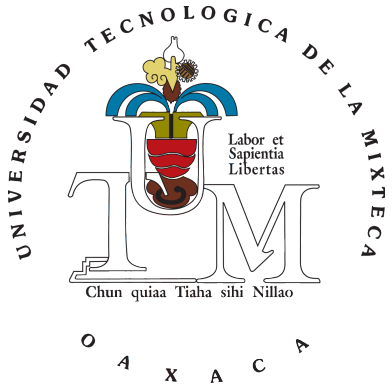


Universidad Tecnológica de la Mixteca

Implementación básica de un dispositivo para recolección y análisis multifractal de señales termoacústicas

Tesis
Para obtener el título de:
Ingeniero en física aplicada

Presenta:
Bladimir Aldaz González

Director de tesis:
Dr. Hugo David Sánchez Chávez

Huajuapan de León, Oaxaca.
Abril 2025

Índice general

1. Aspectos preliminares **3**
- 1.1. Introducción . . . 3
- 1.2. Planteamiento del problema . . . 4
- 1.3. Justificación . . . 4
- 1.4. Hipótesis . . . 5
- 1.5. Objetivos . . . 5
  - 1.5.1. Objetivo General . . . 5
  - 1.5.2. Objetivos específicos . . . 5
- 1.6. Metas . . . 5

2. Fractalidad y Multifractalidad **6**
- 2.1. Ley de Potencia y Escalado . . . 6
- 2.2. Geometría Fractal . . . 6
  - 2.2.1. Dimensión Fractal . . . 7
  - 2.2.2. Auto-similaridad . . . 7
- 2.3. Multifractalidad . . . 7
- 2.4. Métodos de Análisis Multifractal . . . 8
  - 2.4.1. Análisis por re-escalado de Rango R/S . . . 8
  - 2.4.2. Raíz Cuadrática Media (RMS) . . . 10
  - 2.4.3. Análisis de Correlación de Alturas . . . 10
  - 2.4.4. Espectro de Potencias de Fourier . . . 11
- 2.5. Fractales Matemáticos y Naturales . . . 13
  - 2.5.1. Árboles Pitagóricos . . . 13
  - 2.5.2. Curva de Koch . . . 15

3. Apartado Experimental **18**
- 3.1. Sección de montaje . . . 20
- 3.2. Módulo de procesamiento (Hardware) . . . 23
  - 3.2.1. Sensores . . . 23
  - 3.2.2. Parte de potencia . . . 25
  - 3.2.3. Parte electrónica . . . 28

4. Diseño del Software **32**
4.1. Programa para el micro-controlador 33
4.2. Programa para la GUI de muestreo 37
4.3. Programa para el procesamiento de datos 38
4.4. Codificación del Software 40
4.5. Implementación 41
4.5.1. Sección de Montaje 41
4.5.2. Hardware 42
4.5.3. Interfaz gráfica 45

5. Resultados y Discusiones **51**

Conclusiones **54**

A. Apéndice **55**
A. Codificación del programa para el micro-controlador 55
B. Codificación del programa para la GUI de muestreo 58
C. Codificación del programa para el procesamiento de datos 62

Capítulo 1

Aspectos preliminares

1.1. Introducción

El fenómeno del ruido crepitante es una manifestación común en sistemas físicos que liberan energía de manera intermitente y abrupta en respuesta a una excitación externa. Se ha observado en una amplia variedad de materiales y escalas, desde la deformación de metales y polímeros hasta eventos geofísicos como sismos y derrumbes [1]. En términos generales, este fenómeno se caracteriza por la aparición de señales acústicas o variaciones de presión que se generan cuando un sistema acumula tensión y la libera en eventos discretos con distribución de intensidades y duraciones siguiendo leyes de potencia.

Un ejemplo cotidiano de ruido crepitante ocurre en la madera cuando experimenta cambios térmicos. Al calentarse o enfriarse, las variaciones en la expansión térmica generan esfuerzos internos que, al superar un umbral crítico, provocan pequeñas fracturas que emiten ondas acústicas de corta duración [2]. De manera análoga, los materiales sólidos bajo esfuerzos mecánicos pueden experimentar este tipo de liberaciones energéticas, como en el caso de la formación y propagación de grietas en estructuras o el comportamiento de los granos en materiales granularmente consolidados [3].

En los últimos años, investigadores han prestado especial atención al estudio de fenómenos en los que está involucrado el sonido, encontrando que el espectro acústico de diversos sistemas presenta un comportamiento fractal [4]. En particular, el análisis fractal y multifractal ha demostrado ser una herramienta poderosa para caracterizar la dinámica interna de los materiales bajo fenómenos mecánicos, térmicos y de estrés [5]. Este enfoque permite extraer información estructural y de propiedades físicas de los materiales mediante el estudio de sus emisiones acústicas.

El análisis multifractal en señales acústicas ha sido aplicado con éxito en diversos materiales, observándose que ciertos parámetros fractales pueden cambiar cuando se estudian distintos materiales, pero permanecen invariantes cuando se modifica la estructura macroscópica del mismo material [4]. Este hallazgo sugiere que el análisis del espectro acústico podría proporcionar una metodología innovadora para la caracterización de materiales, permitiendo incluso inferir propiedades estructurales de forma no invasiva [4, 6, 7].

Uno de los casos más estudiados ha sido el arrugamiento del papel, donde se ha observado que las emisiones acústicas siguen un patrón fractal y multifractal, proporcionando

información sobre la evolución de su estructura interna [4]. Investigaciones similares han sido realizadas en muestras acústicas generadas por la corteza terrestre [7], con la esperanza de que estos estudios puedan contribuir en el futuro a correlacionar el sonido emitido durante sismos con información sobre la estructura interna de la Tierra.

El estudio del ruido crepitante en materiales sólidos en el contexto de dilatación térmica no solo proporciona una mejor comprensión de las transiciones estructurales internas, sino que también permite el desarrollo de modelos predictivos para evaluar la resistencia de materiales y su estabilidad bajo condiciones de estrés térmico. En este sentido, el análisis multifractal de señales termo-acústicas se ha convertido en una herramienta clave para la caracterización de estas respuestas dinámicas en materiales sometidos a expansión térmica controlada.

1.2. Planteamiento del problema

En la naturaleza, múltiples sistemas exhiben emisiones acústicas asociadas a variaciones de temperatura, fenómeno observado en la madera al calentarse, en la corteza terrestre durante eventos sísmicos y en la fractura de materiales sometidos a estrés térmico. Estas emisiones, comúnmente denominadas ruido crepitante, han sido objeto de estudio en diversos contextos, ya que su análisis puede proporcionar información relevante sobre la dinámica interna de los sistemas que lo generan.

A pesar de la importancia de este fenómeno, actualmente no existen metodologías estandarizadas para la adquisición y análisis de señales termo-acústicas en materiales en expansión térmica. En este trabajo, se desarrollará un sistema experimental compuesto por hardware y software diseñados específicamente para la recolección de datos termo-acústicos, con el objetivo de proporcionar una herramienta que permita registrar, almacenar y procesar estas señales de manera precisa.

Como parte de la validación del sistema, se ha diseñado una prueba experimental utilizando un resistor eléctrico, cuyo calentamiento inducido permitirá evaluar el desempeño del desarrollo implementado.

1.3. Justificación

Dado que en la naturaleza existen múltiples situaciones donde el sonido y la expansión térmica están correlacionados—como en la madera al calentarse o en la actividad sísmica—, el presente estudio se justifica en la necesidad de explorar el análisis multifractal como una herramienta para la caracterización de señales termo-acústicas producidas por la expansión térmica en metales. La implementación del sistema de adquisición de datos termo-acústicos desarrollado en este trabajo permitirá obtener, almacenar y analizar la información acústica con esta herramienta.

1.4. Hipótesis

Los análisis fractales de datos acústicos reportados en la literatura científica, como los estudios sobre el arrugamiento de papel y las emisiones acústicas de la corteza terrestre durante terremotos [4, 6, 7], han demostrado que los valores de sus características fractales no dependen de la forma macroscópica del material emisor, sino de su composición y estructura interna.

En este trabajo, se plantea la hipótesis de que el espectro termo-acústico de sólidos sometidos a expansión térmica exhibe propiedades fractales que siguen el mismo comportamiento de leyes de potencias. Verificar si hay dependencia solo de la composición y estructura internas queda fuera del alcance del mismo.

1.5. Objetivos

1.5.1. Objetivo General

Desarrollar e implementar un sistema experimental que permita la adquisición y análisis de señales termo-acústicas generadas por la expansión térmica de materiales, empleando herramientas de análisis multifractal para caracterizar su comportamiento.

1.5.2. Objetivos específicos

1. Proponer el diseño y la construcción de un sistema experimental que realice la adquisición de datos termo-acústicos, considerando aspectos como el aislamiento de ruido externo y la estabilidad térmica del sistema.

2. Implementar un software capaz de procesar y analizar las señales termo-acústicas adquiridas, empleando técnicas de análisis multifractal.

3. Evaluar el desempeño del sistema experimental mediante ensayos con un sistema de prueba.

1.6. Metas

- Comprender los fundamentos del análisis multifractal y aplicarlos al estudio de señales acústicas.

- Proponer una posible secuencia en el análisis de señales termo-acústicas para la caracterización de materiales en expansión térmica.

- Aplicar los conocimientos adquiridos en cursos de física y matemáticas al diseño e implementación de este experimento.

Capítulo 2

Fractalidad y Multifractalidad

2.1. Ley de Potencia y Escalado

Las distribuciones homogéneas de magnitudes físicas como presión, densidad o temperatura presentan invariancia a la traslación y al cambio de escala. Mandelbrot denominó *escalantes* a los conjuntos que cumplen esta propiedad [8]. Una expresión típica de este comportamiento es la ley de potencia:

$$y(x) = cx^a, \tag{2.1}$$

donde c y a son constantes. Al cambiar de escala $x \to \lambda x$, la ecuación conserva su forma:

$$y(\lambda x) = c\lambda^a x^a = \lambda^a y(x), \tag{2.2}$$

lo que indica la presencia de simetría en todas las escalas. En la práctica, el análisis de datos mediante leyes de potencia se basa en su versión logarítmica,

$$\log y = \log c + a \log x, \tag{2.3}$$

donde el exponente a puede obtenerse de la pendiente en una gráfica log-log. Este tipo de relaciones se ha observado en diversos sistemas físicos, como interfaces en flujo de fluidos, propagación de llamas y formaciones porosas [9, 10].

2.2. Geometría Fractal

La geometría fractal estudia conjuntos con irregularidades a escalas arbitrariamente pequeñas [11]. A diferencia de la geometría euclidiana, que se basa en líneas y superficies regulares, la geometría fractal describe formas mediante algoritmos y propiedades de escalamiento [12]. Su desarrollo, impulsado por Mandelbrot, ha permitido relacionar estructuras matemáticas con fenómenos naturales [8].

Los fractales presentan tres características principales [9]:

- **Dimensión fractal**: Indica el grado de irregularidad y ocupa valores mayormente fraccionarios.

- **Auto-similaridad**: Un fractal contiene copias reducidas de sí mismo en distintas escalas.
- **Auto-afinidad**: Variaciones en las escalas pueden ser anisotrópicas.

2.2.1. Dimensión Fractal

La dimensión fractal cuantifica la complejidad de una estructura. La más común es la dimensión de Hausdorff-Besicovitch, definida como:

$$H^d(E) = \lim_{\rho \to 0} \inf_{\rho_m < \rho} \sum h(\rho_m), \tag{2.4}$$

donde $h(\rho_m)$ representa un recubrimiento óptimo del conjunto E con bolas de radio ρ_m [13]. Un caso ilustrativo es la curva de Koch, cuya dimensión fractal es mayor que su dimensión topológica debido a su auto-similitud infinita [8].

2.2.2. Auto-similaridad

Un fractal es auto-similar si partes de él replican la estructura completa a distintas escalas. Ejemplos clásicos incluyen el triángulo de Sierpinski y la curva de Koch [14]. En la naturaleza, se observa auto-similaridad estadística en fenómenos como la distribución de montañas, estructuras óseas y la ramificación de los árboles [15].

2.3. Multifractalidad

La multifractalidad es una extensión del concepto de fractalidad que permite describir sistemas cuyas irregularidades no pueden ser caracterizadas por un solo exponente fractal, sino que requieren un espectro continuo de dimensiones fractales [8, 13]. Mientras que los fractales tradicionales presentan auto-similaridad con una única ley de escala, los sistemas multifractales muestran una estructura más compleja, en la que distintas regiones exhiben diferentes leyes de escalamiento. Este enfoque es fundamental en el estudio de sistemas con variabilidad en múltiples escalas, tales como turbulencias, estructuras geológicas y mercados financieros [14].

Los sistemas multifractales se dividen en tres categorías principales según la naturaleza de los datos analizados:

- **Multifractalidad en series temporales**: Se analiza la dinámica de sistemas que evolucionan en el tiempo, como variaciones en señales financieras, registros climáticos y flujos turbulentos.
- **Multifractalidad en imágenes y superficies**: Se estudian patrones espaciales complejos, como la textura de materiales, la morfología de superficies naturales y la estructura de imágenes médicas [15].

- **Multifractalidad en distribuciones de probabilidad**: Se aplica en el análisis de sistemas donde la heterogeneidad se manifiesta en la distribución de una cantidad medida, como en la densidad de defectos en materiales o en la distribución de especies en ecosistemas.

Un sistema multifractal se describe mediante el **espectro multifractal**, $f(\alpha)$, que representa la distribución de singularidades locales en el sistema [16]. Este espectro permite cuantificar la diversidad de escalas presentes en los datos, diferenciando regiones de alta y baja complejidad estructural.

Matemáticamente, el análisis multifractal se basa en la medida de partición:

$$\mu_q(l) = \sum_i p_i^q \sim l^{\tau(q)}, \tag{2.5}$$

donde p_i representa la contribución de la caja i de tamaño l a la medida total. A partir de esta relación se obtiene el espectro de singularidades $f(\alpha)$, que describe la frecuencia de ocurrencia de distintos exponentes de escala α en el sistema.

El análisis multifractal proporciona una herramienta poderosa para caracterizar fenómenos complejos, permitiendo identificar estructuras ocultas en datos aparentemente caóticos. Los métodos empleados en este trabajo son: el análisis por re-escalado de rango R/S, el análisis de correlación de alturas y el espectro de potencias de Fourier, cada uno de los cuales ofrece información complementaria sobre las propiedades multifractales de un sistema, las interpretaciones antes mencionadas caen fuera del alcance de esta tesis.

2.4. Métodos de Análisis Multifractal

Existen diversos métodos para analizar sistemas multifractales, cada uno enfocado en distintas características del sistema en estudio. A continuación, se describen cuatro de los métodos más utilizados en el análisis de multifractalidad.

2.4.1. Análisis por re-escalado de Rango R/S

El método R/S, desarrollado por Hurst, se emplea para analizar la memoria a largo plazo en series temporales. Permite distinguir entre procesos aleatorios, persistentes o antipersistentes, midiendo cómo las fluctuaciones en los datos se escalan con el tiempo. El exponente de Hurst (H) obtenido indica si el sistema presenta dependencia a largo plazo o si su comportamiento es completamente aleatorio. Este método fue formalizado por Mandelbrot y Wallis como parte del análisis fractal aplicado a procesos financieros y naturales [8, 17].

El origen del método R/S parte de un problema recurrente al determinar la capacidad de almacenamiento de una presa de agua, basada en una estimación del flujo de agua entrante y la necesidad de agua saliente. El consenso inicial de los expertos fue asumir que el flujo de agua, al ser un problema complejo, es un proceso aleatorio.
Al abordar el problema, el hidrólogo H. E. Hurst desarrolló el análisis de recorrido estandarizado o rango re-escalado, para distinguir sistemas aleatorios y no aleatorios. Partiendo de que

el movimiento browniano se convirtió en el modelo principal para un proceso de caminata aleatoria y que el trabajo previo de Einstein sobre el recorrido de una partícula aleatoria es,

$$R = T^{0.5} \tag{2.6}$$

donde, R es la distancia recorrida y T es un índice de tiempo [8, 17]. Hurst desarrolló una serie de pasos para aplicar su método a un conjunto de datos $X = x_1, ..., x_n$, de tamaño n, con media $x_m = (1/n)\Sigma_{r=1}^{n} x_r$ y desviación estándar $S_n = (1/\sqrt{n})\Sigma_{r=1}^{n}\sqrt{x_r - x_m}$, para $r = 1, .., n$. Los pasos se describen a continuación: crear una serie Z con media cero, es decir, cada elemento $z_r = x_r - x_m$; crear una serie acumulada Y, donde $y_i = \Sigma_{r=1}^{i} z_r$, con $i = 1, \ldots, n$; calcular el rango ajustado R_n para X, de tal manera que, $R_n = \text{máx}(Y) - \text{mín}(Y)$, debido a que Y se ajusta a una media de cero el máximo valor de Y siempre será mayor o igual a cero, y el mínimo siempre será menor o igual a cero, por lo tanto, R_n siempre será no negativo.
Si R_n es la distancia que recorre un sistema durante un índice de tiempo n y se establece $n = T$ la ecuación 2.6 es aplicable siempre y cuando X sea independiente para valores crecientes de n. Para aplicar este concepto a sistemas que no son independientes de n, Hurst encontró que,

$$\frac{R_n}{S_n} = Cn^H \qquad \text{ó} \qquad \frac{R_n}{S_n} \propto n^H \tag{2.7}$$

es una forma más general de 2.6. El valor de R_n/S_n es proporcional a n como una ley de potencia, el exponente H es llamado exponente de Hurst. Esta es la primera conexión del trabajo de Hurst con la geometría fractal [17].
El análisis R/S clásico, desarrollado por Hurst, era tal que el exponente H se estimaba para toda la longitud de una muestra de tamaño N. El procedimiento fue posteriormente modificado por Mandelbrot y Wallis para incorporar la regresión por mínimos cuadrados estimando H en varios subconjuntos de tamaño $n < N$ [5].
El método para obtener el exponente de Hurst incorporando la regresión por mínimos cuadrados aplicado a un conjunto de datos X de tamaño N se describe en [17] de la siguiente manera:

1) Dividir X en A subsecciones de tamaño n, de tal forma que $An = N$. Etiquetar cada subsección de tamaño n como I_a, con $a = 1, .., A$. Cada elemento de I_a se etiqueta como N_{ka}, donde $k = 1, ..., n$. Para cada I_a de tamaño n el valor promedio se define como $e_a = (1/n)\Sigma_{k=1}^{n} N_{ka}$.

2) Calcular una serie de desviación acumulada Y_a para I_a, alrededor de la media de e_a, es decir, cada elemento de Y_a es $Y_{ka} = \Sigma_{i=1}^{k}(N_{ia} - e_a)$.

3) Calcular el rango dentro de cada intervalo I_a, definido como, $R_{Ia} = \text{máx}(Y_{1a}, ..., Y_{na}) - \text{mín}(Y_{1a}, ..., Y_{na})$.

4) Calcular en cada subsección la desviación estándar $S_{Ia} = (1/\sqrt{n})\Sigma_{k=1}^{n}\sqrt{N_{ka} - e_a}$.

5) Calcular el rango re-escalado de I_a defininido como R_{Ia}/S_{Ia}. Dado que existen A subconjuntos de tamaño n, el valor $(R/S)_n = (1/A)\Sigma_{a=1}^{A}(R_{Ia}/S_{Ia})$.

6) Incrementar el tamaño de las subsecciones n al siguiente valor, tal que, N/n es entero. Los pasos 1 al 5 se repiten hasta que este n esté más próximo a $N/2$.

7) Obtenida una serie R/S, con la ecuación 2.7, ejecutar una regresión por mínimos cuadrado con $\log(n)$ y $\log(R/S)$, la pendiente de la linea resultante es la estimación del exponente H de Hurst.

