Universidad de la República
Facultad de Ingeniería

UrbanEar: Monitoreo sonoro urbano de bajo costo

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Leopoldo Agorio, Andrés Corchs, Hernán Pereyra

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Pablo Zinemanas	Universidad de la República
Martín Rocamora	Universidad de la República

Tribunal

Germán Capdehourat	Universidad de la República
Alicia Fernández	Universidad de la República
Nicolás Pérez	Universidad de la República
Javier Schandy	Universidad de la República

Montevideo
domingo 11 agosto, 2019

UrbanEar: Monitoreo sonoro urbano de bajo costo, Leopoldo Agorio, Andrés Corchs, Hernán Pereyra.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 117 páginas.
Compilada el domingo 11 agosto, 2019.
http://iie.fing.edu.uy/

¡Cállense ya, maldita sea! ¡Dejen dormir al prójimo!

Homero J. Simpson

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Corresponde saludar a todos los que nos acompañaron en el desarrollo de este largo proceso.

En primer lugar agradecemos a nuestras familias y amigos por haber sido nuestro sustento anímico.

A nuestros tutores, Pablo Zinemanas y Martín Rocamora por estar siempre disponibles a resolver nuestras inquietudes y por proveernos de referencias y materiales para el buen desarrollo del trabajo.

A Nicolás Pérez por su ayuda conceptual y material en múltiples detalles del armado de la caja. También a Rafael Canetti por ayudarnos a utilizar herramientas para el armado de la caja.

Al taller mecánico del Instituto de Física, particularmente a Antonio Saez y Joaquín Fernández por prestarnos sus herramientas y disponer de su tiempo.

A Pablo Pérez y Francisco Veirano por ayudarnos con el uso de la impresora 3d y los distintos programas asociados.

Al Instituto de Ingeniería Eléctrica que nos abrió sus puertas y nos permitió trabajar con comodidad.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El entorno sonoro urbano es un factor de gran relevancia en la calidad de vida de los ciudadanos. Por esta razón, existe un creciente interés en el desarrollo de sistemas de monitoreo automatizado, en la forma de una red de sensores acústicos distribuidos por la ciudad. En este proyecto se desarrolla un nodo de la red, quedando fuera de alcance el diseño de la red de sensores.

UrbanEar es un sistema autónomo que adquiere señales de audio y calcula indicadores descriptivos de sonido (energía en bandas de frecuencia y nivel de presión sonora). Es posible acceder al dispositivo a través de una red de datos, tanto de forma inalámbrica como cableada. La comunicación entre el dispositivo y el servidor se realiza mediante una aplicación, que permite extraer los datos generados y configurar sus funcionalidades.

El equipo está basado en una mini-PC Raspberry Pi 3B+ y cuenta con un micrófono MEMS digital. El sistema se encuentra en un gabinete estanco que permite su colocación a la intemperie. El costo total en componentes del dispositivo resultó de aproximadamente 160 USD con sugerencias para reducirlo a aproximadamente 100 USD.

El desempeño de los distintos módulos de software desarrollados, la comunicación, la estanquidad del dispositivo y la validez de los indicadores calculados por el equipo fueron ensayados una vez ensamblado el dispositivo obteniéndose resultados satisfactorios.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

1.1. Motivación y antecedentes

En zonas urbanas, la contaminación sonora es un problema importante, con consecuencias tanto económicas como sanitarias [1]. En Montevideo, el Instituto de Mecánica de los Fluidos de la Facultad de Ingeniería, Universidad de la República, en colaboración con la Intendencia Municipal ha realizado en distintas oportunidades mediciones de nivel de presión sonora en la ciudad. Sin embargo, estas medidas solo involucran dicho parámetro y son realizadas a intervalos de tiempo muy largo (varios años), de forma manual y por personal especializado [2].

En este contexto, es de gran interés desarrollar herramientas que permitan obtener un mapa sonoro de la ciudad en tiempo real, y así obtener información estadística en base a los datos relevados a lo largo de extensos períodos de tiempo y con datos específicos de nivel de presión sonora, y energía en bandas de frecuencia[1]. Para este fin es necesario un sistema basado en una red de sensores distribuídos por la ciudad. Por motivos de escalabilidad, cada elemento (nodo) de la red de sensores debe ser de bajo costo para facilitar la cobertura sobre áreas urbanas extensas.

Sistemas de este tipo están siendo desarrollados en otros países. Particularmente, utilizaremos como referencia el proyecto SONYC (*Sounds Of New York City*) llevado a cabo en la ciudad de Nueva York, Estados Unidos [3].

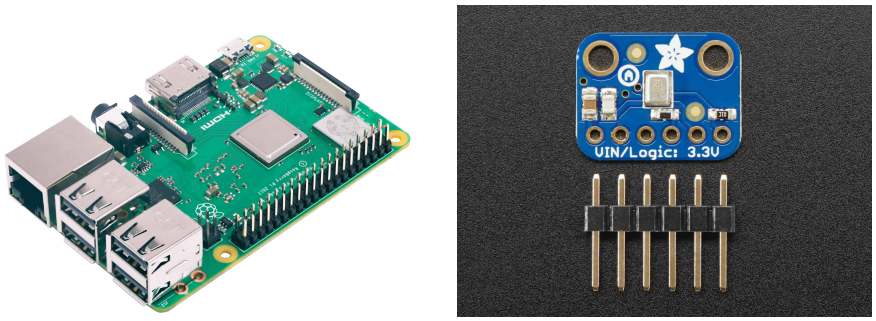
Además existen antecedentes en Facultad de Ingeniería en el desarrollo de dispositivos con características similares. Particularmente es de referencia para este proyecto el dispositivo Pestibee, que está acondicionado para operar en la intemperie, muestrea señales de sonido y envía a un servidor remoto por red celular [4]

En este proyecto, denominado UrbanEar, se desarrolla un dispositivo acondicionado para operar en la intemperie, capaz de muestrar señales de sonido, calcular descriptores específicos y comunicarlos a través de una red de datos. No se diseña en este proyecto la red de sensores, desarrollándose exclusivamente un nodo de la red.

[1] Estos datos son pasibles de utilizarse por algoritmos de aprendizaje automático para procesar que tipos de fuente de sonido se encuentran presentes.

1.2. Descripción general

El procesamiento de UrbanEar se realiza en una mini-PC (ver Figura 1.1), la adquisición se realiza con un micrófono MEMS (Sistema Microelectromecánico, por sus siglas en ingles) digital, y la transmisión se realiza a través de una red de datos. Es posible acceder a la red de datos inalámbricamente (Wi-Fi) o de forma cableada (Ethernet).



(a) Raspberry Pi B3+.

(b) Micrófono MEMS digital Adafruit.

Figura 1.1: El dispositivo está basado en una mini-PC Raspberry y utiliza un micrófono MEMS digital. A su vez, se utiliza una antena de Wi-Fi para comunicación y se colocará el montaje en un gabinete estanco.

El diseño apunta a un bajo costo de construcción y mantenimiento; así como a flexibilidad a nivel de software, permitiendo la adaptación de algoritmos y funcionalidades remotamente. Particularmente, se plantea como límite de costo tentativo 200 USD en origen.

1.3. Alcance

Compete al proyecto el diseño, ensamblado y posterior ensayo del dispositivo. Concretamente, UrbanEar es capaz de relevar el nivel de presión sonora (SPL) y la información tiempo-frecuencia de la señal, representada a través de la energía en bandas Mel. La energía en bandas Mel es un descriptor utilizado por los algoritmos de reconocimiento de sonidos en entornos sonoros [5].

El diseño del software relacionado con la identificación de patrones, por medio de algoritmos de aprendizaje automático queda por fuera del alcance del proyecto, al igual que el diseño de la totalidad de la red de sensores, diseñándose un único sensor.

1.4. Descripción funcional

Para generar los descriptores a transmitir, la señal grabada por el micrófono se pasa por un filtro de calibración que compensa la respuesta en frecuencia. El

micrófono digital se calibró contra un micrófono de referencia de manera de compensar su respuesta en frecuencia.

Con la señal compensada se calculan los descriptores con algoritmos que se explicarán más adelante, y se almacenan en la memoria del dispositivo. Se desarrolló en Python una interfaz para la comunicación UrbanEar-servidor. Además de permitir comunicación cableada, UrbanEar se configuró como Access Point de Wi-Fi, lo que permite la conexión remota de un servidor para descargar los datos relevados por el dispositivo. Un esquemático del funcionamiento del dispositivo comunicándose de forma remota se ilustra en la Figura 1.2.

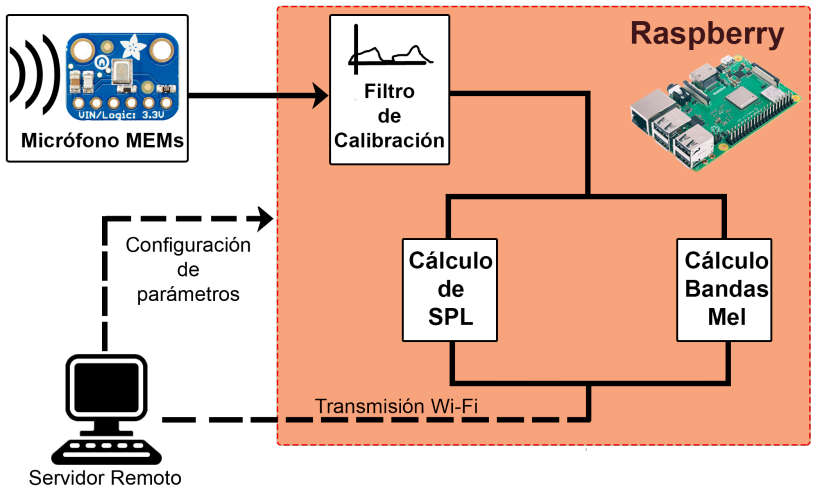


Figura 1.2: Diagrama del funcionamiento general de UrbanEar comunicándose con un servidor remoto.

1.5. Consideraciones adicionales

Además de almacenar los descriptores, UrbanEar es capaz de almacenar los archivos de audio grabados. Habilitar al servidor a recuperar estos archivos es un funcionamiento transitorio. Es importante notar que :

- La grabación de archivos de audio es útil para la etapa de pruebas, pero no sería parte de un producto final. La finalidad del trabajo es la de monitoreo de calidad de sonido urbano y no la vigilancia ciudadana, por lo que la grabación de audio y posible recuperación de conversaciones no es admisible.
- No es posible reconstruir el archivo de audio a partir del SPL y las bandas Mel ya que existe pérdida de información [6]. Sin embargo, los MFCC, descriptores obtenibles a partir de la energía en bandas Mel, se pueden utilizar para reconocimiento de voz [7]. Existen incluso trabajos en torno al reconocimiento de individuo mediante MFCCs a partir de una base de datos entrenada [8]. De todas formas, estos no son aplicables a un conjunto indefinido de hablantes, se centran en conjuntos de hablantes acotados.

- No hay legislación respecto a la audio-vigilancia ciudadana en Uruguay, sin embargo la video-vigilancia se encuentra amparada bajo la Ley de Protección de Datos Personales (Ley N 18.331) [9] según Dictamen 10/010 de Agesic [10] pudiendo ejercerce los derechos establecidos en la Ley. En particular el dictamen exige que la zona donde existe video-vigilancia esté correctamente delimitada y se informe a los ciudadanos de la existencia de cámaras.

1.6. Estructura del documento

El documento se estructuró en 7 capítulos incluyendo este, a saber:

- Capítulo 2: *Procesamiento de señales de audio*: Se describe la teoría de procesamiento de señales de audio a utilizar en el proyecto; particularmente se define SPL, bandas Mel y se aborda distintas técnicas de procesamiento de señales utilizadas.
- Capítulo 3: *Calibración del micrófono*: Se explica tanto el montaje experimental para realizar la calibración del micrófono como los algoritmos de procesamiento de los datos experimentales.
- Capítulo 4: *Diseño y montaje del dispositivo*: Se detalla el diseño físico del dispositivo, elección y fabricación de los distintos elementos.
- Capítulo 5: *Software*: Se describe la arquitectura de software, la vinculación de los distintos fragmentos de código y la comunicación con el servidor para configuración de parámetros y recuperación de datos.
- Capítulo 6: *Ensayos*: Se reportan los distintos ensayos realizados con el dispositivo funcional.
- Capítulo 7: *Conclusiones*: Se establecen las principales conclusiones del trabajo y se proyectan posibles líneas de profundización a futuro.

Capítulo 2

Procesamiento de señales de audio

El sonido consiste en la propagación de una perturbación en un medio mecánico. Según la psicoacústica (la rama de la psicofísica que estudia la relación entre los parámetros físicos de un estimulo sonoro y la respuesta subjetiva que produce en un oyente), cuando escuchamos un sonido, percibimos sensaciones que pueden ser clasificadas en tres tipos: la altura, la sonoridad y el timbre. La altura es la sensación que nos permite distinguir los sonidos graves de los agudos y más específicamente, diferenciar los sonidos de una escala musical. La sonoridad, en cambio, es la sensación por la cual distinguimos un sonido fuerte de uno débil. Por timbre se entienden una serie de cualidades por las cuales es posible distinguir los sonidos de los diversos instrumentos y voces. [11]

En una primera aproximación, cada parámetro físico del sonido se corresponde de manera más o menos directa con un tipo de sensación psicoacústica específica. Así, la frecuencia está relacionada con la sensación de altura, la amplitud con la sonoridad, y el espectro (en particular la envolvente espectral) con el timbre. Sin embargo, la cuestión no es tan sencilla, y existe en general una gran dependencia entre cada sensación y todos los parámetros del sonido. [11]

En esta sección se describen dos descriptores que son relevantes para el análisis y tienen vínculos estrechos con las sensaciones mencionadas. Particularmente se trabaja con el nivel de presión sonora (SPL) que guarda relación con la sonoridad, y con la energía en bandas de frecuencia en escala Mel que se puede relacionar con el timbre.

2.1. Nivel de presión sonora (SPL)

2.1.1. Definición Física

El oído humano tiene una percepción aproximadamente logarítmica de la intensidad de presión (intensidad sonora). [11]

Se define el Nivel de Presión Sonora (SPL por sus siglas en inglés) para una

onda de presión de sobrepresión p como:

$$\mathrm{SPL}_{(\mathrm{dB})} = 20 \log_{10} \left(\frac{p}{p_0} \right), \tag{2.1}$$

donde $p_0 = 20$ μPa se toma como presión de referencia por ser cercana a la mínima presión audible para una señal de frecuencia de referencia $f_0 = 1$ kHz. El SPL es por lo tanto una medida de la intensidad de la onda de sonido.

A su vez, la percepción humana tiene una respuesta en frecuencia no plana [12]. Dos señales de la misma amplitud pero distinta frecuencia son percibidas con distinta sonoridad. Esto se ilustra en la Figura 2.1, donde se representan las distintas curvas empíricas de iso-percepción auditiva (misma sonoridad percibida) referidas al valor de SPL (dB) para 1 kHz. A su vez la figura muestra una de las curvas de ponderación descriptas en el estándar ISO.

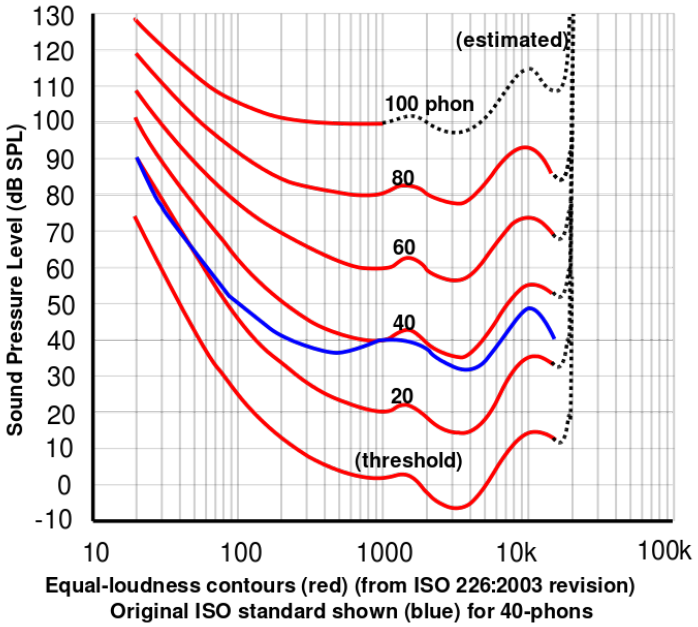


Figura 2.1: Curvas de igual percepción sonora y ejemplo de curva de ponderación. Extraído de [12].

Con el objeto de poder tener una medida de intensidad sonora que guarde relación con la sonoridad, la IEC (Comisión Electrotécnica Internacional) define diversas curvas de ponderación de SPL como se muestra en la Figura 2.2.

Si una señal de audio es la entrada de un filtro de ponderación de este tipo, la salida resulta en una señal cuya intensidad se ajusta a la percepción del oído humano. Se definen el dBA y el dBC como las unidades de medida para estas intensidades.

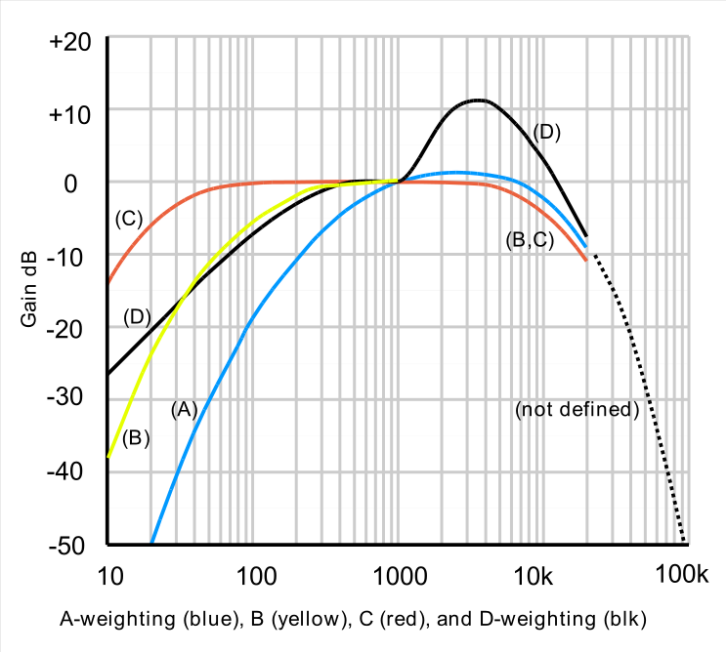


Figura 2.2: Curvas de ponderación definidas por la IEC. Extraído de [12].

2.1.2. Implementación Digital del cálculo de SPL

La definición presentada para SPL a través de la Ecuación 2.1 es un valor instantáneo.

Es necesario realizar una aproximación del SPL en función de valores medios. A estos efectos se define el L_{eq}: el nivel de presión sonora producido por una señal constante pero equivalente en potencia a la que se le desea calcular el SPL. La expresión para calcular el L_{eq} es la siguiente:

$$L_{eq,T} = 10\ log_{10}\left[\frac{1}{T}\int_0^T \frac{p(t)^2}{p_0^2}dt\right]\ (\text{dB}) \tag{2.2}$$

Es necesario ponderar la señal por alguna de las curvas de la Figura 2.2 y definir un período T para calcular el nivel equivalente.

Trabajando en tiempo discreto, con muestras equiespaciadas, y considerando constante la presión en el intervalo entre muestras se obtiene una expresión para el L_{eq} como se muestra a continuación [13].

$$L_{eq} = 10\log_{10}\left(\frac{1}{N}\sum_0^N \frac{p_i^2}{p_0^2}\right)\ (\text{dB}) \tag{2.3}$$

definiendo $P = \sum_0^N p_i^2$ también conocido como la potencia de la señal y α_{dB} una constante de calibración que absorbe factores como la sensibilidad del micrófono a utilizar y la presión de referencia se llega a la Ecuación 2.4.

$$SPL = \alpha_{dB} + 10 \times \log_{10}(P)\ (\text{dB}) \tag{2.4}$$

A efectos de diseñar un sonómetro[1] funcional a partir de un micrófono, basta calcular la potencia de la señal adquirida por el micrófono y luego compensar la constante α_{dB} que ha de obtenerse mediante la calibración contra un sonómetro de referencia.

Existen caminos para calcular la potencia de una señal, que resultan equivalentes en virtud del teorema de Parseval. En el dominio del tiempo, con muestras $y[n]$ de la señal se tiene:

$$P = \sum_{0}^{N} |y[n]|^2 \tag{2.5}$$

mientras que dominio de la frecuencia

$$P = \frac{1}{N} \sum_{0}^{N} |Y(k)|^2 \tag{2.6}$$

donde $Y(k)$ son las muestras de la DTFT de la señal. A continuación se muestra cómo proceder para cada uno de los dominios.

2.1.2.1. Cálculo de SPL en el dominio de la frecuencia

Las curvas de ponderación establecidas por la IEC 61672-1 admiten las siguientes expresiones en dominio de Laplace:

$$H_A(s) = G_A \frac{\omega_4^2 s^4}{(s+\omega_1)^2 (s+\omega_2)(s+\omega_3)(s+\omega_4)^2} \tag{2.7}$$

$$H_C(s) = G_C \frac{\omega_4^2 s^2}{(s+\omega_1)^2 (s+\omega_4)^2} \tag{2.8}$$

dónde $\omega = 2\pi f$ y según el estándar $f_1 = 20{,}6$ Hz, $f_2 = 107{,}7$ Hz, $f_3 = 737{,}9$ Hz y $f_4 = 12194{,}0$ Hz. Las ganancias $G_A = -2{,}0$ dB y $G_C = 0{,}062$ dB están definidas para que el filtro tenga una ganancia de 0 dB a 1 kHz. [14]

A partir de la expresión analógica $H(s)$ se puede construir la representación digital H $[k]$ en dominio de frecuencia. El procedimiento consiste en multiplicar en el dominio de frecuencias la transformada de Fourier discreta (DFT) de la señal con el filtro H para luego integrar la potencia de la señal filtrada.

Los resultados de realizar una simulación en Python se muestran en la Figura 2.3 y la 2.4. Se aplica un filtro de ponderación a partir de la curva C a una señal de audio de muestra, y se obtiene una señal filtrada. Particularmente se nota la eliminación de la frecuencia de continua.

2.1.2.2. Cálculo de SPL en el dominio del tiempo

Para calcular el SPL en el dominio del tiempo se requiere implementar los filtros de ponderación a través de su respuesta al impulso. Para ello se recurrió a coeficientes recursivos y no recursivos para aproximar las curvas en un intervalo

[1] Dispositivo capaz de dar una medida del SPL.

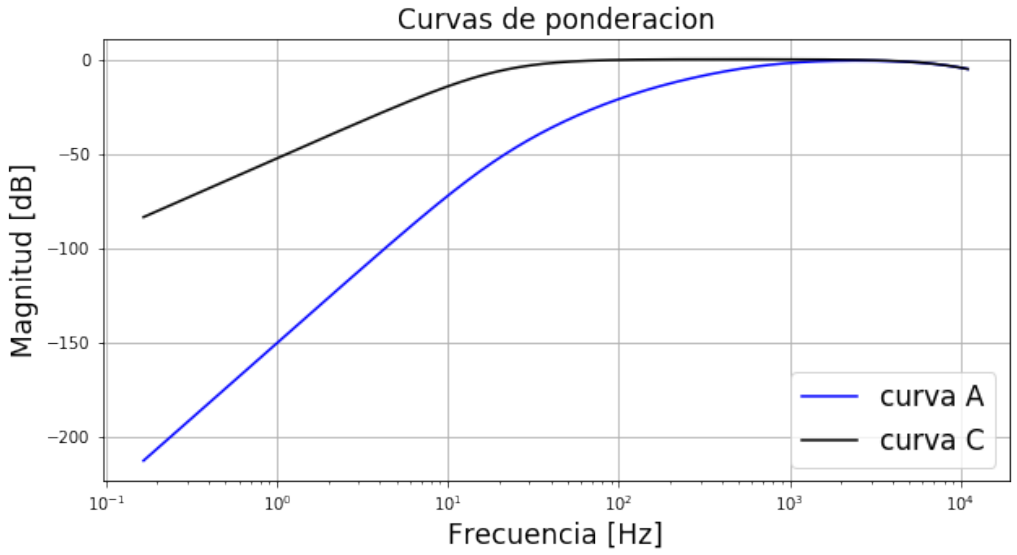


Figura 2.3: Curvas de ponderación implementadas en python con las respuestas $H(j\omega)$.

