



Este documento expone el trabajo realizado para la implementación de un sistema software capaz de cumplir las normativas acústicas exigidas a todos los locales de ocio que reproduzcan música en el establecimiento, de forma que se garantice el bienestar tanto de los clientes y trabajadores del local como de los residentes cercanos al mismo.

A lo largo de este documento se presentan las distintas fases de desarrollo de software: análisis de requisitos, planificación, diseño, implementación, prueba y evaluación del software, y por último, despliegue.

A estas fases hay que añadir una etapa preliminar de Ingeniería Inversa, ya que se dispone de un limitador de sonido funcional, el cual se estudiará a fondo y supondrá el punto de partida del proyecto y una guía a seguir en un campo completamente desconocido.



Alejandro Ruiz Becerra es el estudiante a cargo del diseño e implementación del proyecto, y con este trabajo finaliza el Grado en Ingeniería Informática con mención en Ingeniería del Software.



Andrés María Roldán Aranda es el profesor ingeniero a cargo del presente proyecto, así como el tutor del alumno. Actualmente es profesor del departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

PROYECTO
FIN DE GRADO

Limitador de sonido para locales de música

Alejandro Ruiz Becerra

INGENIERÍA
INFORMÁTICA

2020 - 2021



UNIVERSIDAD DE GRANADA

Grado en
Ingeniería Informática



Limitador de sonido para locales de música

Alejandro Ruiz Becerra
2020-2021

Tutor: Andrés María Roldán Aranda



**GRADO EN
INGENIERÍA INFORMÁTICA
TRABAJO DE FIN DE GRADO**

**Limitador de sonido
para locales de música**

CURSO ACADÉMICO: 2020 - 2021

Alejandro Ruiz Becerra



GRADO EN INGENIERÍA INFORMÁTICA

Limitador de sonido para locales de música

REALIZADO POR:

Alejandro Ruiz Becerra

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de Computadores

D. Andrés María Roldán Aranda, Profesor del Departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de D. Alejandro Ruiz Becerra,

Informa:

Que el presente trabajo, titulado:

Limitador de sonido para locales de música

ha sido realizado y redactado por el mencionado alumno bajo mi dirección, y con esta fecha autorizo a su presentación.

Granada, a 2 de septiembre de 2021

A handwritten signature in black ink, appearing to read "Andrés María Roldán Aranda". The signature is fluid and cursive, with a long, sweeping line extending from the left.

Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Grado se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 2 de septiembre de 2021



Fdo. Alejandro Ruiz Becerra



Fdo. Andrés María Roldán Aranda

Limitador de sonido para locales de música

Alejandro Ruiz Becerra

PALABRAS CLAVE:

GranaSAT, Acústica y audio, Ingeniería Acústica, Ingeniería Inversa, Control de ruidos, Ecualización, Electrónica.

RESUMEN:

El objetivo del presente proyecto es diseñar e implementar el software necesario para la construcción de un limitador de sonido para locales de ocio, de forma que se cumplan las especificaciones legales y ordenanzas exigidas por las instituciones en éste ámbito, siendo beneficiaria del presente trabajo la empresa **Heimdal Sound Control**.

El proyecto puede dividirse en tres grandes bloques: ingeniería inversa, diseño e implementación. Durante la primera fase se estudian y analizan limitadores de sonido de la competencia, ya presentes en el mercado; para luego diseñar el sistema en base a los requisitos extraídos del proceso de ingeniería inversa, y finalmente desarrollar y probar el software del producto.

Este Trabajo de Fin de Grado se sitúa en el ámbito de un proyecto mayor, ambicioso y de largo recorrido, y se apoya en el trabajo realizado por otros alumnos pertenecientes a diversas competencias. Por tanto, el presente trabajo no debe verse como un todo, sino como un gran engranaje dentro de una máquina mayor, el cuál permite que el conjunto de componentes interaccionen entre ellos.

La complejidad y el ámbito multidisciplinar de este Trabajo de Fin de Grado permite cubrir, no sólo algunas de las diferentes especialidades del Grado en Ingeniería Informática, sino también adquirir conocimientos y habilidades transversales o específicos de otros campos de la Ingeniería, como la **Electrónica** y la **Acústica**.

El resultado de todo lo expuesto culmina con un equipo real de limitación de sonido completo y funcional, que cumple con los requisitos definidos en las etapas iniciales del proyecto, y con el cual se cierra la etapa universitaria de Grado.

Sound limiter for music venues

Alejandro Ruiz Becerra

KEYWORDS:

GranaSAT, Acoustics and audio, Acoustical Engineering, Reverse Engineering, Noise control, Equalization, Electronics.

ABSTRACT:

The main objective of this project is to design and implement the required software to build a sound limiter for music venues, so that the legal specifications and ordinances demanded by the institutions are enforced, being benefited of the current work the **Heimdal Sound Control** company.

The project can be divided in three blocks: reverse engineering, design and implementation. During the first phase, a functional sound limiter out of the market will be analyzed and investigated. As a result, our system's design will be based on the functional and non-functional requirements out of the reverse engineering process. Finally, the system's software will be implemented and tested accordingly.

This Bachelor's Thesis is under the scope of a bigger, ambitious and long-run project; and is based on the work of other students from different competences. Therefore, the current work should not be seen as a whole, but as a big gear inside a bigger machine, which allows the set of components interact.

The complexity and multi-disciplinary scope of this Bachelor's Thesis does not only allow to cover some of the different competences from the Degree in Computer Engineering, but also allows to acquire knowledge and skills from other Engineering areas, as **Electronics and Acoustics**.

The defined engineering process produces a complete and functional real-time sound limiter, which complies with the system's requirements defined on the initial steps of the project, and with which the University stage of Bachelor's Degree concludes.

“First do it, then do it right, then do it better. ”

Addy Osmani - Google Developer

Acknowledgments:

This work is dedicated to all those people: family, friends, colleagues, who have supported me throughout my life, and especially in recent years; as well as to all those professors and teachers, who through the great passion and dedication that they pour into their work have contributed to the fact that today I am writing these lines

Agradecimientos:

Este trabajo va dedicado a todas aquellas personas: familiares, amistades, compañeros y compañeras, que me han apoyado a lo largo de mi vida, y especialmente durante estos últimos años; así como a todos aquellos profesores y profesoras, universitarios y no universitarios, que mediante la gran pasión y dedicación que vuelcan en su trabajo han contribuido a que hoy me encuentre redactando estas líneas.

Tabla de contenidos

Autorización Lectura	v
Autorización Depósito Biblioteca	vi
Resumen	vii
Dedicatoria	x
Agradecimientos	xi
Tabla de contenidos	xiii
Lista de figuras	xxi
Lista de tablas	xxv
Acrónimos	xxvii
Glosario	xxxi
1 Introducción	1
1.1 Motivación	2
1.2 Objetivos del proyecto	3
1.3 Contexto	4
1.3.1 Personas involucradas en el proyecto	5
Limitador de sonido para locales de música	xiii

1.4	Estructura del proyecto	5
2	Especificación de requisitos	9
2.1	Requisitos funcionales	9
2.2	Requisitos no funcionales	10
3	Ingeniería Inversa: introducción	11
3.1	Instalación del limitador	12
3.1.1	Resumen	16
3.2	Extracción de credenciales	16
3.2.1	Credenciales del limitador	17
3.2.2	Credenciales del sistema operativo	19
3.3	Especificaciones del sistema	21
3.4	Análisis del rendimiento	22
3.5	Interfaz web	25
3.5.1	Imágenes	25
3.5.2	Análisis técnico	31
3.5.3	Comunicación IU-Limitador mediante ficheros	35
3.5.4	Comunicación IU-Limitador mediante procesos	37
3.6	Estructura del proyecto	37
3.7	Procesos del limitador LM7	41
4	Ingeniería Inversa: la versión LM7	45
4.1	Arranque	48
4.2	Detección de micrófono	50
4.3	Procesamiento de audio	51
4.3.1	Conección a tarjetas de sonido	51
4.3.2	Interpretación de datos	52
4.3.3	Almacenamiento de audio	53

4.4	Gestión de calibración	54
4.4.1	Proceso de calibración	54
4.4.2	Ecualizaciones	55
4.4.3	Emisión de ruido rosa	56
4.4.4	Verificación: sesiones	58
4.4.5	Almacenamiento de calibraciones	59
4.5	Gestión de configuración	59
4.5.1	Proceso de configuración	59
4.5.2	Almacenamiento de la configuración	60
4.6	Gestión de registros	60
4.6.1	Proceso de registro	60
4.6.2	Almacenamiento de registros	60
4.6.3	Lectura de registros	62
4.7	Proceso de atenuación	62
4.7.1	Comunicación con el PGA	63
5	Ingeniería Inversa: la versión LM9	65
5.1	Comunicación entre Raspberry y Arduino	66
5.2	Arranque	67
5.3	Detección de micrófono	70
5.4	Procesamiento de audio	71
5.4.1	Conexión a tarjetas de sonido	72
5.4.2	Interpretación de datos	73
5.4.3	Almacenamiento de audio	74
5.5	Gestión de calibración	74
5.5.1	Proceso de calibración	74
5.5.2	Ecualizaciones	75
5.5.3	Emisión de ruido rosa	75

5.5.4	Verificación: sesiones	76
5.5.5	Almacenamiento de calibraciones	76
5.6	Gestión de la configuración	76
5.6.1	Proceso de configuración	77
5.6.2	Almacenamiento de la configuración	77
5.7	Gestión de registros	77
5.7.1	Proceso de registro	77
5.7.2	Almacenamiento de registros	78
5.7.3	Lectura de registros	78
5.8	Proceso de atenuación	78
5.8.1	Comunicación con el PGA	78
6	Análisis del sistema	81
6.1	Análisis hardware del prototipo	82
6.1.1	Restricciones técnicas impuestas por el hardware	83
6.2	Evaluación de satisfacción requisitos	84
7	Diseño del sistema	89
7.1	Arranque	91
7.2	Instalación	93
7.3	Detección de micrófono	94
7.4	Procesamiento de audio	94
7.4.1	Conexión a tarjetas de sonido	94
7.4.2	Interpretación de datos	97
7.4.3	Almacenamiento de audio	97
7.5	Gestión de calibración	97
7.5.1	Proceso de calibración	97
7.5.2	Ecualizaciones	97
7.5.3	Emisión de ruido rosa	98

7.5.4	Verificación: sesiones	98
7.5.5	Almacenamiento de calibraciones	98
7.6	Gestión de la configuración	98
7.6.1	Proceso de configuración	98
7.6.2	Almacenamiento de la configuración	99
7.6.3	Consulta de la configuración	100
7.7	Gestión de registros	100
7.7.1	Proceso de registro	100
7.7.2	Almacenamiento de registros	100
7.7.3	Consulta de registros	100
7.8	Proceso de atenuación	101
7.8.1	Comunicación con el PGA	101
8	Implementación del sistema	103
8.1	Metodología de trabajo	103
8.2	Herramientas utilizadas	104
8.3	Proceso	104
8.3.1	Creando el entorno	104
8.3.1.1	Nueva estructura	105
8.3.1.2	Nuevo Makefile	106
8.3.2	Re-utilización de código	108
8.3.3	Novedades	108
8.3.3.1	La API-REST	109
8.4	Resultados	110
9	Validación y test	111
10	Conclusiones y trabajo futuro	113
10.1	Trabajo futuro	113

10.2 Conclusiones	114
Bibliografía	117
Bibliografía	117
A Presupuesto	119
A.1 Recursos físicos	119
A.2 Recursos humanos	120
A.3 Software	121
A.4 Presupuesto final	122
B Terminología	123
C Módulos del limitador	125
C.1 Módulos re-utilizados	125
C.1.1 getConfig	126
C.1.2 getStatus	133
C.1.3 getData	137
C.2 Módulos legacy	140
C.2.1 limInfo	140
C.2.2 localService	143
C.2.3 utilslr	146
D Ficheros del limitador	149
D.1 El registro	149
D.2 Ficheros de configuración	151
D.2.1 El fichero /tmp/conf.tmp	151
E Plantilla para la definición de un servicio en Linux	155

F LM7: documentación del código fuente 157

G LM7: constantes y macros 181

Listado de figuras

1.1	Logotipo de GranaSAT	1
1.2	Planificación del proyecto. Diagrama de Gantt	8
3.1	Limitador LM7.	11
3.2	Esquema conceptual del montaje del LM7	12
3.3	Parte trasera. Conexiones del LM7	13
3.4	Configuración IP requerida en el PC auxiliar.	15
3.5	Interfaz web del LM7.	15
3.6	Estructura de directorios de un sistema GNU/Linux	17
3.7	Estructura de las claves en el fichero <i>/etc/shadow</i>	19
3.8	Contenido del fichero <i>/etc/shadow</i> del LM7	20
3.9	Imagen de una placa base ALIX-1B [2]	23
3.10	Limitador LM7 y sus componentes montado en una placa base ALIX-1B. .	24
3.11	Procesos del LM7 [1/2]	24
3.12	Procesos del LM7 [2/2]	24
3.13	Vista principal. Estado del limitador.	26
3.14	Panel de control antes de identificación.	26
3.15	Panel de control después de identificación	27
3.16	Aislamiento.	27
3.17	Normativa [1/2].	28
3.18	Normativa [2/2].	28

3.19	Datos del local	29
3.20	Gestión de usuarios	29
3.21	Calibración [1/2].	30
3.22	Calibración [2/2].	30
4.1	Diagrama de contexto.	45
4.2	Diagrama de contexto extendido.	46
4.3	Diagrama de clases del LM7.	47
4.4	Diagrama de secuencia: detección de micrófono en LM7	50
4.5	Diagrama de clases simplificado del analizador de audio del LM7.	51
4.6	Diagrama de secuencia: calibración de micrófono.	54
4.7	Diagrama de secuencia: calibración de líneas.	55
4.8	Diagrama de secuencia: emisión de ruido rosa en LM7	57
4.9	Relé y otros componentes del LM7.	57
4.10	Detalles de los ficheros registro.slr y /dev/sda4.	62
5.1	Estructura de ficheros y directorios principal del LM9.	66
5.2	Diagrama de dependencias del módulo Analyzer en el LM9.	72
5.3	Formato del flujo de datos de acrshortALSA en modo intercalado.	73
5.4	Ficheros de audio del LM9.	74
6.1	Uno de los modelos del prototipo de limitador disponible.	83
7.1	Entradas balanceadas para canales del audio en el LM11.	89
7.2	Esquema hardware del limitador LM11.	90
7.3	Esquema software del limitador LM11.	91
7.4	Esquema parcial del sistema propuesto.	92
7.5	Bandas de frecuencias de los limitadores.	95
7.6	Recarga de ficheros de configuración y calibración.	99

8.1	Estructura de directorios del proyecto	106
8.2	Compilación con Make	106
10.1	Diagrama conceptual de un Despachador de Eventos.	115
C.1	Diagrama de clases del programa getConfig	126
C.2	Diagrama de clases del programa getStatus	133
C.3	Diagrama de clases del programa getData.	137
C.4	Diagrama de dependencias del programa limInfo.	140
C.5	Diagrama de dependencias del programa localservice.	143
C.6	Diagrama de dependencias del programa utilslr.	146
D.1	Atributos de las clases que componen el registro.	150
D.2	Diagrama de secuencia para actualizar la configuración en el LM11.	151

Lista de tablas

3.1	Especificaciones hardware del LIMITER MODULE 7 (LM7)	22
3.2	Clientes SSH utilizados en el proyecto.	22
3.3	Especificaciones del servidor web del LM7	25
4.1	Especificaciones del PGA2310UA.	63
6.1	Evaluación de satisfacción del RF-1	84
6.2	Evaluación de satisfacción del RF-2	84
6.3	Evaluación de satisfacción del RF-3	84
6.4	Evaluación de satisfacción del RF-4	85
6.5	Evaluación de satisfacción del RF-5	85
6.6	Evaluación de satisfacción del RF-6	86
6.7	Evaluación de satisfacción del RF-7	86
6.8	Evaluación de satisfacción del RF-8	87
7.1	Canales del limitador de GranaSAT.	95
A.1	Costes hardware.	119
A.2	Costes humanos	120
A.3	Costes software	121

Acrónimos

ALSA Advanced Linux Sound Architecture

ARM Advanced RISC Machine

COM COMmunication port

CPU Central Processing Unit

CSS Cascade Style Sheet

DNI Documento Nacional de Identidad

FFTW Fastest Fourier Transform in the West

GIO Gnome Input/Output

HTML HyperText Markup Language

IP Internet Protocol

ISO International Organization for Standardization

JSON JavaScript Object Notation

LCD Liquid Crystal Display

LED Light-Emitting Diode

LM₁₁ LIMITER MODULE 11

LM₇ LIMITER MODULE 7

LM₉ LIMITER MODULE 9

LPT Line Print Terminal

OSS Open Sound System

PC Personal Computer

PCB Printed Circuit Board

PCM Pulse Code Modulation

PGA Programmable Gain Amplifier

PID Process ID

PPID Parent Process ID

RISC Reduced Instruction Set Computer

SCSI Small Computer System Interface

SPI Serial Peripheral Interface

SQL Structured Query Language

SSH Secure SHell

TTY TeleTYpe

UART Universal Asynchronous Receiver-Transmitter

USB Universal Serial Bus

VGA Video Graphics Array

XLR eXternal Line Return

XML eXtensible Markup Language

Glosario

Apache El servidor HTTP Apache, es un servidor web de código abierto y multiplataforma. Es el servidor web más utilizado de la historia, seguido de cerca por Nginx..

API Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente *librerías*)..

API HTTP REST Siglas de Application Programming Interface, HiperText Tranfer Protocol y REpresentational State Transfer. Este concepto hace referencia a un tipo de interfaz web hacia el lado del cliente basada en [HTTP](#) que utiliza los verbos comunes del protocolo HTTP (GET, POST, DELETE...) para acceder y gestionar datos e información existente en el lado del servidor.

CompactFlash Las CompactFlash son dispositivos de almacenamiento de datos basado en [memoria flash](#) usado principalmente en dispositivos electrónicos portátiles, y fue especificado y producido por primera vez por SanDisk Corporation en 1994..

Daemon Se conoce como servicio o proceso *daemon* a aquellos procesos que se ejecutan en segundo plano y no tienen interacción con el usuario dentro del conjunto de procesos de un sistema operativo. En programación multihebrada en Python, se conoce como hilo *daemon* a aquellos hilos secundarios que se ejecutan en segundo plano y que terminan junto con el hilo de ejecución principal, sin necesidad de tener que gestionar su estado desde el mismo.

dBA El decibelio A o decibelio ponderado es una unidad de nivel sonoro medido con un filtro previo para quitar parte de las frecuencias bajas y las muy altas, con el objetivo de conservar solo las frecuencias más dañinas para el oído..

Debian Sistema operativo libre basado en [GNU/Linux](#) ®.

DietPi DietPi es una imagen del sistema operativo Debian altamente optimizada para tener un uso mínimo de [CPU](#) y RAM.

GNU/Linux GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o [kernel](#) libre similar a Unix denominado [Linux](#), que es usado con herramientas de sistema [GNU](#). Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU, en inglés: General Public License) y otra serie de licencias libres..

GranaSAT GranaSAT is an academic project from the University of Granada. Coordinated by the Professor Andrés María Roldán Aranda, GranaSAT is a multidisciplinary project with students from different degrees, where they can acquire and enlarge the knowledge necessary to face an actual aerospace project.

HW Conjunto formado por todas los componentes físicos de un producto electrónico.

Makefile Un Makefile es un fichero de texto en el que se definen reglas y directrices para la compilación automática de programas.

memoria flash La memoria flash permite la lectura y escritura de múltiples posiciones de memoria en la misma operación, por lo que permite velocidades de funcionamiento mayores..

OpenSSL .

RCA Los conectores RCA son un tipo de conector eléctrico para señales analógicas, y debe su nombre a *Radio Corporation of America*. La señal de los RCA no es balanceada, por lo que no suelen usarse profesionalmente..

RJ-45 Interfaz física comúnmente utilizada para conectar redes de computadoras, en especial la red Ethernet.

sesión En el ámbito del presente proyecto, una sesión corresponde al intervalo de tiempo durante el cual los valores de recepción medidos por el limitador se encuentran por encima de un determinado umbral. Durante este intervalo de tiempo, el limitador funciona de forma activa y aplica una atenuación de sonido cada vez que sea necesario. Cuando los valores de recepción caen por debajo de otro umbral, se asume que la sesión ha finalizado. Los umbrales de comienzo y fin de sesión se miden en decibelios y son configurables. Durante la sesión, se asume que se está emitiendo música en el local..

systemd systemd es un administrador de sistemas y servicios para sistemas Linux que corre con PID 1 e inicia el resto del sistema..

Windows Sistema operativo realizado por Microsoft™.

WiringPi WiringPi es una librería de acceso escrita en C basada en pines de GPIO, utilizada por todas versiones de RaspberryPi.

XLR El XLR es un tipo de cable de audio ampliamente en el campo del audio profesional, formado por 3 pines o clavijas, se usa para señales del audio balanceadas. En España se le conoce con el apodo ‘Cannon’, debido a que los primeros que se usaron en este país fueron fabricados por la marca Cannon, y llevaban “Cannon” grabado en el chasis..

Capítulo 1

Introducción

El presente proyecto se desarrolla en el marco del Trabajo de Fin de Grado con el cual se finaliza el Grado Universitario en Ingeniería Informática, con especialidad en Ingeniería del Software, cursado por el alumno Alejandro Ruiz Becerra en la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada. El objetivo principal del presente trabajo es demostrar los conocimientos y habilidades adquiridos por el alumno durante la realización de dicho Grado. Para ello, se ha realizado el presente proyecto, el cual propone el diseño e implementación de un software que permita controlar los niveles acústicos en locales con equipos de música.

Este Trabajo de Fin de Grado se realiza en colaboración con el proyecto académico [GranaSAT](#), un proyecto multidisciplinario, dirigido por el Profesor Andrés María Roldán Aranda, que reúne a personas de diferentes campos que quieren adquirir conocimientos relativos a la Ingeniería Aeroespacial y a la Ingeniería Electrónica.



Figura 1.1 – Logotipo de [GranaSAT](#)

El laboratorio de [GranaSAT](#) así como el equipamiento y materiales necesarios para la

realización de este proyecto se encuentran en la Avenida de Madrid, frente al Escuela Internacional de Posgrado de la Universidad de Granada y el Antiguo Hospital Clínico de Granada (España). Este proyecto ha sido desarrollado tanto presencialmente en el laboratorio como de forma remota, debido a la situación actual de pandemia causada por el virus COVID-19.

1.1 Motivación

Todos los establecimientos comerciales que dispongan de equipos de reproducción musical o audiovisual pueden, especialmente de noche, afectar no solo al descanso de los vecinos, sino también a la salud auditiva de los clientes y a los trabajadores del mismo. Es por ello que surge la necesidad de controlar la emisión de ruidos desde estos establecimientos tanto hacia a las viviendas o oficinas adyacentes, como a la calle, así como mantener unos niveles acústicos adecuados en el interior del local. Esta necesidad se hace requisito para este tipo de establecimientos debido a la Ley del Ruido del año 2003.

La Ley del Ruido (Ley 37/2003) del 17 de noviembre de 2003 dio lugar al comienzo de la lucha legislada contra la contaminación acústica, mediante la cual se dota de un esquema básico a nivel estatal para que, en los niveles autonómico y local puedan elaborarse promulgaciones de esta ley. Por tanto, queda en manos de las comunidades autónomas, y en última instancia, de los ayuntamientos, el definir las medidas, normas e infracciones a aplicar en esta materia.

El Ayuntamiento de Granada aplica su propia normativa en este ámbito mediante la *Ordenanza Municipal de Protección del Medio Ambiente Acústico en Granada* del año 2007. Es de especial interés, dado el proyecto que nos ocupa, los artículos 36 y 37.

1.1.1 Artículo 36. Condiciones acústicas particulares en actividades y edificaciones donde se generan niveles elevados de ruido

En este artículo se definen una serie de condiciones para los establecimientos de espectáculos públicos, actividades recreativas y comerciales que generen elevados niveles de ruido.

A modo de resumen, este artículo:

- Clasifica las edificaciones donde se generan un nivel de ruido superior a 70 dB(A) en 3 tipos.
- Exige la aplicación de un aislamiento acústico al local.
- Prohíbe niveles de presión sonora superiores a 90 dB(A) en zonas destinadas al

público, salvo que se dé advierta correctamente en los accesos a dichas zonas.

Si bien, estos tipos de establecimientos tienen la obligación de aislar acústicamente el local mediante la aplicación de diferentes soluciones arquitectónicas, esto no es suficiente para garantizar que no se sobrepasan los límites de emisión e inmisión de ruido definidos en la normativa, la cual se puede consultar en el Anexo B. Esto mismo queda reflejado en el artículo 37 de la normativa que puede verse a continuación.

1.1.2 Artículo 37. Instalación de equipos limitadores controladores acústicos

En aquellos locales descritos en el artículo 36 de la presente Ordenanza, donde se disponga de equipo de reproducción musical o audiovisual en los que los niveles de emisión sonora pudieran de alguna forma ser manipulados directa o indirectamente, se instalará un equipo limitador-controlador que permita asegurar, de forma permanente, que bajo ninguna circunstancia las emisiones del equipo musical superen los límites admisibles de nivel sonoro tanto en el interior del propio local (artículo 20 bis) como en las edificaciones adyacentes, así como que cumplen los niveles de emisión al exterior exigidos en esta Ordenanza.

Por tanto, resulta evidente observar que existe una necesidad por parte de los establecimientos con equipos de reproducción de música de controlar sus niveles de contaminación acústica de forma que se cumpla la normativa, para evitar molestias a vecinos, y denuncias y sanciones a los propietarios del local.

1.2 Objetivos del proyecto

El objetivo primario de este proyecto es el siguiente:

1. Presentar un software capaz de registrar los niveles acústicos de emisión y recepción, y que actúe sobre dichos niveles en base a la normativa aplicada.

Este objetivo general define de forma concreta qué es lo que se pretende hacer en el proyecto. Asimismo, existen otros objetivos más específicos:

1. Conocer y analizar las necesidades reales del sistema de monitorización de ruido ambiental requerido.
2. Extraer los requisitos funcionales y no funcionales del sistema, a partir de conversaciones con el cliente y del proceso de ingeniería inversa.

3. Realizar una análisis de las tecnologías utilizadas en el software presente en el limitador del laboratorio, y estudiar si pueden solucionar las necesidades de cada uno de los requisitos.
4. Diseñar e implementar un prototipo del producto software requerido.
5. Realizar los tests de validación y pruebas necesarios para verificar el correcto funcionamiento del prototipo.
6. Acercar al alumno a un entorno real de Ingeniería.
7. Poner de manifiesto los conocimientos y habilidades adquiridas en el Grado de Ingeniería Informática.
8. Superar la signatura de Fin de Grado, y por ende, el Grado en Ingeniería Informática.

1.3 Contexto

En el laboratorio de [GranaSAT](#) se dispone de un limitador de sonido operativo, pero bastante antiguo, el cual es necesario estudiar para poder entender su funcionamiento y, por ende, conocer cuales son las tareas concretas que debe realizar un limitador de sonido para efectuar sus funciones. Este limitador será objeto que un minucioso proceso de ingeniería inversa, del cual extraeremos una gran cantidad de conocimiento, en forma de requisitos, diagramas y estrategias de acción para resolver las problemáticas a las que nos enfrentamos en el proyecto. Se dispone también de todo el código fuente de este limitador, lo cual facilitará bastante nuestro trabajo.

Se dedicará un capítulo completo a este limitador, dónde se detallará el proceso de ingeniería inversa al que se le ha sometido, se explicará cómo funciona este limitador y se expondrán todos los conocimientos que se han podido extraer del él. Este limitador, por tanto, supondrá el punto de partida y la principal guía del presente proyecto, siendo así su piedra angular, ya que se volverá él constantemente para comparar y validar el trabajo realizado.

También en laboratorio de [GranaSAT](#) se dispone de un limitador en fase de prototipo, sobre el cual correrá el sistema de limitación y control de sonido que se va a desarrollar. Esta arquitectura supone la inclusión de ciertas restricciones y consideraciones a la hora de diseñar e implementar el sistema, ya que se dispone de un hardware especializado y cuyo centro de cómputo es una Raspberry Pi.

Por otro lado, es necesario poder configurar el sistema cuando se realice la instalación del mismo en el establecimiento. Los datos configurables en el equipo supondrán unos requisitos de datos y especificarán en el capítulo 2, pero a modo de resumen, es necesario

configurar una serie de variables que modificarán el funcionamiento del sistema, como el aislamiento aplicado al local y la normativa del ayuntamiento o comunidad autónoma en el que se encuentra el local.

Nuestro sistema, por tanto, deberá comunicarse tanto con el hardware especializado como con la aplicación de configuración. Para ello deberán identificarse y definirse unas interfaces de comunicación para que los sistemas cooperen, lo cual se traduce en nuevo requisitos en nuestro proyecto. Se profundizará en estas interfaces en el capítulo de diseño, [7](#).

1.3.1 Personas involucradas en el proyecto

- Luis Peinado Córdoba, es un estudiante del Máster en Ingeniería de Telecomunicaciones y el responsable del nuevo prototipo hardware. El trabajo realizado en este prototipo supone su Trabajo de Fin de Máster.
- Dani Ruiz Medina, estudiante del Grado en Ingeniería Informática, como proyecto para su Trabajo de Fin de Grado va a diseñar e implementar una aplicación para configurar nuestro sistema una vez esté implementado y ejecutándose en el prototipo.
- Alejandro Ruiz Becerra, estudiante del Grado en Ingeniería Informática y autor del presente documento, estudiará el limitador existente para desarrollar un software similar capaz de ejecutarse en el nuevo prototipo.
- Andrés María Roldán Aranda, además de tutor del Trabajo Fin de Grado y Máster de los tres alumnos mencionados, será el supervisor del proyecto.

1.4 Estructura del proyecto

El proyecto se divide en 10 capítulos y varios anexos que describen cada una de las partes del proceso de desarrollo del producto propuesto.

Los capítulos que componen el presente documento son:

- El presente capítulo, numerado como [1](#), pretende ser una introducción al proyecto, tanto en su faceta de Trabajo de Fin de Grado como en su faceta de colaboración con GranaSAT. En este capítulo también puede encontrarse la planificación temporal del proyecto en forma de diagrama de Gantt.
- El capítulo [2](#) es un breve resumen de la idea global del producto, extraída de conversaciones con el cliente. A partir de esta información se generan una

definición de necesidades principales y secundarias para el producto, en forma de requisitos del cliente.

- El capítulo 3 se procede a documentar el proceso de ingeniería inversa al que se ha sometido a dos versiones de un mismo limitador disponible en el laboratorio de GranaSAT. La idea principal de este proceso de ingeniería inversa es realizar un reconocimiento de las tareas técnica necesarias para completar los objetivos del proyecto. El capítulo se compone de 3 partes.
 - [Ingeniería Inversa: introducción](#).
 - [Ingeniería Inversa: la versión LM7](#).
 - [Ingeniería Inversa: la versión LM9](#).
- A continuación, en el capítulo 6, se utilizará el conocimiento obtenido mediante proceso del ingeniería inversa realizado en el capítulo anterior para presentar un primer análisis en profundidad del trabajo necesario a realizar para cumplir los objetivos del proyecto, así como un desglose de dicho trabajo en forma de requisitos del sistema. En este capítulo también se detallarán los requerimientos técnicos del producto. Una vez concretados estos requerimientos se analizarán los pros y los contras de diferentes tecnologías o enfoques que solucionan la problemática de cada uno de los subsistemas que componen el proyecto. De este análisis en profundidad se extrae una estrategia de trabajo, compuesta por un conjunto de soluciones técnicas viables para el producto y una planificación temporal para el desarrollo del proyecto que se puede observar en el diagrama de la figura figura X.Y.
- Tras realizar el análisis y siguiendo la metodología propuesta en la figura X.Y, se presenta en el capítulo 7 el diseño del producto, es decir, qué se va a hacer exactamente y cómo. Para ello se han tenido en cuenta las cuestiones relativas al análisis del capítulo anterior y se han seleccionado las soluciones y tecnologías optimas para diseñar el sistema.
- En el capítulo 8 se presentan los detalles sobre la implementación del sistema y la metodología seguida durante este proceso. Los procesos de ingeniería inversa, análisis y diseño tendrán un gran valor en este punto, y serían cruciales durante el proceso de implementación.
- En el capítulo 9 se describen una serie de pruebas de validación y test de forma que se verifique que el producto generado a partir del diseño propuesto en el capítulo 7 cumple con las especificación de requisitos descrita en el 2, y que a demás, lo realiza de forma correcta.
- Por último, el capítulo 10 contiene un conjunto de conclusiones a modo de resumen, donde se repasan los objetivos del proyecto y cómo se han abordado

cada uno de ellos. También se incluye una lista de mejoras del sistema como trabajo futuro y continuación del proyecto.

- En el anexo A se detalla el presupuesto y los costes asociados a este proyecto.

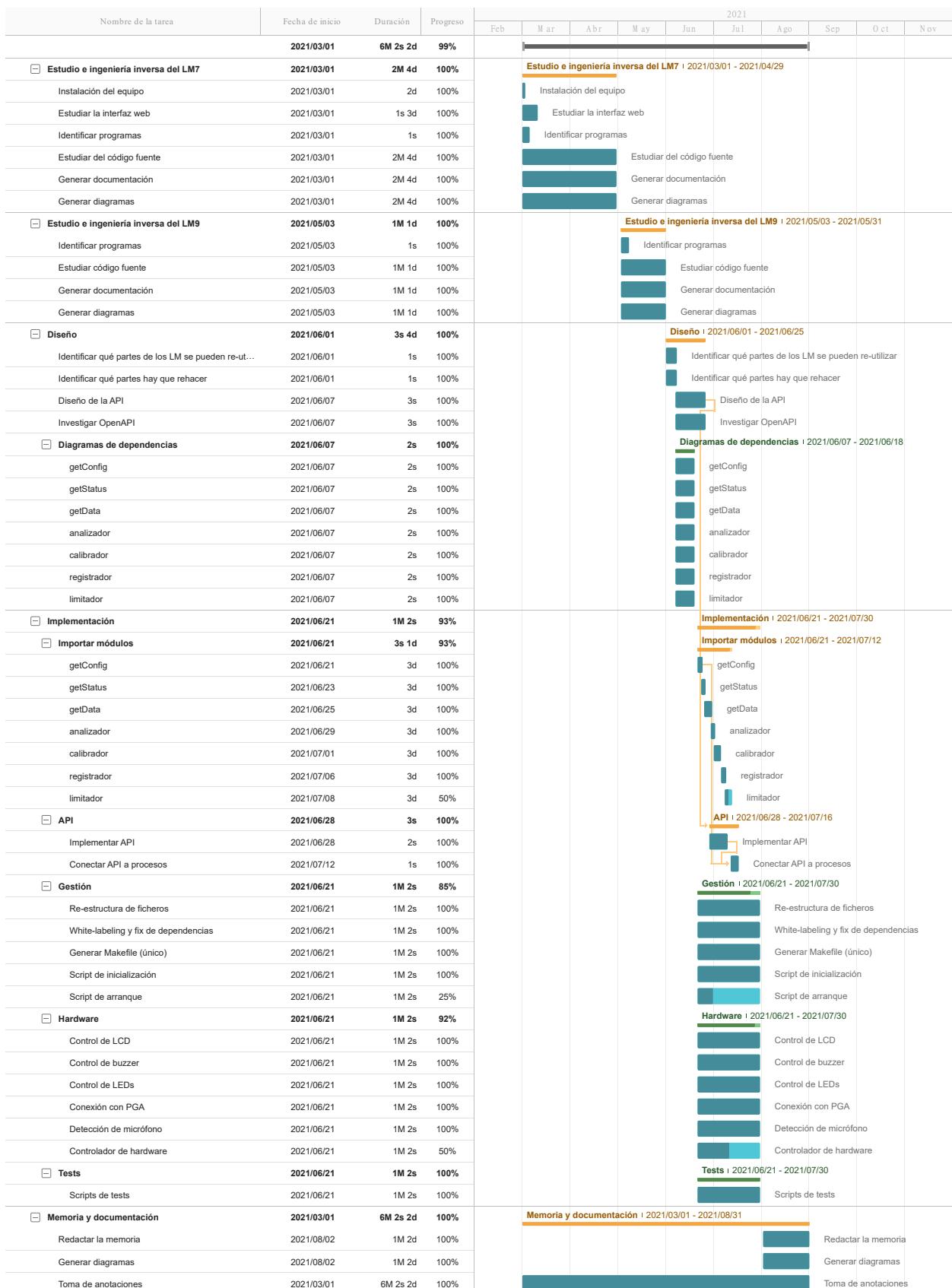


Figura 1.2 – Planificación del proyecto. Diagrama de Gantt.

Capítulo 2

Especificación de requisitos

Una vez introducido el tema de estudio y los objetivos del presente proyecto, se va a dar paso en este segundo capítulo a definir cuáles son los requisitos del sistema, clasificándolos en requisitos funcionales y no funcionales.

Estos requisitos tienen su origen en las conversaciones que se han establecido con el cliente en primera instancia y como resultado del conocimiento extraído del proceso de ingeniería inversa, por lo han ido variando en el tiempo.

2.1 Requisitos funcionales

1. El sistema debe contar con funcionalidad para calibrar sus sensores.
2. El software debe emitir ruido rosa., necesario para los procesos de calibrador y verificación.
3. El sistema debe comprobar que no ha sido manipulado. Para ello se verificará en el inicio y/o fin de cada sesión de ruido (música) que las calibraciones de sus sensores y señales de audio corresponden a los valores actuales de emisión y recepción.
4. El sistema debe poder conservar un registro de sus lecturas por un período al menos 2 meses, con una periodicidad de un minuto.
5. El sistema debe proporcionar un mecanismo de comunicación hacia el exterior, de forma que se pueda configurar y obtener métricas.
6. El sistema debe poder leer los valores de presión acústica en el local mediante el uso de un micrófono.

Es deseable que el software permita leer la presión acústica desde un segundo micrófono.

7. El sistema debe actuar en caso de que los valores de emisión en el local sobrepasen los valores límite definidos en la normativa vigente, limitando el nivel de presión acústica.
8. El software debe incluir un mecanismo capaz de permitir la detección del sistema a otros equipos en la misma red.

2

2.2 Requisitos no funcionales

Como requisitos no funcionales tenemos:

1. El sistema debe poder ejecutarse en el hardware provisto (módulo de computación Raspberry Pi con sistema operativo [DietPi](#) y arquitectura [ARM](#)).
2. Debe usarse el marco de trabajo de [ALSA](#) para la comunicación con la tarjeta de sonido instalada en el equipo.
3. El sistema debe ser robusto y tolerante a fallos.
4. El producto debe ser lo más ligero posible en términos de consumo de recursos de computación y espacio en disco, ya que estos recursos son especialmente limitados en la arquitectura objetivo.
5. El sistema debe usar [JSON](#) como formato estándar de intercambio de datos.
6. Los valores de presión acústica deben darse en decibelios ponderados ([dBA](#)).

Capítulo 3

Ingeniería Inversa: introducción

En este capítulo se procede a documentar el proceso de ingeniería inversa al que se ha sometido a limitador que puede verse en la imagen [3.1](#).

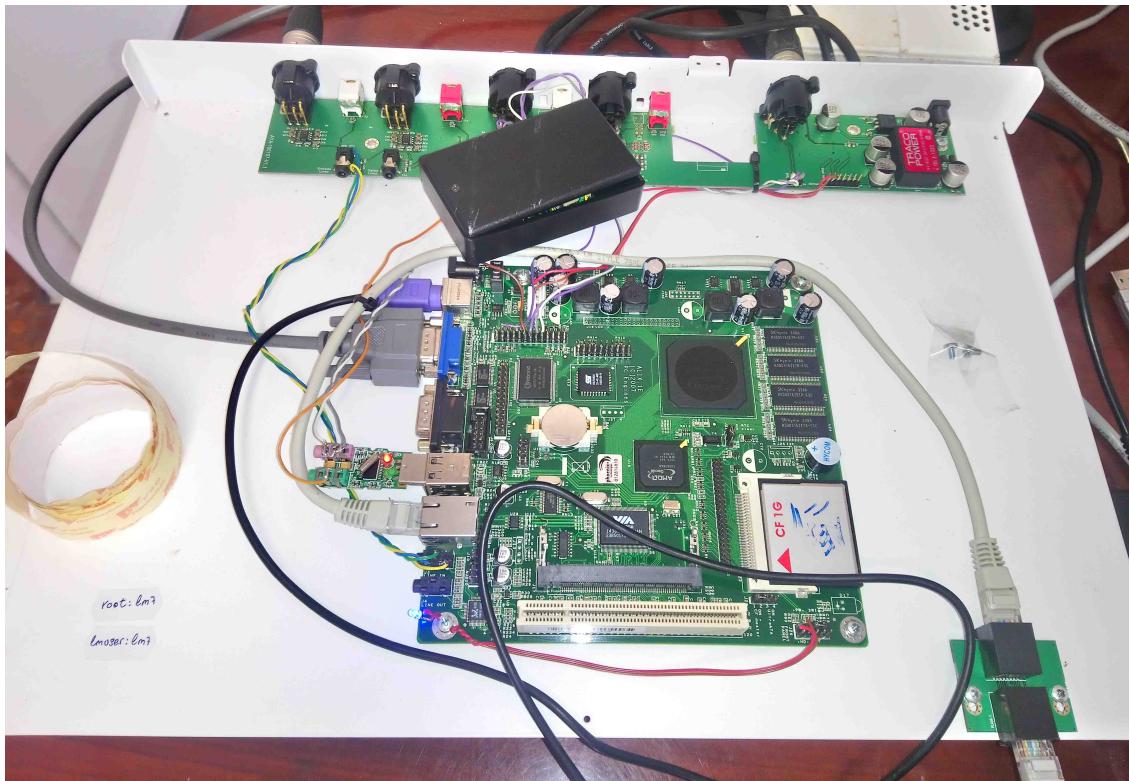


Figura 3.1 – Limitador LM7.

Tal y como se comentó en la sección [1.3](#), el punto de partida del proyecto es el estudio y análisis de un limitador funcional y operativo que se encuentra en el laboratorio de

GranaSAT. Es importante recordar que se dispone del código fuente de este limitador, así como su manual de usuario, el cual nos será de gran ayuda para poder instalarlo correctamente y conocer las capacidades y características generales del producto. De aquí en adelante, se hará referencia al este limitador como **LM7**, ya que ese es el nombre técnico del producto.

3

3.1 Instalación del limitador

El primer paso a realizar para comenzar el proceso de ingeniería inversa es instalar el equipo. Para ello, se recurre al manual de usuario del equipo, y se siguen las instrucciones de instalación. En la figura 3.2 se puede observar el esquema de montaje del limitador en un entorno objetivo, es decir, en un establecimiento. En nuestro caso, nuestra entrada no se corresponde a una mesa de mezclas, sino a un ordenador, mediante el cual podremos enviar audio al limitador y comprobar su respuesta.

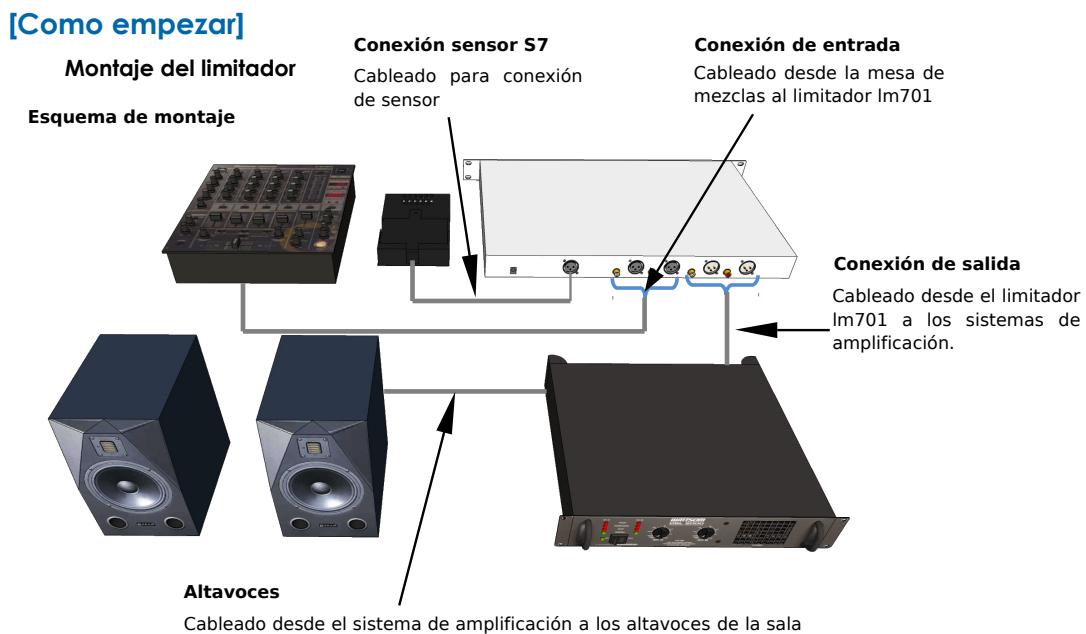


Figura 3.2 – Esquema conceptual del montaje del LM7

En la figura superior, podemos observar de izquierda a derecha y de arriba a abajo los siguientes elementos: mesa de mezclas (entrada), micrófono del limitador, limitador de sonido **LM7**, altavoces, amplificador (salida).

[Descripción e imágenes]

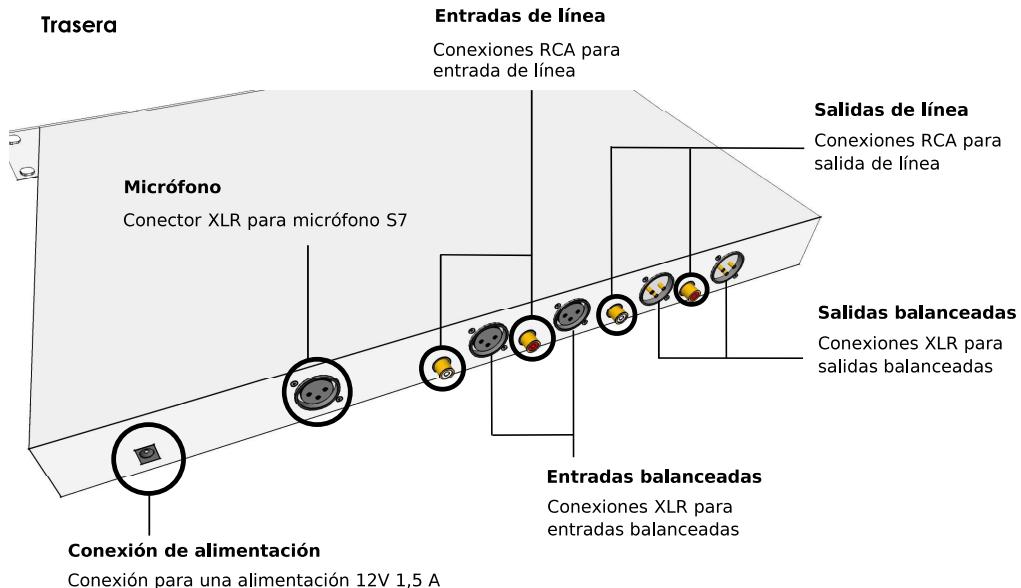


Figura 3.3 – Parte trasera. Conexiones del LM7

El sensor S7 (micrófono) es un componente esencial del limitador y forma parte del mismo. Gracias a él, el limitador puede medir la presión acústica en cada momento y actuar en consecuencia. Asimismo, y como se verá más adelante en este documento, será un elemento esencial en la calibración del limitador. El resto de los elementos que se muestran en la figura 3.2 son elementos externos al limitador, y no alteran el funcionamiento del sistema en ninguna forma.