La interpretación del exponente H se divide en tres partes. Primero, si el valor $H = 0.5$ es característico de sistemas aleatorios con elementos independientes o con dependencia a corto plazo, como el caso de la caminata aleatoria. Por lo tanto, la ausencia de dependencia estadística no periódica a muy largo plazo en conjuntos empíricos puede investigarse comprobando si $H = 0.5$. Segundo, si $0.5 < H \leqslant 1$ implica un sistema persistente, que se caracteriza por efectos de memoria a largo plazo, en otros términos, existe una dependencia sensible a las condiciones iniciales. Por último, si $0 \leqslant H < 0.5$ señala un sistema anti-persistente, es decir, debe revertirse con más frecuencia que uno aleatorio [8, 17].

2.4.2. Raíz Cuadrática Media (RMS)

La raíz cuadrática media es un método ampliamente utilizado en el análisis de superficies auto-afines. Se basa en calcular la desviación estándar de una señal o superficie a diferentes escalas para estimar su exponente de rugosidad (α). Este enfoque es útil para caracterizar la complejidad estructural en imágenes y detectar irregularidades en patrones espaciales. En [18] se presentan diversas aplicaciones de este método en modelos de crecimiento de superficies y caracterización de estructuras auto-afines.

A finales de 1980 y 1990 se desarrolló una amplia variedad de modelos de crecimiento que conducen a la formación de superficies auto-afines principalmente modelos para el crecimiento de campos de altura unidimensional $h(x)$. Estos modelos han contribuido al desarrollo de nuevas formas de analizar curvas y superficies auto-afines y brindado oportunidades para evaluar la precisión y confiabilidad de una variedad de enfoques para la medición de los exponentes de Hurst [10]. Gran parte de los trabajos iniciales sobre la caracterización de superficies auto-afines se llevó a cabo utilizando métodos como el conteo de cajas. Sin embargo, la mejor manera para caracterizar los fractales auto-afines es midiendo los exponentes de Hurst.

2.4.3. Análisis de Correlación de Alturas

Este método estudia la variabilidad en la rugosidad de una superficie mediante la función de correlación altura-altura. A diferencia del análisis RMS, permite detectar variaciones locales en la irregularidad, lo que resulta útil para determinar si una estructura es multifractal. En caso de serlo, el exponente de rugosidad varía en función del orden de la correlación. Su uso en la caracterización de estructuras naturales y materiales ha sido documentado en [9, 19]. Para caracterizar las propiedades de escala de algunas interfaces multifractales el exponente de rugosidad α por sí solo no es suficiente. El comportamiento multifractal se refleja en la existencia de una jerarquía completa de exponentes de rugosidad local, es decir, el exponente de rugosidad cambia de un sitio a otro. Por la necesidad de describir propiedades de algunos

fractales complejos se desarrollaron diversos métodos, mismos que pueden ser utilizados para caracterizar funciones multifractales. Para reducir la subjetividad de α se realiza su cálculo varías veces, utilizando diferentes partes de la función y se promedian los valores obtenidos. Para una función auto-afín los valores de α no diferirán significativamente. Cuando diferentes partes de la función tienen valores de α significativamente diferentes, es decir, α no está bien definido, existe alta probabilidad de que la función sea multi-afín. Un método preciso para determinar la multi-afínidad de una función es midiendo la función de correlación de altura-altura de orden q definida como,

$$C_q(l) =< |h(x) - h(x')|^q > \qquad |x - x'| = l \tag{2.8}$$

Donde

$$C_q(l) \propto l^{q\alpha_q} \tag{2.9}$$

llamamos a una interfaz multi-afín si α_q varía con q. Para superficies α_q es independiente de q y es igual al exponente de rugosidad α. Este método de correlación de alturas, definido en superficies puede reformularse en el dominio temporal, reemplazando la posición espacial x por el tiempo t. La función equivalente en una serie temporal $x(t)$ se define como:

$$C_q(\tau) = \langle |x(t) - x(t+\tau)|^q \rangle \tag{2.10}$$

donde:

- $x(t)$ representa la serie temporal (en lugar de la altura en la superficie).
- τ es el desfase temporal, equivalente a la distancia l, ecuación 2.8.

Al igual que en el caso espacial, la relación de escala en una serie multifractal se mantiene:

$$C_q(\tau) \propto \tau^{q\alpha_q} \tag{2.11}$$

donde α_q representa el exponente multifractal.

2.4.4. Espectro de Potencias de Fourier

Este método utiliza la transformada de Fourier para descomponer una señal en sus componentes de frecuencia y analizar cómo se distribuye la energía en diferentes escalas. En un sistema multifractal, el espectro de potencias sigue una ley de potencia caracterizada por un exponente β que cambia a lo largo de la señal, reflejando la presencia de heterogeneidades en la estructura. Este método es ampliamente utilizado en el análisis de señales temporales y espaciales, como se menciona en [5, 20].

El espectro de potencias de Fourier descompone la señal en sus componentes frecuenciales, permitiendo observar cómo se distribuye la energía a lo largo del espectro. Esta descomposición es crucial para detectar comportamientos multifractales, ya que las singularidades locales en el dominio temporal se reflejan en patrones específicos en el dominio frecuencial [5]. La secuencia a seguir es [20]:

1. Aplicar la transformada de Fourier a la señal para obtener su representación en el dominio de la frecuencia.

2. Determinar el espectro de potencia, es decir, el cuadrado del módulo de la transformada de Fourier.

3. Para señales multifractales, el espectro suele seguir una ley de potencia de la forma $S(f) \propto f^{-\beta}$, donde β es un exponente que proporciona información sobre la distribución de las singularidades.

4. Analizar la variación de β en diferentes segmentos de la señal para construir el espectro multifractal, que describe la dimensión fractal de las distintas singularidades presentes.

2.5. Fractales Matemáticos y Naturales

2.5.1. Árboles Pitagóricos

La representación geométrica del teorema de Pitágoras ha influido en el descubrimiento de algunas sorprendentes construcciones geométricas, conocidas como árboles pitagóricos. El más simple de ellos es cuando la hipotenusa de un triángulo rectángulo es $\sqrt{2}$ y sus catetos son 1, el proceso general de la construcción es representado en la figura 2.1.

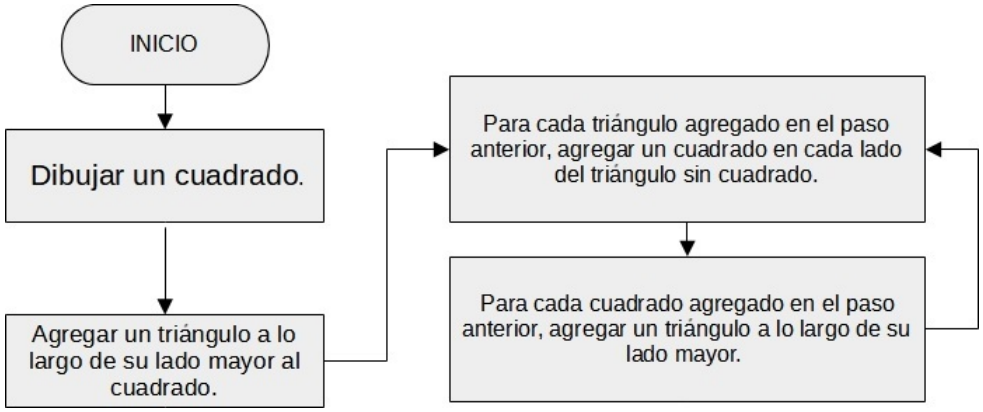


Figura 2.1: Proceso de creación de un árbol pitagórico, en un diagrama de flujo.

A continuación se describen los primeros pasos para el caso particular, donde el triángulo rectángulo tiene por lados $\sqrt{2}$, 1 y 1.

1) Dibujar un cuadrado de lado $\sqrt{2}$.

2) Agregar un triángulo rectángulo en un lado a lo largo de la hipotenusa.

3) Agregar dos cuadrados en cada lado del triángulo.

4) Agregar dos triángulos, uno en cada cuadrado agregado en el paso anterior.

5) Agregar cuatro cuadrados, dos en cada triángulo agregado en el paso anterior.

6) Agregar cuatro triángulos, uno en cada cuadrado agregado en el paso anterior.

Tal como se muestra en el diagrama de flujo, los pasos 3 y 4 se repiten indefinidamente, obteniendo el fractal matemático. En la figura 2.2 se muestran los pasos 1, 2, 3 y 9 del proceso.

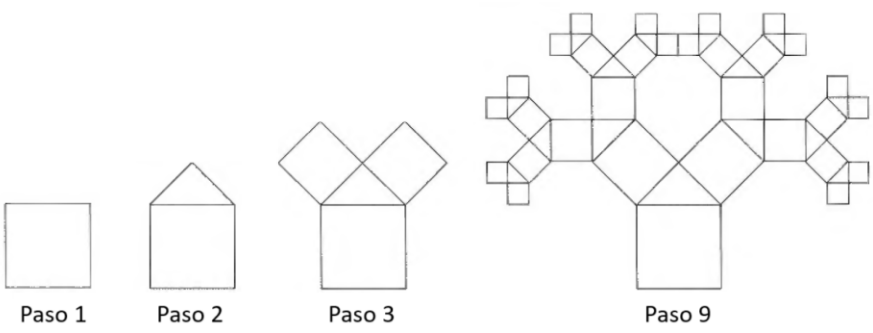
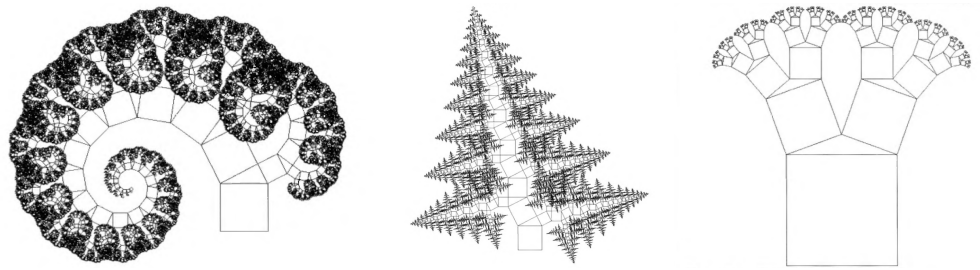


Figura 2.2: Pasos 1, 2, 3 y 9 de la creación de un árbol pitagórico.

Al modificar algunos aspectos de los triángulos en la iteración, se obtiene mayor libertad para crear otros árboles pitagóricos. Por ejemplo, usando un triángulo rectángulo con catetos de diferentes tamaños, voltear su orientación después de cada paso e incluso al cambiar de un triángulo rectángulo a un isósceles con un ángulo mayor a 90°. En las figuras 2.3 se muestran las suficientes iteraciones para poder apreciar la geometría fractal para cada caso mencionado [14]. En definitiva, estas geometrías nos recuerdan a formas presentes en la naturaleza.



(a) (b) (c)

Figura 2.3: Construcción de árboles pitagóricos con mas de 10 iteraciones. (a) Construcción con catetos diferentes y manteniendo la orientación del triángulo. (b) Construcción con catetos de diferente tamaño y con variación en la orientación de triángulo en cada iteración. (c) Construcción con triángulo isósceles de ángulo mayor a 90°.

En el primer caso, figura 2.3a, se puede ver una especie de espiral. Las espirales en la naturaleza son más comunes de lo que parece, algunas de estas se muestran en las figuras 2.4. Es notable la similitud que tiene la figura 2.3a con la geometría de una hoja nueva de helecho a medida que se despliega, figura 2.5a. En el helecho brotando y el primer árbol pitagórico, figura 2.5a y 2.3a, se puede ver una espiral formada por el tronco principal, el tronco principal contiene pequeñas hojas que a su vez forman pequeñas espirales, es decir, cuenta con auto-similaridad.

(a) (b) (c) (d)

Figura 2.4: Espirales en la naturaleza con auto-similaridad estadística. (a) Aloe espiral (Aloe polyphylla) [21]. (b) Caracol moro (Cepaea nemoralis) [22]. (c) Cola de Caballo de mar (Hippocampus barbouri) [23]. (d) Tormenta que gira en la costa del suroeste de Islandia [24].

En caso de la figura 2.3b, esta presenta formas similares a helecho, figura 2.5b, en ambos casos vemos un tallo principal del que se desprenden ramas de derecha a izquierda que a su vez desprenden otras ramas más pequeñas generando un patrón de auto-similaridad. Este mismo patrón se puede observar en un brócoli romanesco partido por la mitad, figura 2.5c. Por último, en la referencia [14] esta configuración es llamada "brócoli como árbol pitagórico" por la evidente similitud con el vegetal cortado por la mitad, figura 2.5d. Cada rama desprendida del tallo principal puede pasar por el objeto completo.

(a) (b) (c) (d)

Figura 2.5: Fractales naturales con formas similares a árboles pitagóricos. (a) Hoja de helecho (Dicksonia antarctica) en proceso de despliegue [25]. (b) Fronda o hoja única de helecho (Dicksonia antarctica) [26]. (c) Brócoli romanesco o romicia (Brassica oleracea var. botrytis) [27]. (d) Brócoli común (Brassica oleracea var. italica) [28].

2.5.2. Curva de Koch

La curva de Koch es un objeto matemático que no contiene líneas rectas ni segmentos suaves. Su construcción es representada por el diagrama de flujo de la figura 2.6, el cual consta de dos procesos, donde el segundo es un proceso recursivo infinito. Los primeros pasos se describen en la figura 2.7a [14].

1) Dibujar una linea recta. Esta también es llamada iniciador.

2) Dividir la linea en tres y remplazar el segmento de en medio por un triángulo rectángulo sin base. Esta forma también es llamada generador.

3) Remplazar cada segmento de linea por el objeto generador.

Al agrupar tres curvas de Koch a lo largo de los lados de un triángulo rectángulo invertido se forma lo que se conoce como "Copo de Koch", figura 2.7b [8], que tiene similitudes con copos de nieve reales, algunos de los cuales se muestran en la figura 2.8a.

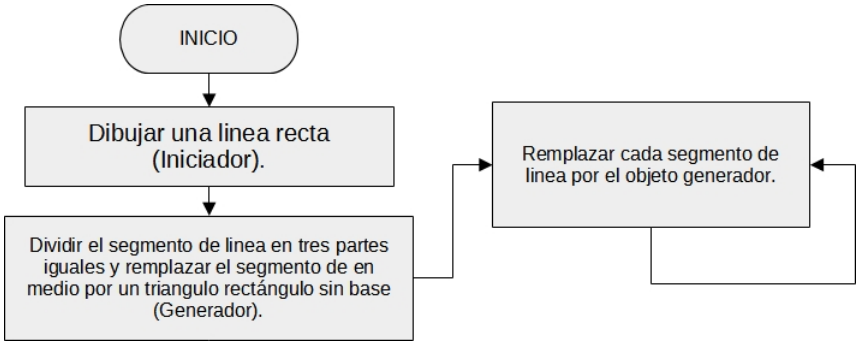
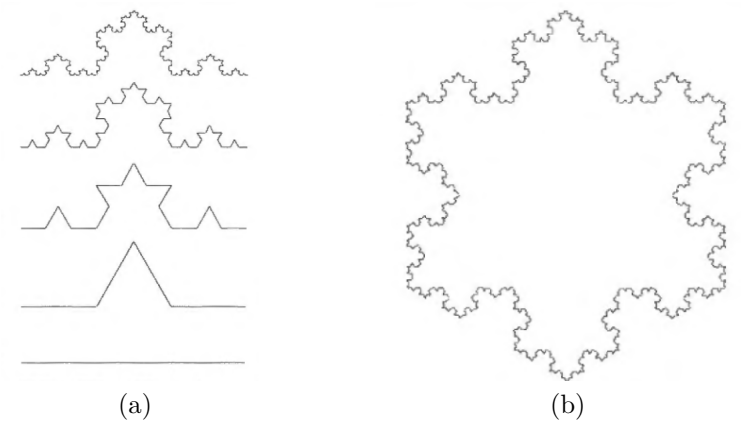


Figura 2.6: Proceso de creación de la curva de Koch, en un diagrama de flujo.

En la naturaleza podemos encontrar formas similares en la construcción de panales de abejas meliponas (sus panales son más irregulares que los de las abejas comunes europeas Apis melífera), en los ojos compuestos de la mosca de fruta, el perímetro de un isla como Gran Canarias y los copos de nieve, figuras 2.8.



(a) (b)

Figura 2.7: (a) Representación gráfica de los cinco primeros pasos para la construcción de la curva de Koch. (b) Copo de Koch, formado por tres curvas de Koch a lo largo de los lados de un triángulo rectángulo invertido.

(a) (b) (c) (d)

Figura 2.8: Fractales naturales con patrones similares a la curva de Koch y sus variaciones. (a) Fotografías de copos de nieve hechas por Wilson Bentley [29]. (b) Panales de abeja melipona [30]. (c) Imagen satelital de la isla Gran Canaria [31]. (d) Ojos compuestos de una mosca de la fruta [32].

Capítulo 3

Apartado Experimental

El estudio de señales acústicas mediante métodos multifractales requiere una amplia biblioteca de muestras [33]. Al querer aplicar métodos de análisis multifractal a datos termo-acústicos en la Universidad Tecnológica de la Mixteca, la recolección de datos implicaba una serie de pasos repetitivos, figura 3.1a, los cuales, mediante el uso de software y hardware, pueden ser simplificados a los pasos mostrados en la figura 3.1b.

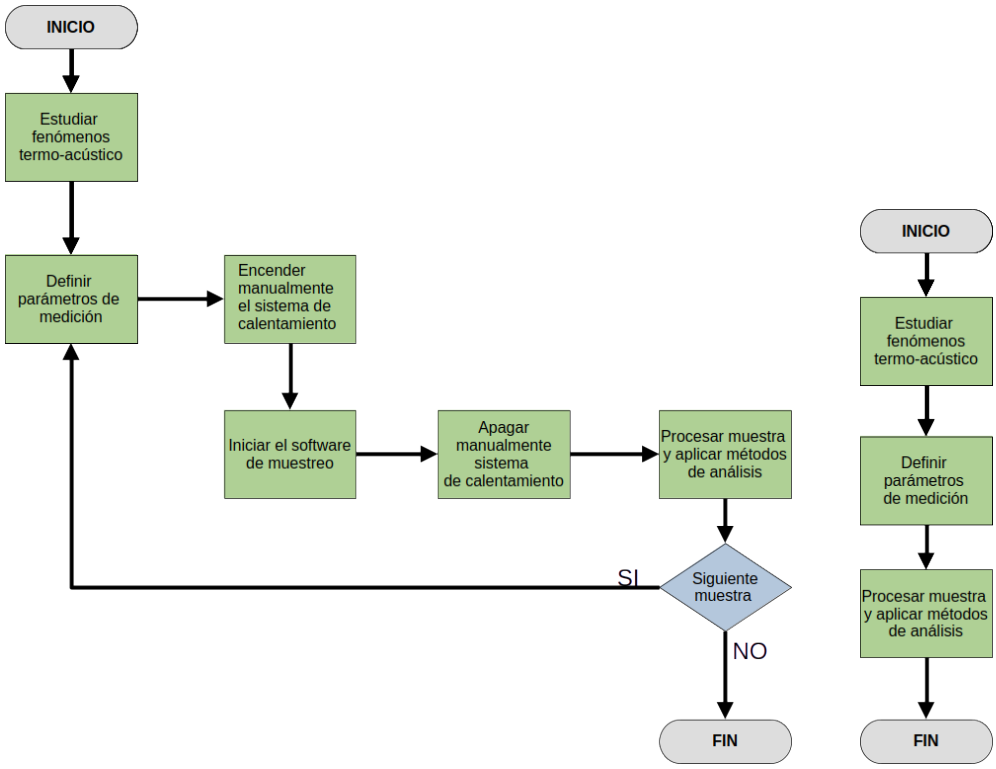


Figura 3.1: Comparación de procesos para obtener, procesar y analizar datos termo-acústicos. (a) Proceso implementado previo al desarrollo de este trabajo. (b) Proceso implementado en el dispositivo desarrollado.