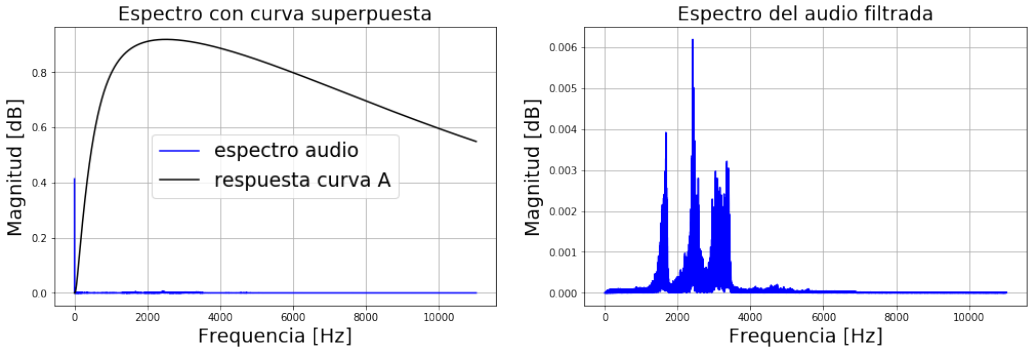


Figura 2.4: Ponderación en frecuencia según la curva A para una señal de audio de muestra. Se observa la cancelación de la continua luego de filtrar.

de frecuencias desarrollados por Rimell, Andrew N et al [14]. Se estudió la implementación de las curvas A y C por filtros de orden 10 y 14 respectivamente, la cantidad indicada por los mismos autores.

La implementación por medio de coeficientes recursivos no es costosa en términos computacionales y puede implementarse en un sistema de baja capacidad de cómputo [15]. Sin embargo presenta varios problemas que se deben solucionar. En primer lugar, como se observa en la Figura 2.5, la aproximación es buena en un rango de frecuencias, alejándose del estándar fuera de ese intervalo e introduciendo incertidumbre adicional a la medida (Al alejarse en ciertos rangos de frecuencia los aportes en energía en esos rangos se ven afectados y por lo tanto los resultados obtenidos de SPL y de bandas Mel van a tener una incertidumbre extra asociada).

A su vez, la transferencia del filtro $H(z)$ presenta polos en $z \approx -1$. La implementación en un microprocesador requiere de truncamiento de los coeficientes,

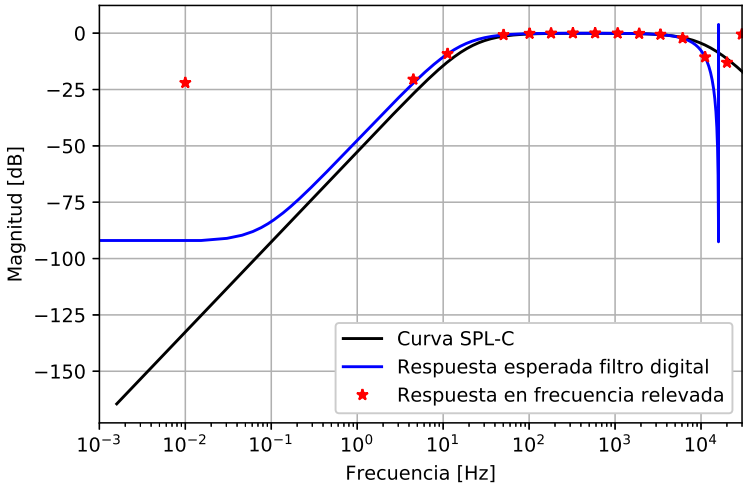


Figura 2.5: Implementación de filtrado IIR para cálculo de SPL en un microprocesador. Se observa en negro la curva C del estándar, en azul la curva de respuesta en frecuencia del filtro IIR, y en rojo la respuesta relevada del microprocesador ante señales de muestra sinusoidales. [15]

proceso que puede llevar a un corrimiento de polos volviendo el filtro inestable [15]. Si bien esto no es un problema crítico en un sistema con capacidad de procesamiento como la mini-PC, es poco recomendable trabajar con un filtro que puede hacerse inestable por problemas de representación numérica.

Este problema de alta sensibilidad a cuantización de los coeficientes es propio de los filtros de alto orden. Esto es mucho menos problemático con los filtros de primer y segundo orden; por lo tanto, los filtros de orden superior generalmente se implementan como filtros de segundo orden en cascada (SOS). [16].

2.2. Técnicas de representación tiempo-frecuencia

2.2.1. STFT - Transformada de Fourier en tiempos cortos

Dada una señal en dominio del tiempo $x(t)$, se define como su transformada de Fourier en tiempo continuo a la función de variable compleja:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \tag{2.9}$$

En un sistema digital, se cuenta con muestras $x[n]$ de una señal continua real $x(t)$. En virtud del teorema de Nyquist-Shannon [17]; para que una señal discreta sea capaz de reconstruir una señal continua de banda limitada, se debe cumplir que la tasa de muestreo f_s verifique:

$$f_s > 2f_{max} \tag{2.10}$$

con $f_{max} = \max_f \{f \setminus X(f) \neq 0\}$ es decir, la máxima frecuencia de la señal continua. Siendo que trabajaremos con sonido, cuya máxima frecuencia audible es de aproximadamente $f_{max} \approx 22000$ Hz, con una frecuencia de muestra de $f_s = 44100$ Hz, se garantiza la reconstrucción de todo el rango audible.

Siendo que trabajaremos en tiempo discreto, es necesario definir la transformada de Fourier en tiempo discreto. La misma queda definida a partir de la ecuación:

$$DFT_x(k) = X(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi}{N}} \tag{2.11}$$

La transformada de Fourier asume una señal estacionaria, ya que la descompone en sinusoides de duración infinita. Sin embargo, las señales de audio son típicamente no estacionarias, es decir, su contenido espectral varía con el tiempo. Por esto es necesario recurrir a representaciones mixtas, o representaciones tiempo-frecuencia (TFRs por sus siglas en inglés).

Un espectrograma [6] es un tipo de representación tiempo-frecuencia basada en la transformada de Fourier en tiempo corto (STFT, por sus siglas en inglés) de una cierta señal. Sea $x(t)$ la señal completa. Se define $w(t)$ (función ventana) como una función suave y de soporte acotado (en general gaussiana, Hanning o Hamming). La función $s^w_{t_0}(t) = x(t) \cdot w(to - t)$, conocida como la trama de señal (*frame*) centrada en tiempo t_0 bajo la ventana w es una versión localizada de $x(t)$ en un entorno determinado por el ancho de la ventana. El proceso de enventanado se observa en la Figura 2.6.

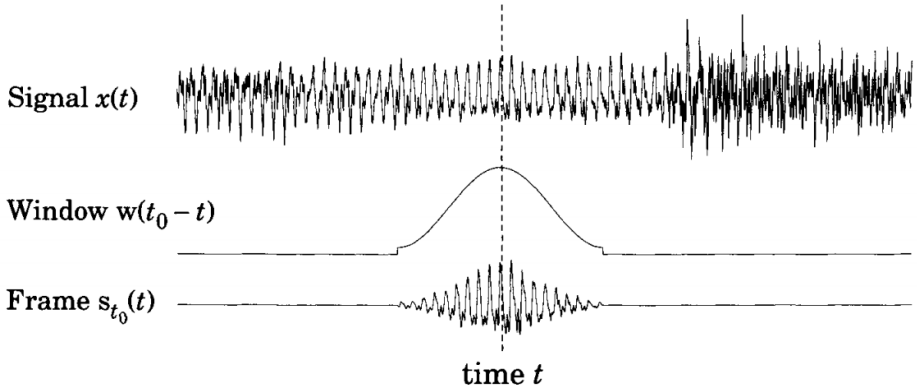


Figura 2.6: Se representa la señal $x(t)$ multiplicada por una función $w(t)$ de soporte acotado centrada en t_0. $s_{t_0}(t) = x(t) \cdot w(to - t)$ es una versión local de $x(t)$, y la FFT de s_{t_0} por lo tanto una versión local de la FFT de $x(t)$. Imagen extraída de Davy, Manuel et al [6].

A partir de la trama $s^w_{t_0}$, se construye $STFT^w_x(t_0, f) = S_{t_0}(f) = FT(s_{t_0}(t))$ (la transformada de Fourier en tiempos cortos de x bajo ventana w en tiempo t_0 y a la frecuencia f) que representa la información frecuencial de $x(t)$ en un entorno del tiempo t_0. El espectrograma es el valor absoluto de la STFT $SP = |STFT^w_x(t, f)|^2$.

La representación es altamente dependiente del largo de la ventana de análisis. La cantidad de puntos en el tiempo determina la cantidad de puntos en frecuencia. Por lo tanto, si la ventana es más chica, se reducen los puntos en frecuencia y eso

hace que la resolución en frecuencia sea menor. Para aumentar la resolución en frecuencia, se debe agrandar la ventana, lo que reduce la resolución temporal ya que el espectro de esa ventana es un promedio de las variaciones de frecuencia que ocurren en ese intervalo. Cuanto más corta es la ventana, más local es el análisis en el tiempo y por ende mayor resolución temporal.

2.2.2. Energía en bandas de frecuencia mel

La escala mel intenta imitar la respuesta no lineal de la percepción del oído humano presentando mayor resolución para bajas frecuencias y menos para altas frecuencias. Esta escala se define como [18]:

$$\text{mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.12}$$

Para filtrar y calcular la energía en bandas mel, se construyen K_{mel} filtros en frecuencia (del orden de 40) centrados equiespaciadamente en escala mel (espaciados exponencialmente en escala lineal). Los filtros son pasabandas triangulares, con los límites de un filtro coincidiendo con la frecuencia central del filtro siguiente. Es posible asignar una respuesta unitaria en la frecuencia central como se representa en la Figura 2.7, y también existe la opción de normalizar la altura de forma que cada triángulo tenga igual área.

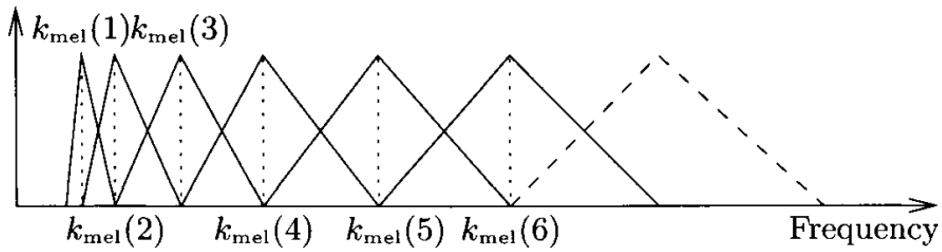


Figura 2.7: Banco de filtros mel. Los mismos tienen forma triangular, con los limites coincidiendo con la frecuencia central del filtro siguiente. Figura extraída de Davy, Manuel et al [6].

Para cada filtro mel, las componentes de frecuencia en su banda pasante son ponderadas por la respuesta del filtro, luego elevadas al cuadrado y sumadas. Por lo tanto, para cada instante t_0 de la STFT, se obtienen K_{mel} coeficientes representando la energía en las bandas mel. Nuestro dispositivo calcula y tramsite estos coeficientes, delegando posteriores filtrados al lado receptor.

El cálculo de energías en bandas mel es un punto intermedio en el cálculo de coeficientes cepstrales (MFCCs), muy utilizados como descriptores para algoritmos de reconocimiento de voz, en los cuáles no entraremos en detalle.

2.3. Técnicas de filtrado de señales

Tanto para el cálculo de SPL como para la calibración de micrófono que se explica en el Capítulo 3, es necesario filtrar señales por medio de aplicación de un filtro lineal.

En esta sección se explican las distintas técnicas existentes y su pertinencia para el objetivo.

2.3.1. Convolución con respuesta al impulso

La convolución lineal de dos señales discretas $x[n]$ y $h[n]$ de $\mathbb{Z} \to \mathbb{R}$ queda definida de la forma más general [19] por:

$$y[n] = x * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{2.13}$$

Cabe notar que la Ecuación 2.13 está definida para funciones de cualquier soporte. El soporte de una función es el rango de valores para el cual una señal toma valores no nulos.

De ahora en más se asumirá que $x[n]$ (a la que se llamará señal de entrada) tiene soporte N y $h[n]$ (*kernel*) tiene soporte $M < N$. De la ecuación se deduce que $y[n]$ (salida) tiene soporte $N + M - 1$.

2.3.2. Convolución FFT

La convolución a partir de la FFT usa la propiedad de que multiplicación en dominio de frecuencia corresponde con convolución en dominio temporal. La señal de entrada se transforma a dominio de frecuencia a partir de la DFT, se multiplica con la respuesta en frecuencia del *kernel* y luego retorna al dominio temporal utilizando la DFT inversa como se indica en la Ecuación 2.14, donde $X[k]$ y $H[k]$ son las FFTs de la entrada y el *kernel*.

$$y[n] = x * h[n] = \text{iDFT}(X[k] \cdot H[k]) \tag{2.14}$$

A partir del desarrollo de la transformada rápida de Fourier (FFT), realizar convolución a partir de la Ecuación 2.14 puede resultar más rápido que a partir de la definición. El resultado final es idéntico pero el número de operaciones es menor debido a que el algoritmo es más eficiente. Por esta razón la convolución FFT es conocida como convolución de alta velocidad. [20]

Para no tener el efecto de aliasing es necesario rellenar de ceros las señales hasta un largo de (al menos) $M + N - 1$ previo a tomar la FFT. De esta forma, la señal obtenida al antitransformar tiene el largo esperado por la Ecuación 2.13. [21]

La ventaja de realizar convolución FFT frente a convolucionar a partir de la definición se aprecia para *kernels* de convolución ($h[n]$) de un tamaño superior a 60 muestras [20]. El número exacto depende del hardware particular, pudiendo ser entre 40 y 80 muestras.

2.3.3. Método de solapamiento y suma

Hay muchas aplicaciones en procesamiento de señales donde una señal larga debe filtrarse en segmentos. Por ejemplo, el audio digital de alta fidelidad requiere una velocidad de datos de aproximadamente 5 MB/min, mientras que el video digital requiere aproximadamente 500 MB/min. Con velocidades de datos tan altas, es común que las computadoras no tengan suficiente memoria para retener simultáneamente toda la señal a procesar. También hay sistemas que procesan segmento por segmento porque operan en tiempo real. En otras aplicaciones, el procesamiento puede requerir que la señal esté segmentada.

El método de solapamiento y suma se basa en una técnica usual del procesamiento de señales:

- Descomponer la señal en componentes de menor tamaño.
- Procesar cada uno de los componentes.
- Volver a combinar los componentes procesados y obtener la señal de salida.

Para el método es clave cómo la convolución afecta el largo de las señales. Se parte la señal de entrada en segmentos que luego se rellenan de ceros de manera de aplicar la convolución FFT para procesar cada una con el *kernel*. En virtud de la linealidad del procesamiento, al sumar la salida para cada segmento se obtendrá la salida para la señal original. El método se conoce como de solapamiento y suma, ya que debido a que cada segmento procesado pasa a tener un largo $N_{segmento} + M - 1$ por lo que se obtiene un solapamiento temporal entre cada segmento procesado al sumar. [20]

La Figura 2.8 muestra un ejemplo del método de solapamiento y suma. La figura (a) es la señal que se va a filtrar mientras que (b) muestra el *kernel* de filtro que se va a usar. Al saltar a la parte inferior de la figura, (i) muestra la señal filtrada, una versión suavizada de (a).

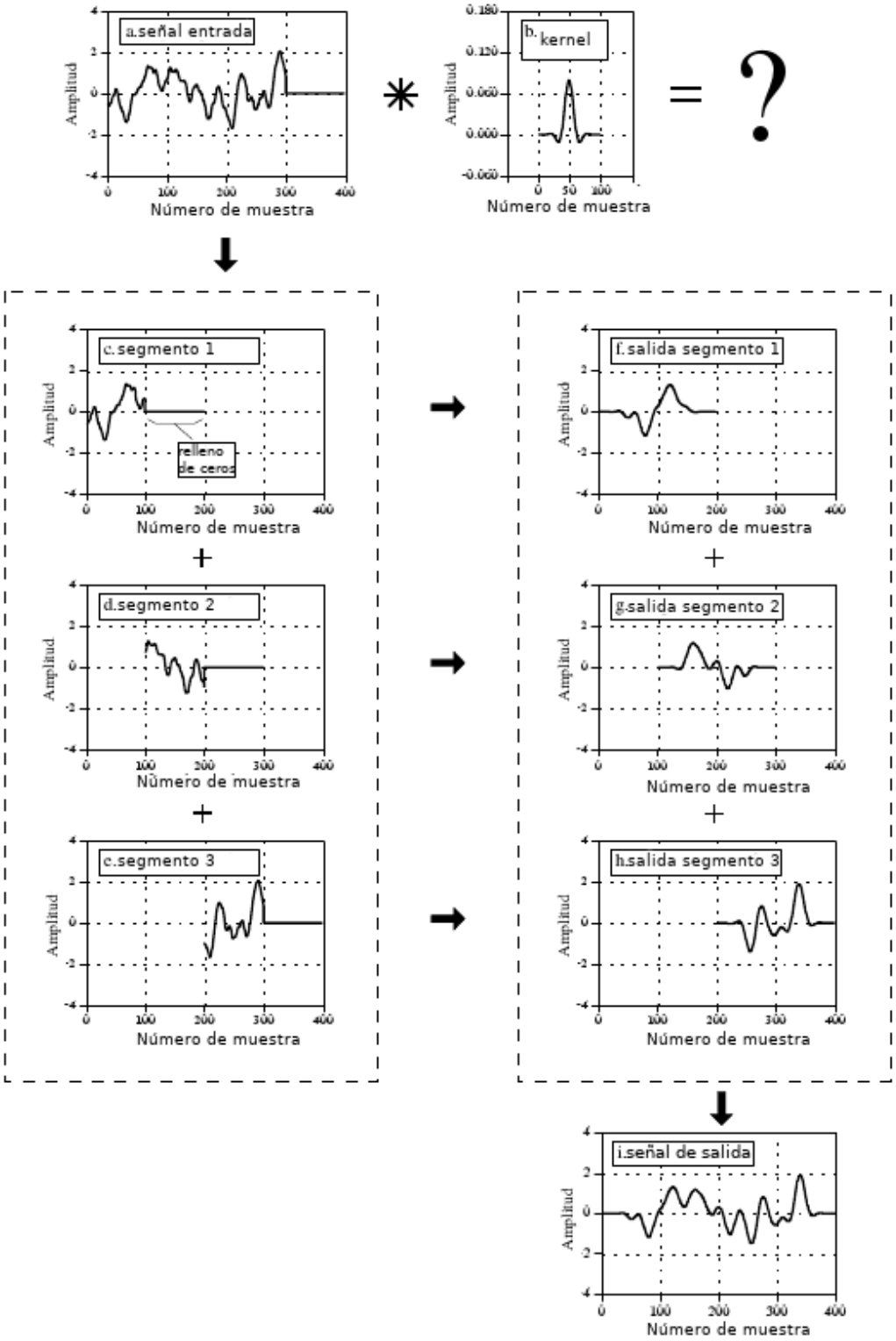


Figura 2.8: Filtrado de señal de entrada por método de solapamiento y suma. La señal (a) se separa en múltiples segmentos (c,d,e) que se convolucionan con el *kernel* (b). Los segmentos filtrados solapados (f,g,h) son luego sumados para obtener la salida (i), que equivale a convolucionar (a) con (b). Figura extraída de [20] (traducida).

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Calibración del micrófono

Uno de los requerimientos planteados en este proyecto es calibrar el micrófono a utilizar. Es necesario compensar la respuesta en frecuencia del micrófono y a su vez desarrollar un procedimiento para tener un SPL de referencia. La compensación en frecuencia resulta relevante ya que al pretender realizar un análisis de la señal de audio en el dominio de la frecuencia, es importante que se vea afectada lo menos posible por la respuesta en frecuencia del micrófono.

En este capítulo se explicará el mecanismo utilizado para obtener la respuesta en frecuencia del micrófono y el mecanismo para calibrar el instrumento, tanto en frecuencia como en SPL, a partir de dispositivos de referencia. A su vez se presentarán los resultados de cada experiencia.

3.1. Compensación en frecuencia

La compensación en frecuencia se basa en comparar la respuesta ante un estímulo conocido del micrófono a calibrar y de uno de referencia, como se esquematiza en la Figura 3.1.

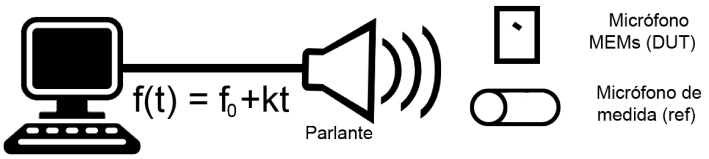


Figura 3.1: Esquema conceptual de la calibración. Se estimula con una función conocida un micrófono de referencia y el micrófono que se desea calibrar.

3.1.1. Dispositivos a utilizar

Para la calibración del micrófono a utilizar contamos con los siguientes instrumentos:

- Micrófono de referencia dbx RTA-M
- Tarjeta de audio Focusrite Scarlett 2i2
- Audacity: software para la grabación y reproducción de audio
- Parlante TANNOY
- Sonómetro de referencia Extech 407736
- Espuma acústica

3.1.2. Montaje experimental y consideraciones

Debido a las limitaciones experimentales se optó por realizar un experimento de calibración de tipo *free field* (campo libre) contra un micrófono de referencia. Existen otras formas de calibración que necesitan equipos especializados como las calibraciones de campo de presión, o de reciprocidad [22].

La calibración por campo libre es un método de sustitución. En lugar de excitar simultáneamente ambos micrófonos, el micrófono de referencia muestrea el campo sonoro, luego lo hace el micrófono a ensayar (de ahora en más DUT, *Device Under Test*), y luego se calcula la respuesta deseada. Para que esto funcione, el campo que reciben el micrófono de referencia y el DUT deben ser cuasi-idénticos. Se deben tener en cuenta varias consideraciones que se listan a continuación para lograr que esto ocurra en una calibración por sustitución. [23]

Es altamente recomendable tomar esta medida en una sala muy grande, o preferentemente, una cámara anecoica. El piso de ruido de la sala debe ser del orden de 30 dB (referenciado a 20 μPa) o menos. La norma ISO 3745 tiene un anexo que describe las condiciones de medición de un ambiente apto para campo libre. La temperatura, la humedad y la presión ambiental deben ser bastante estables, ya que cada uno de ellos puede tener un efecto en la respuesta de los micrófonos. Es altamente recomendable rastrear estos parámetros a lo largo de la calibración, para compensar cualquier cambio resultante en el campo sonoro.

La calibración se realiza exponiendo el micrófono de referencia al campo de sonido, tomando una medición, y luego usando el DUT para medir el campo (idealmente) idéntico. Luego de calcular la respuesta en frecuencia de ambas mediciones al campo libre, se resta la respuesta de la referencia de la del DUT para obtener la respuesta propia del DUT.

Se realizó un montaje como se muestra en la Figura 3.2 teniendo particularmente en cuenta algunas consideraciones vertidas en la literatura al respecto.

Es deseable que la fuente de sonido sea simétrica radialmente respecto a la normal de los diafragmas [23] [24], por lo que se coloca el micrófono alineado con el eje del parlante. En este sentido, para priorizar la ubicación del micrófono en dicho lugar es preferible realizar una grabación independiente para cada micrófono en contraposición a grabar en ambos micrófonos simultáneamente en lugares contiguos [24].

Figura 3.2: Montaje para calibración. Se utiliza un parlante con perfil plano y espuma acústica para minimizar el rebote en la mesa. Se coloca el micrófono en un soporte horizontal en el cual se coloca en dos instancias independientes el micrófono de referencia y el DUT.

Siendo que las ondas de sonido difractan en las superficies [23] y ocasionan picos no deseados en la respuesta en frecuencia, se cubre la superficie de trabajo con espuma acústica a efectos de minimizar este fenómeno.

Se grabaron múltiples iteraciones del experimento tanto para el micrófono de referencia como el DUT, para tener datos de comparación y verificar la consistencia del experimento. Es posible realizar un análisis estadístico para varias repeticiones del experimento si bien se demostrará y es argumentado en varias publicaciones que realizar un promediado estadístico no presenta mejoras (y es hasta no recomendado en algunas circunstancias). Es más relevante tener un estímulo (por ejemplo barrido en frecuencia o *sweep*) de larga duración por lo que se utilizará un minuto, valor que aparece en la literatura [25] [26].

También es relevante considerar las frecuencias del barrido. El rango de interés es el espectro audible, de 20 Hz a 20 kHz; pero se recomienda extenderlo, por ejemplo de 10 Hz a 22 kHz para minimizar efectos transitorios [27].

3.1.3. Chirp y sincronización

Se realiza una calibración preliminar del micrófono compensando la respuesta en frecuencia ante un micrófono de medida. Para esto se realiza un ensayo de respuesta a un *chirp* (señal con variación en frecuencia), estimulando ambos micrófonos. Eligiendo una frecuencia inicial de 10 Hz y una final de 22 kHz se releva el rango de interés.

Existen dos tipos de *chirp* usuales, el lineal y el logarítmico. Por los argumentos vertidos en el trabajo de Farina [25], anteriormente citado se resolvió emplear un *chirp* logarítmico para el procesamiento, cuya expresión temporal es:

$$x(t) = \sin\left[\frac{\omega_1 \cdot T}{\ln\left(\frac{\omega_1}{\omega_2}\right)} \cdot \left(\exp\left(\frac{t}{T} \cdot \ln\left(\frac{\omega_1}{\omega_2}\right)\right) - 1\right)\right], \tag{3.1}$$

siendo ω_1 y ω_2 las frecuencias (angulares) inicial y final del *chirp* (en rad/s) y T la duración total del barrido en segundos.