En la figura 3.3 se pueden observar las principales conexiones del LM7, las cuales constan de:

- Toma de alimentación eléctrica.
- Sensor S7 (micrófono) con conector **XLR**.
- Entradas balanceadas para audio con conectores **XLR**.
- Salidas balanceadas para audio con conectores **XLR**.
- Entradas y salidas no balanceadas con conectores **RCA**.
- Conector **RJ-45** para conexión directa en área local (parte frontal del limitador, no visible en las figuras).

Conectamos el micrófono al limitador, así como la entrada y la salida de audio. La salida de audio se conecta a un amplificador de sonido disponible en el laboratorio,

y como salida a este sistema de amplificación se conecta un dodecaedro. De forma que podamos tener interacción con el equipo, abrimos el limitador retirando la carcasa metálica exterior, para así poder conectar un teclado y una pantalla a la placa base del equipo. De esta forma descubrimos por primera vez las entrañas del limitador, y nos encontramos con una tecnología bastante desfasada y un ensamblaje prácticamente casero. Tal y como podemos ver en la imagen 3.10, el **Hardware** del limitador se compone de una placa base de tipo industrial, cuyas características de detallan en la tabla 3.1, un circuito integrado para las entradas y salidas de audio del limitador vistos en la figura 3.3 y una pequeña caja negra, la cual contiene un relé, y cuya funcionalidad se verá más adelante. Además de esto podemos observar la existencia de una tarjeta de sonido **USB** conectada a un puerto **USB** de la placa base y cuyas salidas van hacia la caja negra que podemos ver en la imagen. Como dispositivo de almacenamiento interno tenemos un **CompactFlash**, con 1 GB de capacidad. En fases posteriores del actual proceso de ingeniería inversa extraeremos esta tarjeta de almacenamiento para acceder a sus archivos, ya que será necesario descubrir o modificar las credenciales de acceso tanto al equipo como al sistema de limitación, pues todas ellas son, por ahora, desconocidas.

Tras revisar todas las conexiones, conectamos el equipo a la corriente eléctrica, y podemos por primera vez ver el equipo en funcionamiento. Mientras arranca el sistema, vemos en la pantalla que el sistema operativo instalado es Debian, una distribución de **GNU/Linux**® que se caracteriza por ser minimal. Esta versión en concreto no contiene interfaz gráfica.

Una vez completado el arranque, la pantalla se llena de caracteres que desaparecen rápidamente dando lugar a otros nuevos. De entre lo que es posible leer, se deduce que estos caracteres son datos relativos al limitador (su estado en cada momento, conteniendo lecturas de micrófono y líneas, atenuación aplicada, etc) y que los procesos del limitador vuelcan sus salidas por pantalla a la salida estándar, saturando la terminal principal y dejándola inutilizable. Por tanto, se accede a otra terminal (**TTY**) mediante la cual podamos trabajar. Esta nueva terminal nos pide usuario y contraseña, las cuales desconocemos.

Continuando con el manual de usuario se descubre que el equipo tiene un servidor web en el cual hay desplegada una interfaz web del limitador, mediante la cual podemos ver su estado, modificar sus configuraciones y obtener informes. Por ello, el próximo y último paso para completar la instalación del limitador es conectarlo a una red interna vía Ethernet, así como conectar y configurar un ordenador adicional mediante el cual podamos acceder, no solo a dicha interfaz web, sino al equipo en sí mediante **SSH**.

Una vez aplicada la configuración **IP** que se muestra en la imagen 3.4 en el equipo auxiliar, probamos a acceder a la interfaz web de limitador que se encuentra desplegada en la dirección **IP** <http://192.168.1.223> y vemos por primera vez la aplicación web vista en la imagen 3.5. Al entrar en la interfaz web del limitador puede

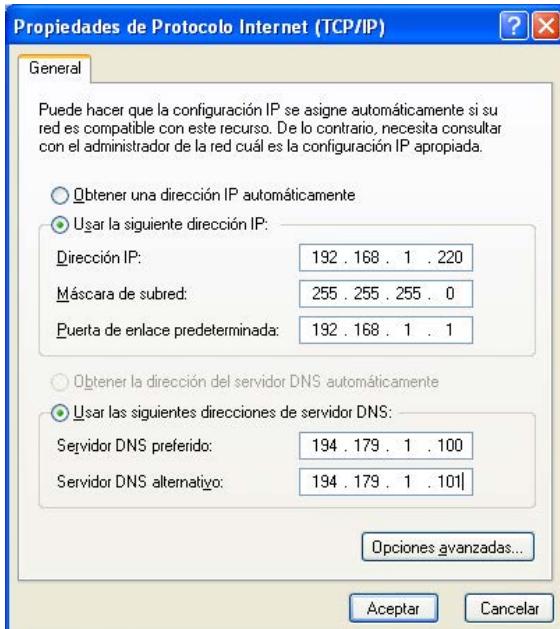


Figura 3.4 – Configuración IP requerida en el PC auxiliar.



Figura 3.5 – Interfaz web del LM7.

verse la ventana de estado, que informa sobre el estado actual del limitador. Esta interfaz arroja información básica, como:

- La presión actual en **dBA** tanto de las líneas como del sensor.
- Atenuación aplicada por el limitador en ese instante.
- El número de serie del limitador.
- El local de instalación
- Si el sensor (micrófono) se encuentra conectado o no.
- Un gráfico de los últimos 5 minutos de actuación del limitador.

En la zona superior derecha, se puede acceder al panel de control y a la sección de obtención de informes. Mediante este panel de control podremos también acceder y modificar la configuración del limitador, aunque para ellos será necesario proveer una clave de acceso válida. En el manual de usuario, se indica que existe un usuario *consultor*, cuya contraseña es a su vez *consultor*. Se trata de un usuario sin privilegios mediante el cual podremos consultar la configuración actual del limitador. Podemos a su vez cerrar sesión mediante el botón “Cerrar sesión”. Para poder acceder más allá necesitamos averiguar las claves de acceso al limitador.

Para acabar con el proceso de instalación, se comprueba si es posible conectarse mediante **SSH** al limitador. Tal y como se esperaba hay conectividad entre los equipos

pero necesitamos proporcionar un usuario y una contraseña para acceder al sistema operativo.

Llegados a este punto, el limitador se encuentra instalado y funcionando, pero nuestro control sobre el mismo es completamente nulo ya que no disponemos de las credenciales necesarias para acceder al sistema ni al limitador. Por tanto, procederemos a extraer su dispositivo de almacenamiento para investigar desde otro ordenador su contenido y poder encontrar dichas credenciales, dando lugar así al proceso que da nombre a este capítulo, comenzaremos con el proceso de Ingeniería Inversa.

3.1.1 Resumen

3

Como resumen a esta sección, se han realizado las siguientes acciones:

1. Se ha conectado el limitador LM7 a corriente y se han conectado a él un monitor (interfaz [VGA](#)) y un teclado (interfaz PS/2).
2. Se ha instalado un [PC](#) auxiliar con el sistema operativo Windows 10.
3. Se han conectado los dos equipos mediante Ethernet, usando un Switch de la marca OvisLink, con la configuración de red vista en la imagen [3.4](#).

Tras esto, ambos equipos quedan conectados en red local, por lo que se puede acceder al limitador desde el equipo auxiliar mediante [SSH](#), aunque desconoce el usuario y contraseña del sistema.

3.2 Extracción de credenciales

Para conseguir el acceso al sistema limitador, extraemos el dispositivo de almacenamiento y lo exploramos en otro ordenador mediante un adaptador. Explorando el sistema de archivos se descubre la existencia de un directorio peculiar dentro del directorio /var en el nivel principal de la estructura de directorios de Linux.

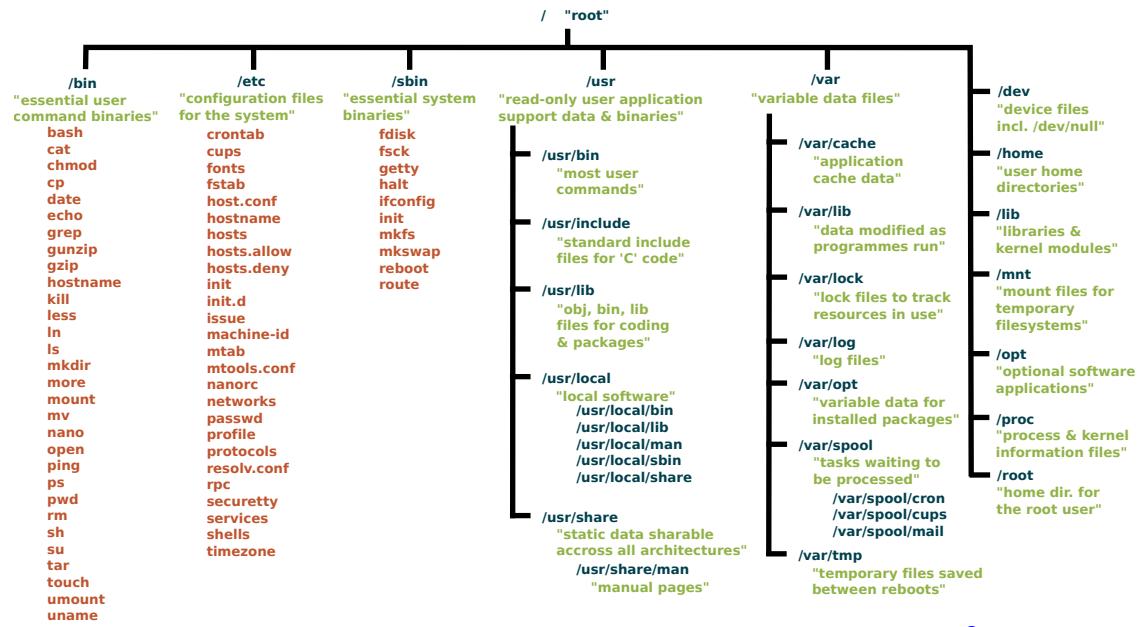


Figura 3.6 – Estructura de directorios de un sistema GNU/Linux ®
Fuente : [28]

3

3.2.1 Credenciales del limitador

Dentro de esta carpeta denominada `s1r/` encontramos rápidamente un fichero llamado `users.auth`, con usuarios, claves y permisos. Estas credenciales no son las del sistema operativo que corre sobre la máquina, sino las de los usuarios que tienen acceso a la configuración del limitador, es decir, son los usuarios y contraseñas necesarios para acceder a la aplicación web del limitador (Imagen 3.5), así como los permisos de estos usuarios sobre la configuración del limitador.

El directorio `s1r/` será de gran importancia en el ámbito del proyecto, ya que en este directorio se almacenarán la gran mayoría de ficheros relacionados con el limitador: datos de configuración, información de usuarios, ficheros de sonido, e incluso ficheros que funcionarán a modo de variables globales del limitador. Conforme se vaya avanzando en el documento se irá describiendo y explicando cada uno de estos ficheros.

El nombre del directorio se deduce que es un acrónimo de *SoundLimiterRecords*.

Para nuestra sorpresa, se descubre que los datos no sólo están almacenados en un fichero de texto plano, sino que tampoco se encuentran cifrados. Cada una de las líneas del fichero define un usuario con los siguientes campos:

- DNI.

- Nombre.
- Contraseña.
- Gestor de usuarios (si puede crear, modificar o eliminar usuarios).
- Gestor de configuración (si puede modificar la configuración del limitador).
- Fecha del alta del usuario en el sistema.

3

En el listado 3.1 pueden verse claros ejemplos del patrón definido justo sobre estas líneas. Adicionalmente, existen en el fichero otras líneas que no siguen este patrón y definen otros patrones nuevos. Ejemplos de estas líneas son las que encontramos en la línea 4 y 14 del listado 3.1. Estas líneas definen el modo de acceso al limitador y la eliminación de un usuario en el sistema, respectivamente, así como la marca de tiempo de la acción que dio origen a la inserción de esas líneas en el fichero.

Para confirmar el descubrimiento de las credenciales del limitador se comprueban algunos de los usuarios y claves encontradas en la interfaz web del limitador, con resultado satisfactorio. Se consigue por tanto el acceso a la configuración del limitador y tenemos ahora el control sobre él, aunque seguimos sin disponer de control sobre el sistema operativo sobre el que corre.

```

1  dni=*&lm7-&passwordUser&name=Password user&password=\t*****&userManager=1&
2   configManager=1&time=2012/05/11-11:48:36
3  dni=*&lm7-&remoteUser&name=Remote system user&password=\t*****&userManager=0&
4   configManager=0&time=2012/05/11-11:48:36
5  dni=consultor&name=Consultor&password=consultor&userManager=0&configManager
6   =0&time=2012/05/11-11:48:36
7  key=authMethod&value=onlyPassword&time=2012/08/30-05:35:16
8  dni=E19578186&name=NOISEOFF&password=COCHA012015&userManager=1&configManager
9   =1&time=2015/06/25-23:07:38
10 dni=E19578186&name=NOISEOFF&password=COCHA012015&userManager=1&configManager
11  =1&time=2015/06/25-23:08:13
12 dni=E19578186&name=NOISEOFF&password=01COCHA2015&userManager=1&configManager
13  =1&time=2015/06/26-14:55:51
14 key=authMethod&value=onlyPassword&time=2015/06/26-14:57:58
15 dni=E19578186&name=NOISEOFF&password=COCHA0130115&userManager=1&
16   configManager=1&time=2016/09/20-19:14:48
17 dni=E19578186&name=NOISEOFF&password=COCHA0130115&userManager=1&
18   configManager=1&time=2016/09/20-19:16:21
19 key=authMethod&value=onlyPassword&time=2016/09/20-19:16:50
20 dni=E19578186&name=NOISEOFF&password=cocha0130115&userManager=1&
21   configManager=1&time=2017/01/02-12:08:33
22 dni=44289989Q&name=NOISEOFF&password=cocha0130115&userManager=1&
23   configManager=1&time=2017/01/02-12:10:05
24 dni=E19578186&deleted=1
25 key=authMethod&value=onlyPassword&time=2017/01/02-12:12:01
26 dni=44289989Q&name=NOISEOFF&password=cocha0130115&userManager=1&
27   configManager=1&time=2017/01/05-09:50:40
28 key=authMethod&value=onlyPassword&time=2017/01/05-09:52:03

```

Listado 3.1 – Contenido del fichero users.auth

3.2.2 Credenciales del sistema operativo

Para obtener el control sobre el sistema operativo es necesario disponer de un usuario y contraseña válidos de forma que podamos acceder a él mientras el equipo está en funcionamiento.

Del sistema operativo sabemos que es [Debian](#), una distribución [GNU/Linux](#)®, por tanto, se investiga dónde almacena este sistema sus usuarios. Son datos estáticos así que deben encontrarse almacenados en alguna parte dentro del sistema de archivos contenidos en el disco, el cual, recordamos, tenemos conectado a otro ordenador mediante un adaptador [USB](#).

A través de la consulta de foros [16] y manuales [6] en línea, descubrimos que los ficheros que necesitamos son `/etc/passwd` y `/etc/shadow`. Aunque ambos contienen información crítica sobre los usuarios y sus permisos, existen pequeñas diferencias. En resumen, mientras que `/etc/passwd` almacena información mayormente relativa al usuario, `/etc/shadow` contiene las claves de usuario (encriptadas) e información relacionada a ellas, no al usuario.

Las contraseñas encriptadas y otra información relacionada, como la información de caducidad de la contraseña, se almacenan en el fichero `/etc/shadow`. Este fichero contiene una entrada por línea por cada uno de los usuarios listados en el fichero `/etc/passwd`. Cada una de estas entradas están compuestas por una serie de campos separados por dos puntos (:), y generalmente, sigue este patrón.

```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::
    ↓   ↓   ↓   ↓   ↓   ↓
  1     2     3     4     5     6
```

Figura 3.7 – Estructura de las claves en el fichero `/etc/shadow`

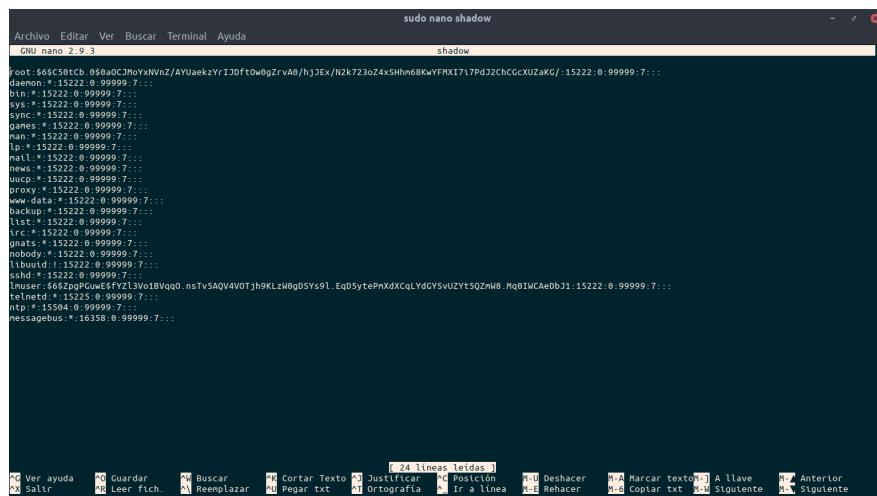
1. Nombre de usuario.
2. Contraseña encriptada (**hash**). Sigue el formato **\$id\$salt\$hash**. El campo **\$id** representa el algoritmo usado:
 - (a) **\$1\$** es MD5.
 - (b) **\$2a\$** es Blowfish.
 - (c) **\$2y\$** es Blowfish.
 - (d) **\$5\$** es SHA-256.
 - (e) **\$6\$** es SHA-512.
3. Días desde que se cambió la contraseña, contando desde el 1 de enero de 1970.

4. Días tras los cuales la contraseña debe ser cambiada.
5. Días de antelación a la expiración de la contraseña en los que se avisa al usuario.
6. Días tras los cuales se desactiva una cuenta cuya contraseña está caducada.

Una contraseña **hash** no es más que una cadena de caracteres única, la cual es resultado de la ejecución de un algoritmo de encriptación, dada otra cadena de caracteres como entrada. Esta contraseña **hash** es la que se almacena en el sistema, y no la contraseña original. Durante el proceso de inicio de sesión, se comprueba el **hash** de la contraseña insertada por el usuario y la contraseña **hash** almacenada, verificando así la integridad de la misma.

3

Abrimos el fichero **/etc/shadow** y encontramos la información que necesitamos. La imagen 3.8 muestra el contenido de este fichero.



```
shadow
root:$6$c58tCb$69a0DCMoYxNvNz/AYUaeKzYrIJDft0w0gZrvA0/hjEx/N2k723oZ4xSHhm68KwYFHKI7l7PdJ2chCgcXUzaKG/:15222:0:99999:7:::
daemon:*:15222:0:99999:7:::
bin:*:15222:0:99999:7:::
sys:*:15222:0:99999:7:::
sync:*:15222:0:99999:7:::
games:*:15222:0:99999:7:::
man:*:15222:0:99999:7:::
lp:*:15222:0:99999:7:::
mail:*:15222:0:99999:7:::
news:*:15222:0:99999:7:::
uucp:*:15222:0:99999:7:::
proxy:*:15222:0:99999:7:::
www-data:*:15222:0:99999:7:::
backup:*:15222:0:99999:7:::
list:*:15222:0:99999:7:::
irc:*:15222:0:99999:7:::
nobody:*:15222:0:99999:7:::
libuuid!:15222:0:99999:7:::
sshd!:15222:0:99999:7:::
sshd:15222:0:99999:7:::
telnetd!:15222:0:99999:7:::
http!:15804:0:99999:7:::
messagebus!:16358:0:99999:7:::
```

Figura 3.8 – Contenido del fichero */etc/shadow* del LM7

Los usuarios que nos interesan son **root** y **lmuser**. Ambas contraseñas se han cifrado usando el algoritmo SHA-512, ya que su primer campo es **\$6\$**. En un primer intento se modifica el fichero y se eliminan los campos que contienen la contraseña *Hash*, con la intención de forzar el inicio de sesión sin la necesidad de contraseña (contraseña vacía). Sin embargo, este enfoque no produce el resultado esperado y se busca otra solución. Como conocemos tanto el formato de las entradas de este fichero como el algoritmo de cifrado utilizado, el segundo enfoque será sustituir estas claves *Hash* con otras nuevas, generadas por nosotros. Generamos una nueva contraseña usando la utilidad **OpenSSL**.

```
1 | $~ openssl passwd -6 <cadena>
```

Listado 3.2 – Generación de Hash usando cifrado SHA-512

La ejecución del comando [3.2](#) nos genera una clave Hash única para la cadena que le proporcionemos como entrada. La opción -6 indica que debe usarse el algoritmo de cifrado SHA-512. Para más información sobre este comando y sus opciones puede consultarse la documentación oficial [\[20\]](#).

Una vez cambiadas las contraseñas por las que hemos generado, re-instalamos el [CompactFlash](#) en la placa base del limitador y se procede a encenderlo para comprobar que podemos acceder al sistema con las nuevas credenciales. Tras finalizar el arranque se verifica que el inicio de sesión con ambos usuarios es satisfactorio, y que por tanto, se tiene control sobre el sistema operativo. La conexión al equipo mediante [SSH](#) también funciona correctamente usando las nuevas contraseñas. Anteriormente, se ha comprobado que la configuración [SSH](#) del sistema permite la conexión mediante el usuario *root*. Para ello se comprueba que en el fichero de configuración en la ruta /etc/sshd_conf contiene la directiva *PermitRootLogin* y que su valor es YES. Tras comprobar el fichero, se verifica que la directiva existe y está configurada correctamente. Este no es el valor por defecto, por lo tanto ha tenido que ser activado con anterioridad. Además, se añade la siguiente directiva para garantizar el acceso a los usuarios *root* y *lmuser*:

```
1 | AllowUsers root lmuser
```

Listado 3.3 – Directiva del fichero

3

3.3 Especificaciones del sistema

El primer paso para comenzar a investigar el equipo es descubrir ante que tipo de hardware nos enfrentamos. En secciones anteriores (imagen [3.1](#)) se mostró el hardware del limitador bajo estudio, y se comentó brevemente sus características. En esta sección, se va a realizar un análisis más profundo sobre las capacidades y características hardware del sistema:

Nos encontramos ante un equipo en el que se ha montado una placa base de tipo industrial, en la que vienen integrados todos los recursos hardware necesarios para correr un sistema operativo. En la siguiente tabla pueden verse los componentes básicos con los que cuenta el sistema:

Como componentes que no forman parte de la placa base, tenemos una tarjeta de sonido [USB](#) conectada a uno de los puertos y un circuito integrado para las entradas y salidas de audio, conectado a la placa base mediante su puerto serie.

Placa base	ALIX-1E
Procesador	AMD Geode LX
Memoria	128/256MB SDRAM
Almacenamiento	CompactFlash 1GB
Interfaces	4xUSB, 1xVGA, 1xLPT, 2xCOM
Conectividad	Ethernet
Dimensiones	17x17cm

Tabla 3.1 – Especificaciones hardware del LM7

3

3.4 Análisis del rendimiento

Tal y como se comentó en la primera sección de este capítulo, el ordenador auxiliar que se ha instalado en el laboratorio, y en red como el LM7, tiene Windows® 10 como sistema operativo. Para facilitar el acceso remoto mediante SSH al equipo limitador, se hace uso de clientes SSH. En la tabla 3.2 se lista el cliente usado tanto en Windows® 10 como en GNU/Linux.

Sistema Operativo	Cliente SSH
Linux	Snowflake SSH
Windows	MobaXterm

Tabla 3.2 – Clientes SSH utilizados en el proyecto.

Estos clientes aportan ciertas ventajas, como por ejemplo:

- Permiten guardar los datos de acceso a equipos remotos. Una conexión SSH a otro equipo se resume en hacer click sobre un botón.
- Proporcionan una interfaz gráfica.
- Disponen de un navegador de archivos.
- Monitorizan el sistema remoto, lo que nos permite analizar el uso de recursos (disco, memoria, CPU y red) en tiempo real.

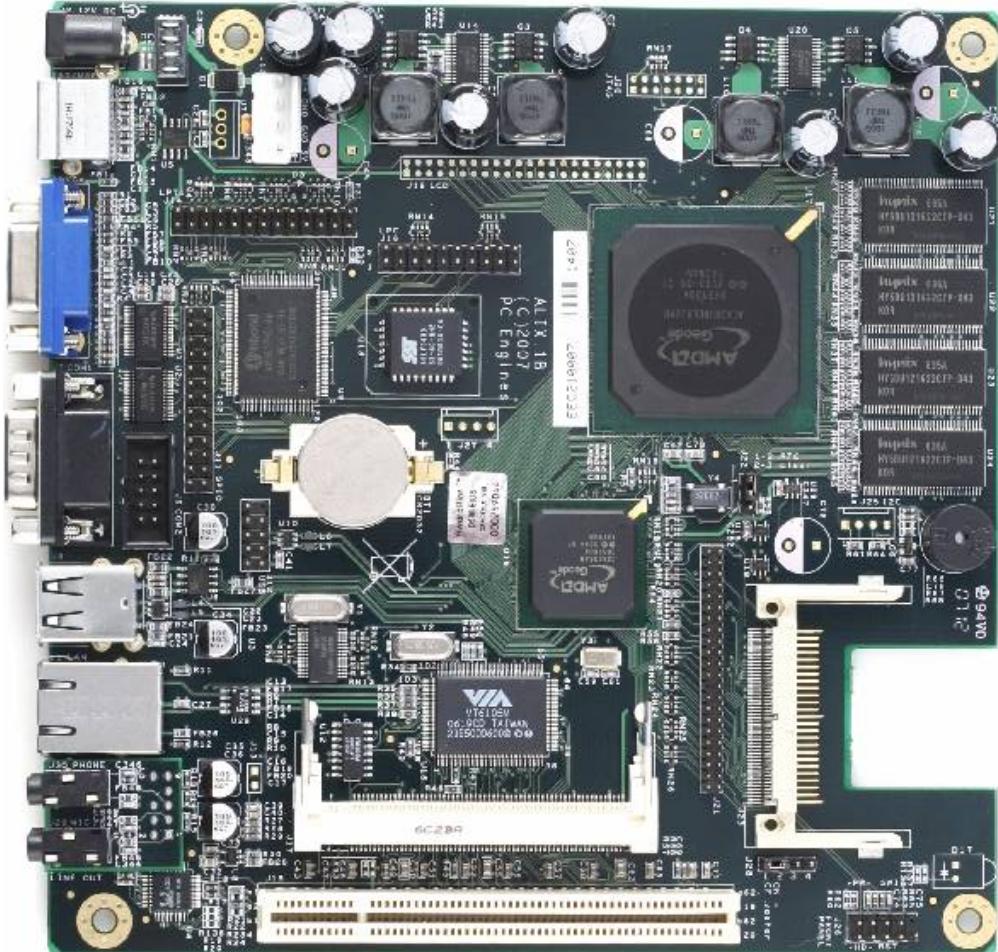


Figura 3.9 – Imagen de una placa base ALIX-1B [2]

Tras configurar el cliente [SSH](#), se lanza una conexión remota al [LM7](#). Los datos de monitoreo muestran una alta tasa de utilización del [CPU](#), en torno al 85-100% (ver imágenes [3.11](#) y [3.12](#)). Indudablemente, el hecho de que el [CPU](#) disponga de una único núcleo es significativo, sin embargo, se procede a identificar los procesos en funcionamiento que pertenecen al limitador. Para ello, se recurre al código fuente del limitador, en concreto al fichero [Makefile](#), el cual se encarga de compilar los distintos ejecutables (a los que llamaremos [módulos](#)) y de moverlos al directorio raíz del sistema: `/bin`. Al colocar los programas en esta carpeta, se pueden lanzar de forma global, es decir, sin especificar su ruta.

El fichero [Makefile](#) nos proporciona una lista de nombres de procesos a buscar. Listamos los procesos activos con la orden `ps`. El comando completo así como el

3

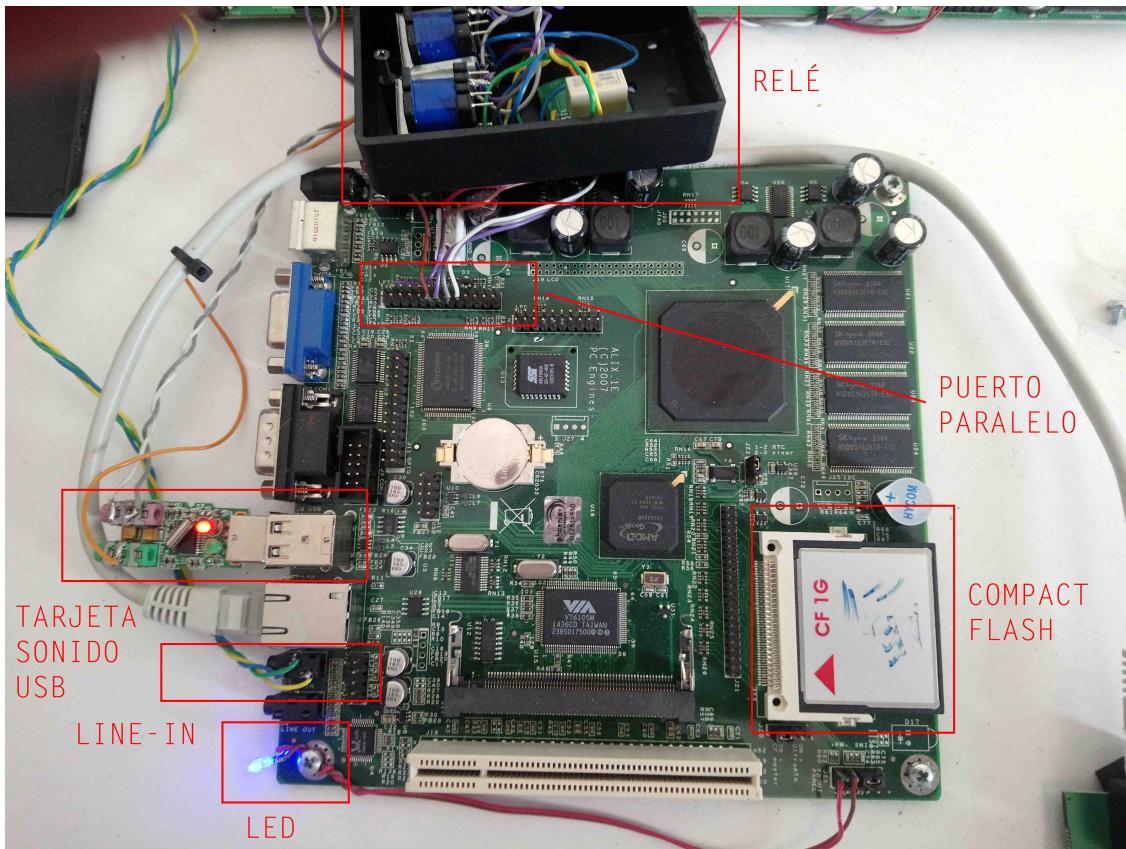


Figura 3.10 – Limitador LM7 y sus componentes montado en una placa base ALIX-1B.

```
lent@192.168.1.223:root
```

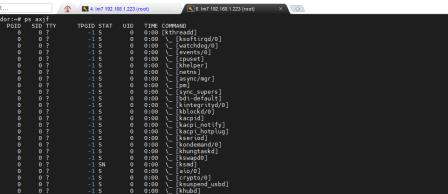


Figura 3.11 – Procesos del LM7 [1/2]

Figura 3.12 – Procesos del LM7 [2/2]

resultado devuelto pueden observarse en las captura de pantalla [3.11](#) y [3.12](#). Las opciones pasadas al invocar al comando `ps` permite mostrar no solo los procesos en activo, sino también las relaciones jerárquicas que existen entre ellos. La mayoría de los procesos del limitador son fácilmente detectables incluso en el supuesto de que se conocieran sus nombres con anterioridad. Los procesos relativos al limitador se encuentran remarcados en rojo, y se puede apreciar el detalle de que todos ellos tiene

como proceso padre a `init`, con [PID 1](#). Podemos ver esta información en la primera columna de la tabla, [PPID](#). Esto significa que los procesos del limitador son lanzados automáticamente al arrancar el sistema. Se profundizará sobre esto en la sección [4.1](#), donde se explicará proceso de arranque del [LM7](#).

Los procesos que componen en limitador se verán con profundidad en la sección [3.7](#), pero antes, se presentará y estudiará la interfaz web del limitador en su totalidad (ya se presentó la sección principal con la imagen [3.5](#)), mediante la que podemos configurarlo y visualizar las lecturas de los sensores y la atenuación aplicada en cada momento.

3.5 Interfaz web

La interfaz web del limitador se encuentra desplegada en un servidor [Apache](#), y está programada usando PHP, [CSS](#), [HTML](#) y Javascript. En la tabla [3.3](#) puede consultarse un análisis más detallado de las tecnologías usadas en la interfaz web.

Tecnología	Versión
Sistema Operativo	Debian 4.3.5
Servidor Web	Apache 2.2.16
PHP	5.5.3

Tabla 3.3 – Especificaciones del servidor web del LM7

El estudio de la interfaz representa un aspecto clave en el proceso de ingeniería inversa en el que nos encontramos. La interfaz web arroja una gran cantidad de información mediante la cual somos capaces no sólo de comprender qué hace el sistema, qué datos almacena y como los representa y exporta, sino que también nos permite generar un conjunto de requisitos funcionales, no funcionales y de datos que nuestro proyecto deberá satisfacer, y como mínimo, debe hacerlo tan bien (o mal) como el sistema estudiado (requisitos de calidad). Por otra parte, será una guía y un recurso al cual recurriremos una y otra vez a lo largo de este proyecto. En definitiva, la interfaz web nos permite, de un vistazo, comprender el sistema y sus capacidades.

3.5.1 Imágenes

En esta subsección se va a presentar las interfaz web en imágenes de forma que el lector pueda ver la interfaz web por sí mismo, aunque de forma estática. Todas estas

imágenes has sido extraídas del manual de usuario del limitador LM7.

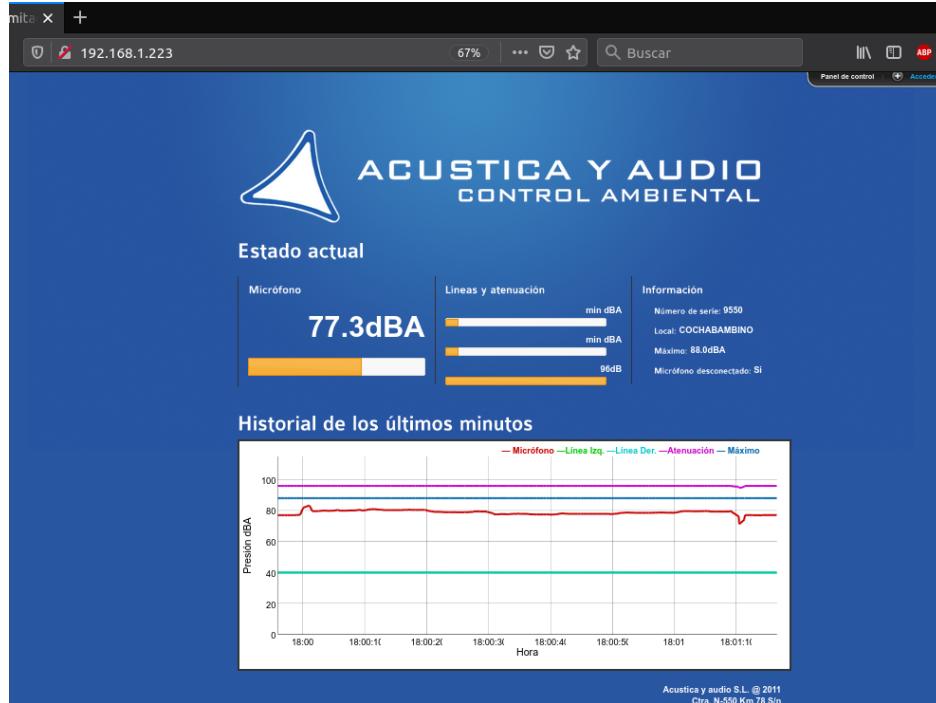


Figura 3.13 – Vista principal. Estado del limitador.



Figura 3.14 – Panel de control antes de identificación.

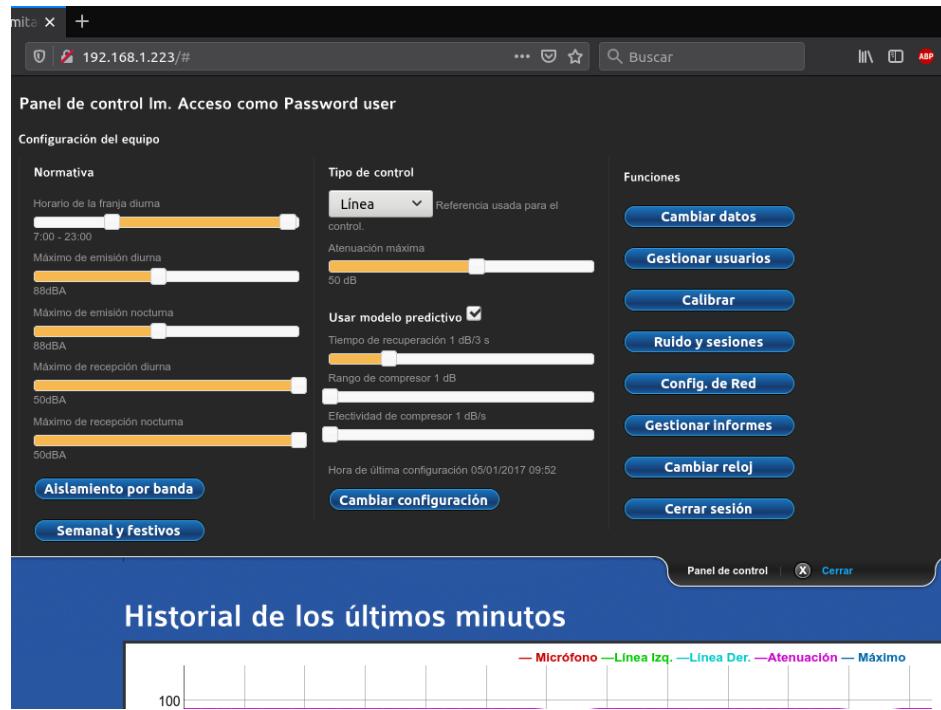


Figura 3.15 – Panel de control después de identificación

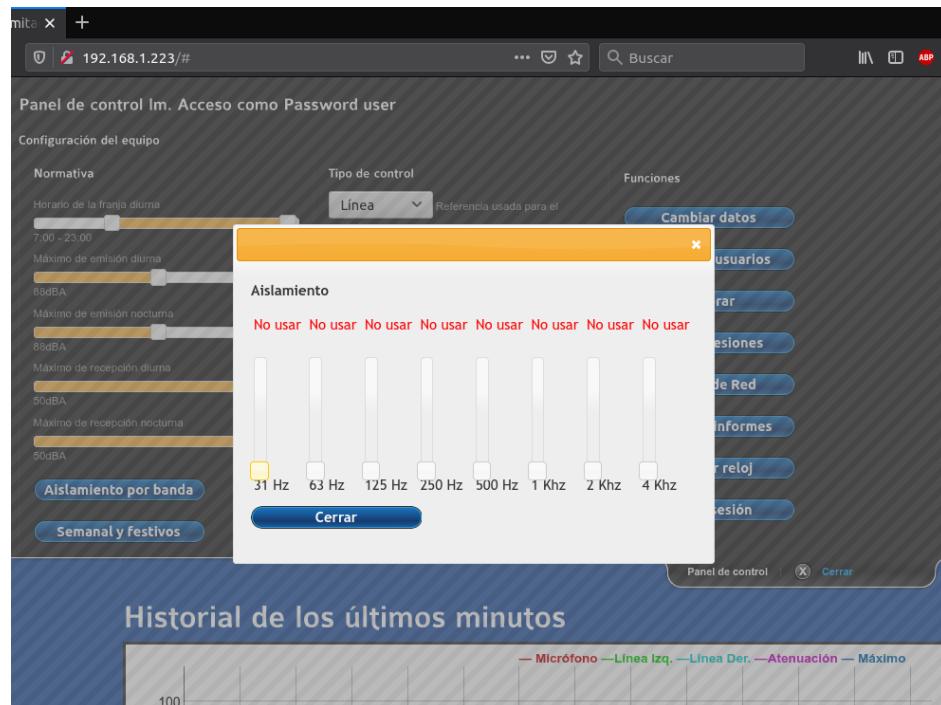


Figura 3.16 – Aislamiento.

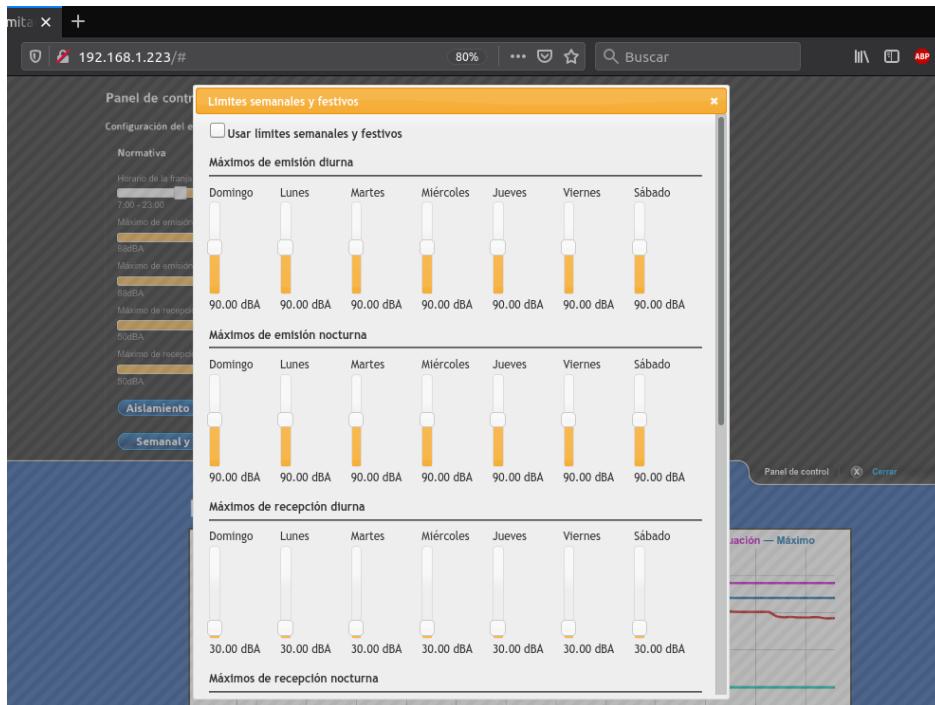


Figura 3.17 – Normativa [1/2].

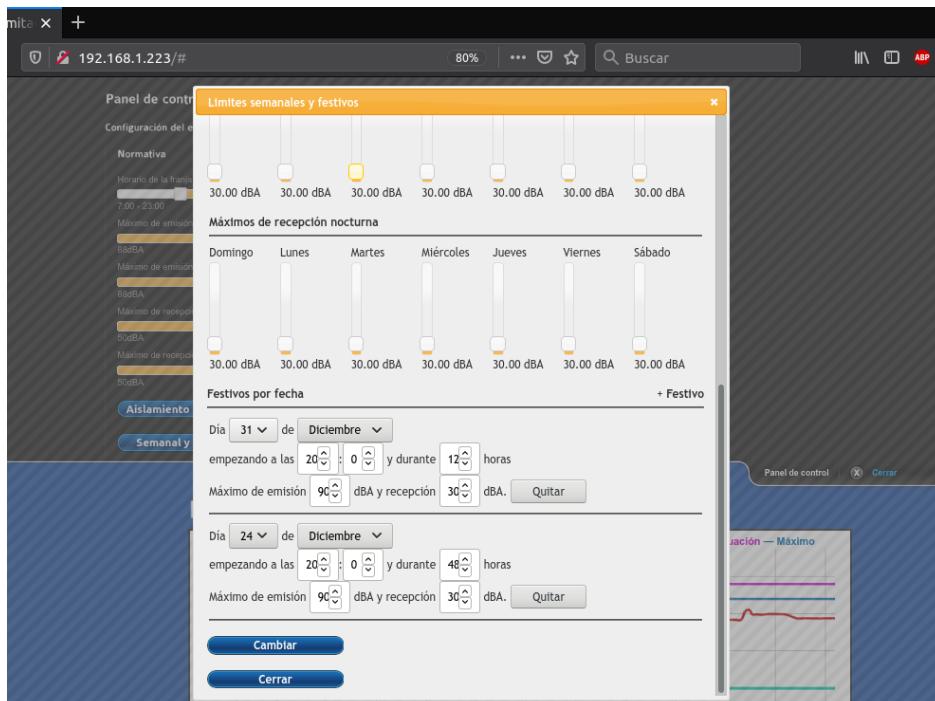
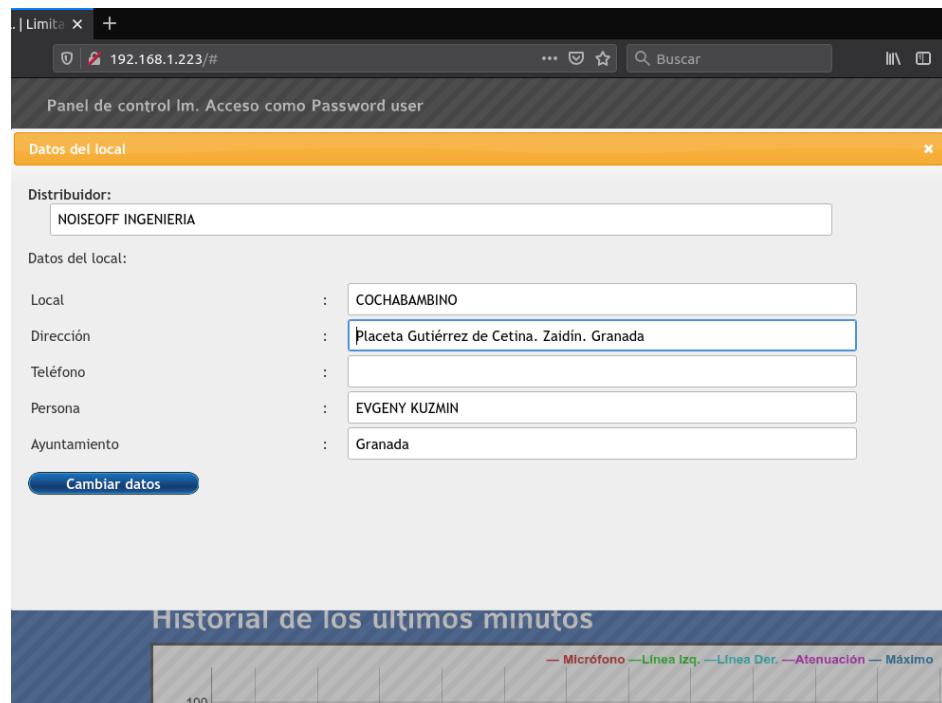


Figura 3.18 – Normativa [2/2].



3

Figura 3.19 – Datos del local.



Figura 3.20 – Gestión de usuarios.