El objetivo de esta sección es describir el experimento concebido para recopilar datos acústicos durante el proceso de expansión térmica, con el propósito de realizar un análisis multi-fractal por el usuario. El experimento está conformado por una resistencia eléctrica sujeta a una diferencia de potencial, la cual se mantiene mediante una base de lámina negra de hierro. Este proceso genera chasquidos significativos. La esencia primordial de esta tesis radica en la creación de un dispositivo que capture el espectro acústico de estos chasquidos, permitiendo la definición de parámetros a través de una interfaz de usuario. Además, se implementará una interfaz de usuario adicional cuya finalidad será el procesamiento y análisis multi-fractal de los datos. Este enfoque se orienta a proporcionar una herramienta para caracterizar la acústica generada durante la expansión térmica en el sistema objeto de estudio.

En general el experimento se divide en tres partes: la sección de montaje, el módulo de procesamiento y la interfaz de usuario (GUI), como se aprecia en la figura 3.2.

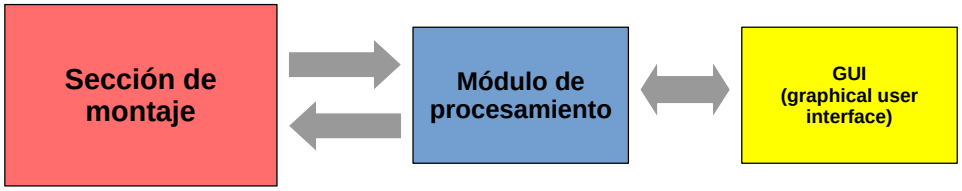


Figura 3.2: Diagrama a bloques del experimento.

- Sección de montaje: el fenómeno físico a estudiar es el sonido producido durante la expansión térmica en metales, para esto se propone el uso de una resistencia eléctrica de una parrilla convencional sostenida por una base de metal como se muestra en la figura 3.3.
- Módulo de procesamiento: cumple la función de procesar la información de los sensores y controlar los actuadores. Además, recibe y manda información a la GUI.
- GUI: permite al usuario definir y monitorear las variables del experimento.

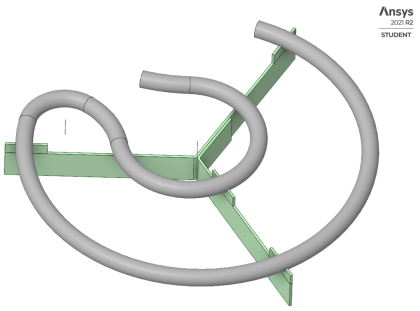


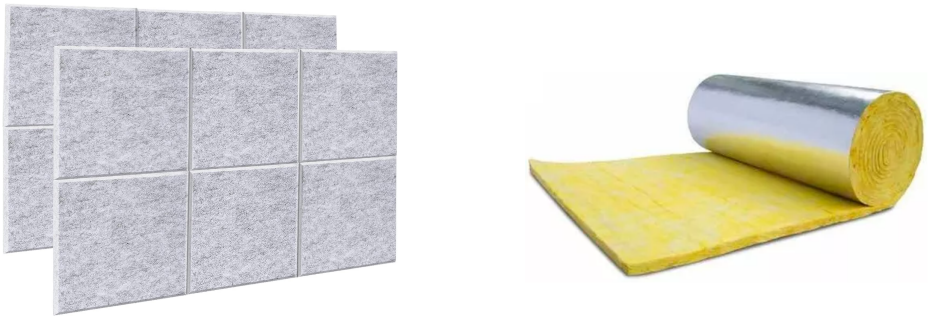
Figura 3.3: Sistema para estudiar el espectro acústico durante la expansión térmica en metales.

3.1. Sección de montaje

Para la sección de montaje se tuvieron en cuenta las siguientes características:

- Sistema a prueba de eco.
- Aislamiento térmico.
- Acceso a sensores de audio y temperatura.
- Resistencia eléctrica con acceso a una diferencia de potencial de linea ($120V_{RMS}$).
- Facilidad para la manipulación del sistema de estudio.

Al tratar con datos acústicos surge la necesidad de evitar medios que puedan contaminar este tipo de datos, como el eco, para esto se consideraron paneles acústicos. Los paneles AGP-TEK, de dimensiones $30 \times 30 \times 1$ cm, están compuestos por fibras de poliéster con retardante de fuego. En cuanto al aislamiento térmico-acústico, para separar la fuente de calor de otros dispositivos, se emplea el material de fibra de vidrio y aluminio de OWENS CORNING. Este aislante presenta un rango de temperaturas de operación de hasta 232°C y cumple la función adicional de proporcionar aislamiento acústico.



(a) (b)

Figura 3.4: Materiales para la sección de montaje. (a) Paneles acústicos contra eco. (b) Aislante térmico de fibra de vidrio.

La resistencia eléctrica considerada es una resistencia comercial diseñada para parrillas eléctricas, como se muestra en la figura 3.5. Estas resistencias cuentan con terminales de latón tipo Faston. Por conveniencia, se ha etiquetado la figura 3.5 como R1.

Figura 3.5: Resistencia eléctrica para la sección de montaje, etiquetada como R1.

Una vez definidas las características y materiales disponibles para la sección de montaje, se procedió a planificar la distribución de cada componente del sistema. La figura 3.6 ilustra la distribución de los elementos, donde 12 paneles acústicos se disponen para formar una caja sin techo. Con el propósito de facilitar la manipulación de los elementos dentro de la caja, se posiciona un cuadrado de aislante térmico en el centro, donde reposa una resistencia eléctrica sobre una base de hierro. Los bloques azul y rojo en la representación indican la ubicación de los sensores de audio y temperatura, respectivamente.

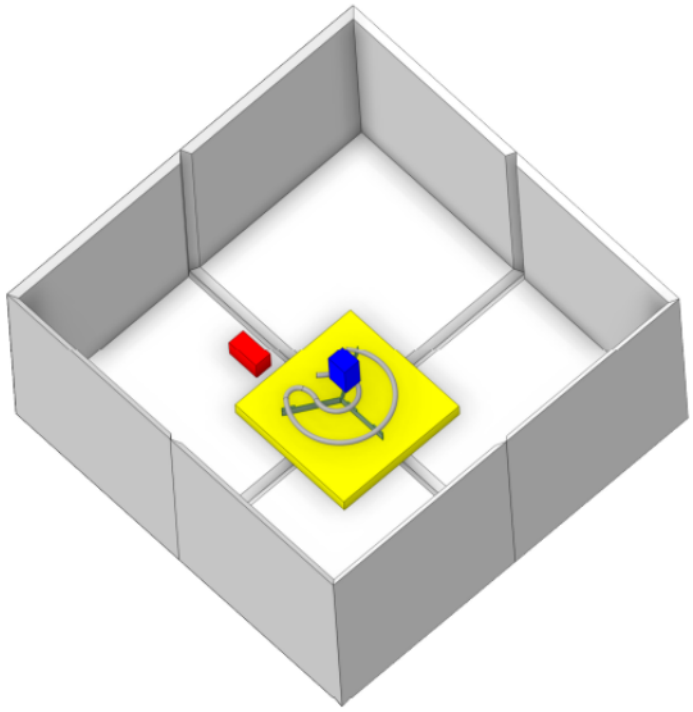
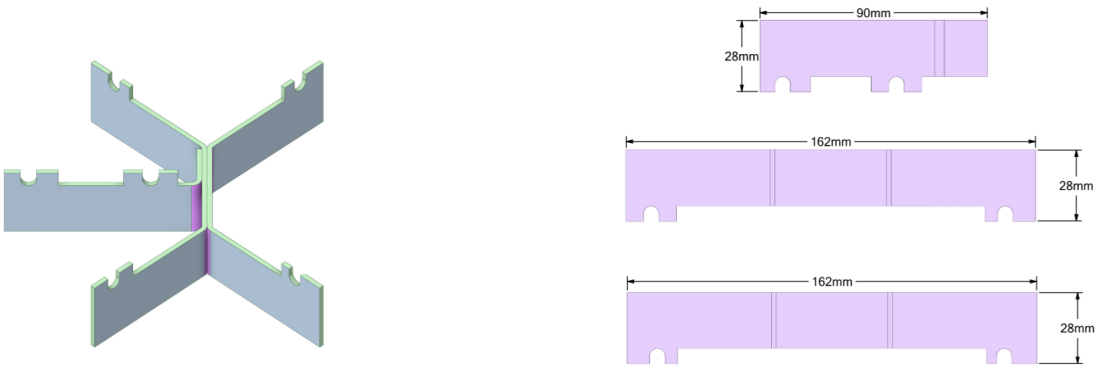


Figura 3.6: Representación de la sección de montaje modelado con SpaceClaim.

Para llevar a cabo las pruebas, se modeló una base para la resistencia utilizando lámina negra de hierro calibre 18, como se visualiza en la figura 3.7. Para el proceso, se perfiló la forma de la resistencia en una hoja de papel, y posteriormente se importó al software SpaceClaim. Utilizando las herramientas específicas para metales y plegado en el software, se modelaron los componentes de las bases, tomando como referencia las formas y posiciones de las muescas presentes en bases comerciales de parrillas eléctricas.
El software tiene en cuenta la distancia necesaria para el plegado y ofrece la forma y medidas del modelo desplegado, listo para imprimir, tal como se muestra en la figuras 3.7. Para mayor practicidad, se añadió el subfijo b1 designándose así como R1b1, a la combinación de la resistencia R1 con la base b1.



(a) (b)

Figura 3.7: Bases modeladas en SpaceClaim. (a) Base R1b1. (b) Base R1b1 desplegada.

3.2. Módulo de procesamiento (Hardware)

Considerando el diagrama de la figura 3.2, las partes esenciales del hardware forman parte del "Módulo de Procesamiento", encargado de procesar la información proveniente de los sensores y el actuador. El actuador corresponde es R1, seleccionado en la sección anterior. En cuanto a la elección de sensores y los demás componentes, se fundamenta en las siguientes características:

- Rango de temperaturas comprendido entre 20°C y 100°C.
- Control de rangos de temperatura y número de muestras a medir.
- Almacenamiento de datos de temperatura y audio.
- Disposición de una interfaz frontal considerando: LCD (pantalla de cristal líquido por sus siglas en inglés) e interruptor.
- Comunicación a través del puerto serial con la computadora personal.
- Implementación de una interfaz de usuario mediante la computadora personal.

3.2.1. Sensores

Los sensores esenciales para el sistema se dividen en dos categorías: electroacústico (representado por un micrófono), y sensor de temperatura. En este contexto, el sensor electroacústico seleccionado es un MOV-034, micrófono de solapa de la marca STEREN. Las características más relevantes de este dispositivo se detallan a continuación [34]:

- Respuesta en frecuencia entre 20 y 200000Hz.

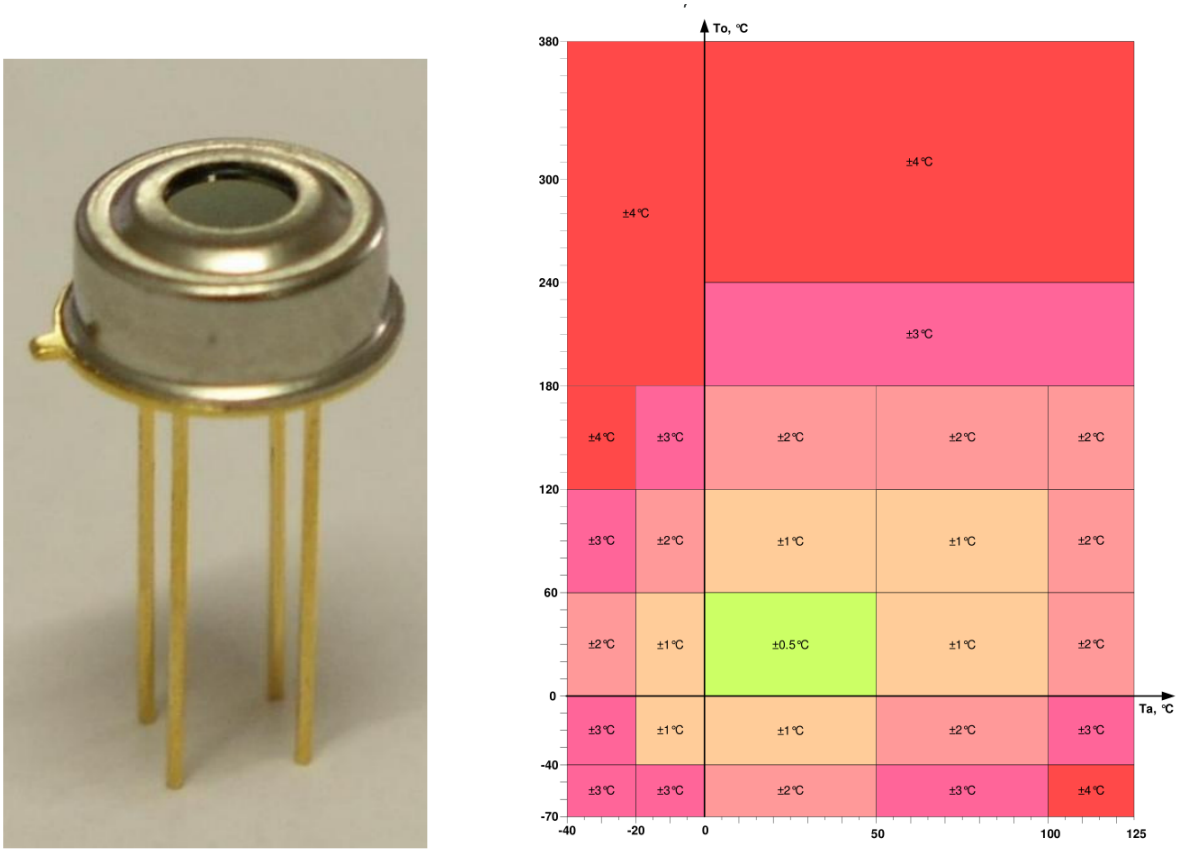
- Impedancia de 900Ω.
- Sensibilidad de $86 \pm 3dB$.

El micrófono se conectó directamente a la computadora personal a través de la entrada de audio de $3.5mm$.

Figura 3.8: Micrófono de solapa $3.5mm$ MOV-034 [35].

El sensor de temperatura empleado es del tipo MLX90614, sensor fabricado por Melexis Microelectronics Integrated Systems, tal como se muestra en la figura 3.9a. Sus características más destacadas incluyen [36]:

- Sensor con calibración en grados Celsius [°C].
- Rango de temperatura ambiente entre −40°C y 85°C.
- Rango de temperatura objeto −40°C y 125°C.
- Voltaje de operación de $3.3V$.
- Precisión de hasta ±0.2°C, figura 3.9b.



(a) (b)

Figura 3.9: Sensor de temperatura infrarrojo. (a) Encapsulado sensor MLX90614. (b) Precisión de MLX90614 en términos de Ta (temperatura ambiente) y de To (temperatura objeto) [36].

Las conexiones del sensor se detallan en la librería para Arduino de SparkFun [37]. Tras seleccionar los sensores y actuador, se eligieron los demás componentes considerando su voltaje de operación. El actuador trabajan con voltajes de corriente alterna (CA) de $120V_{RMS}$, mientras que los sensores operan con voltajes de corriente continua (DC) no superiores a 5V. Los circuitos se pueden clasificar en dos partes: la parte de potencia (para altos voltajes) y la parte electrónica (bajos voltajes).

3.2.2. Parte de potencia

Esta etapa es la encargada de suministrar energía al actuador (resistencia eléctrica) para su correcto funcionamiento, al mismo tiempo que aísla la parte electrónica de los voltajes elevados mediante el uso de un opto-acoplador.
La hoja de datos del optoacoplador MOC3010 ofrece configuraciones para las aplicaciones más frecuentes, y en nuestro caso, la disposición presentada en la figura 3.10 resulta apropiada para los objetivos de este trabajo [38].

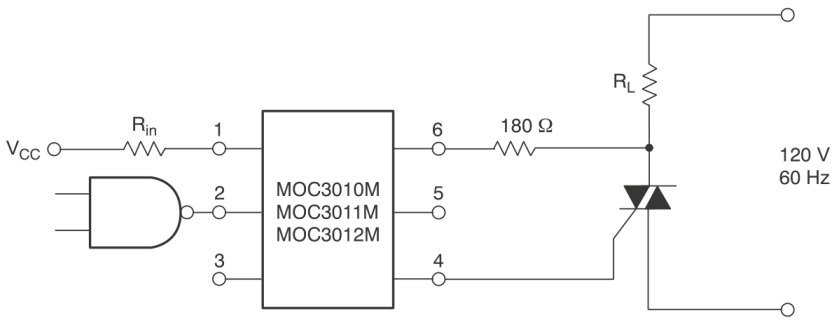


Figura 3.10: Circuito de potencia. Donde V_{CC} es el voltaje proporcionado por el microcontrolador, R_{in} es una resistencia cuyo valor depende de V_{CC} (esta resistencia delimita la corriente para el correcto funcionamiento del opto-acoplador) y R_L es el actuador (resistencia eléctrica) [38].

El valor de R_{in} se calcula del análisis de la malla correspondiente, esto es, $-V_{CC}+i_F R_{in}+V_D=0 \Rightarrow R_{in}=(V_{CC}-V_d)/i_F$. Donde $V_{CC}=5V$ (voltaje del micro-controlador), i_F es la corriente que pasa por R_{in} y V_D es el voltaje entre las terminales 1 y 2 del opto-acoplador (MOC3010), la hoja de datos del componente [38] recomienda corrientes de operación i_F entre $15mA$ y $60mA$ lo que corresponde a voltajes entre $1.25V$ y $1.45V$ respectivamente, entonces, los valores para R_{in} están entre 59.16Ω y 250Ω.
La elección del TRIAC está determinada por la corriente requiere R_L a un voltaje CA de $120V$ a $60Hz$, siendo esta corriente de $6A$. El TRIAC BTA08-800BW cumple con estas especificaciones [39].

(a) (b)

Figura 3.11: Encapsulados para circuito de potencia. (a) Opto-acoplador MOC3010. (b) TRIAC BTA08-800b.

La figura 3.12 presenta la reconstrucción del circuito de la figura 3.10 utilizando el software Proteus 8 Professional. La señal “s” representa los pulsos suministrados por el microcontrolador, oscilando entre 0 y 5 V. Las resistencias $R1$ y $R3$, figura 3.12, limitan la corriente a sus respectivos diodos para asegurar su correcto funcionamiento, mientras que $R2$ corresponde a

la resistencia sugerida en la figura 3.10. R_L simboliza el actuador y se representa como una lámpara, permitiendo visualizar su comportamiento. La fuente de corriente CA de $120V$ a $60Hz$ está representada por V1.