Para obtener la respuesta en frecuencia a partir de la señal adquirida basta calcular la ganancia por ventanas para un entorno de cada instante (cada instante se corresponde a una frecuencia según Ecuación 3.1). Una vez relevadas ambas respuestas se construye un filtro de compensación para aplicar en futuras medidas al micrófono digital.

La correspondencia tiempo-frecuencia es muy sensible a sincronización. Siendo que las dos señales son adquiridas por fuentes distintas y en instantes de tiempo distintos, es necesario sincronizarlas antes del procesamiento. Para esto, se calculó la correlación cruzada entre las tiras de datos y se le aplicó un corrimiento temporal a la señal del micrófono a ensayar hasta maximizar la función correlación. Esto se ilustra en la figura 3.3

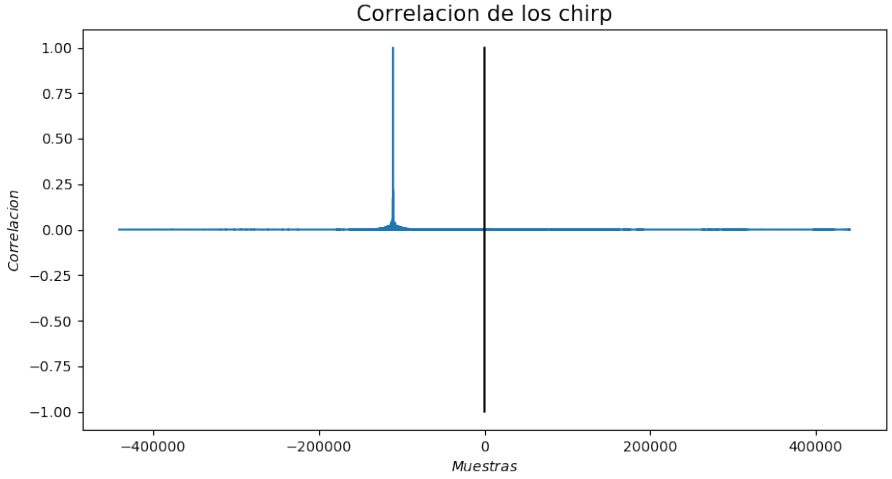


Figura 3.3: Correlación temporal entre grabación del micrófono de referencia y bajo ensayo según corrimiento temporal de señal del DUT. La correlación se maximiza cuando las señales están sincronizadas. Es determinante tener una sincronización temporal para que valga la correspondencia tiempo-frecuencia de la Ecuación 3.1.

3.1.4. Tratamiento estadístico

Para minimizar los efectos de ruido ambiente y dispersiones puntuales en un ensayo, se realizó el experimento múltiples veces realizando un tratamiento estadístico. De esta manera, se repitió 5 veces cada experimento y se promediaron los resultados; no percibiéndose una mejora de la respuesta en frecuencia; como ya se había argumentado.

A su vez, se ensayaron diversas duraciones de *chirp*. Un *chirp* largo tiene la ventaja de tener mejor resolución (en tiempo y en frecuencia por la correspondencia entre ellos) pero puede llevar a que para cada frecuencia se logren estabilizar patrones de resonancia en la sala. Se realizó la experiencia con *chirps* de duración

10 segundos, 1 minuto y 3 minutos de manera de comparar posibles diferencias. Se resolvió utilizar *chirps* de 1 minuto como sugiere la bibliografía consultada ya que presenta un buen compromiso entre resolución en frecuencia y tiempo de procesamiento posterior [25].

3.1.5. Resultados

El filtro de compensación necesario para equiparar la respuesta del DUT al de referencia se calcula como:

$$H_{comp}(f) = \frac{G_{ref}(f)}{G_{DUT}(f)} \tag{3.2}$$

donde $G_{ref}(f)$ y $G_{DUT}(f)$ son los promedios de las respuestas en frecuencia obtenidas para el micrófono de referencia y el de ensayo respectivamente.

Los resultados de este procedimiento para el caso de *chirp* exponencial de un minuto se ilustran en la Figura 3.4 y 3.5.

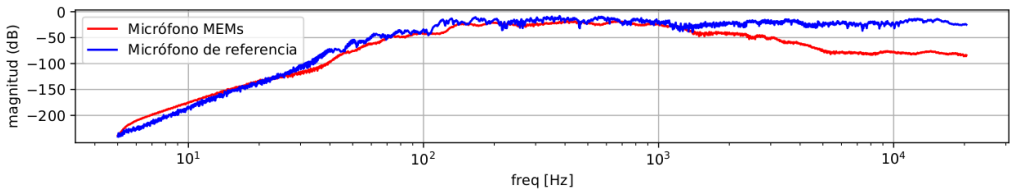


Figura 3.4: Respuesta en frecuencia calculada para el micrófono de referencia (azul), el DUT (rojo) y corrimiento de offset para tener ganancia 0 dB en banda plana.

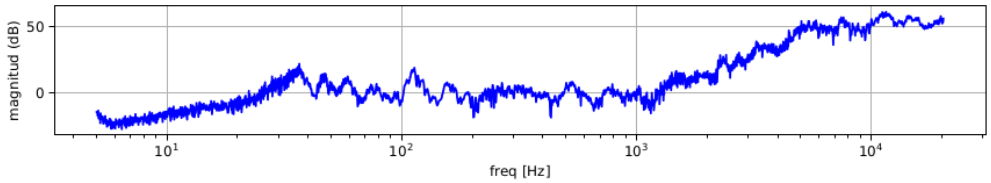


Figura 3.5: Respuesta de compensación obtenida de calcular la transferencia del micrófono de referencia al DUT.

En coherencia con lo expuesto por Mydlarz, Charlie et al [3], se encuentra un comportamiento plano en la banda $40 - 1000\ Hz$ y un comportamiento creciente en altas frecuencias asociado a las posibles resonancias de la placa sobre la cual se monta el micrófono y el soporte que se utilizó para el mismo (a altas frecuencias la longitud de onda del sonido es comparable con las dimensiones de estos elementos macroscópicos). En la bibliografía también se recomienda regularizar el filtro de forma de evitar aplicar atenuaciones/amplificaciones muy altas en baja y alta frecuencia para no amplificar el ruido de fondo.

3.2. Calibración del SPL

3.2.1. Procedimiento

Para la calibración de las rutinas de cálculo de SPL fue necesario utilizar un sonómetro de referencia.

Se procedió emitiendo con el parlante una señal de frecuencia fija ($f = 1$ KHz) y variando la amplitud en sucesivas repeticiones. Se realizó en esta instancia un barrido discreto: se grabó el audio y se registró el valor devuelto por el sonómetro para 10 amplitudes distintas de la señal de audio, repitiendo a su vez 5 veces cada experiencia.

Para realizar la calibración de forma adecuada, siendo que se supone que el SPL calculado por nuestro algoritmo sobre la grabación del micrófono (de ahora en más SPL_{DUT}) debería quedar a un factor de escala aditivo del de referencia (SPL_{ref}), es necesario hallar el factor de escala α que minimiza la distancia entre $SPL_{DUT} + \alpha$ y SPL_{ref}. Esto se puede formular como:

$$\hat{\alpha} = \text{argmin}_{\alpha}(||SPL_{DUT} + \alpha - SPL_{ref}||^2) \tag{3.3}$$

Donde SPL_{DUT} y SPL_{ref} son los vectores de valores de SPL a diferentes amplitudes para una tirada. Para hacer uso de la repetición estadística se utiliza la siguiente definición para el α:

$$\hat{\alpha} = \text{argmin}_{\alpha} \left\{ \Sigma_i \left(||SPL^i_{DUT} + \alpha - SPL^i_{ref}||^2 \right) \right\} \tag{3.4}$$

Donde se calcula el α que minimiza la suma de las compensaciones en cada tirada.

3.2.2. Resultados

De realizar el procedimiento explicado se llega a lo expuesto en la Figura 3.6 en la cual se constata la hipótesis de que las dos cantidades difieran a menos de un factor de escala, presentando el mismo comportamiento funcional ante la variación de amplitud.

Esta forma de calibrar la medida de SPL permite garantizar una medida con buena fidelidad respecto a un instrumento profesional.

De aplicar esta corrección de escala se llega a una diferencia media entre lo medido por el sonómetro profesional y lo que devuelve nuestro método de 1,04 dB. A su vez se obtiene una diferencia máxima de 4,08 dB.

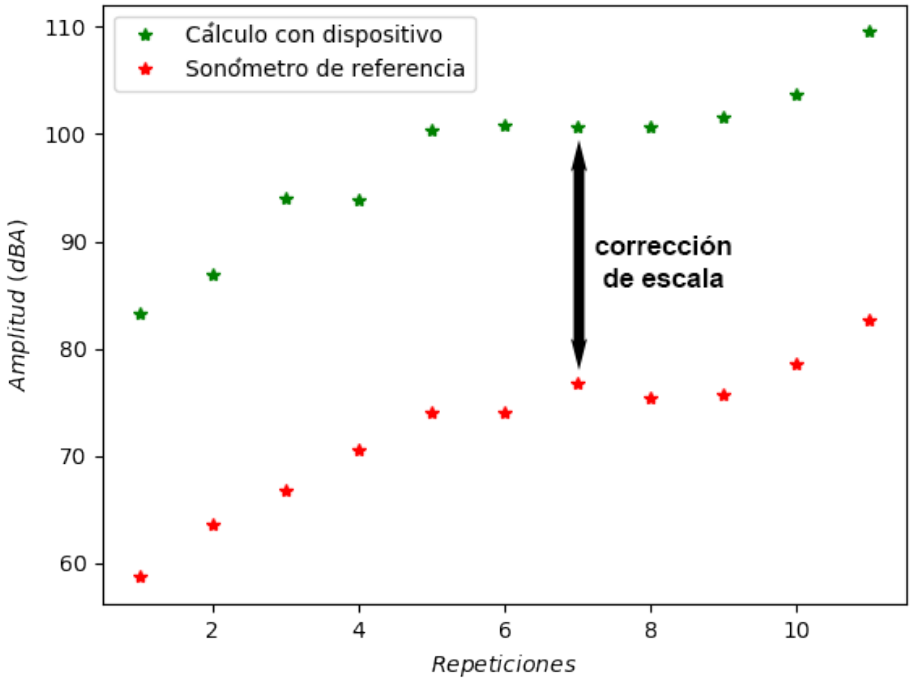


Figura 3.6: SPL calculado a partir de procesamiento de datos del micrófono, versus SPL de sonómetro de referencia, y representación de obtención del factor de escala α.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Diseño y montaje del dispositivo

En este capítulo se describe la elección de los componentes utilizados para el armado del equipo detallando el motivo de uso de los mismos, una comparación con otras opciones existentes en el mercado y las adaptaciones que se les realizaron.

En el Apéndice A se encuentra una guía de ensamblado por lo que este capítulo solo se centra en los criterios de elección de los distintos componentes. A su vez en el Apéndice D se desglosa el costo de los componentes del dispositivo, resultando en un total de 160 USD en origen, con sugerencias para reducirlo.

4.1. Componentes del dispositivo

En las secciones siguientes se detalla la elección o diseño de los diversos elementos que componen el dispositivo. A saber:

- La mini-PC que realiza el procesamiento de la señal.
- El micrófono encargado de realizar la grabación.
- El gabinete en que se contiene la mini-PC y otros componentes para protegerlos.
- Los distintos componentes accesorios al micrófono, particularmente el *gooseneck* que lo separa del gabinete y permite maniobrarlo.
- El soporte en el que se apoya el micrófono.
- El protector antiviento (windshield) para el micrófono.

4.2. Mini-PC

Para el procesamiento de la señal de audio se estudió la opción de utilizar un microcontrolador o una mini-PC. Las dos posibilidades son apropiadas en términos de consumo tamaño y costo.

Si bien el uso de un microcontrolador podría permitir una reducción de precio, las tareas de procesamiento que se deben realizar resultan costosas de desarrollar para este tipo de plataformas y los protocolos de comunicación difíciles de implementar. Sin embargo, se estudió que el uso de un microprocesador es adecuado para realizar una versión reducida en tareas del proyecto, por ejemplo el cálculo de SPL. [15]

Se optó por el uso de una mini-PC ya que para estas existe abundancia de bibliotecas desarrolladas y soporte, y el uso de un microcontrolador presenta varias dificultades prácticas. Para la elección de la mini-PC a utilizar se realizó un estudio comparativo que se encuentra en la Tabla 4.1. Para las computadoras se tomó como insumo la velocidad de procesamiento, la memoria RAM, soporte para Wi-Fi (W) o Bluetooth (β), capacidad de almacenamiento, y el tipo y cantidad de puertos USB. Los precios de las mini-PC varían en un amplio rango, por lo que se debió elegir teniendo en cuenta este factor.

Se optó por comprar la Raspberry Pi 3 B+ que cuenta con 4 puertos USB, es de bajo costo (35 USD) y cuenta con suficiente cantidad de memoria. A su vez, presenta facilidades para la migración a plataformas aún menos costosas (como Raspberry Pi Zero) y es una plataforma probada y popular, por lo que se dispone de copiosa literatura que hace sencillo resolver problemas conocidos.

Tabla 4.1: Comparación entre distintos modelos de mini-PC. Precios a mayo de 2018.

Nombre	CPU	RAM	W-β	USB	US$
Raspberry Pi 3 B+	Q-1.4GHz	1Gb	✓ - ✓	4x2.0	$\approx$ 35
NanoPi Neo Plus2	Q-1.5Ghz	1Gb	✓ - ✓	2x2.0	$\approx$ 45
Nano PC-T3	O-1.5GHz	2 Gb	✓ - ✓	4x2.0	$\approx$ 75
ODROID-XU4	O-1.5GHz	2 Gb	x - x	2x3.0+1x2.0	$\approx$ 60
ODROID-C2	Q-1.5GHz	2 Gb	x - x	4x2.0	$\approx$ 46
Orange Pi Lite 2	Q-1.5GHz	1 Gb	✓ - ✓	1x3.0+1x2.0	$\approx$ 30
Orange Pi Plus 2	Q-1.5GHz	2 Gb	✓ - x	4x2.0	$\approx$ 50
Rock64	Q-1.5GHz	1-2 Gb	x - x	1x3.0+2x2.0	$\approx$ 25-35
BananaPi M1+	D-1.5GHz	1Gb	✓ - x	2x2.0	$\approx$ 35
BananaPi M64	Q-1.5GHz	2Gb	✓ - ✓	2x2.0	$\approx$ 65
Asus SBC Tinker	Q-1.8GHz	2Gb	✓ - ✓	4x2.0	$\approx$ 75
Pine A64	Q-1.5GHz	1Gb	x - x	2x2.0	$\approx$ 19

Se verificó a su vez que dicha plataforma cuenta con GPIOs adecuados para la conexión del micrófono, particularmente pines de salida de 3,3 V.

4.3. Micrófono

Los micrófonos de tipo MEMS son de creciente interés debido a su diseño versatil, buena inmunidad a interferencia de radiofrecuencias y electromagnéticas, su bajo costo y buen comportamiento bajo distintas condiciones ambientales. [3]

Por estos motivos, se estudió la posibilidad de utilizar micrófonos MEMS de tipo analógico y digital. En la Tabla 4.2 se comparan la banda pasante de frecuencias, la relación señal ruido (SNR), la distorsión armónica (THD) y el rechazo a la fuente de alimentación (PSR) para distintos micrófonos.

Constatando que no existen mayores diferencias entre los valores de distorsión y calidad de señal entre micrófonos analógicos y digitales, se optó por comprar un micrófono MEMS digital para evitar el uso de tarjetas de audio. Los MEMS digitales no presentan especificaciones sustancialmente diferentes a los de un micrófono analógico y se conectan directamente a la Raspberry.

Se eligió trabajar con el micrófono SPH0645LM4H-B que presenta banda pasante en el rango de interés (20 Hz - 20 kHz), distorsión (THD = 1 %) y calidad de señal (SNR = 65 dB y PSR = −80 dB). Si bien todos los micrófonos cuentan con características similares, para este micrófono fue posible adquirir un *breakout* [1] diseñado y comercializado por Adafruit [28].

Con base en el *breakout*, y para su tamaño y perforaciones se realizó el diseño de soporte del micrófono como se explica en la sección correspondiente.

Nombre	Comunicación	Banda Pasante (Hz)	SNR	THD*	PSR
TDK ICS43432	I^2S	50-20k	65 dB	1 %	-80 dB
TDK INMP441	I^2S	60-15k	61 dB	3 %	-75 dB
KNOWles - SPH0645LM4H-B	I^2S	20-20k	65 dB	1 %	-80 dB
TDK ICS43434	I^2S	60-20k	65 dB	1 %	-99 dB
TDK ICS40618	Analog	50-20k	67 dB	0,2 %**	-85 dB
TDK INMP510	Analog	60-20k	65 dB	1 %	-78 dB
ADMP401	Analog	100-15k	62 dB	3 %	-70 dB

Tabla 4.2: Comparativa entre micrófonos en stock.
* Valores máximos de distorsión.
** Valor típico de distorsión.

4.4. Gabinete

El gabinete (ver Figura 4.1) es el espacio físico donde se integra todo el sistema, conteniendo tanto el micrófono y la Raspberry como los elementos que permiten la interconexión de estos. Contiene a su vez la alimentación de la Raspberry y mecanismos para la comunicación con la misma, ya sea por Wi-Fi o por Ethernet.

La caja estanca debe contener:

- La alimentación del sistema.
- La Raspberry Pi 3 B+.
- Salida Ethernet.
- Antena Wi-Fi.

[1] placa sobre el cual se monta el micrófono

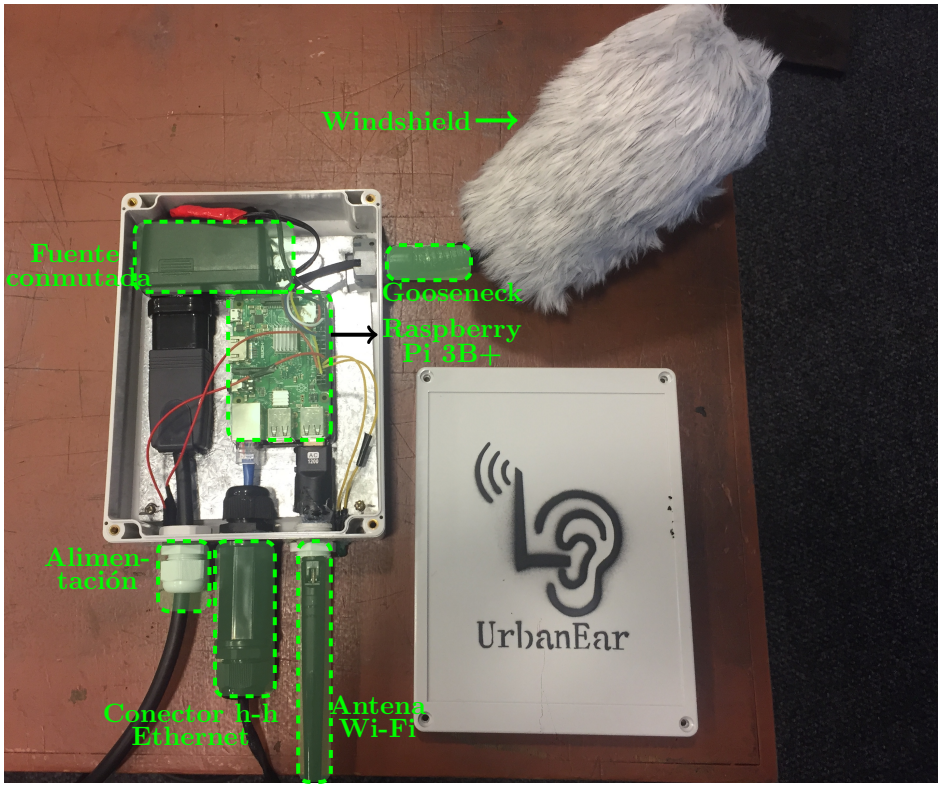


Figura 4.1: Contenido del gabinete en versión final del dispositivo.

- Salida para colocar el *gooseneck* y el micrófono.

El objetivo básico para la caja es que pueda almacenar los componentes en su interior, siendo del menor tamaño posible. Utilizando una caja de 15 cm × 20 cm de base y 5,5 cm de altura se logró contener los elementos necesarios. Otro requisito a cumplir es la estanquidad del dispositivo. El gabinete utilizado (Figura 4.2) es estanco, pero es necesario verificar su estanquidad luego de concluido el armado del dispositivo como se detalla en el Capítulo 6. En ese capítulo se explican los procedimientos por los cuales se verificó esta hipótesis; concluyendo en el uso de la caja de la Figura 4.2.

Para la conexión por Ethernet se utilizó un conector hembra-hembra (ver Figura 4.3) a prueba de agua. Este conector permite tener un pequeño cable Ethernet adentro del gabinete y tener un puerto hembra exteriorizado. El puerto hembra exterior posee un prensacable que logra que una vez conectado el cable Ethernet al dispositivo externo, se aísle la ficha del cable de cualquier exposición a inclemencias climáticas.

Para mejorar la conexión inalámbrica, se optó por extender el rango de Wi-Fi. A estos fines se colocó la antena Wi-Fi correspondiente a la Figura 4.4 con interfaz USB para conectarse a la Raspberry. Esta antena fue configurada de manera que la Raspberry emita su propia señal Wi-Fi, permitiendo que el servidor se conecte a través de esta red. La configuración de la antena y del servidor se detallan en

Figura 4.2: Caja utilizada como gabinete para el dispositivo. Imagen extraída de Ebay.

Figura 4.3: Conector hembra-hembra utilizado para exteriorizar la conexión Ethernet de la Raspberry. Imagen de Amazon.

Capítulo 5.

Para la alimentación del dispositivo se utilizó la fuente conmutada de la Figura 4.5. La salida de la fuente conmutada se modificó de forma de conectarse a los pines de 5 V y GND de Raspberry . La fuente contaba con una salida mini-USB que se resolvió cortar. Es posible alimentar la Raspberry con esta entrada pero no resultaba cómodo a efectos de disposición.

La fuente conmutada se alimenta de una conexión $220V$ AC $(50Hz)$ que es

Figura 4.4: Antena Wi-Fi utilizada para extender el rango de alcance para el dispositivo. Imagen de Amazon.

una de las salidas de la caja. Al interior del gabinete, la conexión se realiza con un enchufe hembra.

Figura 4.5: Fuente conmutada Canakit para la alimentación de la Raspberry. Imagen extraída de Amazon.

4.5. Gooseneck

Una de las aspiraciones para el producto final es lograr cierta direccionabilidad del micrófono, a los efectos de percibir el sonido con la mayor fiabilidad posible ya que la percepción del micrófono no es la misma en todas las direcciones [28].

Con este fin se adquirió un *gooseneck* (cuello de ganso) estándar de 16 cm el cual cumple con la maniobrabilidad deseada. El *gooseneck* no poseía características de permeabilidad admisible para las condiciones del proyecto por lo que fue necesario modificarlo para conseguir cierta resistencia al agua. En concreto, se logró que una incidencia directa de agua corriente en el exterior del gooseneck no ingrese a su interior, como se detalla en el Capítulo 6.

Para lograr la estanquidad se ensayaron distintas soluciones como cubrirlo de neopreno o PVC, pero estas resultaron poco prácticas debido al aumento de peso y/o la reducción de maniobrabilidad. Se optó finalmente por rellenar el *gooseneck* con espuma de poliuretano, material aislante e impermeable que se expande en contacto con el aire. El procedimiento consiste en cubrir de espuma por dentro y por fuera del *gooseneck* de manera que toda apertura resulte cubierta. Además, este material ha sido utilizado para impermeablizar cajas para efectos de procesamiento de audio [29].

4.6. Soporte para el micrófono

Figura 4.6: Pieza impresa para soportar el micrófono y enroscar con el gooseneck.

Definido el *gooseneck* y el micrófono a utilizar, fue necesario diseñar un soporte que conecte con el *gooseneck* y permita apoyar y conectar el micrófono (ver esquema de Figura 4.6).

Para el diseño del soporte se siguieron las consideraciones encontradas en la bibliografía respecto a resonancias y protecciones [3]. Del soporte a diseñar se desea:

- Que no sea completamente cerrado, ya que en una cavidad cerrada en presencia de un campo sonoro se produce el fenómeno de resonancia de Helmholtz. [30]
- Proteger al micrófono de una incidencia directa sobre él de la lluvia. Para esto es preciso que el soporte posea un alero.
- Poder colocar el micrófono de forma sencilla y segura en el soporte.
- Poder colocar el soporte de forma sencilla y segura en el *gooseneck*.
- Que se puedan pasar los cables de forma sencilla y segura a través del soporte y hacia el micrófono.

Para cumplir con los requerimientos mencionados antes, se tomó como referencia de partida el soporte utilizado por el proyecto SONYC [3]. Este modelo presenta soluciones para los problemas anteriormente listados pero con un diseño para un micrófono diferente al que se definió para este proyecto. Se adaptó el mecanismo de colocación del micrófono al *breakout* del SPH0645LM4H-B (el *breakout* usado presenta dos perforaciones para colocarlo, mientras que el usado por SONYC cuenta con cuatro).