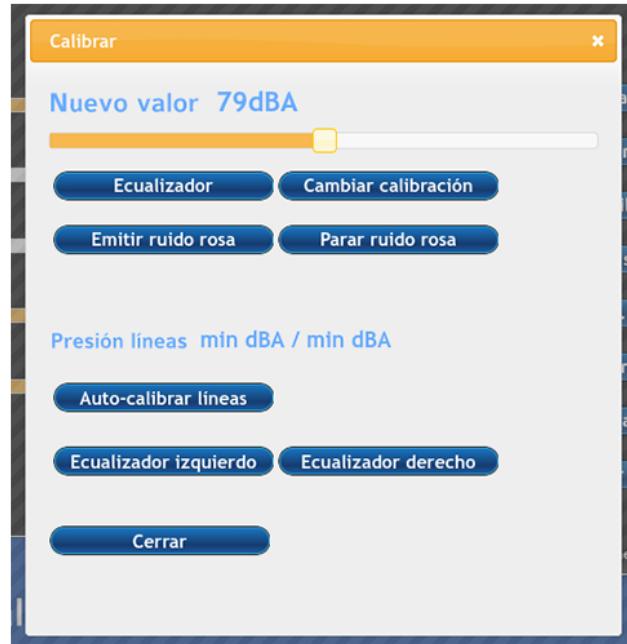


Figura 3.21 – Calibración [1/2].

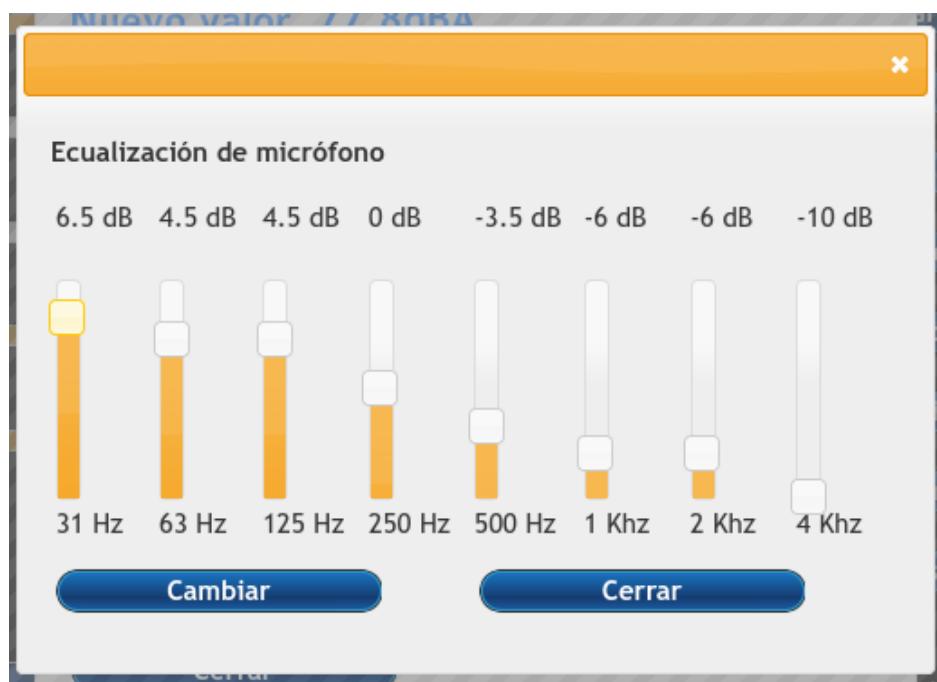


Figura 3.22 – Calibración [2/2].

3.5.2 Análisis técnico

En esta subsección profundizamos un poco más en la interfaz web recurriendo a su código fuente. Bajamos por tanto un escalón importante a nivel de abstracción. Por razones obvias, no se va a incluir la totalidad del código de la interfaz web del limitador en esta subsección, sino que se va a realizar un análisis técnico del mismo, destacando las cualidades o deficiencias más importantes que se han detectado y presentando bloques de código que den soporte al análisis y ayuden a comprenderlo.

Como se ha comentado en la introducción a esta sección, el código de la interfaz web se compone de código PHP, [HTML](#), [CSS](#) y JS. Parte del código [CSS](#) y JS corresponde a librerías estáticas, descargas y usadas por el proyecto, como por ejemplo [jQuery](#), y su plugin [SlideJS](#), [dygraphs](#) o [ExCanvas](#). Estas librerías se usan de forma extensiva para construir toda la parte reactiva y asíncrona de la interfaz, como las acciones lanzadas mediante la pulsación de botones, el envío de datos al servidor y la actualización de datos en la interfaz, sin necesidad de recargar la página; para ello, se hace uso de AJAX. Las dos últimas librerías se usan concretamente para la generación de gráficos, en este caso, la gráfica principal que muestra las lecturas de los sensores y la atenuación aplicada por el limitador en cada momento.

En el esquema 3.6 podemos observar la estructura de la carpeta www dentro del código fuente del limitador, la cual contiene el código de la interfaz web. Por simplicidad, se han omitido algunos los ficheros no esenciales, ya que el listado sería demasiado extenso y la intención es mostrar la estructura y organización del proyecto que materializa la aplicación web, así como describir brevemente el propósito y responsabilidad de cada fichero. Para generar esta representación del directorio y su contenido se ha utilizado la herramienta [GIO](#), disponible en distribuciones de Linux.

```

1  # gio tree .../lms.7/lms/www
2
3  file:///lms.7/lms/www
4      |-- access.php
5          # Gestión de usuarios, sesiones
6          y permisos
7      |-- authPart-ExtendedNormative.php
8      |-- calibrationControl.lib.php
9      |-- config.xml
10     |-- configFail.inc
11     |-- configuration.php
12         # Definición de variables
13         globales
14     |-- css/
15     |-- fonts/
16     |-- functions/
17         |-- calibrate.php
18         # Scripts AJAX
19         |-- network.php
20             # Actualizador de calibración
21             # Controlador de configuración
22             de red
23             |-- pink.php
24                 rosa
25             # Control de la emisión de ruido
26             |-- restart.php
27             |-- soundControl.php
28                 # Consulta el estado del
29             |-- status.php
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
765
766
767
767
768
768
769
769
770
771
772
773
774
775
775
776
777
777
778
778
779
779
780
781
782
783
784
784
785
785
786
786
787
787
788
788
789
789
790
791
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1
```

```

    limitador
19 |     |-- updateCnfg.php                      # Actualizador de configuración
20 |     |-- updateDataCu.php                     # Actualizador de datos del
21 |     local
22 |         |-- updateExtendedNormative.php      # Actualizador de la normativa
23 |         |-- updatePwd.php                   # Actualizador de contraseña
24 |         |-- updateTime.php                 # Actualizador de fecha y hora
25 |         `-- validateUserCode.php
26 |
27 |     |-- help.png                           # Controlador principal
28 |     |-- images/
29 |     |-- index.php                          # Librerías JS (jQuery, dygraph,
30 |     |-- jquery.css                         # Template de ecualizador
31 |     |-- js/
32 |         slide, excanvas...
33 |     |-- leftEqualizer.php
34 |         izquierdo (HTML, CSS, JS, PHP)
35 |     |-- login.php                           alibra
36 |     |-- logoPagina.png
37 |     |-- logs.php                           # Logger
38 |     |-- manual.php
39 |     |-- manual3.pdf
40 |         limitadores serie 3
41 |     |-- manual7.pdf
42 |         limitadores serie 7
43 |     |-- new/
44 |     |-- old/
45 |         aplicación web
46 |             |-- www/
47 |     |-- reportOutputTest.html
48 |     |-- reports/
49 |         informes
50 |             |-- CreateReport.php
51 |             |-- PhpDataGetter.php
52 |             |-- data.php
53 |             |-- data.xml.php
54 |             |-- engine.php
55 |             |-- engine_render.php
56 |             |-- engine_render2.php
57 |             |-- engine_render3.php
58 |             |-- getReport.php
59 |             |-- images/
60 |             |-- index.php
61 |             |-- logoReports.png
62 |             |-- reportContent.php
63 |             |-- reportError.html
64 |             |-- reportError.php
65 |             |-- reportHasNoScreenableOutput.html
66 |             |-- reportHelp.html
67 |             |-- reportTemp -> /tmp/templateTempDirectory/
68 |             |-- saveCreatedReport.php
69 |             |-- testEngine.php
70 |             |-- testSaved.php
71 |             `-- tmp -> /tmp
72 |
73 |     |-- rightEqualizer.php
74 |         derecho (HTML, CSS, JS, PHP)
75 |     |-- scripts.js
76 |         está embebido en los .php)
77 |     |-- simple/
78 |     |-- smallInfoPanel.php
79 |     |-- statusPart.php
80 |     |-- style-Blue.css
81 |     |-- style.css
82 |     |-- tabAuth.php
83 |
84 |         # Template de ecualizador
85 |         # Código JS global (la mayoría
86 |         # Versión simplificada de la web
87 |         # Template del panel superior si

```

```

72      se está autenticado          # Template del panel superior
73      |-- tabAuthRegister.php
74      para el registro          # Template del panel superior si
75      |-- tabNormal.php
76      NO se está autenticado
77      |-- templateMng.php
78      |-- tests/
79      |-- tractis/               # DNI-e (no usado)
80      `-- users.php             # Gestión de usuarios (
81          deprecated en favor de access.php)

```

Listado 3.4 – Estructura de directorios y ficheros de la interfaz web

La página web se genera de forma dinámica mediante PHP, en función de:

- Tipo de usuario (roles y permisos).
- Si el usuario ha iniciado sesión o no.
- Tipo de sistema (registrar o limitador).
- Los datos (actualización asíncrona).

La generación de código dinámico se basa en la renderización condicional de bloques de código fuente, así como la inclusión (a veces también condicional) de otros ficheros con extensión PHP. Es importante destacar que en este aspecto **no se hace uso de ninguna herramienta** que ayude a gestionar la generación dinámica del código, como puede ser un **gestor de plantillas** como Twig. Esto, junto con el **uso de código en varios lenguajes** (PHP, JS, HTML y CSS) de manera simultánea en el mismo fichero e incluso **en el mismo bloque de código** hace que el código de la aplicación web sea **altamente complejo y difícil** de leer, entender y sobretodo, **de mantener**. Podemos observar un ejemplo claro de esto en el siguiente extracto de código, perteneciente al fichero *tabAuth.php*:

```

1     ...
2     <div id="chTime" title="Cambio de hora">
3         <p style="font-size:bigger; font-weight:bold;">Cambio de hora del sistema</p>
4     >
5
6     <p>Hora actual <span id="chTime_actualTime"></span></p>
7     <p>
8         <form id="chTime_form">
9             Nueva hora <input type="text" id="chTime_new" name="chTime_new" style="width
10                :120px" value="" <?php if (!$_configurator){ echo( 'disabled="disabled"' );
11                }?> ></input>
12         </form>
13     </p>
14     <br/>
15      Recuerde: El sistema actualizará automáticamente la hora al estar conectado a internet.
16     <table width=100%>
17         <tr>
18             <td>

```

```

16     <input style="float:right" type="button" value="Cerrar" id="" class=""
17         bt_register" onclick="javascript:$('chTime').dialog('close');;" />
18   </td>
19   <td align=right>
20     <?php if ($configurator){ ?>
21       <input style="float:right" type="button" value="Ok" id="" class=""
22         bt_register" onclick="javascript:ChTime_UpdateClock();" />
23     <?php } ?>
24   </td>
25   </tr>
26   </table>
27   </div>
28   <script>
29     $('#chTime_new').datetimepicker({
30       dateFormat: 'yy/mm/dd'
31     });
32   </script>
33   <script>
34     function ChTime_UpdateClock() {
35       $.ajax({
36         data: $("#chTime_form").serialize(),
37         type: "POST",
38         dataType: "json",
39         url: "functions/updateTime",
40         success: function(data) {
41           alert("Hora actualizada");
42         }
43       });
44     };
45   </script>
...

```

Listado 3.5 – Código HTML, CSS, JS y PHP usado de forma conjunta

El código 3.5 ha sido ligeramente indentado y reestructurado para mejorar su comprensión, sin embargo, al problema que supone desarrollar código como el que acabamos de ver, hay que sumar una indentación prácticamente inexistente, la ausencia casi total de comentarios y la prolongación de líneas de código por encima de los 100 caracteres de longitud.

Tan solo los ficheros tabNormal.php, tabAuth.php y tabAuthRegister.php suman un total de 3000 líneas de código, por lo que la complejidad del código crece rápidamente en un proyecto de estas características cuando se siguen malas prácticas como las que se acaban de ver. Además de las claras deficiencias del código en sí, la interfaz web está **fuertemente acoplada** al sistema operativo y al software del limitador, ya que la comunicación IU-Limitador se realiza exclusivamente mediante ficheros o mediante la invocación directa de procesos / servicios del limitador que se ejecutan sobre el sistema operativo. Estos dos procedimientos de comunicación se verán en detalle en la siguiente sección.

En definitiva, la complejidad del código debido a malas prácticas y las complejidades técnicas que supone el uso de PHP 5 a la fecha de la redacción de este documento provocan que no sea factible la reutilización de la aplicación web, y por tanto, queda

exclusivamente como objeto de estudio y recurso de soporte a nuestro proyecto.

A fecha de 10 de agosto de 2021, la última versión estable de PHP es la versión 8.0.8, con fecha de lanzamiento el 1 de julio de 2021. La versión 5.5.3 de PHP usada en la aplicación web del limitador [LM7](#) fue lanzada el 22 de agosto de 2013, es decir, hace 8 años.

3.5.3 Comunicación IU-Limitador mediante ficheros

Los ficheros son usados de forma muy extensa por el limitador en su conjunto, y en general, interpretan el papel tanto de almacenamiento persistente de datos como vía de comunicación entre procesos, y entre estos y la interfaz web. Estos últimos serán los estudiados en esta sección.

Es tanta la importancia de los ficheros que **el limitador no hace ningún uso de bases de datos**, sino que almacena todo lo que requiere almacenar en ficheros de texto o directamente en formato binario. El control (escritura y lectura) de estos ficheros requiere el uso de procesos diseñados e implementados exclusivamente para este propósito, y que se identificarán en la siguiente sección.

Los ficheros que utiliza la interfaz web son los siguientes:

- /var/slrl/users.auth
 - Almacén de cuentas de usuario, contraseñas y permisos.
 - Accedido por access.php.
- /var/slrl/logs.serial
 - Registros (logs) sobre las acciones que realizan los usuarios.
 - Accedido por logs.php
- /var/slrl/sessions.serial
 - Registros (logs) sobre las sesiones de calibración.
 - Se indica el comienzo y el final de la cada **sesión¹**, así como el valor de calibración.
 - También registra la hora de arranque del sistema.
 - Almacenado en formato **base64**.

¹Una sesión corresponde al intervalo de tiempo durante el cual el limitador se encuentra funcionando de forma activa debido a que los valores de recepción están por encima de un determinado umbral

- Accedido por logs.php.
- /var/slrf/network.config
 - Almacena la configuración de red del equipo para su lectura.
 - Accedido en varias partes de la UI.
- /var/slrf/network.script
 - Almacena la configuración de red del equipo para su actualización.
 - Corresponde a un script Bash que hace uso del comando ifconfig y route.
 - Accedido en varias partes de la UI.
- /tmp/inSession
 - Fichero vacío para indicar si el equipo se encuentra en una [sesión](#), a modo de variable de estado.
 - Los procesos que necesitan conocer esta información detecta el archivo y actúan en consecuencia [fopen()].
 - Se elimina al finalizar la sesión.
 - Accedido por calibrationControl.php.
- /tmp/conf.tmp
 - Fichero temporal para almacenar la configuración recientemente cambiada por el usuario desde la UI.
 - Inmediatamente después de la actualización de este archivo por parte de la UI, se invoca a un proceso del limitador para que recoja esta nueva información y aplique la nueva configuración.
 - Accedido por varias partes de la UI.
- /tmp/pink
 - Fichero vacío que indica si se debe emitir ruido rosa, a modo de variable de estado.
 - Los procesos que necesitan conocer esta información detectan el archivo y actúan en consecuencia (emitir ruido rosa) [fopen()].
 - Accedido por calibrationControl.php.
- /tmp/externalAttenuation
 - Utilizado para indicar si se debe aplicar un nuevo valor de atenuación.
 - Almacena el valor de atenuación indicado por el usuario en la UI.

- El proceso principal del limitador detecta este fichero, lee el nuevo valor de atenuación contenido en él, y lo aplica.
- Accedido por `calibrationControl.php`.

Todos los ficheros son de relativa importancia ya que desempeñan un papel en el limitador, pero de entre todos ellos destaca notablemente por su importancia el fichero `/tmp/conf.tmp`. Este fichero es la principal vía de comunicación entre la interfaz web y el limitador, y a través de él fluyen todas las configuraciones definidas por el instalador o el usuario. En primer lugar, se configura el sistema desde la interfaz web y mediante botones, se aplica la nueva configuración. El proceso de configuración es transparente para el usuario de la interfaz y tiene efecto inmediato, pero lo que en realidad ocurre es que esta nueva configuración ha sido volcada al fichero `/tmp/conf.tmp` para que sea procesada por el proceso del limitador encargado de ello. Por supuesto, los contenido del fichero debe estar en un formato que ambas partes conozcan de forma que la comunicación pueda llevarse a cabo: la **interfaz**.

Para más información sobre el fichero `conf.tmp`, puede consultarse en anexo [D.2.1](#).

3.5.4 Comunicación IU-Limitador mediante procesos

En esta sección se presenta una lista de los módulos software del limitador sobre los que se apoya la interfaz web, y que son invocados de forma directa desde ella. Estos programas se analizarán con mayor profundidad en la sección [Procesos del limitador LM7](#), junto con el resto de programas que componen el núcleo software del limitador.

- | | |
|--|---|
| <ul style="list-style-type: none"> • <code>getConfig</code> • <code>getStatus</code> • <code>getData</code> | <ul style="list-style-type: none"> • <code>limInfo</code> • <code>utilslr</code> • <code>localservice</code> |
|--|---|

Estos programas son invocados desde la interfaz web mediante la función `popen()` de PHP.

3.6 Estructura del proyecto

De la misma forma que se hizo en con la interfaz web, en este apartado se va a presentar la estructura de directorios del proyecto y sus ficheros más importantes. De nuevo, algunos de ellos se han eliminado del listado por ser ficheros no útiles para el proyecto, normalmente restos de versiones anteriores. Asimismo, los ficheros que contiene el código de la interfaz web se ha omitido ya que ha sido estudiado en el listado [3.6](#).

El anexo LM7: documentación del código fuente supone una ampliación de información considerable sobre esos archivos.

```
1 # gio tree .../lms.7/lms/
2
3 file:///lms.7/lms
4 |-- Makefile
5 |-- comun
6 |   |-- Calibracion.cpp
7 |   |-- Calibracion.h
8 |   |-- Config.h
9 |   |-- Configuracion.cpp
10 |   |-- Configuracion.h
11 |   |-- Configurador.cpp
12 |   |-- Configurador.h
13 |   |-- EstadoDeModificadores.cpp
14 |   |-- EstadoDeModificadores.h
15 |   |-- EstadoDelLimitador.cpp
16 |   |-- EstadoDelLimitador.h
17 |   |-- Funciones.cpp
18 |   |-- Funciones.h
19 |   |-- Makefile
20 |   |-- Serial.cpp
21 |   |-- TestConfig.cpp
22 |   |-- aes.cpp
23 |   |-- aes.h
24 |   |-- compacc
25 |   |-- compact.cc
26 |   |-- gSerial
27 |   |-- getSerial.cpp
28 |   '-- serial.h
29 '-- comv2
30 |   |-- Funciones.h
31 |   |-- Main.cpp
32 |   |-- Makefile
33 |   |-- Test.cpp
34 |   |-- TestRegistrador.cpp
35 |   |-- WhiteRabbit.cpp
36 |   |-- WhiteRabbit.h
37 |   |-- base64.c
38 |   |-- cliente.c
39 |   |-- encrypter.cpp
40 |   |-- funciones_Conexion.cpp
41 |   |-- funciones_envioDeDatos.cpp
42 |   |-- funciones_formato.cpp
43 |   |-- funciones_hora.cpp
44 |   |-- globales.h
45 |   '-- slrComm
46 '-- configuracion
47 '-- connectNetwork
48 '-- construye
49 '-- defines.txt
50 '-- instalador
51 |   |-- MAKEDEV
52 |   |-- fdiskCommands
53 |   |-- lm701f.localCommands
54 |   |-- sdbInstall
55 |   '-- sdcInstall
56 '-- limitador
57 |   |-- Atenuador.cpp
58 |   |-- AtenuadorMk163.cpp
59 |   '-- AtenuadorPga.cpp
```

```

60   |   |-- AtenuadorPga.h
61   |   |-- Calibracion.cpp
62   |   |-- Calibracion.h
63   |   |-- CalibrationTest.cpp
64   |   |-- Estado.cpp
65   |   |-- Estado.h
66   |   |-- Lector.cpp
67   |   |-- LectorAlternativo.cpp
68   |   |-- Lectura2C.cpp
69   |   |-- Limitador.cpp
70   |   |-- Makefile
71   |   |-- PruebaAtenuador
72   |   |-- PuertoParalelo.cpp
73   |   |-- Registrador.cc
74   |   |-- Registro.cpp
75   |   |-- Registro.h
76   |   |-- ServidorDeEstado.cpp
77   |   |-- ServidorDeEstado.h
78   |   |-- Sonometro.cpp
79   |   |-- Sonometros.cpp
80   |   |-- SwitchPink.cpp
81   |   |-- TP.cpp
82   |   |-- TestAtenuador.cpp
83   |   |-- TestPga.cpp
84   |   |-- audioOrder
85   |   |-- filterTest.cpp
86   |   |-- filtroA
87   |   |   `-- filtroA.cpp
88   |   |-- iir.cpp
89   |   |-- limitador
90   |   |-- mixers
91   |   |-- octaveBandFilter.c
92   |   |-- qир
93   |   |   |-- compila.sh
94   |   |   |-- datosSOS_fbank_scaled.h
95   |   |   |-- in_int.dat
96   |   |   |-- out_int.dat
97   |   |   |-- pruebabairq
98   |   |   |-- pruebabairq.c
99   |   |   |-- qир.c
100  |   |   |-- qир.h
101  |   |   |   |-- tQир.c
102  |   |   |   `-- tQир
103  |   |   |-- sonometro
104  |   |   |-- sonometros
105  |   |   |-- testAt
106  |   |   |-- testDeCalibracion.cpp
107  |   |   `-- testPga
108  |-- preparatoria
109  |-- reports
110  |   |-- Makefile
111  |   |-- cabeceraInformes.svg
112  |   |-- dataGet.cpp
113  |   |-- dygraph-combined.js
114  |   |-- excanvas.min.js
115  |   |-- gRegList.php
116  |   |-- gReport.php
117  |   |-- getConfig.cpp
118  |   |-- getData.cpp
119  |   |-- getEvents
120  |   |-- getInputs.cpp
121  |   |-- getStatus.cpp
122  |   `-- testPerformance.php

```

```

123 |   |-- scripts
124 |   |   |-- connectNetwork
125 |   |   |-- continuousPink
126 |   |   |-- controlDeCalibracion
127 |   |   |-- keepComm
128 |   |   |-- keepLeds
129 |   |   |-- keepLm
130 |   |   |-- keepScreen
131 |   |   |-- keepTime
132 |   |   |-- killComm
133 |   |   |-- ledControl
134 |   |   |-- playPink
135 |   |   |-- refreshProcess
136 |   |   `-- remoteShellService
137 |   |-- setAsNew
138 |   |-- tests
139 |   |   |-- estadisticos.cpp
140 |   |   |-- estadisticos.php
141 |   |   |-- normativaExtendida.cpp
142 |   |   |-- tActivo
143 |   |   |-- tActivo.cpp
144 |   |   |-- tReg.cpp
145 |   |   |-- tcal
146 |   |   `-- testDeCal.cpp
147 |   |-- utiles
148 |   |   |-- AutoEqualizador.cpp
149 |   |   |-- BoanergesConfigurator.cpp
150 |   |   |-- Inicializador.cpp
151 |   |   |-- LimInfo.cpp
152 |   |   |-- Linet.cpp
153 |   |   |-- Makefile
154 |   |   |-- ServidorSerie.cpp
155 |   |   |-- autoEq
156 |   |   |-- controlDeCalibracion
157 |   |   |-- gSerial
158 |   |   |-- gen
159 |   |   |-- gen.cpp
160 |   |   |-- getSerial.cpp
161 |   |   |-- helper.cpp
162 |   |   |-- playPink
163 |   |   |-- serial.h
164 |   |   |-- utilslr.cpp
165 |   |   `-- x.cpp
166 `-- www

```

Listado 3.6 – Estructura de directorios y ficheros de la interfaz web

El proyecto se encuentra segmentado en carpetas, en lo que se asume que es un intento de organizar el código por categorías en base a funcionalidades comunes. Cada uno de los directorios, además del conjunto de ficheros que compone el código fuente, suele contener un fichero Makefile mediante el cual compilar el código fuente y generar ejecutables de forma automatizada, por lo que cada carpeta, por lo general, proporciona uno o varios ejecutables. Aunque en esta versión el concepto no termina de tomar significado, de aquí en adelante nombraremos a cada uno de estos directorios **módulo**, ya que a priori, parece que la idea de los programadores originales era el generar distintos módulos que trabajasen de forma conjunta, y de ahí la separación en distintas carpetas.

Esta segregación sin embargo, lejos de mejorar la situación, la empeora considerablemente, añadiendo una complejidad innecesaria debido a las siguientes razones:

1. Aunque a primera vista parece que los ejecutables y su código queda perfectamente englobado en su directorio, la realidad es que prácticamente todos ellos necesitan de código pertenecientes a otros módulos, por lo que los módulos acaban siendo muy interdependientes y las constantes inclusiones a código que se encuentra en otras carpetas acaba complicando y ensuciando el código.
2. Se ha decidido crear un Makefile para cada módulo en lugar de un solo Makefile global que se encargue de gestionar la compilación y el enlazado del código, por lo que en lugar de gestionar un solo Makefile hay que gestionar tantos Makefiles como módulos haya más el Makefile general, el cual se encarga de invocar a cada uno de los Makefiles de los módulos.

Cada uno de esos ficheros ha sido analizado en profundidad. Para ello, se ha generado una hoja de cálculo en la que se lista cada uno de estos ficheros, su propósito, su funcionalidad y anotaciones sobre su diseño o importancia. Esta hoja de cálculo se adjunto al documento actual como el anexo F. **Este documento supone la columna vertebral de todo el proceso de ingeniería inversa**, ya que en él se vuelca todo el conocimiento adquirido sobre el sistema, su funcionamiento, y su diseño.

3.7 Procesos del limitador LM7

El software del LM7 se encuentra generalmente implementado en C++, aunque también hace un uso importante de scripts de Bash y PHP (sin tener en cuenta la interfaz web). Por lo general, estos programas tienen la responsabilidad de dar soporte al limitador, y no influyen de manera directa en el proceso de control acústico. Tan sólo el proceso limitador junto con algunos scripts en PHP son responsables de esta función.

El software del limitador estudiado que se encuentra implementado en C++ consta de los siguientes programas:

- getConfig
 - Devuelve los datos del local y la configuración del limitador en formato XML o JSON.
- getStatus

- Devuelve los datos del local e información sobre el estado del limitador en formato [XML](#) o [JSON](#):
 - * Tipo de sistema (registrador o limitador).
 - * Lecturas.
 - * Atenuación actual.

- `getData`

- Devuelve los registros del limitador.
- Usado para generación de reportes desde la aplicación web.
- Recibe 4 parámetros:

- * Formato: [XML](#) o [JSON](#)
- * Fecha de inicio: desde la fecha base del registro hasta la fecha actual.
- * Fecha de fin: desde la fecha de inicio hasta la fecha actual.
- * Intervalo de muestreo: intervalo de tiempo entre cada registro, en minutos.

La fechas van en el formato YYYY/MM/DD-hh:mm, siguiendo el estándar [ISO 8601](#).

- `limInfo`

- Devuelve información general sobre el limitador.
- Recibe como parámetro una cadena de caracteres, en la que cada carácter implica la consulta de una propiedad. Por ejemplo, "limInfo 5" muestra la dirección de local, "limInfo A" muestra el valor de atenuación actual, y "limInfo 5A" muestra ambas cosas.
- En el anexo [C.2.1](#) se puede consultar el listado completo de opciones que admite este programa.

- `utilslr`

- Proporciona utilidades para la verificación de cuentas y la generación de informes en varios formatos.
 - * Gráficos y listados.
 - * Volcados [SQL](#), [SQLite](#), [YML](#), [XML](#) y [JSON](#).

- `localservice`

- Su finalidad es presentar un intérprete de comandos con el cuál interactuar con el limitador, con sus datos y con su configuración.
- Recibe una gran cantidad de parámetros.

- Permite configurar el sistema a partir del contenido del fichero de texto plano `/tmp/conf.tmp`.
- Permite lanzar el proceso de calibración.
- Permite consultar y actualizar los valores de ecualización del micrófono y de las líneas.
- Implementa un bucle infinito, hasta que se le indique salir.
- Muchas de las funciones que permite realizar ya las realizan los programas mencionados anteriormente (redundancia).
- **limitador**
 - Es el programa principal, la piedra angular del limitador.
 - Puede funcionar como limitador y/o como registrador.
 - Controla el atenuador [PGA](#).
 - Controla el gestor de registro (registrador).
 - Controla los sonómetros.
 - Implementado como un bucle infinito que toma lecturas desde los sonómetros (los cuales leen los datos que pasan por las tarjetas de sonido) y decide la atenuación que debe aplicarse en cada momento según la normativa configurada.
 - En la sección siguiente se analizará en detalle el funcionamiento de este programa.
- **slrComm**
 - Implementa un socket HTTP en C++.
 - Una vez abierto el socket, registra el equipo en el servidor remoto configurado y mantiene el socket a la escucha para poder establecer una comunicación en ambos sentidos (full-dúplex).
 - Los mensajes son cifrados/descifrados en cada extremo de la comunicación, mediante el uso de claves.
- **inicializador**
 - Inicializa y levanta el registrador.
 - Al inicializar, se reserva en disco el espacio disponible escribiendo en el fichero `/var/slrm/registro.slr` tantos registros como sea posible.
 - Escribe en el fichero de registro la cabecera.
 - Inicializa la hora base del registro a la hora actual.

- `boanerges.config`
 - Permite configurar el servidor remoto (IP, puerto, claves...), así como consultar sus datos.
- `autoEq`
 - Ecualiza la línea izquierda y derecha de forma automática.
 - Toma una serie de muestras (por defecto 3000) de los valores de micrófono y líneas (recepción y emisión) y genera sus medias.
 - Calcula la ecualización: para cada banda de frecuencia, la ecualización es igual al valor medio leído por el micrófono menos el valor medio de emisión en esa línea.

3

Para convertir estos programas en servicios (procesos tipo [Daemon](#)), se utilizan scripts Bash que implementan un bucle infinito en el cual invocan a estos programas o a otros scripts. De esta forma, la ejecución del script queda bloqueada a la espera de que el programa al que invoca finalice. Este enfoque se utilice también en el caso del programa. De esta forma, si el proceso encargado del control acústico falla por algún motivo, el script se encarga de reiniciarlo.

Los scripts pueden consultarse en el listado [3.6](#).

Capítulo 4

Ingeniería Inversa: la versión LM7

Hasta el momento solo se ha presentado el limitador bajo estudio, y se ha descrito brevemente la estructura y las funcionalidades de su software. En esta sección entraremos en detalles sobre el funcionamiento del software mediante la presentación de diagramas que ilustren el diseño del sistema y nos permita comprender su funcionamiento con un mayor nivel de detalle.

En las figuras [4.1](#), [4.2](#) y [4.3](#) tenemos un diagrama de contexto, un diagrama de contexto extendido y el diagrama de clases del programa de atenuación del [LM7](#). En los anexos [C](#) pueden consultarse los diagramas de clases del resto de programas que componen el limitador.

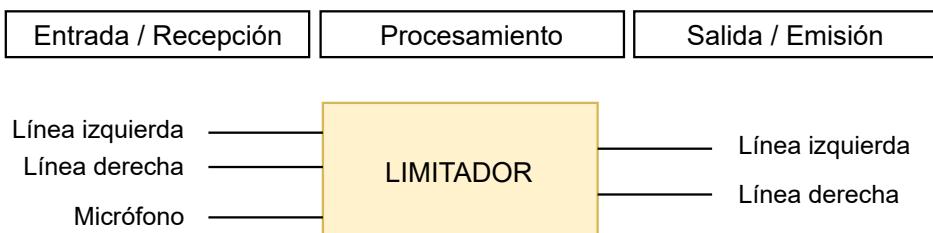


Figura 4.1 – Diagrama de contexto.

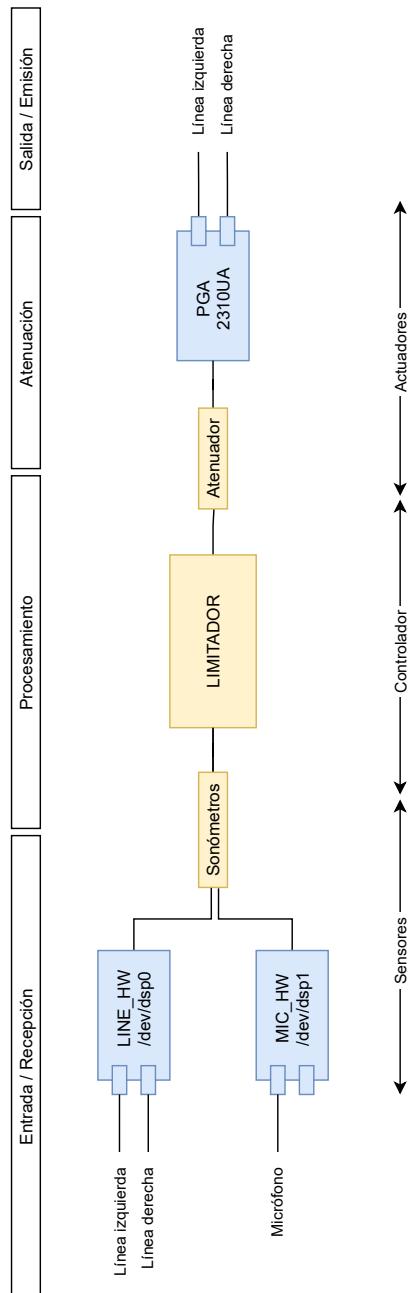


Figura 4.2 – Diagrama de contexto extendido.

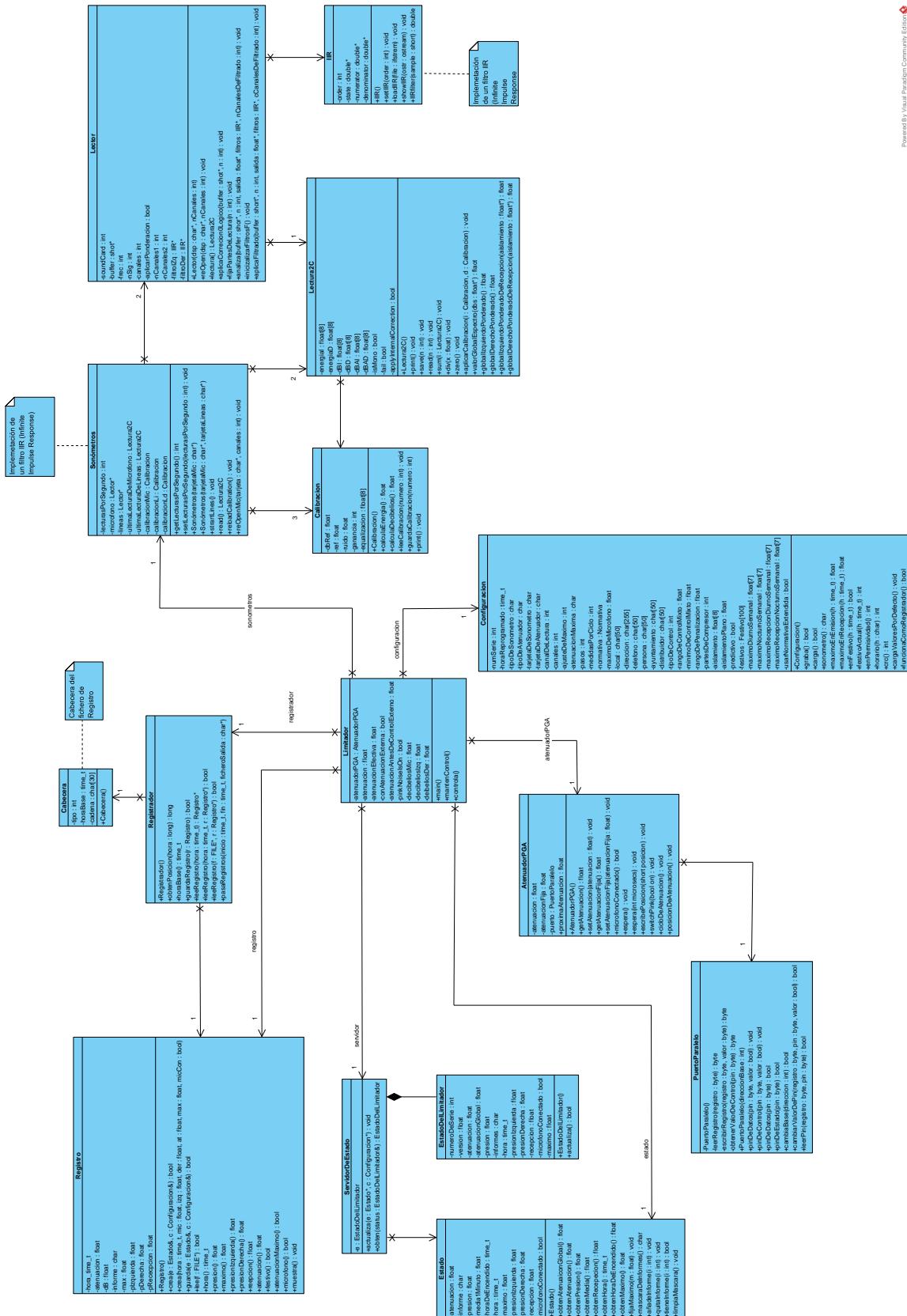


Figura 4.3 – Diagrama de clases del LM7.

Limitador de sonido para locales de música

4.1 Arranque

En la sección de [Análisis del rendimiento](#) se identificaron los programas del limitador analizando los procesos que se estaban ejecutando en la máquina. Además de identificar los procesos, se comentó que todos ellos tienen como “padre” al proceso con [PID 1](#), el proceso `init`.

El proceso `init` o `systemd` es un gestor de servicios para sistemas operativos Linux. Es el primer proceso en arrancar (con [PID 1](#)) y el último en acabar durante el apagado. Su función es levantar los procesos a nivel de usuario.

Para el arranque del sistema, el [LM7](#) dispone de un script de arranque en el directorio `/etc/init.d`. Este directorio contiene una serie de scripts Bash que sirven para definir servicios, así como para proveer de la funcionalidad necesaria para poder iniciar, apagar, reiniciar comprobar el estado de los servicios.

El script de arranque del [LM7](#) tiene el nombre de *Inicializador*. En [4.1](#) tenemos su código, y se puede observar que carece de la estructura y las funcionalidades requeridas por `init`. En el anexo E se puede consultar la estructura a la que se hace referencia.

```

1 # root@Limitador:~# cat /etc/init.d/Inicializador
2
3 ifconfig eth1 192.168.1.223 netmask 255.255.255.0
4 route add -net default gw 192.168.1.1
5 echo "" >/tmp/mtab
6 mount /dev/hda3 /var/slr
7
8 #!/bin/sh
9 Linux_string=Limitador
10
11 echo "Iniciando..." 2>/dev/null
12
13 PATH=/bin:/sbin:/usr/sbin:/usr/bin 2>/dev/null
14
15 # replace the ramdisk with the tmpfs for /var and /tmp
16 echo $Linux_string: /var y /tmp temporales 2>dev/null
17 mkdir -p /dev/shm/log/apache2
18 mkdir -p /dev/shm/run
19 mkdir -p /dev/shm/lock
20 mkdir -m 777 /dev/shm/tmp /dev/shm/php4 /dev/shm/php5 2>/dev/null
21 mount -o bind /dev/shm/tmp /tmp 2>/dev/null
22 mount -o bind /dev/shm/php4 /var/lib/php4 2>/dev/null
23 mount -o bind /dev/shm/php5 /var/lib/php5 2>/dev/null
24
25 mkdir /tmp/lock
26
27 mount /dev/sda3 /var/slr
28
29 echo $Linux_string: P=192.168.1.223 2>/dev/null
30
31 for interfaz in eth usb; do

```

```

32  for i in $(seq 0 9); do
33    ifconfig $interfaz$i 192.168.1.223 netmask 255.255.255.0;
34  done;
35 route add -net default gw 192.168.1.1
36
38 cd /lms/
39 ./preparatoria
40 cd /
41
42 /usr/local/bin/ntpdate &
43 chmod 777 /var/slrm/configuracion
44
45 apache2ctl start &
46
47 #/bin/soundInputs.gen
48
49 amixer sset Line 5% cap -c 0
50 amixer sset Mic 6% cap -c 1
51 aumix -d /dev/mixer -lr
52
53 /bin/keepLm &
54 /bin/keepTime &
55
56 chmod 777 /var/slrm
57
58 /bin/keepComm &
59
60 /var/slrm/network.script &
61
62 /bin/hostname $Linux_string 2>/dev/null
63
64 echo $Linux_string: Configurando red 2>/dev/null
65 /sbin/ifconfig lo 127.0.0.1 up 2>/dev/null
66 /sbin/route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 dev lo 2>/dev/null
67
68 chmod 777 /dev/*
69 chmod -R 777 /tmp
70 echo "" >/tmp/conf.tmp
71 chmod 777 /tmp/conf.tmp
72
73 echo "" >/tmp/cuVerified
74
75 /bin/adjtime &
76
77 in.telnetd -debug 1035 -L /bin/localservice &
78
79 amixer sset Line 5% cap -c 0
80 amixer sset Mic 6% cap -c 1
81 aumix -d /dev/mixer -lr
82
83 #Configuración de puertos
84 a=$(dmesg | grep "ttyU" | egrep "uart|serial|FTDI")
85 b=$(echo ${a##*ttyUSB})
86 rm /var/slrm/leds/port -rf
87 ln -s /dev/ttyUSB$b /var/slrm/leds/port
88
89 c=$(dmesg | grep ttyU | grep "modem" | sed -n 1p)
90 d=$(echo ${c##*ttyUSB})
91 rm /tmp/modemPort -rf
92 ln -s /dev/ttyUSB$d /tmp/modemPort
93
94 playPink >/dev/null 2>/dev/null &

```

```

95 /bin/controlDeCalibracion &
96 /bin/refreshProcess &
97 /bin/keepLeds &
98 /bin/watchDog &
99 /bin/remoteShellService &

```

Listado 4.1 – Script de arranque primario del LM7.

```

1 ./preparatoria           # Ejecuta los makefile y mueve los binarios y
2   los scripts a la carpeta /bin.
3 /bin/keepLm &            # Mantiene activo el programa limitador.
4 /bin/keepTime &          # Mantiene el sistema en hora.
5 /bin/keepComm &          # Mantiene la conexión a internet (3G <->
6   Ethernet).
7 /var/slrv/network.script & # Mantiene la configuración de red.
8 playPink >/dev/null 2>/dev/null & # Emisión continua de ruido rosa.
9 /bin/controlDeCalibracion & # Servicio de verificación de la calibración.
10 /bin/refreshProcess &    # Mantiene activo el socket HTTP.
11 /bin/keepLeds &          # Controla el LED.
12 /bin/remoteShellService & # Mantiene activo el programa localservice.

```

Listado 4.2 – Scripts de arranque secundarios del LM7.

4

4.2 Detección de micrófono

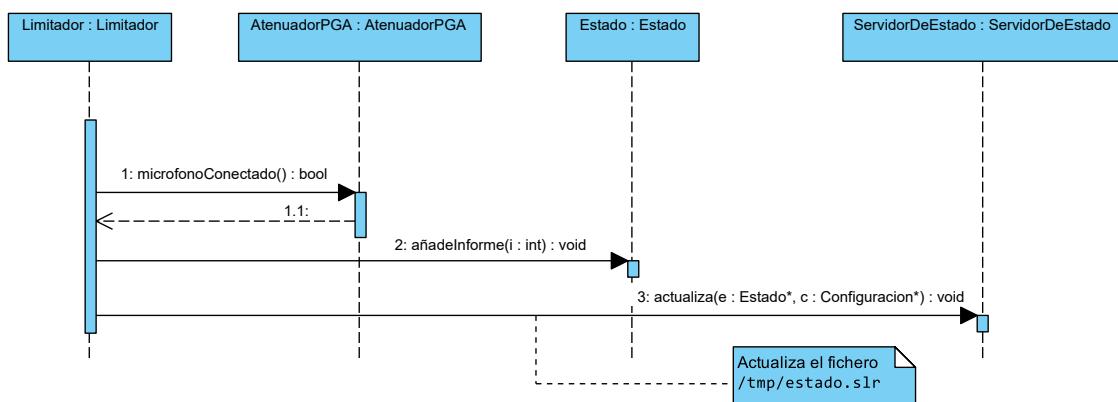


Figura 4.4 – Diagrama de secuencia: detección de micrófono en LM7

El limitador no puede funcionar correctamente si el sensor (micrófono) de limitador no se encuentra conectado al equipo. Para detectar si está o no conectado, el programa principal (limitador) **consulta su estado leyendo desde el pin de estado número 3 del puerto paralelo**.

```

1 #define RegistroDeEstado 1
2 #define PinDeMicrofono 3
3
4 bool AtenuadorPga::microfonoConectado()
5 {
6     return puerto.leerPin(RegistroDeEstado, PinDeMicrofono);
7 }

```

Listado 4.3 – Detección de micrófono en C++.

4.3 Procesamiento de audio

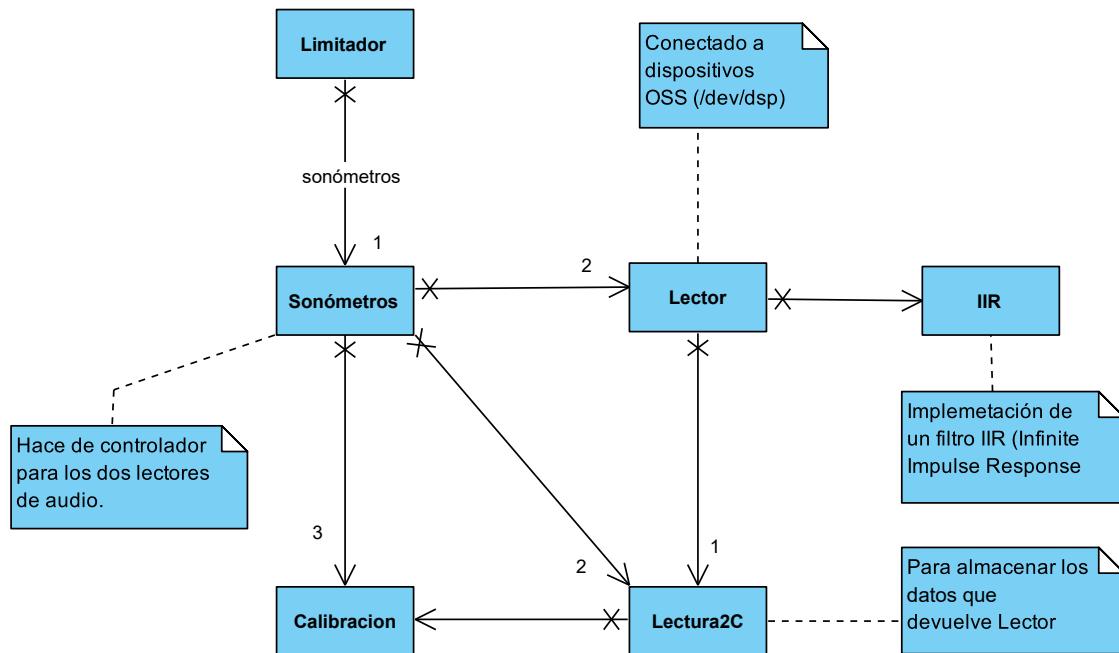


Figura 4.5 – Diagrama de clases simplificado del analizador de audio del LM7.

