeak, fuente: Thingspeak.com

Aquí se podrá definir qué tipo de datos se van a introducir, en este caso, se va a realizar una captura simple del dato de los decibelios, y el propio programa añadirá la fecha y la hora donde se introducen nuevas muestras, como ocurría de forma parecida con MYSQL.

Ahora ya se ha creado el canal y está configurado para que pueda recibir las medidas. Como se ha mencionado anteriormente, para poder acceder a este entorno, se realizará una conexión a través de una API (Application Programming Interface), así que, dentro del menú de opciones del canal, se generará una API Key [Figura 19] que se utilizará en el código Arduino para realizar la conexión a ThingSpeak.

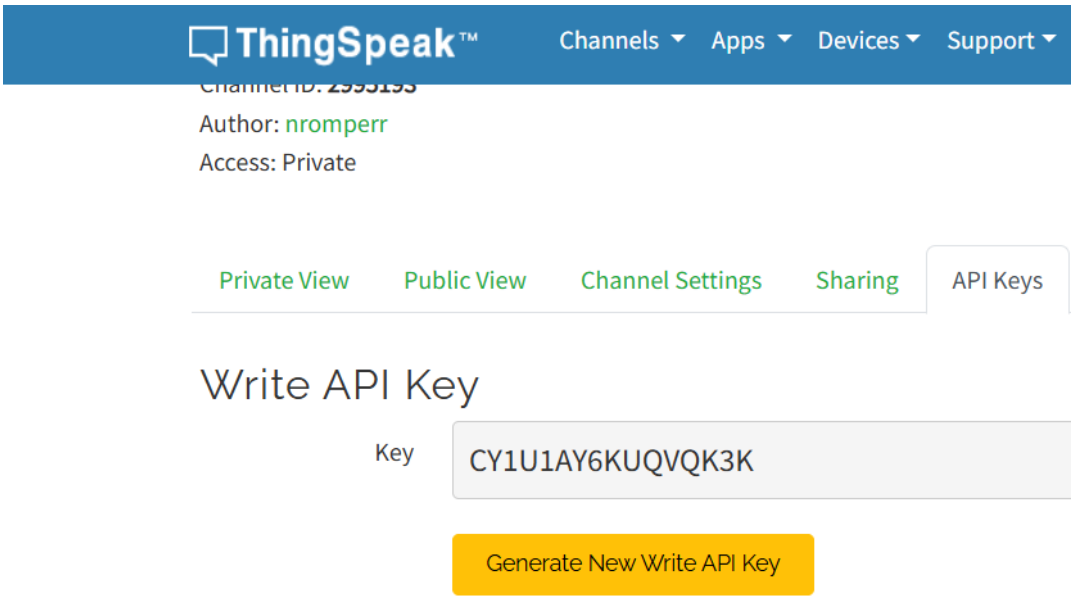


Figura 19: Creación de la API KEY, fuene: ThingSpeak.com

En esta captura, se puede ver cómo se ha generado la API Key de escritura, se utilizará para hacer la solicitud de registro de datos.

4.3.2 Codificación del envío de datos a ThingSpeak

El siguiente paso sería escribir el código para el M5Core2, para que sea capaz de hacer mediciones, y enviarlas al entorno de MathWorks.

Para ello, se volverá a partir del código inicial que realizaba la tarea de medición de dBs, pero esta vez, los valores de dBs calculados deberán ser enviados mediante la API que se generado anteriormente.

Estos cambios, reflejados en el código, se verían de la siguiente manera [Figura 20]:

```
1   #include <M5Unified.h>
2   #include <WiFi.h>
3   #include <HTTPClient.h>
4   #include <cmath>
5
6   // ===== CONFIGURACIÓN DE WIFI Y THINGSPEAK =====
7   const char* ssid = "Redmi Note 12";  // Red WiFi
8   const char* password = "romaguera";
9   const char* server = "http://api.thingspeak.com/update";
10  const String apiKey = "CY1U1AY6KUQVQK3K"; // API Key de escritura del canal
```

Figura 20: Fragmento de código donde se definen librerías, conexión wifi y url de conexión con la API key

Aquí se ver cómo se han mantenido las librerías que se han usado hasta el momento, y esta vez se están introduciendo como variables la url de la api de ThingSpeak y la clave para la conexión.