Para comunicar el soporte con el *gooseneck*, basta con imprimir en el modelo una rosca compatible con la rosca hembra del mismo. El *gooseneck* presenta una rosca hembra de 5/8" por lo que se utilizó una rosca macho de esa medida ya diseñada y disponible para acceso público [31]. El soporte se diseñó en el programa Blender [32] de licencia libre, basándose en los diseños del soporte y la rosca ya mencionados. Para generar el archivo de impresión requerido por la impresora Prusa i3 RepRap [33], que se encuentra disponible para su uso en el Instituto de Ingeniería Eléctrica, se utilizó el software Cura [34] partiendo del diseño 3D. En este software se especifican las condiciones de trabajo para la impresora, así como el material a utilizar, que en este caso es un filamento de 1.75 mm para una temperatura de entre $220 - 260°C$. El modelo que se diseñó se puede apreciar en la Figura 4.7.

4.7. LEDs y Windshield

Los distintos componentes detallados se conformaron en el gabinete estanco con un total de 4 perforaciones mayores hacia el exterior (alimentación, Ethernet, antena, micrófono) como se observa en la Figura 4.1.

A su vez, se realizaron dos agujeros pequeños para ubicar LEDs, de colores verde y rojo, a efectos de indicar el estado del equipo (el LED verde se prende cuando el dispositivo está grabando y el rojo cuando el dispositivo está encendido). La conexión del micrófono a los pines de la Raspberry se realizó siguiendo la guía

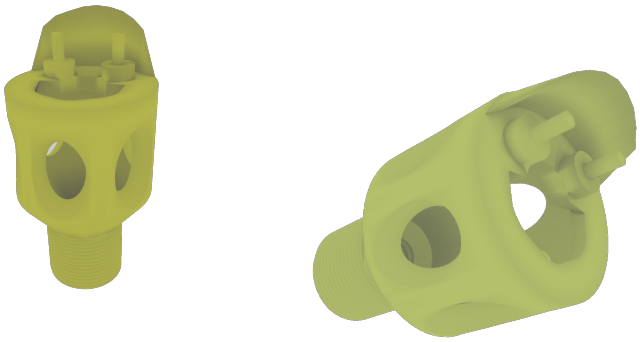


Figura 4.7: Soporte 3D diseñado en Blender.

disponible en Adafruit [28]. Los detalles del conexionado de los LEDs a los pines de la Raspberry y detalles prácticos en general se pueden ver en el manual de armado del Apéndice A.

La fuente conmutada y la Raspberry están contenidas dentro del gabinete sobre una superficie plana formada por una chapa de acero. Ambos se encuentran en una unión rígida a la misma que impide movimientos.

Se colocó un *windshield* al micrófono como el de la Figura 4.8, para garantizarle protección al micrófono del viento y el agua en condiciones moderadas. A su vez, esta solución es de compromiso entre protección y perdida de sensibilidad; lo que no descarta otras posibles soluciones como instalar la caja debajo de un techo para evitar el contacto con la lluvia.

Figura 4.8: Windshield de pelo artificial. Imagen extraída de Amazon

Capítulo 5

Software

El software presente en el dispositivo UrbanEar es una parte central del proyecto. Contiene tanto librerías de código abierto disponibles públicamente como código implementado a los efectos del proyecto. Por software del proyecto se entiende el código que implementa los algoritmos de procesamiento, particularmente grabación de la señal de audio, filtrado de compensación de la respuesta del micrófono, estimación del SPL, y cálculo de la energía en bandas Mel. En este capítulo se detalla dicho software y además se menciona el sistema operativo que corre sobre el Raspberry 3 B+, la instalación de los drivers necesarios para usar el micrófono MEMS, la configuración del equipo como Access Point (AP) de WiFi independiente, y la implementación de una interfaz en el servidor que permite visualizar los datos obtenidos y realizar cambios en la configuración.

El Raspberry Pi 3 B+ tiene instalado Raspbian, un sistema operativo basado en Debian específicamente construido para la plataforma Raspberry. El SO es relativamente fácil de utilizar por línea de comando y no presenta ninguna limitación seria a los planes del proyecto. En este sistema se instalaron las herramientas necesarias para correr el código de procesamiento. En primera instancia se instaló BerryConda [35], que es una versión para Raspberry de Conda, un manejador de paquetes, dependencias y entornos. Sobre el entorno manejado por BerryConda se instaló:

- Python 3.6
- Numpy 1.14.0
- SciPy 1.0.0
- SoundDevice 0.3.11
- PyYaml 3.11

Además se instalaron otros paquetes necesarios por ser dependencias de las librerías mencionadas.

Para la toma de datos es necesario solucionar la adaptación del micrófono MEMS, el cual se comunica con la Raspberry mediante el protocolo $I2S$, por lo

que se deben instalar drivers que permiten a la Raspberry comunicarse con el micrófono como una tarjeta de audio [28]. Se utiliza la librería SoundDevice [36] la cual toma los datos de audio a partir de la tarjeta ficticia y los convierte en arreglos en formato `.npy`.

Los algoritmos de procesamiento de este dispositivo se encuentran en su totalidad implementados en Python. La elección de este lenguaje se debe a las facilidades que este provee para el procesamiento de señales; además de tratarse de un software libre y gratuito. Las librerías de cálculo numérico y científico NumPy y SciPy proveen múltiples funciones útiles para el procesamiento de audio.

5.1. Algoritmos de procesamiento

Para el procesamiento requerido existen librerías de libre acceso como LibROSA [37] diseñadas específicamente para procesamiento de audio. En este trabajo se utilizaron ideas de dicha biblioteca y otras disponibles junto con desarrollo de código propio. Esto se debe a que LibROSA presenta problemas de compatibilidad con la arquitectura de Raspberry y a que es más provechoso en términos formativos desarrollar el código de procesamiento de forma transparente y comprendiendo cada paso. De todas formas, en el Capítulo 6 se detallan comparaciones entre los resultados obtenidos utilizando LibROSA y el código implementado en este proyecto.

La Figura 5.1 representa un diagrama de flujo del procesamiento que realiza el equipo cuando está activo. Además, en paralelo se tiene al servidor adquiriendo los datos que se van guardando y liberando espacio en la Raspberry.

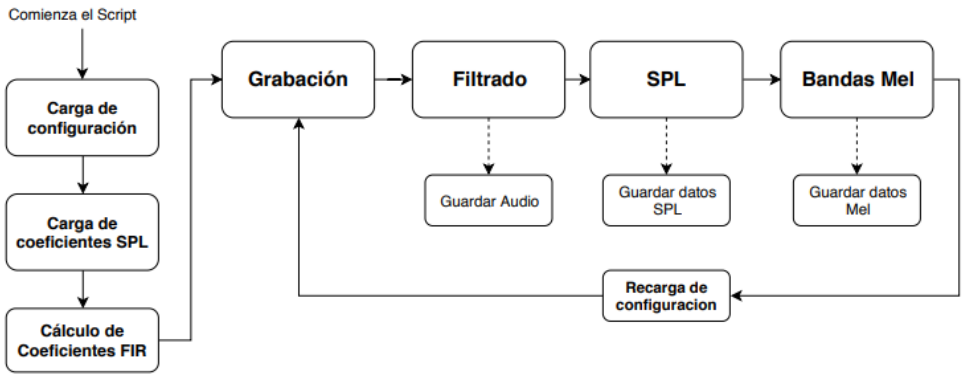


Figura 5.1: Flujo de procesamiento de datos.

El procesamiento consiste en un bucle infinito de grabación y procesamiento de datos a partir de un script de Python. Este bucle se mantiene en funcionamiento con otro script auxiliar escrito en Bash que se ejecuta periódicamente y se encarga de verificar que el script principal se mantiene funcionando y lo vuelve a ejecutar en caso de que por algún motivo se corte la ejecución. A su vez inicia el bucle cuando se conecta el equipo.

Como inicialización, el programa carga la configuración desde un archivo en formato YAML, un estándar para serialización de datos. Esta configuración se actualiza en cada iteración para permitir cambios de los parámetros de procesamiento sin detener el bucle. Con los parámetros determinados, el primer paso consiste en calcular los coeficientes de las curvas SPL y los coeficientes que conforman el filtro FIR, el cual se utiliza para calibrar la respuesta en frecuencia del micrófono. Estos cálculos se realizan fuera del bucle ya que solo se necesitan una vez, a menos que se reconfiguren ciertos parámetros como la frecuencia de muestreo, en cuyo caso se debe reiniciar el script y recalcular los coeficientes FIR y de SPL. Luego de estos pasos previos, se inicia un bucle sin fin que consta de cuatro grandes pasos:

- Grabación de audio
- Filtrado de compensación
- Cálculo de SPL
- Cálculo de Bandas Mel

Se describen los pasos a continuación.

5.1.1. Grabación de audio

El primer punto del contenido del script, es la grabación de audio a través del micrófono MEMS utilizando la librería SoundDevice de Python. La rutina toma como parámetro la cantidad de segundos útiles T que se desean grabar para luego procesar. Debido a que el micrófono presenta un transitorio no deseado en el primer segundo de grabación, se graba $T + 1$ segundos y se descarta el primer segundo de la grabación

5.1.2. Filtrado de compensación

Luego de obtener la señal de audio en un vector, se pasa a filtrar la señal para compensar la respuesta en frecuencia del micrófono utilizado. Para esto se analizaron 3 posibilidades en función de la eficiencia.

El primer método consiste en aplicar la convolución lineal de la señal de audio y el filtro FIR ya mencionado. Este método, si bien es muy sencillo de implementar, es extremadamente ineficiente para filtros de tamaño significativo ($\gg 64$ muestras), por lo que se descartó de inmediato.

El segundo método se basa en que la convolución de dos señales en el tiempo equivale a una multiplicación en frecuencia de las representaciones espectrales de la señal ($FFTs$). Como se explica en el Capítulo 2, para los largos de señales que se manejan, el método que resulta en menor overhead de procesamiento[1] resulta el método de *solapamiento y suma* o *Overlap-Add Method*.

[1]Tiempo que debe existir entre el final de una grabación y el comienzo de otra debido al procesamiento de datos.

La ventaja de este método radica en que procesando la señal de a pedazos, se necesita menos espacio en memoria (RAM) para almacenar los resultados intermedios que si se procesa la señal entera. A partir de cierto largo de señal,las ventajas en uso de memoria compensan el sobrecosto operacional de particionar y luego realizar sumas para llegar al resultado. Siendo que la memoria RAM es la principal limitante de la Raspberry Pi, este método lleva a una importante reducción del tiempo de procesamiento, como se detalla en Capítulo 6.

Para efectos de depuración, la configuración actual presenta la funcionalidad de guardar los datos de audio en formato WAV. Como se explicó en el Capítulo 1 no es admisible en términos legales y éticos contar con un dispositivo capaz de grabar datos de audio del espacio público, sin embargo resulta muy útil para una etapa de prueba poder contar con ese insumo.

Una vez filtrado, si se utiliza la funcionalidad de guardado de WAVs, se almacenan las sucesivas grabaciones hasta llegar a un cierto tope de almacenamiento. En ese momento el equipo comienza a sobrescribir los archivos de audio más viejos de forma circular, evitando así que se sature la capacidad de almacenamiento en caso de que el servidor esté un período extendido sin extraer datos del dispositivo.

5.1.3. Cálculo de SPL

El cálculo del valor de SPL se realiza siguiendo el marco teórico presentado en la Sección 2.1 por lo que se explica de forma breve.

Este proceso consiste en separar la señal de entrada en N segmentos correspondientes a los N valores que se desean calcular del audio grabado. Cada partición pasa por el filtro que implementa una de las curvas A ó C (Figura 2.2).

Para la implementación en el dominio temporal del cálculo se utiliza un filtro IIR. Este filtro es altamente sensible a la cuantificación de sus coeficientes, por lo que el filtro se construye como una cascada de filtros de segundo orden los cuales se aplican sucesivamente al audio.

Luego del filtrado, se calcula la potencia en el vector resultante que corresponde al nivel equivalente en ese intervalo de tiempo. La Raspberry almacena el vector `.npy` que contiene los valores de SPL calculados con ambas curvas.

5.1.4. Cálculo de energía en bandas Mel

El último procesamiento a realizar sobre el audio grabado es el cálculo de energía en bandas Mel. Nuevamente las consideraciones teóricas están detalladas en el Capítulo 2, por lo que se hará una breve descripción del algoritmo implementado.

Un paso previo en este algoritmo es pasar la señal de audio por un filtro de pre-énfasis en altas frecuencias, esto básicamente para balancear el espectro ya que los valores en altas frecuencias suelen ser de magnitud mas baja y por lo tanto también ayuda a evitar errores de cálculo computacional.

A continuación se aplica la STFT al audio y se calcula la potencia en cada segmento. Luego se generan los bancos de filtros según los parámetros configurados, que son el número de bancos y los límites superior e inferior en frecuencia. Este banco se multiplica por la matriz de la STFT para y se multiplican ambas matrices obteniendo una representación de la energía en bandas de frecuencia de la señal de audio. Por último, se guarda el resultado de dicha multiplicación en formato `.npy` de la misma forma que los audios y los datos de SPL.
Este algoritmo fue tomado de [38] y se repite cíclicamente, con la opción de configurar un tiempo inactivo donde el equipo no grabe ni procese datos. Esta capacidad puede ser útil si se desea tener grabaciones mas espaciadas en el tiempo sin necesitar que el servidor central tenga que retirar los datos constantemente.

5.2. Comunicación con servidor

A partir de lo que se desprende de la sección anterior, los datos generados por los algoritmos de procesamiento son:

- Vector en formato `.npy` con los valores de SPL para las curvas A y C.
- Matriz en formato `.npy` con los valores de energía en bandas para cada ventana de tiempo procesada.
- Forma de onda de la señal de audio en formato `.wav`.

Estos datos se deben transferir al servidor central que luego es el encargado de generar estadísticas y gráficas sobre dichos datos. Para esta transferencia se determinó usar el protocolo SFTP [39], por su robustez y por la seguridad que provee, la cual es imprescindible en el caso de estar transfiriendo el audio sin procesar. Un esquema de la interacción servidor-dispositivo se ilustra en la Figura 5.2.

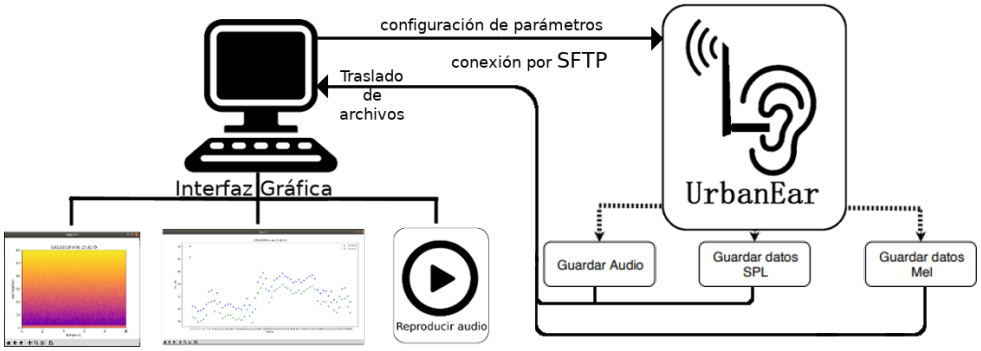


Figura 5.2: Interacción servidor-dispositivo. El dispositivo genera los archivos de interés que el servidor recupera por medio de una conexión SFTP. Los archivos se pueden visualizar/reproducir a través de una interfaz gráfica.

El servidor periódicamente (con un período parametrizable) inicia una conexión SFTP con el Raspberry, y toma los archivos mencionados anteriormente de diferentes carpetas y los envía a un repositorio local, eliminándolos de la memoria del Raspberry. El Raspberry cuenta con una tarjeta SD de 16GB, de la cual se pueden considerar disponibles para almacenamiento de datos 10 GB. Esto implica una eventual limitante si el servidor por algún motivo no puede tomar los datos del equipo y este sigue adquiriendo. Para esto, se diseñó un sistema de adquisición circular, también parametrizable en cantidad, que simplemente consiste en almacenar una cantidad dada de cada tipo de archivo (SPL, bandas Mel y audios) y cuando se llega a este tope se elimina el archivo con fecha más antigua para hacer espacio al archivo nuevo. De esta forma se pierde la menor funcionalidad posible en caso de mal funcionamiento de la conexión servidor-Raspberry y se tiene un cierto almacenamiento local en el equipo que permite no necesitar una conexión constante. De todas formas cuanto más amplio sea el período en que se traen los archivos, más pesado es el tráfico de datos debido a la acumulación de los mismos.

5.3. Configuración como AP

La comunicación entre el nodo y el servidor es a través de Wi-Fi o por cable Ethernet. En el caso de la conexión inalámbrica, usualmente la comunicación entre equipos se da mediante un Access Point (AP), que se encarga de conectar equipos entre sí en una WLAN (Wireless Local Area Network), como lo hace un *switch* para el caso de las conexiones por Ethernet o Fibra Óptica.

Para el proyecto presente, el alcance del mismo considera un solo nodo y un servidor central, que para la conexión Ethernet, implica que no sea necesario un *switch* y se considere una conexión cableada punto a punto. La comunicación inalámbrica es similar, en el sentido de que es excesivo pensar en tener un AP dedicado a la conexión entre un nodo y un servidor. Utilizar un AP externo (Ej: uno de los que brinda Wi-Fi en la FIng) genera la incomodidad de depender de la cercanía de los dos equipos a conectar y el AP para obtener una señal estable (también genera la dependencia de disponer del AP en cuestión). Para solucionar esto existen dos opciones: una conexión "Ad-Hoc", que es básicamente el equivalente a la conexión punto a punto cableada, o que uno de los equipos presentes haga el trabajo de AP. La primer opción se descartó debido a que para "Ad-Hoc" en Raspberry-Pi el único método de encriptación compatible encontrado es WEP, el cual ya no es recomendado por ser vulnerable.

Se tomó la opción de establecer como AP el nodo, aunque también es posible que la Raspberry sea el servidor. Se eligió implementar el AP en la Raspberry para mantener la configuración lo más posible en el dispositivo de forma de poder utilizar cualquier PC o Laptop con Wi-Fi como servidor.

Para realizar esto se utilizó *hostapd*, un *daemon*[2] que implementa IEEE 802.11 Access Point Management [41] entre otras cosas.

[2]Un daemon, servicio o programa residente es un tipo de proceso no interactivo, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario. [40]

Listing 5.1: Configuración necesaria del archivo hostapd.conf para que el dispositivo opere como un AP con la ssid y pass deseadas

```
interface = wlan0
hw_mode = g
channel1 = 7
wmm_enabled = 0
macaddr_acl = 0
auth_algs = 1
ignore_broadcast_ssid = 0
wpa = 2
wpa_key_mgmt = WPA-PSK
wpa_pairwise = TKIP
rsn_pairwise = CCMP
ssid = UrbanEarless
wpa_passphrase = m4r10k4rt
```

Junto a esto se configura en el archivo `dhcpcd.conf`, que es el encargado de la configuración de las interfaces de red, la red que se pretende utilizar y el rango DHCP lo da el programa *dnsmasq*. La configuración del archivo `hostapd.conf` es la que se muestra en el Listing 5.1. La guía de configuración se encuentra en [42] de los pasos 1 a 5.

5.4. Interfaz de usuario

Se realiza una interfaz de usuario con la librería Tkinter de Python en su versión 8.6 para el manejo del dispositivo. La biblioteca Tkinter resulta práctica y fácil de usar y permite mantener todo el código del proyecto en Python. Esto no descarta que en el futuro se utilice un entorno más atractivo y comercial para la interfaz.

La interfaz cuenta con dos funcionalidades principales: el control de la transferencia de archivos con la Raspberry y el despliegue de los datos obtenidos.

La interfaz tiene una presentación como se puede ver en la Figura 5.3 donde se presentan pestañas que permiten consultar la configuración de parámetros que se están utilizando y realizar cambios en los mismos. Se permite definir con qué escala se desea graficar las bandas Mel, el mecanismo para mostrar los valores de SPL obtenidos, ya sea un histórico o un archivo en particular y activar o desactivar el servidor. Cuando esta activo el servidor se realiza la consulta a la Raspberry por archivos nuevos y su descarga al dispositivo en el que corre la interfaz por protocolo SFTP. El estado de la comunicación se indica en la esquina inferior izquierda de la interfaz.

Así mismo la interfaz presenta dos botones centrales. El botón *Mostrar Datos* permite la representación de datos adquiridos en una ventana auxiliar (ver Figu-

Figura 5.3: Interfaz gráfica de usuario. Las pestañas de la ventana permiten configurar los distintos parámetros del dispositivo, mientras que el botón central *Mostrar Datos* habilita la representación de datos adquiridos.

ra 5.4 y Figura 5.5); mientras que el botón *Salir* desactiva el servidor y termina la ejecución del programa.

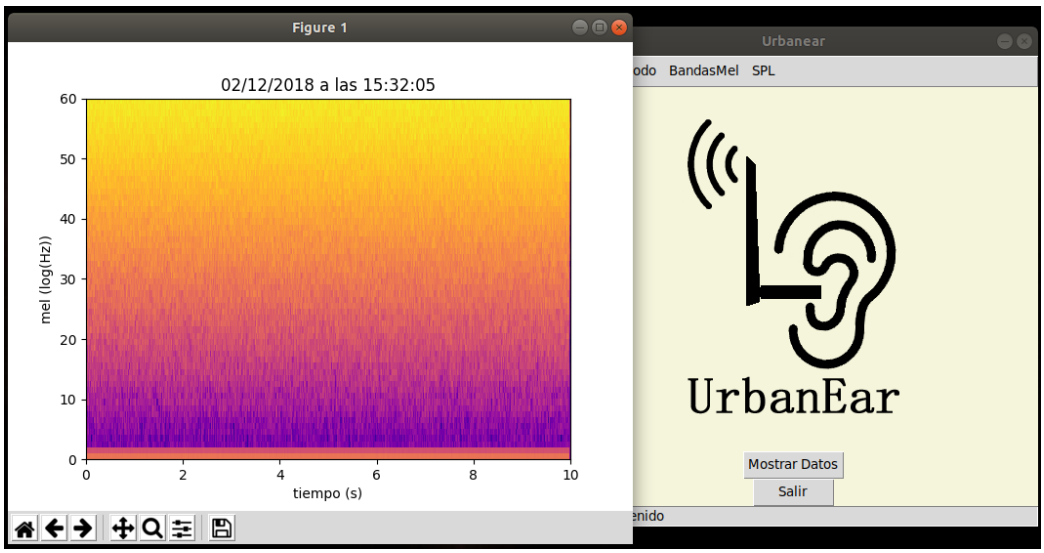


Figura 5.4: Gráfica de la energía en bandas de frecuencia según banda mel para una grabación de diez segundos comenzada el 2/12/2018 a las 15:32:05.

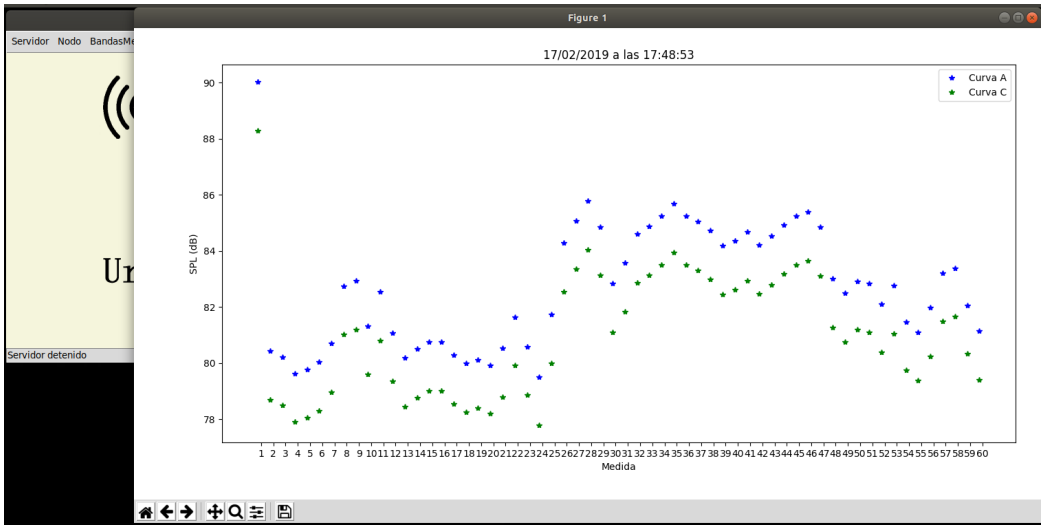


Figura 5.5: Gráfica de puntos de SPL para una grabación de un minuto comenzada el 17/02/2019 a las 17:46:53.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Ensayos

En este capítulo se detallan los distintos ensayos que se realizaron sobre el dispositivo para corroborar y cuantificar múltiples aspectos de su desempeño.