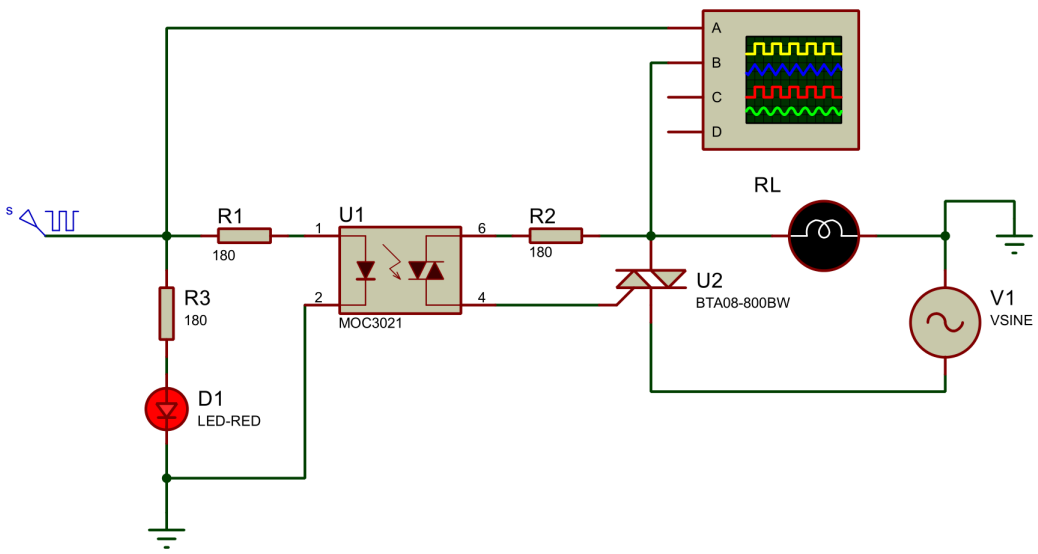


Figura 3.12: Representación del circuito de potencia realizado en Proteus 8 Professional. Este circuito incorpora un LED conectado a una resistencia de 180Ω que se utilizó para visualizar la señal del micro-controlador.

El funcionamiento del circuito consiste en encender y/o apagar el actuador mediante el uso de un TRIAC, controlado con una señal proveniente de un micro-controlador. Cuando R_L deba encenderse el micro-controlador manda una señal que activa el opto-acoplador el cual, a su vez, activa el TRIAC, permitiendo que R_L se encienda. Cuando R_L deba estar apagado el micro-controlador deja de mandar la señal y el TRIAC deja de estar activo.

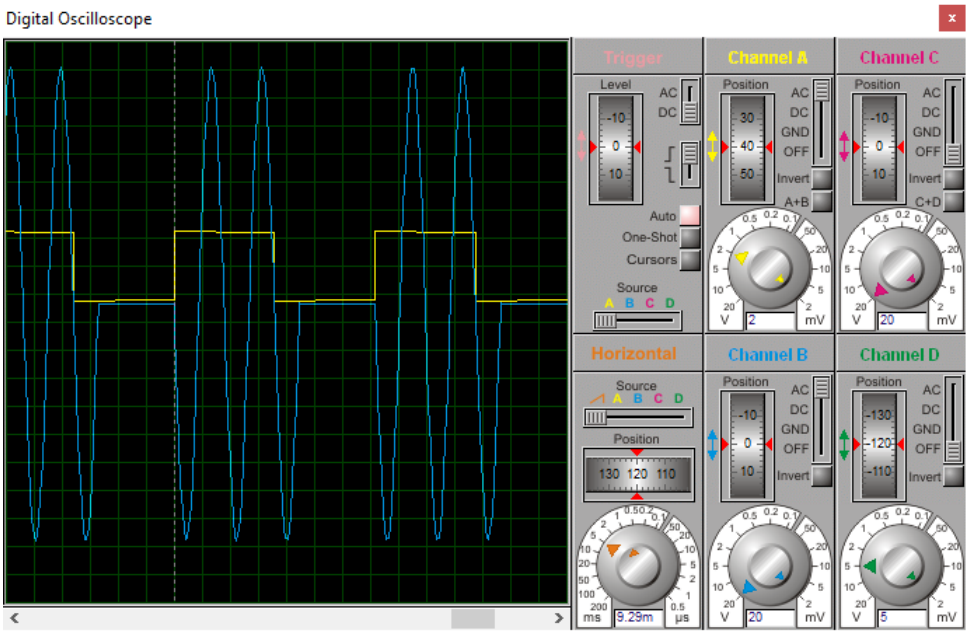


Figura 3.13: Simulación de voltajes del actuador y micro-controlador. El voltaje de RL es mostrado a un orden de 20^1 mientras que el voltaje en el micro-controlador es del orden de 20^{-1}.

En la figura 3.13, se muestra la simulación del circuito representado en la figura 3.12. Las gráficas reflejan los voltajes de la señal del micro-controlador (señal amarilla) y el voltaje en R_L (señal azul). Durante las crestas de la señal del microcontrolador, el TRIAC está en conducción, mientras que en los valles, se encuentra apagado.

3.2.3. Parte electrónica

La parte electrónica es la encargada de controlar la señal que recibe la parte de potencia, controlar la interfaz frontal y establecer comunicación con la computadora. Su componente principal es un microcontrolador y, para esta aplicación, un Arduino UNO cumple con los requisitos. Algunas de sus características más destacadas son:

- Procesador ATMega328P.
- 32KB Flash.
- 2KB SRAM.
- 1KB EEPROM.
- Velocidad de reloj 16 MHz.

Dado que es necesario recuperar y guardar los datos de temperatura también se hace uso de un módulo de lectura y escritura de tarjeta SD compatible con Arduino UNO modelo OKY3001. Las conexiones se muestran en la figura 3.14 [40].

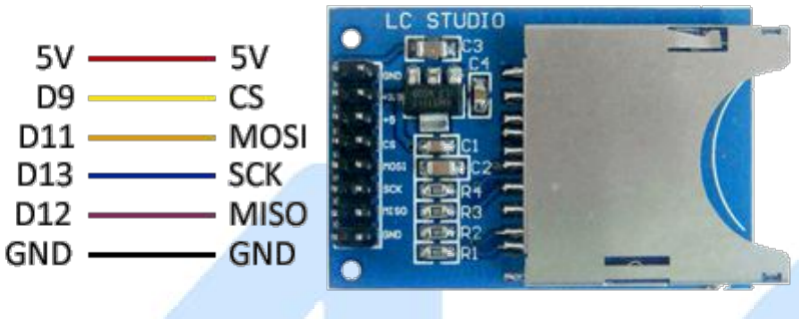
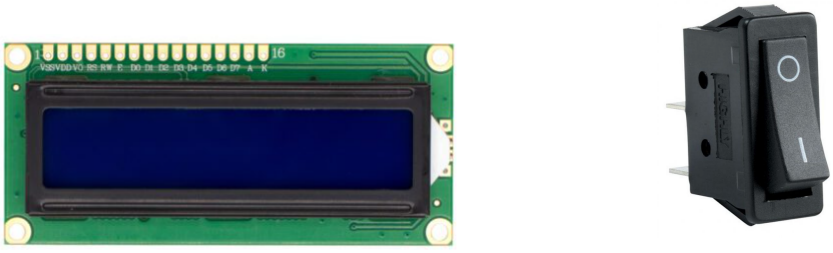


Figura 3.14: Diagrama de conexiones, Arduino UNO a módulo de lectura y escritura de tarjeta SD modelo OKY3001 [40].

El resto de hardware corresponde a los componentes que se mostrarán en el panel frontal del módulo de procesamiento, es decir, un display (LCD), un switch de balancín y un led rojo. En el LCD de despliegan errores de conexión y las temperaturas a tiempo real del objeto y ambiente del fenómeno físico. El switch tendrá la función de permitir o no el paso de la corriente de línea CA al actuador. El led rojo indicará los momentos en que el actuador está encendido.
El display LCD usado cuenta con las siguientes características [41]:

- Pantalla LCD Monocromática.
- Voltaje de alimentación de 5V DC.
- Interfaz de comunicación paralelo 4 u 8 bits.
- 2 filas y 16 columnas.
- Controlador HD44780.
- Fondo azul y texto blanco.
- Modo de operación de 4 y 8 bits.
- Corriente máxima de 25mA.



(a) (b)

Figura 3.15: Componentes para el panel frontal del módulo de procesamiento. (a) LCD 2x16 [41]. (b) Switch de balancín BTS-10 [35].

Las conexiones del LCD se especifican en [41].

El switch de balancín usado es el modelo BTS-10 distribuido por Steren, sus características son las siguientes [35]:

- 1 polo, 1 tiro, 2 posiciones.
- Soporta 125/250 V_{CA} 12/10 A.

El procesamiento de la parte electrónica se lleva a cabo mediante un firmware, como se explica en la siguiente sección. El diagrama de conexión de la parte electrónica se presenta en la figura 3.16.

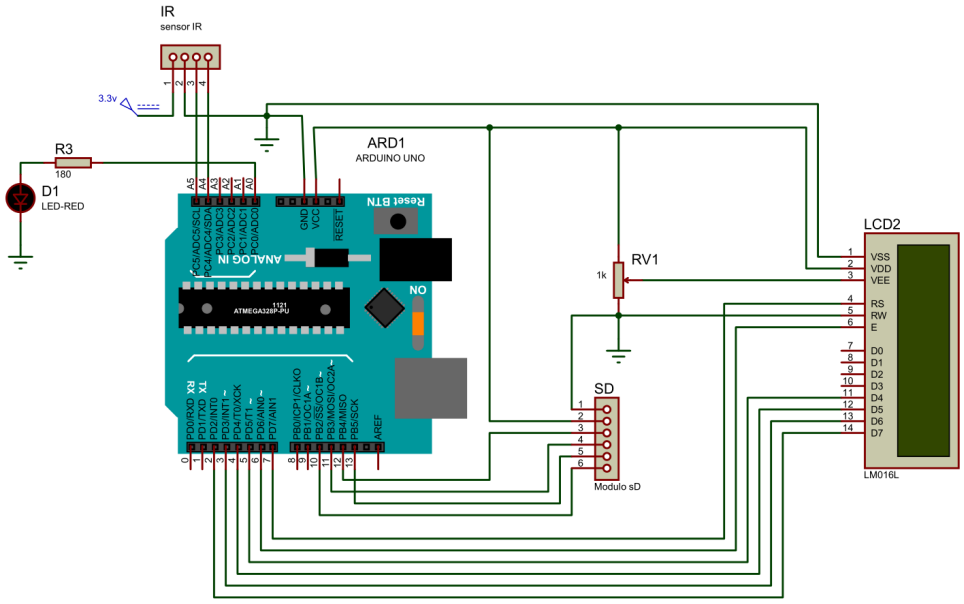


Figura 3.16: Diagrama de conexiones, Arduino UNO, LCD, adaptador micro SD y sensor de temperatura IR.

La comunicación con la GUI se realiza mediante puerto serial, por lo cual, podrá mandar y recibir datos. En la figura 3.17 se puede ver el diagrama a bloques detallado del diseño del hardware.

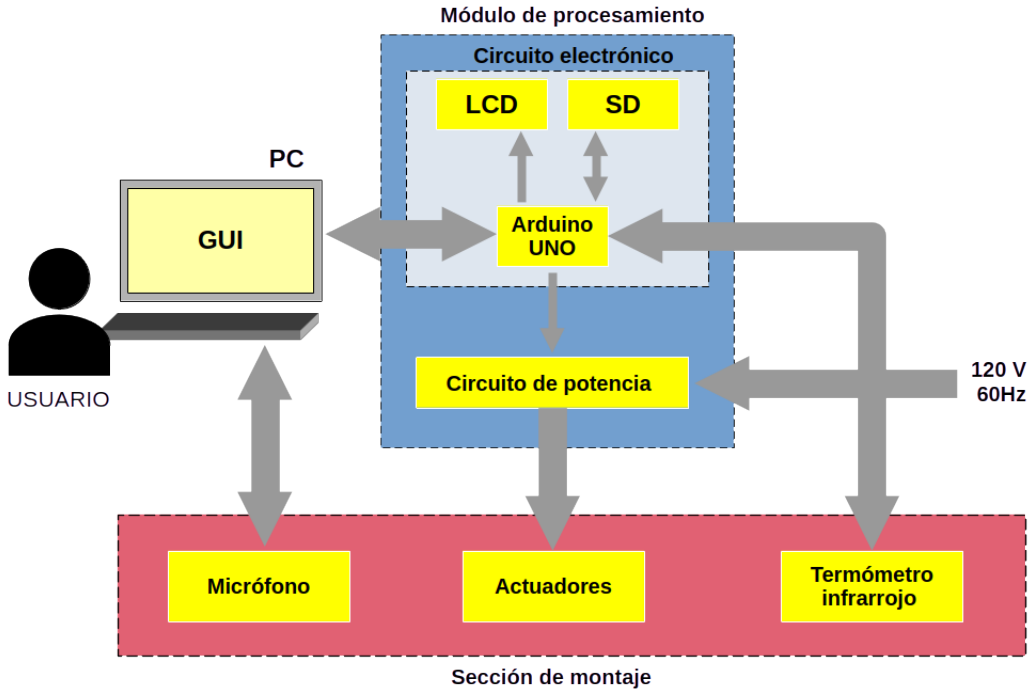


Figura 3.17: Diagrama a bloques detallado del diseño del hardware.

Capítulo 4

Diseño del Software

El software juega un papel importante en el funcionamiento del sistema de control, es el encargado de coordinar las acciones de actuadores y sensores, así como procesar la información obtenida. El modelo de desarrollo implementado para el software de este proyecto es el modelo de desarrollo evolutivo incremental, este modelo de desarrollo permite crear versiones de software cada vez más complejas, entregando avances pequeños pero funcionales de manera que cada vez el software queda más complejo [42]. La figura 4.1 esquematiza el modelo de desarrollo que consta de cuatro etapas principales:

- Análisis: consiste en la comprensión de los requerimientos iniciales, función, comportamiento e interconexión de los programas a construirse.
- Diseño: proporciona la funcionalidad del programa a través de sus diferentes componentes.
- Codificación: lleva el diseño a un lenguaje de programación que pueda ser interpretado por una computadora.
- Pruebas: se verifican que se cumplan los requerimiento del usuario, caso contrario se obtienen los nuevos requerimientos para comenzar de nuevo el ciclo hasta obtener un software que satisfaga todas las necesidades del usuario.

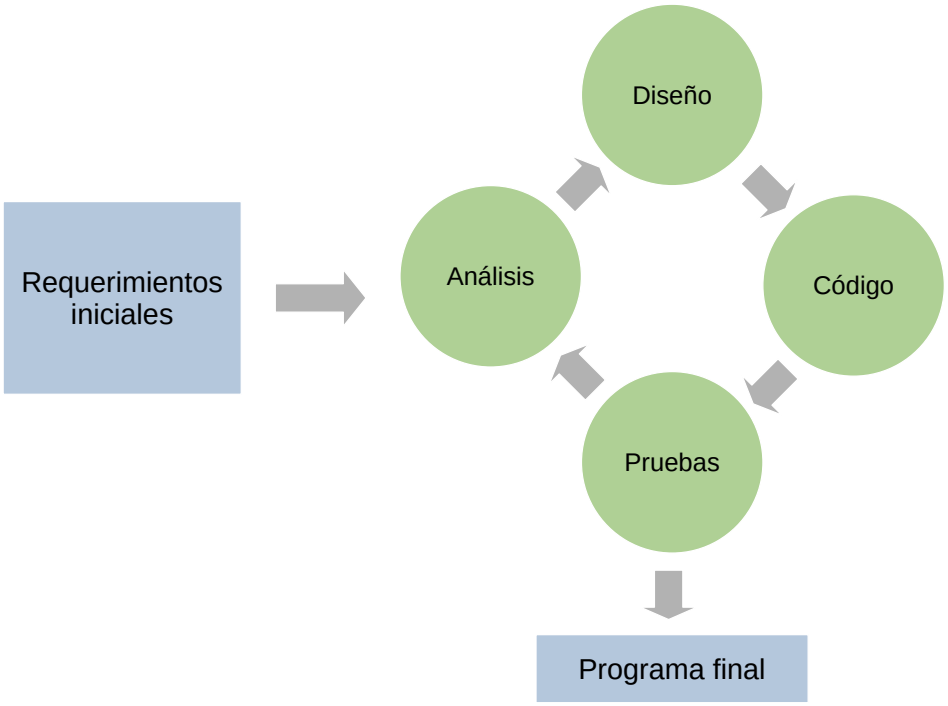


Figura 4.1: Esquema del modelo evolutivo incremental.

La función principal del software en este proyecto es proporcionar al usuario el control para definir un intervalo de temperaturas y tiempo durante el cual se tomará una muestra de audio y temperatura. Además, debe ser capaz de realizar este procedimiento varias veces bajo las mismas condiciones y facilitar el análisis multi-fractal de los datos. El software se divide en tres partes: el programa para el microcontrolador, el programa para la interfaz gráfica de usuario (GUI) de muestreo, y la GUI de procesamiento y análisis. Las dos primeras partes se centran en el dispositivo de muestreo, mientras que la tercera se dedica al procesamiento y análisis. A partir de ahora, por practicidad, cada parte se denominará **SmicroControlador**, **SguiMuestreo** y **SguiProcesamiento**, respectivamente.
Para representar el diseño de los algoritmos que codifican las partes del software, se utilizan herramientas como diagramas de flujo y pseudocódigo. La idea principal del diseño del software para el dispositivo es que el programa para **SmicroControlador** realice los procesos cuando **SguiMuestreo** se lo indique.

4.1. Programa para el micro-controlador

De forma general, el funcionamiento del programa para el micro-controlador **Smicro-Controlador** puede ser descrito mediante el diagrama de flujo de la figura 4.2.

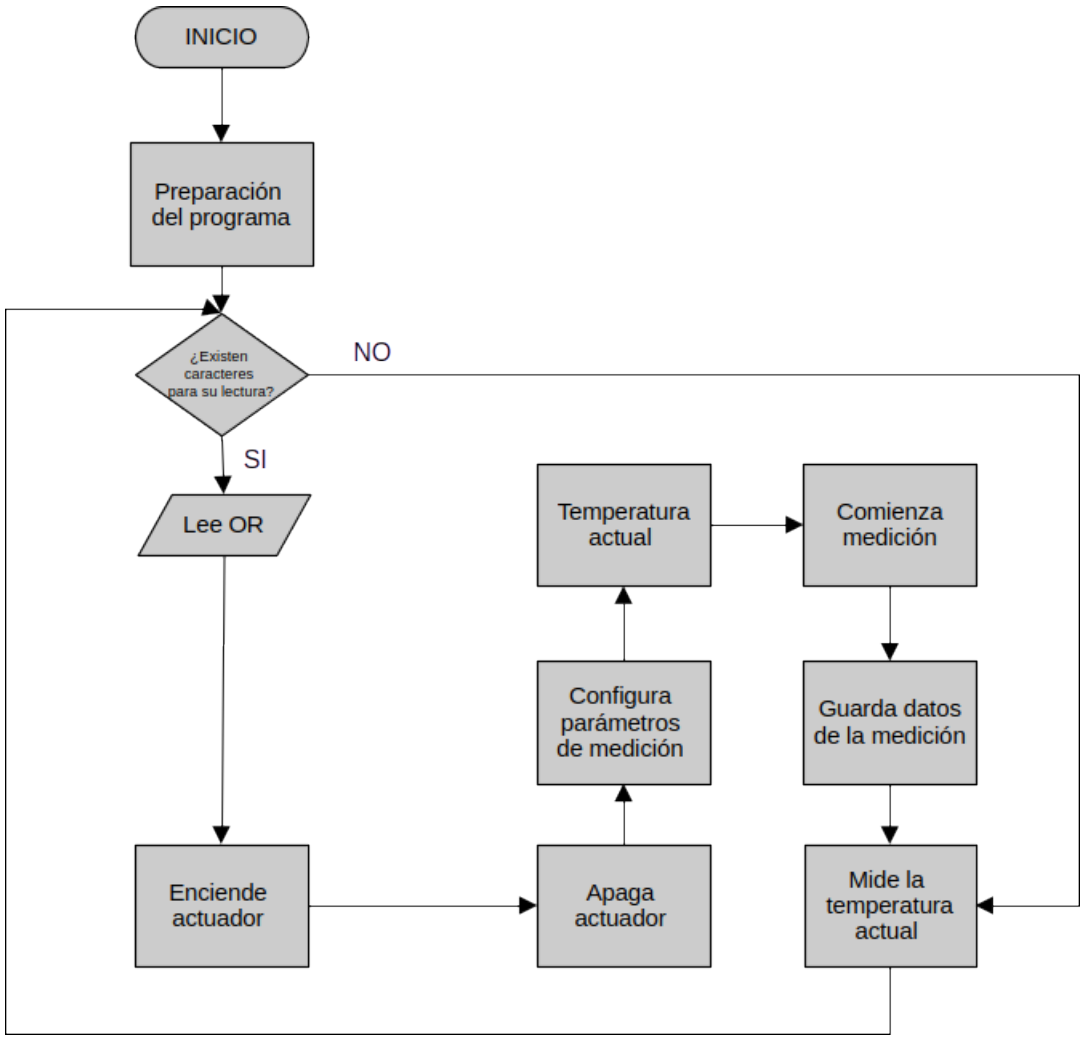


Figura 4.2: Diagrama de flujo del programa del micro-controlador.