El procedimiento de envío de datos resulta bastante simple, tan solo se debe tomar el valor de decibelios que se ha calculado, y enviarlo mediante una instrucción sencilla:

```
71          String url = server;
72          url += "?api_key=" + apiKey;
73          url += "&field1=" + String(dbValue, 2);
74
75          http.begin(url);
76          int httpCode = http.GET();
```

Aquí se puede ver cómo se crea un string concatenado con el valor de decibelios, la API key y el nombre del dato definido en ThingSpeak que se ha definido como “field1”. Mediante el uso de la función http.GET() se realiza el envío de datos al canal.

4.3.3 Arquitectura final con ThingSpeak

En la siguiente imagen [Figura 21], se muestra un diagrama de flujo con todas las conexiones entre los módulos que están integrados en el diseño.

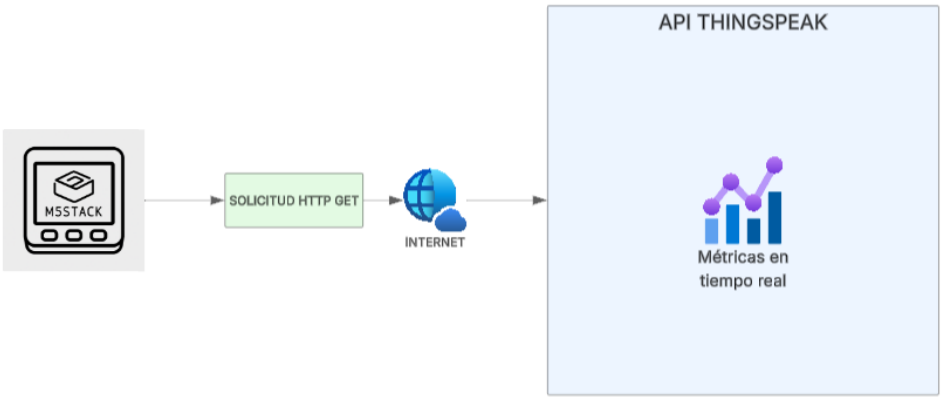


Figura 21: Diagrama de Flujo de la conexión M5core2 a API ThingSpeak

Se puede ver cómo el M5Core2 envía una solicitud HTTP GET a través de internet hacia la API, y esta se encarga de registrar los valores que van llegando.

5. Pruebas funcionales

En este apartado, se pondrán en marcha todos los elementos que se han definido a lo largo del proyecto, y se verá si responden de la manera en la que se han percibido.

Se analizarán las tres fases que se han implementado, pasando primero por la prueba del código que realiza las mediciones de decibelios, seguido a esto, se realizará la conexión al servidor local Apache, y finalmente se hará una prueba de conexión con la API de Thingspeak, y se tratará de visualizar los resultados en las gráficas que proporciona la herramienta.

5.1 Pruebas de la medición y el cálculo de decibelios

Ahora, se va a montar el código en la placa M5Core2 para analizar los resultados [Figura 22] del primer diseño que se ha implementado, es necesario que este código tenga un funcionamiento correcto, con tal de que las siguientes fases que hacen uso de este, también hagan su función tal y como se espera.

Figura 22: Muestreo en pantalla de la medición y el cálculo en dBs

Aquí se puede ver un pequeño texto en la parte central, donde se muestra el cálculo de decibelios que está siendo calculado en ese instante.

Se hicieron diferentes pruebas para comprobar el rango de intensidad acústica que el micrófono es capaz de detectar, haciendo un testeo exponiendo el M5stack core 2 a fuentes de ruido con una intensidad sonora conocida. A continuación, se describe en la siguiente tabla los resultados obtenidos en el micrófono.

Fuente de ruido	dB SPL teóricos	Db SPL medidos
Sala en silencio	30-35	52-55
Conversación normal a 1m del M5stack	60-65	62-67
Secador de pelo a 1m del M5stack	90	93
Soplido fuerte directamente en el M5stack	115	115

Se han definido cuatro niveles de intensidad acústica a medir. Primeramente, se puede observar que el micrófono SPM1423, cuando se mide una habitación vacía, el cálculo no es capaz de medir el nivel de intensidad acústica correcto. Esto puede ser debido al ruido eléctrico del propio sistema, que introduce pequeños valores en el valor RMS calculado (10-15). Una pequeña variación en los valores de RMS, cuando el sistema se encuentra en sus valores más bajos, puede repercutir en un cambio drástica en la medida final, por lo tanto, estas medidas no deben tomarse como válidas.