Se divide la exposición en dos secciones:

- **Ensayos de funcionamiento:** Se entiende por ensayos de funcionamiento los asociados al cumplimiento del funcionamiento proyectado del dispositivo. En particular se cuantificó el consumo de corriente del dispositivo, se verificó su correcto funcionamiento por un tiempo prolongado, y se verificó su resistencia al agua y el viento.
- **Ensayos de software:** Se entiende por ensayos de software a la caracterización de los resultados obtenidos, como tamaños de archivos, tiempos de procesamiento y datos procesados, directamente asociados al código desarrollado en el proyecto.

Una extensa evaluación del desempeño del dispositivo desarrollado permite verificar si se cumplieron los objetivos planteados para el proyecto y a su vez permite trazar líneas de trabajo a futuro.

6.1. Ensayos de funcionamiento

En esta sección se detallan los distintos ensayos que se realizaron para verificar que el dispositivo cumple los requerimientos mínimos de operación definidos para el proyecto. A su vez se cuantificaron distintos parámetros del funcionamiento.

- Se relevó el consumo de corriente del dispositivo en las condiciones de operación.
- Se verificó la consistencia de la grabación, procesamiento y comunicación para un período prolongado de tiempo comprobando la obtención de los archivos esperados y la robustez de la ejecución de los scripts implementados.
- Se verificó que el dispositivo quedó correctamente condicionado para operar en la intemperie. Particularmente se ensayó su desempeño frente al viento y el agua en un ambiente controlado.

6.1.1. Ensayo de consumo

Este ensayo fue realizado con el objetivo de cuantificar el consumo de potencia del sistema en funcionamiento. Para reducir el consumo de potencia, se utilizaron algunas sugerencias comunes en la bibliografía [43] como desactivar los LEDs de la placa (se cuenta con LEDs externos conectados a GPIOs), y otras funcionalidades innecesarias como HDMI. También en el ensayo se deshabilitaron los módulos de Wi-Fi (de la placa, no así de la antena/tarjeta externa) y BlueTooth.

Para medir la potencia se utilizó el montaje basado en un voltímetro y un amperímetro que se diagrama en la Figura 6.1 y se observa en la Figura 6.2. Se mide simultáneamente el voltaje de la fuente de alimentación a la Raspberry, así como la corriente que entrega dicha fuente.[1]

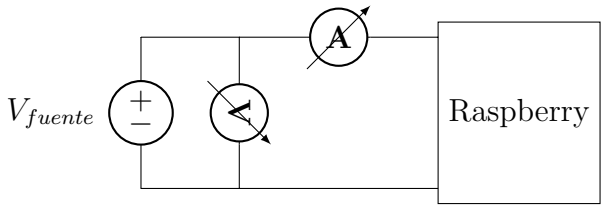
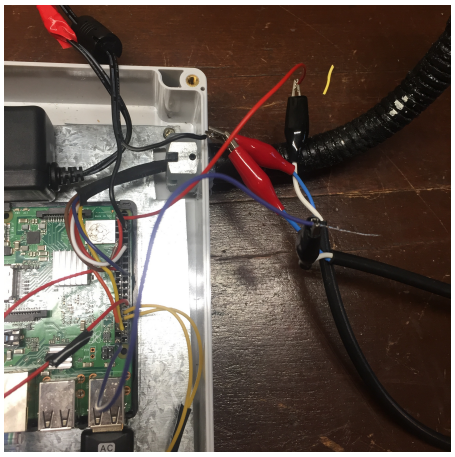
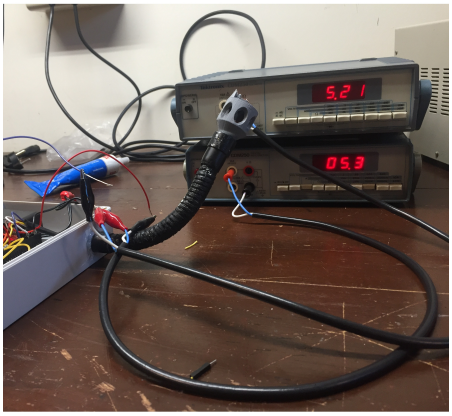


Figura 6.1: Montaje experimental para la medida de consumo de potencia.



(a) Conexiones a la fuente conmutada y Raspberry.



(b) Multímetro y amperímetro utilizados.

Figura 6.2: Montaje experimental para medida de potencia.

Se registraron los valores de la medida para el dispositivo en reposo y para el dispositivo procesando, calculándose la potencia para ambos valores como se indica en la Tabla 6.1.

[1]Nota: este ensayo no mide la posible disipación de potencia en la fuente conmutada. Se asume que dicha disipación resulta marginal en el total de potencia.

	V_{fuente}(V)	I(mA)	P(W)
RPi en reposo	$5{,}24 \pm 0{,}01$	460 ± 10	$2{,}41 \pm 0{,}05$
RPi grabando	$5{,}24 \pm 0{,}01$	490 ± 10	$2{,}57 \pm 0{,}05$
RPi procesando	$5{,}22 \pm 0{,}01$	680 ± 10	$3{,}55 \pm 0{,}05$

Tabla 6.1: Resultados para ensayo de potencia del dispositivo.

Los resultados son coherentes con los valores típicos de consumo de una Raspberry Pi 3B+, que tiene un consumo mínimo posible de 350 mA y máximo de 980 mA [44].

Si se desea reducir el consumo, se podría pensar en reducir el porcentaje de tiempo que se está grabando e implementar un LPM[2]. En las condiciones ensayadas, la operación del dispositivo durante un año consumiría aproximadamente 25,22 kWh, lo que equivale a 3,80 USD [3]

6.1.2. Comportamiento frente a la intemperie

6.1.2.1. Ensayos de estanquidad

Se realizaron ensayos para constatar la resistencia del dispositivo frente a la incidencia del agua. Particularmente se ensayó la estanquidad del *gooseneck*, de la tapa del gabinete, y de las aperturas (selladas con silicona) que se realizaron en el gabinete.

Gooseneck

Luego de utilizar poliuretano expandido para impermeabilizar el *gooseneck* como se detalla en el Capítulo 4 se verificó su estanquidad colocando una varilla de metal cubierta de papel absorbente en el interior del *gooseneck* y sometiéndolo a los efectos del agua. Se constató que al retirar la varilla el papel permanecía seco. El proceso se ilustra en la Figura 6.3

Tapa del gabinete

El gabinete cuenta con una goma que se coloca entre la tapa y el cuerpo principal sellando la juntura para prevenir la presencia de humedad en el interior de la caja. Para verificar que dicha unión fuera estanca, se utilizó un procedimiento análogo a lo realizado con el *gooseneck*, colocando papel del lado interior de la tapa y observando si el papel permanecía seco luego de hacer incidir agua sobre el gabinete. El proceso se ilustra en la Figura 6.4. Se pudo concluir que no se filtra agua a través de la tapa del gabinete.

[2] *Low Power Mode.*

[3] Tarifas y tasas de cambio de enero 2019. [45]

(a) Varilla de metal cubierta de papel absorbente.

(b) Incidencia externa de agua sobre el *gooseneck*, con la varilla en su interior.

Figura 6.3: Ensayo de estanquidad en *gooseneck*

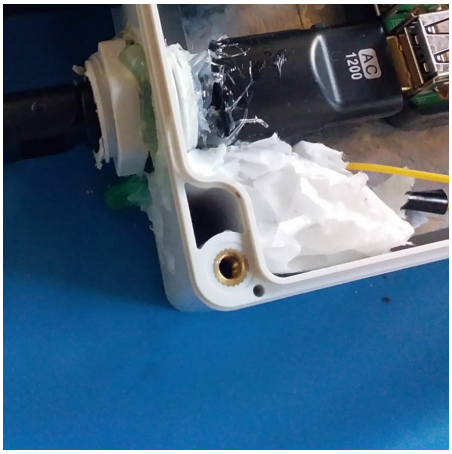
(a) Tapa del gabinete con papel para realizar prueba de estanquidad.

(b) Incidencia externa de agua sobre el gabinete.

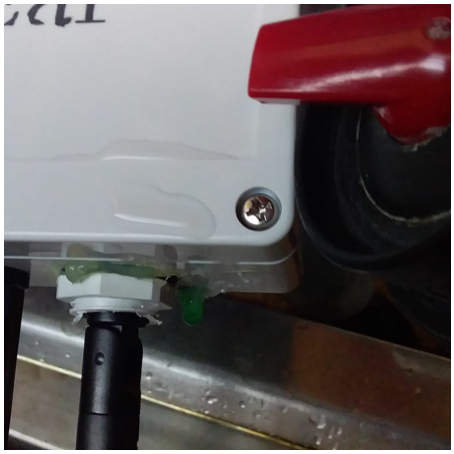
Figura 6.4: Ensayo de estanquidad a la juntura del gabinete.

Aperturas

Repitiendo el procedimiento detallado, se colocó papel absorbente en el interior del gabinete en contacto con las aperturas del mismo. Se verificó que el papel permanecía seco luego de realizar el experimento. El proceso se ilustra en la Figura 6.5.



(a) Apertura del gabinete con papel para realizar prueba de estanquidad.



(b) Incidencia externa de agua sobre el gabinete.

Figura 6.5: Ensayo de estanquidad de las aperturas.

Discusión

A pesar que los experimentos no cumplen normas establecidas (por ejemplo por la IEC-60592 [46]) para medir la estanquidad del dispositivo, se considera que los resultados obtenidos son suficientes para el desarrollo de este prototipo.

6.1.2.2. Efecto del Windshield

Se realizó una prueba sencilla para determinar el efecto del windshield (WS) en la reducción de los efectos del viento en las grabaciones.

La prueba, que se ilustra en la Figura 6.6 se basó en la reproducción de un sonido conocido (se utilizó un tono de $f = 1$ kHZ) a través de un parlante. El sonido fue grabado por el micrófono sin proteger y protegido por el WS (Figura 6.6a y Figura 6.6b respectivamente).

(a) Montaje para ensayar funcionamiento sin el WS.

(b) Montaje para ensayar funcionamiento con el WS

Figura 6.6: Ensayo de los efectos del windshield

Adicionalmente se grabó la respuesta ante ese tono adicionando un estímulo que pretende simular el efecto del viento. Para éste efecto, se generaron manualmente ráfagas de viento abanicando una superficie en frente del micrófono en las situaciones con y sin el WS. Este procedimiento, si bien poco riguroso ya que se desconoce con que velocidad se mueve el aire que incide sobre el micrófono, es útil a efectos de estudiar la influencia del choque de una corriente de aire con el micrófono y como afecta la calidad de la grabación.

Resultados

En el experimento definido anteriormente, se grabaron 4 archivos de audio para analizar. La duración de cada archivo de audio es de 10 segundos. Para cada uno de esos audios se calcularon distintos indicadores listados en la Tabla 6.2, a saber:

- $E_{audio} = \Sigma_n x[n]^2$ es la energía de la señal, en unidades arbitrarias.
- E_{notch} es la energía de la señal en unidades arbitrarias, luego de aplicarle un filtro *notch* digital. El objetivo de este parámetro es cuantificar la energía por fuera de la asociada al tono, es decir el ruido y el viento.

  El filtro *notch* utilizado es un filtro IIR con frecuencia de centro $f = 1$ kHZ y factor de calidad $Q = \frac{f_c}{bw} = 30$. Es posible utilizar un filtro "peine" para también filtrar los armónicos de 1 kHz, pero este resulta menos selectivo [47] y la energía de los armónicos resulta despreciable frente a la del tono fundamental.

- $\text{SNR}_{(\text{dB})} = 10 \log_{10}\left(\frac{E_{audio}}{E_{notch}}\right)$ es la relación en dB entre las dos energías anteriores.
- $\text{THD}_{(\text{dBc})}$ es la distorsión armónica de la señal.

	$E_{audio}(u.a.)$	$E_{notch}(u.a.)$	$SNR_{(dB)}$	$THD_{(dBc)}$
Sin WS sin viento	8625	42	23,13	−45,260
Sin WS con viento	8603	1118	8,86	−44,866
Con WS sin viento	7227	36	23,03	−47,344
Con WS con viento	6442	221	14,64	−47,279

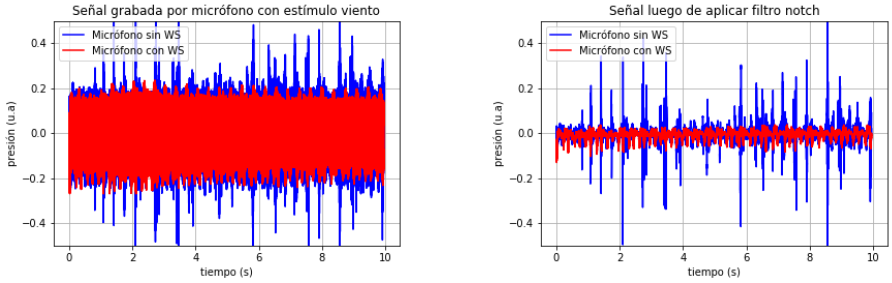
Tabla 6.2: Valores calculados a partir de las grabaciones para analizar la calidad del audio.

Los resultados de la Tabla 6.2 resultan esperables.

La energía en el caso de utilizar el WS es menor debido a la atenuación que introduce; particularmente se comprueba una atenuación de 0,76 dB en la situación sin viento con WS respecto a no tener WS.

La energía luego de aplicar el filtro *notch* es despreciable para los casos sin viento, pero de relevancia particularmente en la situación con viento y sin WS.

La SNR, que mide el peso de la energía entera de la señal respecto a la energía luego de aplicar el filtro *notch* es muy alta en la situación sin viento; pero se reduce cuando se está bajo los efectos del mismo. Adicionalmente, el peso relativo de la energía fuera de frecuencia $f = 1$ kHz es de mayor relevancia para el caso de no usar el WS, constatándose la menor SNR. De forma similar, la distorsión armónica THD de la señal, que se asocia con la influencia del viento, es mayor para los casos con viento que en su ausencia, y es menor cuando se cuenta con el WS.



(a) Señal grabada por el micrófono.

(b) Señal luego de aplicar filtro notch.

Figura 6.7: Serie temporal de datos de audio, mostrando señal adquirida y filtro notch eliminando el tono de referencia. En rojo la grabación con WS, en azul sin WS. En ambas grabaciones estaba presente el estímulo tipo viento.

De esto, y de la apreciación cualitativa de los archivos de audio, se concluye que la colocación del WS mejora la respuesta del micrófono ante efectos de corrida de aire. También es posible graficar el archivo de audio de forma de observar el

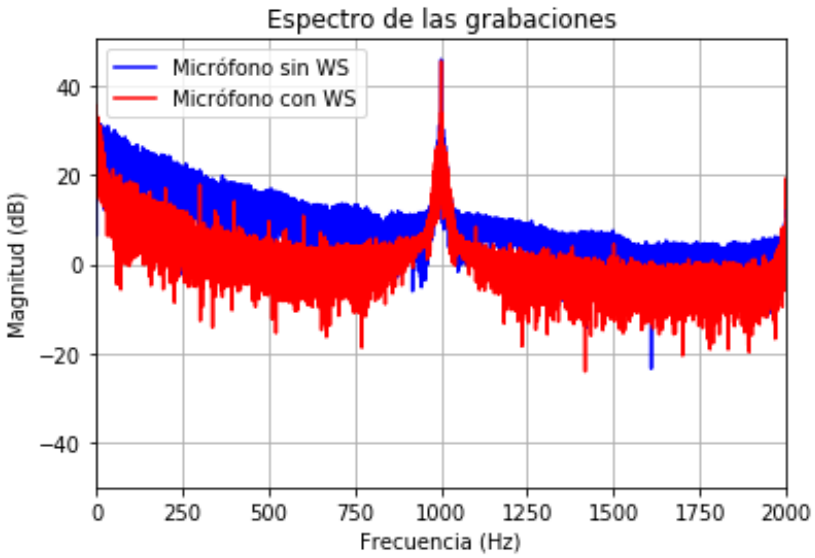


Figura 6.8: Espectro de las señales adquiridas frente al estímulo viento. En azul se grafica el espectro cuando el micrófono no cuneta con WS, mientras que en rojo se utiliza WS.

efecto funcional del estímulo, como se observa en la Figura 6.7. Ya que la agitación realizada es de una frecuencia muy baja respecto al tono emitido por el parlante, se observan picos cada varios ciclos de la señal sonora. Se puede representar este efecto también a través de la FFT como se ilustra en la Figura 6.8, donde se observa que la energía del espectro por fuera de la frecuencia de interés ($f = 1$ kHz) es mayor si no se coloca WS.

Se concluye que los resultados obtenidos, si bien no siguen normas de ensayos acústicos (por ejemplo, la norma ISO-3740 [48]), son suficientes para confirmar las ventajas del uso de WS.

6.1.3. Prueba de funcionamiento prolongado

Un aspecto importante a verificar del sistema en general es la capacidad de funcionar por períodos de tiempo prolongados. Para verificar esto se lo puso a operar bajo condiciones que potencialmente podrían generar problemas en el funcionamiento del mismo. Esto se conoce en la literatura como un *endurance test* o ensayo de resistencia:

> Las pruebas de resistencia son un subconjunto de las pruebas de carga. Una prueba de resistencia es un tipo de prueba de rendimiento centrada en determinar o validar las características de rendimiento del producto sometido a prueba cuando se somete a modelos de carga de trabajo y volúmenes de carga anticipados durante las operaciones de producción durante un período prolongado de tiempo. [49]

Se dejó el dispositivo funcionando de manera continua durante 4 días y 12 minutos grabando audios de 1 minuto. Para detectar más fácilmente posibles erro-

res en la transmisión de datos se estableció que el nodo solo pudiera almacenar un archivo de cada tipo, sobrescribiendo y por tanto perdiendo información si no era consultado entre escrituras. En este modo de operación cualquier falla de comunicación es crítica al implicar pérdida de información.

Se estableció un servidor que consulta cada 30 segundos a la Raspberry por archivos nuevos. Esto es coherente con la operación del nodo, ya que para evitar la perdida de archivos es necesario un intervalo de consulta menor al largo del audio a grabar. Con esta configuración se produce una cantidad de consultas considerable al nodo que también es preciso verificar si son soportadas.

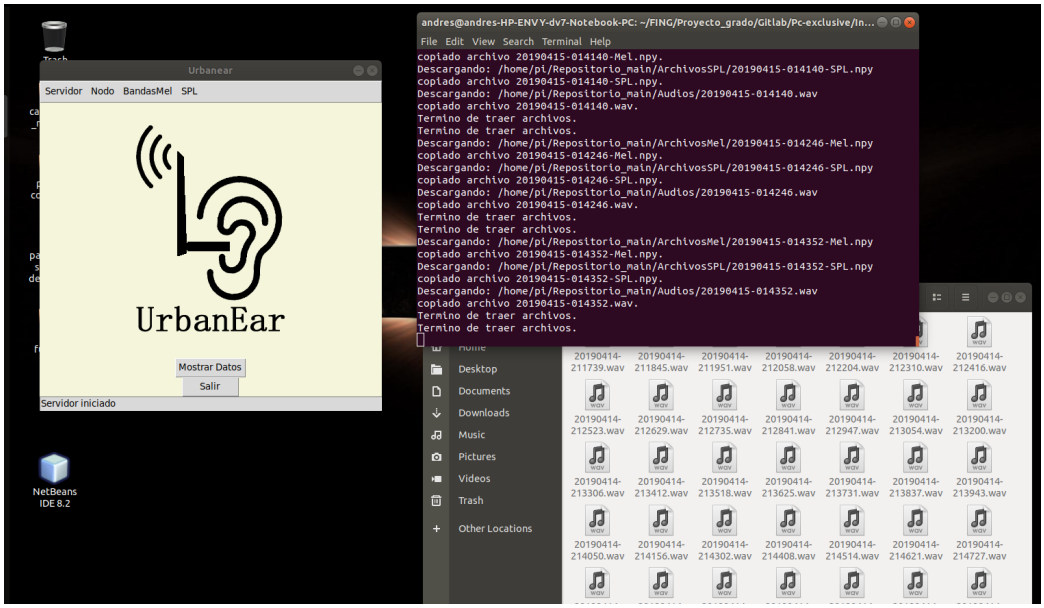


Figura 6.9: Imagen del servidor en funcionamiento.

En la Figura 6.9 se puede apreciar el servidor en funcionamiento a través de la interfaz desarrollada. En terminal se imprimen los resultados de cada una de las consultas al nodo, y se muestra la carpeta donde se almacenan los archivos de audio obtenidos, uno de los tres tipos de archivos transmitidos.

Luego de transcurrido el tiempo del ensayo se procedió a verificar el estado del sistema. En primera instancia se corroboró la ausencia de notificaciones por parte de la interfaz frente a problemas al obtener los archivos del nodo. A su vez en una conexión por *ssh* desde una terminal con el nodo se verificó la inexistencia de errores por parte del programa. Finalmente se corroboró el almacenamiento de los archivos, teniendo un total de 5212 archivos.

Este número resulta coherente con los timestamps de los archivos. La diferencia de tiempos entre los timestamps de dos archivos consecutivos resultó de 66 segundos (aproximadamente cinco de cada 9 veces) y de 67 segundos (aproximadamente cuatro de cada 9 veces). De aquí que el tiempo medio entre audios resulta de 66,44 s.

La cantidad de archivos esperada se puede calcular a partir de la ecuación:

$$\text{cantidad de archivos} = \frac{4 \times 24 \times 60 \times 60 + 12 \times 60}{\text{tiempo entre audios}(s)} \tag{6.1}$$

de donde se deduce que se grabaron 5212 archivos, en coherencia con lo corroborado.

Se concluye que el ensayo fue exitoso al encontrarse la cantidad de archivos obtenidos esperada y no presentar alertas por errores, ya sea de conexión con el nodo o referentes a la ejecución del programa. Cabe notar que se trató de un ensayo controlado, con poca incidencia de factores externos como temperatura, movimientos y cambios de tensión de alimentación.

6.2. Ensayos de software

Los ensayos realizados sobre el código tienen los siguientes objetivos:

- Obtener una medida del *overhead* que introducen los módulos al tiempo de grabación.
- Comprobar de forma empírica algunos límites que impone el hardware del dispositivo a los parámetros configurables.
- Determinar el costo en memoria de los archivos a transmitir y almacenar.
- Comparar los resultados de los módulos con otras implementaciones de referencia (LibROSA) o con dispositivos diseñados para el mismo cometido (Sonómetro).

6.2.1. Ensayo de tiempos

Es de interés práctico poder determinar, variando ciertos parámetros clave, la demora que introduce el procesamiento del audio relevado al proceso que realiza el dispositivo, recordando que el algoritmo implementado graba y procesa de forma secuencial.

Cada módulo se analizó por separado, para obtener de forma independiente los tiempos agregados por cada proceso. Cabe destacar que este análisis se puede realizar de varias maneras. En este caso se optó por utilizar primero implementaciones en Python de *profiling* [50], una forma de análisis de código que ayuda en la optimización del código. Con este análisis se realizaron correcciones de desempeño en la implementación inicial. Una vez satisfechos con el rendimiento en general de los módulos la segunda etapa de análisis consistió en tomar el tiempo de ejecución de dichos módulos variando algunos parámetros cruciales; por ejemplo la duración en segundos del audio.

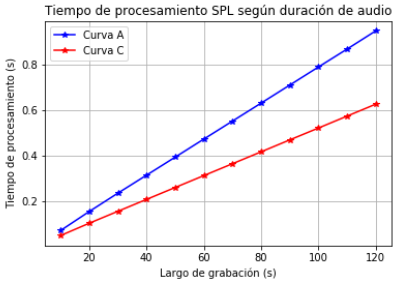
Este análisis se puede realizar con varias funciones de Python, como por ejemplo la función *timeit* [51], que realiza varias iteraciones sobre un segmento de código y devuelve el tiempo en el que se realizaron. Como para este ensayo el interés es obtener el costo en tiempo del procesamiento en la práctica, se decidió por utilizar la función *time* de una forma similar a las funciones *tic* y *toc* de MATLAB.

Este enfoque simple presenta ventajas respecto a alternativas usuales. Por un lado la función *timeit* desactiva funciones de Python, particularmente el *garbage collector* que se encarga de liberar memoria de variables no referenciadas. Las funciones de *benchmarking* [52] solo toman en cuenta el tiempo en que efectivamente se está ejecutando el código, no contabilizando los tiempos generados por interrupciones de otros procesos. La alternativa elegida tiene la ventaja de mantener la ejecución del código lo más similar al funcionamiento en producción y de medir el tiempo de ejecución de reloj.

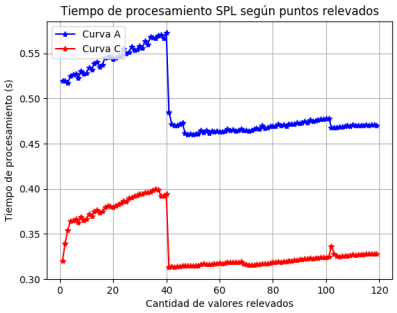
Este análisis se iteró una cantidad de veces considerable (en el orden de 100) para obtener valores estadísticos, generando vectores de valores aleatorios para cada muestra.