1.- Proceso `Preparación del programa`: verifica el correcto funcionamiento del hardware.

2.- Condicional `¿Existen caracteres para su lectura?`: se asegura que los demás procesos se ejecuten por las órdenes de la GUI.

3.- Entrada de datos `Leer OR`: recibe datos del puerto serie y los guarda en la variable `OR`.

4.- Proceso `Enciende actuador`: activa el circuito de potencia, figura 4.3a.

5.- Proceso `Apagar el actuador`: desactiva el circuito de potencia, figura 4.3b.

6.- Proceso `Configura parámetros de medición`: lee información del puerto serie y establece los parámetros para la medición, figura 4.3c.

7.- Proceso `Temperatura actual`: mide la temperatura actual del sistema e imprime el dato en el LCD y puerto serie, figura 4.3d.

8.- Proceso `Comenzar medición`: mide la temperatura del sistema, imprime en el LCD, enciende y apaga el sistema de potencia y guarda los datos de temperatura en una memoria microSD, figura 4.3e.

9.- Proceso `Guardar datos de la medición`: extrae los datos de la memoria microSD y los escribe en el puerto serie, figura 4.3f.

10.- Proceso `Medir la temperatura actual`: Mide la temperatura actual del sistema e imprimir en el LCD, este proceso realizará mientras que la GUI no mande datos al puerto serie.

El orden en que se ejecutarán los procesos de la figura 4.2 será determinados por la variable `OR` cuyos valores posibles son `'1','0','a','b','c','d','e','f','g'`, este valor será definido por el usuario a través de **SguiMuestreo**.

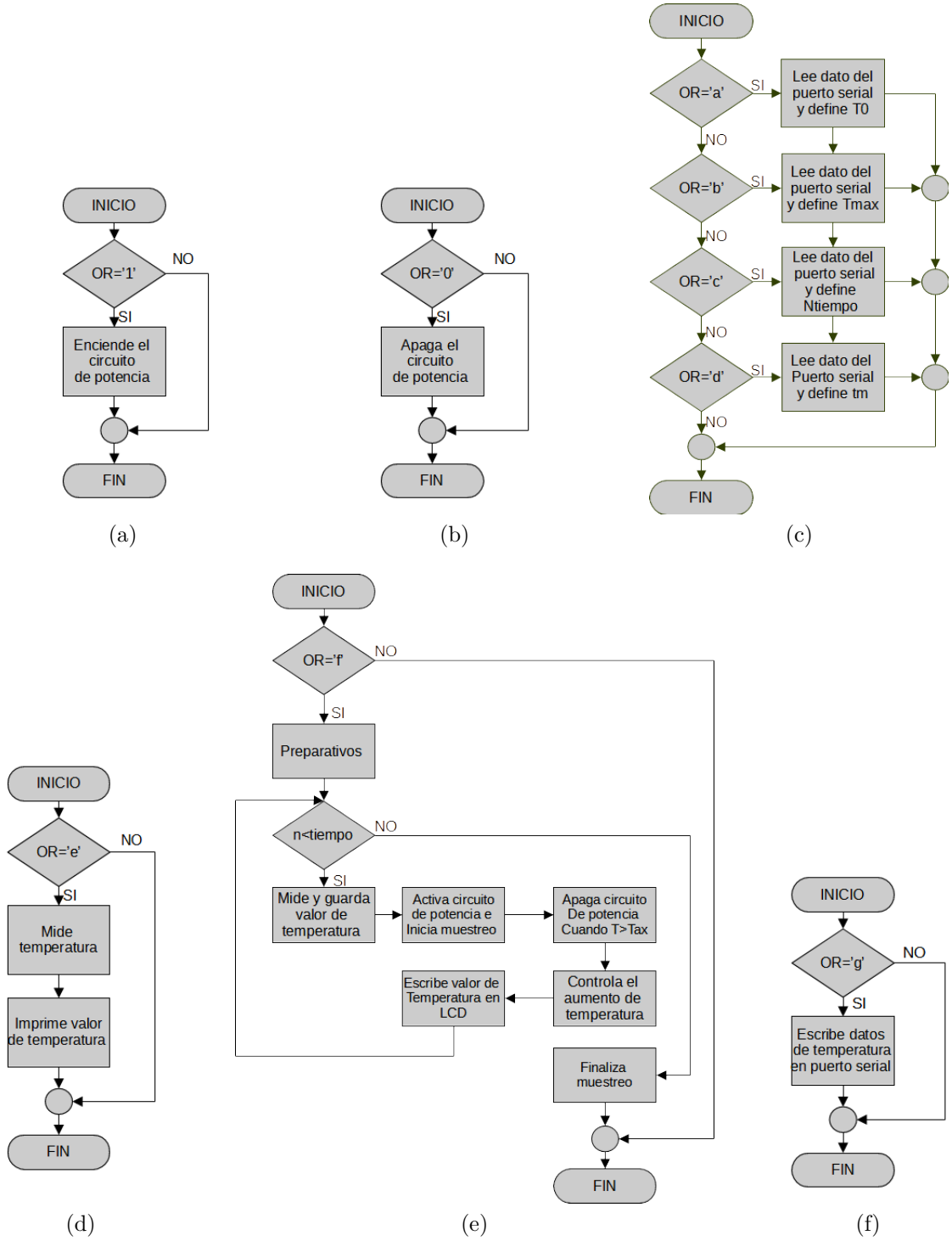


Figura 4.3: Procesos detallados. (a) Enciende actuador. (b) Apaga actuador. (c) Configura parámetros de medición. (d) Temperatura actual. (e) Comienza medición. (f) Guarda datos de la medición.

4.2. Programa para la GUI de muestreo

El funcionamiento general del programa para la GUI de muestreo **SguiMuestreo** es descrito en el diagrama de flujo de la figura 4.4.

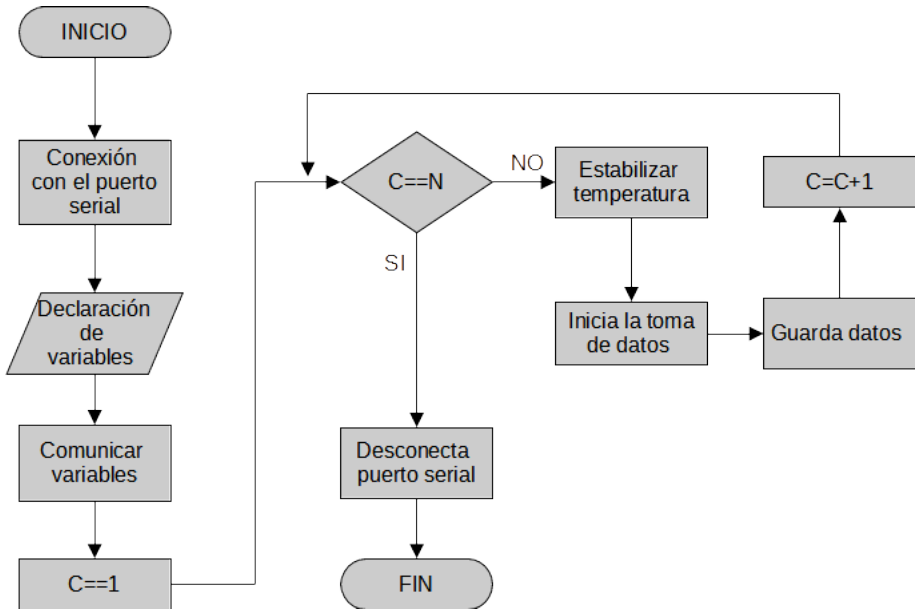


Figura 4.4: Diagrama de flujo del programa para GUI.

1.- Proceso `Conexión con el puerto serial`: establece los parámetros para la comunicación con el micro-controlador a través del puerto serie.

2.- Entrada de datos `Declaración de variables`: recibe del usuario los parámetros de medición.

3.- Proceso `Comunicar variables`: escribe en el puerto serie los parámetros para la medición, acompañados de caracteres que los identifican, figura 4.5a.

4.- Proceso `C==1`: inicia un contador `C` en `1`, es el contador de número de muestras.

5.- Proceso `Estabilizar temperatura`: escribe en el puerto serie `e`, `0` y/o `1` para estabilizar el sistema a las condiciones iniciales, figura 4.5b.

6.- Proceso `Inicia la toma de datos`: sincroniza la muestra de audio con el proceso `Comienza medición` del programa para el micro-controlador, 4.5c.

7.- Proceso `Guardar datos`: escribe `g` en el puerto serie, lee los datos de temperatura y organiza los datos de temperatura y audio en un archivo, figura 4.5d.

8.- Proceso `Desconecta el puerto serie`: desconecta el puerto serie cuando el usuario lo indique.

El valor `N` representa el número de muestras que se tomarán bajo las mismas condiciones.

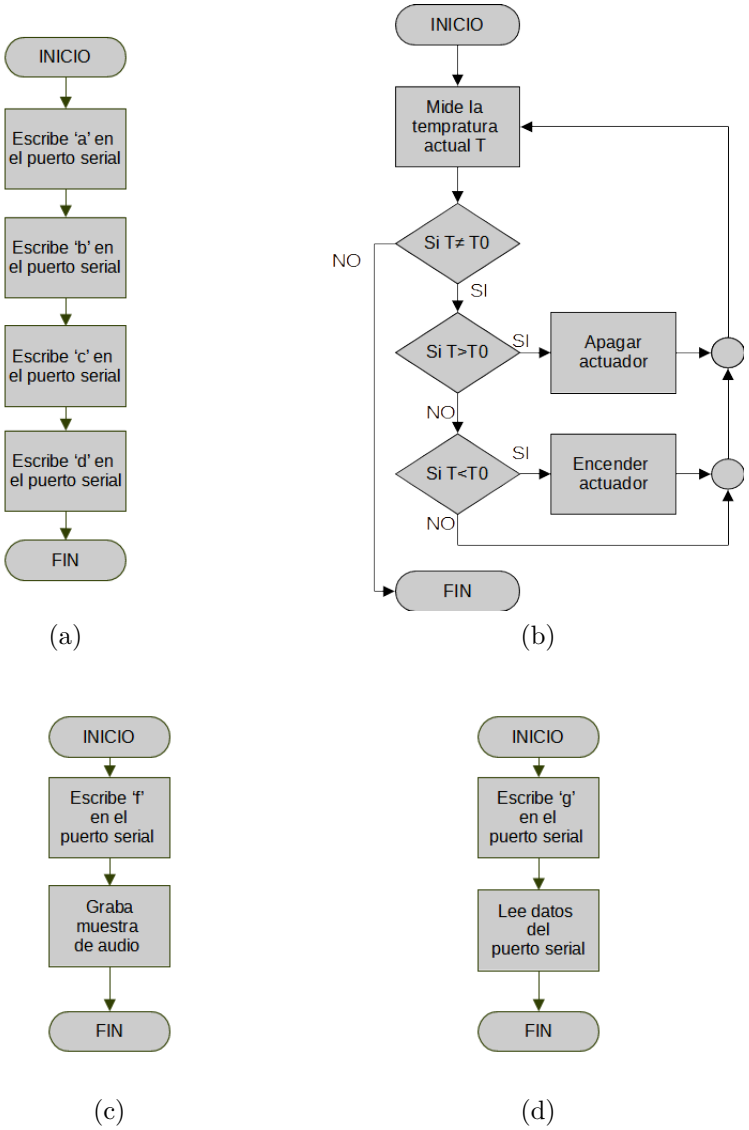


Figura 4.5: Procesos detallados, programa para la GUI. (a) Comunica variables. (b) Estabiliza temperatura. (c) Inicia la toma de datos. (d) Guarda datos.

4.3. Programa para el procesamiento de datos

A diferencia de los diseños de **SmicroControlador** y **SguiMuestreo** el diseño de **Sguiprocesamiento** no se basa en componentes electrónicos elegidos, este se basa solo en las necesidades del usuario. **SguiProcesamiento** se puede dividir en cuatro procesos principales, `Obtención de datos`, `Editar datos`, `Filtrar datos` y `Analizar datos`, incluidos en

el diagrama de flujo de la figura 4.6.

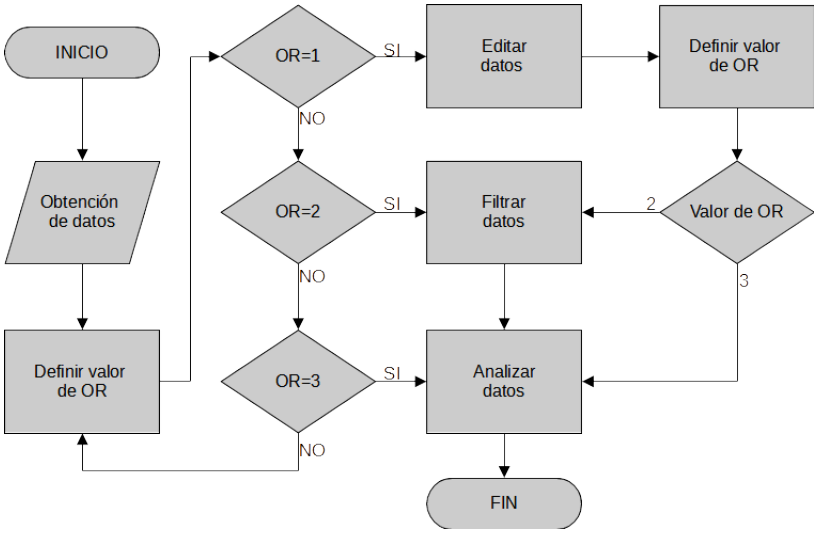


Figura 4.6: Diagrama de flujo para el procesamiento de datos.

1.- Entrada `Obtención de datos`: importan los datos a procesar.

2.- Proceso `Definir valor de OR`: el usuario define una variable `OR`.

3.- Las condicionales determinan el siguiente proceso con base al valor de `OR`.

4.- Si `OR=1` continua en Editar datos, figura 4.7a, importa los datos a editar, el usuario define los parámetros para la edición, se aplican los parámetros y el usuario elige si desea guardar cambios. Si `OR=2` continua en Filtrar datos, figura 4.7b, importa los datos a filtrar, el usuario define los parámetros para el filtro y si desea aplicarlo en datos locales o a todos los datos, al aplicarlos a datos locales, el programa da la opción de aplicar un filtro diferente, en el caso de aplicar el filtro a todos los datos el usuario elige si desea guardar cambios. Si `OR=3` continua en Analizar datos, figura 4.7c, importa los datos a analizar, el usuario elige un tipo de intervalo $[T_i, Tmax]$ o $[t_i, tf]$, (donde T_i es una temperatura inicial, T_{max} temperatura máxima de la muestra, t_i tiempo inicial y t_f tiempo final), el usuario elige el tipo de análisis (PDF, R/S, RMS, H-H y FPS) y por ultimo si se desea guardar los resultados del análisis.

5.- Al concluir el proceso `Editar datos`, el proceso `Definir valor de OR` solicita al usuario si continuar en `Filtrar datos` o en `Analizar datos`. Si se continua en `Filtrar datos` le sigue el proceso `Analizar datos` antes de terminar con el programa.

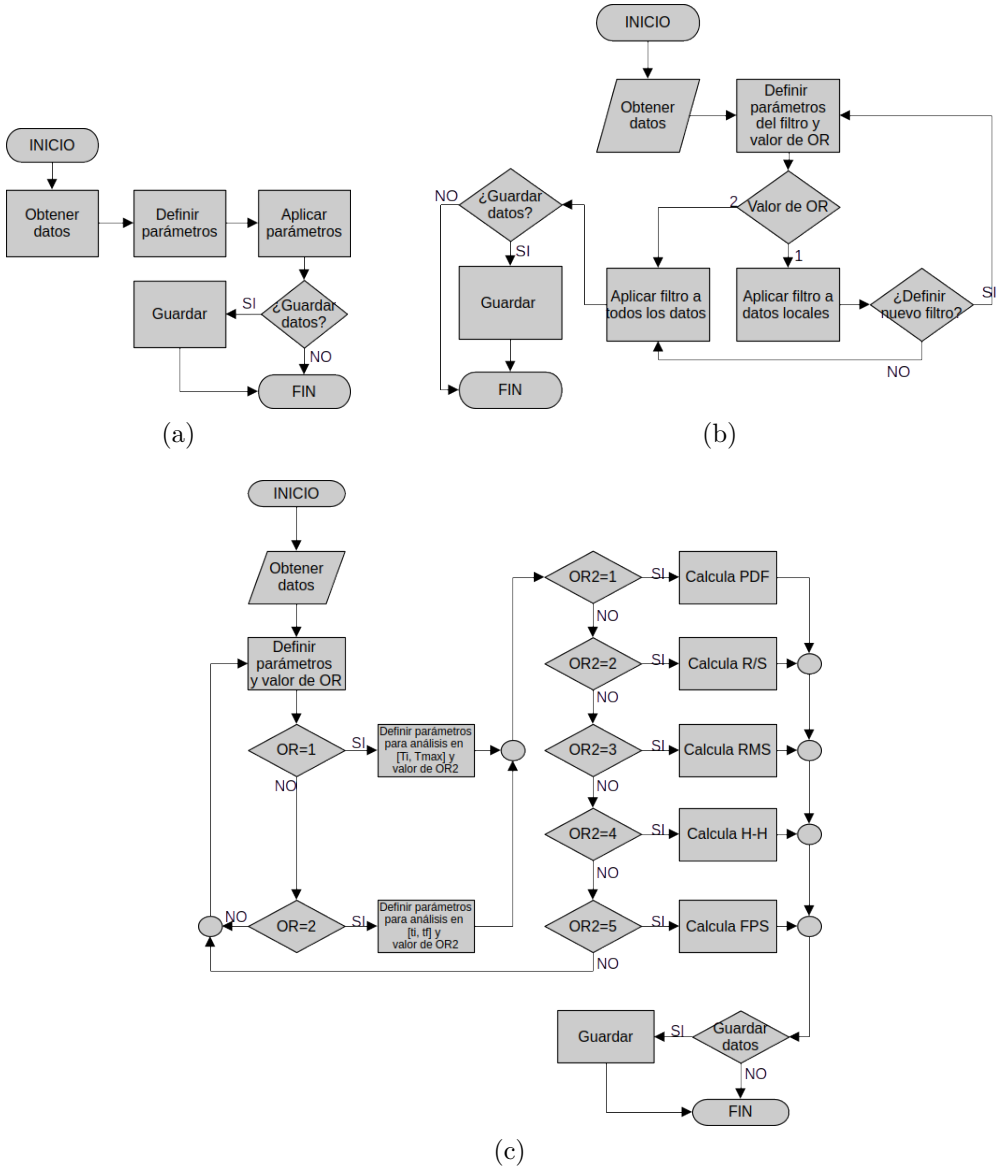


Figura 4.7: Procesos detallados, programa para el procesamiento. (a) Editar datos. (b) Filtrar datos. (c) Analizar datos.