El cálculo final siempre dará un mínimo de 52-53 dBs, pero una vez se hace una medición por encima de estos valores, el cálculo de RMS se vuelve más preciso, debido a que el ruido real que se está midiendo sobrepasa al ruido térmico que genera el propio dispositivo. Afortunadamente, el objetivo del proyecto es mantener un margen de precisión en un umbral de 60-80dBs, debido a que se ha establecido el umbral de "aviso" en 70dBs, dentro de este rango, gracias a las mediciones siguientes, se puede ver que mantiene una precisión aceptable.

Seguido a esto, se han hecho mediciones de fuentes de 60-65 dBs y de 90dBs. Se puede apreciar cómo el micrófono ha respondido con un nivel de precisión adecuado, siendo estos márgenes, donde se hará el aviso de ruido molesto.

Posteriormente, se ha tratado de provocar la saturación del micrófono, incidiendo sobre este, a escasos centímetros, una fuente ruidosa. El objetivo era ver si el valor final de saturación se acercaba al que muestra el datasheet (120dBs) [Figura 7] Se ha podido apreciar que el dispositivo ha saturado su cálculo en 115dBs, un resultado razonable teniendo en cuenta el valor teórico.

Limitación inferior de medición del micrófono

El micrófono utilizado en este proyecto presenta un nivel de relación señal-ruido (SNR) de 61,5 dB(A) cuando se mide con un tono calibrado de 94 dB SPL. A partir de este dato, se puede estimar un nivel de ruido equivalente de entrada (EIN) de $94 - 61{,}5 = 32{,}5$ dB SPL, lo que establece teóricamente el límite inferior de medición del micrófono.

No obstante, en la práctica, el cálculo del nivel sonoro se realiza a partir del valor RMS de las muestras PCM de 16 bits. Para representar niveles sonoros de 30-35 dB SPL, el RMS debería ser aproximadamente 1-2. Sin embargo, debido al EIN del micrófono, al ruido eléctrico residual del circuito y al ruido ambiental presente durante las mediciones, el valor mínimo de RMS obtenido no puede alcanzar esos niveles teóricos. En las pruebas realizadas, el valor más bajo registrado ha sido un RMS de 18,25, lo que corresponde aproximadamente a 50,91 dB SPL.

Por tanto, la limitación práctica de medición del sistema se sitúa en torno a 51-55 dB SPL. Este valor mínimo viene determinado por la combinación del EIN del micrófono, el ruido eléctrico inherente al sistema y el ruido ambiental real presente durante las mediciones, estableciendo así un umbral por debajo del cual no es posible obtener mediciones fiables.

5.2 Pruebas con el servidor Apache y XAMPP

El siguiente paso será probar el código que se creó para hacer mediciones y enviarlas al servidor local ,para ello, se iniciarán los procesos de Apache y Mysql dentro de la herramienta XAMPP, y se cargará en el m5core2 el código [Figura 23].

Figura 23: M5core2 con la conexión a servidor local

Se ha conectado el m5core2 a la red local Wifi, y se ha cargado la dirección IP del servidor local, a continuación, se deberá revisar la base de datos MySQL para ver si los datos se están recibiendo.

Para poder visualizar el contenido de la base de datos, se entrará en el menú de phpMyAdmin.