6.2.1.1. Módulo de calculo de SPL

Para el SPL se tomaron en cuenta dos parámetros variables, la duración de los audios y la cantidad de puntos a relevar en dicho audio. En ambos casos cada iteración se repitió 100 veces y se tomo como resultado la media del experimento. Los resultados de los ensayos fueron los siguientes:



(a) Ensayo variando duración tomando un valor de SPL por segundo.



(b) Ensayo variando cantidad de puntos relevados en un audio de 60 segundos.

Figura 6.10

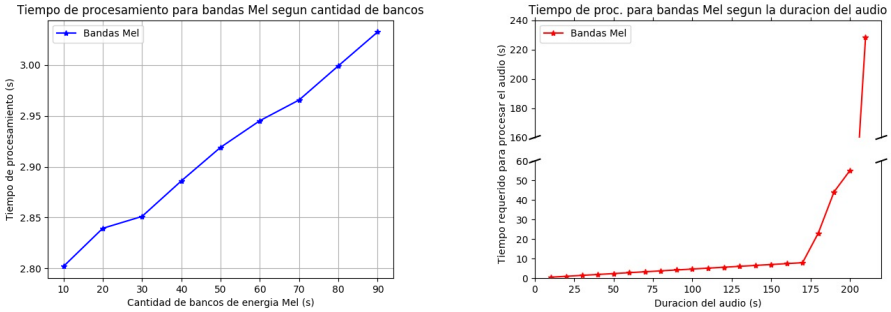
De la figura Figura 6.10a se puede ver claramente que el 'overhead' es proporcional a la duración del audio, y el procesamiento del SPL no introduce demoras significativas en relación a la duración del audio a procesar.

La figura Figura 6.10b muestra que los comportamientos para ambas curvas son similares variando el número de puntos a calcular, con un salto alrededor de 40 puntos (para el caso de un audio de 60 s.). Este comportamiento se explica dado que en ese punto los vectores que se toman para el cálculo de SPL superan las 2^16 muestras, y esto resulta un punto de quiebre en el rendimiento de las funciones de SciPy involucradas. No se investigó en más detalle ya que de todas formas en relación a la duración del audio total, la variación en el tiempo de procesamiento es casi despreciable. Por último, como ensayo más conceptual, se aumentó aún más la duración del audio a procesar hasta lograr saturar los recursos del dispositivo. Si bien no es representativo del rendimiento del equipo completo, este ensayo se realizó como búsqueda del 'cuello de botella' del aparato. Sin embargo, se logró aumentar la duración hasta 20 minutos sin experimentar desvíos importantes de la tendencia marcada en Figura 6.10a.

6.2.1.2. Módulo de calculo de bandas Mel.

En este módulo los parámetros principales estudiados son la duración del audio obtenido y la cantidad de bancos Mel a obtener. Otros parámetros que podrían generar variaciones son los largos de las ventanas y el solapamiento entre ellas para el cálculo intermedio de las STFT, pero el efecto de éstos en el procesamiento no se

analizaron por falta de tiempo. Como en el caso del SPL, se tomaron 100 muestras de cada iteración y se calculó la media del ensayo.



(a) Ensayo variando cantidad de bancos Mel con duración de grabación fija de 60 segundos.

(b) Ensayo variando duración de grabación con cantidad de bancos Mel fija e igual a 40 bancos.

Figura 6.11: Ensayos de tiempos realizados para el cálculo de bandas Mel.

De la figura Figura 6.11b se observa que el costo en tiempo crece linealmente con el largo del vector de entrada, aproximadamente 0.47 segundos cada 10 segundos de audio grabado. Esta tendencia se cumple hasta los 170 segundos, a partir de donde la memoria del dispositivo comienza a limitar la capacidad de procesamiento y se aprecian aumentos excesivos en la demora. Desde los 180 segundos a los 210 la demora apreciada se debe a la necesidad de la Raspberry Pi de utilizar memoria *swap*[4] , la cual es mucho mas lenta que la memoria RAM. Por encima de los 210 segundos el proceso no se puede culminar y se obtiene error de memoria. Cabe destacar que este ensayo se hizo tomando 40 bancos de energía Mel, el cual es un valor usual para este tipo de aplicaciones, si se varía este parámetro variaría también la duración límite de procesamiento.

Por otro lado, se realizo un ensayo variando la cantidad de bancos Mel tomados de un audio de 60 segundos, el tiempo por defecto configurado en el dispositivo. Como se aprecia en la Figura 6.11a, la variación es lineal dentro de un rango bastante acotado de tiempo, por lo que no introduce una variación notable en el costo en tiempo del proceso. Se aclara nuevamente que si se variaran otros parámetros como la duración o los parámetros mencionados al comienzo de esta sección los resultados pueden variar.

6.2.2. Tamaño de los archivos almacenados.

Un dato que puede resultar relevante al funcionamiento del equipo es cuánto ocupan en disco los archivos almacenados que luego se transmitirán al servidor. Se calcula este tamaño en bytes en función de algunos parámetros configurables.

[4]La memoria *swap* consiste en reservar espacio en disco duro para ser usada como memoria dinámica. Al no estar específicamente hecha para este uso es bastante mas lenta que la memoria dinámica usual.

Audio: El audio se almacena en formato WAV. Se puede calcular el tamaño de un archivo de audio con la fórmula:

$$wav\_file = dtype \times duracion \times fs \tag{6.2}$$

donde *dtype* es el tamaño del tipo de dato utilizado para representar valores, en este caso *float32* es el tipo de dato y ocupa 4 bytes. *fs* es la frecuencia de muestreo y *duración* la duración de grabación en segundos.

SPL: Los valores de SPL se almacenan en formato *.npy*. El tamaño cumple con la fórmula:

$$SPL\_file = \big(2 \times dtype \times cant\_SPL\big) + overhead\_NPY \tag{6.3}$$

en donde *cant_SPL* es la cantidad de puntos a relevar en el audio y se multiplica por 2 por tener los valores de las curvas de ponderación 'A' y 'C'.

Además, se suma al costo de los datos mismos de SPL un costo adicional de tamaño debido a que para guardar el archivo y poder ser retomado en otro momento se agrega un encabezado con formato establecido [53], el cual en este caso suma 128 bytes al cálculo (esto es aplicable tanto para SPL como para bandas Mel).

Bandas Mel: Los datos de energía en bandas Mel también se almacenan como matriz en formato *.npy*. La fórmula es la siguiente:

$$Mel\_file = dtype \times nfilt \times \Big(ceil\big(\frac{duracion \times fs}{ceil\big(f\_stride \times fs\big)}\big) + 1\Big) + overhead\_NPY \tag{6.4}$$

nfilt es la cantidad de bancos de filtros Mel y la variable *f_stride* corresponde al paso entre una ventana y otra para el cálculo de STFTs en segundos.
Para tener una idea de los tamaños, con parámetros relativamente estándar como duración de 60 segundos, frecuencia de muestreo 44100 Hz, un valor de SPL por segundo y 40 bancos de filtros Mel con un paso de 0,0125 segundos (la mitad de una ventana típica de 0,025) se tendrian audios de 10 MB, archivos Mel de 767 KB y archivos de SPL de 0,6 KB.

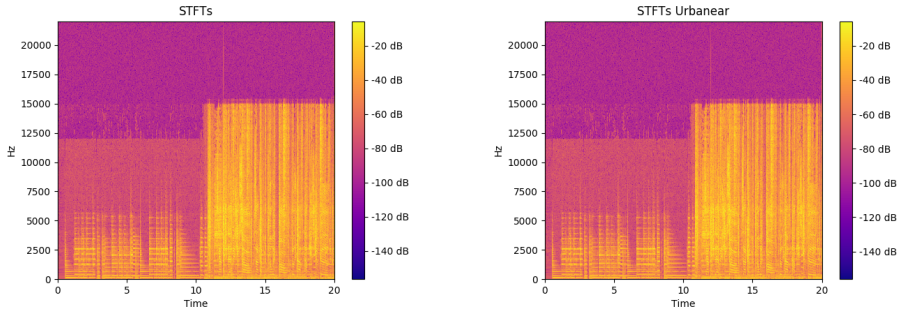
6.2.3. Comparativa con referencias externas.

Para corroborar que los resultados obtenidos de los módulos de calculo de SPL y calculo de bandas Mel sean correctos se contrastaron dichos resultados contra los obtenidos a través de implementaciones externas. En particular, el caso de SPL se ensayó con un sonómetro comercial , mientras que las bandas Mel se compararon contra las obtenidas de las funciones implementadas por LibROSA [37], que es la biblioteca generalmente utilizada para este tipo de aplicaciones y fue tomada como referencia para este proyecto.

6.2.3.1. Comparativa con LibROSA.

A partir de un mismo vector de datos, se desea obtener una comparación entre los resultados del código implementado para este proyecto y el disponible en la

biblioteca LibROSA. En primera instancia, se compara el cálculo de STFTs, el primero realizado con la función *stft()* de SciPy y el segundo con una función específica para STFTs implementada en LibROSA.



(a) STFTs LibROSA. (b) STFTs UrbanEar.

Figura 6.12

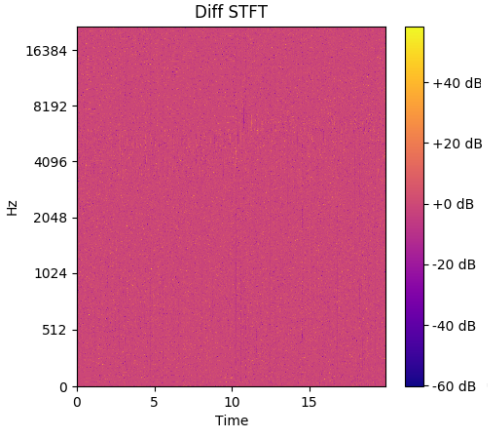


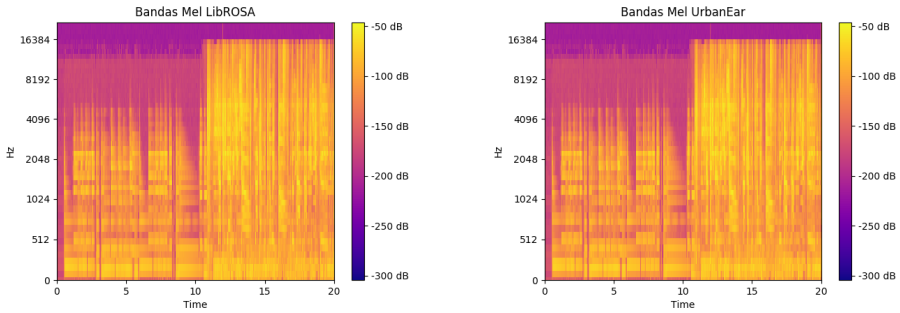
Figura 6.13: La diferencia entre las STFTs calculadas por SciPy y por LibROSA.

Los resultados son casi idénticos, con diferencias en el cálculo, por ejemplo por errores de cálculo numérico, diferencias en la forma de rellenar los bordes, etc.

De la Figura 6.15, se observa que existen diferencias entre ambos algoritmos, particularmente a bajas frecuencias. Esto se debe a la diferencia en el cálculo de los bancos de filtros triangulares, discutida a continuación. Esta diferencia se ve más o menos atenuada según el nivel de resolución en frecuencia determinado en el cálculo de las STFTs.

En los filtros de uno y otro algoritmo existen pequeñas diferencias numéricas debido a cómo se generan los bancos. Estas son perceptibles, principalmente a baja frecuencia en la Figura 6.16.

En el caso de LibROSA (Figura 6.16a), el algoritmo define los bancos triangulares a partir de intersectar rectas con la pendiente y término independiente



(a) Espectrograma de bandas Mel - LibROSA.

(b) Espectrograma de bandas Mel - UrbanEar.

Figura 6.14

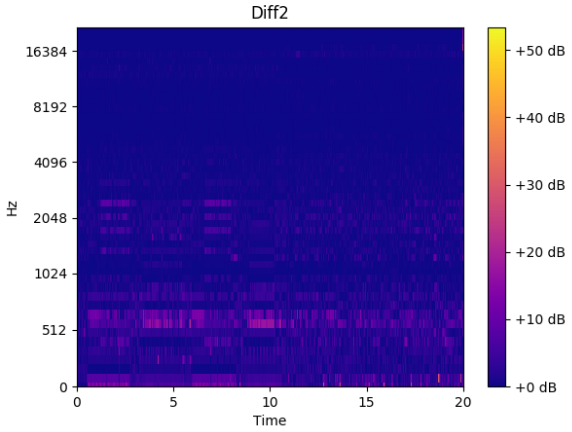


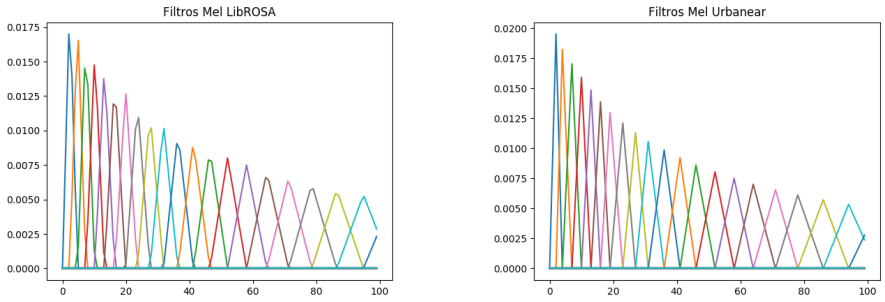
Figura 6.15: La diferencia entre las bandas Mel calculadas por los 2 algoritmos.

deseadas[5]. Este enfoque permite que el vértice del tríangulo no quede representado, ya que al tener un dominio discreto no necesariamente existe una muestra correspondiente al punto de intersección. En la representación esto resulta en que la punta (y los ceros) del triángulo no cierren, dibujando trapecios.

El código desarrollado (Figura 6.16b) en cambio, redondea las frecuencias calculadas en la escala Mel a la frecuencia más cercana representada con la resolución disponible con lo que se obtienen triángulos completos pero en frecuencias ligeramente distintas a las definidas originalmente y potencialmente con pendientes diferentes.

Las diferencias entre uno y otro caso no tienen un efecto notorio en el resultado final, como se observa en la figura Figura 6.14.

[5]Las rectas se definen a partir de las frecuencias centrales definidas para los bancos, equiespaciadas en escala Mel



(a) Banco de filtros Mel - LibROSA. (b) Banco de filtros Mel - UrbanEar.

Figura 6.16

6.2.3.2. Comparación de SPL con sonómetro

Como se explica en el Capítulo 3, la respuesta en frecuencia del micrófono fue calibrada contra un micrófono de referencia y el valor de referencia de SPL contra un sonómetro a 1 kHz. En esta sección se compara el SPL retornado por el dispositivo contra el retornado por el sonómetro profesional B&K para un barrido de tonos de distintas frecuencias y amplitudes. Sin pérdida de generalidad, todos los ensayos fueron realizados para la curva SPL-A.

Para esto se registró la medida obtenida por el sonómetro y la calculada por el dispositivo al exponerlos simultáneamente a 12 frecuencias (espaciadas logarítmicamente entre 100 Hz y 18 kHz) para tres posibles amplitudes de sonido generadas por el parlante. El montaje se ilustra en la Figura 6.17.

Se constató que la fidelidad de la medida calculada por el dispositivo es muy dependiente de la cantidad de muestras utilizadas por el filtro de calibración del micrófono. Particularmente se obtuvo un desempeño pobre para 512 muestras del filtro, y un desempeño tolerable para 4096 que es el que se indica en este capítulo.

Como se observa en la Figura 6.18, los datos del sonómetro y UrbanEar presentan correlación. El coeficiente de correlación es $r = 0{,}88$ para todos los datos y de $r = 0{,}95$ para frecuencias menores a 4000 Hz. Estudios señalan que la información relevante para sonido urbano se encuentra por debajo de esta frecuencia, por lo que es particularmente deseable una buena correlación a estas frecuencias. Para analizar la dependencia de los datos relevados según la frecuencia se realiza la representación de la Figura 6.19. Coherentemente con la Figura 6.18, se observa una cierta correlación principalmente hasta $f = 4000$ Hz, perdiéndose correspondencia a frecuencias superiores. Esto se debe a que a pesar de estar compensado, el micrófono utilizado tiene una sensibilidad pobre a altas frecuencias. El desempeño a $f = 18000$ Hz es particularmente pobre. Para esta frecuencia la longitud de onda de la señal de sonido es de 2 cm, comparable con las dimensiones del soporte y el micrófono, por lo que la misma disposición afecta el desempeño.

También se observa una dependencia en frecuencia no coincidente con la de la curva SPL-A. Esto se debe básicamente a que se está trabajando en una sala

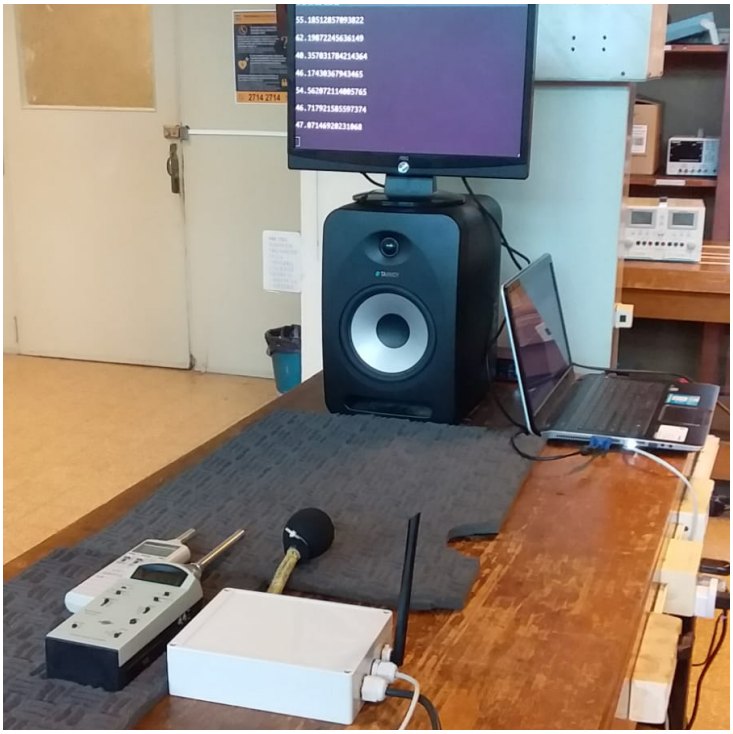


Figura 6.17: Montaje experimental para comparación de SPL. Se despliega en pantalla una rutina de UrbanEar que calcula el SPL en tiempo real, y se registran los valores de sonómetros de referencia.

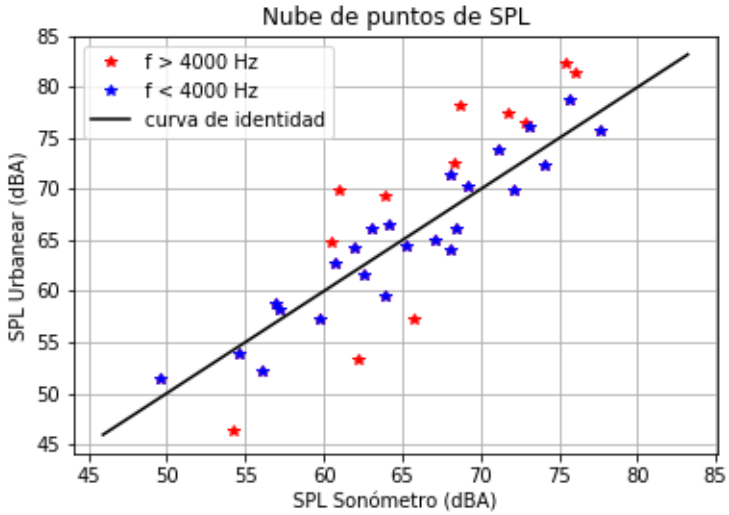


Figura 6.18: Nube de puntos del ensayo realizado. En azul frecuencias menores a 4000 Hz.

no anecoica con una respuesta propia que también se suma a lo relevado por los micrófonos. De todas formas esto no es un impedimento para realizar una comparación entre los valores relevados por los sonómetros.

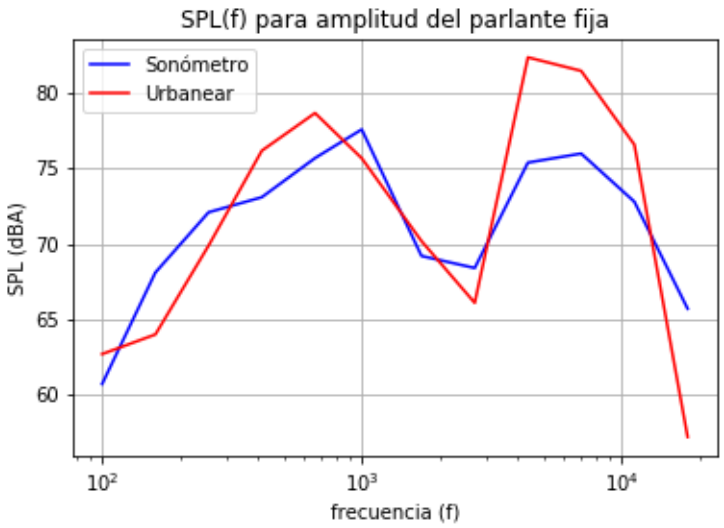


Figura 6.19: SPL para sonómetro y UrbanEar variando la frecuencia. Se observa que las curvas presentan menor correspondencia a partir de $f = 4000$ Hz

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones y trabajo futuro

Este capítulo recoge las conclusiones del proyecto presentado. Se analiza que tanto fue posible cubrir el alcance propuesto originalmente así como los resultados académicos y de aprendizaje. Ademas se detallan ciertos puntos que resultaron desafíos durante el transcurso del proyecto. Como cierre, se listan posibles líneas de trabajo a futuro para que continúe el actor correspondiente.

7.1. Conclusiones generales del proyecto.

La conclusión más importante es que se logró implementar un dispositivo económico, robusto, con capacidad de adquirir, procesar y almacenar señales de audio. Se implementaron los algoritmos para dicho procesamiento, se configuró el dispositivo de forma de asegurar el continuo funcionamiento del sistema sin necesitar de la interacción directa con el equipo. Se implementó una interfaz funcional que permite el uso y configuración del dispositivo. Se calibró el dispositivo frente a equipos de referencia satisfactoriamente.

Se ensayó el aparato en distintas situaciones, verificando el cumplimiento de los requerimientos planteados al comienzo del proyecto, tanto físicos como de software; es decir, la estanquidad y el correcto procesamiento de los descriptores. Además, el dispositivo cumple las limitantes económicas planteadas en un principio, totalizando por debajo de 200 USD.

El proyecto como tal implicó una amplia gama de conocimientos a poner en práctica y nuevos conocimientos a adquirir. Requirió de todos los integrantes del grupo un repaso de teoría de señales y aprender de cero casi todo lo relacionado con procesamiento de señales de audio, como son la definición, cálculo y aplicación tanto del SPL como de las bandas de energía Mel. Ademas, implicó un fuerte contenido de programación en Python, manejo de sistema operativo Linux y de la plataforma Raspberry Pi. La familiarización con equipo de audio, con sonómetros para el relevamiento de SPL, tarjetas de audio y micrófonos de referencia fue un aprendizaje adicional y necesario. La calibración de un micrófono con las diferentes técnicas presentes para lograr ese cometido fue todo un desafío.

Se tomó contacto con la parte manual del problema presentado. El hecho de

armar el gabinete donde se instalarían los componentes y lograr proveer una estanquidad conforme a los requerimientos del proyecto probó ser un desafío.

Por último pero no por eso un detalle menor, el hecho de trabajar durante poco mas de un año en un proyecto tan complejo implicó una necesidad de aprender a gestionar el mismo, y el grupo de trabajo que lo conformó. El manejo de tiempos y de riesgos fue muy importante en algunos momentos del trabajo, en especial cuando las cosas no salían. Por lo tanto cada uno de los miembros se lleva una experiencia didáctica completa y útil a futuro.

7.2. Resultados

Las conclusiones respecto al proyecto en términos de metas propuestas y cumplidas son las siguientes:

Robustez del dispositivo

Se ensayó el dispositivo durante 4 días de funcionamiento continuo en comunicación con un servidor externo. El resultado de este ensayo fue un éxito, sin la necesidad de intervencion de los scripts que están configurados como respaldo. Asegurando en su conjunto una gran autonomia respecto al funcionamiento.

Estanquidad

El dispositivo se ensayó por partes frente a la presencia de agua y el resultado fue satisfactorio, manteniéndose la estanquidad del gabinete. Cabe notar que el punto donde falla la estanquidad es en el windshield, pero esto se debe a la necesidad de mantener un balance entre la protección que provee el windshield frente al viento y la humedad y la permeabilidad que debe tener el mismo para no interferir sustancialmente en la adquisición de audio, en particular para la medición del nivel de presión sonora. Se entiende por lo tanto que se debe mejorar la protección frente a lluvia para el micrófono con algún método externo, como es por ejemplo la colocación de un alero sobre el micrófono.