4.4. Codificación del Software

El proceso de codificación se basó en crear funciones para cada proceso mostrado en los diagramas de flujo de las figuras 4.2, programa para el **SmicroControlador**, figura 4.4, programa para **SguiMuestreo** y figura 4.6 para **SguiProcesamiento**.
La elección del lenguaje y software para la codificación se describe a continuación:

- **SmicroControlador**: escrito en lenguaje C utilizando el entorno de desarrollo Arduino IDE. Detalles en Apéndice A.

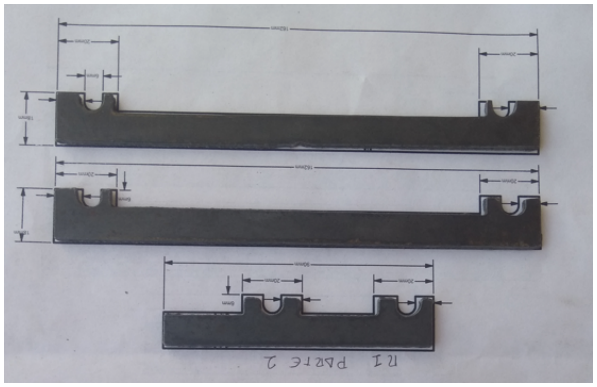
- **SguiMuestreo**: escrito en lenguaje Matlab con el software MATLAB. Detalles en el Apéndice B.
- **SguiProcesamiento**: escrito en lenguaje Matlab con el software MATLAB. Detalles en el Apéndice C.

4.5. Implementación

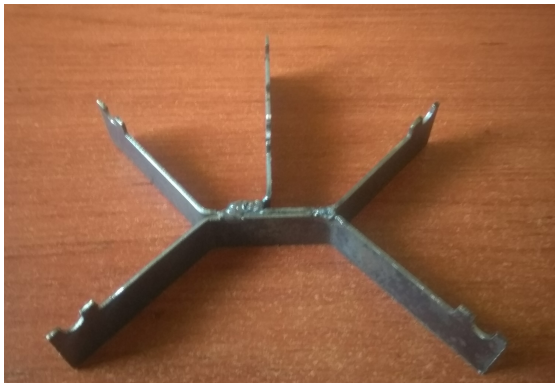
Una vez completada la propuesta procedemos a la construcción de la sección de montaje y el hardware e implementación del software.

4.5.1. Sección de Montaje

Se construyó una caja de cartón que se forró con paneles acústicos, con aislante térmico fibra de vidrio fijado en el centro, figura 4.8c. Para las bases se consideró una lámina negra de calibre 12, cuyas partes se cortaron con una máquina de CNC figura 4.8a, posteriormente fueron dobladas y unidas por soldadura, para que las bases encajaran bien en las resistencias, una vez soldadas, las muescas fueron limadas con una lima circular figura 4.8b.



(a)



(b)

(c)

Figura 4.8: Parte experimental. Proceso de construcción y montaje de elementos. (a) Base cortada en maquina CNC. (b) Base doblada. (c) Componentes montados.

4.5.2. Hardware

El hardware está conformado de los circuitos mostrados en las figuras 3.12 y 3.16. Como primer paso se verificó el funcionamiento de los circuitos en una protoboard, figura 4.11a, para después recrear su placa de circuito impreso (PCB) en Proteus 8 Professional, figura 4.9.

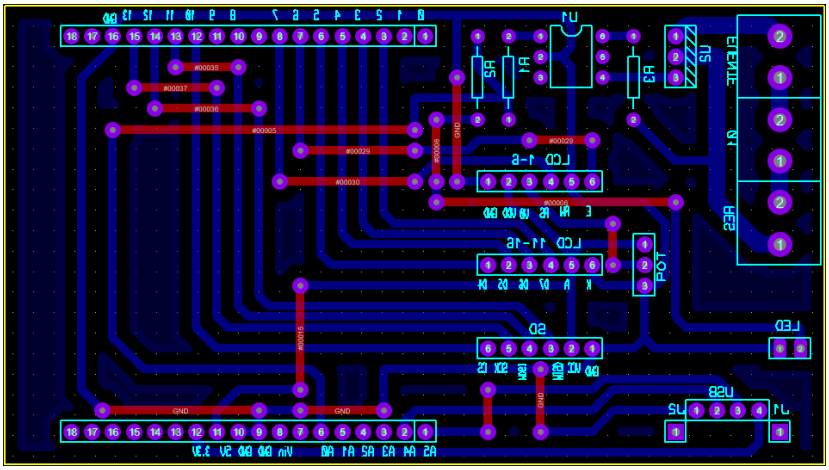


Figura 4.9: Diseño de circuito impreso de la parte de potencia y parte electrónica del proyecto.

El diseño se realizó considerando los siguientes materiales para la construcción del dispositivo:

- Gabinete de plástico con tapa GP-11.
- Contacto para chasis AMB-2P.
- Cable toma corriente calibre 18.
- Conector tipo C7 505-390 hembra.
- Conector tipo C7 macho.
- Cable duplex calibre 18.
- Terminales TRTG-02.
- Terminales hembra tipo Faston TFAB-1/4L.
- Cable modular 4 vías MO4T-305BL.
- Tira de pines hembra.
- Cables tipo dupont.
- Cable de conexiones calibre 22.
- Plug USB para soldar.
- Jack USB para soldar sin cubierta.

Componentes como las resistencias, el TRIAC y el optoacoplador estarán conectados directamente a la placa. Otros, como el Arduino UNO, el LCD, el adaptador microSD y el LED, se conectarán mediante tiras de pines. El sensor de temperatura IR tendrá una conexión adaptada tipo USB, mientras que las salidas a la fuente, el interruptor y actuador (resistencia eléctrica) estarán conectados a través de las terminales TRTG-02. En la figura 4.10 se muestra el modelo 3D de la PCB, proporcionado por Proteus 8 Professional.

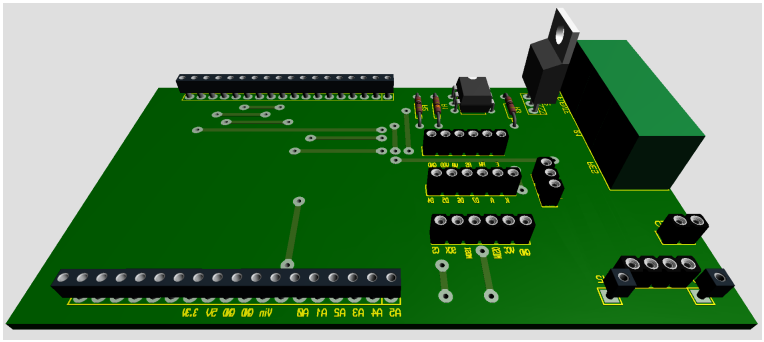
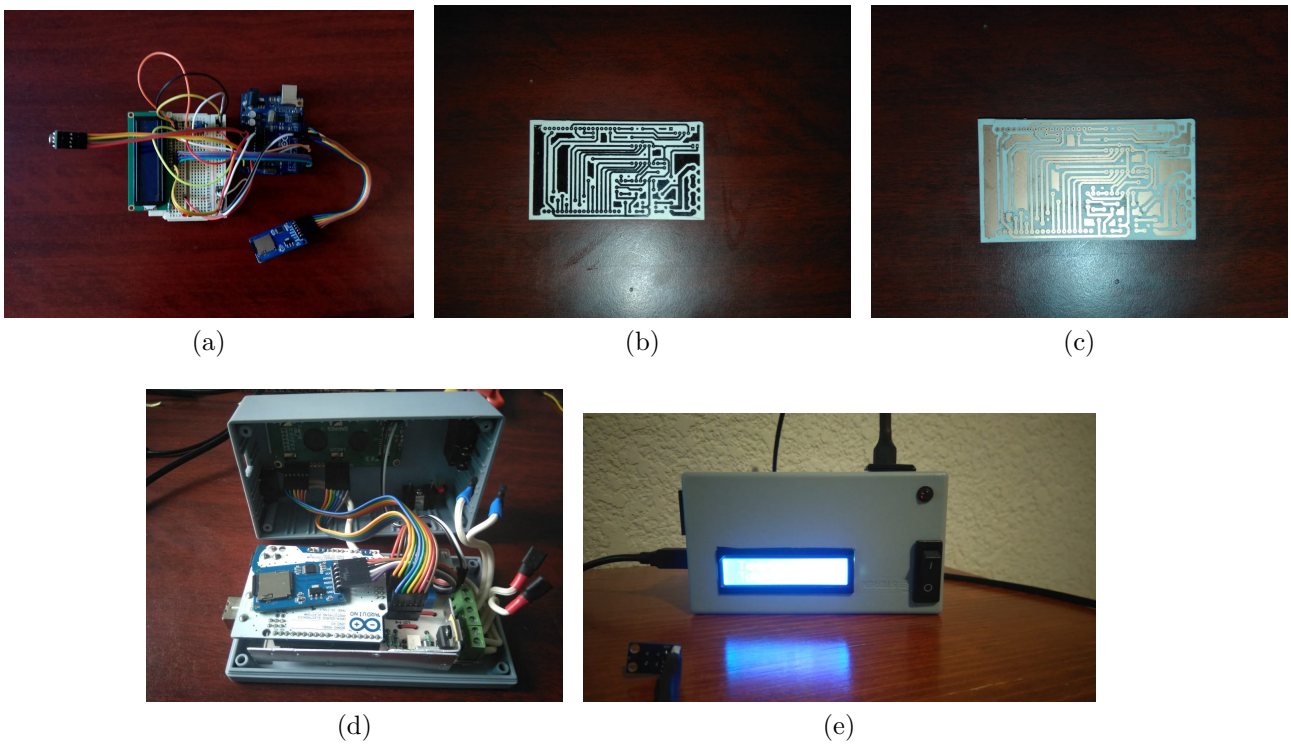


Figura 4.10: Modelo 3D del circuito impreso de la figura 4.9.

Definido la distribución y materiales necesarios en una placa fenólica cubierta de cobre de $10 \times 7cm$ se grabó el diseño de la PCB por el método del planchado, figura 4.11b, a continuación se hizo el perforado y soldado de componentes, figura 4.11c. Por seguridad y comodidad al manipular el dispositivo, este se montó dentro del gabinete de plástico GP-11, figura 4.11d, el dispositivo terminado se muestra en la figura 4.11e.



(a) (b) (c)

(d) (e)

Figura 4.11: Proceso de construcción del módulo de control de temperatura. (a) Pruebas en protoboard. (b) Impresión del circuito impreso. (c) Componentes soldados. (d) Montaje dentro del gabinete de plástico. (e) Dispositivo terminado.

4.5.3. Interfaz gráfica

La interfaz de usuario (GUI) fue desarrollada utilizando la herramienta de Matlab GUIDE. Esta interfaz consta de seis ventanas principales:

- Ventana de inicio, figura 4.12: permite al usuario seleccionar la acción a realizar, ofreciendo las dos opciones, medir y analizar.
- Ventana de muestreo, figura 4.13: en esta ventana, el usuario puede definir los siguientes parámetros:
  - Nombre del puerto al que está conectado el dispositivo.
  - Tiempo de muestra, indicado en minutos, necesario para completar una medición.
  - Número de muestras que se tomarán bajo las mismas condiciones.
  - Temperatura inicial, que marca el punto de inicio de la medición y debe ser superior a la temperatura ambiental, ya que el sistema solo cuenta con mecanismo de calentamiento.
  - Temperatura máxima, que al ser alcanzada detiene el sistema de calentamiento, aunque la captura de audio y la medición de temperatura continúan.

  - Espacios destinados para la visualización de las gráficas de audio y temperatura.

- Ventana para importar datos, figura 4.14: permite al usuario exportar cualquier conjunto de datos de audio y temperatura que cumpla con el formato:

  - Archivo tipo .txt.
  - Las columnas están separadas por comas o espacios. Para los datos de audio, la primera columna corresponde al tiempo, y las siguientes columnas representan los datos de intensidad. En cuanto a los datos de temperatura, la primera columna es el tiempo, la segunda corresponde a la temperatura del objeto, seguida de la temperatura ambiente. Posteriormente, se incluyen la temperatura del objeto y la temperatura ambiente para las demás muestras.

- Ventana de edición, figura 4.15: permite al usuario descartar muestras o segmentos de muestras, ya sea con respecto al tiempo o de la temperatura.

- Ventana para filtrar audio, figura 4.16: permite al usuario aplicar filtros digitales, mostrando la representación gráfica antes y después de aplicar el filtro, junto con sus respectivas transformadas de Fourier.

- Ventana para aplicar análisis multifractal: el usuario selecciona las muestras y el tipo de análisis. Los resultados se presentan en gráficas y tablas. En las figuras 4.17, 4.18, 4.19 y 4.20 se muestran los análisis R/S, RMS, H-H y FPS aplicados a un conjunto de muestras obtenidos con el dispositivo desarrollado en este trabajo.

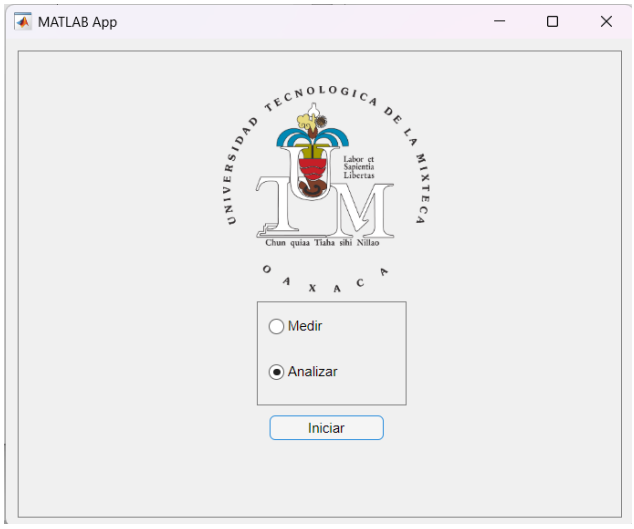


Figura 4.12: Ventana de inicio.

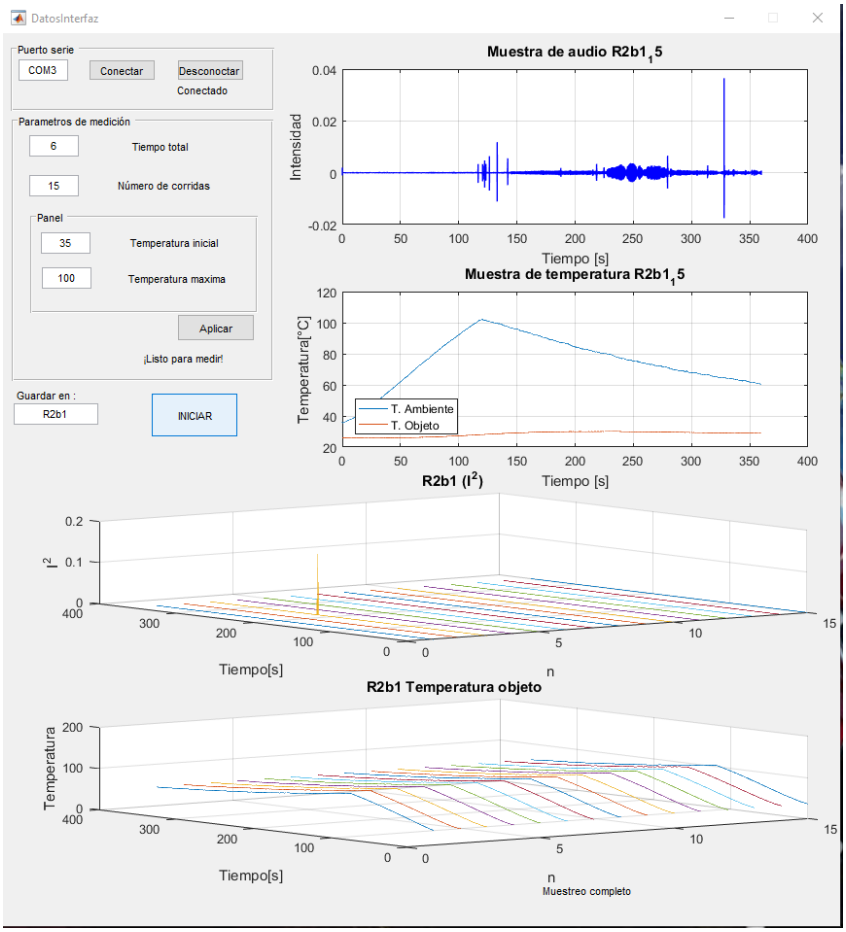


Figura 4.13: GUI muestreo.

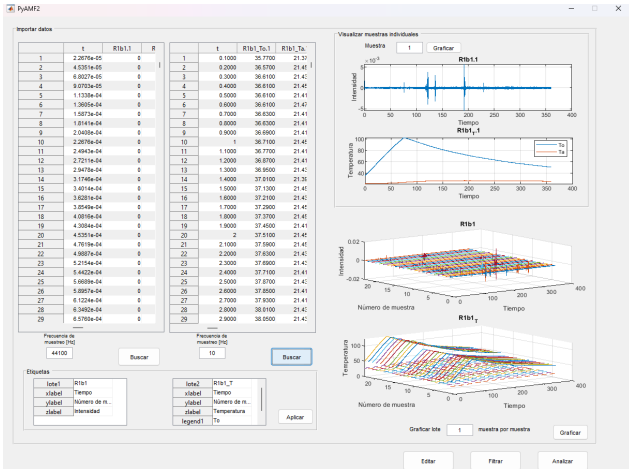


Figura 4.14: Interfaz de usuario, exportar datos para el análisis.

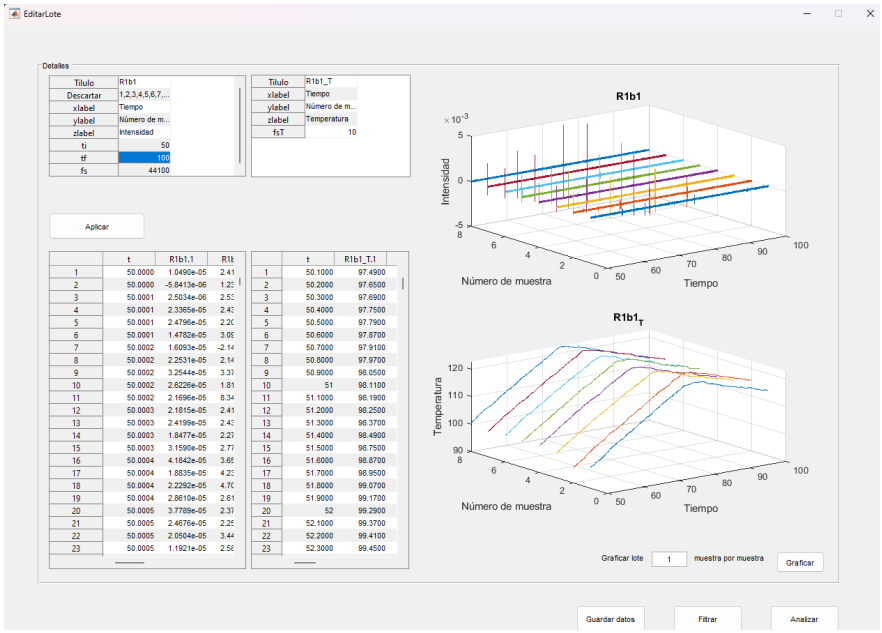


Figura 4.15: Interfaz de usuario, editar datos para el análisis.