4

Para que el limitador pueda funcionar es necesario generar un modelo con el trabajar. El audio entra al limitador mediante las entradas balanceadas XLR. Esta señal no deja de ser una señal eléctrica, hasta este punto analógica. La tarjeta de sonido la transforma a una señal digital, de la que es necesario extraer un modelo de presión acústica, es decir, medir esa señal en decibelios.

4.3.1 Conexión a tarjetas de sonido

La conexión a las tarjetas de sonido se realiza mediante la API de **Open Sound System (OSS)**, natural de los sistemas Unix y Unix-like. La API está diseñada para utilizar el framework tradicional de Unix para manejo de ficheros, es decir mediante las operaciones `read()`, `write()`, `open()` e `ioctl()`. En este caso los ficheros son realmente dispositivos, pero se puede acceder a ellos mediante los ficheros especiales de dispositivos. Por ejemplo, el dispositivo por defecto para la entrada y salida de audio es `/dev/dsp`.

```

1 Lector::Lector(char *fichero, int nCanales)
2 {
3     frec = S_Frec;
4     nSig = frec;
5     canales = nCanales;
6     applyAWeight = true;
7     reOpen(fichero, nCanales);
8     inicializaFiltrosF();
9 }

```

Listado 4.4 – Constructor de la clase Lector.

```

1 void Lector::reOpen(char *fichero, int nCanales)
2 {
3     if (sd > 0)
4         close(sd);
5     sd = open(fichero, O_RDONLY, 0);
6     int form = AFMT_S16_LE;
7
8     ioctl(sd, SNDCTL_DSP_SETFMT, &form);
9     ioctl(sd, SNDCTL_DSP_CHANNELS, &canales);
10    ioctl(sd, SNDCTL_DSP_SPEED, &frec);
11
12    if (form != AFMT_S16_LE)
13        puts("Cambio de formato");
14    if (canales != nCanales)
15        puts("Número de canales cambiado");
16    if (frec != S_Frec)
17        puts("Frecuencia cambiada");
18 }

```

Listado 4.5 – Apertura de un dispositivo de audio en el LM7.

4.3.2 Interpretación de datos

Una vez conectados al dispositivo de audio podemos capturar los datos de audio que por él transcurren. Para poder interpretar los datos hay que configurar los parámetros de la señal, como la frecuencia, el formato del audio, el número de canales... Esta configuración debe darse antes de leer cualquier muestra desde el flujo de datos. Como los datos “fluyen” por la tarjeta de sonido, a partir de ahora nos referiremos al conjunto de datos que transcurren por el dispositivo como **flujo**.

Los datos se leen desde el flujo de datos por bloques de tamaño configurable (normalmente múltiplos de 2). Para flujos de más de un canal, los datos se dan en modo intercalado (*INTERLEAVED*). Por ejemplo para un flujo estéreo (2 canales), primero se dan los datos del canal izquierdo y luego los del derecho, de manera intercalada. Mientras que el micrófono utiliza tan sólo un canal (mono), la entrada de audio es estéreo, por lo que hay que separar los datos y almacenarlos en buffers de datos distintos para poder procesarlos.

(bytes) LL-RR-LL-RR-LL-RR...

Una vez se tiene los datos en el buffer correspondiente, se procesan mediante un *Filter Bank* [27]¹. Tras este proceso, obtenemos la descomposición de la señal en bandas de frecuencia, de las cuales obtenemos la energía. A partir de la energía podemos calcular los decibelios en dicha banda mediante la aplicación de la fórmula B.o.1, teniendo en éste cálculo especial importancia la calibración, ya que se utilizará como valor de referencia. Tras calcular los decibelios de todo el espectro, se puede calcular la presión acústica global como una ponderación en promedio (media).

Para la generación de los filtros se ha utilizado la herramienta mkFilter [9]. La orden utilizada para su generación se encuentra comentada en uno de los ficheros del analizador de audio, y se incluye en el listado 4.6

```

1  /* Digital filter designed by mkfilter/mkshape/gencode  A.J. Fisher
2   ./mkfilter -Bu -Hp -o 6 -a 6.4399092971e-02 0.000000000e+00 -l | ./gencode
```

Listado 4.6 – Generación de filtros con mkfilter.

4.3.3 Almacenamiento de audio

Los ficheros de audio se almacenan en formato binario en los ficheros mencionados a continuación de este párrafo. La clase Lectura2C es la única capaz de procesar estos ficheros ya que la escritura y la lectura de los datos consiste simplemente en un **volcado de memoria a disco** de la instancia de la clase Lectura2C. Esta clase contiene el modelo y funcionalidad requerida para exportar e importar los datos. Los atributos que componen estos ficheros de audio se listan en el código 4.7.

- /tmp/.lx0: audio de micrófono.
- /tmp/.lx1: audio de línea izquierda.
- /tmp/.lx2: audio de línea derecha.

```

1 #define NBandasOctava 8
2
3 int64_t energiaI[NBandasOctava];
4 int64_t energiaD[NBandasOctava];
5
6 float dBI[NBandasOctava];
7 float dBd[NBandasOctava];
8
9 float dBAI[NBandasOctava];
10 float dBAD[NBandasOctava];
11
12 bool isMono;
```

Listado 4.7 – Datos miembro de la clase Lectura2C.

¹Un banco de filtros consiste en un array formado por más de un filtro paso banda que separa la señal de entrada en varias componentes, cada una de las cuales transporta la subbanda de una sola frecuencia de la señal original

4.4 Gestión de calibración

4.4.1 Proceso de calibración

El proceso de calibración se inicia desde la aplicación web (imagen 3.21). Para poder calibrar los sensores es necesario emitir ruido rosa en la sala a máxima capacidad de volumen.

Para la **calibración del micrófono**, se debe emitir ruido rosa y a continuación medir los decibelios que hay en la sala mediante el uso de un sonómetro profesional. En la interfaz web, ajustamos el valor de calibración del micrófono desplazando el slider hasta que el nuevo valor marcado se ajuste al valor medido por el sonómetro. Cuando ambos valores coincidan, presionamos el botón “Cambiar calibración”. En ese momento, se comunica el nuevo valor de calibración al limitador mediante uno de sus programas auxiliares (en concreto `localservice`) y se actualiza la nueva calibración.

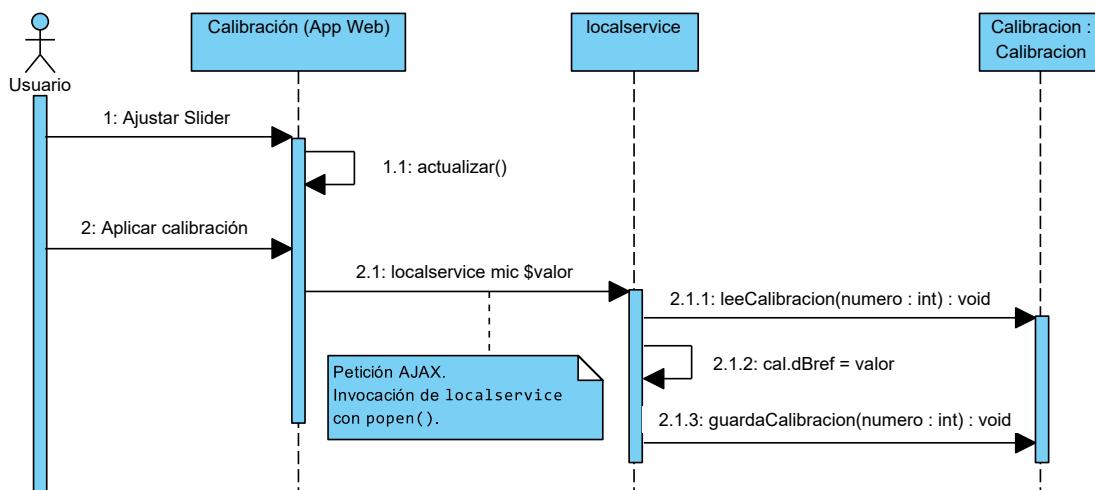


Figura 4.6 – Diagrama de secuencia: calibración de micrófono.

La **calibración de las líneas** se realiza de forma automática por parte del limitador. Como requisitos, el micrófono debe estar calibrado, la emisión de ruido rosa debe permanecer activa durante el tiempo que dure el proceso y los máximos del local (máximos de emisión y recepción regidos por la normativa y la acústica del local) deben estar correctamente configurados.

El proceso de calibración consiste en la toma de una serie de muestras, para luego calcular sus medias (en decibelios ponderados) y finalmente actualizar las nuevas calibraciones en base a las medias calculadas.

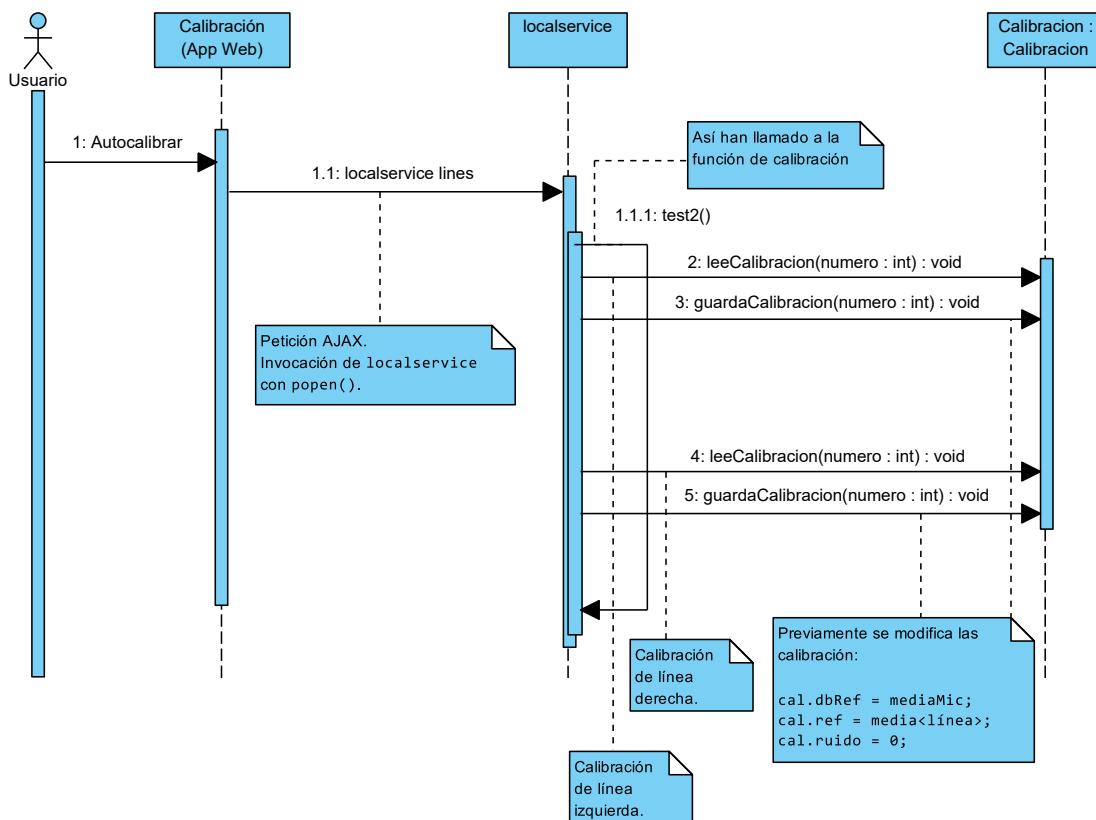


Figura 4.7 – Diagrama de secuencia: calibración de líneas.

4

4.4.2 Ecualizaciones

Las calibraciones constan de un vector de tipo `float` para representar la ecualización. Este vector es de tamaño igual al número de bandas de frecuencia del limitador; en el caso del LM7: 8 bandas. La configuración de la ecualización se realiza mediante la interfaz web, ajustando los sliders que se pueden ver en la imagen 3.21. Cuando se hace click en el botón “Cambiar” los datos se envían al programa `localservice` como argumentos en su llamada, quién actualiza la ecualización en el fichero de calibración correspondiente. El envío se hace de forma asíncrona con AJAX. Para conocer más sobre el programa `localservice` consulte el anexo C.2.2.

La aplicación de la ecualización de cada una de las frecuencias se basa en la suma del valor de la ecualización al valor del sonómetro, como pude comprobarse en el código 4.11.

```

1 void aplicaCalibracion(Calibracion ki, Calibracion kd)
2 {
3     for (int i = 0; i < NBandasOctava; i++)
4     {
5         // Decibelios
6         dBI[i] = ki.calculadBs(energiaI[i]) + ki.equalization[i];
7         dBD[i] = kd.calculadBs(energiaD[i]) + kd.equalization[i];
8
9         // Decibelios ponderados
10        dBAI[i] = dBI[i] + FiltroA[i];
11        dBAD[i] = dBD[i] + FiltroA[i];
12    }
13}

```

Listado 4.8 – *Calculo de decibelios y aplicación de la ecualización.*

4.4.3 Emisión de ruido rosa

La emisión de ruido rosa viene dada por el programa limitador. En cada ciclo de su bucle comprueba si el fichero pink existe, en cuyo caso reproduce el ruido rosa. **El ruido rosa se reproduce de forma continua en una de las tarjetas de sonido**, pero el sonido no llega a la salida debido a la circuitería controlada por el relé instalado en el equipo (imagen 4.9), el cual está soldado al puerto paralelo de la placa base. Este relé hace de conmutador entre las dos tarjetas de sonido y la salida de audio, dando camino a la salida a sola una de las tarjetas a la vez.

El ruido rosa se toma de un fichero .wav almacenado en el sistema, cuya ruta absoluta dentro del mismo puede observarse en el código 4.9

La clase AtenuadorPGA controla el puerto paralelo apoyándose en la clase PuertoParalelo y está conectado físicamente al relé y al **PGA**, cada uno en un pin distinto. **Para activar este conmutador se usa el pin número 4** del puerto paralelo.

Por otro lado, la creación y destrucción del fichero pink viene dada por la aplicación web, bien mediante los botones de la interfaz de usuarios vistos en la imagen 3.21 o bien mediante el proceso en segundo plano que detecta el inicio y el fin de las sesiones. En la sección 4.4.4 se detalla este proceso.

```

1 #!/usr/bin/php
2 <?php
3 while (true){
4     system("alsaplayer -d hw:1,0 /var/sl/r/pink.wav");
5 };

```

Listado 4.9 – *Script de emisión continuo de ruido rosa.*

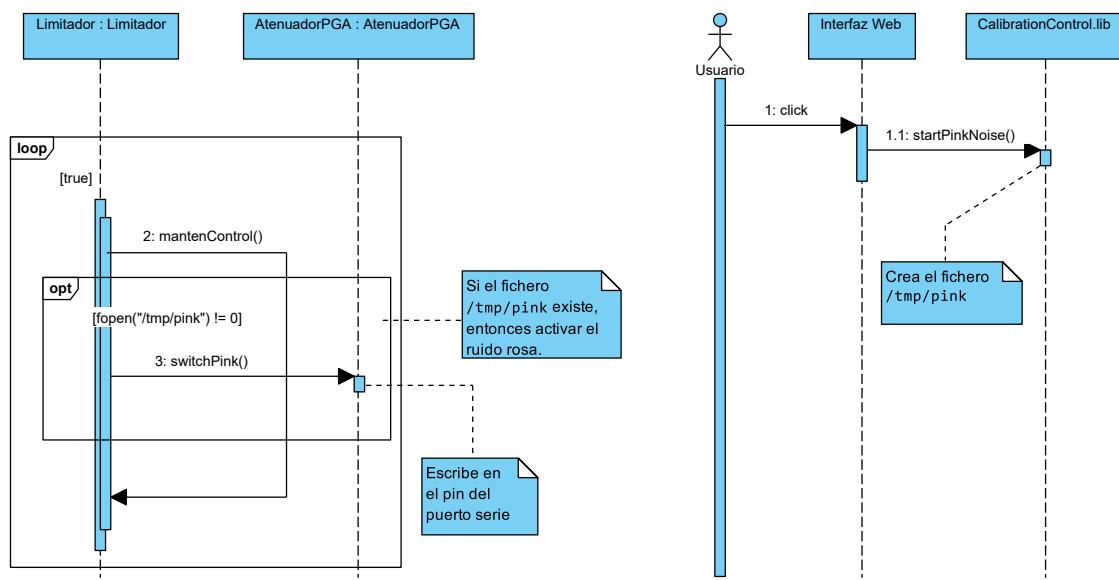


Figura 4.8 – Diagrama de secuencia: emisión de ruido rosa en LM7

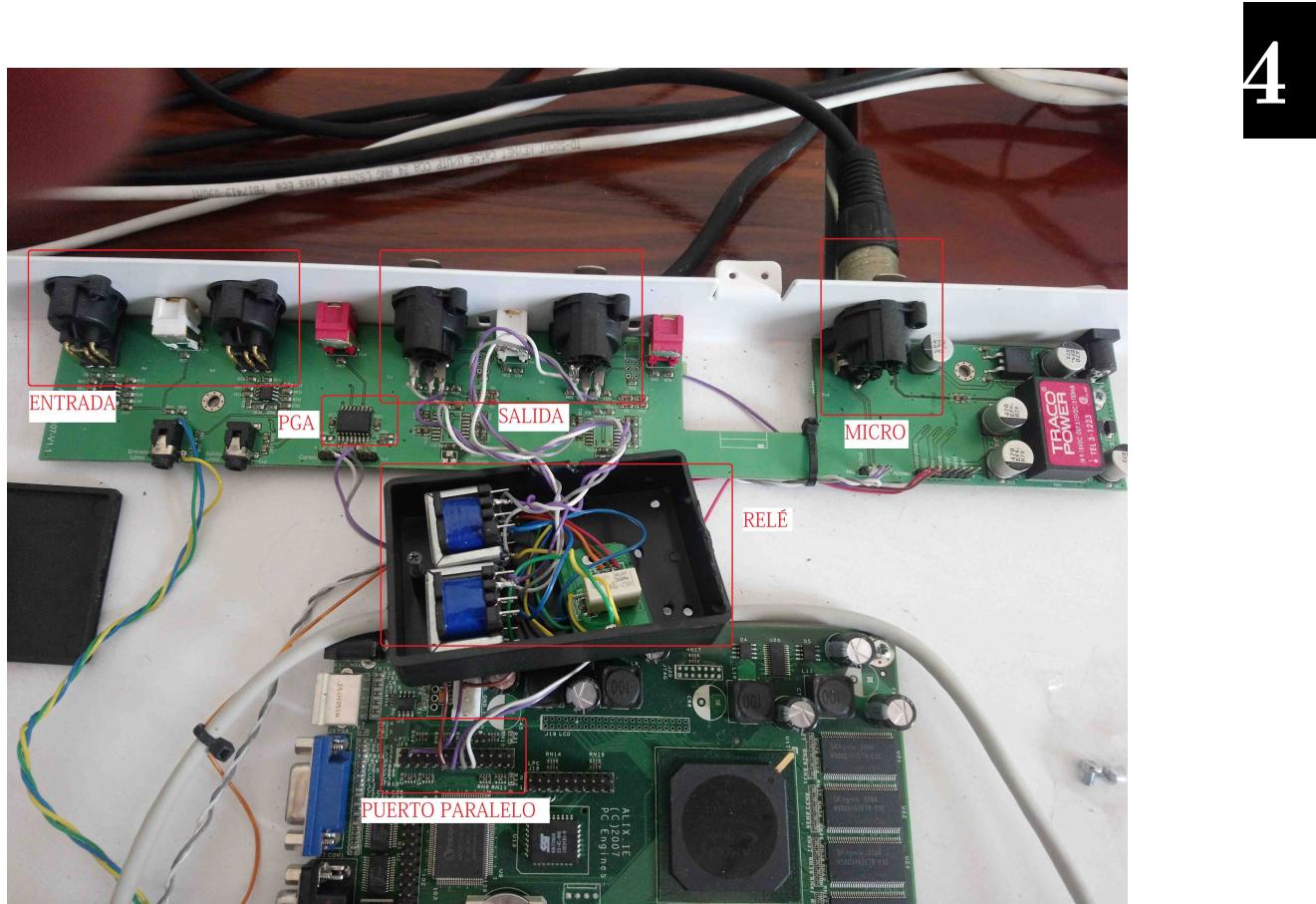


Figura 4.9 – Relé y otros componentes del LM7.

4.4.4 Verificación: sesiones

Para mantener garantizar el buen funcionamiento del sistema y evitar manipulaciones y fraudes, el equipo comprueba sus calibraciones cada vez que detecta una nueva sesión de ruido. Para ello se ejecuta de forma continua un script PHP que detecta cuando los valores de micrófono y línea superan un cierto umbral (por defecto 65 dBs).

```

1 function main()
2 {
3     $ultimosValores = array();
4
5     echo ("Registrando actividad del equipo\n");
6
7     while (true) {
8         sleep(1);
9         $estado = getStatus();
10        if ($estado == null) continue;
11        $ultimosValores[] = $estado;
12
13        echo ("Número de registros " . count($ultimosValores) . "\n");
14
15        $config = GetSessionAndNoiseConfig();
16        $sst = isset($config->sessionStartThreshold) ? $config->
17            sessionStartThreshold : 65;
18        $set = isset($config->sessionEndThreshold) ? $config->
19            sessionEndThreshold : 65;
20
21        echo ("Rangos ($sst, $set)\n");
22
23        if (count($ultimosValores) >= 5 * 60) { // La media de 10 minutos
24            $medias = getAveragesFromStatus($ultimosValores);
25
26            for ($i = 0; $i < 10; $i++)
27                array_shift($ultimosValores); // Quitamos 10 segundos de
28                lecturas
29
30            if (isInSession() && isUnderThresshold($medias, $estado, $set)) {
31                // El equipo está en sesión y se ha bajado de los niveles
32                setNoSession();
33            };
34            if (!isInSession() && isOverThresshold($medias, $estado, $sst)) {
35                // El equipo no está en sesión y se han superado los niveles.
36                setInSession();
37                $results = calibrationTest();
38                addCalibrationResultsLog($results);
39            };
40        };
41    };
42}

```

Listado 4.10 – Script de control de sesiones.

4.4.5 Almacenamiento de calibraciones

Las calibraciones se almacenan en formato binario en los ficheros mencionados en el listado que continúa a este párrafo. La clase Calibración es la única capaz de procesar estos ficheros ya que la escritura y la lectura de los datos consiste simplemente en un **volcado de memoria a disco** de la instancia de la clase Calibración, como se puede ver en el código 4.11. Esta clase contiene el modelo y funcionalidad requerida para exportar e importar los datos.

- /var/slṛ/calibration0: micrófono.
- /var/slṛ/calibration1: línea izquierda.
- /var/slṛ/calibration2: línea derecha.

```

1 int Calibracion::leeCalibracion(int numero)
2 {
3     char nombreF[250];
4     sprintf(nombreF, 240, "%s%d", F_Kal, numero);
5
6     FILE *f = NULL;
7     f = fopen(nombreF, "r");
8
9     if (f != NULL)
10    {
11        fread(this, sizeof(Calibracion), 1, f);
12        fclose(f);
13    }
14
15    return 1;
16 }
```

4

Listado 4.11 – Lectura de calibración desde fichero.

4.5 Gestión de configuración

4.5.1 Proceso de configuración

La configuración del equipo se muestra y se modifica mediante la aplicación web del limitador. Para poder modificarla, el usuario debe tener unas credenciales que le autoricen para ello. Tras guardar la configuración, la interfaz exporta la configuración que necesita actualizarse al fichero conf.tmp y luego notifica al configurador para que recoja y aplique la nueva configuración. En las versiones LM7, el configurador forma parte del programa utilslr.

Pueden consultarse los anexos D.2.1 y C.2.3 para ampliar la información sobre el fichero conf.tmp y el programa utilslr, respectivamente.

4.5.2 Almacenamiento de la configuración

La configuración en vigor es almacenada en formato binario en el fichero `/var/sl7/configuracion`. Antes de su almacenamiento, los datos de la configuración se cifran, y se descifran tras su lectura. La clase Configuración es la única capaz de procesar este fichero, además de contener el modelo almacenado. La escritura y la lectura binaria de los datos consiste simplemente en un **volcado de memoria a disco** por parte de la instancia existente en memoria de la clase Configuración.

4.6 Gestión de registros

4.6.1 Proceso de registro

El proceso de generación y almacenamiento de registro está integrado en el bucle del programa limitador. En cada iteración de este bucle se obtienen los valores de emisión y recepción presión acústica, se calcula un valor de atenuación si es necesario. Estos valores se van sumando a acumuladores que mantiene el programa, y cada minuto se genera y almacena un nuevo registro con las medias de los valores acumulados, junto a otros valores como la hora del sistema o si el micrófono se encuentra conectado o no.

```

1   if (time(NULL) - horaUltimoRegistro >= 60)
2   {
3       horaUltimoRegistro = time(NULL);
4
5       estado.mediaDeUnMinuto = media1m / (float)ctr;
6       estado.presionIzquierda = left1m / (float)ctrLeft;
7       estado.presionDerecha = right1m / (float)ctrRight;
8
9       media1m = left1m = right1m = 0;
10      ctr = ctrLeft = ctrRight = 0;
11
12      registro.crea(estado, configuracion);
13      estado.limpiaMascara();
14      if (!regrador.guardaRegistro(registro))
15      {
16          puts("Error al guardar registro");
17      };
18      registro.muestra();
19 }
```

Listado 4.12 – Generación de registros en el LM7.

4.6.2 Almacenamiento de registros

Tal y como se comenta en el anexo D, el fichero en el que se almacenan los registros es en realidad un fichero especial de dispositivos de bloque, creado durante la instalación del software mediante la utilidad `mknod`, y quedando mapeado a una

partición del disco duro. Esto significa que aunque aparentemente se está realizando una escritura sobre un fichero regular, en realidad se está escribiendo de forma directa en sectores del disco duro. En parte, esto es gracias a la diseño del sistema operativo *Linux* (y *Unix*) donde todo es un fichero. En el código 4.13 se incluye la orden utilizada para crear el fichero de registro.

```
1 | $ mknod registro.slr b 8 4
```

Listado 4.13 – Creación del fichero de registro como un fichero especial de bloques.

La utilidad `mknod` permite especificar el nombre del fichero a crear y los números principal (*major*) y secundario (*minor*). Mientras que el número principal determina el controlador al que está conectado el dispositivo, el número secundario determina el dispositivo en sí.

Según la documentación oficial de *Linux* [26], los archivos de bloque con número principal 8 corresponden a dispositivos de disco **SCSI**, y el número secundario a la partición del disco duro (o al disco completo).

```
1 | 8 block SCSI disk devices (0-15)
2 |   0 = /dev/sda      First SCSI disk whole disk
3 |   16 = /dev/sdb     Second SCSI disk whole disk
4 |   32 = /dev/sdc     Third SCSI disk whole disk
5 |   ...
6 |   240 = /dev/sdp    Sixteenth SCSI disk whole disk
7 |
8 |   Partitions are handled in the same way as for IDE
9 |   disks (see major number 3) except that the limit on
10|    partitions is 15.
```

Listado 4.14 – Descripción de los ficheros con *major number* 8.

Para comprobar que efectivamente el fichero de registro es un enlace duro a una partición de disco, en primer lugar comparamos sus detalles con las del la partición de disco `/dev/sda4`. Hemos identificado esta partición como resultado del número mayor y menor vistos en el código 4.13. Tras comparar sus detalles con la utilidad `ls -li`, comprobamos que ambos ficheros tienen el mismo números principal y secundario, 8 y 4. Para no dejar lugar a dudas, se utiliza la herramienta `blockdev`, para generar un reporte más detallado. Podemos ver el resultado en la imagen 4.10.

En la versión **LM7**, cada uno de los registros ocupa un tamaño de 17 Bytes. El fichero de registro (o la partición `/dev/sda4`, ya que son lo mismo) tiene un tamaño de 203 MB. Dado que se genera y almacena un registro cada minuto, el limitador **LM7** tiene capacidad para almacenar 25 años de registros. En el manual de usuario se indica que puede almacenar registros por 30 años, aunque como se ha visto parece que esto no es así, pero se acerca bastante.

```
root@Limitador:~# blockdev --report /dev/sda4
R0 RA SSZ BSZ StartSec Size Device
rw 256 512 4096 1564672 212860928 /dev/sda4
root@Limitador:~# blockdev --report /var/slrl/registro.slr
R0 RA SSZ BSZ StartSec Size Device
rw 256 512 4096 1564672 212860928 /var/slrl/registro.slr
```

Figura 4.10 – Detalles de los ficheros registro.slr y /dev/sda4.

4.6.3 Lectura de registros

La lectura de los registros corre a cargo del programa auxiliar `getData`. El anexo C.1.3 está dedicado exclusivamente al estudio de este programa. En el anexo se trata la versión utilizada en el nuevo limitador, denominado [LM11](#), pero su funcionalidad e interfaz es idéntica a la del [LM7](#).

4.7 Proceso de atenuación

4

El proceso de atenuación de sonido consiste esencialmente en la reducción de los niveles de emisión por parte del sistema de sonido de la sala en aquellos momentos en los que se requiera, esto es, cuando los niveles superen un cierto umbral previamente configurado. Este umbral forma parte de la normativa y viene regida por el ayuntamiento de la localidad en la que se encuentre el local. De forma general, la normativa depende de:

1. El lugar en el que se encuentre el local (zona residencial, zona industrial...).
2. El tipo de día (laboral, fin de semana, festivo).
3. Si es de noche o de día.

La atenuación se realiza mediante un [PGA](#) (Programmable Gain Amplifier), un amplificador de ganancia que actúa sobre la señal digital de audio como si de un regulador de volumen se tratase. En equipo tiene instalado un [PGA2310UA](#), del fabricante *Texas Instruments*.

En la tabla 4.1 se incluyen las especificaciones de este modelo, mientras que en la imagen 4.9 podemos verlo instalado como parte del equipo del [LM7](#).

El limitador dispone de **3 tipos de control de atenuación**: micrófono, línea y mixto. Esto significa que para decidir si es necesario actuar se toma como referencia el valor de recepción (lo medido por el micrófono), el valor de emisión (de las líneas calculado mediante la fórmula B.0.1) o la media de ambas, respectivamente. En el anexo de [Terminología](#) puede consultar las definiciones usadas.

Number of channels (#)	2
Power supply (V)	+/-15
Gain & attenuation (dB)	+31.5 to -95.5 with 0.5dB steps
Interface	SPI
Interchannel crosstalk @ 1 kHz (dBFS)	-126
Gain error (gain=31.5dB) (dB)	± 0.05
Dynamic range (dB)	120
Voltage swing with max supply (Vpp)	27
Load capacitance (pF)	1000
Operating temperature range (C)	-40 to 85

Tabla 4.1 – Especificaciones del PGA2310UA.

4

Independientemente del tipo de control, cuando el limitador detecta que se están superando los niveles permitidos por la normativa, calcula y aplica una atenuación mediante la actuación del [PGA](#). En cada ciclo de atenuación se aplica una ganancia de ± 1 , dependiendo de si se quiere aumentar o reducir el nivel de atenuación. **La ganancia se aplica en pasos de 0.5dB**, tal y como se ve en la tabla [4.1](#).

4.7.1 Comunicación con el PGA

La comunicación con el [PGA](#) se hace mediante la escritura en ciertos pines del puerto paralelo, al que el PGA se encuentra físicamente conectado. A nivel de software, la comunicación se realiza mediante el uso de los métodos de la clase AtenuadorPGA y PuertoParalalo. Los pines sobre los que se actúan son:

```

1 #define PGA_CS 7
2 #define PGA_CLK 6
3 #define PGA_SDI 5
4
5 #define PGA_Wait 250

```

Listado 4.15 – Pines del puerto paralelo conectados al PGA

Capítulo 5

Ingeniería Inversa: la versión LM9

La versión LM9 representa una actualización de la versión LM7, por lo que la mayoría de sus componentes ya son conocidos. Sin embargo, hay algunos cambios significativos que deben analizarse con profundidad, y que se describen a continuación.

1. La principal diferencia con respecto a la versión anterior es que en este versión el software corre sobre una **arquitectura hardware distinta**. Ya no se utiliza una placa base de un PC industrial, sino que se corre sobre una **plataforma compuesta por una Raspberry Pi y un Arduino**. Mientras que la Raspberry Pi mantiene el sistema principal y el software, el Arduino controla los componentes hardware como el PGA, los LEDs y el LCD bajo las órdenes que recibe de la Raspberry.
2. En el ámbito de análisis de sonido digital, se ha reemplazado la interfaz **Open Sound System (OSS)** por **Advanced Linux Sound Architecture (ALSA)**. Para ello, todo el código relativo la captura de sonido desde las tarjetas de sonido y su interpretación ha tenido que re-implementarse, por lo que esta nueva versión provee una serie de clases y programas nuevos que debe estudiarse.
3. Se ha ampliado al análisis espectral de sonido desde las 8 bandas del LM7 hasta las 31 de esta versión.
4. Aunque en esta nueva versión se sigue haciendo un uso intensivo de ficheros, todos aquellos ficheros que hacían el papel de variables globales han sido reemplazadas por variables globales reales en memoria compartida. Para ello se ha creado un nuevo módulo al que han llamado *SharedMemory*. Este módulo tiene como finalidad crear un segmento de memoria compartida de forma que otros programas puedan hacer uso de él.

5. El LM9 dispone de dos micrófonos en lugar de uno.
6. La estructura de directorios y ficheros ha cambiado ligeramente, tal y como se puede ver en el listado 5.1

```
file:///lms.9/lms.no
|-- HwPort/
|-- Makefile
|-- analyzer/
|-- boanergesBridge/
|-- comun/
|-- configuracion
|-- construye
|-- dataInterfaces/
|-- limitador/
|-- preparatoria
|-- scripts/
|-- tests/
`-- utiles/
```

Figura 5.1 – Estructura de ficheros y directorios principal del LM9.

5

A diferencia de la versión anterior, **en esta versión solo disponemos de parte del código fuente** del limitador y de su hoja de especificaciones técnicas, y no del limitador físicamente. La conclusión de que el LM9 corre de una arquitectura nueva compuesta por una Raspberry Pi y un Arduino es resultado, nuevamente, del proceso de ingeniería inversa al que se ha sometido al código fuente del que se dispone. En él se ha podido encontrar algunos comentarios breves de los desarrolladores donde nombran al Arduino. Por parte de la Raspberry, se debe al uso de módulos Kernel en el script de arranque que tan solo existen o tienen sentido en una Raspberry.

Para detallar algo más, el código del que se dispone corresponde al que corre en la Raspberry. En este código no se incluye el de la aplicación web, como era el caso en el LM7, aunque se sabe con seguridad que hace uso de ella (así se indica en las especificaciones) y que ha sido actualizada, ya que se disponen de algunos scripts PHP que incluyen a otros desde el directorio raíz de Apache (/var/www), los cuáles no existen en la versión del LM7.

5.1 Comunicación entre Raspberry y Arduino

La comunicación entre estos dos sistemas se realiza mediante un **Universal Asynchronous Receiver-Transmitter (UART)** con la ayuda de las librerías termios y fcntl. Conectados mediante el puerto serie, usan de interfaz de comunicación el fichero /var/slra/hardware/port a una velocidad de transmisión (*baud rate*) de 38400

bps. Para controlar el dispositivo, se utiliza la clase `SerialPort`, la cual parece haber sido adaptada de un recurso online¹, sin demasiada modificación. Se puede ver al cabecera de esta clase en el código 5.1

Mediante esta interfaz de comunicación el la Raspberry y el Arduino intercambian información y cooperan para mantener el funcionamiento del limitador. El Arduino, encargado de controlar los dispositivos hardware de más bajo nivel, parece funcionar simplemente como un orquestador; es el software que corre sobre la Raspberry quien realiza la toma de decisiones y se las comunica al Arduino mediante el `UART` para que las lleve a cabo. La comunicación es bidireccional, ya que el software también lee datos del Arduino para actualizar su estado interno.

```

1 #ifndef HH_SerialPort
2 #define HH_SerialPort
3
4 #include <stdio.h>
5 #include <unistd.h> //Used for UART
6 #include <fcntl.h> //Used for UART
7 #include <termios.h> //Used for UART
8
9 #define SerialPortError_NotOpen 1024
10 #define SerialPort_MaxDefaultReadBufferSize 1024
11
12 class SerialPort
13 {
14 private:
15     int uartFd;
16
17 public:
18     SerialPort();
19
20     int Open(const char *portName, int baudRate);
21
22     int Write(const char *string, int len = -1);
23
24     int WriteLn(const char *string);
25
26     int ReadString(char *output, int maxLength =
27                     SerialPort_MaxDefaultReadBufferSize);
28 };
29 #endif

```

Listado 5.1 – Clase `SerialPort`

5.2 Arranque

El arranque es bastante parecido al del LM7, aunque ya no se hace uso del servicio `init.d`. Obviamente el script de arranque se ha tenido que adaptar para que pueda funcionar sobre una Raspberry, pero en general realiza las mismas tareas que el script de arranque de su predecesor. En este caso, el script de arranque lo podemos encontrar

¹<https://tldp.org/HOWTO/Serial-Programming-HOWTO/x115.html>

como parte del código fuente del software, dentro de la carpeta scripts/ y se llama StartUp.

De entre todos los comandos que ejecuta el script de arranque (5.3) hay uno que llama especialmente la atención por la falta de consistencia (como viene siendo normal en el software de los LM), y es que se hace uso varias veces de las órdenes modprobe y rmmod, las cuales sirven respectivamente para cargar y descargar (en el sentido de “eliminar”) módulos del kernel de Linux. Todos los módulos siguen por estándar una nomenclatura, letras minúsculas y palabras separadas por guiones, sin embargo en el script de arranque puede fácilmente detectarse que el módulo SoundCapturer no sigue este estándar. Esto lleva a pensar que este módulo de **kernel** puede ser de implementación propia y específica para el LM9, quizás como driver para una tarjeta de sonido de fabricación personalizada, pero como no se dispone del limitador físico, no hay manera de confirmarlo y no puede dejar de ser una suposición.

```

1 #!/bin/bash
2
3 ifconfig eth0 192.168.1.223 netmask 255.255.255.0
4 ifconfig eth1 192.168.1.223 netmask 255.255.255.0
5 route add -net default gw 192.168.1.1
6
7 echo "" >/tmp/mtab
8 mount /dev/mmcblk0p7 /var/slrv
9 echo "Iniciando..." 2>/dev/null
10
11 PATH=/bin:/sbin:/usr/sbin:/usr/bin 2>/dev/null
12
13 modprobe snd-pcm-oss
14 modprobe snd-mixer-oss
15 modprobe SoundCapturer
16
17 # replace the ramdisk with the tmpfs for /var and /tmp
18 mkdir -p /tmp/log/nginx
19 mkdir -p /var/run/ssh
20 mkdir -p /tmp/run
21 mkdir -m 777 /tmp/tmp /tmp/php4 /tmp/php5 2>/dev/null
22 mount -o bind /tmp/php4 /var/lib/php4 2>/dev/null
23 mount -o bind /tmp/php5 /var/lib/php5 2>/dev/null
24 mkdir /tmp/channels
25
26 for interfaz in eth usb; do
27     for i in $(seq 0 9); do
28         ifconfig ${interfaz}${i}:2 192.168.1.223 netmask 255.255.255.0
29     done
30 done
31 route add -net default gw 192.168.1.1
32 /sbin/ifconfig lo 127.0.0.1 up 2>/dev/null
33 /sbin/route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 dev lo 2>/dev/null
34
35 cd /lms/
36 ./preparatoria
37 cd /
38
39 /usr/local/bin/ntpdate &
40 chmod 777 /var/slrv/configuracion
41 chmod 777 /var/slrv

```

```

42 chmod 777 /dev/*
43 chmod -R 777 /tmp
44 echo "" >/tmp/conf.tmp
45 chmod 777 /tmp/conf.tmp
46 echo "" >/tmp/cuVerified
47
48 # Modules
49 #rmmmod snd_bcm2835 snd_soc_pcm512x_i2c snd_soc_tas5713 snd_soc_pcm512x
50     snd_soc_pcm512x_i2c \acrshort{LED}s_gpio \acrshort{LED}_class
50 rmmmod snd-usb-audio
51 rmmmod SoundCapturer
52 modprobe SoundCapturer
53 modprobe snd-usb-audio
54
55 # Modulos a guitar
56 rmmmod \acrshort{LED}s_gpio
57
58 # RTC Setup
59 modprobe i2c-dev
60 #modprobe rtc-pcf8563
61 modprobe rtc-ds1307
62 echo ds1307 0x68 >/sys/class/i2c-adapter/i2c-1/new_device
63 #echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
64 #echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-1/new_device
65 hwclock -s
66
67 mkdir /tmp/sessions
68 mkdir /tmp/modules
69 chmod 777 /tmp/sessions /tmp/modules
70 mkdir -p /var/log/nginx/
71
72 service php5-fpm start &
73 service udev stop
74 service udev start &
75
76 # Register memory
77 chmod a+r /dev/mmcblk0p4
78
79 mkdir /var/run/sshd
80 service ssh restart &
81
82 /bin/runVersionCommands

```

5

Listado 5.2 – Script StartUp.

```

1 #!/bin/bash
2 #Adding a StartUp Log
3 fecha=$(date +%Y/%m/%d-%H:%M:%S)
4 echo "type=startUp&dni=noUser&name=noUser&time=$fecha&other=" >>/var/slrlogs.
    serial
5
6 echo -n "Hardware Service"
7 runHwController >/dev/null &
8 sleep 2
9 hardwareClient -input 1
10 sleep 1
11 hwUcWD >/dev/null &
12
13 echo "...firstConfigs..."
14 LoadAudioDefaultPreset >/dev/null 2>/dev/null &
15 echo " Ok"
16

```

```

17 #Limiter services
18 echo -n "Limiter/Register services"
19 if getConfig | grep soundLevelLimiter; then
20     echo " Limiter:"
21     echo "    HID type: \acrshort{LCD} and \acrshort{LED}s"
22     #llControl >/dev/null 2>/dev/null &
23     \acrshort{LCD}Control &
24
25     echo "Calibration control Service : "
26     controlDeCalibracion >/dev/null 2>/dev/null &
27
28     echo "Audio stream Services"
29     runLines >/dev/null 2>/dev/null &
30     runMicAnalyzer >/dev/null 2>/dev/null &
31
32     echo "Register & Limiter"
33     keepRg >/dev/null 2>/dev/null &
34     keepLm >/dev/null 2>/dev/null &
35 else
36     echo " Limiter:"
37     echo "    HID type: \acrshort{LCD} and \acrshort{LED}s"
38     #panelControl >/dev/null 2>/dev/null &
39     \acrshort{LCD}Control &
40
41     echo "Audio stream Services"
42     runMicAnalyzer >/dev/null 2>/dev/null &
43
44     echo "Register & Limiter"
45     keepRg >/dev/null 2>/dev/null &
46 fi
47 echo " Ok"
48
49 echo -n "Audio stream keepers"
50 refreshAnalyzers >/dev/null 2>/dev/null &
51 Stablyzer >/dev/null 2>/dev/null &
52 sleep 2
53 killall analyzer
54 echo " Ok"
55
56 echo -n "Network Services"
57 keepWebServer &
58 /var/slr/network.script >/dev/null 2>/dev/null
59 keepKnownIp >/dev/null 2>/dev/null &
60 keepComm >/dev/null 2>/dev/null &
61 # /bin/adjtime >/dev/null 2>/dev/null &
62 in.telnetd -debug 1035 -L /bin/localservice &
63 remoteShellService >/dev/null &
64 echo "Ok"
65
66 echo -n "WatchDog services"
67 controlUptime &

```

Listado 5.3 – Script runVersionCommands.

5.3 Detección de micrófono

El micrófono, como el resto de componentes hardware, está controlado por el Arduino, por lo que no se ha podido identificar el método de detección utilizado. El software recibe esta información mediante una interfaz de comunicación descrita en la

sección 5.1. Para detectar si el micrófono está conectado o no se lee desde el dispositivo como si fuese un fichero, en el que se recibe un bloque de datos en formato JSON con el esquema 5.4. Esta información se carga en la memoria compartida, quedando así disponible para el resto de programas del limitador.

```

1  {
2      "Status": {
3          "type": "object",
4          "properties": {
5              "version": {
6                  "type": "string"
7              },
8              "ChannelPressure": {
9                  "type": "array",
10                 "minItems": 4,
11                 "maxItems": 4,
12                 "items": {
13                     "type": "number"
14                 }
15             },
16             "MicConnection": {
17                 "type": "boolean"
18             },
19             "InputPressure": {
20                 "type": "array",
21                 "minItems": 2,
22                 "maxItems": 2,
23                 "items": {
24                     "type": "number"
25                 }
26             },
27             "attenuation": {
28                 "type": "array",
29                 "minItems": 3,
30                 "maxItems": 3,
31                 "items": {
32                     "type": "number"
33                 }
34             }
35         }
36     }
37 }
```

Listado 5.4 – Lectura del estado del micrófono en el LM9.

5.4 Procesamiento de audio

Este módulo es con diferencia el gran cambio de esta nueva versión. Al pasar desde OSS a ALSA todo lo relacionado con el análisis del audio ha tenido que ser re-diseñado y re-escrito: La idea general sigue siendo la misma, se disponen de 2 tarjetas de sonido a las cuales nos conectamos mediante la API de la interfaz de sonido, ALSA en este caso, para capturar y procesar el audio que por ellas transcurren. Mientras que una de las tarjetas nos proporciona la música procedente de la tabla de mezclas, la otra nos proporciona el audio procedente del micrófono instalado en la sala. De este modo

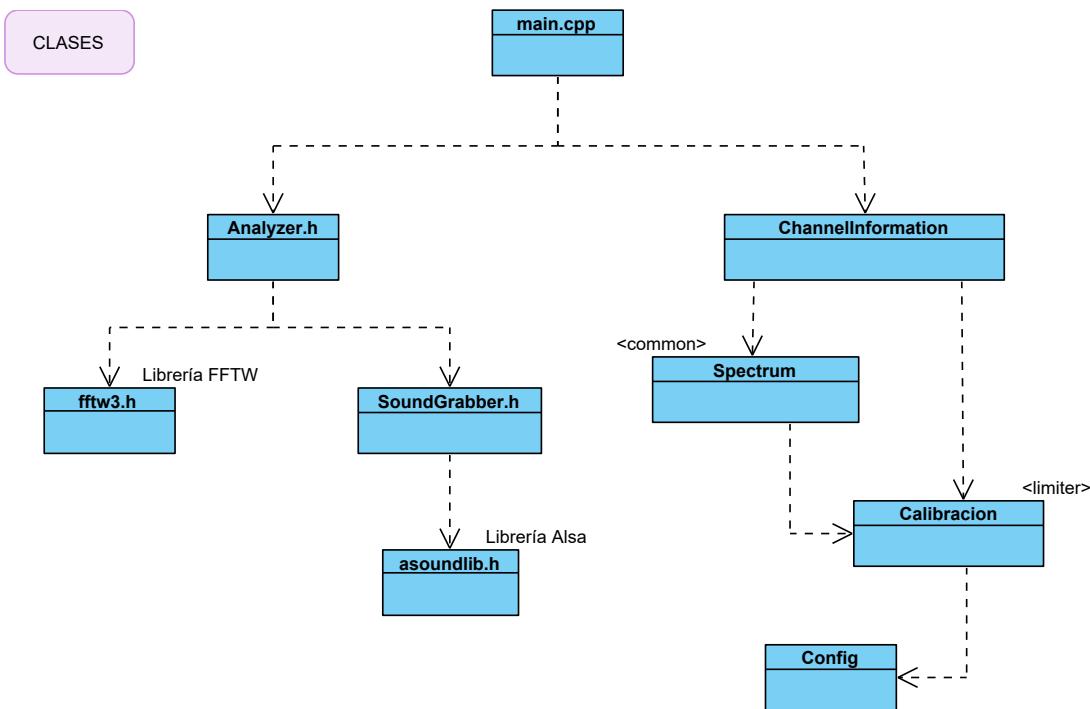


Figura 5.2 – Diagrama de dependencias del módulo Analyzer en el LM9.

tenemos las materias primas para poder generar los modelos necesarios y poder controlar tanto los niveles de emisión como los niveles de recepción.