Datos obtenidos

El equipo adquiere y procesa datos de forma secuencial, por lo que se debe tener en cuenta el tiempo de procesamiento como tiempo muerto para la grabación de audio. El hardware que constituye a la Raspberry Pi tiene limitantes claras respecto a un dispositivo de mayor costo o construido para esta aplicación específica. Estas limitaciones se notan principalmente en la memoria dinámica, como se percibe en los ensayos de las bandas Mel, donde se encuentra como limite de duración para la grabación continua aprox. 3 minutos en condiciones normales.

De todas formas se concluye que bajo parámetros de tiempo de grabación y cantidad de descriptores a calcular normales, el dispositivo mantiene un desempeño aceptable y sólido.

Costo total

Uno de los objetivos planteados es limitar el costo económico del armado del equipo. Para esto se tomó como meta no superar los 200 USD, lo que se cumplió satisfactoriamente. Como se detalla en el Apéndice D, el precio en componentes es de $\approx$ 160 USD ($\approx$ 180 USD sumando un estimado por mano de obra), pero se concluye que un ensamblado más masivo costaría en el entorno de $\approx$ 100 USD la unidad.

7.3. Desafíos y aprendizaje

En esta sección discutimos con un poco más de detalle los desafíos encontrados en el transcurso de este proyecto y cómo se resolvieron, adquiriendo diferentes conocimientos y herramientas como resultado.

Los desafíos en su mayoría resultaron asociados al área de audio y del manejo de los equipos de esa índole. Al no ser el área de experiencia de ninguno de los integrantes del grupo, llevó un tiempo repasar los conocimientos básicos de Fourier, muestreo y diseño de filtros y luego aprender el conocimiento específico necesario para implementar el procesamiento de SPL y de energía en bandas Mel.

Además, un desafío que en un principio parecía menor y demostró llevar tiempo y esfuerzo fue la calibración del micrófono y la construcción de un filtro de compensación en base a los datos obtenidos.

Esto nos llevó horas de armado del *setup* en el laboratorio de Medidas y de toma de datos, repetidas en varias ocasiones hasta estar convencidos de que el método utilizado era el correcto y los resultados obtenidos válidos para el uso en la construcción del filtro.

Otro desafío interesante fue familiarizarse con la Raspberry Pi 3 B+ y el sistema operativo Raspbian. Se aprendieron muchas cosas necesarias para montar el micrófono mediante el protocolo I2S, la instalación de la antena Wi-Fi y su configuración como Access Point, la configuración de scripts que supervisaran la ejecución del código de procesamiento y que estos se ejecutaran al encenderse el equipo.

En resumen, el proyecto resultó valioso desde el punto de vista de aprendizaje, al resultar una experiencia integradora de los contenidos de la carrera de Ingeniería Eléctrica y al presentar la necesidad de enfrentarse a tareas para las que se contaba con menor preparación.

7.4. Trabajos a futuro

En esta sección del capítulo se revisa el alcance planteado de este proyecto, las metas cumplidas y las propuestas para trabajos futuros que se desprenden de este proyecto.

El objetivo a grandes rasgos del proyecto consistía en disponer de un dispositivo funcional con la capacidad de adquirir, procesar y trasmitir datos de audio a

un servidor externo. Esto se logró, implementando inclusive una interfaz "prototipo" del servidor externo para propósitos demostrativos.

El proyecto nace de la idea de disponer de estos dispositivos distribuidos a gran escala sobre un área urbana, por lo que inmediatamente surge como trabajo a futuro el análisis de este prototipo de dispositivo para uso a gran escala. Se debe analizar si el método de comunicación nodo-servidor implementado es suficiente y adecuado para una red de dispositivos de audio y desarrollar un esquema de comunicación adecuado en el caso que no se encuentre conforme a las necesidades de tal despliegue.

También se desprende como trabajo a futuro el análisis del uso de plataformas de menor costo, como las Raspberry Pi Zero, con una implementación parecida a la realizada en este proyecto, teniendo en cuenta la presencia de mayores limitantes de hardware. De todas formas el rápido avance de las tecnologías en el ámbito de computadores de bajo costo sugiere que el costo de armar dispositivos como éste seguirá disminuyendo y aumentara la capacidad del mismo tanto en procesamiento como en memoria.

Respecto al dispositivo construido y al código implementado, queda trabajo por hacer. El código fue optimizado desde los comienzos del proyecto pero aún hay lugar para las mejoras. Un posible punto a considerar es la implementación de las bibliotecas utilizadas en un lenguaje pre-compilado para mejorar el rendimiento y reducir los tiempos entre grabaciones. A su vez, una revisión externa del código puede encontrar espacio para optimizar. El prototipo no hizo hincapié en la alimentación, por lo que se puede analizar la alternativa de alimentar el equipo mediante fuentes que le otorguen autonomía, como paneles solares.

Además, queda como trabajo a futuro el uso de los datos relevados por el dispositivo. El Grupo de Procesamiento de Audio del Instituto de Ingeniería Eléctrica se encuentra actualmente investigando algoritmos de aprendizaje automático para reconocimiento de fuentes de sonido urbano. Un primer paso para este trabajo sería clasificar los datos de bandas Mel y SPL obtenidos por el dispositivo a través de una red neuronal entrenada. Se podría pensar en un sistema que envíe la clasificación directamente si dicha red pudiera correr en la Raspberry.

Por último, el servidor externo que obtiene y muestra los datos obtenidos se puede mejorar con una interfaz más atractiva. A su vez es posible agregarle funcionalidades que den información sobre el nodo, y ver como extender la aplicación cuando exista una red distribuida. También resulta útil desarrollar una base de datos para acumular y organizar los datos obtenidos de una forma eficiente.

Apéndice A

Manual de armado de Urbanear

En este anexo se presenta una guía especificando los pasos a seguir para el armado del dispositivo Urbanear a partir de un gabinete estanco. En primer lugar se ofrece una lista de materiales necesarios, y luego se detalla como proceder con las distintas partes del armado.

El armado a su vez se separa en el acondicionamiento del *gooseneck* para resistir el agua, en la disposición y fijación de los componentes que deben ir adentro del gabinete y en un apartado detallando las conexiones realizadas a pines GPIO de la Raspberry.

A.1. Materiales utilizados

Se hizo uso de los siguientes materiales:

- Raspberry Pi 3 B+
- Estuche para Raspberry Pi 3 B+
- Cargador de Raspberry Pi
- Windshield
- Gooseneck
- Tuerca para el *gooseneck*
- Caja
- Puerto Ethernet hembra
- 1 prensacables pg 11
- 1 prensacable pg 13.5
- Cable de 2 mm
- Micro sd 16 gb
- Antena Wi-Fi
- Micrófono MEMS digital
- Soporte para el micrófono
- Cables
- Espuma de poliuretano
- Lámina de acero
- Pegamento
- Silicona
- Ficha hembra
- Ficha macho/dado
- Cable Ethernet corto
- Vaina termocontraible

- LEDs
- Crimpadora
- Pines de 6 en linea
- Conectores de 6x1 hembra
- 2 conectores de 20x1 hembra
- 2 resistencias de 170 Ω
- Cinta aisladora

A.2. Esquema del equipo

En la Figura A.1 se presenta de manera esquemática la conexión entre los componentes que conforman el prototipo de nodo de UrbanEar.

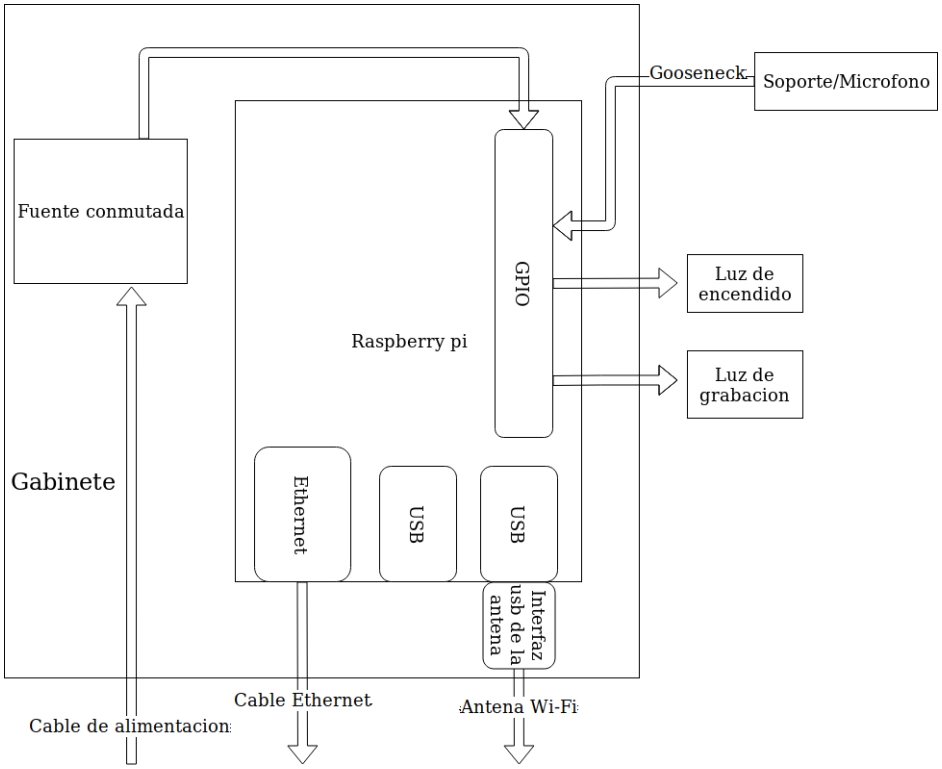


Figura A.1: Esquema de disposición de componentes en el prototipo.

A.3. Gooseneck y soporte del micrófono

En esta sección se detalla el procedimiento llevado a cabo para impermeabilizar el *gooseneck*. A su vez se explica la colocación de los cables en su interior y del soporte del micrófono y la tuerca en sus extremos.

Se realiza un tratamiento con espuma de poliuretano a los efectos de impermeabilizar el *goosneck*. Los pasos a seguir para esto son:

1. Colocar el *gooseneck* en un tubo de PVC como se puede apreciar en la Figura A.2.

2. Aplicar espuma de poliuretano [1] tanto en el interior del *gooseneck* como en el espacio comprendido entre el tubo de PVC y el *gooseneck*. El objetivo de esto es garantizar que cuando la espuma expanda, rellene cualquier posible filtración del *gooseneck*.

3. Dejar que la espuma expanda y seque.

4. Remover la espuma del interior del *gooseneck*. Para esto es posible introducir una varilla de metal larga y de diámetro tal de poder atravesar el *gooseneck*.

5. Limpiar extremos para facilitar colocación de rosca y soporte.

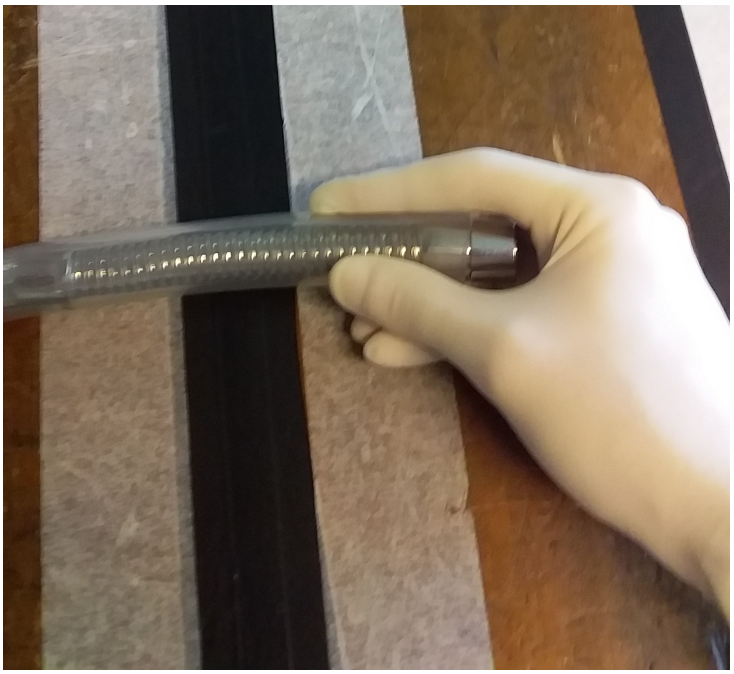


Figura A.2: Gooseneck cubierto por un tubo de pvc.

El soporte a enroscar en el extremo hembra del *gooseneck* se imprimió en una impresora 3D. El archivo .stl del soporte diseñado en este proyecto se encuentra disponible en un repositorio git [54].

Una vez impreso el soporte se procede de la siguiente manera:

1. Soldar los pines en el micrófono.

[1] Se recuerda tener precauciones en la manipulación de la espuma de poliuretano al ser un material corrosivo. Utilice guantes para evitar el contacto con el mismo, recuerde agitar bien el recipiente contenedor de la misma, limpie la cánula y el dispensador con acetona luego del uso.

2. Soldar los cables correspondientes al conector de 6x1. Preferentemente de 6 colores distintos para facilitar el conexionado en la Raspberry.

3. Colocar el conector en los 6 pines del micrófono. Recordar a que pin del micrófono corresponde cada color.

4. Colocar el microfono en el soporte, pasando los cables por su interior.

5. Colocar una vaina termocontraible en los cables y calentar la misma para ajustarla.

6. Pasar los cables por el *gooseneck*.

7. Enroscar el soporte en el *gooseneck*.

Se recomienda en el proceso anterior mantener los cables de un largo considerable para luego ajustarlos en función de la posición de la Raspberry respecto al *gooseneck* en la caja.

Luego de definir el largo de los cables se puede proceder a colocar el *gooseneck* en la caja, afirmando este con una rosca del lado interior. Para que la misma no gire relativa al *gooseneck* una vez fijo el mismo, se atraviesa ambos con un tope.

A.4. Disposición de componentes en el gabinete

A.4.1. Lámina de acero y perforaciones

Para acondicionar la caja el primer paso consiste en cortar una superficie plana (en el caso desarrollado, lámina de acero) que haga de soporte para los demás elementos que se colocaran adentro de la misma.

A su vez, se deben realizar 4 perforaciones fundamentales a la caja, siendo las mismas necesarias para la colocación del *gooseneck*, la antena, el cable de alimentación y para la conexión por Ethernet. Se recomienda definir cual cara será la superior y no realizar perforaciones en dicha cara para disminuir la probabilidad de filtraciones de agua. En Urbanear se realizaron 3 perforaciones en la cara inferior y una en la cara lateral como se puede apreciar en la Figura A.3. Se realizaron a su vez dos perforaciones menores para la ubicación de LEDs para verificar dos estados, prendido/apagado y grabando/procesado.

Para las perforaciones se debe tener en cuenta el diámetro de los prensacables a utilizar, del *gooseneck*, del puerto Ethernet hembra y la disposición definida para los componentes dentro de la caja. La antena con conexión por USB debe poderse conectar a la Raspberry y estar en el exterior de la caja, de forma similar al puerto de Ethernet hembra, ya que la ubicación del mismo define el largo del cable que conecta dicho puerto con el propio de la Raspberry. Por esto se sugiere el siguiente procedimiento:

1. Definir la correspondencia de los agujeros con los elementos en la caja.

2. Ubicar la Raspberry de forma tal que permita la colocación de la antena.

Figura A.3: Disposicion de la lámina y los agujeros en la caja.

3. Dada esta ubicación definir el largo del cable de Ethernet necesario.
4. Armar el cable de Ethernet del largo definido. Para esto disponer de una crimpadora y seguir cualquier guía de armado de cables Ethernet como por ejemplo [55].
5. Verificar el correcto acoplamiento de todos los elementos en dichas posiciones.
6. Realizar las perforaciones en la lamina para la ubicación de la Raspberry.
7. Fijar Raspberry con su respectiva conexión a la antena y al puerto Ethernet hembra. En el caso mostrado, se atornilló el case de Raspberry a la placa.

Para ubicar la salida de la antena se cortó el prensacable correspondiente a la antena Wi-Fi para permitir el pasaje de la misma y mantener la capacidad de direccionarla como se aprecia en la Figura A.5.

A.4.2. Fuente conmutada

La fuente conmutada se pega con pegamento para fijarla a la placa. Se la conecta a una ficha hembra unida al cable de 2 mm y este se pasa por el prensacable en la posición ubicado para finalmente en el extremo que queda por fuera de la caja colocar una ficha macho o un dado dependiendo de la manera en la que se pretenda alimentar el dispositivo. La disposición interna de la fuente así como de la Raspberry, la antena y la salida Ethernet se ilustran en Figura A.4.

Figura A.4: Disposición de los componentes internos a la caja.

A.4.3. LEDs

Otro elemento a colocar en la caja son los LEDs. Sobre dichas perforaciones se sugiere proceder de la siguiente manera:

1. Se verifica la polaridad de las patas del led.
2. Soldar un cable a cada extremo de la resistencia.
3. Cubrir la resistencia con cinta aisladora. Esto es para evitar posibles contactos eléctricos con otro dispositivo.
4. Soldar el cable con la resistencia al positivo del LED. Esto a los efectos de facilitar la ubicación de la polaridad para conectarlo a la Raspberry.
5. Cubrir con cinta aisladora la soldadura hecha para evitar contactos con la otra pata del LED.
6. Soldar cable a la pata restante del LED.
7. Cubrir esta ultima soldadura con cinta aisladora.
8. Presentar el LED en el agujero pequeño con los cables en el interior de la caja.
9. Cortar los cables verificando el largo necesario según la posición de la Raspberry.

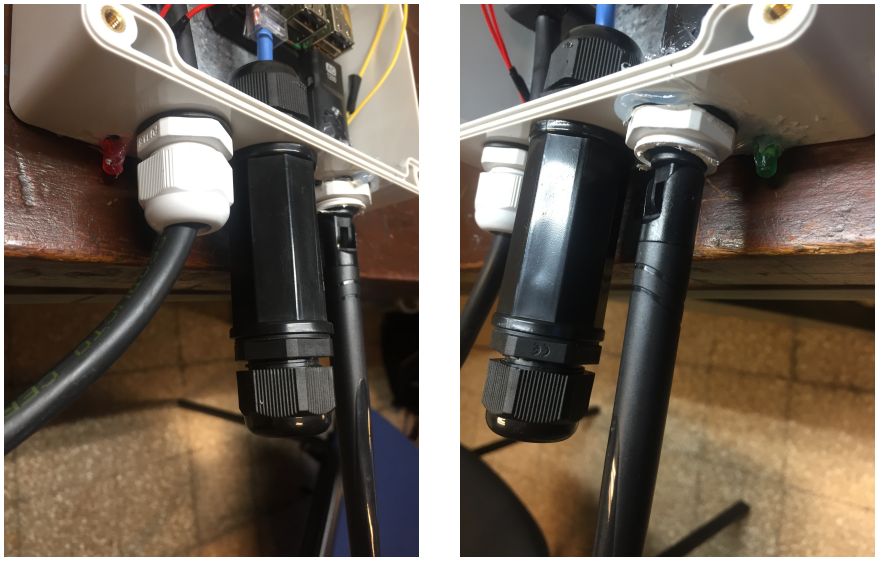


Figura A.5: Terminación de la cara inferior de la caja.

A.4.4. GPIOs

Se debe realizar la conexión eléctrica de todos los elementos con la Raspberry. Para esto en la Figura A.6 se presenta la disposición de pines de la Raspberry y en la Tabla A.1 la Correspondencia de dichos pines con los pines del micrófono, bornes de la fuente y los LEDs. Se utilizó un LED rojo (LED de alimentación) a efectos de representar si la Raspberry se encuentra alimentada, y un LED verde (LED de grabación) para desplegar si se encuentra grabando.

Tabla A.1: Correspondencia de pines de la Raspberry con los elementos del gabinete.

Pin de Raspberry	Conectado a
1	Pin de 3 V del micrófono
4	Borne 5 V de la fuente de alimentación
6	Borne tierra de la fuente de alimentación
9	Pin SEL del micrófono
12	Pin BCKL del micrófono
14	Pin tierra del micrófono
17	Borne positivo del LED de alimentación
25	Borne tierra del LED de alimentación
30	Borne tierra del LED de grabación
32	Borne positivo del LED de grabación
35	Pin LRCL del micrófono
38	Pin DOUT del micrófono

Para realizar esta conexión se soldaron los cables y se ubicaron en los 2 conectores de 20x1 colocados en la Raspberry segun corresponda.

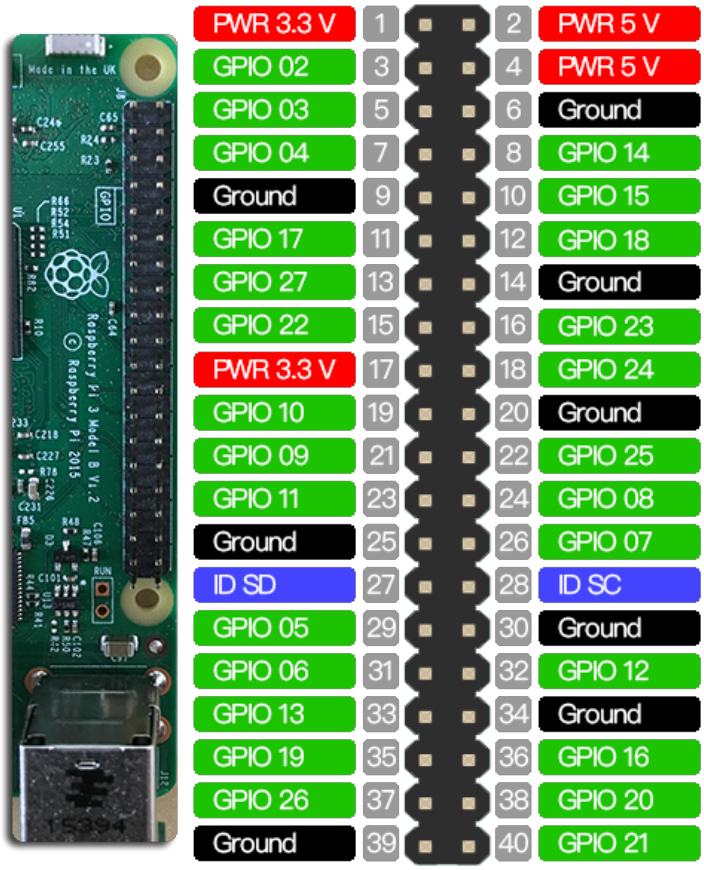


Figura A.6: Esquima de pines de la Raspberry Pi 3 B+.

A.4.5. Sellado de la caja

Una vez colocados todos los componentes y realizadas las conexiones eléctricas se debe sellar la caja. Para esto se utilizó silicona fría para sellar las uniones de los distintos elementos en sus correspondientes perforaciones, y se utilizó una banda de goma provista por el fabricante del gabinete para su sellado, como se observa en la Figura A.7. Para la presentación final, se atornilla la caja para dejarla cerrada y se coloca el WindShield protegiendo el *gooseneck* y el micrófono y obteniendo el resultado de la Figura A.8.

Figura A.7: Colocado de goma en la union de la caja.

Figura A.8: Dispositivo final.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Manual de usuario

Este proyecto presenta consigo una interfaz de usuario la cual explicaremos como ejecutarla y como utilizarla.

B.1. Instalación

Para utilizar la interfaz se necesita descargar la carpeta Interfaz del repositorio git [56]. Dicha interfaz esta desarrollada en Python 3.6.7 por lo tanto es necesario tener el mismo instalado en la pc, así como contar con una serie de paquetes para el funcionamiento de la misma, estos paquetes se listan a continuación con su correspondiente linea de comando para instalarlos en caso de contar con un sistema operativo basado en Debian. Se recomienda para esto instalar pip3, ya que usaremos el mismo para instalar algunos paquetes: *sudo apt-get install python3-pip*.

- numpy: *sudo pip3 install numpy*
- tkinter: *sudo apt-get install python3-tk*
- tkSnack: *sudo apt-get install python3-tksnack*
- yaml: *sudo pip3 install pyyaml*
- matplotlib: *sudo pip3 install matplotlib*
- subprocess.run: *sudo pip3 install subprocess.run*
- paramiko: *sudo pip3 install paramiko*

B.2. Uso

Luego de tener los paquetes mencionados anteriormente instalados se prosigue a ejecutar la interfaz. Para esto se debe ejecutar el archivo interfaz.py con el siguiente comando *sudo ./interfaz.py* , donde se nos presentara una ventana como la de la Figura B.1. En esta ventana se puede apreciar el logo del proyecto y dos

botones, el de "Salir"para terminar la ejecución del programa y el de "Mostrar Datos"para visualizar todos los datos obtenidos del nodo. A su vez esta ventana cuenta con una barra de estados abajo del todo, donde menciona el estado del servidor.

Figura B.1: Imagen de inicio de la interfaz grafica.

Esta ventana cuenta con 4 pestañas que presentan distintas funcionalidades según su nombre. Las pestañas de "SPLz "BandasMel"presentan opciones referentes a las distintas maneras de graficar los datos de bandas mel y SPL. En particular el SPL puede ser representado como gráfica de barras o de puntos según la opción marcada, así como también permite graficar los datos de un archivo solo, o de un histórico de archivos lo cual sera explicado mas adelante. También las bandas mel pueden ser graficadas tomando una escala lineal en su eje vertical o utilizando la escala mel según la opción marcada.