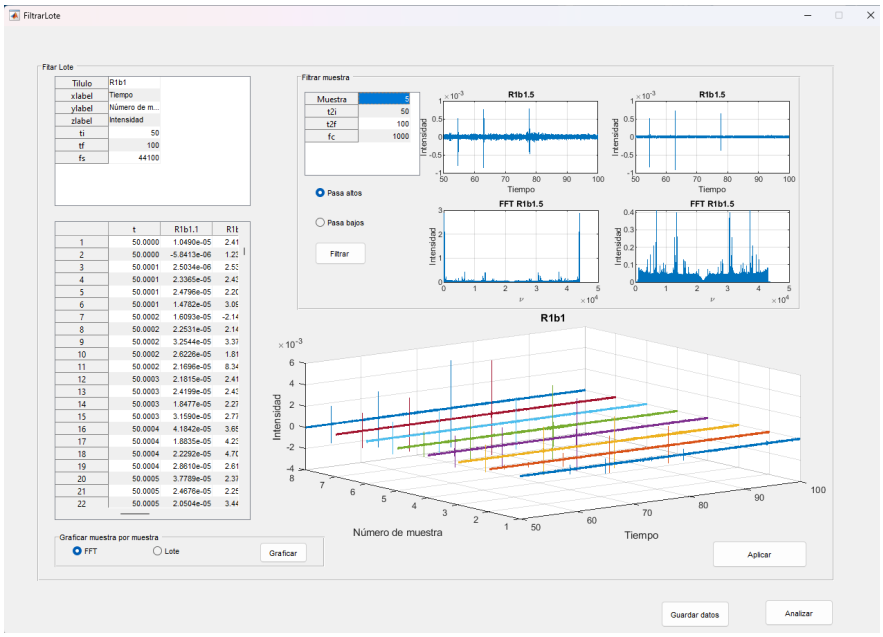


Figura 4.16: Interfaz de usuario, filtrar datos de audio para el análisis.

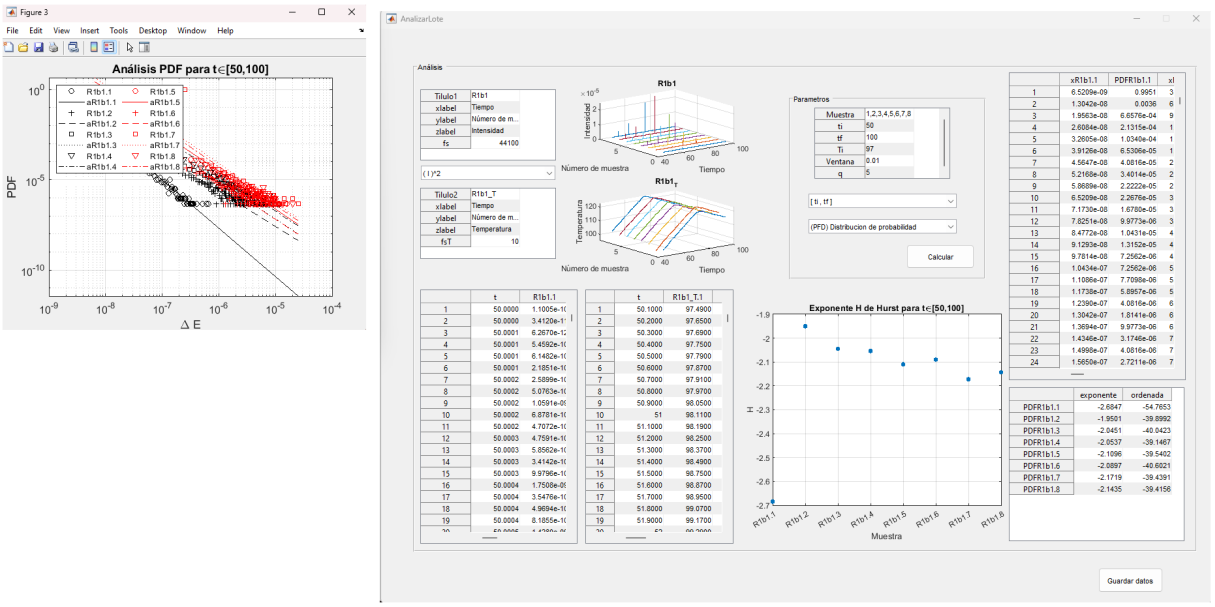


Figura 4.17: Aplicación del análisis R/S.

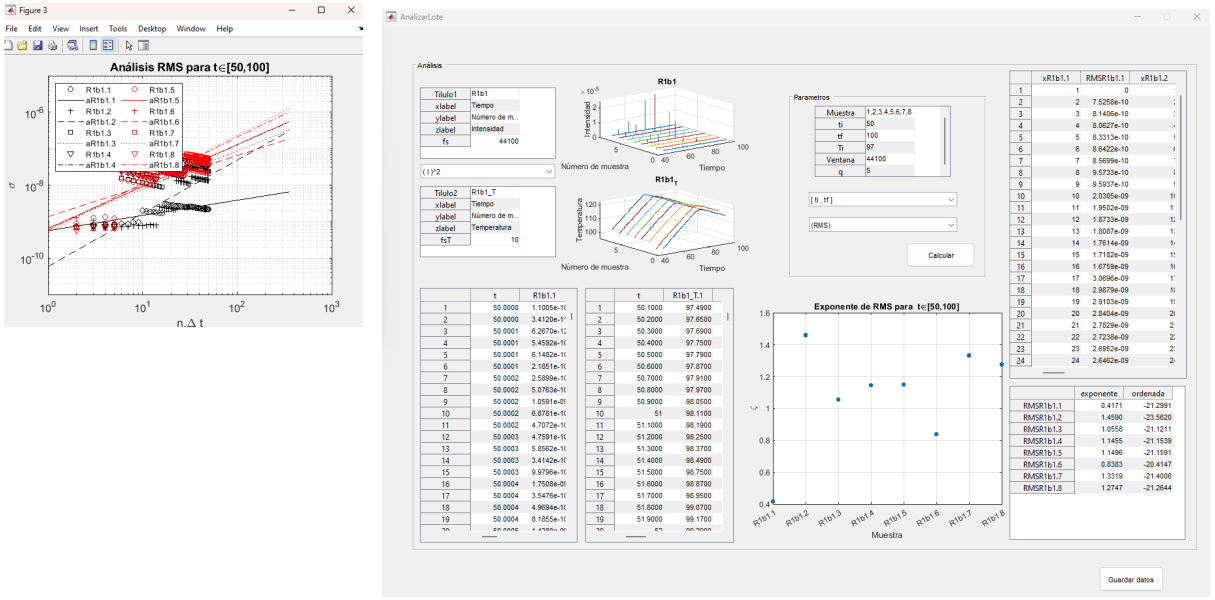


Figura 4.18: Aplicación del análisis RMS.

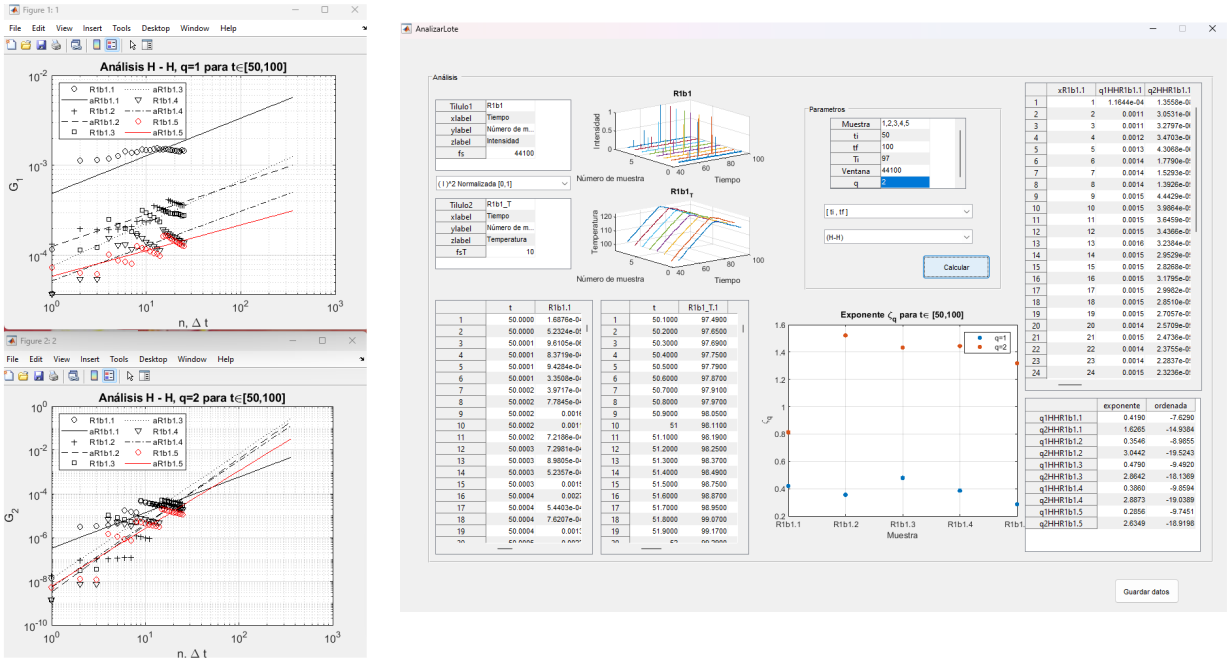


Figura 4.19: Aplicación del análisis H-H.

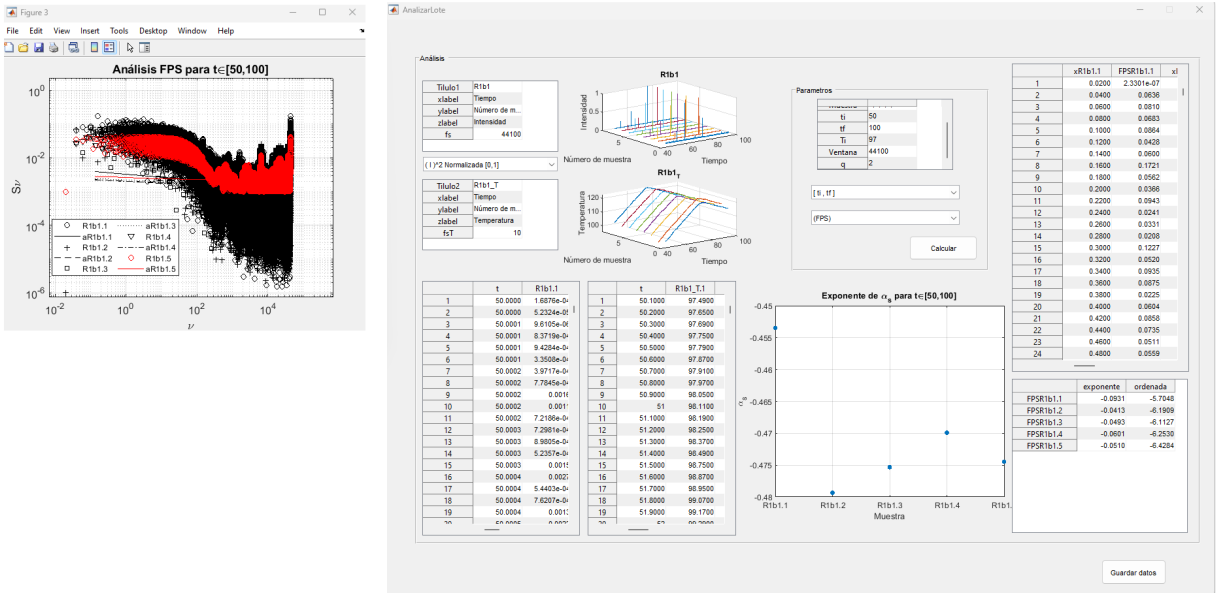


Figura 4.20: Aplicación del análisis FPS.

Capítulo 5

Resultados y Discusiones

Esta tesis presenta el desarrollo y documentación experimental de un sistema para el censo de datos termo-acústicos, complementado por un software diseñado específicamente para la recolección y análisis de dichos datos, enfocados en los fenómenos de expansión térmica de metales. Este sistema integra hardware y software en un dispositivo de propósito específico.

El diseño experimental y su implementación se enfocaron en la caracterización de los chasquidos acústicos generados durante el proceso de expansión térmica en los metales. A continuación, se describen en detalle los elementos destacados del proceso.

Diseño e Implementación del Hardware

- Los siguientes sensores se emplearon para la recolección de los datos:
  - Micrófono MOV-034: De alta sensibilidad con una respuesta en frecuencia de 20 Hz a 20 kHz, conectado directamente al sistema de adquisición de datos.
  - Sensor infrarrojo MLX90614: Sensor de temperatura capaz de medir tanto la temperatura del entorno como la de los metales en análisis, con una precisión de $\pm 0.2°C$.
- El control y procesamiento de señales fueron gestionados por un microcontrolador Arduino UNO, que además controla los actuadores y realiza la adquisición sincronizada de datos. Este microcontrolador fue complementado por:
  - Circuito de potencia basado en optoacopladores MOC3010 y TRIACs BTA08-800BW, asegurando el aislamiento entre la parte de control y los actuadores.
- **Sección de montaje:** Se construyó una cámara experimental utilizando paneles acústicos (AGPTEK) y material aislante de fibra de vidrio con recubrimiento de aluminio, garantizando un entorno libre de ecos y protegido contra interferencias térmicas. Este montaje se diseñó específicamente para optimizar la captura de datos acústicos y de temperatura.
- Microcontrolador Arduino UNO, con el que se gestionan las señales de los sensores y se controlan los actuadores del sistema.

- El microcontrolador Arduino UNO se complementa con un circuito de potencia basado en los optoacopladores MOC3010 y TRIACs.

Desarrollo del Software El software, dividido en tres módulos principales, desempeñó un papel crucial en el control, adquisición, y procesamiento de los datos experimentales:

Software para el Arduino UNO

- Gestión de tareas de control y adquisición de datos térmicos.
- Encendido y apagado de actuadores desde la interfaz gráfica.
- Registro continuo de datos de temperatura en una tarjeta microSD.
- Comunicación bidireccional con la GUI a través del puerto serial.

Interfaz Gráfica de Usuario (GUI)

- Permite al operador establecer parámetros experimentales como intervalos de temperatura y tiempo.
- Sincronización precisa entre los sensores de audio y temperatura, asegurando que las mediciones sean consistentes y fiables.
- Organización y almacenamiento automático de los datos en un formato estructurado para su análisis posterior.

Software de procesamiento y análisis

- Filtrado y edición de datos, permitiendo al usuario ajustar y limpiar las señales recolectadas.
- Implementación de los algoritmos que calculan los parámetros multifractales tales como el espectro de potencias de Fourier, el análisis R/S y correlación altura-altura.
- Generación de gráficos y reportes preliminares que buscan facilitar la interpretación de resultados.

El diseño y la implementación del sistema experimental para la recolección y análisis de datos termo-acústicos han representado un avance significativo en la caracterización de los fenómenos asociados a la expansión térmica en metales. Uno de los principales logros de este trabajo ha sido la integración efectiva del hardware y software, permitiendo obtener datos experimentales, su análisis y la visualización preliminar de los mismos. Los sensores utilizados, como el micrófono MOV-034 y el sensor infrarrojo MLX90614, demostraron ser adecuados para capturar tanto las señales acústicas como las mediciones de temperatura. Además, aunque el rango de frecuencia del micrófono permitió registrar con precisión los chasquidos acústicos, es necesario que el dispositivo se emplee en el entorno adecuado, es

decir, sin ruido, de no ser así será recomendable implementar sistemas de filtrado adicionales o posiblemente considerar el uso de micrófonos con mayor sensibilidad.

El uso del microcontrolador Arduino UNO fue determinante para gestionar las señales de los sensores y controlar los actuadores del sistema. Este dispositivo demostró ser eficiente para las necesidades del experimento, sin embargo, para experimentos más complejos o que requieran mayores tasas de muestreo, sería pertinente considerar microcontroladores más avanzados, como el Arduino Due o el ESP32, que ofrecen mayor capacidad de procesamiento. Asimismo, la dependencia de una tarjeta microSD para el almacenamiento de datos podría ser un factor limitante en experimentos prolongados, por lo que integrar opciones de almacenamiento en la nube sería una mejora significativa para facilitar la gestión de grandes volúmenes de información.

En cuanto al software desarrollado, este se destacó como una herramienta clave para el procesamiento y análisis de datos recolectados. La implementación de algoritmos multifractales, como el análisis R/S y el espectro de potencias de Fourier, permitió una descripción de las señales termo-acústicas. Sin embargo, la inclusión de métodos más avanzados, como la transformada wavelet, podría enriquecer aún más el análisis multifractal y proporcionar una mayor profundidad en la interpretación de los datos. Además, aunque la interfaz gráfica de usuario fue funcional y permitió un control adecuado del experimento, su diseño podría beneficiarse de mejoras que la hagan más intuitiva y adaptable a las necesidades del usuario, incluyendo opciones para personalizar el análisis en tiempo real.

A pesar de los resultados satisfactorios obtenidos, el sistema experimental aquí presentado, tiene algunas limitaciones que podrían abordarse en trabajos futuros. Por ejemplo, aunque la cámara experimental utilizada provee un entorno controlado, sería valioso integrar un sistema de monitoreo ambiental que registre parámetros como la humedad y la presión, los cuales también pueden influir en los fenómenos termo-acústicos. Además, el diseño actual está desarrollado para un conjunto específico de metales, por lo que una futura ampliación podría considerar el análisis de una gama más amplia de materiales, como medios porosos, por ejemplo.

En términos generales, el dispositivo de propósito específico desarrollado en este trabajo proporcionan un punto de partida importante para el estudio del fenómeno de crackling presente en muchos sistemas físicos, particularmente en fenómenos termo-acústicos en metales. Los algoritmos multifractales implementados se destacan como herramientas robustas para caracterizar la complejidad de los patrones observados, ofreciendo nuevas perspectivas sobre la interacción entre las propiedades térmicas y acústicas de los materiales. Estos resultados tienen el potencial de ser aplicados en diversas áreas industriales y científicas, como el diseño de materiales resistentes a choques térmicos y la monitorización estructural mediante técnicas acústicas. A pesar de las limitaciones identificadas, este trabajo representa un paso importante en la dirección de un análisis más completo y sofisticado de los fenómenos termo-acústicos, sentando las bases para futuras investigaciones y desarrollos tecnológicos.

Conclusiones

En el presente trabajo de tesis se desarrollaron tanto el diseño experimental como la programación de la interfaz de usuario y los algoritmos necesarios para el análisis multifractal, cabe señalarse que en sí conforman un proyecto de propósito específico para el usuario. Este diseño e implementación permiten un primer paso en la automatización en la adquisición de datos termo-acústicos derivados del experimento y su posterior análisis multifractal, de este modo se optimizan los procesos y se reduce la intervención manual de modo significativo.

El sistema desarrollado tiene la ventaja de ser versátil y flexible, ya que se espera que su diseño se pueda extender a una amplia variedad de fenómenos físicos que requieran caracterización acústica a través del análisis multifractal. Además, su arquitectura admite ajustes y mejoras, lo que lo convierte en una herramienta adaptable para abordar nuevas necesidades o escenarios experimentales.

El diseño y la implementación aquí presentados no solo proporcionan una herramienta práctica para el estudio de fenómenos complejos basados en el análisis acústico, sino que también establecen una base sólida para futuras investigaciones en este campo. El sistema no es muy robusto pero puede actualizar su capacidad con nuevas adaptaciones a desarrollos más sofisticadas en contextos experimentales.