5

El nuevo diseño se traduce en el desacople de funcionalidades desde una sola clase, como era el caso en el LM7, a varias en esta nueva versión. De este modo, se obtiene una estructura de clases ligeramente más compleja, pero más fácil de comprender, ya que **cada clase tiene una única responsabilidad**, lo cual es buen principio de diseño (*Principio de Responsabilidad Única*).

5.4.1 Conexión a tarjetas de sonido

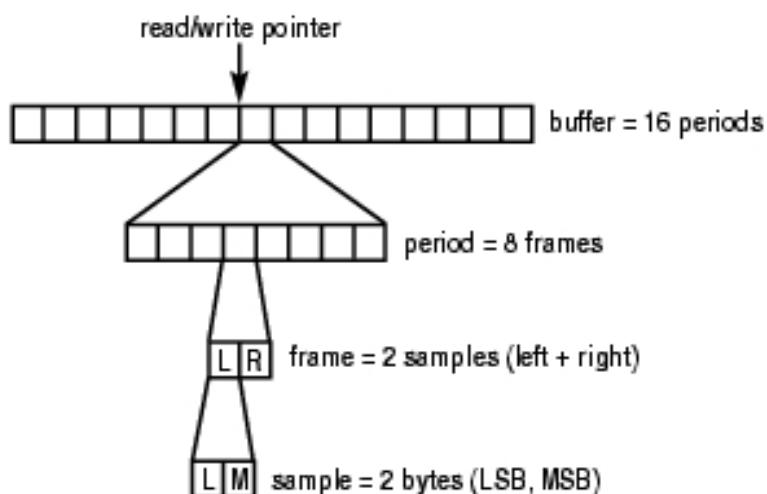
La conexión a las tarjetas de sonido se realiza mediante la clase SoundGrabber. Comparando con el código de la versión LM7, esta clase se correspondería a la clase Lector, salvo por el hecho de que SoundGrabber no realiza ningún tipo de análisis ni transformaciones sobre los datos, tan sólo se conecta a los dispositivos de audio de **ALSA** y lee el flujo de datos. Tal y como se puede observar en el diagrama 5.2, SoundGrabber por un lado, consume datos del flujo de audio mediante la librería de **ALSA** (asoundlib.h), y por otro, proporciona datos a la clase Analyzer para que ésta los procese.

La nomenclatura de los dispositivos en **ALSA** varía con respecto a OSS. La conexión a los dispositivos de audio ya no se realiza mediante ficheros, como /dev/dsp, sino que se realiza a través de las interfaces de **ALSA** hacia estos dispositivos, como hw:0,0.

En [ALSA](#) el dispositivo por defecto para captura y reproducción recibe el nombre de default. Los dispositivos de hardware, como las tarjetas de sonido, reciben un nombre que sigue el patrón hw:<card>:<subdevice>, aunque sobre ellos se pueden configurar plugins y mixers. Los artículos [10] y [11] son de lectura obliga para poder comprender un poco la arquitectura de [ALSA](#) y su funcionamiento. También es de consulta obligada, por supuesto, la documentación oficial de [ALSA](#) [3].

La lectura desde datos de nuevo se realiza por bloques de tamaño configurable, en este caso, el tamaño por defecto es 8 192, a una frecuencia de sampleo de 48 kHz. Al abrir el dispositivo de [ALSA](#) se deben configurar los parámetros del flujo antes de realizar ninguna acción de lectura. El formato de los datos se da en modo intercalado (*INTERLEAVED*), de igual manera que en el [LM7](#).

(bytes) LL-RR-LL-RR-LL-RR...



5

Figura 5.3 – Formato del flujo de datos de acrshortALSA en modo intercalado.

5.4.2 Interpretación de datos

El audio capturado debe procesarse de forma que se pueda obtener el modelo de presión acústica necesario para el limitador. Para ello se implementa una serie de **Transformadas de Fourier** mediante el uso de la biblioteca [Fastest Fourier Transform in the West \(FFTW\)](#) [8].

El analizador de sonido obtiene los datos *raw* desde la tarjeta de sonido mediante el uso de la clase SoundGrabber, quien le da los datos separados en canales. A continuación, el analizador ejecuta las transformadas de Fourier sobre dichos datos para obtener su composición por bandas de frecuencia, es decir, su espectro.

A partir de este punto se sigue el enfoque de la versión anterior. se calculan los decibelios de cada una de las bandas del espectro mediante la fórmula [B.0.1](#) y las calibraciones, para luego calcular el valor de presión global del espectro como la media de las presiones en todas las frecuencias.

5.4.3 Almacenamiento de audio

El programa Analyzer almacena el audio ya procesado en ficheros para su posterior uso. Para cada uno de los canales analizados se crean dos ficheros, uno “lento” y otro “rápido”. La diferencia entre ambos es la velocidad con la que se actualizan. Los ficheros *slow* corresponden al espectro de audio promedio del último segundo de lecturas, mientras que los ficheros *fast* se actualizan en cada iteración del programa.



Figura 5.4 – Ficheros de audio del LM9.

5.5 Gestión de calibración

5.5.1 Proceso de calibración

El proceso de calibración se ha modificado ligeramente en esta nueva versión. De nuevo, no se tiene la certeza de cómo es exactamente el proceso de calibración ya que no disponemos del código de la interfaz web para esta versión, origen de este tipo de

acciones, disparadas por el usuario.

El proceso de calibración bien podría ser exactamente igual que en el LM7, ya que el programa localservice sigue existiendo en la misma versión y con las mismas capacidades en lo que al proceso de calibrado se refiere. Por otra parte, esta nueva versión proporciona un nuevo ejecutable llamado Calibrador. Tras comparar los algoritmos de calibración de ambos programas podemos afirmar que, aunque con pequeñas diferencias en el código, el algoritmo de calibración implementado es el mismo. Sin embargo, este programa permite calibrar las líneas del limitador, pero no su micrófono, por tanto para este último caso debería seguir utilizándose la funcionalidad que proporciona localservice. Sea como fuere, todo lo descrito en el proceso de calibración del LM7 (4.4.1) puede aplicarse a esta versión.

5.5.2 Ecualizaciones

En el campo de las ecualizaciones se realizaron ligeros cambios:

1. Se ha añadido un nuevo vector por cada sensor (micrófono, línea derecha y línea izquierda), haciendo la distinción entre ecualización y ecualización interna.
2. Todos los vectores son ahora de tamaño 31 (nº de bandas de frecuencias).

La diferencia entre ecualización y ecualización interna viene dada por el ámbito de las mismas. Mientras que las ecualizaciones son definidas y configuradas por el usuario o instalador, las ecualizaciones internas son generadas de forma automática por el limitador durante el proceso de calibración, aunque también se permite su modificación manual.

5

5.5.3 Emisión de ruido rosa

La emisión de ruido rosa es una de las grandes incógnitas de esta versión, ya que no se ha encontrado nada al respecto en el código fuente, ni siquiera en la definición de órdenes para el Arduino. Como esta versión supone una actualización de la anterior, se puede suponer que la emisión de ruido rosa se realiza directamente desde la aplicación web, sin pasar por un proceso C++ del limitador. En la aplicación web del LM7 se invocan órdenes del sistema y procesos de forma directa con `exec()` o `popen()`, y de hecho existen funciones para controlar directamente las tarjetas de sonido y reproducir el fichero de ruido rosa. Sin otra alternativa, se ha de suponer que esto es así en esta versión y que la reproducción de ruido rosa se ha desacoplado de los procesos a bajo nivel del limitador.

Otra alternativa, la cual tendría más sentido teniendo en cuenta la arquitectura

hardware vista en el LM7 (sección 4.4.3), sería que la emisión de ruido rosa estuviera controlada nuevamente por el Arduino. El software del limitador, de forma análoga a como se ha visto en otras secciones, ordenaría al Arduino la emisión de ruido rosa mediante el UART conectado al puerto serie.

Recordemos que en el LM7 una de las tarjetas de sonido reproduce de forma continua ruido rosa, pero solo obtiene un camino físico a la salida de audio del limitador cuando el conmutador (relé) se lo permite. Esto podría también ser así en esta versión, pero lo único que tenemos para poder al menos sospecharlo es la existencia de un comando en la interfaz de comunicación con el Arduino llamado noise.

5.5.4 Verificación: sesiones

Este proceso no ha mostrado ningún cambio en su funcionamiento y sigue llevándose a cabo por el script PHP controlDeCalibracion, sin embargo sí que ha introducido la generación y almacenamiento de un fichero en formato JSON con los datos de la última verificación realizada. Este fichero se almacena en la ruta /tmp/lastCalibrationTest.json.

5.5.5 Almacenamiento de calibraciones

En este aspecto la versión actual no presenta ningún cambio con respecto a la versión LM7. Los ficheros de calibración se siguen almacenando en las mismas rutas y en formato binario.

5

5.6 Gestión de la configuración

El modelo de la configuración ha sufrido algunos cambios, mayormente limpieza de atributos importados desde la versión 7 que ya no son necesarios. A pesar de esto, toda la gestión de la configuración es idéntica a la versión anterior. La configuración se almacena y recupera de la misma manera y en los mismos ficheros que en la versión LM7, por lo que lo descrito en la sección 4.5 para la versión 7 es igualmente aplicable a esta versión.

Entre algunos de los cambios introducidos se encuentra la ampliación del vector de *aislamiento* a 31 elementos, así como la creación de otro vector con nombre *márgenes* con igual tamaño y tipo.

5.6.1 Proceso de configuración

Se ha añadido la capacidad de configurar el nuevo atributo *márgenes*. El proceso de configuración en sí no presenta cambios respecto a la versión anterior. Véase el anexo D.2.1 para más detalles.

5.6.2 Almacenamiento de la configuración

Sin cambios respecto a la versión LM7. Véase la sección 4.5 sobre el almacenamiento de la configuración en el LM7.

5.7 Gestión de registros

5.7.1 Proceso de registro

El proceso de registro ha sido llevado a un programa independiente, el Registrador, por lo que la generación de registros ya no forma parte del programa Limitador. Aún así, ahora ambos programas se ejecutan de forma ininterrumpida en paralelo. Aunque independiente, el algoritmo de registro de datos es idéntico al de la versión anterior, aunque ahora utilizad los datos que le proporciona el módulo Analyzer y hace uso de las nuevas clases, así como de unas clases auxiliares para acumular los datos durante los 60 segundos que representa el registro.

```

1 // Guardar registro
2     if (time(NULL) - horaUltimoRegistro >= 60)
3     {
4         printf("Preparing to save\n");
5
6         horaUltimoRegistro = time(NULL);
7         if (ctr == 0)
8             ctr = 1;
9
10        estadoAGuardar = estado;
11        estado.limpiaMascara();
12
13        //Calculando medias
14        printf(" Calculating averages (%d)\n", ctr);
15        estadoAGuardar.mediaDeUnMinuto = micAccumulator / ctr;
16        estadoAGuardar.presion = micAccumulator / ctr;
17        estadoAGuardar.presionIzquierda = leftAccumulator / ctr;
18        estadoAGuardar.presionDerecha = rightAccumulator / ctr;
19        estadoAGuardar.recepcion = receptionAccumulator / ctr;
20
21        //Creamos la media de las mediciones
22        printf(" Spectrum averages (%d)\n", medicionesAcumulativas.ctr);
23        medicionesAcumulativas.convertToAverage();
24        medicionesAcumulativas.print();
25
26        //Limpieza de valores acumulativos
27        ctr = 0;

```

```

28     micAccumulator = 0;
29     leftAccumulator = 0;
30     rightAccumulator = 0;
31     receptionAccumulator = 0;
32
33     //Creando el registro
34     printf("  Creating registry entry\n");
35     registro.crea(estadoAGuardar, configuracion);
36
37     //mic ,left ,right ;
38     registro.setSpectrums(
39         medicionesAcumulativas.mic.aWeighted ,
40         medicionesAcumulativas.left.aWeighted ,
41         medicionesAcumulativas.right.aWeighted );
42
43     //Guardando registro
44     if (!registrador.guardaRegistro(registro))
45     {
46         puts("Error al guardar registro");
47     };
48     registro.muestra();
49
50     //Preparamos el objeto para recibir más datos
51     medicionesAcumulativas.zero();
52 }
```

Listado 5.5 – Generación de registros en el LM9.

5.7.2 Almacenamiento de registros

5

Los únicos cambios respecto a la versión anterior ha sido la adición de una matriz de datos de tipo flotante de tamaño [3] [31] en la clase Registro, para almacenar el espectro del micrófono y las líneas. El resto no ha presentado cambios. Para ampliar la información diríjase al anexo [D.1](#).

5.7.3 Lectura de registros

Sin cambios respecto a la versión [LM7](#). La lectura de los registros sigue realizándose mediante el uso del programa `getData`. Puede ver el anexo [C.1.3](#) para completar la información.

5.8 Proceso de atenuación

5.8.1 Comunicación con el PGA

El [PGA](#), como el resto de componentes hardware, está controlado por el Arduino, de hecho ni siquiera queda completamente claro que exista en [PGA](#), aunque todo parece indicar que sí. El último paso del proceso de atenuación es comunicar la atenuación deseada al [PGA](#), y en este caso, eso se traduce en comunicárselo al Arduino.

La interfaz de comunicación con el Arduino ha quedado descrita en la sección [5.1](#). El programa limitador escribe en el dispositivo la atenuación deseada tal y como se en el código [5.6](#).

```
1 // El arduino da la atenuación en negativo.  
2 char command[1024];  
3 sprintf(command, "echo max %.1f 10 10 %.1f 5 > /var/slrl/hardware/port", -  
4 desiredAt, 0.0f);  
system(command);
```

Listado 5.6 – Comunicación de la atenuación deseada en el LM9.

Capítulo 6

Análisis del sistema

Con este nuevo capítulo se da por terminado el proceso de ingeniería inversa al que se ha sometido a los limitadores [LM7](#) y [LM9](#). De ellos se extrae una inmensa cantidad de información que nos será útil para diseñar y construir un sistema software que pueda correr bajo el nuevo prototipo de limitador disponible en el laboratorio ed [GranaSAT](#).

El software de los limitadores estudiados, aunque no presenta la mejor calidad, realiza su función, por lo que gran parte de éste será reutilizado en el nuevo sistema. Damos lugar por tanto a un proceso de análisis en el que se evaluará en qué grado los módulos de los [LM7](#) y [LM9](#) satisfacen las necesidades de nuestro sistema, así como las posibles mejores aplicables.

Por lo general, se prefiere el diseño y la implementación de la versión [LM9](#) dado que está diseñado para funcionar sobre un hardware casi idéntico al del prototipo disponible: una Raspberry Pi. Esto desde luego no significa que la versión vaya a ser descartada; es importante recordar que ésta es la versión disponible físicamente en el laboratorio, y de la cuál se tiene certeza sobre su correcto funcionamiento. Además, recordemos que la versión [LM9](#) supone una actualización, y no un software independiente. Por tanto, **siempre que surjan dudas sobre el diseño o implementación de alguno de los módulos se irá sobre seguro y se dará total preferencia a la versión LM7**.

Tras el estudio de ambas versiones se hace evidente el hecho de que, al menos el código, no ha sido realizado de forma profesional, ya que hay problemas graves en el mismo. Algunos de ellos se listan a continuación:

1. No separar la declaración de las clases de su implementación (ficheros .h ó .hpp y .cpp, respectivamente).
2. Inclusión directa de ficheros .cpp.

3. Utilizar funciones y variables globales en lugar de clases.
4. Violar completamente el *Principio de Encapsulamiento*¹ de la Programación Orientada a Objetos. Absolutamente todos los datos miembro de todas las clases tienen accesibilidad pública y son modificados directamente.
5. Falta casi absoluta de un formato legible² y de documentación.
6. Nombre de variables poco descriptivos.
7. Mal uso de la herramienta Makefile.
8. Funcionalidades duplicadas.
9. Excesiva complejidad en algunos programas (“MacGyver”)³.

En la medida de los posible, se intentará solventar estos problemas en aquellos módulos que sean seleccionados para su integración en el proyecto, con el objetivo de dotar al código de un cierto grado **calidad** que facilite su lectura, comprensión y mantenimiento.

6.1 Análisis hardware del prototipo

En el laboratorio se dispone de un prototipo hardware sobre el que deberá ejecutarse el software a desarrollar, por tanto es necesario conocer sus componentes ya que pueden injectar ciertas restricciones técnicas que deben ser tomadas en cuenta durante el proceso de diseño y desarrollo del producto.

6

- En esencia, este equipo se basa en una **PCB** o placa base hecha a medida sobre la que se conectan los distintos componentes.
- Como módulo de computación se tiene una Raspberry Pi modelo 3, sobre la que corre un sistema operativo **DietPi**, basada en una distribución de Linux.
- Para la entrada/salida de los canales de audio se dispone de 6 entradas balanceadas . Estas salidas pueden verse en la imagen [7.1](#). Por el momento el prototipo tan sólo dispone de un micrófono, aunque el objetivo es permitir la conexión de ambos en siguientes versiones del prototipo.

¹En programación orientada a objetos, se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto

²Doy gracias al formateador de Visual Studio Code por preservar mi salud mental en éste sentido.

³Término acuñado por un profesor de la ETSIIT, haciendo referencia a clases que “intentan hacerlo todo.”

- El equipo dispone de un **PGA** para llevar a cabo la atenuación. La diferencia entre por ejemplo, el **LM7**, es que se dispone de un conmutador digital en lugar de un relé para realizar la redirección del audio hacia la salida.
- Como complementos, el prototipo dispone de una pequeña pantalla **LCD** de 4 filas por 20 columnas; de un buzzer, y de una serie de **LEDs** RGB. Es deseable que el software pueda controlar estos componentes para poder mostrar mensajes haciendo uso de ellos.



Figura 6.1 – Uno de los modelos del prototipo de limitador disponible.

6.1.1 Restricciones técnicas impuestas por el hardware

Las especificaciones hardware del limitador impone una serie de restricciones técnicas a nuestro proyecto que debemos tener en cuenta. Las restricciones identificadas se transforman a requisitos no funcionales y se listan a continuación:

1. Una Raspberry utiliza un microprocesador con arquitectura **ARM**, por lo que debe tenerse a la hora de compilar el código C++.
2. En este equipo disponemos de una única tarjeta de sonido personalizada, con 8 canales digitales.
3. El interfaz de sonido a usar debe ser **ALSA** (requisito impuesto por el cliente).
4. Está planeado que el nuevo hardware admita dos micrófonos en un futuro, por lo que el software debe estar preparado para ello.

6.2 Evaluación de satisfacción requisitos

A continuación se analizan los requisitos funcionales en base a lo aprendido en el proceso de ingeniería inversa. Para ello se estudia si el software estudiado permite satisfacer parte de los requisitos funcionales del proyecto, para así poder re-utilizarlos.

Requisito	RF 1.- Calibración de sensores
Módulo	LM9-calibrador
Análisis	Esta implementación y la del LM7 se basan en el mismo algoritmo, dónde funciona correctamente.
Evaluación	Válido

Tabla 6.1 – Evaluación de satisfacción del RF-1

Requisito	RF 2.- Emisión de ruido rosa
Módulo	LM7-script
Análisis	No aplicable debido al uso de dos tarjetas de sonido, mientras que nuestro prototipo sólo dispone de una. La idea subyacente, sin embargo, resulta útil al disponer el prototipo de un conmutador digital.
Evaluación	Re-utilizable

Tabla 6.2 – Evaluación de satisfacción del RF-2

Requisito	RF 3.- Verificación
Módulo	LM7-script
Análisis	El script, escrito en PHP, depende de funciones de la interfaz web. Habría que exportar estas dependencias incluirlas en el propio script.
Evaluación	Válido

Tabla 6.3 – Evaluación de satisfacción del RF-3

Requisito	RF 4.- Gestión de registro
Módulo	Ambos-registrador
Análisis	Aunque en el LM9 el registrador se ha extraído a un programa independiente del limitador, el proceso de registro es el mismo. La manera en la que se almacenan los registros no es para nada la mejor, pero es la única que se tiene. Recrear el módulo (mediante por ejemplo una base de datos) supondría un gran riesgo para el proyecto debido al escaso tiempo del que se dispone. También supondría tener que rehacer por completo el módulo getData, ya que quedaría inservible. La mejora del sistema actual se deja como trabajo futuro.
Evaluación	Válido

Tabla 6.4 – Evaluación de satisfacción del RF-4

Requisito	RF 5.- Interfaz de configuración y consulta
Módulo	LM7-Aplicación web
Análisis	La aplicación web está fuertemente acoplada al software del limitador y al sistema operativo. Por otra parte el código es bastante pobre, usa el obsoleto PHP 5 y depende de muchas librerías que a día de hoy también se encuentran obsoletas o directamente han desaparecido. Se ha intentado varias veces portar la interfaz por petición explícita del cliente pero sin éxito. En su lugar, se propone una API HTTP REST .
Evaluación	Inviable

Tabla 6.5 – Evaluación de satisfacción del RF-5

Requisito	RF 6.- Lectura de presión acústica
Módulo	LM9-analizador
Análisis	<p>El analizador de audio del LM9 supone una importante mejora respecto a su versión anterior. Hace uso de ALSA, lo que satisface uno de los requisitos no funcionales. Por otra parte, está diseñado para conectarse a un dispositivo de audio de ALSA, lanzando 2 instancias de este programa, uno para cada tarjeta de sonido. Por una de ellas se obtiene el audio de las líneas y por la otra el audio de los micrófonos. Esto debe revisarse ya que en nuestro caso tan solo hay una tarjeta de sonido. El programa tendría por tanto que recuperar y analizar los datos de los 4 canales de entrada de forma simultánea, y el programa no está diseñado para leer de más de dos canales.</p>
Evaluación	Re-utilizable

Tabla 6.6 – Evaluación de satisfacción del RF-6

6

Requisito	RF 7.- Aplicación de atenuación
Módulo	Ambos-limitador
Análisis	<p>Los algoritmos en ambas versiones son completamente distintos. Se elige el proceso de atenuación del limitador LM7 como candidato. Sin embargo, debería someterse a un profundo proceso de refactoring ya que el resto de componentes software con los que se comunica se han tomado del LM9, como el analizador de sonido o el registrador. La conexión con el nuevo hardware también debe rehacerse (detección de micrófono y comunicación con la PGA).</p>
Evaluación	Re-utilizable

Tabla 6.7 – Evaluación de satisfacción del RF-7

Requisito	RF 8.- Discovery
Módulo	Prototipo
Análisis	El equipo de GranaSAT ya dispone de este mecanismo en versiones anteriores del prototipo. El mecanismo se basa en la emisión broadcast para llegar a todos los equipos en la misma red.
Evaluación	Válido

Tabla 6.8 – Evaluación de satisfacción del RF-8

Capítulo 7

Diseño del sistema

Como este software va a reutilizar gran parte del software de los limitadores [LM7](#) y la [LM9](#), se ha decidido bautizar al nuevo limitador como **LM11**, para poder así referirnos a él de forma breve al mismo tiempo que se mantiene la consistencia en la nomenclatura.

Para la etapa de diseño se tiene en cuenta el estudio y las consideraciones tomadas en los capítulos de Ingeniería Inversa y Análisis, mediante las cuales hemos descubierto las necesidades del sistema que se pretende desarrollar, las características del equipo sobre el que se va a instalar nuestro software y el grado de aceptación del software disponible en base a los requisitos del proyecto.

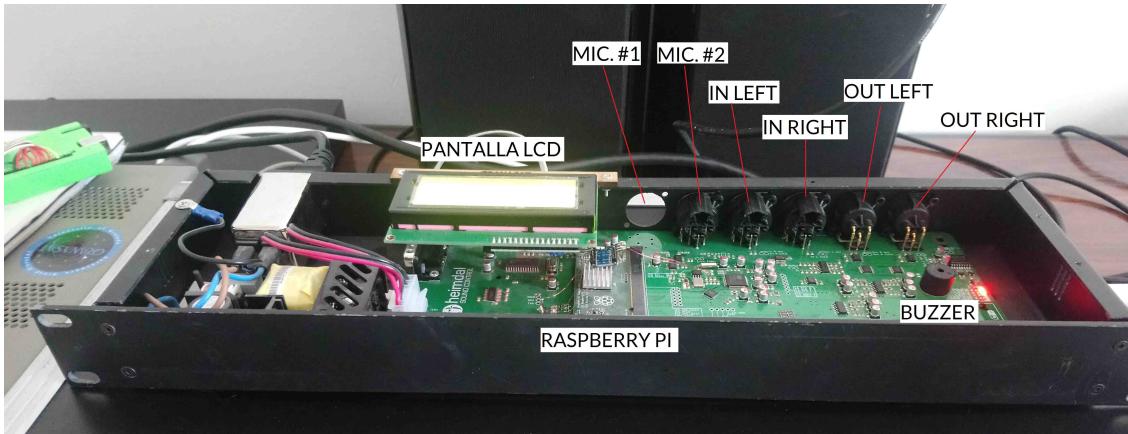


Figura 7.1 – Entradas balanceadas para canales del audio en el [LM11](#).

En la imagen [7.1](#) podemos ver una imagen del limitador y algunos de sus componentes. Se puede observar con facilidad las importantes diferencias en comparación con el limitador que se ha venido estudiando, el cuál podemos ver en la imagen [3.1](#). Las diferencias son notables, ya que nos encontramos ante un hardware

más moderno, compacto y más elaborado, diseñado exclusivamente para éste propósito, mientras que el [LM7](#) utilizaba una placa base de ordenador industrial conectada mediante cables soldados a una pequeña [PCB](#) de fabricación casi casera. El prototipo es bastante más pequeño que el [LM7](#).

A nivel de sistema, podemos visualizar el nuevo limitador mediante el diagrama [7.2](#). En este diagrama podemos observar que los elementos principales a nivel de hardware son tan solo tres: el módulo de computación (Raspberry Pi), el [PGA](#) y el conmutador. Tanto la música como el audio llegan al software de limitación instalado en la Raspberry mediante **la tarjeta de sonido** (no representada en el diagrama por simplicidad). El software analiza el audio y decide si debe actuar sobre los niveles de ruido, en cuyo caso **actúa mediante el PGA**, conectado a la Raspberry a través de [SPI](#). El [PGA](#) recibe a su vez el audio de entrada, pero sólo el de la música, y actúa sobre la señal **aplicando una ganancia**, que puede ser positiva o negativa. Finalmente el audio sale del limitador hacia los amplificadores y/o altavoces **pasando por el conmutador**, siempre y cuando éste lo permita.

El conmutador proporciona un **caminio único a la salida** de audio del limitador, y solo puede proporcionar este camino a la [PGA](#) o a la Raspberry, pero no a ambos a la vez (de la misma forma que se vio en el [LM8 4.9](#)). Normalmente se encontrará en la posición que se ve en el diagrama, de forma que dé salida al audio procesado por el [PGA](#), pero en aquellos momentos en los que se requiera **emitir ruido rosa** para calibrar el sistema o verificar su calibración, se deberá dar salida al audio proveniente de la Raspberry, esto es, el audio que se reproduzca dentro del sistema operativo.

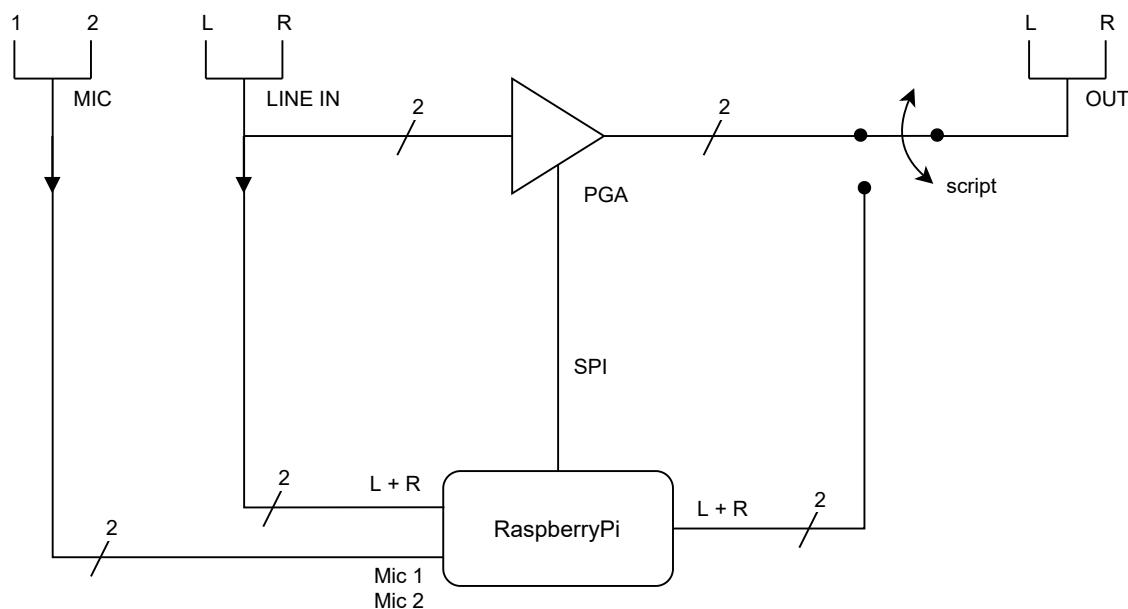


Figura 7.2 – Esquema hardware del limitador [LM11](#).

Toda la información y conocimiento extraídos del proceso de ingeniería inversa se puede resumir en el esquema de la figura 7.3. Principalmente, se requieren los tres elementos que se ven en el diagrama: un analizador de audio que procese las entradas de audio y genere un modelo con el que poder trabajar; un calibrador; y un limitador, que tome las decisiones oportunas en base al modelo generado, la calibración y la configuración (la normativa).

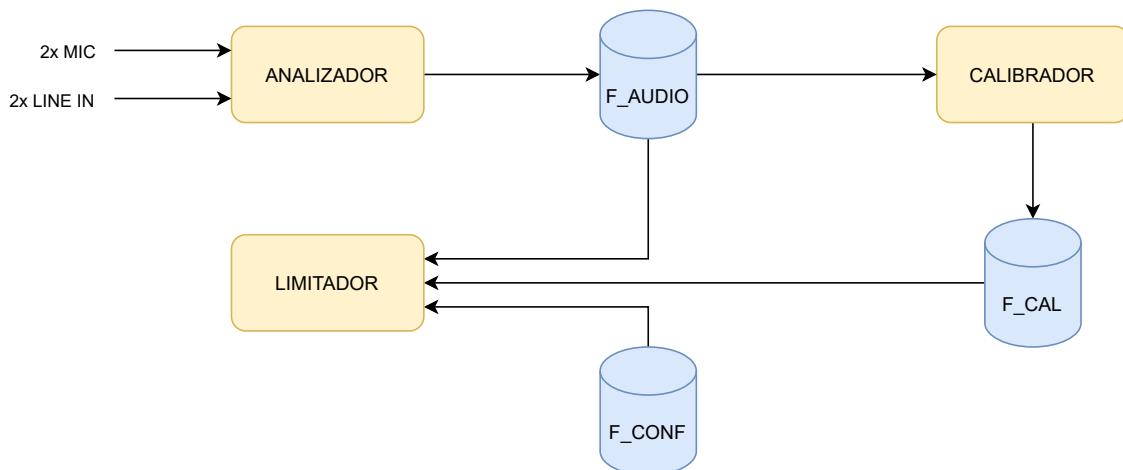


Figura 7.3 – Esquema software del limitador LM11.

Los limitadores estudiados dependían de una aplicación web para su configuración y la consulta de sus datos. Como esta aplicación no se va a importar, debemos buscar otra forma de realizar estas tareas. Como interfaz de comunicación con el exterior del limitador, se propone una [API HTTP REST](#) que proporcione una serie de endpoints mediante los cuales se pueda consultar los datos de limitador y configurarlo. El sistema propuesto se traduce en el diagrama 7.4.

7.1 Arranque

systemd

Al arrancar el sistema se debe lanzar el software y todos los programas que lo componen. Para ello se propone el uso de un script de arranque definido como un servicio de [systemd](#) [15], y por lo tanto, siguiendo la estructura que este gestor exige.

7

La idea es definir un servicio que lance el script de arranque contenido en el directorio del proyecto, y este a su vez levante todos los programas necesarios. Distinguimos entonces entre script de arranque principal ([systemd](#)) y script de arranque secundario (Bash). De esta forma, cuando se reinicie el servicio, todos los programas del limitador se reiniciarán en bloque. Cada uno de los programas a

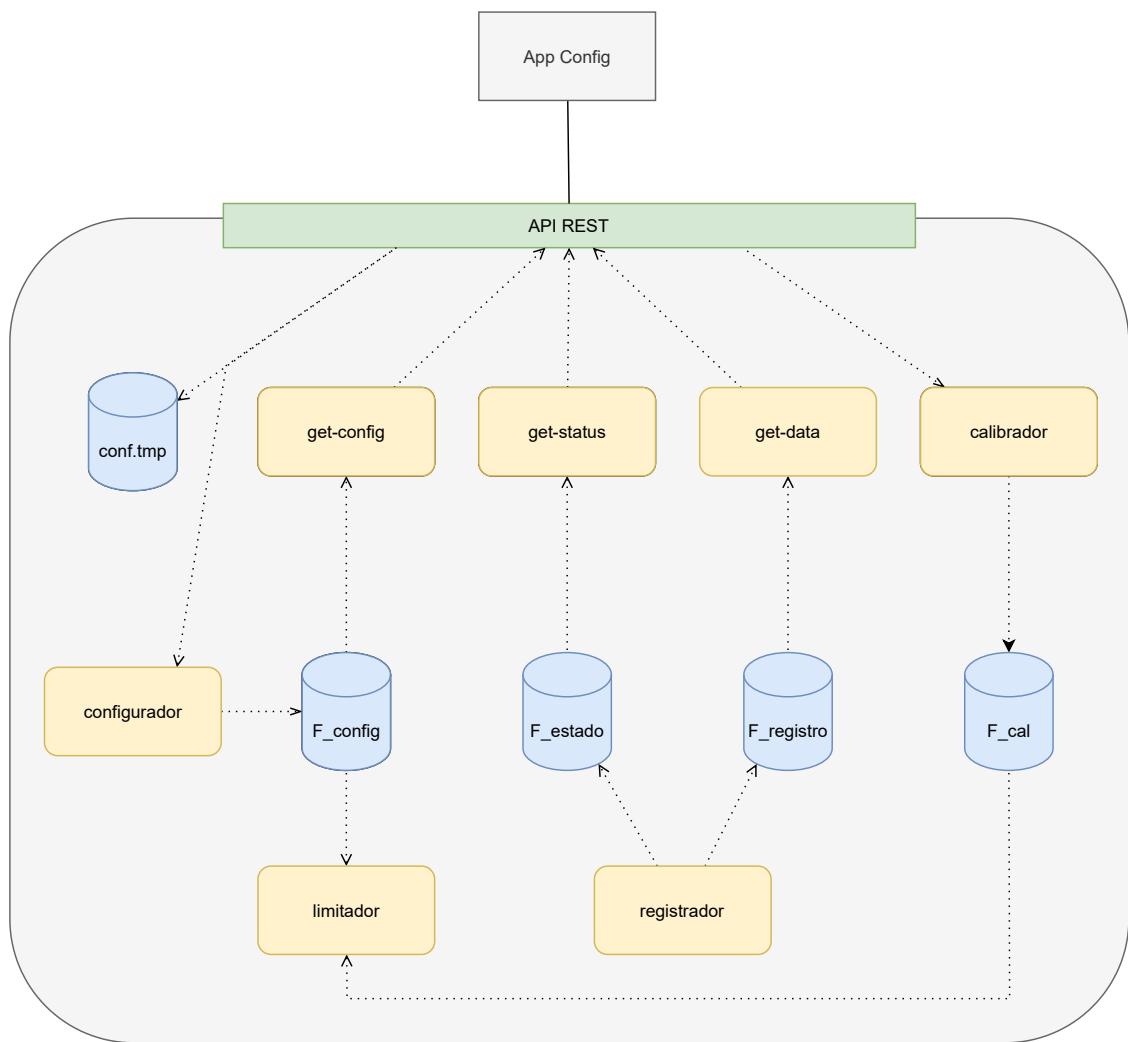


Figura 7.4 – Esquema parcial del sistema propuesto.

levantar podrían perfectamente tener su propio servicio, pero por el momento no se va invertir tiempo en eso y se propone como trabajo futuro.

Uno de los procesos que deben levantarse al iniciar el sistema es la aplicación *Discovery* o de descubrimiento, de forma que el limitador puede ser detectado por la aplicación de configuración dentro de la red local, por lo tanto existe una dependencia con el servicio de red, *networkd*.

```

1 [Unit]
2 Description=GranaSAT SoundLimiter bootup service
3 After=network.target
4 StartLimitIntervalSec=0
5 [Service]
6 Type=simple
7 Restart=always
8 RestartSec=1
9 User=root
10 ExecStart=/root/SoundLimiter/scripts/run.sh
11
12 [Install]
13 WantedBy=multi-user.target

```

Listado 7.1 – Ejemplo de servicio de arranque para el LM11

7.2 Instalación

Script Bash

El software debe ser instalado en el equipo. Por petición del cliente, los ficheros relativos al nuevo software de limitación deben estar centralizados en un sólo directorio, a diferencia de en el LM7 y LM9 que se encontraban dispersos por el sistema de archivos del sistema (/bin, /var, /tmp, /shm...).

El software a desarrollar será centralizado en la carpeta **SoundLimiter**, y dentro de ella habrá subdirectorios para almacenar y organizar las distintas partes del software (véase la figura 8.1). En el sistema propuesto, se centralizará el software en el directorio SoundLimiter.

Para la instalación del software se propone un script Bash que genere realice las siguientes tareas:

1. Crear los subdirectorios y ficheros utilizados por el software, así como el ajuste de sus permisos.
2. Instalar las librerías necesarias ([ALSA](#), [FFTW3](#) y [nCurses5](#)).
3. Actualizar el *PATH* del sistema, añadiendo las rutas absolutas a los directorios *bin/* y *scripts/* de nuestro proyecto. De esta forma se puede lanzar cualquiera de los programas y scripts sin especificar su ruta.
4. Compilar el código mediante el fichero *Makefile*.
5. Inicializar el registro, invocando al programa *inicializador*.

La instalación solo es necesaria realizar una vez. También es deseable un script que haga justamente lo contrario, desinstalar el software. Este script tan solo debe restaurar el *PATH* del sistema y eliminar la carpeta del proyecto de forma recursiva.

7.3 Detección de micrófono

WiringPI - PIN 28

Para la detección del micrófono se recurre a Luis Peinado Córdoba y Andrés María Roldán Aranda, ya que conocen el hardware del limitador ampliamente mejor que yo. Luis Peinado Córdoba proporciona un programa de prueba en el que se consulta el estado de conexión del micrófono. Este programa utiliza la librería [WiringPi](#), instalada por defecto en todas las Raspberry Pi.

El código se reduce a llamar a una función de [WiringPi](#), consultando el estado del PIN 28. Si la función devuelve 0, entonces el micrófono está conectado, en otro caso, desconectado. [30]

7.4 Procesamiento de audio

7.4.1 Conexión a tarjetas de sonido

Analizador del LM9 | [asound.conf](#)

El procesamiento de audio es en esencia el mismo que en el [LM9](#), aunque como se ha comentado anteriormente, en este caso tenemos una sola tarjeta de sonido que recibe todos los canales de entrada, en lugar recibir el audio del micrófono y de la música por sendas tarjetas de sonido.

De cara a permitir la reutilización de este módulo en nuestro proyecto, debemos encontrar una forma de que se puedan lanzar dos instancias del mismo programa para que lean de canales distintos de la tarjeta de sonido. Por suerte para nosotros, el prototipo viene semi-configurado para ello mediante el fichero de configuración global de [ALSA](#) [asound.conf](#), localizado en el directorio /etc. Tras un extenso estudio de la arquitectura de [ALSA](#) y con configuraciones, se identifica que la configuración existente mapea los canales 0 y 7 (**canal de entrada izquierdo** y **canal de entrada derecho** respectivamente) a un [PCM](#) [10] con el nombre “airplay”. Pasando este nombre el analizador podemos conectarnos y a los canales de entrada por los que se recibe la música.

A cada uno de los canales de la tarjeta de sonido del prototipo le corresponde un número positivo a modo de identificador. En los diagramas técnicos del prototipo, encontramos que los canales vienen dados tal y como se ve en la tabla [7.1](#).

Con esta información podemos completar la configuración de [ALSA](#) para crear un nuevo [PCM](#) que nos proporcione el audio de los micrófonos. Como el prototipo tiene tan sólo un micrófono instalado de momento, solo se ha generado la configuración para

BANDAS DE FRECUENCIA				
LM7 (8)	LM9 (31)			
31 Hz	20 Hz	125 Hz	800 Hz	5 KHz
63 Hz	25 Hz	160 Hz	1 KHz	6.3 KHz
125 Hz	31 Hz	200 Hz	1.25 KHz	8 KHz
250 Hz	40 Hz	250 Hz	1.6 KHz	10 KHz
500 Hz	50 Hz	315 Hz	2 KHz	12.5 KHz
1 KHz	63 Hz	400 Hz	2.5 KHz	16 KHz
2 KHz	80 Hz	500 Hz	3.15 KHz	20 KHz
4 KHz	100 Hz	630 Hz	4 KHz	

Figura 7.5 – Bandas de frecuencias de los limitadores.

el micrófono #2 (ver imagen 7.1). Añadir el primer micrófono al nuevo PCM resulta trivial, tan sólo hay que añadir la línea `ttable.0.2 2`, que significa “mapea al canal 2 de este PCM el canal 2 del dispositivo 0”.

Una vez configurado ALSA, podemos usar los analizadores de audio conectándolos a los PCM con nombre “airplay” y “mic”, en lugar de usar la tarjeta de sonido de forma global (“hw:0,0”).

0	Línea de entrada izquierda
7	Línea de entrada derecha
1	Micrófono #1
2	Micrófono #2
5	Línea de salida izquierda
6	Línea de salida derecha

Tabla 7.1 – Canales del limitador de GranaSAT.

```
1 # Record mixer
2 pcm.mixin {
3     type dsnoop
4     ipc_key 1024
5     ipc_key_add_uid false
6     ipc_perm 0666 # mixing for all users
7     slave {
8         pcm "hw:0,0"
9         channels 8
10        period_time 0
11        period_size 1024
12        buffer_size 8192
13        rate 48000
14    }
15    bindings {
16        0 1 # Mic: map channel 0 on this device to channel 1 on slave device
17        1 0 # Left: map channel 1 on this device to channel 0 on slave device
18        2 7 # Right: map channel 2 on this device to channel 7 on slave device
19    }
20}
21 # Microphone channel
22 pcm.mic {
23     type plug
24     slave.pcm "mixin"
25     ttable.0.0 1 # Mic: map channel 0 on this device to channel 0
26             # on slave device with 100% volume.
27 }
28 # Input channels
29 pcm.input {
30     type plug
31     slave.pcm "mixin"
32     ttable.0.1 1 # Left: map channel 0 on this device to channel 1
33             # on slave device with 100% volume.
34     ttable.1.2 1 # Right: map channel 1 on this device to channel 2
35             # on slave device with 100% volume.
36 }
37 # Playback mixer
38 pcm.mixout {
39     type dmix
40     ipc_key 2048
41     ipc_key_add_uid false
42     ipc_perm 0666 # mixing for all users
43     slave {
44         pcm "hw:0,0"
45         channels 8
46         period_time 0
47         period_size 1024
48         buffer_size 8192
49         rate 48000
50     }
51     bindings {
52         0 4 # Left: map channel 1 on this device to channel 4 on slave device
53         1 3 # Right: map channel 2 on this device to channel 3 on slave device
54     }
55 }
56 # Output channels
57 pcm.airplay {
58     type plug
59     slave.pcm "mixout"
60     ttable.0.0 1
61     ttable.1.1 1
62 }
```

Listado 7.2 – Fichero `asound.conf` completamente configurado

7.4.2 Interpretación de datos

Analizador del LM9 | Librería FFTW

Heredado del [LM9](#), ver [5.4](#). No se prevén cambios más allá del refactoring requerido por los problemas descritos en el capítulo [6](#).

7.4.3 Almacenamiento de audio

Ficheros *slow* y *fast* | SoundLimiter/tmp

Los ficheros de audio serán los mismos que el [LM9](#), con la diferencia de que van a ser centralizados y se van a almacenar dentro de la carpeta del proyecto: SoundLimiter/var.

Ver figura [5.4](#).

7.5 Gestión de calibración

7.5.1 Proceso de calibración

Calibrador | API HTTP REST

El proceso de calibración se encuentra como parte de la funcionalidad que ofrece el programa localservice en los limitadores estudiados. Para nuestro sistema, se va a recortar la funcionalidad relativa a la calibración de este programa y se va a llevar a un programa independiente, el Calibrador. Este programa ya existe de hecho en la versión 9 pero no contiene la funcionalidad completa ya que solo puede calibrar las líneas y no el micrófono. Para nuestro software se desea que este programa implemente ambas cosas y se puede tener un programa exclusivamente dedicado a calibrar el sistema.

7.5.2 Ecualizaciones

Ficheros de calibración | API HTTP REST

Se utilizarán las mismas ecualizaciones vistas en el [LM9 \(5.5.2\)](#), con la diferencia de que ahora se configurarán y consultarán mediante la [API HTTP REST](#) del limitador.

7.5.3 Emisión de ruido rosa

[Scripts](#) | [API HTTP REST](#)

Para la emisión del ruido rosa en el nuevo software se va a utilizar el fichero audio del [LM7](#) pink.wav. Para emitir ruido rosa simplemente reproducimos el fichero de audio. Para ello, se van a implementar dos scripts sencillos para lanzar y parar el ruido rosa (play-pink.sh y stop-pink.sh). La reproducción del fichero se realizará mediante [ALSA](#), con la orden `aplay`.