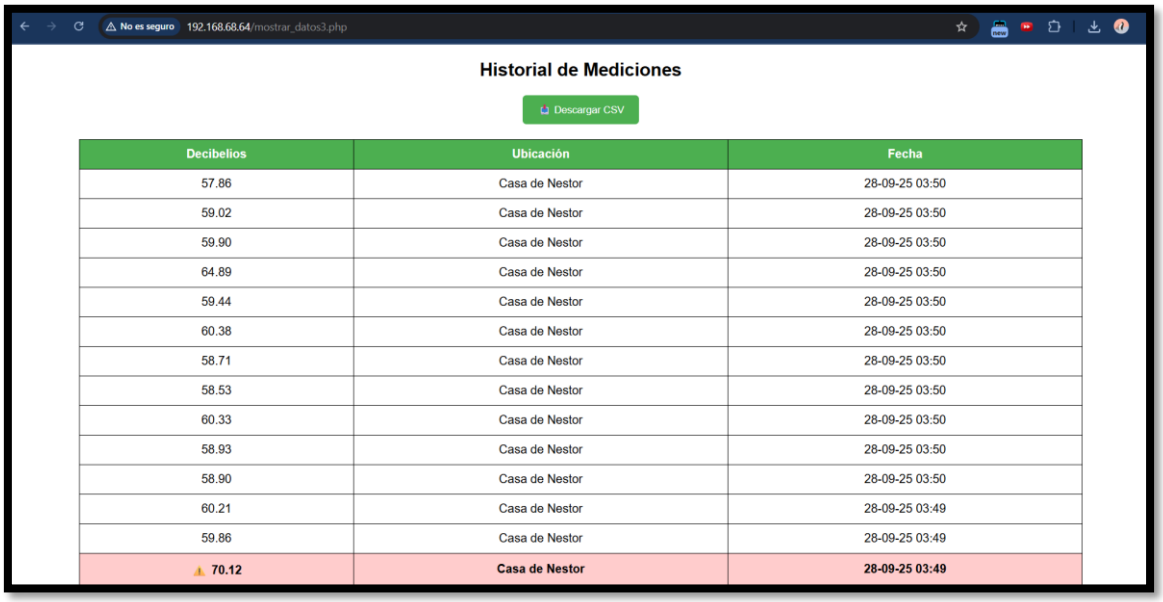
←T→		id 1	ubicacion	decibelios	fecha
Editar	Copiar Borrar	614	Casa de Nestor	62.29	2025-09-28 04:19:44
Editar	Copiar Borrar	613	Casa de Nestor	61.54	2025-09-28 04:19:38
Editar	Copiar Borrar	612	Casa de Nestor	60.86	2025-09-28 04:19:33
Editar	Copiar Borrar	611	Casa de Nestor	61.45	2025-09-28 04:19:28
Editar	Copiar Borrar	610	Casa de Nestor	60.42	2025-09-28 04:19:23
Editar	Copiar Borrar	609	Casa de Nestor	61.07	2025-09-28 04:19:18
Editar	Copiar Borrar	608	Casa de Nestor	61.40	2025-09-28 04:19:12
Editar	Copiar Borrar	607	Casa de Nestor	62.98	2025-09-28 04:19:07
Editar	Copiar Borrar	606	Casa de Nestor	61.91	2025-09-28 04:19:02
Editar	Copiar Borrar	605	Casa de Nestor	60.09	2025-09-28 04:18:57
Editar	Copiar Borrar	604	Casa de Nestor	61.63	2025-09-28 04:18:52
Editar	Copiar Borrar	603	Casa de Nestor	61.18	2025-09-28 04:18:46

Figura 24: Contenido la base de datos

En esta tabla [Figura 24] se pueden visualizar las mediciones que se han realizado durante unos minutos, se puede ver cómo se han rellenado los cuatro campos que se han definido, y los datos se corresponden con los que se han medido en el M5core2.

Finalmente, nos queda ejecutar el script PHP que nos genera el documento HTML a partir de la información guardada.

Para poder acceder a él, se accederá dentro de la carpeta donde se tenga instalada la herramienta XAMPP, a la ruta xampp/htdocs, allí se tendrñan alojados los scripts PHP. Se introducirá en el explorador la dirección http://localhost/mostrar_datos3.php, así se podrá ejecutar el script llamado mostrar_datos3, que se encarga de generar el HTML.



Decibelios	Ubicación	Fecha
57.86	Casa de Nestor	28-09-25 03:50
59.02	Casa de Nestor	28-09-25 03:50
59.90	Casa de Nestor	28-09-25 03:50
64.89	Casa de Nestor	28-09-25 03:50
59.44	Casa de Nestor	28-09-25 03:50
60.38	Casa de Nestor	28-09-25 03:50
58.71	Casa de Nestor	28-09-25 03:50
58.53	Casa de Nestor	28-09-25 03:50
60.33	Casa de Nestor	28-09-25 03:50
58.93	Casa de Nestor	28-09-25 03:50
58.90	Casa de Nestor	28-09-25 03:50
60.21	Casa de Nestor	28-09-25 03:49
59.86	Casa de Nestor	28-09-25 03:49
⚠ 70.12	**Casa de Nestor**	**28-09-25 03:49**

Figura 25: Página web HTML generada con el script PHP

En esta imagen [Figura 25] se puede ver cómo la tabla que se ha generado en HTML corresponde con los datos que se han insertado en MYSQL. También se puede ver cómo en el dato de 70.12 dBs, se ha generado la alerta visual. Se ha comprobado también si el documento CSV se generaba correctamente, y la información contenida en el archivo es la misma que se puede observar en la página, por lo que toda la implementación ha sido acorde al diseño planteado.