Luego la pestaña "Nodo"presenta las funcionalidades de configuración del nodo, ya sea consultar la configuración actual que esta utilizando el mismo como imponer una configuración nueva. Para esto se generan ventanas nuevas en cada caso, donde al consultar la configuración, en la nueva ventana se halla un cuadro de texto mencionando cada uno de los parámetros utilizados como se puede ver en la Figura B.2. En el caso de de querer imponer una configuración se abre una nueva ventana con una serie de cajas de texto y botones de opciones entre otros, que definirán los parámetros para la nueva configuración del nodo[1], como se puede

[1]Es necesario que los parámetros a utilizar sean coherentes, de otra forma la interfaz

ver en la Figura B.3.

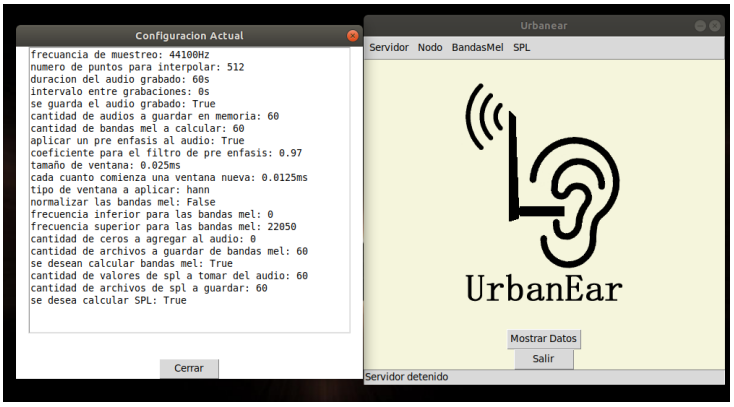


Figura B.2: Imagen de la ventana que muestra la configuración de la interfaz gráfica.

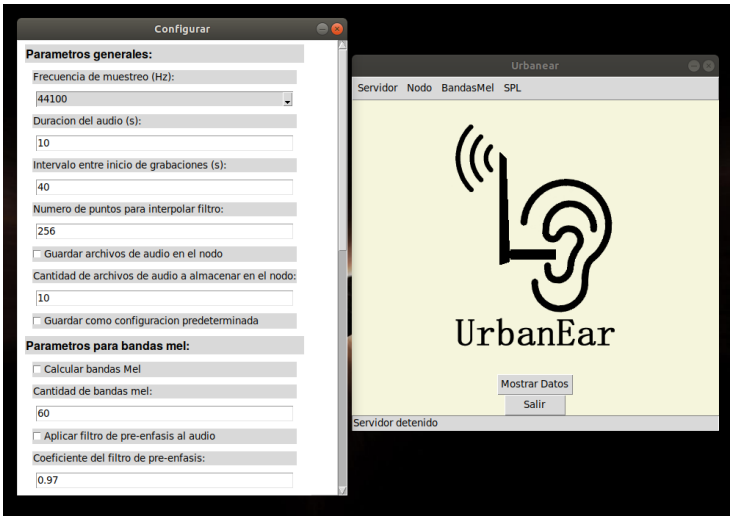


Figura B.3: Imagen de la ventana de configuracion de la interfaz grafica.

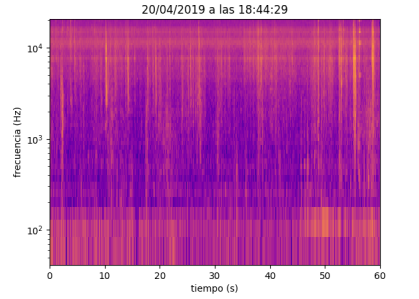
La pestaña "Servidor"presenta dos funcionalidades: la primera de ellas es determinar el tipo de conexión que tendrá el servidor con el nodo, siendo posible conectarse por Wifi o por cable Ethernet. Cabe recordar que para cualquier acción que requiera conexión con el nodo, es necesario que este seleccionada la interfaz correcta, de lo contrario la interfaz mostrara un mensaje diciendo que es imposible conectarse con el nodo. La otra funcionalidad que se presenta en dicha pestaña es la de iniciar o detener el servidos, de estas solo se permite la acción valida, por ejemplo no se permite detener el servidor si esta ya esta inactivo.

Finalmente se encuentran las funcionalidades del botón "Mostrar Datos". Este botón se encarga de mostrar los datos que fueron obtenidos por el servido del nodo, abriendo una ventana para seleccionar el archivo a ser mostrado, este detecta

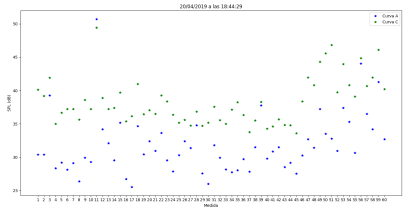
levantara un mensaje de error y no realizara dicha configuración sobre el nodo.

cual es el contenido del archivo según el nombre y lo muestra de la manera que corresponde. Ya sean archivos de bandas mel o SPL, e incluso archivos de audio, generando una ventana de reproductor para poner play, pausa o stop al audio, Esto se aprecia en las imágenes de la Figura B.4. También en caso de estar seleccionada la opción de mostrar el histórico de SPL, en caso de seleccionar un archivo de SPL generara otra ventana nuevamente para que se seleccione el archivo correspondiente al final del intervalo. Esta opción tiene la particularidad de que gráfica todos los puntos de SPL en los archivos contenidos en dicho intervalo, poniendo como marcas en el eje x el nombre del archivo al que corresponden los puntos que siguen como se puede ver en la Figura B.5.

(a) Reproductor de audio.



(b) Grafico de bandas mel.



(c) Grafico de SPL.

Figura B.4: Exposición de datos por medio de la interfaz gráfica.

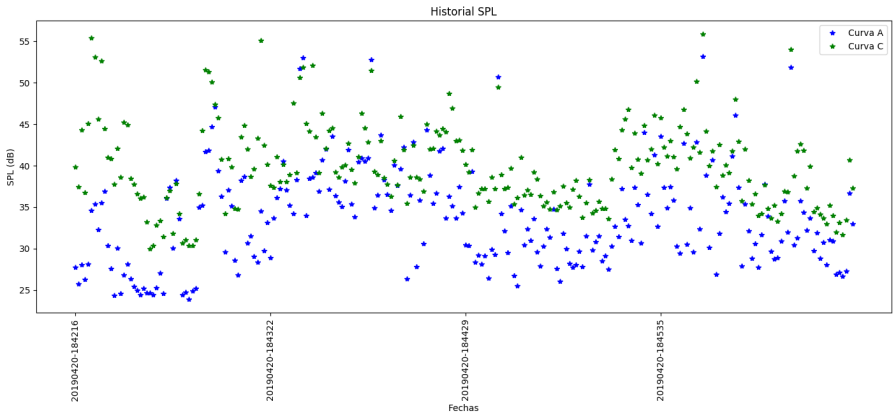


Figura B.5: Imagen de un histórico de SPL mostrado por la interfaz grafica.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Configuración de Wi-Fi

En este anexo se proporciona una breve guía de cómo configurar la Raspberry Pi para usar la antena Wi-Fi externa y para funcionar en el modo Access Point, generando su propia red WLAN.

En primera instancia, se debe instalar la antena Wi-Fi comprada. Para esto, hay que instalar los drivers necesarios para el funcionamiento de la antena. Los drivers dependen del chipset que usa el fabricante para hacer la antena, que no siempre está especificado por el proveedor, por lo que es útil conocer los comandos para determinar que driver es necesario para la antena.

La forma de realizar esto es conectar la antena a la RaspBerry, correr el comando **lsubs** para ver el número de dispositivo asociado a la antena USB y con ese número el comando **lsubs -t** devuelve el driver de ese dispositivo como muestra la Figura C.1.

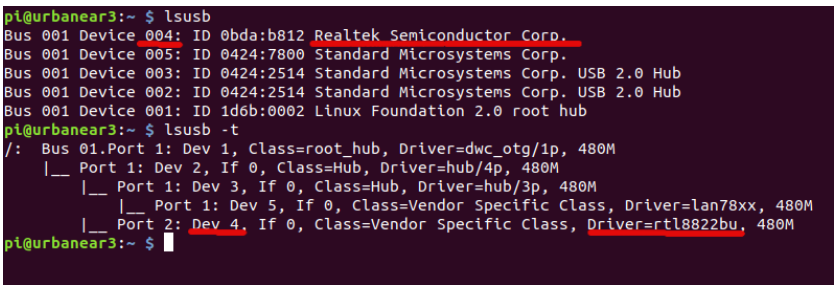


Figura C.1: Datos de driver Wi-Fi.

Para el chipset que se muestra en Figura C.1, se encontraron una serie de drivers compilados de Realtek por un usuario de *https://www.raspberrypi.org/forums/*, que hicieron bastante fácil la tarea de instalar el driver necesario.

Estos drivers se encuentran disponibles en [57]. Para obtener estos drivers e instalarlos, se corren los siguientes comandos:

```
sudo wget http://www.fars-robotics.net/install-wifi -O /usr
    /bin/install-wifi
sudo chmod +x /usr/bin/install-wifi
sudo install-wifi
```

También se puede correr el comando **sudo install-wifi -h** para más detalles de cómo usar el script.

Una vez que esto está instalado, se puede utilizar la antena como sustituto del adaptador Wi-Fi que ya viene incorporado a la RaspBerry. Si se corre el comando **ifconfig** se pueden ver las interfaces **wlan0** y **wlan1** correspondiendo a la interfaz *on-board* y a la recién instalada.

Para la configuración a utilizar, como el adaptador Wi-Fi principal es el USB, se decidió que este lleve la denominación **wlan0**. Es útil poder activar/desactivar el adaptador *on-board* y no afectar el funcionamiento, por lo que agregamos al archivo de configuración **/etc/udev/rules.d/72-persistent.rules** las siguientes líneas.

```
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="<MAC-DE-
    ADAPTADOR-USB>", NAME="wlan0"
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="<MAC-DE-
    ADAPTADOR-ONBOARD>", NAME="wlan1"
```

La configuración de la antena en el modo Access Point, implica modificar algunos archivos de configuración. Los pasos a realizar fueron basados en [42], con algunas ligeras variaciones para el caso presente.

En el archivo **/etc/dhcpcd.conf** se encuentra la configuración relacionada con las interfaces, y la configuración necesaria para la interfaz **wlan0** es colocar al final del documento:

```
interface wlan0
static ip_address=192.168.127.1/24
denyinterfaces wlan0
```

Con esta configuración se establece que la red Wi-Fi del AP serán las IP 192.168.127.1/24. La última línea evita que el proceso *dhcpd* intente establecer la IP de la interfaz **wlan0** mediante DHCP (IP dinámica).

El siguiente paso es configurar el servidor DHCP que se encargara de dar IPs al servidor que se conecte a la RaspBerry. Para esto utilizamos **dnmasq** como servidor DHCP, y este se configura en **/etc/dnsmasq.conf**. En dicho archivo agregamos las líneas:

```
interface=wlan0
  dhcp-range=192.168.0.11,192.168.0.30,255.255.255.0,24h
```

Donde se determina el rango de direcciones para asignar mediante DHCP y la duración del *lease*[1].

[1]Tiempo por el cual es válida la asignación IP del DHCP.

Luego, como ya se mencionó en Capítulo 5, se edita el archivo **/etc/hostapd/hostapd.conf** con la configuración necesaria para la red Wi-Fi establecida por la Raspberry Pi. En [41] se puede encontrar un ejemplo de configuración más completo, con los parámetros explicados.

```
interface=wlan0
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
ssid=NETWORK
wpa_passphrase=PASSWORD
```

Los parámetros principales a tener en cuenta son **ssid** y **wpa_passphrase**, donde **ssid** es el SSID de la red Wi-Fi que se verá al conectarse y **wpa_passphrase** la contraseña requerida para la conexión.

Por último, se reinicia la RaspBerry Pi con el comando **sudo reboot** y ya está lista para funcionar como AP.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

Costo del dispositivo

Durante el transcurso del proyecto se ensamblaron dos iteraciones independientes y funcionales del dispositivo. La desarrollada en primer instancia, *Urbanear 1* sirvió de base para proponer algunas modificaciones deseables de cara al segundo ensamblaje. Particularmente *UrbanEar 1* no contaba con un puerto hembra Ethernet externo, lo que resulta incómodo a efectos de proteger el cable y definir el largo del cable a utilizar. A su vez, para *UrbanEar 1* se utilizó un WindShield comprado en Uruguay, mientras que *Urbanear 2* cuenta con un WindShield de mayor calidad. Si bien no afectan el costo del dispositivo, en *Urbanear 2* se realizó de forma más prolija la soldadura interna de la alimentación, y se colocaron LEDs que indican de forma externa el estado del dispositivo.

En la Tabla D.1, y Tabla D.2 se desglosa el costo de fabricación de ambas versiones. En dicho desglose se indica para cada artículo si se pagó en pesos o dólares, se lista un subtotal en cada moneda y un total único convertido a dólares[1].

A los datos de la tabla corresponde sumar un estimado de mano de obra de acuerdo al tiempo estimado de ensamblaje. Se estima que al equipo de proyecto le demoró 8 horas en total de trabajo manual el armado de la caja. Contando con la inexperiencia del grupo, este trabajo es realizable en el orden de 4 horas o menos. Estimando valores por exceso, se puede dimensionar un tiempo de ensamblaje de 4 horas y un precio de mano de obra de $180[2] la hora, resultando en un costo por mano de obra de $720, o 21 USD[3]. Esto resulta en un precio neto de 162 USD para la versión de *Urbanear 1* (sin Ethernet H-H y con Windshield de menor calidad) y de 183 USD para la versión *Urbanear 2*.

Es preciso notar que el análisis fue conservador para los estimados, utilizando hipótesis de máximo precio. A su vez, los precios se reducirían en primer lugar por mayoreo si se realizan múltiples unidades. También hay soluciones que resultan en un precio alto y se podrían reducir costos, por ejemplo aprovechando el uso de la impresora 3D. A saber, la tuerca utilizada para el *gooseneck* y el estuche en el cual se colocó la RaspBerry se podrían imprimir ahorrando hasta 15 USD en

[1] Utilizando la cotización del dolar a la fecha de 4/2019

[2] Valor correspondiente al valor por hora de un salario correspondiente a media canasta básica nacional a la fecha (4/2019).

[3] Según cotización a la fecha (4/2019)

	Precio en $	Precio en USD
Gabinete estanco		24
Gooseneck+tuerca	287	
WindShield	89	
Micrófono Adafruit		11
Antena		17
Fuente alimentación		10
Estuche RaspBerry		10
Tarjeta micro SD 16GB		6
Raspberry 3B+		35
Cables y varios	600	
Subtotal	976	113
Total Dólares		141

Tabla D.1: Costo de fabricación *UrbanEar 1*

	Precio en $	Precio en USD
Gabinete estanco		24
Gooseneck+tuerca	287	
WindShield		15
Ethernet H-H		9
Micrófono Adafruit		11
Antena		17
Fuente alimentación		10
Estuche RaspBerry		10
Tarjeta micro SD 16GB		6
Raspberry 3B+		35
Cables y varios	600	
Subtotal	887	137
Total Dólares		162

Tabla D.2: Costo de fabricación *UrbanEar 2*

materiales.

A su vez en una etapa futura en que se desee definir la forma de comunicación para una red compuesta por sensores, puede no resultar necesaria la antena de Wi-Fi según la solución implementada. En última instancia es reductible el costo de la alimentación si se resuelve fabricar la fuente conmutada en lugar de utilizar la fuente comercial proporcionada por RaspBerry.

Si se considera una reducción de 25 %[4] por concepto de mayoreo, y se contemplan las consideraciones anteriores (por lo tanto descontando el 80 % del precio de la antena, la tuerca, la alimentación y el estuche), el precio de mínima resulta 97 USD; por lo que se puede concluir que es posible costear los materiales en el orden de 100 USD por unidad.

[4]Descuento por mayoreo para el micrófono MEMS [28]

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] Martha Georgina Orozco Medina and Alice Elizabeth González. La importancia del control de la contaminación por ruido en las ciudades. *Ingeniería Revista Académica de la Facultad de Ingeniería Universidad Autónoma de Yucatán*, 19(2):129–136, 2016.

[2] Alice Elizabeth González, Esteban Gaja Díaz, Andrés Jorysz, and Gonzalo Torres. Desarrollo de un modelo predictivo de ruido urbano adaptado a la realidad de la ciudad de montevideo, uruguay. *Tecnoacústica Madrid. Ref. Pacs*, 43, 2000.

[3] Charlie Mydlarz, Justin Salamon, and Juan Pablo Bello. The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics*, 117:207–218, 2017.

[4] Alejandro Draper, Nicolás Obrusnik, and Pablo Zinemanas. *Documentación del Proyecto de fin de carrera Pestibee*. PhD thesis, Universidad de la República Montevideo, 2015.

[5] Justin Salamon and Juan Pablo Bello. Feature learning with deep scattering for urban sound analysis. In *EUSIPCO*, pages 724–728, 2015.

[6] Manuel Davy. An introduction to statistical signal processing and spectrum estimation. In *Signal Processing Methods for Music Transcription*, pages 21–64. Springer, 2006.

[7] Chadawan Ittichaichareon, Siwat Suksri, and Thaweesak Yingthawornsuk. Speech recognition using mfcc. In *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012) July*, pages 28–29, 2012.

[8] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.

[9] Ley n 18.331 - protección de datos personales y acción de habeas data. `https://www.agesic.gub.uy/innovaportal/v/302/1/agesic/ley-n%C2%B0-18331-de-11-de-agosto-de-2008.html`. Accedido por última vez: 2018-7-12.

[10] Dictamen 10/010 - solicitud realizada por la directora de derechos ciudadanos respecto a la videovigilancia en el derecho nacional.

https://www.gub.uy/unidad-reguladora-control-datos-personales/comunicacion/publicaciones/dictamen-10010. Accedido por última vez: 2018-7-12.

[11] Federico Miyara. *Acústica y sistemas de sonido*. Universidad Nacional de Rosario, 2003.

[12] Wikipedia "a-weighting", wikipedia "equal-loudness contour". https://en.wikipedia.org/wiki/A-weightinghttps://en.wikipedia.org/wiki/Equal-loudness_contour. Accedido por última vez: 2018-7-12.

[13] Nivel continuo equivalente leq. http://www.inercoacustica.com/acustipedia/item/236-%C2%BFqu%C3%A9-es-el-nivel-continuo-equivalente-leq. Accedido por última vez: 2018-7-12.

[14] Design of digital filters for frequency weightings. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4331191/. Accedido por última vez: 2018-7-12.

[15] Leopoldo Agorio and Andrés Corchs, Florencia Montaldo. Sonómetro basado en microcontrolador. https://eva.fing.edu.uy/mod/page/view.php?id=31445. Accedido por última vez: 2018-7-12.

[16] Transposed direct forms. https://ccrma.stanford.edu/~jos/fp/Transposed_Direct_Forms.html. Accedido por última vez: 2019-3-30.

[17] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[18] Douglas O'shaughnessy. *Speech communication: human and machine*. Universities press, 1987. Accedido por última vez: 2019-8-7.

[19] Linear filtering based on the discrete fourier transform. https://www.allaboutcircuits.com/technical-articles/linear-filtering-based-on-the-discrete-fourier-transform/. Accedido por última vez: 2019-3-30.

[20] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[21] Center for computer research in music and acoustics - examples. https://ccrma.stanford.edu/~jos/sasp/Example_2_Time_Domain.html. Accedido por última vez: 2019-3-30.

[22] Randall P Wagner and Steven E Fick. Pressure reciprocity calibration of a mems microphone. *The Journal of the Acoustical Society of America*, 142(3):EL251–EL257, 2017.

[23] Acoustic methods of microphone calibration. http://www.pcb.com/Contentstore/mktgcontent/WhitePapers/WPL_36_Acoustic_methods_calibration_PCB.pdf. Accedido por última vez: 2019-30-1.

[24] Acoustic methods of microphone calibration. http://www.tdl-tech.com/miccalax.pdf. Accedido por última vez: 2019-30-1.

[25] Impulse response methods. http://pcfarina.eng.unipr.it/Public/Papers/238-NordicSound2007.pdf. Accedido por última vez: 2019-30-1.

[26] Angelo Farina. Advancements in impulse response measurements by sine sweeps. In *Audio Engineering Society Convention 122*. Audio Engineering Society, 2007.

[27] Guy-Bart Stan, Jean-Jacques Embrechts, and Dominique Archambeau. Comparison of different impulse response measurement techniques. *Journal of the Audio Engineering Society*, 50(4):249–262, 2002.

[28] Adafruit i2s mems: Raspberry pi wiring and test. https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout/raspberry-pi-wiring-and-test. Accedido por última vez: 2019-3-19.

[29] Luis Gascó Sánchez. *Diseño y caracterización de un sistema microfónico de intemperie para la localización de fuentes sonoras distantes*. PhD thesis, Industriales, 2014.

[30] Gary R Smith and PD Loly. The great beer bottle experiment. *American Journal of Physics*, 47(6):515–518, 1979.

[31] Microphone to camera mount adapter. https://www.thingiverse.com/thing:2370212. Accedido por última vez: 2019-3-19.

[32] blender.org - home of the blender project - free and open 3d creation software. https://www.blender.org/. Accedido por última vez: 2018-7-12.

[33] Impresora 3d del instituto de ingeniería eléctrica. https://iie.fing.edu.uy/personal/nacho/impresora-3d/. Accedido por última vez: 2019-4-23.

[34] Ultimaker cura: Advanced 3d printing software, made accessible. https://ultimaker.com/en/products/ultimaker-cura-software. Accedido por última vez: 2018-7-12.

[35] Berryconda: Conda para raspberry pi. https://github.com/jjhelmus/berryconda. Accedido por última vez: 2019-3-19.

[36] Cómo instalar sounddevice. https://python-sounddevice.readthedocs.io/en/0.3.11/. Accedido por última vez: 2019-3-19.

[37] Librosa, python package for music and audio analysis. https://librosa.github.io/librosa/. Accedido por última vez: 2019-3-19.

[38] Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. Accedido por última vez: 2019-3-19.

[39] Secure file transfer protocol. https://www.ssh.com/ssh/sftp/. Accedido por última vez: 2019-3-19.

[40] Daemon definition. http://www.linfo.org/daemon.html. Accedido por última vez: 2019-3-19.

[41] Hostapd. https://w1.fi/hostapd/. Accedido por última vez: 2019-3-17.

[42] Hostapd. https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/. Accedido por última vez: 2019-3-21.

[43] Raspberry pi zero - conserve power and reduce draw to 80ma. https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-conserve-energy. Accedido por última vez: 2019-3-30.

[44] Power consumption benchmarks. https://www.pidramble.com/wiki/benchmarks/power-consumption. Accedido por última vez: 2019-3-30.

[45] Pliego tarifario ute. https://portal.ute.com.uy/sites/default/files/docs/Pliego%20Tarifario%20Vigente.pdf. Accedido por última vez: 2019-3-30.

[46] International Electrotechnical Commission et al. *Degrees of protection provided by enclosures (IP Code)(IEC 60529: 1989+ A1: 1999+ A2: 2013)*. IEC, 2013.

[47] Monika Singh, Saroj Dogra, and Rajesh Mehra. Iir filter design and analysis using notch and comb filter. *Int J Sci Res Eng Technol (IJSRET)*, 2(6):358–361, 2013.

[48] International Organization for Standardization. *ISO 3740:2019 – Acoustics – Determination of sound power levels of noise sources – Guidelines for the use of basic standards)*. ISO, 2019.

[49] J Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. *Performance testing guidance for web applications: patterns & practices*. Microsoft press, 2007.

[50] Profiling in python. https://docs.python.org/3/library/profile.html#module-cProfile. Accedido por última vez: 2018-7-12.

[51] Measure execution time of small code snippets. https://docs.python.org/2/library/timeit.html. Accedido por última vez: 2018-7-12.

[52] Benchmarking your code - rbspy docs. https://rbspy.github.io/benchmarking-your-code/. Accedido por última vez: 2018-7-12.

[53] Npy format. https://www.numpy.org/devdocs/reference/generated/numpy.lib.format.html. Accedido por última vez: 2018-7-12.

[54] Repositorio con el soporte para el micrófono. `https://github.com/andrescorchs/UrbanEar_Soporte.git`. Accedido por última vez: 2019-5-5.

[55] Armado de cables ethernet. `https://www.howtogeek.com/60486/how-to-make-your-own-custom-length-network-cables/`. Accedido por última vez: 2019-4-28.

[56] Repositorio con interfaz grafica. `https://github.com/andrescorchs/UrbanEar_Interface`. Accedido por última vez: 2019-5-5.

[57] Drivers realtek. `http://downloads.fars-robotics.net/`. Accedido por última vez: 2019-4-23.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el domingo 11 agosto, 2019.
http://iie.fing.edu.uy/