Para poder emitir ruido rosa debemos primero conectar la salida de audio de la Raspberry Pi a la salida de audio del equipo (ver diagrama [7.2](#)). Para ello se disponen de dos scripts proporcionados por Luis Peinado Córdoba para cambiar la salida del [PGA](#) a la tarjeta de sonido y viceversa. Estos scripts reciben el nombre de pgamode.sh y csmode.sh.

Para poder activar/desactivar el ruido rosa de forma remota, se proveerá de sendos *endpoints* en la [API HTTP REST](#) del limitador.

7.5.4 Verificación: sesiones

[Script del LM7](#)

Para el control de la sesiones de ruido y la verificación automática de las calibraciones se va a reutilizar el script del [LM7](#), junto con las dependencias que éste requiere para su funcionamiento.

7.5.5 Almacenamiento de calibraciones

[Ficheros](#) | SoundLimiter/var

El almacenamiento de las calibraciones será idéntico al visto en el [LM7](#) y [LM9](#), con la diferencia de que van a ser centralizados en el directorio del proyecto: SoundLimiter/var.

7

7.6 Gestión de la configuración

7.6.1 Proceso de configuración

[Configurador](#) | conf.tmp | [API HTTP REST](#)

La configuración del nuevo sistema ya no realizará mediante la aplicación web vista

en el capítulo 3. En su lugar, se propone una [API HTTP REST](#) para la consulta del estado del limitador y su configuración.

Durante el proceso de ingeniería inversa se averiguó que la configuración a modificar pasaba por un fichero plano antes de ser recuperada y aplicada por el configurador, el cual era parte del programa `localservice`. El sistema a desarrollar va a utilizar este fichero y a mantener el formato con el fin de poder reutilizar el proceso de configuración lo máximo posible.

Por otra parte, como el proceso de calibración se encuentra embebido en un programa que no va a ser de utilidad para nosotros, se va a extraer esta funcionalidad a un nuevo programa independiente, con el fin de no importar más código del estrictamente necesario.

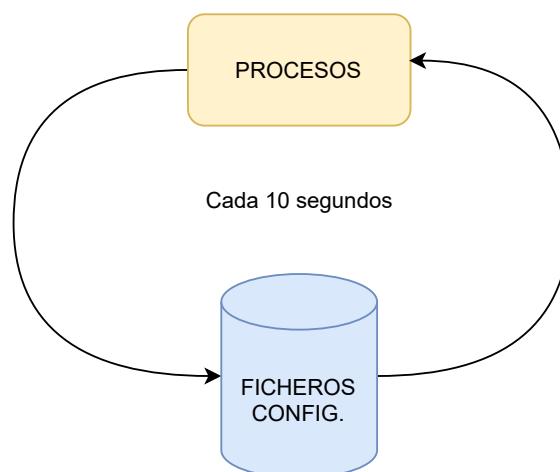
En resumen, el proceso de configuración se compondrá de tres partes bien definidas: la [API HTTP REST](#), el fichero de configuración temporal (en formato de texto plano) y el configurador.

La [API HTTP REST](#) recibirá la nueva configuración en formato [JSON](#) y llevará estos datos al fichero temporal con un formato que el configurador entiende, para finalmente invocar al programa Configurador para que lee y aplique la nueva configuración. Para más información sobre el fichero temporal y su formato se puede consultar el anexo [D.2.1](#).

7.6.2 Almacenamiento de la configuración

Fichero

El fichero de configuración será el mismo que en las versiones estudiadas pero será centralizado al directorio dedicado al limitador: `SoundLimiter/var/configuracion`



7

Figura 7.6 – Recarga de ficheros de configuración y calibración.

7.6.3 Consulta de la configuración

Fichero | get-config | API HTTP REST

La lectura de la configuración se realizará mediante el uso del programa `get-config`, al igual que en el [LM7](#) y el [LM9](#). La [API HTTP REST](#) consultará este programa y devolverá los datos en formato [JSON](#). Puede verse el anexo [C.1.1](#) para más detalles.

7.7 Gestión de registros

7.7.1 Proceso de registro

Registrador del LM9

El proceso de registro va a ser el mismo que el del [LM9](#), pero para poder reutilizarlo deben arreglarse algunos problemas, como los listados en el apartado [6](#). Este programa se compone esencialmente de un `main()` y varias funciones y variables globales. La idea es llevar esto a una clase y separar el bucle principal del algoritmo de registro. Además, las inclusiones directas de ficheros fuente deben eliminarse, así como todas las inclusiones de ficheros que no sean estrictamente necesarias.

7.7.2 Almacenamiento de registros

Fichero | SoundLimiter/var

El almacenamiento de los registros se realiza de forma análoga a como se ha visto en el [LM7](#) y en el [LM9](#), pero centralizando el fichero al directorio del proyecto: `SoundLimiter/var/registro.slr`.

Además, se va a cambiar el tipo de fichero a un fichero binario común, en lugar de utilizar un fichero especial de bloques como se vio en [4.10](#).

7.7.3 Consulta de registros

7

Fichero | get-data | API HTTP REST

Para la consulta de los registros se reutilizará el programa `get-data` y se conectará con la [API HTTP REST](#). El usuario proporciona los parámetros requeridos al endpoint de la [API HTTP REST](#), ésta consulta al programa `get-data` y devuelve los datos en [JSON](#). El anexo [C.1.3](#) amplia la información sobre este programa.

7.8 Proceso de atenuación

El proceso de atenuación del [LM9](#) parece demasiado complejo en comparación con el del [LM7](#), y no se comprende del todo su funcionamiento. Por otra parte, tampoco se tiene ninguna certeza de que realmente esa implementación funcione, por lo tanto, se va a romper con la tendencia y en este caso se va a preferir el proceso de atenuación del [LM7 5.8](#).

7.8.1 Comunicación con el PGA

La comunicación con el [PGA](#) se va a realizar mediante el uso de la clase `AudioModes` que puede verse en el diagrama de clases [??](#). Simplemente se llamará al método `writePGAdata` pasándole la ganancia deseada como parámetro.

Capítulo 8

Implementación del sistema

8.1 Metodología de trabajo

Para el desarrollo del proyecto, al menos en la parte relativa a la implementación, se ha escogido una metodología de trabajo ágil que permita cambios de todo tipo durante todo su ciclo de vida. Esto resulta necesario debido a la naturaleza del proyecto, ya que por una parte, aunque el objetivo del proyecto es claro (desarrollar un software capaz de controlar el nivel de ruido de un local), las tareas para conseguirlo no lo son tanto, y se asume que irán evolucionando y cambiando lo largo del proyecto. Por otro lado, la total inexperiencia en el campo de la acústica y la electrónica suponen un riesgo añadido.

Desde el comienzo del proyecto se han realizado cada martes en el laboratorio de [GranaSAT](#) reuniones semanales en las que participan todos los miembros involucrados en el desarrollo del proyecto de alguna u otra manera, esto es, las personas nombradas en la sección [1.3.1](#) al comienzo de este documento. Las veces que no se han podido realizar de forma presencial debido a la situación actual de pandemia causada al Covid-19, se han realizado de forma remota mediante videoconferencias.

Aunque la metodología seguida no se corresponde fielmente a la metodología SCRUM, sí que se asemeja en muchos aspectos. Si hacemos la correspondencia, Andrés María Roldán Aranda asume, por una parte, el rol de *Product Owner*, ya que entabla comunicación directa con el cliente, y por otra el rol de *Scrum Master*, ya que ha sido el supervisor de todo el trabajo realizado y ha tomado las decisiones técnicas de más alto nivel. El resto de los participantes asumiríamos el rol del *Equipo de desarrollo*.

La etapa de implementación se ha planificado con una duración de 5 semanas, sin ningún tipo de restricción o limitación dentro de ellas (sin *sprints*). La implementación se ha realizado de forma incremental y en bloque, desarrollando el backend, la [API HTTP REST](#) y pruebas conforme se iba avanzando en el desarrollo.

8.2 Herramientas utilizadas

En la tabla A.3 se presentan una lista con el software utilizado durante el proyecto.

- Para leer y desarrollar código se ha utilizado el [IDE](#) de Microsoft: Visual Studio Code, mientras que para el control de versiones y se ha utilizado Git.
- Para la generación de diagramas se ha usado Visual Paradigm e Inkscape.
- OpenAPI [18] ha permitido describir y especificar la [API HTTP REST](#) para luego generar código de forma automática. Para ello se ha utilizado un contenedor Docker con las herramientas de OpenAPI [18].
- Con Miktex y TexStudio se ha generado el presente documento.
- Los clientes [SSH](#) se han utilizado para gestionar las conexiones a los limitadores. Para conexiones desde fuera de la red de la [UGR](#) es necesario establecer una conexión mediante [VPN](#). Para ello se ha sudo el software CiscoAnyconnect.
- Google Spreadsheet, y en general Google Drive se ha utilizado de forma extensiva para la generación de documentos y presentaciones.
- Telegram se ha utilizado para la comunicación con el resto de participantes en el proyecto.

8.3 Proceso

En estas sección se describe cómo se han abordado las tareas necesarias para implementar el nuevo sistema.

8.3.1 Creando el entorno

Tal y como se comentó en la sección 3.6, la organización y estructura del proyecto de los LM era un auténtico desastre, y pronto se vio que trabajar con ese estructura iba a retrasar bastante el proceso de implementación. Con la intención de mejorar la organización para ganar en tiempo y calidad se decidió re-estructurar por completo el proyecto, creando así un nuevo entorno para nuestro proyecto.

Además del código fuente en sí, los ficheros utilizados por el software de los LM se encuentran dispersos en diferentes carpetas del sistema operativo. Por **petición explícita del cliente**, estos ficheros deben centralizarse y estar contenidos en un solo directorio.

8.3.1.1 Nueva estructura

Bajo el nombre de SoundLimiter, el proyecto obtiene la estructura de directorios que se ve en la figura 8.1. Por defecto, el directorio raíz del proyecto es /home/root/SoundLimiter. Esta **ruta** es **configurable** ya que ha sido **centralizada** en ficheros de configuración, reduciendo el número de ficheros a modificar a solamente 3: el Makefile, el fichero de constantes Config.h y el fichero de variables de entorno de la **API HTTP REST** (.env - actualmente esto no está implementado y las rutas se dan de forma explícita).

- api/

Todo lo relacionado con la **API HTTP REST**: ficheros de definición, Makefile, README, Dockerfile y código fuente.

- bin/

Destino de los programas compilados. Los programas de prueba van al subdirectorio test/.

- discovery/

Parte del servidor del sistema de descubrimiento.

- scripts/

Contiene scripts varios: para emitir y parar el ruido rosa, para instalar y desinstalar el software, para realizar pruebas...

- src/

Contiene todo el código fuente, compuesto por ficheros .h/.hpp y .cpp, con un sólo nivel secundario que contiene las clases relacionadas con el hardware.

- obj/

Aquí se alojarán el código objeto pre-compilado, con formato .o.

- test/

Programas de prueba implementados en C++.

- testUI/

Pequeña aplicación web para la realización de pruebas.

- tmp/

Aquí se guardan los ficheros temporales del sistema, esto es, los ficheros de audio y el fichero de configuración temporal.

- var/

Aquí van los ficheros auxiliares, como el fichero de ruido rosa, la configuración y las calibraciones.

8.3.1.2 Nuevo Makefile

La nueva estructura permite la generación de un fichero Makefile más potente de forma más sencilla. en este nuevo makefile se expresa al máximo la potencia de esta herramienta mediante la compilación por etapas del código fuente; primero se compila el código fuente, obteniendo código objeto, y luego se enlazan los código objeto y librerías necesarias para generar el ejecutable del programa. Con esta metodología sólo se recompilan las partes del código estrictamente necesarias cada vez que se modifica el código, y no todas ellas. En la figura 8.2 se puede visualizar este proceso.

El nuevo Makefile, cuyas primeras líneas pueden verse en el código 8.1, se compone de casi 400 líneas de código. Se divide en varios bloques, uno por cada programa a compilar, en los que se describen las dependencias entre el código fuente, las librerías necesarias, y las instrucciones de compilado en base a estas dependencias.

```
file://SoundLimiter
|-- Makefile
|-- README.md
|-- api
|-- bin
|-- discovery
|-- obj
|-- scripts
|-- src
|-- test
|-- testUI
|-- tmp
`-- var
```

Figura 8.1 – Estructura de directorios del proyecto. Los dos primeros elementos son ficheros, no directorios

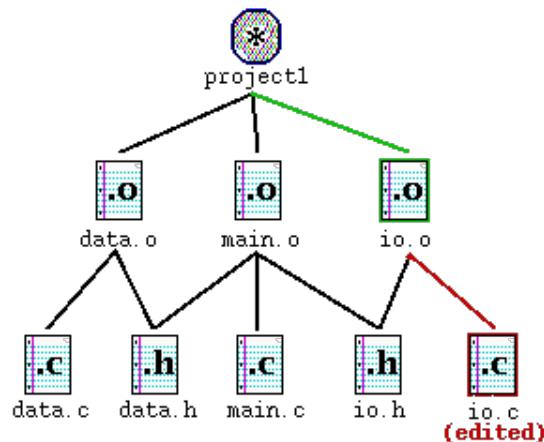


Figura 8.2 – Compilación con Make. Sólo se recompila el fichero modificado.

```

1 ######
2 #
3 # Limitador de Sonido - \glsname{granasat}
4 # Grado en Ingeniería Informática
5 #
6 # 2021 - Alejandro Ruiz Becerra <alexruiz@correo.ugr.es>
7 #
8 #
9 # Fichero Makefile para generación de ejecutables
10 #
11 #####
12
13 SLR_DIR = ~/SoundLimiter
14
15 # -----
16 # Compilador (C++)
17 # -----
18 CXX = g++
19
20 # -----
21 # Banderas para el compilador (C++)
22 # -----
23 CXXFLAGS = -std=c++17 -g -O3 -I$(shell pwd)
24 CXXEXTRA = -Wno-write-strings -Wno-unused-result -mcpu=arm7
25
26 # -----
27 # Macros generales
28 # -----
29 # Definición de directorios principales
30 #
31 SRC_DIR := ./src
32 BIN_DIR := ./bin
33 OBJ_DIR := ./obj
34 TEST_DIR := ./test
35
36 # ----- ##### ----- #
37 #          OBJETIVOS GENERALES
38 # ----- ##### ----- #
39 all: $(BIN_DIR)/analyzer \
40     $(BIN_DIR)/calibrator \
41     $(BIN_DIR)/get-config \
42     $(BIN_DIR)/get-data \
43     $(BIN_DIR)/get-status \
44     $(BIN_DIR)/init-station \
45     $(BIN_DIR)/sound-limiter \
46     $(BIN_DIR)/sound-register \
47     $(BIN_DIR)/update-config \
48     $(BIN_DIR)/hardware-controller
49
50 clean:
51     rm -rf $(OBJ_DIR)
52
53 superclean: clean
54     rm -rf $(BIN_DIR)
55
56 ... OBJETIVOS

```

Listado 8.1 – Makefile del LM11

8.3.2 Re-utilización de código

El objetivo es importar aquellos módulos seleccionados de los LM durante la etapa de [Análisis del sistema](#) al nuevo limitador, con el objetivo de “echarlos a andar” sobre el nuevo hardware. Mediante el uso de los diagramas de clases y de dependencias generados durante el proceso de ingeniería inversa, se ha podido extraer el código estrictamente necesario para compilar los programas seleccionados de forma independiente. Antes de llevarlos al nuevo limitador se han sometido a un proceso de refactoring en el que se ha intentado solucionar los problemas vistos en la sección 6.

Algunos de los ficheros han sido renombrados por nombres más descriptivos y para evitar confusiones. Por ejemplo cada uno de los módulos disponía de un fichero `main.cpp` en el que se encuentra el punto de entrada al programa. En la nueva estructura todos los ficheros de código fuente se encuentran al mismo nivel y no se pueden tener varios ficheros con el mismo nombre, además de ser poco descriptivo. En su lugar se han renombrado estos ficheros por `<Programa>Main.cpp` para distinguir a cada uno de ellos (por ejemplo `RegistradorMain.cpp`).

Un caso más importante es el del registrador, en el que tanto la clase que implementa el registrador (`Registrador.cc` en [LM9](#)), como el fichero que contiene el programa principal (`Registrador.cpp` en [LM9](#)) no sólo compartían el mismo nombre, sino que definen e implementan la funcionalidad en el mismo fichero (no separan en `.h` y `.cpp`).

En el nuevo software, se ha dividido las definiciones de las clases de su implementación y se ha separada la funcionalidad del registrador de su función principal, ampliando el número de ficheros de 2 a 5, pero bien estructurados, formateados y comentados. Además de todo esto, la salida del programa se ha actualizado para hacer uso del framework `nCurses` en lugar de la salida estándar, de forma que la vista se actualiza en lugar de sumarse a la salida anterior, lo que hacía que la salida del programa fuera poco legible.

Estos ficheros reciben el nombre de `GestorRegistro` (`.hpp` y `.cpp`), `Registrador` (`.hpp` y `.cpp`) y `RegistradorMain`.`cpp`

8.3.3 Novedades

Aunque es cierto que el grueso de la implementación del sistema ha sido heredada, también ha sido necesaria la implementación de clases y programas

8

Las novedades en el diseño se traducen en la implementación de nuevas clases y programas, descritos en la sección de [Diseño del sistema](#).

Principalmente todo lo relacionado con el nuevo hardware ha necesitado una nueva

implementación, por razones obvias. La comunicación con el hardware se ha realizado mediante las librerías **WiringPi**, **I2C** y **WiringSPI**. Los compañeros de **GranaSAT**, Luis Peinado Córdoba y Sharif Al-Husein Raie, con mucha más experiencia que yo en el campo de la electrónica y sistemas embebidos, me han proporcionado programas de prueba que implementan la comunicación con los componentes hardware del equipo. Estos programas han sido de gran ayuda ya que han supuesto una base y una guía importante a seguir, tanto que a veces tan solo ha sido necesario encapsular el código en clases e integrarlo en el proyecto.

PGA, **LCD**, **LEDs**, **buzzer** y la detección de micrófono hacen uso de estas librerías.

En las versiones estudiadas, el configurador formaba parte del programa **utilslr**. En el nuestro software se ha implementado un programa llamado **update-config** cuya única responsabilidad es actualizar la configuración del sistema a partir del contenido del fichero **tmp.conf**.

8.3.3.1 La API-REST

Sin duda el cambio la introducción de una **API HTTP REST** al sistema para consultar su estado y controlarlo de forma externa supone la novedad más rompedora a nivel de software. La antigua aplicación, aunque funcional y bien desarrollada visualmente, tenía problemas importantes que han hecho imposible su portabilidad. La **API HTTP REST** se ha diseñado precisamente bajo este pretexto y con el fin de solucionar el problema expuesto.

La **API HTTP REST** ha sido implementada en *Python*, concretamente mediante el uso del framework de *Connexion*, construido sobre el framework de *Flask* y diseñado para funcionar mano a mano con *Swagger/OpenAPI* [24]. Gracias a la maravillosa herramienta de OpenAPI [18] se ha podido generar código de forma automática tan sólo instalando un contenedor de Docker. La herramienta permite generar código en una gran variedad de lenguajes y marcos de trabajo, gracias a los patrones que estos frameworks definen, y todo esto tomando como origen el fichero de definición de la **API HTTP REST**, un fichero en formato YML o **JSON** en el que se definen los servicios que expone la **API HTTP REST**, así como los modelos y controladores que necesita.

La definición de la **API HTTP REST** del sistema se extiende casi a las 1 000 líneas de código YML.

```

1 def update_config(configurable_properties: ConfigurableProperties = None):
2     """Updates limiter's configuration
3
4     Updates limiter's configuration based on the data
5     received within this POST request. The data must be on
6     JSON format.
7
8     See the example below for a list of all available

```

```
9     properties that are configurable.
10
11    :param configurable_properties: List of every configurable
12        property of the sound limiter. None of the fields are required.
13    :type configurable_properties: dict | bytes
14
15    :rtype: GenericResponse
16    """
17
18    if connexion.request.is_json:
19        configurable_properties = ConfigurableProperties.from_dict(connexion.
20            request.get_json())
21
22        # Construir conf.tmp con los datos recibidos desde el JSON
23        configurator = Configurator(configurable_properties)
24        print(configurable_properties)
25        configurator.configure()
26
27        # Una vez exportados los datos al fichero conf.tmp, lanzar el
28        # configurador.
29        cmd = "/root/SoundLimiter/bin/update-config"
30
31    try:
32        success = os.popen(cmd)
33        if success:
34            status_code, message = 200, "Configuration updated"
35        else:
36            status_code, message = 500, "An unexpected error occurred during
37            the configuration"
38        return GenericResponse(status_code, message)
39    except OSError:
40        return GenericResponse(503, "Could not query limiter's configurator
41        service")
```

Listado 8.2 – *API HTTP REST del LM11*

8.4 Resultados

AÑADIR IMÁGENES DE API EN LIMITADOR

Capítulo 9

Validación y test

La validación del software se ha realizado principalmente mediante tests manuales a medida que se iban desarrollando e incluyendo las funcionalidades en el proyecto. La naturaleza del proyecto hace especialmente complejo el uso de herramientas que permitan la realización de tests automáticos, debido a que la mayoría de los tests requieren una comprobación sensorial, como el caso de la escritura al display [LCD](#), o la atenuación de sonido por parte del [PGA](#).

En algunos casos, se ha podido automatizar estos comportamientos para facilitar el proceso de pruebas y validación. En concreto, para las comprobar el correcto funcionamiento de los componentes hardware se han podido desarrollar scripts automatizando los procesos de prueba.

- Pantalla [LCD](#)
 - Se ha implementado un script que automatiza las tareas de encendido, apagado y escritura de una matriz de caracteres de 4x20.
- Detección de micrófono
 - Se ha implementado un programa que consulta el estado de conexión del micrófono. Por defecto indica si el micrófono está conectado o no una sola vez, aunque mediante el paso de argumentos se le puede indicar que haga un sondeo continuo, en un intervalo de un segundo.
- Buzzer de audio
 - La librería utilizada provee un script de pruebas mediante el cual se reproduce una sinfonía.
- [LEDs RGB](#)

- La librería utilizada provee un script de pruebas que enciende y apaga los **LEDs** aplicando distintos colores.

- **PGA**

- Se ha desarrollado un programa de pruebas que permite al usuario aplicar distintos valores de ganancia. Para comprobar su funcionamiento se requiere que se reproduzca sonido, de forma que se pueda notar la diferencia de volumen.

- Emisión de audio y ruido rosa

- Se ha desarrollado un script que aplica redirige la entrada de audio a la salida. Aunque esto parezca trivial, es necesario configurar el driver de audio de la tarjeta de sonido para que esto funcione. El script se encarga de aplicar estas configuraciones y reproducir la emisión de un canal de radio en internet.
- Se ha creado un script que cambia la configuración del driver y reproduce ruido rosa.

- **API HTTP REST**

- Además de las pruebas manuales, el generador de OpenAPI [18] genera una serie de tests unitarios con el framework de Python UnitTest [23].

Capítulo 10

Conclusiones y trabajo futuro

10.1 Trabajo futuro

Lamentablemente el proyecto no ha podido completarse al 100% por falta de tiempo. La incertidumbre causada por un roadmap que no siempre estuvo claro, la gran inexperiencia en la materia y la dependencia de una infraestructura software y hardware que no siempre estuvo exenta de fallos han supuesto unos riesgos en el proyecto que finalmente han acabado materializándose. Sin embargo, sí que se ha conseguido realizar gran parte del camino, llegando a ver la meta final a tan solo unas pocas semanas más de trabajo (se estima que el proyecto se encuentra entorno al 90% de completitud).

Principalmente ha faltado la realización de más pruebas sobre el software, para detectar errores y refinar el sistema. Todas las funcionalidades integradas en el proyecto se han sometido a una serie de pruebas que garantizar su funcionamiento, pero en aquellas partes relacionadas con el campo de la acústica es difícil como Ingeniero Informática dar por válido o no el funcionamiento.

Relacionado con esto está **el proceso de limitación**, del cuál no se ha podido realizar ningún test más allá de su correcta compilación, **el proceso de calibración** el cual se ha probado y parece funcionar correctamente pero no se sabe con seguridad por la inexperiencia en el campo.

Además de lo comentado hasta ahora, como trabajo futuro se propone lo siguiente:

1. Proporcionar a la **API HTTP REST** de una capa de seguridad mediante **OAuth2** [1] o API tokens [17].

Actualmente la **API HTTP REST** carece de seguridad en este sentido y es completamente abierta y pública a la red. Como el equipo se encuentra en fase de

prototipo esto no es esencial de momento, pero sí que lo es para poder pasar a producción.

2. Mejorar el manejo de la configuración (y posiblemente el de los ficheros de audio también) mediante el uso de una base de datos (relacional o no relacional).
3. Simplificar el registro.

El archivo de registro podría ser perfectamente un fichero con formato [JSON](#) en el que cada una de las líneas representa un registro. De esta forma, su manejo y mantenimiento sería inmensamente más simple, y el programa `get-data` tan sólo tendría que consultar y devolver los registros solicitados, sin necesidad de darles formato.

4. Diseño e implementación de un sistema que notifique a los servicios del limitador de los eventos que les interesen, como por ejemplo un despachador de eventos. Ver imagen [10.1](#).

Por ejemplo, notificar a los servicios si se modifica la configuración. Actualmente recargan la configuración cada 10 segundos.

5. Continuar con el proceso de mejora y refactoring para arreglar los problemas mencionados en la sección [6](#) respecto al código.
6. Verificar el buen funcionamiento del equipo.
7. Integrar el uso del [LCD](#), [LEDs](#) y [Buzzer](#) como parte del software principal.

Actualmente estos elementos funcionan y pueden ser controlados mediante código, pero no hay ningún programa que los utilice, tan sólo los de prueba.

8. Promocionar todos los programas que deben correr de forma ininterrumpida a servicios de [systemd](#).

10.2 Conclusiones

Como punto final a la realización de este Trabajo de Fin de Grado se procede a repasar cada uno de los objetivos del proyecto, indicando su grado de cumplimiento y cómo se han abordado.

Los resultados obtenidos en este proyecto han sido mayormente satisfactorios, ya que se han cumplido la gran mayoría de los objetivos y requisitos marcados y se han puesto en práctica los conocimientos adquiridos en durante la realización del Grado en Ingeniería Informática. De hecho, la naturaleza del proyecto, no sólo ha exigido el uso del conocimiento adquirido en varias áreas de la informática, sino que ha requerido el

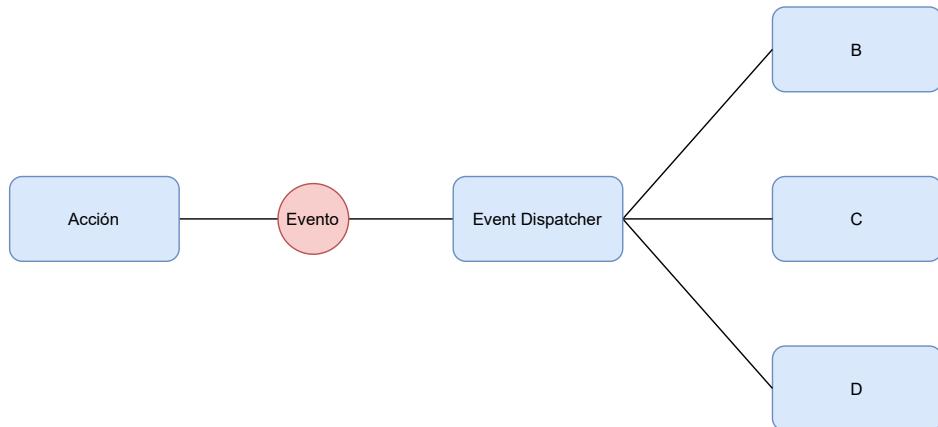


Figura 10.1 – Diagrama conceptual de un Despachador de Eventos.

aprendizaje de metodologías y herramientas nuevas, tanto del área de la informática, como del área de la ingeniería y la electrónica. Además, se ha trabajado en un entorno plural en el que la cooperación y la comunicación entre los compañeros y compañeras para el alcance de los objetivos de éste, y de otros proyectos, ha tenido un papel determinante.

El objetivo principal casi ha quedado completado. Se dispone de un equipo capaz de registrar los niveles de presión acústica de emisión y recepción en el local donde se encuentra instalado. Esto se ha conseguido mediante el uso de un micrófono y transformaciones matemáticas, que quedan fuera del ámbito de este proyecto, para obtener un modelo acústico medido en decibelios a partir de la intensidad de la señal de entrada. La atenuación del sonido por el contrario no ha podido completarse debido a falta de tiempo, pero sí que se ha generado un diseño, una implementación (no probada) y funcionalidad para controlar el actuador, el PGA instalado en el hardware. Todas estas cuestiones han sido descritas a mayor nivel de detalle en el capítulo de [Diseño del sistema](#).

El resto de objetivos y requisitos han sido completados, se han estudiado las necesidades de un sistema de estas características mediante el extenso proceso de ingeniería inversa que se ha realizado sobre los limitadores [LM7](#) y [LM9](#), redactado en los capítulos [Chapter, Ingeniería Inversa: la versión LM7](#) y [Ingeniería Inversa: la versión LM9](#). De estas necesidades se han extraído los requisitos del sistema, descritos en el apartado [Especificación de requisitos](#). Como resultado también del proceso de ingeniería inversa, se ha estudiado si había elementos en estos sistemas que pudieran ser de utilidad en nuestro proyecto en el capítulo [Análisis del sistema](#), y se ha diseñado e implementado el sistema requerido: capítulos [Diseño del sistema](#) y [Implementación del sistema](#). Finalmente, se ha realizado una serie de pruebas para comprobar el buen funcionamiento del diseño propuesto y la implementación realizada, capítulo [Validación y test](#)

El hecho de que en el momento de dar punto final al proyecto no se disponga de un equipo completamente operativo para entrar a producción no eclipsa todo el trabajo, esfuerzo y tiempo invertido en el proyecto. Todas las herramientas y piezas necesarias para su finalización están completamente disponibles llegados a este punto, por lo que llevar el producto a su estado final no debería implicar demasiado esfuerzo adicional. De hecho, todo el trabajo realizado durante este proyecto se traduce de forma directa en una sólida base sobre la que poder trabajar y continuar el desarrollo del producto, de forma que un futuro pueda entrar al mercado.

A nivel personal, considero sin duda alguna que los mayores logros durante la realización de este proyecto ha sido el proporcionar una extensa documentación en la que se reflejan las necesidades de un sistema de estas características, su diseño y su funcionamiento, así como proporcionar un proyecto software con una estructura y un código inmensamente mejor en cuanto a calidad se refiere.

Como comentario final, decir que aunque ha sido una experiencia inesperadamente dura, y muchas veces frustrante, me doy por satisfecho con el titánico trabajo realizado y considero que finalizo este proyecto con más y mejores capacidades como profesional.

Bibliografía

- [1] O. 2.0. Oauth 2.0 official site. <https://oauth.net/2/>.
- [2] ALIX. Alix-1e. <https://www.pcengines.ch/alix1e.htm>.
- [3] ALSA. Documentacion de alsa. <https://www.alsa-project.org/alsa-doc/alsa-lib/index.html>.
- [4] AMD. Amd's cs5536 companion chip. http://modelofreality.org/snd_amd5536.html.
- [5] cs.washington.edu. Introduction to digital filters. <https://courses.cs.washington.edu/courses/cse474/18wi/pdfs/lectures/Intro%20to%20DSP.pdf>.
- [6] Debian. Debian gnu/linux system administrator's manual - chapter 8 - managing user accounts. <https://www.debian.org/doc/manuals/system-administrator/ch-sysadmin-users.html>.
- [7] Docker. Docker. <https://www.docker.com/>.
- [8] fftw.orh. Fastest fourier transform in the west. <https://www.fftw.org/>.
- [9] T. Fisher. mkfilter. <https://github.com/university-of-york/cs-www-users-fisher>.
- [10] D. G. GarzAdvanced linux sound architecture. <http://canvoki.net/Modders/Docs/alsadoc-pcm.html>.
- [11] L. Journal. Introduction to sound programming with alsa. Linux Journal, 2004.
- [12] L. man pages. fcntl. <https://man7.org/linux/man-pages/man2/fcntl.2.html>.
- [13] L. man pages. fcntl. <https://man7.org/linux/man-pages/man2/fcntl.2.html>.
- [14] L. man pages. termios. <https://man7.org/linux/man-pages/man3/tcflow.3.html>.
- [15] B. Morel. Creating a linux service with systemd.

References

- [16] nixCraft. Faq - where are the passwords of the users located in linux? <https://www.cyberciti.biz/faq/where-are-the-passwords-of-the-users-located-in-linux>.
- [17] OpenAPI. Definici un esquema de seguridad en openapi. <https://spec.openapis.org/oas/v3.1.0#security-scheme-object>.
- [18] OpenAPI. Documentaciicial de openapi v3.1.0. <https://spec.openapis.org/oas/v3.1.0>.
- [19] OpenAPI. Openapi generator. <https://github.com/OpenAPITools/openapi-generator#16---docker>.
- [20] OpenSSL. Documentacion oficial de opensll. <https://www.openssl.org/docs/manmaster/man1/openssl-passwd.html>.
- [21] OSS. Documentaci oss. <http://manuals.opensound.com/>.
- [22] K. Panic. Makefile and c compiler basics. <https://krnlpanic.com/wp/c-compiler-basics/>.
- [23] Python. Python unittest. <https://docs.python.org/3/library/unittest.html>.
- [24] Swagger. Documentaciicial de swagger. <https://swagger.io/specification/>.
- [25] top. Display linux processes manual. <https://man7.org/linux/man-pages/man1/top.1.html>.
- [26] L. Torvals. Linux kernel devices. <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/devices.txt>.
- [27] Wikipedia. Filter bank. https://en.wikipedia.org/wiki/Filter_bank.
- [28] Wikipedia. Standard unix filesystem hierarchy. <https://upload.wikimedia.org/wikipedia/commons/f/f3/Standard-unix-filesystem-hierarchy.svg>.
- [29] Wikipedia. Uart. https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.
- [30] WiringPi. Wiringpi core functions. <http://wiringpi.com/reference/core-functions/>.
- [31] zalando. Documentaciicial de connexion. <https://connexion.readthedocs.io/en/latest/>.

Apéndice A

Presupuesto

A.1 Recursos físicos

Recurso	Coste (€)
Limitador LM7	Gratis (GranaSAT)
Limitador LM11 (prototipo)	Gratis (GranaSAT)
Amplificador y/o altavoces	Gratis (GranaSAT)
Sonómetro profesional	Gratis (GranaSAT)
Micrófono	Gratis (GranaSAT)
Cables balanceados XLR	Gratis (GranaSAT)
Ordenador portátil	580
Ordenador de torre	420
TOTAL	1 000

Tabla A.1 – Costes hardware.

Los recursos hardware relativos al limitador no han supuesto un coste adicional ya que los proporciona la empresa para la que se realiza el proyecto. El hardware del

nuevo prototipo queda fuera del ámbito de este proyecto (además, el coste del prototipo se desconoce completamente). Aún así, se incluyen en el listado de recursos hardware los componentes que han resultado esenciales para el desarrollo del proyecto, de forma simbólica, con un coste de 0 €.

Al margen de esto, ha sido necesario un ordenador portátil y otro de sobremesa para trabajar en el proyecto, los cuales se incluyen en el presupuesto como parte de equipamiento técnico.

A.2 Recursos humanos

Para estimar los costes del personal, se ha consultado la página web www.indeed.com con el fin de obtener datos referente a los salarios de los profesionales requeridos para el proyecto. Esa fuente se considera fiable al proporcionar información salarial estimada a partir de empleados, ofertas de empleo o directamente desde empresas.

Última actualización: 13 de agosto de 2021.

Los perfiles buscados se describen en el listado siguiente, mientras que el desglose del coste de recursos humanos se representa en la tabla A.2

1. Junior Software Engineer:

- (a) Con un salario de 18 000 € por año (10 €/h en un contrato de 1 800 horas).
- (b) Trabajará en el proyecto un total de 600 horas a media jornada, es decir, 5 horas al día.
- (c) Implementará el sistema.

2. Senior Software Engineer:

- (a) Con un salario de 45 000 € por año (25 €/h en un contrato de 1 800 horas).
- (b) Trabajará en el proyecto un total de 240 horas, es decir, 2 horas al día.
- (c) Investigará el ecosistema del proyecto y diseñará el sistema.

Posición	Tiempo (h)	Coste (€)
Junior Software Engineer	600	6 000
Senior Software Engineer	240	6 000
TOTAL		12 000

Tabla A.2 – Costes humanos

A.3 Software

Para la realización del proyecto se han usado herramientas software gratuitas. La mayor parte de ellas son herramienta de gestión y productividad, como los clientes SSH. En el ámbito de la ingeniería se ha utilizado la herramienta *Visual Paradigm* para la generación del diagramas y el IDE *VSCode* para el desarrollo de código.

La comunicación con el equipo ha tenido lugar de forma presencial principalmente, mediante la realización de reuniones semanales en el laboratorio de GranaSAT. Para la comunicaciones en remoto se ha utilizado la herramienta de mensajería instantánea Telegram, a través de un grupo creado para este propósito y en el que se encuentran los miembros involucrados en el proyecto.

Software	Coste (€)	Categoría
Visual Studio Code	Gratis	IDE
Visual Paradigm CE	Gratis	Diagramas
Git	Gratis	Versionado
OpenAPI	Gratis	Desarrollo
Docker	Gratis	Desarrollo
Snowflake SSH Client	Gratis	Conectividad
MobaXterm SSH Client	Gratis	Conectividad
Cisco AnyConnect	Gratis	Conectividad
MikTex	Gratis	Documentación
TexStudio	Gratis	Documentación
Inkscape	Gratis	Diagramas
Google Drive	Gratis	Productividad
Telegram	Gratis	Comunicación

Tabla A.3 – Costes software

A.4 Presupuesto final

Teniendo en cuenta los costes calculados anteriormente, el presupuesto final para poder llevar a cabo este proyecto asciende a un total de **13 000 €**.

Apéndice B

Terminología

1. Nivel de Emisión.- Se entiende por nivel de emisión el nivel de presión acústica con ponderación normalizada A (dBA) originado por una fuente sonora.

El nivel de presión acústica (dB) queda definido por la relación:

$$dB = 20 \log \left(\frac{V}{V_0} \right) \quad (\text{B.o.1})$$

Siendo:

- (a) P .- el valor de presión acústica generada por la fuente.
(b) P_0 .- el valor de la presión acústica de referencia.
2. Nivel de Recepción.- Es el nivel de presión acústica existente en un determinado lugar, originado por una fuente sonora en un emplazamiento diferente. En el contexto del presente proyecto, es el nivel de presión acústica medido por el micrófono.

References

Apéndice C

Módulos del limitador

En este anexo se presentan los módulos del limitador bajo un enfoque técnico. Para cada uno de los programas se adjunto su diagrama de clases (simplificado) y sus interfaces de entrada y salida de datos. Estos programas los discriminamos por re-utilizados y legacy, dependiendo de si forman parte del nuevo software o no.

C.1 Módulos re-utilizados

Estos programas han sido sometidos a una revisión de calidad mediante la cual se ha re-extracturado el código o se han re-implementado clases enteras, pero su diseño y su funcionamiento se han mantenido intactos.

C.1.1 getConfig

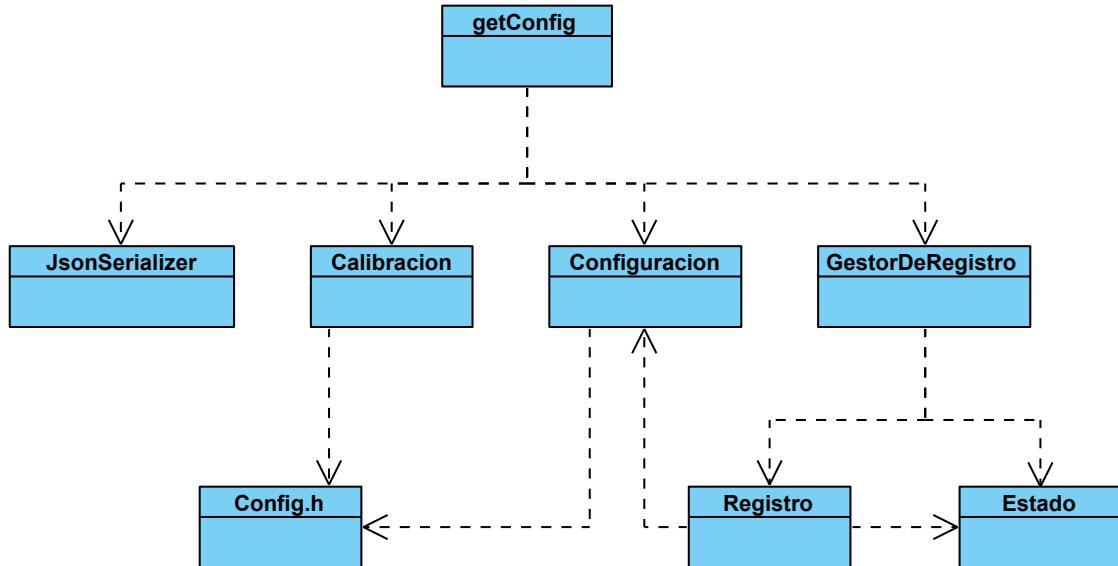


Figura C.1 – Diagrama de clases del programa getConfig

El programa getConfig devuelve la configuración del limitador en formato JSON. Para ello, accede a la información alojada en varios ficheros del sistema. Los ficheros a los que accede, así como el conjunto de datos que devuelve el programa pueden consultarse en los listados que se muestran a continuación en este anexo.

Ficheros:

- /var/slrl/configuracion
- /var/slrl/registro.slr
- /var/slrl/calibration[i]
- o: micrófono.
- 1: línea izquierda.
- 1: línea derecha.

Datos:

- Calibraciones.
- Normativa.
- Datos del local.
- Tipo de equipo.
- Propiedades del compresor.
- Umbral de sesiones.
- Aislamiento.
- Versión del software.
- Número de serie del equipo.
- Fecha y hora del sistema.
- Fecha y hora base del registro.

```
1  {
2    "definitions": {
3      "data": {
4        "type": "object",
5        "properties": {
6          "serialNumber": {
7            "type": "integer"
8          },
9          "version": {
10            "type": "number"
11          },
12          "time": {
13            "type": "string"
14          },
15          "configTime": {
16            "type": "string"
17          },
18          "place": {
19            "type": "string"
20          },
21          "establishment": {
22            "type": "string"
23          },
24          "address": {
25            "type": "string"
26          },
27          "council": {
28            "type": "string"
29          },
30          "responsible": {
31            "type": "string"
32          },
33          "phonenumbers": {
34            "type": "string"
35          },
36          "vendor": {
37            "type": "string"
38          },
39          "dayStartHour": {
40            "type": "integer"
41          },
42          "dayEndHour": {
43            "type": "integer"
44          },
45          "dayMaximumEmission": {
46            "type": "number"
47          },
48          "nightMaximumEmission": {
49            "type": "number"
50          },
51          "dayMaximumReception": {
52            "type": "number"
53          },
54          "nightMaximumReception": {
55            "type": "number"
56          },
57          "plainIsolation": {
58            "type": "number"
59          },
60          "minumum": {
61            "type": "number"
62          }
63    }
64  }
```

References

```
63     "maximum": {
64         "type": "number"
65     },
66     "maximumattenuation": {
67         "type": "integer"
68     },
69     "compressorRange": {
70         "type": "number"
71     },
72     "compressorEfectivity": {
73         "type": "integer"
74     },
75     "laeqtime": {
76         "type": "integer"
77     },
78     "recoverTime": {
79         "type": "integer"
80     },
81     "registryStartTime": {
82         "type": "string"
83     },
84     "deviceType": {
85         "type": "string"
86     }
87   }
88 }
89 }
```

Listado C.1 – Esquema JSON devuelto por getConfig en la versión LM7.

```
1 {
2   "definitions": {
3     "data": {
4       "type": "object",
5       "properties": {
6         "serialNumber": {
7           "type": "string"
8         },
9         "version": {
10           "type": "string"
11         },
12         "time": {
13           "type": "string"
14         },
15         "configtime": {
16           "type": "string"
17         },
18         "activeSince": {
19           "type": "string"
20         },
21         "registryBaseTime": {
22           "type": "string"
23         },
24         "minimum": {
25           "type": "number"
26         },
27         "maximum": {
28           "type": "number"
29         },
30         "recoverTime": {
31           "type": "integer"
32         }
33       }
34     }
35   }
36 }
```

```

32 },
33 "maximumAttenuation": {
34   "type": "integer"
35 },
36 "maximumBandAttenuation": {
37   "type": "number"
38 },
39 "minimalAttenuation": {
40   "type": "number"
41 },
42 "thresholdCoefficient": {
43   "type": "number"
44 },
45 "attenuationSteps": {
46   "type": "integer"
47 },
48 "laeqTime": {
49   "type": "integer"
50 },
51 "dayStartHour": {
52   "type": "integer"
53 },
54 "dayEndHour": {
55   "type": "integer"
56 },
57 "dayMaximumEmission": {
58   "type": "number"
59 },
60 "nightMaximumEmission": {
61   "type": "number"
62 },
63 "dayMaximumReception": {
64   "type": "number"
65 },
66 "nightMaximumReception": {
67   "type": "number"
68 },
69 "compressorRange": {
70   "type": "number"
71 },
72 "compressorEffectivity": {
73   "type": "integer"
74 },
75 "isolation": {
76   "type": "array",
77   "items": {
78     "type": "number"
79   }
80 },
81 "margins": {
82   "type": "array",
83   "items": {
84     "type": "number"
85   }
86 },
87 "controlType": {
88   "type": "string"
89 },
90 "usePredictiveModel": {
91   "type": "boolean"
92 },
93 "useReceptionControl": {
94   "type": "boolean"

```

References

```
95 },
96 "audioBlocking": {
97   "type": "boolean"
98 },
99 "useFrecuentialControl": {
100   "type": "boolean"
101 },
102 "controlEachFrequency": {
103   "type": "boolean"
104 },
105 "establishment": {
106   "type": "string"
107 },
108 "address": {
109   "type": "string"
110 },
111 "council": {
112   "type": "string"
113 },
114 "responsible": {
115   "type": "string"
116 },
117 "phonenumerber": {
118   "type": "string"
119 },
120 "vendor": {
121   "type": "string"
122 },
123 "equipment": {
124   "type": "string"
125 },
126 "activeDaysAfterConfiguration": {
127   "type": "integer"
128 },
129 "daysOff": {
130   "type": "array",
131   "items": {
132     "$ref": "#/definitions/DaysOff"
133   }
134 },
135 "weekDayEmisionMaximum": {
136   "type": "array",
137   "items": {
138     "type": "number"
139   }
140 },
141 "weekNightEmisionMaximum": {
142   "type": "array",
143   "items": {
144     "type": "number"
145   }
146 },
147 "weekDayReceptionMaximum": {
148   "type": "array",
149   "items": {
150     "type": "number"
151   }
152 },
153 "weekNightReceptionMaximum": {
154   "type": "array",
155   "items": {
156     "type": "number"
157   }
```

```

158 },
159 "useExtendedNormative": {
160   "type": "boolean"
161 },
162 "deviceType": {
163   "type": "string"
164 },
165 "pinkNoiseLevel": {
166   "type": "number"
167 },
168 "sessionStartThreshold": {
169   "type": "number"
170 },
171 "sessionEndThreshold": {
172   "type": "number"
173 },
174 "calibrations": {
175   "type": "array",
176   "items": {
177     "$ref": "#/definitions/Calibration"
178   }
179 }
180 },
181 "Calibration": {
182   "type": "object",
183   "properties": {
184     "deviceNumber": {
185       "type": "integer"
186     },
187     "dbRef": {
188       "type": "number"
189     },
190     "referentia": {
191       "type": "number"
192     },
193     "noiselevel": {
194       "type": "number"
195     },
196     "hardwareLevel": {
197       "type": "number"
198     },
199     "referenciaGlobal": {
200       "type": "number"
201     },
202     "equalization": {
203       "type": "array",
204       "items": {
205         "type": "number"
206       }
207     },
208     "internalEqualization": {
209       "type": "array",
210       "items": {
211         "type": "number"
212       }
213     }
214   }
215 },
216 "DaysOff": {
217   "type": "object",
218   "properties": {
219     "type": {

```

References

```
221     "type": "integer"
222 },
223 "day": {
224     "type": "integer"
225 },
226 "month": {
227     "type": "integer"
228 },
229 "weekDayNumber": {
230     "type": "integer"
231 },
232 "startHour": {
233     "type": "integer"
234 },
235 "startMinutes": {
236     "type": "integer"
237 },
238 "durationHours": {
239     "type": "integer"
240 },
241 "durationMinutes": {
242     "type": "integer"
243 },
244 "emissionMaximum": {
245     "type": "number"
246 },
247 "receptionMaximum": {
248     "type": "number"
249 }
250 }
251 }
252 }
253 }
```

Listado C.2 – Esquema JSON devuelto por getConfig en las versiones LM9 y LM11.