Adicionalmente, mientras se visualiza la página, se puede seguir manteniendo el M5core2 encendido y enviando muestras, tan solo se deberá refrescar la página para poder ver todos los registros nuevos, esto hace que la aplicación registre los datos casi el momento, con el pequeño detalle que se deberá volver a ejecutar el script PHP refrescando la página para volver a generar el HTML con los datos nuevos.

5.3 Pruebas con la API de Thingspeak

A continuación, se hará la prueba funcional con el código que se implementó para realizar la conexión a herramienta de Mathworks.

Se carga el código Arduino en el M5Core2, y se prueba a entrar en la página web de ThingSpeak [Figura 26] para poder acceder al canal.

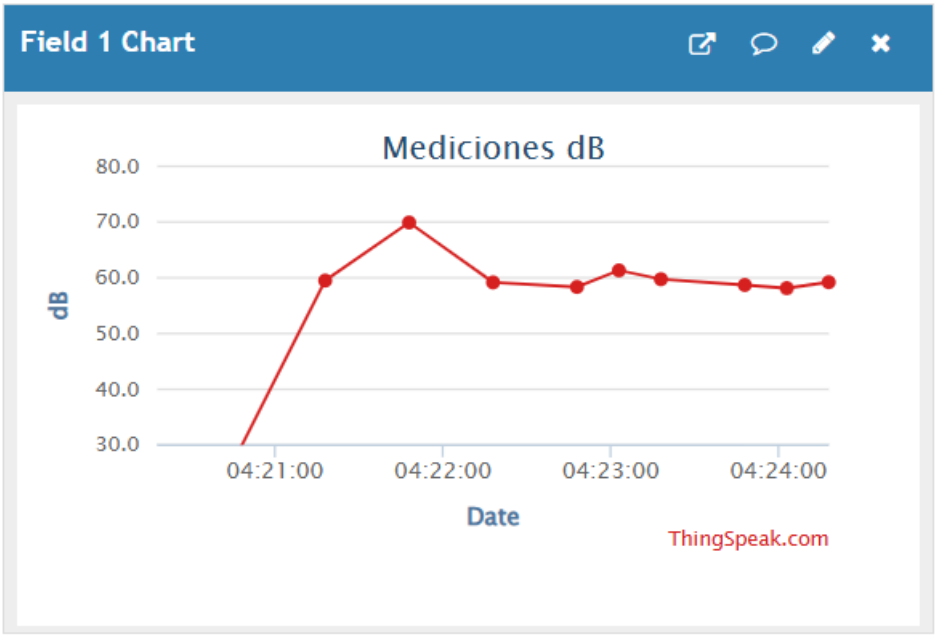


Figura 26: Gráfica de visualización de datos ThingSpeak, fuente: ThingSpeak.com

Se puede apreciar cómo se han ido mandando diferentes muestras durante unos minutos. Se hizo una pequeña prueba donde, al principio de la medición, se acercó el microcontrolador a un altavoz que emitía música, durante unos 10 segundos, y luego se alejó del altavoz, y se pudo apreciar cómo afectaba ese suceso en la gráfica. Cabe recalcar que, la inserción de datos dentro de ThingSpeak está limitada a una muestra cada 15 segundos, para evitar colapsos en su plataforma.

Un aspecto muy relevante de esta herramienta es que permite ver los datos en tiempo real, no es necesario refrescar la página en ningún momento. También, se puede ver cómo la gráfica va tomando las muestras y va configurando sus ejes de forma completamente automática.

6. Conclusiones y posibles aplicaciones

A lo largo del desarrollo del proyecto se han planteado y ejecutado tres diseños principales, cada uno con una finalidad específica relacionada con la medición, tratamiento y visualización de datos acústicos. Tras la implementación de dichos diseños, puede afirmarse que todos ellos han sido exitosos, cumpliendo adecuadamente con los requisitos funcionales definidos en su concepción. Se ha logrado integrar la captura de datos mediante sensores de sonido, su procesamiento local en el dispositivo M5Stack Core2, y su posterior envío a plataformas externas como ThingSpeak para su almacenamiento y análisis.

Una de las aplicaciones más prometedoras que se deriva de este proyecto es la posibilidad de implantar un sistema real de monitorización acústica en tiempo real. Este sistema podría estar compuesto por una red distribuida de micrófonos instalados en distintos puntos estratégicos, lo que permitiría una vigilancia continua del entorno sonoro. Los datos recogidos podrían visualizarse a través de diferentes esquemas, ya sea de forma local —mediante dispositivos físicos o interfaces gráficas integradas— o bien a través de servicios web accesibles desde cualquier dispositivo con conexión a Internet.