C.1.2 getStatus

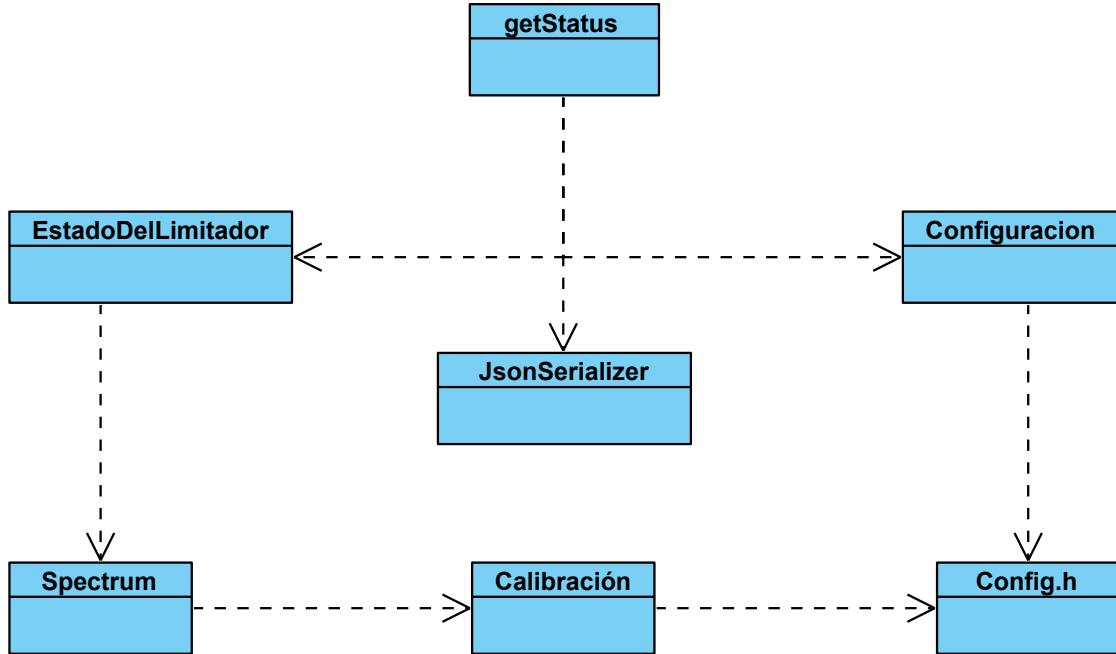


Figura C.2 – Diagrama de clases del programa getStatus

El programa `getStatus` devuelve información sobre el estado del limitador en el momento de la llamada. Para ello, lee el fichero `/tmp/estado.slr` y devuelve su contenido en formato JSON. El fichero `/tmp/estado.slr` es mantenido por el proceso de limitación en el caso del LM7, o por el proceso registrador en el caso del LM9 y el LM11.

En la figura C.2 se muestra el diagrama de clases simplificado del programa.

Los listados C.3 y C.4 muestran los esquemas JSON devueltos por este programa en cada una de las versiones.

```

1  {
2    "definitions": {
3      "data": {
4        "type": "object",
5        "properties": {
6          "serialNumber": {
7            "type": "string",
8            "format": "integer"
9          },
10         "version": {
11           "type": "string"
12         },
13         "time": {
14           "type": "string"
15         },
16         "configTime": {
17           "type": "string"
18         }
19       }
20     }
21   }
  
```

References

```
17         "type": "string"
18     },
19     "place": {
20         "type": "string"
21     },
22     "direction": {
23         "type": "string"
24     },
25     "council": {
26         "type": "string"
27     },
28     "responsible": {
29         "type": "string"
30     },
31     "phoneNumber": {
32         "type": "string"
33     },
34     "vendor": {
35         "type": "string"
36     },
37     "deviceType": {
38         "type": "string"
39     },
40     "leftline": {
41         "type": "string"
42     },
43     "rightline": {
44         "type": "string"
45     },
46     "microphone": {
47         "type": "string"
48     },
49     "maximum": {
50         "type": "string"
51     },
52     "microphoneConnected": {
53         "type": "string"
54     },
55     "running": {
56         "type": "string",
57         "format": "integer"
58     },
59     "attenuation": {
60         "type": "integer"
61     },
62     "micSpectrum": {
63         "type": "array",
64         "items": {
65             "type": "number"
66         }
67     },
68     "leftSpectrum": {
69         "type": "array",
70         "items": {
71             "type": "number"
72         }
73     },
74     "rightSpectrum": {
75         "type": "array",
76         "items": {
77             "type": "number"
78         }
79 }
```

```

80     }
81   }
82 }
83 }
```

Listado C.3 – Esquema JSON devuelto por `getStatus` en la versión LM7.

```

1  {
2    "definitions": {
3      "data": {
4        "type": "object",
5        "properties": {
6          "time": {
7            "type": "string",
8            "format": "date-time"
9          },
10         "attenuation": {
11           "type": "number"
12         },
13         "isActive": {
14           "type": "boolean"
15         },
16         "running": {
17           "type": "boolean"
18         },
19         "microphoneConnected": {
20           "type": "boolean"
21         },
22         "maximum": {
23           "type": "number"
24         },
25         "serialNumber": {
26           "type": "string"
27         },
28         "version": {
29           "type": "string"
30         },
31         "establishment": {
32           "type": "string"
33         },
34         "address": {
35           "type": "string"
36         },
37         "council": {
38           "type": "string"
39         },
40         "responsible": {
41           "type": "string"
42         },
43         "phonenumbers": {
44           "type": "string"
45         },
46         "configTime": {
47           "type": "string"
48         },
49         "vendor": {
50           "type": "string"
51         },
52         "deviceType": {
53           "type": "string"
54         },
55         "noiseAttenuation": {
56       }
```

References

```
56         "type": "number"
57     },
58     "noiseIsActive": {
59         "type": "boolean"
60     },
61     "microphone": {
62         "type": "number"
63     },
64     "leftline": {
65         "type": "number"
66     },
67     "rightline": {
68         "type": "number"
69     },
70     "mic": {
71         "$ref": "#/definitions/spectrum"
72     },
73     "left": {
74         "$ref": "#/definitions/spectrum"
75     },
76     "right": {
77         "$ref": "#/definitions/spectrum"
78     }
79   },
80 },
81 "spectrum": {
82   "type": "object",
83   "properties": {
84     "energy": {
85       "type": "array",
86       "items": {
87         "type": "number"
88       }
89     },
90     "dB": {
91       "type": "array",
92       "items": {
93         "type": "number"
94       }
95     },
96     "aWeighted": {
97       "type": "array",
98       "items": {
99         "type": "number"
100      }
101    },
102    "globaldB": {
103      "type": "number"
104    },
105    "dBA": {
106      "type": "number"
107    }
108  }
109},
110}
111}
```

Listado C.4 – Esquema JSON devuelto por `getStatus` en las versiones LM9 y LM11.

C.1.3 getData

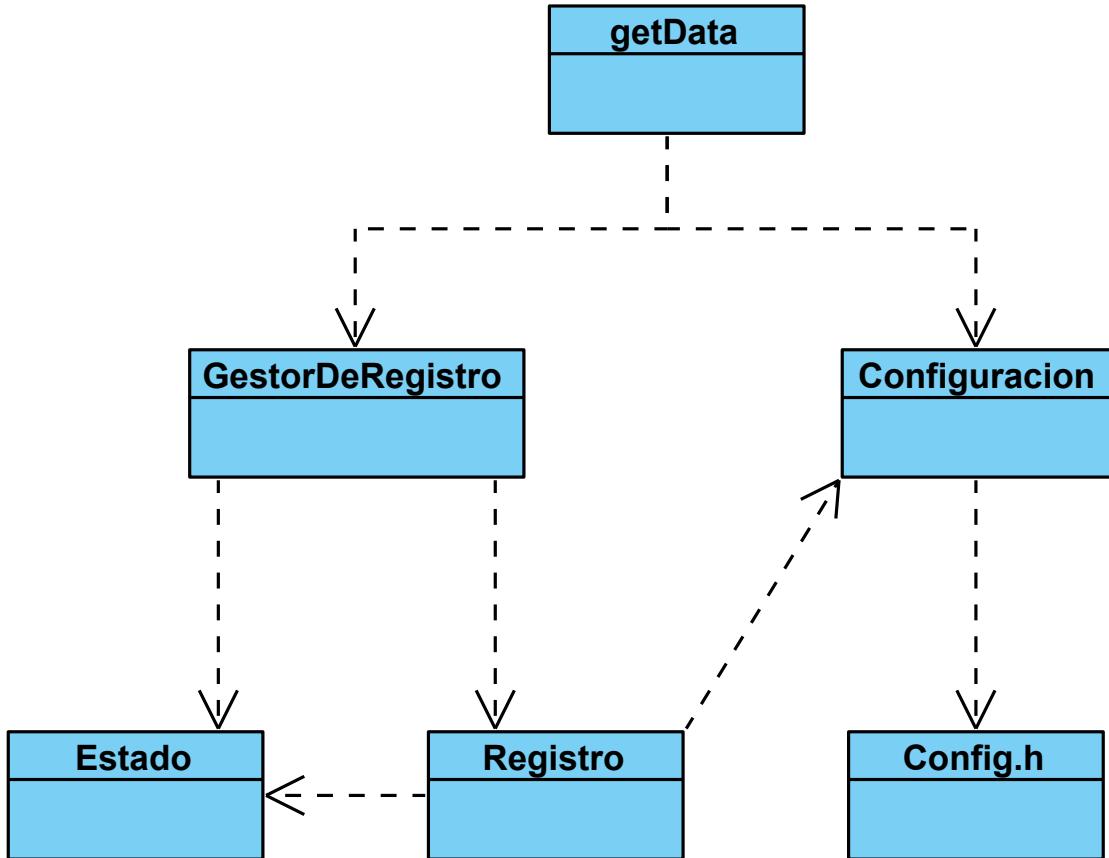


Figura C.3 – Diagrama de clases del programa *getData*.

El programa *getData* se encarga de recuperar los registros almacenados por el limitador pertenecientes a un intervalo de tiempo determinado, devolviéndolos en formato JSON. *getData* recibe 4 parámetros y la sintaxis para invocarlo es la siguiente:

`./getData $formato $fechaIni $fechaFin $intervalo`

donde:

- Formato: [XML](#) o [JSON](#)
- Fecha de inicio: desde la fecha base del registro hasta la fecha actual.
- Fecha de fin: desde la fecha de inicio hasta la fecha actual.
- Intervalo de muestreo: intervalo de tiempo entre cada registro, en minutos.

La fechas van en el formato `YYYY/MM/DD-hh:mm`, siguiendo el estándar [ISO 8601](#).

References

Los listados C.5 y C.6 muestran los esquemas JSON devueltos por este programa en cada una de las versiones.

```
1  {
2      "definitions": {
3          "data": {
4              "type": "object",
5              "properties": {
6                  "registries": {
7                      "type": "array",
8                      "items": {
9                          "$ref": "#/definitions/Registry"
10                     }
11                 }
12             }
13         },
14         "Registry": {
15             "type": "object",
16             "properties": {
17                 "time": {
18                     "type": "string"
19                 },
20                 "mic": {
21                     "type": "number"
22                 },
23                 "leftline": {
24                     "type": "number"
25                 },
26                 "rightline": {
27                     "type": "number"
28                 },
29                 "attenuation": {
30                     "type": "integer"
31                 },
32                 "maximum": {
33                     "type": "number"
34                 },
35                 "disconnectedMicrophone": {
36                     "type": "string",
37                     "format": "integer"
38                 }
39             }
40         }
41     }
42 }
```

Listado C.5 – Esquema JSON devuelto por `getData` en la versión LM7.

```
1  {
2      "definitions": {
3          "data": {
4              "type": "object",
5              "properties": {
6                  "registries": {
7                      "type": "array",
8                      "items": {
9                          "$ref": "#/definitions/Registry"
10                     }
11                 }
12             }
13         },
14     }
```

```

14 "Registry": {
15   "type": "object",
16   "properties": {
17     "time": {
18       "type": "string"
19     },
20     "mic": {
21       "type": "number"
22     },
23     "leftline": {
24       "type": "number"
25     },
26     "rightline": {
27       "type": "number"
28     },
29     "attenuation": {
30       "type": "integer"
31     },
32     "maximum": {
33       "type": "number"
34     },
35     "disconnectedMicrophone": {
36       "type": "string",
37       "format": "integer"
38     },
39     "microphone": {
40       "type": "array",
41       "items": {
42         "type": "number"
43       }
44     },
45     "left": {
46       "type": "array",
47       "items": {
48         "type": "number"
49       }
50     },
51     "right": {
52       "type": "array",
53       "items": {
54         "type": "number"
55       }
56     }
57   }
58 }
59 }
60 }
```

Listado C.6 – Esquema JSON devuelto por `getData` en las versiones LM9 y LM11.

C.2 Módulos legacy

Estos programas no han sido seleccionados para formar parte del nuevo software, bien por su excesiva complejidad, funcionalidad duplicada o no requerida, o mezcla de ambas cosas.

C.2.1 limInfo

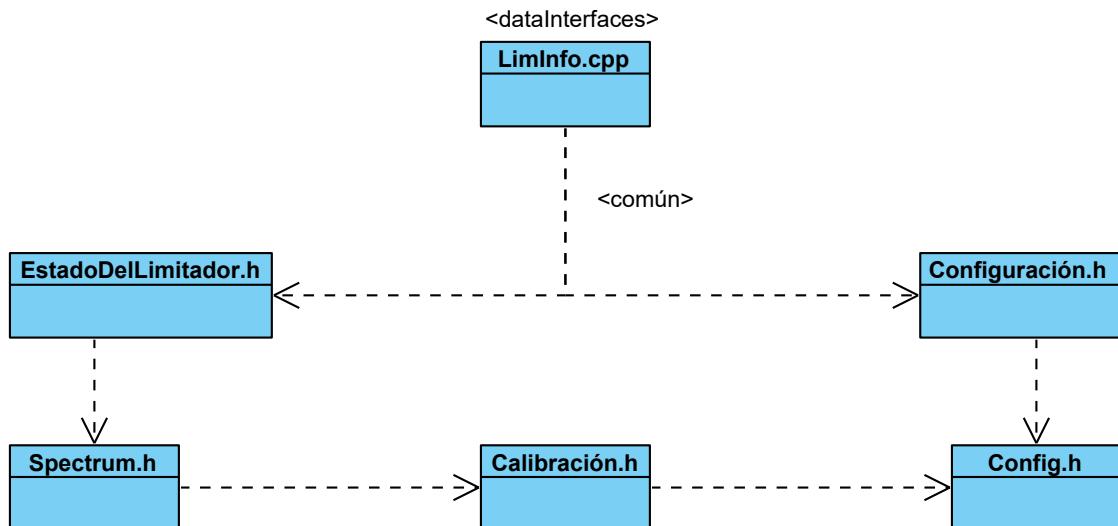


Figura C.4 – Diagrama de dependencias del programa limInfo.

El programa recibe como parámetro una cadena de caracteres, en la que cada carácter corresponde a la consulta de una propiedad del limitador. En el listado de códigos de este anexo se muestra la información que exporta el programa `limInfo` y el carácter necesario para su consulta.

En la figura C.4 se muestra el diagrama de dependencias del programa. Este diagrama representa la inclusión de otros ficheros de código fuente C++ en el fichero `limInfo.cpp`. Estas inclusiones no siempre son clases, por lo que **no debe confundirse con un diagrama de clases**. En las relaciones se muestra entre ángulos el módulo al que pertenece (la carpeta en la que se encuentra dentro del proyecto).

Este programa se consideró redundante ya que se puede obtener la misma información mediante el `getConfig`, por lo que no ha sido exportado ni utilizado por nuestro sistema.

Leyenda:

- El símbolo igual (=) indica cuál es el valor por defecto de la propiedad a la que acompaña.

- La flecha hacia la derecha (\rightarrow) da comienzo a un comentario sobre la propiedad que la precede.
- La flecha en ambos sentidos (\leftrightarrow) indica que esa propiedad está duplicada.

Códigos:

- A: configuración.atenuaciónMáxima = 90
- I: configuración.normativa.intervalo.horaInicio
- F: configuración.normativa.intervalo.horaFin
- D: configuración.normativa.máximoDiurno
- N: configuración.normativa.máximoNocturno
- {: configuración.normativa.máximoRecepciónNocturno
- }: configuración.normativa.máximoRecepciónDiurno
- T: configuración.tiempoBajoMáximo = 5 \rightarrow cada cuánto reduce la atenuación si no hay subidas.
- P: configuración.pasos = 1 \rightarrow pasos de atenuación por cada segundo que se sobrepasa el máximo.
- E: configuración.penalización
- S: configuración.partesDeCompresor
- G: configuración.rangoDePenalización = 120
- I: configuración.activo
- @: configuración.local
- Q: configuración.medidasPorCiclo = 2
- C: configuración.tipoDeControl \rightarrow 0: Mic, 1: Líneas, 2: Mixto
- R: estado.recepción
- l: estado.presiónIzquierda
- L: estado.presiónDerecha
- n: configuración.númeroDeSerie

References

- v: configuración.versión
- f: enFuncionamiento
- p: estado.presión
- a: estado.atenuación
- d: estado.micrófonoConectado
- m: estado.media
- M: configuración.máximoEnEmisión(ahora)
- h: estado.hora
- s: configuración.servidorDeDatos
- r: configuración.horaDeReprogramado
- u: configuración.válido = true → Siempre es true.
- 3: configuración.local ↔ @
- 4: número de serie.
- 5: configuración.dirección
- 6: configuración.teléfono
- 7: configuración.persona
- 8: configuración.ayuntamiento
- 9: configuración.distribuidor
- ..: configuración.predictivo
- ?: configuración.aislamiento
- X: número de serie + versión + presión + atenuación + media + máximo + hora + enFuncionamiento + horaReprogramado + atenuación máxima

C.2.2 localService

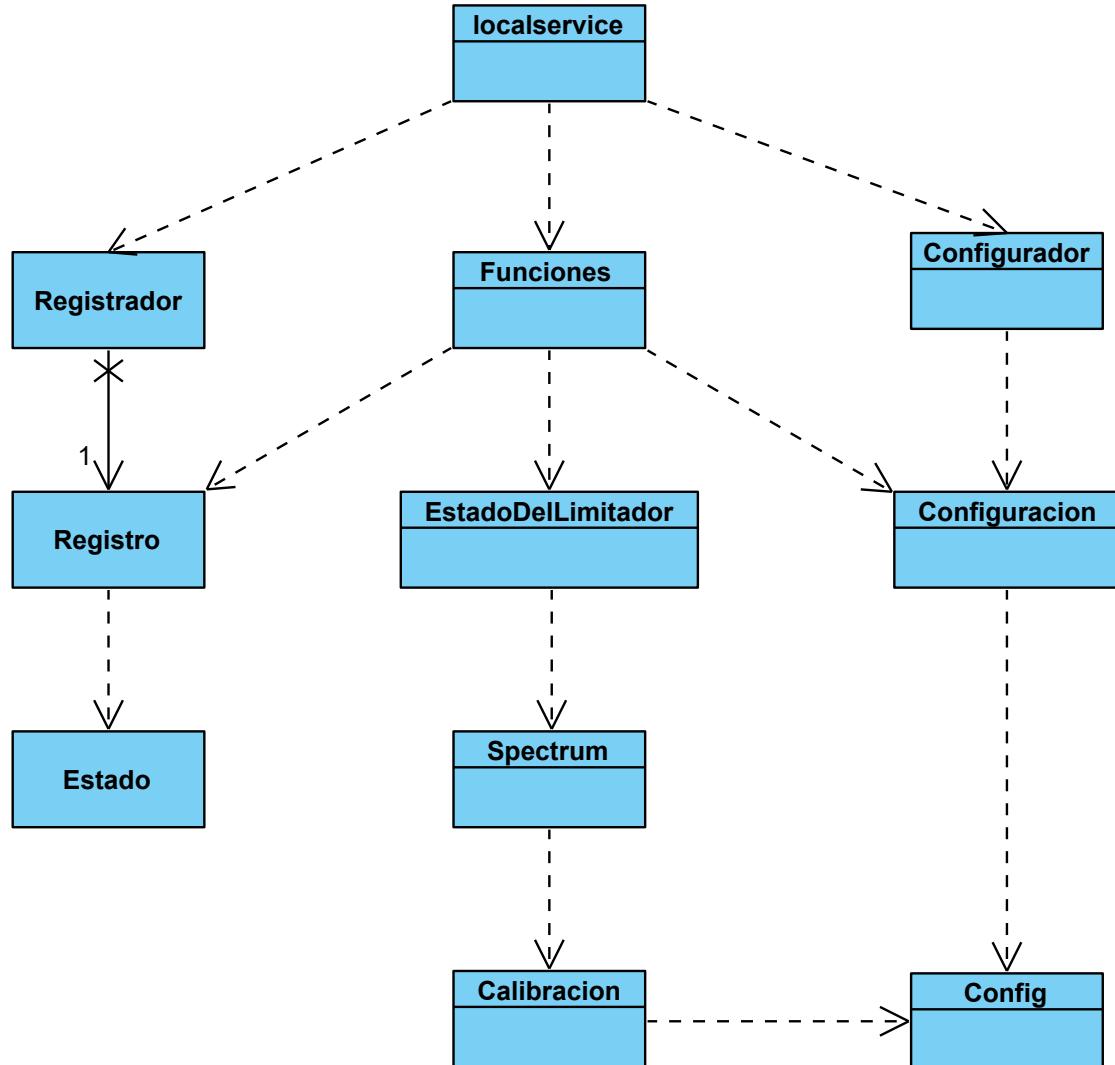


Figura C.5 – Diagrama de dependencias del programa localservice.

Al igual que `utilslr` recibe una serie de comandos como parámetro para modificar y consultar los datos del limitador. Los datos se devuelven **sin formato** útil (no JSON). El programa resulta excesivamente complejo, ya que contiene más del 1000 líneas de código, y ofrece funcionalidades que ya realizan otros de los programas auxiliares. Este programa pretende funcionar como un intérprete de comandos o una consola remota. Puede abrir y mantener un socket HTTP si se compila con las banderas requeridas.

Aunque el programa se llama `localservice`, la clase principal se llama `ServidorSerie`.

A continuación se listan los comandos que puede recibir este programa. Aquellos resaltados en negrita representan funcionalidad útil que ha sido exportada al LM11. Es

References

fácil observar que mucha de las funcionalidades que ofrece este programa también la ofrecen otros programas auxiliares mucho menos complejos que este, y no solo eso, sino que mismamente dentro programa hay funcionalidad duplicada.

Comandos vía argumentos:

- **calibrolineas**: calibra el micrófono y las líneas.
- **miceq**: devuelve la ecualización del micrófono.
- **lefteq**: devuelve la ecualización de la línea izquierda.
- **righteq**: devuelve la ecualización de la línea derecha.
- **mic \$float**: re-calibra el micrófono con el valor recibido.
- **lefteq \$float[8]**: actualiza la ecualización de la línea izquierda con los valores recibidos.

Los valores se dan separados por comas.

- **righteq \$float[8]**: actualiza la ecualización de la línea derecha con los valores recibidos.

Los valores se dan separados por comas.

Comandos vía intérprete:

- **datos**: permite modificar los datos del local.
- **update**: descarga y aplica la actualización dada una URL.
- **configuración**: muestra la configuración.
- **configura**: permite modificar parte de la configuración.
- **actva**: activa o desactiva el sistema.
- **concha**: abre conexión a telnet en el puerto 1036. Promociona la conexión.
- **concharemota**: abre conexión al terminal remoto en sheel.boanergesnetwork.com con la utilidad nc.
- **identifica**: login.
- **calibra**: calibra un sensor manualmente.
- **prepara**:

- Muestra la configuración.
 - Permite borrar la configuración.
 - Permite cambiar datos del local.
 - Permite inicializar el registro.
 - Permite cambiar parte de la configuración.
- operativo:
 - Muestra la configuración.
 - Permite cambiar algunos datos de la configuración.
 - **test:** calibra las líneas.

C.2.3 utilslr

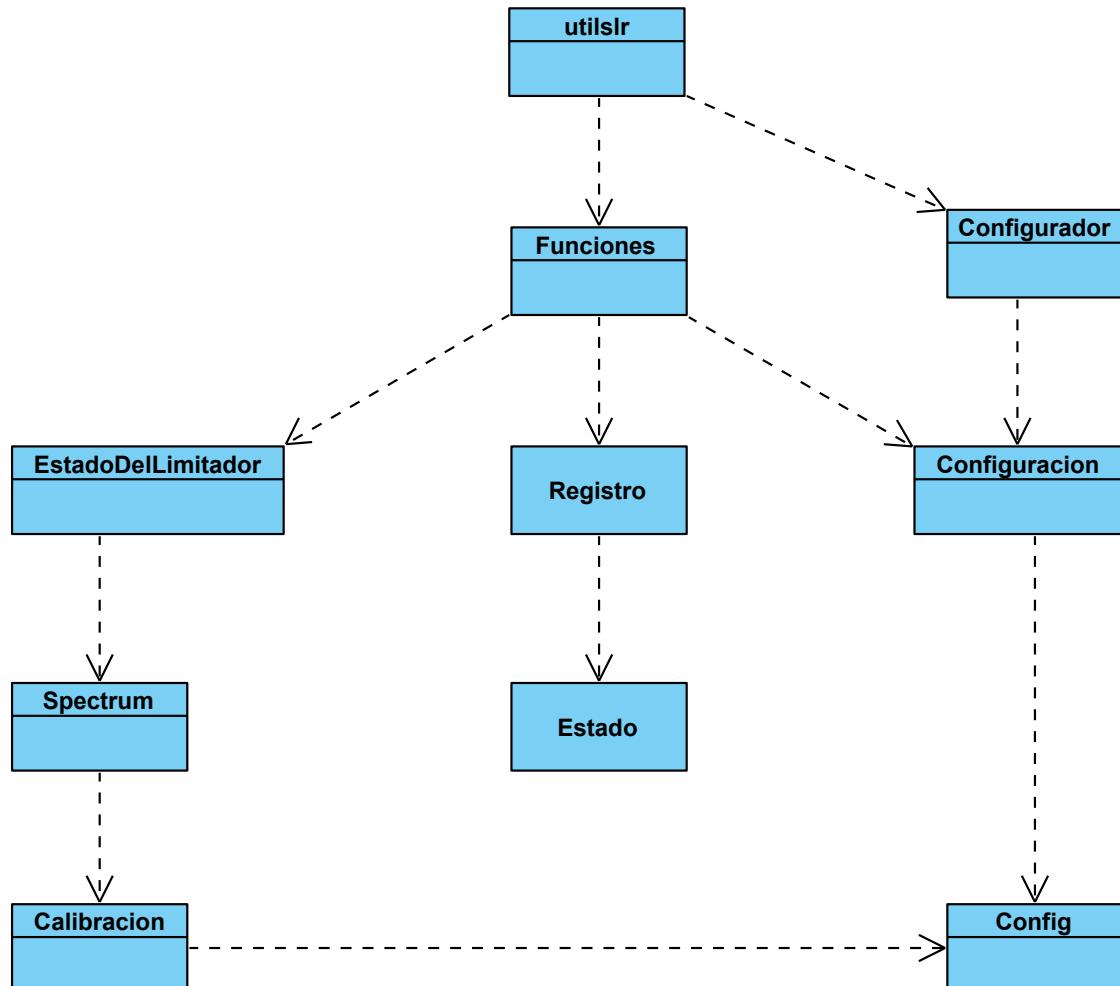


Figura C.6 – Diagrama de dependencias del programa utilslr.

El programa utilslr recibe serie de comandos como parámetro, principalmente para modificar la configuración del equipo (con `utilslr config`), pero también para consultar información e incluso generar informes en varios formatos (aunque tras probarlos la mayoría no funcionan).

El programa utilslr se usa de forma intensiva en la versión LM7, por parte de la aplicación web. Generalmente se usa para aplicar la configuración y comprobar las contraseñas de usuario.

A continuación se listan los comandos que puede recibir este programa. Aquellos resaltados en negrita representan funcionalidad útil que ha sido exportada al LM11.

- clave \$p: comprueba si la contraseña p es correcta.

- activaweb \$código \$nSerie \$nDistribuidor: activa la aplicación web.
- configuración: devuelve la configuración del equipo, igual que getConfig.
- **configura** \$fichero: aplica la configuración definida en el fichero indicado (siempre es /tmp/conf.tmp).
- {preauth, verifycu}: relacionados con verificación de usuarios nuevos. Carecen totalmente de interés para nuestro proyecto.
- {sql, xml, yml, json, sqlite}\$fechaIni \$fechaFin \$intervalo: genera volcados en el formato indicado. Solo funcionan sql, xml y json.
- {listado, gráfico, actividad}: sin usos conocidos.

References

Apéndice D

Ficheros del limitador

Todos los datos generados o requeridos, ya sean persistentes o temporales, son almacenados en ficheros. Cada uno de estos ficheros tiene un formato propio, que va desde ficheros en texto plano hasta ficheros binarios. En este anexo se describen los ficheros más importantes del limitador.

Los limitadores LM7 y LM9 almacenan los ficheros en los directorios del sistema Linux /var y /tmp, sin embargo, en nuestra versión (LM11) se ha creado un directorio con la finalidad de alojar todo lo relativo a nuestro software en un mismo lugar, pero manteniendo la misma estructura interna (bin/, var/, tmp/). Por tanto, cuando se haga referencia al path de un fichero, debe comprenderse que en la versión LM11 ese path hace referencia a los directorios internos de esta carpeta y no a los directorios raíz del sistema operativo.

D.1 El registro

Durante su funcionamiento, **el limitador genera un registro cada minuto**. El programa encargado de generarlos difiere según la versión, pero en todas ellas se almacenan en el fichero /var/slrl/registro.slr. Este fichero corresponde en las versiones LM7 y LM9 a un fichero especial de dispositivo de bloques creado mediante la utilidad mknod, quedando mapeado¹ a una partición del disco duro, mientras que en la versión LM11 corresponde a un fichero en formato binario.

El archivo de registros se compone de una cabecera, presente una sola vez al comienzo del fichero, y los registros en sí. El fichero de registro es mantenido por la clase Registrador, o GestorDeRegistro en la versión LM11, la cual proporciona utilidades para crear, guardar y leer registros. En la figura D.1 pueden consultarse los

¹Crea el fichero registro.slr como un enlace a la partición.

campos que componen la cabecera y cada uno de los registros.

Cabecera	Registro
-tipo : int -horaBase : time_t -cadena : char[30]	-atenuacion : float -crc : short -dB : float -hora_time_t -informe : char -max : float -pDerecha : float -plzquierda : float -pRecepcion : float -spectrum : char[3][31]

Figura D.1 – Atributos de las clases que componen el registro.

D.2 Ficheros de configuración

D.2.1 El fichero /tmp/conf.tmp

El fichero conf.tmp es un fichero temporal en el que se escriben las nuevas configuraciones a la espera de ser aplicadas por el configurador. El ciclo de vida de este fichero se describe en el diagrama de secuencia de la figura D.2. En resumen, el instalador configura el equipo mediante una aplicación, la cual se encarga de escribir el fichero en disco con un formato predeterminado para luego notificar al *configurador*, quien lee y aplica la nueva configuración. La aplicación puede ser web, como en el caso del LM7, o de escritorio, como en el caso de nuestro sistema (LM11). En este último caso, la aplicación no escribe los datos directamente en disco ya que es un sistema externo al limitador, en su lugar, se pasa por la API del limitador, enviándole los datos de la nueva configuración. La API comprueba y escribe los datos en disco, para finalmente dar paso al *configurador*.

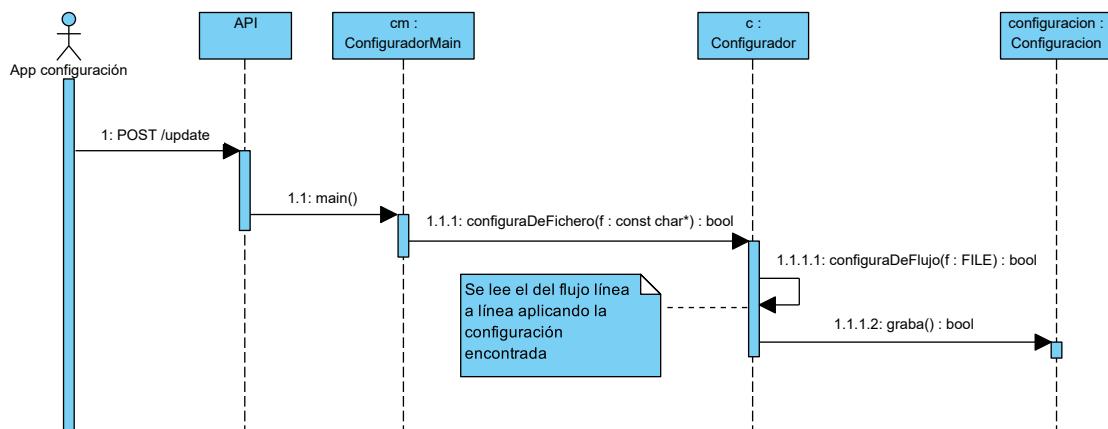


Figura D.2 – Diagrama de secuencia para actualizar la configuración en el LM11.

En el listado inferior se muestra cada una de las propiedades configurables del limitador y su formato en el fichero conf.tmp. Cada una de las propiedades debe ocupar una y solo una línea en el fichero, y no es obligatorio configurarlas todas, ya que el configurador solo actualiza las propiedades que encuentra en el fichero.

Nota: las falta de tildes en los nombres de las propiedades es intencionada.

- Identificacion \$char[50]
- Version \$char[50]
- Activacion \$bool
- Desactivacion \$bool

References

- Local \$char[50]
- Direccion \$char[255]
- Telefono \$char[50]
- Persona \$char[50]
- Ayuntamiento \$char[50]
- Distribuidor \$char[50]
- BloqueoDeAudio \$bool
- Predictivo \$bool
- Laeq \$int
- Tiempo \$int
- Atenuación \$float
- Pasos \$int
- AtPorBanda \$float
- Password \$char[50]
- Control \$int ²
- RangoDePenalización \$float
- PartesDeCompresor \$int
- atenuacionMinima \$float
- coeficienteDeThreshold \$float
- Norma \$hI \$hF \$mD \$mN \$mRD \$mRN ³
- Aislamiento \$float[n] ^{4 5}
- Margenes \$float[n] ^{6 5}
- UsarControlDeRecepcion \$bool

²0: mic, 1: línea, 2: mixto

³horaIni, horaFin, maxDiurno, maxNocturno, maxRepcionDiruno, maxRepcionNocturno

⁴n = nº de bandas de frecuencia. LM7: 8, LM9 y LM11: 31

⁵Array separado or comas, sin espacios.

⁶Solo en LM9.

- PinkNoiseLevel \$float
- SessionStartThreshold \$float
- SessionEndThreshold \$float
- UsarControlFrecuancial \$bool
- ControlarCadaFrecuencia \$bool
- UsarNormativaExtendidad \$bool
- LimiteSemanalDiurno \$float[7] ⁵
- LimiteSemanalNocturno \$float[7] ⁵
- LimiteSemanalRepcionDiurno \$float[7] ⁵
- LimiteSemanalRepcionNocturno \$float[7] ⁵
- FestivosClear
- Festivo \$tipo(int)⁷ \$día(int) \$numDía(int) \$mes(int) \$horaIni(int) \$horaFin(int) \$minIni(int) \$minDuración(int) \$maxEmisión(float) \$maxRecepción(float)
- equipoInstalado \$char[9999]
- otrosValores \$char[4096]

Durante la implementación del LM11 se modificaron algunos de los nombres por otros más significativos:

- Laeq → MedidasPorCiclo
- Atenuacion → AtenuacionMaxima
- Tiempo → TiempoBajoMaximo
- atenuacionMinima → AtenuacionMinima

También se añadieron propiedades nuevas que eran necesarias poder configurar:

- | | |
|--|---|
| <ul style="list-style-type: none"> • SessionStartDelay \$int • SessionEndDelay \$int | <ul style="list-style-type: none"> • Licencia \$char[150] • Expiración \$time_t |
|--|---|

⁷0: desactivado, 1: activado, 2: primer lunes del mes

References

- Sala \$char[150]
- Fuentes \$char[150]
- Altavoces \$char[150]
- Etapas \$char[150]
- Crossovers \$char[150]
- Ecualizadores \$char[150]
- Mesas \$char[150]

Los siguientes datos estaban disponibles para su configuración en las versiones LM7 y LM9, pero se eliminaron en la LM11.

- Boanerges.Server
- Boanerges.Port
- Boanerges.IncomingMessagesEncryptionKey
- Boanerges.OutcomingMessagesEncryptionKey
- Boanerges.Secret
- Boanerges.ServerRevision
- equipoInstalado
- otrosValores

Apéndice E

Plantilla para la definición de un servicio en Linux

```
1 #!/bin/bash
2 # chkconfig: 2345 20 80
3 # description: Description comes here....
4
5 # Source function library.
6 . /etc/init.d/functions
7
8 start() {
9     # code to start app comes here
10    # example: daemon program_name &
11 }
12
13 stop() {
14     # code to stop app comes here
15    # example: killproc program_name
16 }
17
18 case "$1" in
19     start)
20         start
21         ;;
22     stop)
23         stop
24         ;;
25     restart)
26         stop
27         start
28         ;;
29     status)
30         # code to check status of app comes here
31         # example: status program_name
32         ;;
33     *)
34         echo "Usage: $0 {start|stop|status|restart}"
35 esac
36
37 exit 0
```

References

Apéndice F

LM7: documentación del código fuente

Directorio	Archivo	Tipo	Descripción	Notas de interés
/	configuracion	data : binario	[No se puede abrir]	
	connectNetwork	script bash	Inicializa wvdial	wvdial /var/slr/3g.wvdial.conf /var/slr/network.script
	construye	script bash	Ejecuta los make de cada directorio (módulos) en cascada Compila e instala el sistema.	Los archivos de /var/slr están en Vol50MB
	Makefile	Makefile	Para la compilación se llama a ./construye Para la instalación, se copian los ejecutables recién compilados a /bin <ul style="list-style-type: none"> - Remonta el sistema de archivos en modo lectura-escritura. - Ejecuta el Makefile - Sincroniza el sistema de archivos con sync - Pone el sistema en hora con ntpdate - Ejecuta utilies/initializador - Comprueba si /ms se ha marcado para borrar, en tal caso, elimina la carpeta y todos sus archivos. - Sincroniza el sistema de archivos con sync - Reinicia el equipo con init 6 Resetea el sistema al estado de fábrica.	mount -n -o remount, rw / init 6 sync
	preparatoria	script sh		
	setAsNew	script bash	<ul style="list-style-type: none"> - Ejecuta utilies/initializador - Limpia los logs en [/var/slr/logs.serial] - Resetea el fichero de datos `configuracion` y `configuracion.default` - Limpia el registro de reinicios del sistema (./restarts) - Elimina calibracionReference - Elimina firstLogin 	
comun/				https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
	aes.cpp	cpp	Anesma, an AES encryption library for C++	
	aes.h	h	Cabeceras del algoritmo de encriptado AES	Tanto el .h como el .cpp son código externo (open source) por lo que está debidamente documentado.
	Calibracion.cpp	cpp		Controla la calibración del limitador. Controla los 3 propiedades: <ul style="list-style-type: none"> -referencia: por defecto 0. Se inicializa como ref², siendo ref el valor pasado en el constructor. Se usa para calcular los dB. -ganancia: no se actúa sobre el, solo se inicializa en el constructor y se devuelve en el getter -db_en_referencia: por defecto 0. Se usa para calcular los dB.
	Calibracion.h	h	Definición de la clase Calibración. Define los métodos y atributos de clase.	Incluye: <ul style="list-style-type: none"> - Config.h

Directorio	Archivo	Tipo	Descripción	Notas de interés
compact.cc	cc		Contiene una serie de funciones las cuales manejan cadenas de texto. La finalidad de las funciones es encriptar cadenas de texto. La cadena encriptada es el doble de grande que la cadena actual. Para cada carácter de la cadena original, se generan dos nuevos caracteres correspondientes a los bits más y menos significativos de la representación en código ASCII del carácter.	Incluye: - Configuracion.h - EstadoDelLimitador.h
Importante	Config.h	h	<p>Ver código adjunto</p> <p>Contiene exclusivamente definiciones (defines). Es por tanto un almacén de constantes globales y valores por defecto que se usan a lo largo del código del limitador.</p> <p>Implementación de algunos de los métodos de las clases definidas en el.h</p> <p>Unos cuantos métodos tienen el comentario: "Cambios por la ampliación de la normativa semanal y festivos"</p> <p>Todos estos métodos tienen la finalidad de gestionar la configuración por normativa extendida, es decir, la configuración según día de la semana, festivo u hora del día.</p>	Incluye: sys/soundcard.h PuertoDeEstado: 9099 ? GananciaPorDefecto: 50 DSD , CCC PuertoDeConexion: 9182 ?

Directorio	Archivo	Tipo	Descripción	Notas de interés
ABOMINACIÓN!	Configuracion.h	h	<p>Controla todo lo que es la configuración del limitador, como los parámetros de configuración, datos del local, del distribuidos, del servidor, incluyendo la configuración específica por franja horaria, día de la semana, festivos y demás (la normativa).</p> <p>Contiene varias clases e incluso la implementación de algunos de sus métodos en el mismo archivo de cabecera (de ahí lo de abominación). En concreto, este archivo define las siguientes clases:</p> <ul style="list-style-type: none"> - Intervalo - Normativa - Festivo - Configuración <p>E incluye la implementación de los métodos de las 3 primeras.</p> <ul style="list-style-type: none"> - Intervalo: <ul style="list-style-type: none"> - Atributos: horaInicio y horaFin - Métodos: un constructor y un método dentro() que comprueba si la hora dada está dentro de ese intervalo (hora_inicio < hora < hora_fin) - Normativa: <ul style="list-style-type: none"> - Atributos: un Intervalo Y 4 floats: maximoNocturno, maximoDiurno, máximoRecepcionNocturno, máximoRecepcionDiurno - Métodos: dos constructores, uno por defecto y otro con argumentos. Por defecto el intervalo es de 8 a 22h, con un maximo diurno y nocturno de 95dB. 2 funciones que devuelven el maximo y maximoRecepcion dependiendo del horario (intervalo dentro(h)) - Festivo: es la clase más extensa <ul style="list-style-type: none"> - En general, contiene los atributos propios de un día (todos int o float), hora de inicio, horas de duración y el maximo de emisión y recepción para ese día festivo. Los métodos manejan fechas y horas del tipo struct tm y time_t <p>Implementación de la clase.</p> <ul style="list-style-type: none"> - limpiaCadena: reemplaza todos los '\n' por espacios. - escribeConfiguración: escribe la Configuración en el archivo especificado como argumento. - configuraDefFichero: abre el fichero dado en modo lectura. - configuraDefFlujo: invocado desde configuraDefFichero, lee el fichero en trozos de 200 caracteres, y va actualizando los atributos del objeto Configuración según los va encontrando en el fichero. Finalmente, actualiza la Configuración con el método grabar() de su clase. 	<p>Incluye:</p> <ul style="list-style-type: none"> - Configuración.h - Configuración.cpp
	Configurador.h	h	Definición de la clase con mismo nombre. Solo tiene métodos.	<p>Incluye:</p> <ul style="list-style-type: none"> - Configurador.h - Configuración.h

Directorio	Archivo	Tipo	Descripción	Notas de interés
			Implementación de la clase.	
EstadoDelLimitador.cpp	cpp		<p>Solo existe un método: <code>actualiza()</code>, el cual abre el archivo de estado en modo lectura y comprueba si el estado ha cambiado o no. No modifica ningún atributo de la clase, por lo que no queda claro porque se ha nombrado como "actualiza".</p> <p>El método devuelve <code>bool</code>, por lo que se intuye que el método devuelve si el estado debe ser actualizado o no, pero no se sabe quién se encarga de dicha actualización (debería ser la propia clase la que cambie su estado...)</p> <p>Definición de la clase con el mismo nombre. Define las propiedades y el estado del limitador como atributos:</p> <ul style="list-style-type: none"> - Número de serie - Versión - Penalización, atenuación y atenuación global - Presión, presión media, presión izquierda y presión derecha - Hora del día. - Nivel de recepción y nivel máximo. - Micrófono conectado 	Incluye: <ul style="list-style-type: none"> - Config.h - EstadoDelLimitador.h
EstadoDelLimitador.h	h			Se almacena en el fichero -> /tmp/estado.slr
EstadoDeModificadores.cpp	cpp		<p>Implementación de la clase.</p> <p>5 métodos: constructor (ambos atributos a false), getters, guarda y lee.</p>	Incluye: <ul style="list-style-type: none"> - Config.h - EstadoDeModificadores.h
EstadoDeModificadores.h	h		<p>Definición de la clase con mismo nombre.</p> <p>Dos atributos de tipo bool: inhibidor y microfono</p> <p>Funciones varias, algunas bastante interesantes para exportar el estado completo del limitador a distintos formatos.</p>	F_Llave -> /tmp/slr.k
Funciones.cpp	cpp		<ul style="list-style-type: none"> - volcado - volcadoJSON - volcadoXML - volcadoYML - volcadosSQL (2) - volcadoSQLite - imprime - generaGráfico - generaInforme - monta - desmonta - configura - configuraDefFichero - limpia - guarda - informeEncendido - informeDeConfiguracion - generador 	Ruta de archivos: /tmp

Directorio	Archivo	Tipo	Descripción	Notas de interés
Funciones.h	h		Definición de la clase con mismo nombre. Encapsula una serie de funciones de uso general, ya que manipula o usa objetos del tipo Configuración EstadoDelLimitador y Registro	Incluye: - Configuración.h - Limitador/Registro.h - EstadoDelLimitador.h - Config.h
getSerial.cpp	cpp		Parece que solo es un fichero de prueba para testear Serial.cpp	Incluye: - Serial.cpp
gSerial	ejecutable		Usa las funciones getID, testID y printDefinesSerial de Serial.cpp Obtiene el ID (HWaddr de eth) y lo escribe como un define en Serial.h	
Makefile	Makefile		Construye el ejecutable gSerial a partir de getSerial.cpp, luego lo ejecuta y redirige su salida a Serial.h	
			Varias funciones realizacionadas con el serial (ID) del equipo. El ID se presupone único para cada equipo.	
			- getID: genera el ID a partir de los dos fd (ver siguiente celda), los concatena, (fd2 + fd1) y da la vuelta al string resultante, para finalmente devolverlo en id (puntero a char dado como parámetro) - testID: comprueba si el ID es igual al ID dado como parámetro - strip: *creo * que elimina los espacios en blanco de la cadena de texto dada como parametro - getHSerial: consulta y devuelve la identidad del controlador IDE maestro del sistema y devuelve los primeros 20 caracteres en out (puntero a char pasado como parámetro en la llamada a la función). - printDefinesSerial: imprime un define para el ID (#define id <i0>) - maskString: recibe in y out, realiza una encriptación sobre cada carácter de in y lo almacena en out	El ID es generado a partir de los siguientes file descriptors fd1 -> ifconfig grep eth grep HWaddr fd2 -> /dev/hda ioctl, strndup
Serial.cpp	cpp		Vacio. gSerial redirige su salida a este archivo. Esto ocurre al ejecutarse el Makefile de este directorio.	
serial.h	h			
TestConfig.cpp	cpp		Fichero de prueba de Configuración.cpp	
comunicacion.old/				
	cpp			
	old			
	a.out			
	cliente.c			
	funciones_Conexion.cpp			
	funciones_envioDeDatos.cpp			
	funciones_formato.cpp			
	Funciones.h			
	funciones_hora.cpp			
	globales.h			
	Main.cpp			
	Makefile			
	sirComm			
	Test.cpp			
	TestRegistrador.cpp			

Directorio	Archivo	Tipo	Descripción	Notas de interés
comv2/				
	base64.cpp	directorio	Contiene los ficheros de un proyecto que implementa un Socket en C++. Contiene el fichero de cabecera para el socket y otra para sus excepciones, la implementación del servidor, la implementación del cliente, una demo, y un Makefile para compilar el proyecto.	
old	base64.c	directorio	Funciones para codificar y decodificar char a base64 Se conecta a un servidor remoto en un puerto determinado . Si no puede devuelve -1	
	cliente.c	c		
	crypter.cpp	cpp	Contiene varias funciones para encriptar y desencriptar strings	Incluye: - funciones.h - cliente.c - crypter.cpp - WhiteRabbit.h - comm2/cpp/Socket.{h, .cpp} - comun/serial.h
	funciones_Conexion.cpp	cpp	Implementación de la mayoría de las funciones definidas en funciones.h, salvo las de envío de datos que se implementan en funciones_envioDeDatos.cpp.	
	funciones_envioDeDatos.cpp	cpp	contiene 2 funciones para enviar información al servidor (data). boanergesnetwork.com). Una se encarga de enviar la configuración del sistema, la otra se encarga de enviar registros (logs) del limitador	incluir funciones.h globales.h sprintf(char *str, char *format, *opts) Formatea el string str siguiendo el formato dado en format, como se haría con printf, pero en lugar de imprimir el resultado, lo almacena (machaca) en str.
	funciones_formato.cpp	cpp	Funciones para generar cadenas de texto en un formato predefinido. Las cadenas de texto van a ser posteriormente enviadas a un servidor, el cual espera estos formatos.	incluir funciones.h Destacable el hecho de que las funciones están bastante bien comentadas. Se indica la descripción de la función, parámetros que recibe, condiciones y valor devuelto. Además, en la mayoría de los casos se indica también el fichero en el que se encuentra la implementación.
	Funciones.h	h	Definición de funciones relacionadas con el envío / recepción de mensajes al servidor configurado (imagine que es el servidor del Ayuntamiento).	
	funciones_hora.cpp	cpp	Contiene funciones para manejar la hora y fecha en varios tipos de estructuras de datos (tm, time_t, char). Las funciones reciben la hora en formato "yy/mm/dd hh:mm:ss". mktime, sscanf, strftime incluir funciones.h	Contiene una función 'actualizarHora(char *hora)', la cual intenta actualizar la hora del sistema a la hora recibida como parámetro, sin embargo, el código para actualizar la hora del sistema está comentado y la función solo imprime por pantalla que no se cambiará la hora del sistema por ser peligroso.

Directorio	Archivo	Tipo	Descripción	Notas de interés
	globales.h	h	Definición de funciones y macros.	#define FicheroDeMarcasSinConexion /tmp/nonAvailableConnection
	Main.cpp	cpp	Código principal (main) del programa ejecutable srlComm. Se comunica con un servidor en la dirección data.boanergesnetwork.com:3002 , al cual envía información. Este programa realiza varias funciones, pero principalmente intenta pasar por un shell remoto. Puede recibir acciones como getConfig, getEvents, getRegistry, etc. Ejecuta la acción requerida y envía el resultado de la operación.	Hay definidas 2 macros de configuración del sistema que indican la frecuencia con la que el sistema actualiza la hora (cada 5 días, si no se reinicia antes) y la configuración (re-carga de archivos de configuración imaginio; cada hora)
	slrComm	ejecutable	Genera el ejecutable srlComm	
	Test.cpp	cpp	Prueba el módulo conectándose a localhost:8081	Incluye cliente.c ctime: convert time_t value to string
	TestRegistrador.cpp	cpp		Realiza un test sobre cada uno de los registros almacenados en el sistema. Al parecer existe un registro para que almacena el estado del sistema en cada instante de tiempo (todo el texto que inunda la pantalla son estos registros)
	WhiteRabbit.cpp	cpp	Implementación del algoritmo de encriptado Rabbit	
	WhiteRabbit.h	h	Definición de la clase WhiteRabbit. Rabbit es un algoritmo de encriptado	
	WhiteRabbit.o	o	-	
	WhiteRabbit_Original_AllInOneFile.cpp	cpp	Implementación del algoritmo de encriptado Rabbit. Definición e implementación.	
instalador/				
	fdiskCommands	???	???	
	Im70If.localCommands	???	???	
	MAKEDEV	script sh	"A very simple script that can be used to create some of the more common device nodes found in /dev."	https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html

Directorio	Archivo	Tipo	Descripción	Notas de interés
sdbinstall	script bash -dev -proc	script bash	Inicializa la estructura de directorios principal del sistema de archivos:	mkfs.ext3, fdisk, sed, mknod
sdclInstall	script bash -dev -proc	script bash	TODO Inicializa la estructura de directorios principal del sistema de archivos:	mkfs.ext3, fdisk, sed, mknod
limitador/	1 a.out	config file ejecutable	Archivo de configuración de alsaplayer	Incluye: - sys/soundcard.h - mixers
				Comentario: Para el control de varios ambientes esto debe cambiar por el dispositivo que haya en la configuración y deben existir tantos atenuadores como mezcladores. Otra opción es crear tantos atenuadores como mezcladores y asignar a cada canal su mezclador. Ej: /dev/dsp > /dev/mixer /dev/dsp1 > /dev/mixer1
Atenuador.cpp	AtenuadorMk163.cpp	cpp	Clase Atenuador. Contiene un file descriptor a cada mixer, los cuales vienen definidos en el archivo "mixers", y los abre en modo lectura-escritura. El método atenuación se encarga de calcular los valores necesarios y escribir (actualizar) los mixers, usando <u>MIXER_WRITE</u>	Un comentario en el código dice lo siguiente: "Ya posee los cambios necesarios para trabajar con el atenuador DS"
		cpp	La clase no se usa en ninguna parte.	Incluye: - PuertoParalelo.cpp
Este es el atenuador que se usa en el limitador	AtenuadorPga.cpp	cpp	Declaración de la clase AtenuadorPga.	Incluye: - comun/config.h - PuertoParalelo.cpp
	AtenuadorPga.h	h	Contiene varias constantes (defines) para identificar los pinos del puerto paralelo sobre los que comunicarse, varios float para controlar el nivel de atenuación y una instancia de PuertoParalelo, para poder leer/escribir sobre los pinos del puerto paralelo. Re-monta los módulos kernel de sonido: - snd_usb_audio - snd_cs5535audio	rmmod, modprobe

Directorio	Archivo	Tipo	Descripción	Notas de interés
	Calibracion.cpp	cpp	Implementación de la clase Calibración. El cálculo de los decibelios se realiza comparando la potencia leída con la potencia de referencia, según la fórmula que puede verse aquí . Definición de la clase Calibracion.	Datos de calibración a fichero F_KAL
	Calibracion.h	h	Atributos: - dbRef, ref, ruido, ganancia, equilizacion Métodos: - leerCalibracion - guardarCalibracion - calculaBs - calculaEnergia - print	Incluye: - ./comun/Config.h
	CalibrationTest.cpp	cpp	Incompleto. No se usa.	
	Estado.cpp	cpp	Implementación de los métodos de la clase Estado. Son todos setter y getters. Para la gestión de los informes se utiliza una máscara de bits, por lo que las operaciones sobre el atributo 'informe' se realizan utilizando operadores a nivel de bit .	Incluye: - Estado.h - ./Config.h
	Estado.h	h	Define los atributos para representar el estado del limitador, así como métodos para manejarlos. Lo más destacables son: - atenuación, penalización y máximo - presión, presionizquierda, presionDerecha y recepcion	
	filterTest.cpp	cpp	Despreciable	
	filtroA	directorio	Contiene filtroA.cpp, una implementación descargada de internet para el filtro IIR	http://www.winfitter.20m.com
	iir.cpp	cpp	Clase que implementa el filtro IIR. Atributos: - orden : int - estado : double - numerador : double - denominador : double	Fichero de datos: FilterBankFile
	LectorAlternativo.cpp	cpp	No se usa (solamente se usa en Sonometros.cpp pero está comentado)	Incluye: - octaveBandFilter.c

Directorio	Archivo	Tipo	Descripción	Notas de interés
	Lector.cpp	cpp	<p>Contiene un par de funciones al principio autogeneradas por el generador de código para filtros digitales mkfilter. La orden utilizada para la generación de dichas funciones están comentadas al comienzo del fichero.</p> <p>Clase Lector. Utiliza tipos IIR, qir y Lectura2C.</p> <p>Atributos:</p> <ul style="list-style-type: none"> -sd -buffer[muestrasAl leer * canales] -frec -nSig -canales -applyAWeight -IIR filtroizquierda, filtroDerecha <p>Métodos:</p> <ul style="list-style-type: none"> -aplicaFiltrado: por cada filtro, recorre el buffer y obtiene su media positiva. -inicializarFiltros: lee los filtros desde FilterBankFile -analiza: inicializa y aplica los filtros mediante los métodos qir_init y qir_filter de la clase qir -fijaPartesDelectura(int n): nSig = frec / n -reOpen(fichero, nCanales); abre /dev/dsp (ioctl) -aplicaCorrecciónLogico: obtiene la media de los valores de buffer[] y se la resta a cada uno de los valores de buffer[] -lectura(): Lectura2C -> devuelve una salida de tipo Lectura2C, de la cual solo se inicián los valores isMono y applyInternalCorrection (ambos de tipo lógico) dependiendo de si la señal es mono o no. Aplica los filtros a los canales usando aplicaFiltrado 	<p>https://github.com/sven337/mkfilter/blob/master/gencode.C</p> <p>dsp -> /dev/dsp</p>

Directorio	Archivo	Tipo	Descripción	Notas de interés
	Lectura2C.cpp	cpp	<p>Clase Lectura2C: lectura de 2 canales (izquierda-derecha). Realiza la función de sensores para las salidas, de forma que es capaz de medir, procesar y almacenar las señales de salida en sus dos canales.</p> <p>Define 3 vectores con parámetros para los filtros (FrecuenciaCentral, FiltroA, FiltroDeAjuste).</p> <p>Para almacenar el estado de los canales, utiliza las siguientes propiedades:</p> <ul style="list-style-type: none"> - energiail[8] : int - energiaDil[8] : int - dBl[8] : float - dBDi[8] : float - dBAl[8] : float - dBADi[8] : float <p>En cuanto a métodos, contiene los siguientes:</p> <ul style="list-style-type: none"> - sum(Lectura2C);: suma de dos Lectura2C - div(float x);: divide cada propiedad mencionada anteriormente entre x - zero();: pone todo a 0 constructor - print(): imprime por pantalla - valorGlobalEspectro(float dbs) : float --> calcula y devuelve el resultado de aplicar la formula que se ve a la derecha (decibelios ponderados) - globalizquierdoPonderado(): devuelve valorGlobalEspectro(dBAI) - globalderechoPonderado(): devuelve valorGlobalEspectro(dBAD) - globalizquierdoPonderadoDeRecepcion Y globalDerechoPonderadoDeRecepcion(): recibe un vector de float (aislamiento), y los resta a dBAI, dBAD (según el caso) si y solo si aislamiento[i]<3 - save(int n);: vuela la memoria de la instancia actual en el fichero /tmp/.lx[n] - read(int n);: aplica el vuelco de memoria del archivo /tmp/.lx[n] a la instancia actual 	<p>Incluir:</p> <ul style="list-style-type: none"> - Calibracion.h - qirr/qirrc - qirr/datosOS_fbank_scaled.h <p>Archivo de sensor -> E_Lectura</p> <p><u>Niveles ponderados</u></p>
	Limitador.cpp	cpp		<p>Incluir:</p> <ul style="list-style-type: none"> - Atenuador.cpp - SevidorDelEstado.cpp - comun/Configuracion.h - comun/EstadoDeModificadores.h - Registro.h - Registrador.cc - sys/i/o.h - AtenuadorPga.h - Sonometros.cpp <p>En la salida estándar, aparece una línea que dice <!-- Reopening --> cada 4 salidas normales (registros). ¿Qué significa?</p>
	Makefile mixers	C/C++	<p>makefile</p> <p>Construye el ejecutable 'limitador'</p> <p>Define un vector con el path a 3 mixers: /dev/mixer1, /dev/mixer2, /dev/mixer3</p>	Se incluye en Atenuador.cpp

Directorio	Archivo	Tipo	Descripción	Notas de Interés
	octaveBandFilter.c	c	Conjunto de filtros (funciones) autogenerados por mkfilter. Las órdenes que se utilizaron para generarlos se encuentran comentadas en las cabeceras de cada función.	
PruebaAtenuador		ejecutable	Define e implementa la clase PuertoParalelo, la cual permite leer y escribir en los registros de datos del puerto paralelo de la máquina.	
PuertoParalelo.cpp	cpp		El registro de datos del puerto o bus paralelo se compone de 8 bits bidireccionales en la dirección 0x378. Pines de datos [2-9]	
qirr	directorio		Contiene la implementación de un filtro digital en cascada en Forma Directa II Traspuesta. Los archivos más importantes son qirr.h y qirr.c, ambos comentados y bastante limpios en cuanto a código. Esta clase se incluye en la compilación de LIMITADOR	Comentario sobre la cabecera de la función qirr_filter(quirr *f, int x); "Calcula la salida para la entrada x utilizando una Casdada en Forma Directa II Traspuesta Proakis, Manolakis, "Digital Signal Processing" 4th Ed, pg 589"
			Implementa los métodos de la clase Cabecera, y define e implementa la clase Registrador.	
			En la clase Cabecera, el constructor inicializa sus atributos a valores estáticos, los cuales son: - tipo=370 - horaBase=time(NULL) - cadena=" rfs (Acusticayaudio.com) "	Incluye: - Registro.h Archivo de datos: F_Registro, F_SalidaRegistro
Registrador.cc		cc	La clase Registrador se encarga de las operaciones de gestión de los Registros, mayormente lectura y escritura.	
			- inicializa(): inicializa el fichero F_Registro, con una cabecera y 1000 registros vacíos. - horaBase(): lee la hora base desde la cabecera del archivo F_Registro - obtienPosicion(hora): devuelve la posición del registro de la hora pasada como argumento - guardarRegistro(registro r): escribe el registro r en la posición que le corresponde (según su hora) - leeRegistro(): varias implementaciones, pero todas leen el registro dado desde F_Registro y lo almacena en un registro R. - pasaRegistros(): lee una franja de registros, desde time_t inicio a time_t fin. Los escribe en F_SalidaRegistro	
Registrador.cpp	cpp		Implementación de los métodos de la clase Registrador.	Incluye: - comun/Config.h - Registro.h Archivo de datos: F_Registro
			La mayoría de los métodos son getters. Los métodos para devolver las presiones y la atenuacion_maxima devuelven los valores divididos por 2.	Archivo de datos: F_Registro

Directorio	Archivo	Tipo	Descripción	Notas de interés
			Contiene dos clases, Cabecera y Registro.	
Registro.h	h		Registro consiste en una serie de atributos tomados de Estado Y Configuración, y una lista de métodos para manejar dichos atributos y leer/escribir el registro a disco. Atributos: - hora - informe - atenuacion, atenuacion_max, db - crc - presion_Izquierda, presion_Derecha, presion_Repcion	Incluye: - Estadio.h - comun/Configuracion.h"
ServidorDeEstado.cpp	cpp		Implementación de la clase ServidorDeEstado. Sendos métodos para escribir y leer el estado del limitador hacia/desde el fichero F_Estado.	Incluye: - ServidorDeEstado.h
ServidorDeEstado.h	h		Definición de la clase. Contiene un atributo del tipo EstadoDelimitador, y dos métodos para leer y actualizar el estado.	Fichero de datos: <u>F_Estado</u> Incluye: - Estadio.h - comun/EstadoDelimitador.h - comun/Configuracion.h
sonometro	ejecutable			
Sonometro.cpp	cpp		Solo contiene un main vacío.	Incluye: - Sonometros.cpp
sonometros	ejecutable			

Directorio	Archivo	Tipo		Descripción	Notas de interés
			Clase Sonometro		
Sonometros.cpp	cpp			<p>Atributos:</p> <ul style="list-style-type: none"> - int _lecturasPorSegundo - Lector *lineas - Calibracion calibracionMic - Calibracion calibracionLi - Calibracion calibracionLd - Lectura2C ultimaLecturaDeMicrofono - Lectura2C ultimaLecturaDeLineas <p>Métodos:</p> <ul style="list-style-type: none"> - Sonometros(char *tarjetaMic, char *tarjetaLinea); inicializa los lectores (sensores) para el microfono y las líneas (audio). El segundo puede ser NULL, de forma que el limitador pueda actuar en modo registrador (solo registra, no limita "CREO": - reloadCalibracion; fuerza el refresho de calibracionMic, calibracionLi, calibracionLd - numeroDeLecturasPorSegundo(int n); _lecturasPorSegundo = min(n, 10). - lecturasPorSegundo -> microfono y líneas - startLines(); crea y lanza una hebra para el control de las líneas. Ejecuta la función Sonometros._auxFunction - Sonometros._auxFunction: bucle infinito. Realiza lo mismo que read pero para la líneas (audio) en lugar de para el microfono - read(); actualiza ultimaLecturaDeMicrofono, invocando a la función lectura de la clase Lectura2C. Renicia las lecturas cuando se alcanzan las lecturas por segundo. 	<p>Incluye:</p> <ul style="list-style-type: none"> - sys/soundcard.h - sys/otcl.h - Lectura2C.cpp - Calibracion.cpp - iir.cpp - Lector.cpp <p>Fichero de datos: <u>E_Kal</u></p> <p>Micrófono --> /dev/dsp1 Audio --> /dev/dsp</p>
SwitchhPink.cpp	cpp			<p>Programa que recibe como argumento un entero "pos", y lo pasa como paramámetro al método pinDeDatos() de la clase PuertoParalelo.</p> <p>El pin al que se accede es el número 4. Se activa o desactiva el pin si el número pos es mayor que 0 o no.</p>	<p>Incluye:</p> <ul style="list-style-type: none"> - PuertoParalelo.cpp <p><u>PINK SWITCH</u></p>
testAt	ejecutable			Ejecutable de TestAtenuador.cpp	
TestAtenuador.cpp	cpp			AtenuadorPga.ataenua(0)	<p>Incluye:</p> <ul style="list-style-type: none"> - AtenuadorPga.cpp
testDeCalibracion.cpp	cpp			Define e implementa la clase ControlDeCalibracion. No se ha acabado, y tampoco se usa en ningún lugar.	
testPga	ejecutable			Ejecutable de testPga.cpp AtenuadorPga.ataenua(at)	<p><u>MAX_POINTS</u></p> <p>Incluye:</p> <ul style="list-style-type: none"> - AtenuadorPga.h - Sonometros.cpp
TestPga.cpp	cpp			at es un entero que se pasa al programa por línea de comandos	<p>Incluye:</p> <ul style="list-style-type: none"> - AtenuadorPga.cpp (y por cascada, incluye todos los include de AtenuadorPga)

Directorio	Archivo	Tipo	Descripción	Notas de interés
	TP.cpp	cpp	Test PuertoParalelo.cpp. Solo contiene un main al que se le pasa un entero por línea de comandos (x), para luego instancia un objeto de PuertoParalelo Y activar solo el pin de datos x. El resto de pines se ponen a false (son 8 en total, un byte)	Incluye: - AtenuadorPga.cpp (y por cascada, incluye todos los include de AtenuadorPga)
reports/	a.out cabeceraInformes.svg dataGet.cpp dygraph-combined.js excanvas.min.js	ejecutable SVG	Versión "desactualizada" de getData https://d3graphs.com/ https://github.com/GerHobbel/excanvas	
	getConfig.cpp		Genera informes sobre la configuración del limitador en varios formatos: - JSON - XML	Incluye: - limitador/Registrador.cc - limitador/Registro.cpp - limitador/Estado.cpp - limitador/Calibracion.cpp - comun/Configuración.cpp
	getdata getData	ejecutable ejecutable	Genera informes sobre los registros en varios formatos: - Uri ¿? - JSON - XML	Incluye: - limitador/Registrador.cc - limitador/Registro.cpp - limitador/Estado.cpp - comun/Configuración.cpp
	getData.cpp		Recibe 4 parámetros: - formato, fechaini, fechaFin, intervalo entre registros Script que recibe 3 parámetros, formato, fecha de inicio y fecha de fin, de modo que la invocación del script sería como sigue: php?> getEvents <format> <startDate> <endDate>	Se hace referencia a este script en commv2/main.cpp
	getEvents	script PHP	donde format puede ser uri o json y las fechas se dan en formato year/month/day-hour:minute	Fichero de logs: /var/slr/logs/serial
	getInputs.cpp		El script abre el fichero de logs y recupera los logs de eventos (aquellas acciones llevadas a cabo por usuarios) ocurridos entre las fechas de inicio y final, y los devuelve en el formato especificado. URI == raw -> los devuelve tal y como los lee del fichero de logs	
	getStatus.cpp		Genera informes sobre las entradas del limitador. Los muestra por pantalla usando printf.	Incluye: - limitador/Lectura2C.cpp - limitador/Calibracion.cpp

Directorio	Archivo	Tipo	Descripción	Notas de interés
graphicalReport		directorio	Contiene una serie de ficheros con patrones de texto y un fichero CPP con el mismo nombre que el directorio.	
gRegist.php		html, php, CSS, js	No lo han completado y no se usa	
gRep		directorio	Vacio	
gReport.php		html, php, CSS, js	No se usa	
Makefile		Makefile	Genera los ejecutables getStatus, getData, getConfig	
testPerformance.php		php	Nada importante	
scripts/				
connectNetwork	script bash	Inicializa wvdial	wwdial	
continuousPink	script bash	Bucle infinito de "ruido rosa"	alsapiayer	
controlDeCalibracion		Se apoya fuertemente en la librería de funciones calibrationControl.	Incluye: - www/calibrationControl.lib	
keepComm	script bash	Registra la actividad del equipo y genera la media de las lecturas para los últimos 10 minutos.		
keepPeds	script bash	Bucle infinito. Parece que comprueba cada cierto período de tiempo si hay conexión a internet haciendo un ping a Google, en tal caso ejecuta sircomm, en otro caso regenera la conexión con connectNetwork	killComm	
keepLm	script bash	Bucle infinito. Ejecuta ledControl cada segundo.	ledControl	
keepScreen	script bash	Bucle infinito. Reinicia constantemente el limitador y registra el timestamp en var/sl/r/restarts	bin/limitador	
	script bash	Mantiene la terminal activa para ledControl con screen /var/sl/eds/port en un bucle while infinito.	screen	
keepTime	script bash	Bucle infinito. Cada 60 segundos comprueba si hay conexión a internet realizando un ping a Google. En caso afirmativo, actualiza la hora con ntpdate y duerme durante 1000min(16h). Para actualizar la hora del sistema primero remonta el sistema de archivos en modo lectura-escritura, actualiza la hora y luego lo devuelve a modo solo lectura.		
killComm	script bash	Bucle infinito. Espera 4 HORAS antes de matar el proceso slrComm		
ledControl	script PHP	Solo invocado por keepPeds. Se apoya en las funciones de las librería calibrationControl.lib y define e implementa algunas funciones propias. Abre una terminal con screen y muestra la media de las medidas del micrófono en ella, cambiando el color de la fuente según la cercanía de las medidas a los umbrales máximos (presión / dBs). Muestra el estado y la media de las medidas (mic, ld, Ll), carácter a carácter, usando echo con redirección a la terminal abierta por screen.	Incluye: - www/calibrationControl.lib	
playPink	script PHP	Igual que continuousPink pero en PHP		
refreshProcess	Script sh	Bucle infinito. Mata al proceso slrComm cada 15min (lo refresca)		

Directorio	Archivo	Tipo	Descripción	Notas de interés
remoteShellService	script bash		Bucle infinito. [Función desconocida] Establece una conexión mediante netcat con boanergesnetwork.com , en el puerto \$serial = {limInfo n}; Una vez establecida la conexión ejecuta /bin/localservice	nc boanergesnetwork.com -e bin\localservice
tests/	a.out estadisticos.cpp estadisticos.php normativaExtendida.cpp	ejecutable cpp php cpp	<p>La web citada arriba parece ser una herramienta de acceso remoto. LDAP (Lightweight Directory Access Protocol)</p> <p>Se repite cada 30 segundos.</p> <p>bubblesort y media aritmética sobre un vector de float</p> <p>Pruebas sobre getData. No se usa.</p> <p>Tests sobre los valores máximos permitidos en los días semanales y los días festivos según la normativa extendida. (La cual desconozco)</p> <p>Test que comprueba si el sistema está activo. Instancia un objeto de la clase Configuracion, y llama a los método carga() y estaActivo()</p> <p>-carga(): carga el fichero de configuración "/var/slr/configuracion" (definido en comun/Config.h) y desencripta su contenido.</p>	incluye comun/Configuracion.cpp
tActivo.cpp		cpp	<p>NOTA</p> <p>Al parecer han definido un encriptado a nivel de bits para almacenar la información sensible de la configuración del sistema, como los detalles del local, claves y demás.</p> <p>Cada uno de estos campos se almacena en un vector de tipo char. El vector se rellena carácter a carácter coo resultado del desencriptado. El desencriptado consta de la suma de dos caracteres que dependen del carácter encriptado x:</p> <ul style="list-style-type: none"> - p1 = (x & 0x0f)<<4 - p2 = (x & 0x0f)>>4 - x = p1 + p2 <p>Realiza un test de calibración. Define y configura un objeto de la clase Calibracion.</p>	incluye limitador/Calibracion.cpp
testDeCal.cpp		cpp	Hace una llamada al método calcula_dbs de dicha clase	incluye limitador/Registrador.cpp
utils/	a.out autoEq	ejecutable ejecutable	Llama al método obtenerPosicion(time) de la clase Registrador. Este método devuelve un número entero que corresponde al nº de registro de la hora dada.	

Directorio	Archivo	Tipo	Descripción	Notas de interés
			Función getLecturas y main.	
AutoEqualizador.cpp	cpp		<p>getLecturas(int lecturas, Lectura2C &mic, Lectura2C &lin)</p> <p>- Lee el número de lecturas indicado desde los archivos "/tmp/.lx[0-1]", donde 0 es la lectura del micrófono y 1 la lectura de las líneas en ese momento. Las lecturas se van almacenando en dos vectores para luego obtener sus medias y devolverlas en mic y lin.</p>	<p>Incluye: -limitador/Lectura2C.cpp</p> <p><u>NúmeroDeLecturas</u></p>
BoanergesConfigurator.cpp	cpp		<p>El main se tiene Lectura2C media_mic, media_lin y Calibracion media, izquierda, derecha. Se leen las medias y las calibraciones de cada línea (previamente se calculan), se calcula la equalización para cada línea y se aplican. Finalmente se escribe a fichero las calibraciones.</p> <p>Contiene una función print y un main. El print muestra por pantalla algunos datos de configuración. El main permite recibir de forma opcional 2 parámetros. De forma general, la invocación sería /BoanergesConfigurator <Attributo> <Valor>. Los valores admitidos para Attributo son: [Server, ServerDataRevision, ServerPort, GuidRemoto, ServerKey, LocalKey]. Cuando se recibe uno de estos atributos, se actualiza el valor de dicho atributo en la configuración al valor pasado como segundo argumento, tanto en memoria como en disco. Solo se permite actualizar un atributo cada vez. Si no se reciben argumentos, se muestra la configuración actual por pantalla.</p> <p>Controla el nivel de calibrado del sistema.</p>	<p>Incluye: - comun/Configuracion.cpp</p>
controlDeCalibracion	script php		<p>TODO usa muchas funciones de la librería, las cuales habría que estudiar</p>	<p>Incluye: - www/calibrationControl.lib</p>
gen	ejecutable	cpp	Generador "aleatorio" de cadenas. ¿Para contraseñas?	
gen.cpp	cpp		Muestra por pantalla el número de serial (ID) del dispositivo, y lo vuelve en serial.h	<p>Incluye: - comun/serial.h</p>
getSerial.cpp	ejecutable	cpp		
gSerial	ejecutable	cpp	No se usa. Muestra por pantalla información general sobre el limitador. Es menos completo que LimInfo	<p>Incluye: - comun/Configuracion.cpp - comun/EstadoDelLimitador.cpp</p>
helper.cpp	ejecutable	cpp		<p>Incluye: - comun/Serial.h</p>
Inicializador.cpp	cpp		Arranca e inicializa el registrador	<p>Incluye: - limitador/Registrador.cc</p>
LimInfo.cpp	cpp		Imprime por pantalla información relativa al limitador, estado, configuración, detalles del local, etc. El programa recibe como parámetros una cadena de caracteres los cuales indican qué mostrar, por ejemplo /liminfo 5 muestra la dirección del local, /liminfo A muestra el valor de atenuación máxima, y /liminfo A5 muestra ambas cosas	<p>Incluye: - comun/Configuracion.cpp - comun/EstadoDelLimitador.cpp - comun/Funciones.h - comun/Serial.h</p>
Linet.cpp	cpp		Obsoleto. Utiliza una clase llamada Comunicador, la cual no existe. La busca dentro de la carpeta comun/	

Directorio	Archivo	Tipo	Descripción	Notas de interés
			Genera los siguientes archivos: - utilisr - gen - limInfo - localService - inicializador - boanerges.config - eq - gSerial	Ejecuta y redirecciona la salida de gSerial (getSerial.cpp) a serial.h
Makefile		Makefile		
playPink	script php		Lanza alsaplayer para que reproduza ruidos rosa en un bucle infinito.	alsaplayer -d hw:1,0 /var/sir/pink.wav pinkVolume: amixer -c 1 sset PCM \$nivel Incluye: - www/calibrationControl.lib
	serial.h	h	Archivo vacío. Recibe la salida de gSerial (getSerial.cpp) Compilado como localService, tiene como finalidad el crear una especie de interpretador de comandos / consola remota, la cual acepta las siguientes órdenes: - datos: - update: - configuracion: - configura: - activa: - concha: - concharemota: - identifica: - calibra: - prepara: - operativo: - test: - calibracion: - salir: - ayuda: - sql: - xml: - yml: - id: - version: - identificame: - identificate:	Incluye: - serial.h - comun/aes.cpp - comun/Configuracion.h - comun/Configurador.cpp - comun/EstadoDelLimitador.h - comun/Funciones.cpp - limitador/Lectura2C.cpp - limitador/Registrador.cc - limitador/AtenuadorPga.cpp Incluye condicionalmente: - comunicacion/servidor.cpp La carpeta que contiene este archivo esta marcada como obsoleta, por lo que imagino que este interprete quedó obsoleto.
	ServidorSerie.cpp	cpp		- calibralineas: - mic: - miceq: - lefteq: - righteq:

Directorio	Archivo	Tipo	Descripción	Notas de interés
Importante	utilisr.cpp	cpp	Backend de la interfaz web. La interfaz web en php se comunica con este programa de forma extensiva, por ejemplo para el login, se invoca a este programa de la siguiente forma: php \$> utilizar clave <password>	Incluye: - comun/Funciones.h - comun/Configuracion.cpp - comun/Configurador.cpp
	x.cpp	cpp	Lee y muestra por pantalla el último registro del sistema, representado por la macro F_Registro	Incluye: - limitador/Registro.cpp - limitador/Registrador.cc
www/	access.php	php	Conjunto de funciones para la gestión de usuarios y login. Usa el fichero users.auth como base de datos	
	authPart-ExtendedNormative.php	php, html, css, js	Para la gestión de la normativa extendida (máximos permitidos por día y hora) Petición AJAX a functions/updateExtendedNormative.php	Usa getConfig, limInfo
	calibrationControl.lib	php	Colección de funciones. Se entienden fácilmente y están brevemente comentadas. Tal y como el nombre sugiere son funciones relacionadas con el control de calibración y las sesiones de ruido. Contiene funciones para comenzar y parar a emitir ruido rosa, comenzar y parar sesiones de calibrado, aplicar y eliminar atenuación a las líneas, obtener información del limitador (delegando en getStats, getConfig...) y leer / escribir logs (delegando en logs.php)	Incluye: - logs.php
	configFail.inc	html	Trozo de HTML a mostrar cuando la identificación falla	
	configPart.php	Vacio	Vacio	
	configuration.php	php	Define lo siguiente: \$GlobalConfiguration[reportOutputFile]="/tmp/reportEngine.output"; \$GlobalConfiguration[reportsBaseDir]="/var/slr/tpl/"; \$GlobalConfiguration[reportOutputType]= "/tmp/reportEngine.output.type";	
	config.xml	xml	Información del equipo y valores por defecto	
	css	css	--	
	fonts	fonts	--	
Importante	functions	php	Funciones AJAX. Comunicación con el backend.	functions/calibrate.php: Utiliza localService para aplicar los cambios
	help.png	--	--	
	images	--	--	

Directorio	Archivo	Tipo	Descripción	Notas de interés
reports	rightEqualizer.php		Petición AJAX a functions/calibrate.php TODO	Web UI -> Calibrar -> Ecualizador derecho
	scripts.js		Realiza una petición AJAX a functions/status.php para obtener el estado del limitador. Tras obtener los datos, los procesa y actualiza la IU (la parte de statusPart.php)	
	simple		Web simple que muestra la parte central de la web normal (barras y gráfico) en formato reducido	
	smallInfoPanel.php		No se usa. Corresponde al panel de información en el menú desplegable, el que contiene los sliders para configurar el equipo.	
	statusPart.php		Parte del estado actual del limitador. La parte central de la UI que muestra las lecturas actuales. Incluye: - scripts.js	Petición AJAX a functions/status.php mediante scripts.js
	style-Blue.css	css		--
	style.css	css		--
tabAuth.php			Menú desplegable superior cuando se está identificado.	Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/: - updatePwd - updateCfg - calibrate - pink - updateDataCu - validateUserCode - network - updateTime
			No se usa (comentado). Menú desplegable superior cuando se está identificado y el equipo es un registrador.	Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/: - updatePwd - soundControl - calibrate - updateDataCu - validateUserCode - network
	tabAuthRegister.php			

Directorio	Archivo	Tipo	Descripción	Notas de interés
			Menú desplegable superior cuando NO se está identificado.	
tabNormal.php			Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/: - updatePwd - soundControl - calibrate - updatedDataCu - validateUserCode - network	
templateMng.php			No se usa.	
tests			--	
tractis			http://www.tractis.com/	
updateDataCu.php			Actualiza los datos del local. Para ello escribe la nueva información en el archivo /tmp/conf.tmp y luego ejecuta 'utilslr configura /tmp/conf.tmp'	
users.php			Gestión de usuarios	

Apéndice G

LM7: constantes y macros

Ims7-Defines

Archivo	Define	Valor
comun/Funciones.h:	Ext2	TRUE
comun/Funciones.h:	Vfat	FALSE
comun/Funciones.h:	_TextoRobado	Este aparato ha sido ROBADO. Contacte con la POLICIA y con Acustica y Audio S.L. en el telefono (+34) 981 80 33 93. Su aparato sera sustituido y sus datos accesibles
comun/Configurador.cpp:	ifComp(x)	if(strstr(buffer,x)==buffer)
comun/Config.h:	LocalBase	Dispositivo ILEGAL
comun/Config.h:	AyuntamientoBase	Avise a Acustica y Audio 981 803 393
comun/Config.h:	EPresion	presion
comun/Config.h:	EMedia	media
comun/Config.h:	ENumeroDeSerie	numeroDeSerie
comun/Config.h:	EVersion	version
comun/Config.h:	EAtenuacion	atenuacion
comun/Config.h:	EMaximo	maximo
comun/Config.h:	EAtenuacionMax	atenuacionMaxima
comun/Config.h:	PasswordPorDefecto	ABCD
comun/Config.h:	Debug(x)	puts(x)
comun/Config.h:	MargenDeSeguridad	0
comun/Config.h:	TipoControlMic	0
comun/Config.h:	TipoControlLinea	1
comun/Config.h:	TipoControlMixto	2
comun/Config.h:	DiferenciaMixto	3
comun/Config.h:	MinMixto	10
comun/Config.h:	DiasActivoTrasConfigurar	21
comun/Config.h:	Formato16	AFMT_S16_LE
comun/Config.h:	Linea	SOUND_MASK_LINE
comun/Config.h:	CD	SOUND_MASK_CD
comun/Config.h:	AUX	SOUND_MASK_LINE1
comun/Config.h:	Microfono	SOUND_MASK_MIC
comun/Config.h:	_Octava	12
comun/Config.h:	CanalMicrofono	1
comun/Config.h:	CanalLinea	2
comun/Config.h:	CanalAux	3
comun/Config.h:	CanalCd	4
comun/Config.h:	S_Frec	44100
comun/Config.h:	PuertoDeEstado	9099
comun/Config.h:	OrdenanzaBase	22:00 10H 090.0 095.0
comun/Config.h:	NormaBase_Inicio	8
comun/Config.h:	NormaBase_Fin	22
comun/Config.h:	NormaBase_MaximoDiurno	95
comun/Config.h:	NormaBase_MaximoNocturno	95
comun/Config.h:	TiempoBajoMaximo	5
comun/Config.h:	CalibracionBase	94,196
comun/Config.h:	Calibracion_dB	94
comun/Config.h:	Calibracion_Ref	196
comun/Config.h:	MaximoDeMicrofono	120
comun/Config.h:	GananciaPorDefecto	50
comun/Config.h:	MascaraLm307	0x00F0
comun/Config.h:	MascaraLm301	0x0FO0
comun/Config.h:	MascaraNoLm	0x000F
comun/Config.h:	Version	7.1
comun/Config.h:	IdentificacionBase	1
comun/Config.h:	MaximoPermisos	12
comun/Config.h:	M_At_Rel	214
comun/Config.h:	M_At_Abs	56
comun/Config.h:	T_Buffer1	60
comun/Config.h:	T_Buffer2	10
comun/Config.h:	INoInforme	0
comun/Config.h:	IAtenucion	1
comun/Config.h:	ILlave	2
comun/Config.h:	INoLlave	4
comun/Config.h:	IMaximaAtenuacion	8
comun/Config.h:	IEncendido	16
comun/Config.h:	IPermisividad	32
comun/Config.h:	IMicro	64
comun/Config.h:	IPenal	128
comun/Config.h:	F_Config	/var/sl्र/configuracion
comun/Config.h:	F_Registro	/var/slր/registro.slr
comun/Config.h:	F_SalidaRegistro	/tmp/registro.extracto
comun/Config.h:	F_SalidaComReg	/tmp/salidaComReg.ext
comun/Config.h:	F_FicheroDeRegistro	F_Registro
comun/Config.h:	ConfiguracionExterna	Configuracion.slr

Ims7-Defines

Archivo	Define	Valor
comun/Config.h:	ActualizacionExterna	lm307.act
comun/Config.h:	NumeroDeComprobacion	361540
comun/Config.h:	F_Exportable	/tmp/informe.slr
comun/Config.h:	F_ExportableCom	/tmp/informeCom.slr
comun/Config.h:	F_Estado	/tmp/estado.slr
comun/Config.h:	F_Llave	/tmp/sl.r.k
comun/Config.h:	F_Kal	/var/sl.r/calibracion
comun/Config.h:	F_Permitido	644
comun/Serial.cpp:	HDOIO_GET_IDENTITY	0x030d
comun/Serial.cpp:	MarcaLectura	HWaddr
comun/Funciones.cpp:	F_ExportacionXml	/tmp/export.xml
comv2/cliente.c:	MAXDATASIZE	100
comv2/globales.h:	_TiempoDeEsperaEntreReconexiones	30
comv2/globales.h:	_IntentosDeAutentificacion	3
comv2/globales.h:	_TamanhoDelBufferDeRespuesta	150
comv2/globales.h:	_SeparadorDeParametros	"::"
comv2/globales.h:	SegundosEntreActualizacionesDeConfiguracion	(60 * 60 * 1 * 1)
comv2/globales.h:	_ComandoEnvioDeConfiguracion	sendconfig
comv2/globales.h:	_ComandoEnvioDeDatos	senddata
comv2/globales.h:	_ComandoPeticionDeAutenticacion	auth
comv2/globales.h:	_ComandoRespuestaDeAutenticacion	authanswer
comv2/globales.h:	_ComandoPing	ping
comv2/globales.h:	_ComandoHello	hello
comv2/globales.h:	_ComandoLocalTime	localclock
comv2/globales.h:	_ComandoLastConfigDate	lastconfigdate
comv2/globales.h:	FicheroDeMarcaSinConexion	/tmp/nonAvailableConnection
limitador/PuertoParalelo.cpp:	byte	unsigned char
limitador/qiir/qiir.h:	IQmult(a,b,c)	((int32_t)((int64_t)(a)*(int64_t)(b))>>(c)))
limitador/qiir/qiir.h:	IQ23toIQ29(A)	((int32_t)(A) << (29 - 23))
limitador/qiir/qiir.h:	IQ29toIQ23(A)	((int32_t)(A) >> (29 - 23))
limitador/qiir/qiir.c:	B0	0
limitador/qiir/qiir.c:	B1	1
limitador/qiir/qiir.c:	B2	2
limitador/qiir/qiir.c:	A1	4
limitador/qiir/qiir.c:	A2	5
limitador/qiir/qiir.c:	W1	0
limitador/qiir/qiir.c:	W2	1
limitador/qiir/datosSOS_fbank_scaled.h:	NBIQ	3
limitador/qiir/datosSOS_fbank_scaled.h:	FILTER_BANDS	8
limitador/qiir/datosSOS_fbank_scaled.h:	SOS_FRACBITS	29
limitador/qiir/datosSOS_fbank_scaled.h:	G_FRACBITS	29
limitador/Limitador.cpp:	MIC_HW	/dev/dsp1
limitador/Limitador.cpp:	LINE_HW	/dev/dsp
limitador/Limitador.cpp:	RUN_LINES	
limitador/Limitador.cpp:	MIC_HW	/dev/dsp
limitador/Limitador.cpp:	LINE_HW	/dev/dsp1
limitador/Limitador.cpp:	LmiteDePenalizacion	3
limitador/Limitador.cpp:	LmiteParaAntiguo	150
limitador/Limitador.cpp:	PARTES_POR_SEGUNDO	5
limitador/Limitador.cpp:	base	0x378
limitador/iir.cpp:	FilterBankFile	/var/sl.r/datafilter.dat
limitador/Registrador.cc:	HoraBase	((float)(36 * 365.25 * 24 * 3600))
limitador/Registrador.cc:	Numero	1
limitador/testDeCalibracion.cpp:	MaxPoints	15
limitador/Atenuador.cpp:	C_AT	
limitador/Atenuador.cpp:	MaxLinea	85
limitador/Lectura2C.cpp:	F_Lectura	/tmp/.lx
limitador/Lectura2C.cpp:	NBandasOctava	8
limitador/Lectura2C.cpp:	Banda125Hz	2
limitador/Lectura2C.cpp:	Banda1KHz	5
limitador/filterTest.cpp:	Spectrum_BandCount	8
limitador/filterTest.cpp:	IndexOf500Hz	17
limitador/filterTest.cpp:	FragmentSize	11025
limitador/filterTest.cpp:	SamplingFrequency	11025
limitador/AtenuadorPga.h:	PGA_CS	7
limitador/AtenuadorPga.h:	PGA_CLK	6
limitador/AtenuadorPga.h:	PGA_SDI	5
limitador/AtenuadorPga.h:	PINK_SWITCH	4
limitador/AtenuadorPga.h:	PGA_Wait	250
limitador/AtenuadorPga.h:	RegistroDeEstado	1
limitador/AtenuadorPga.h:	PinDeMicrofono	3
limitador/Sonometros.cpp:	F_Kal	/var/sl.r/calibracion
limitador/SwitchPink.cpp:	PINK_SWITCH	4

Ims7-Defines

Archivo	Define	Valor
limitador/filtroA/filtroA.cpp:	NCoef	2
reports/getInputs.cpp:	Uri	0
reports/getData.cpp:	FormatoFecha	%d/%d/%d-%d:%d
reports/getData.cpp:	TipoUri	1
reports/getData.cpp:	TipoJson	2
reports/getData.cpp:	TipoXml	3
reports/getData.cpp:	TipoUri2	4
reports/dataGet.cpp:	FormatoFecha	%d/%d/%d-%d:%d
reports/dataGet.cpp:	TipoUri	1
reports/dataGet.cpp:	TipoJson	2
reports/dataGet.cpp:	TipoXml	3
utiles/utilslr.cpp:	Clave	clave
utiles/utilslr.cpp:	Listado	listado
utiles/utilslr.cpp:	Grafico	grafico
utiles/utilslr.cpp:	Actividad	actividad
utiles/utilslr.cpp:	Sql	sql
utiles/utilslr.cpp:	Xml	xml
utiles/utilslr.cpp:	Yml	yml
utiles/utilslr.cpp:	SQLite	sqlite
utiles/utilslr.cpp:	Json	json
utiles/utilslr.cpp:	ACTIVA_WEB_COMMAND	activaweb
utiles/utilslr.cpp:	CConfiguracion	configuracion
utiles/utilslr.cpp:	CConfigura	configura
utiles/utilslr.cpp:	Si	si
utiles/utilslr.cpp:	No	no
utiles/utilslr.cpp:	Paso1	paso1
utiles/utilslr.cpp:	Paso2	paso2
utiles/utilslr.cpp:	PreAuth	preauth
utiles/utilslr.cpp:	VerifyCu"verifycu"	
utiles/utilslr.cpp:	FormatoFech	%d/%d/%d-%d:%d
utiles/utilslr.cpp:	//PreAuth	preauth
utiles/utilslr.cpp:	//VerifyCu	verifycu
utiles/AutoEqualizador.cpp:	NumeroDeLecturas	3000
utiles/ServidorSerie.cpp:	FormatoFecha	%d/%d/%d-%d:%d
utiles/ServidorSerie.cpp:	NombreDelPuerto	/dev/ttyS0