Como propuesta de mejora, se podría contemplar la implementación de sistemas más avanzados de análisis de datos, incluyendo algoritmos de clasificación de ruido, detección de anomalías acústicas, o integración con sistemas de alerta en tiempo real. Además, podría explorarse el uso de otras plataformas IoT que ofrezcan mayores capacidades de visualización, almacenamiento o escalabilidad. También sería recomendable reforzar la seguridad en las comunicaciones y en el acceso a los datos, especialmente en un escenario con múltiples nodos conectados a la red.

En conjunto, el trabajo realizado sienta una base sólida sobre la cual desarrollar sistemas inteligentes de monitorización sonora con aplicaciones reales en ámbitos como el control medioambiental, la salud pública o la gestión urbana.

7. Bibliografía

[1] Libelium. (2024). Waspmote Plug & Sense! Smart Environment
https://www.libelium.com Fecha de consulta: 15-05-2025

[2] NTi Audio. (2024). NoiseScout - Remote Noise Monitoring
https://www.nti-audio.com Fecha de consulta: 15-05-2025

[3] Ajuntament de Barcelona. (2023). Mapa del soroll i xarxa de sensors
https://ajuntament.barcelona.cat Fecha de consulta: 15-05-2025

[4] Documentación oficial de la gama de productos M5Stack,
https://docs.m5stack.com/en/core/core2 Fecha de consulta: 7-02-2025

[5] Página oficial de Arduino
https://www.arduino.cc Fecha de consulta 7-02-2025

[6] Documentación sobre XAMPP en YouTube
https://www.youtube.com/watch?v=IQ22Nme9t0M&t=11s
https://www.youtube.com/watch?v=tHoBYIBx2zs Fecha de consulta 12-04-2025

[7] Página oficial de ThingSpeak, herramienta de MathWorks
https://thingspeak.mathworks.com Fecha de consulta 7-06-2025

8. Glosario de siglas

SPL - Sound Pressure Level

MEMS - Micro-Electro-Mechanical System

HTML – HyperText Markup Language

PHP – Hypertext preprocessor

API – Application Programming Interface

IoT – Internet of Things

RMS – Root Mean Square

DC - Direct Current

IDE – Integrated Development Enviorment

PCM – Pulse Code Modulation

PDM – Pulse-Density Modulation

9. Acceso al código completo

Se ha depositado todo el código implementado en una carpeta compartida en Google Drive, para cualquier persona que quiera consultarlo.

El enlace para dicha carpeta se encuentra en el siguiente enlace:

https://drive.google.com/drive/folders/1U7YoBeupbyvRI4Wx51rp8BjLqXBcqXPk?usp=sharing

10. Anexos

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster

Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar	X			
ODS 4. Educación de calidad			X	
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante			X	
ODS 8. Trabajo decente y crecimiento económico			X	
ODS 9. Industria, innovación e infraestructuras	X			
ODS 10. Reducción de las desigualdades				X
ODS 11. Ciudades y comunidades sostenibles	X			
ODS 12. Producción y consumo responsables	X			
ODS 13. Acción por el clima				X
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas			X	
ODS 17. Alianzas para lograr objetivos			X	

Descripción de la alineación del TFG/TFM con los ODS con un grado de relación más alto

Objetivo 3: Salud y bienestar. La contaminación acústica es un elemento que perjudica el bienestar físico, mental y emocional de las personas. El proyecto trata de monitorizar el nivel de ruido con el objetivo de controlar que no sobrepase un nivel que llegue a resultar dañino para las personas.

Objetivo 9: Industria, innovación e infraestructuras. Al diseñar un nuevo medidor acústico con su sistema de visualización de datos, se contribuye a la innovación tecnológica de la industria de monitorización de ruido

Objetivo 11: Ciudades y comunidades sostenibles. El proyecto se plantea como propuesta de mejora para los medidores acústicos instalados en ciudades.

Objetivo 12: Producción y consumo responsables. El diseño del proyecto promueve el consumo responsable al optimizar el uso de recursos electrónicos y fomentar la reutilización de hardware accesible. Además, contribuye a una producción más sostenible al reducir la necesidad de dispositivos costosos o especializados para tareas de medición.