Apéndice A

Apéndice

A. Codificación del programa para el micro-controlador

SmicroControlador está escrito en lenguaje C utilizando el entorno de desarrollo Arduino IDE. El diagrama de flujo consta de ocho procesos, como se muestra en la figura 4.2. El primer proceso, `Preparación del programa`, abarca la inclusión de librerías, la declaración e inicialización de variables, así como la ejecución de funciones necesarias para el correcto funcionamiento del hardware. Dado que estas acciones se ejecutan una sola vez, el código correspondiente se escribe tanto antes como dentro de la función **`setup`**.

Variables	Descripción
a	Contador de tiempo para la actualización de temperatura en LCD durante el proceso de muestreo
v	Contador de tiempo para encendido y apagado del actuador al iniciar el proceso de muestreo
x	Bandera que señala si el proceso de calentamiento continua o ha terminado
b	Contador de tiempo para la actualización de temperatura en el LCD cuando no hay datos para leer en el puerto serial
c	Bandera, señala el inicio del muestreo
ordenRecibida	Selecciona el proceso siguiente
T0	Temperatura inicial
Tmax	Temperatura máxima
tm	Tiempo de muestreo
Ntiempo	Número de muestras
T	Temperatura actual del sensor de temperatura
salida	Número del pin conectado al circuito de potencia

Cuadro A.1: Variables para la codificación de **SmicroControlador**, figura 4.2.

Las variables utilizadas en el código se describen en la tabla A.1. La estructura principal del programa es un ciclo infinito acompañado de una condicional, el fragmento de código A.1

lo representa con la función **loop** que emula un bucle infinito que se repetirá mientras la tarjeta Arduino se le suministre energía, la condición `if(Serial.available()>0)` verifica si hay datos para leer en el puerto serie, si no los hay, el programa pasará a la línea 6 del código, `if(b==1000)`, donde se asegura que el proceso `Mide la temperatura actual`, figura 4.2, se ejecute cada $1000ms$, `term.read()` lee la temperatura que marca el sensor y las funciones `lcd.setCursor, lcd.print` escriben la temperatura en el LCD, al final `delay(50)` define la frecuencia con la que se repetirá `loop`, en este caso una frecuencia de $20Hz$, `b=b+50` registra el tiempo que ha pasado el puerto serie sin caracteres para lectura. Para los casos en los que existen datos en el puerto serie la línea 3 lee y lo guarda el dato en `ordenRecibida`, los siguientes procesos dependerán del valor de `ordenRecibida`.

```
1  void loop() {
2    if (Serial.available() > 0) {
3      ordenRecibida=Serial.read();
4      //PROCESOS...
5      }
6    if(b==1000){
7      therm.read();
8      lcd.setCursor(1,0);lcd.print(''Tamb='');lcd.print(therm.ambient());lcd
       .print(''C'');
9      lcd.setCursor(1,1);lcd.print(''Tobj='');lcd.print(therm.object());lcd.
       print(''C'');
10     b=0;}
11   delay(50); b=b+50;
12 }
```

Código A.1: Codificación de la estructura principal del programa y el proceso `Mide la temperatura actual`, figura 4.2.

Si la variable `ordenRecibida` guarda el carácter `'1'` se ejecuta el código A.2, la función `Serial.ptintln` escribe `1` en el puerto serie y `digitalWhite` enciende el circuito de potencia, figura 3.12. Si la variable `ordenRecibida` guarda el carácter `'0'` se ejecuta el código A.3, escribe `1` en el puerto serie y apaga el circuito de potencia, esto corresponde a los procesos `Enciende actuador`, figura 4.3a y `Apaga el actuador`, figura 4.3b del diagrama de flujo.

```
1    if(ordenRecibida=='1'){
2       Serial.println(1);
3       digitalWrite(salida,HIGH);  }
```

Código A.2: Codificación del proceso `Enciende el actuador` 4.3a.

```
1    if(ordenRecibida=='0'){
2       Serial.println(1);
3       digitalWrite(salida,LOW); }
```

Código A.3: Codificación del proceso `Apaga actuador` figura 4.3b.

El código A.4 corresponde al proceso `Configura parámetros de medición`, figura 4.3c. Se ejecuta cuando la variable `ordenRecibida` guarde los caracteres `'a'`, `'b'`, `'c'`, o `'d'`, en cada caso la función `Serial.readString` lee datos del puerto serie y el programa asigna estos datos a las variables `T0`, `Tmax`, `Ntiempo` o `tm` según corresponda, una vez asignado el valor el programa escribe este en el puerto serie.

```
1     if(ordenRecibida=='a'){
2       T0 = Serial.readString().toInt(); Serial.println(T0);}
3     if(ordenRecibida=='b'){
4       Tmax = Serial.readString().toInt(); Serial.println(Tmax);}
5     if(ordenRecibida=='c'){
6       Ntiempo = Serial.readString().toInt(); Serial.println(Ntiempo);}
7     if(ordenRecibida=='d'){
8       tm = Serial.readString().toInt(); Serial.println(tm);}
```

Código A.4: Codificación del proceso `Configura parámetros de medición`, figura 4.3c.

Cuando la `ordenRecibida` es `'e'` se ejecuta el código A.5 que codifica el proceso `Temperatura actual`, figura 4.3d. La función `Therm.read()` lee la temperatura que registra el sensor y el programa asigna el valor de temperatura objeto a la variable `T`, después se escribe este valor en el puerto serie y LCD.

```
1     if(ordenRecibida=='e'){
2       therm.read(); T=therm.object();
3       Serial.println(T);
4       lcd.setCursor(1,0);lcd.print(''Tamb='');lcd.print(therm.ambient());
    lcd.print(''C'');
5       lcd.setCursor(1,1);lcd.print(''Tobj='');lcd.print(therm.object());
    lcd.print(''C'');}
```

Código A.5: Codificación del proceso `Temperatura actual`, figura 4.3d.

El código A.6 codifica el proceso `Comienza la medición`, figura 4.3e, se ejecuta cuando la variable `ordenRecibida` es igual a `'f'`. Primero se prepara lo necesario para la muestra, es decir,se inicializan variables y la función `SD.open` crea un archivo de nombre G.txt en la memoria microSD, este archivo guardará los datos de temperatura, si el archivo se abrió correctamente la función `for (int i=0,i<Ntiempo; i++)` inicia un ciclo que se repetirá `Ntiempo` veces. El proceso comienza midiendo temperaturas ambiente y objeto, se le asigna el valor de temperatura objeto a la variable `T`, la función `archivo.println` escribe el valor de `T` en el archivo G.txt, posteriormente se activa el actuador y se indica al puerto serie que el muestreo ha comenzado, `if(T>Tmax)` apaga el actuador cuando se cumpla la condición y cambia el valor de la bandera `x`, para mantener un calentamiento controlado `if (x==1)` se encarga de prender y apagar el actuador cada segundo durante el proceso de calentamiento, la acción `if(a==1000)` actualiza los datos de temperatura al LCD, la función `delay(tm)` define el tiempo de espera entre cada muestra, terminado el ciclo `for` se apaga el actuador y cierra el archivo G.txt.

```
1     if(ordenRecibida=='f'){
2       archivo=SD.open(''G.txt'',FILE_WRITE);
3       c=1; x=1; a=0; v=0;
4       if(archivo){
5         for(int i=0; i<Ntiempo; i++){
6           therm.read();T=therm.object();
7           archivo.println(therm.object());archivo.println(therm.ambient())
    ;
8           if(c==1){
9             digitalWrite(salida,HIGH);
```

```
10            Serial.println(1); c=0;}
11          if(T>Tmax){
12            digitalWrite(salida,LOW);
13            x=0;}
14          //*******************
15          if(x==1){
16            if(v==1000){digitalWrite(salida,LOW);}
17            if(v==2000){digitalWrite(salida,HIGH); v=0;}}
18          //*******************
19          if (a==1000){
20             lcd.setCursor(1,0);lcd.print(''Tamb='');lcd.print(therm.
    ambient());lcd.print(''C'');
21             lcd.setCursor(1,1);lcd.print(''Tobj='');lcd.print(therm.
    object());lcd.print(''C'');
22             a=0;}
23          delay(tm); a=a+tm; v=v+tm;
24        }
25        digitalWrite(salida,LOW);
26        archivo.close();
27      }
28    }
```

Código A.6: Codificación del proceso `Comienza medición' figura 4.3e`.

Por último, cuando `ordenRecibida` es `'g'`, se ejecuta el proceso `Guarda datos de medición`, figura 4.3f. La función `SD.open(G.txt)` abre el archivo G.txt, confirmada la apertura el programa escribe `2` en el puerto serie, `while(archivo.available())` comienza un ciclo que escribe en el puerto serie todos los datos de G.txt, al terminar se cierra y elimina el archivo G.txt.

```
1     if(ordenRecibida=='g'){
2       archivo=SD.open(''G.txt'');
3       if(archivo){
4         Serial.println(2);
5         while(archivo.available()){
6         Serial.write(archivo.read());}
7       archivo.close();
8       SD.remove(''G.txt'');
9       }
10    }
```

Código A.7: Codificación del proceso `Guarda datos de medición`, figura 4.3f.

B. Codificación del programa para la GUI de muestreo

El programa para la GUI de muestreo, **SguiMuestreo**, está escrito en lenguaje Matlab con el software MATLAB R2016a. La codificación se muestra en los códigos A.8 - A.12 y las variables usadas se describen en la tabla A.2.

Variables	Descripción
estado	Cadena de caracteres que indica el proceso en ejecución
PuertoDeComunicacion	Nombre del puerto serie
s	Puerto serie
AT0	Temperatura inicial
ATmax	Temperatura máxima
tiempo	Tiempo total de muestreo en minutos
Fmt	Frecuencia de muestreo de temperatura
dato1, dato2, dato3, dato4, dato5, dato6, dato7	Variables auxiliares
Tobjeto	Arreglo, datos de temperaturas objeto
Tambiente	Arreglo, datos de temperaturas ambiente
FmA	Frecuencia de muestreo del audio
n	Número de datos de audio
t	Arreglo, datos de tiempo audio
recObj	Objeto de audio
x	Arreglo, datos de tiempo temperatura
temperatura	Temperatura actual del sensor de temperatura
audio	Arreglo, datos de audio

Cuadro A.2: Variables para la codificación del programa para la GUI figura 4.4.

El programa comienza asignando valores a las variables `estado` y `PuertoDeComunicacion`, la función `delate` se asegura que puerto seleccionado este disponible para funcionamiento, se define `s` como un puerto serie con características compatibles con Arduino UNO, `fopen` habilita el puerto serie y se actualiza la variable `estado`, código A.8.

```
estado='conectando con el puerto serie'
PuertoDeComunicacion='COM4';
delete(instrfind({'Port'},{PuertoDeComunicacion}));
s = serial(PuertoDeComunicacion,'BaudRate', 9600,'Terminator','CR/LF');
warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
fopen(s);
estado='conectado con puerto serie '
```

Código A.8: Codificación del proceso `Conexión con el puerto serie`, figura 4.4.

Definidas las condiciones de muestreo el código A.9 actualiza la variable `estado`, inicia un ciclo `while` que termina cuando el valor de `AT0` sea igual a `dato1`, escribe en el puerto serie el carácter `a` y lee del puerto serie un entero que guarda en `dato1`, el ciclo se detiene cuanto el valor de **SguiMuestreo** y el comunicado a **SmicroControlador** sean iguales, este proceso se repite para `ATmax`, `ANtiempo` y `AtmT` escribiendo en el puerto serial `'a'` `'b'`, `'c'` y `'d'` respectivamente, por último se actualiza la variable `estado`.

```
estado='Mandando parametros para la muestra de temperado'
while(str2double(AT0)~=dato1)
    fwrite(s,'a');
    dato1=fscanf(s,'%d')
end
```

```
6   while(str2double(ATmax)~=dato2)
7       fwrite(s,'b');
8       dato2=fscanf(s,'%d')
9   end
10  while(str2double(ANtiempo)~=dato3)
11      fwrite(s,'c');
12      dato3=fscanf(s,'%d')
13  end
14  while(str2double(AtmT)~=dato4)
15      fwrite(s,'d');
16      dato4=fscanf(s,'%d')
17  end
18  estado='LISTO PARA TOMAR MUESTRAS'
```

Código A.9: Codificación del proceso `Comunicar variables`, figura 4.5a.

Para emular el ciclo mostrado en la figura 4.4 se puede hacer uso de un ciclo `for` que se repetirá N veces, es decir, `for 1:1:N`. Los códigos A.10-A.12 se encuentran dentro de este ciclo. Código A.10 que corresponde al proceso `Estabiliza temperatura` estabiliza el sistema a la temperatura deseada por el usuario, comienza actualizando la variable `estado`, escribe `'e'` en el puerto serie, esto solicita a **SmicroControlador** la temperatura actual del sistema, la temperatura es leída del puerto serie y guardada en la variable `temperatura`, comienza un ciclo `while` que termina cuando `temperatura` sea igual a `AT0`, dentro del ciclo `while` la primera condicional, `if(temperatura<str2double(AT0))`, se hace efectiva cuando sea necesario calentar el sistema, para eso, se escribe *'1'* en el puerto serie `while(1 =dato5)` se detendrá hasta que **SmicroControlador** active el circuito de potencia, la siguiente condicional, `if(temperatura>str2double(AT0))`, es efectiva cuando el sistema necesite enfriarse, entonces, se escribe `'0'` en el puerto serie y **SmicroControlador** apaga el circuito de potencia, por último, `fwrite(s,'0'); dato5=fscanf(s,'%d')` escribiendo `'e'` en el puerto serie actualizando el valor de `temperatura` cada que se repite el primer `while`, la función `pause(0.5)` define el periodo de repetición de este, un periodo de $2Hz$. El proceso termina actualizando la variable `estado`.

```
1   estado='preparando sistema'
2   fwrite(s,'e'); temperatura=fscanf(s,'%d')
3   while(temperatura~=str2double(AT0))
4       if(temperatura<str2double(AT0))
5           while(1~=dato5)
6               fwrite(s,'1'); dato5=fscanf(s,'%d');
7           end
8           dato5=0;
9       end
10      if(temperatura>str2double(AT0))
11          while(1~=dato5)
12              fwrite(s,'0'); dato5=fscanf(s,'%d');
13          end
14          dato5=0;
15      end
16      fwrite(s,'e'); temperatura=fscanf(s,'%d')
17      pause(0.5);
18  end
```

```
19  while(1~=dato5)
20      fwrite(s,'0'); dato5=fscanf(s,'%d')
21  end
22  dato5=0;
23  estado='sistema listo'
```

Código A.10: Codificación del proceso `Estabiliza temperatura`, figura 4.5b.

Una vez que el sistema esté a temperatura `ATO` se inicia el muestreo, código A.11. Para iniciar el muestreo se escribe `'f'` en el puerto serie dentro en un ciclo `while` indicando a **SmicroControlador** para sincronizar las muestas de audio con las de temperatura, se actualiza la variable estado `estado` la función `tic` registra la hora de inicio del muestreo, `recordblocking` comienza a tomar la muestra de audio con las características de `recObj` por un tiempo `tiempo`, una vez terminada la muestra la función `getaudiodata` convierte la información de audio en un vector tipo float y por ultimo la función `toc` imprime el tiempo que pasó desde que se activo `tic`.

```
1   while(1~=dato6)
2       fwrite(s,'f');
3       dato6=fscanf(s,'%d')
4   end
5   dato6=0;
6   estado='Tomando muestras'
7   tic
8   recordblocking(recObj,tiempo);
9   audio=getaudiodata(recObj);
10  toc
```

Código A.11: Codificación del proceso `Inicia la muestra de datos`, figura 4.5c.

Terminado el muestreo se importan los datos de temperatura de **SmicroControlador**, código A.12. Escribe en el puerto serie `'g'`, se actualiza la variable `estado`, `for i=1:str2double (ANtiempo)` se repite hasta guardar los valores de temperatura ambiente y objeto en los vectores `Tobjeto` y `Tambiente`, se actualiza la variable `estado`. Una vez concluidas la `N` muestras `fclose` desconecta el puerto serie para finalizar el programa, código A.13.

```
1   while(2~=dato7)
2       fwrite(s,'g');
3       dato7=fscanf(s,'%d')
4   end
5   dato7=0;
6   estado='Recolectando datos de Temperatura'
7
8   for i=1:str2double(ANtiempo)
9       Tobjeto(i,1)=fscanf(s,'%f');
10      Tambiente(i,1)=fscanf(s,'%f');
11  end
12  estado='Listo!!'
```

Código A.12: Codificación del proceso `Guarda datos`, figura 4.5d.

```
1   estado='Desconectando puerto serie'
2   fclose(s);
```

```
3   delete(instrfind({'Port'},{PuertoDeComunicacion}));
4   estado='Puerto serie desconectado'
```

Código A.13: Codificación del proceso `Desconectar puerto serie`,figura 4.4.

C. Codificación del programa para el procesamiento de datos

Tal como explica la sección 4.3, **SguiProcesamiento** se divide en cuatro procesos,

para su funcionamiento necesitan las siguientes funciones: importa datos, descartar muestras contaminadas, definir los parámetros para aplicar un filtro digital a una porción de muestra, aplicar filtro a un lote completo, calcular R/S, RMS, H-H y FPS. Para importar datos, código A.14, `uigetfile` abre una ventana de búsqueda que muestra los archivos .txt, la dirección del archivo seleccionar se almacena en `Ruta`, después, se importa a `A` la información del archivo seleccionado

```
1 estado='Importando datos ...';
2 [nombre, direccion]=uigetfile({'*.txt'}, 'Buscar archivo');
3 Ruta=[direccion,nombre];
4 A=importdata(Ruta); A=single(A);
5 [~,n]=size(nombre);file  nombre=nombre(1:n-4);
```

Código A.14: Importar datos.

con la funcion `uigetfile`, con argumentos '*.txt' y 'Buscar archivo' , el primero limita el formato de la búsqueda a archivos de texto y segundo establece el nombre de la ventana de búsqueda, a continuación, se crea la ruta de de archivo en Ruta, `importdata(Ruta)`, importa los datos y los guarda en A, por último se definen los datos de n y nombre.

```
1 % definir Ad y Td por el usuario
2 Ad=single(Ad);
3 Td=single(Td);
4 [nombre,direccion]=uiputfile('.txt');
5 [~,d]=size(nombre); nombre=nombre(1:d-4);
6 writematrix(Ad,[direccion,nombre,'_A','.txt']);
7 writematrix(Td,[direccion,nombre,'_T','.txt']);
```

Código A.15: Guardar datos.

Se definen como variables de tipo single los datos de audio y temperatura, con `uiputfile` se construye el nombre y ruta de datos, con la siguiente linea se conserva el nombre sin extensión, con `writematrix` se guardan los datos de Ad y Td en la ruta y con el nombre mas los sufijos A y T para datos de audio y temperatura respectivamente.

```
1 [~,ny]=size(vd);[~,ny2]=size(datos);
2 tiempo=datos(:,1); audio=datos(:,2:ny2);
3 [nx2,ny2]=size(audio);
4 vf=zeros(nx2,ny2-ny);
5 d=1;f=1;
6 for i=1:ny2
```

```
    if vd(1,d)~=i
        vf(:,f)=audio(:,i); f=f+1;
    end
    if i<vd(1,ny)
    if vd(1,d)==i
        d=d+1;
    end
    end
end
vf=[tiempo,vf];
end
```

Código A.16: Descartar muestras contaminadas.

Se define ny y ny2 como la longitud el número de muestras contaminadas y el número de muestras más uno respectivamente, de la variable datos se definen las variables tiempo y audio, nx2 y ny2 definen las dimensiones de la variable audio, se inicializan en uno los contadores d y f, el ciclo for se ejecuta para i $i\epsilon[1 : ny2]$.
Los filtros empleados son los definidos en Matlab ajustados a la necesidad del usuario código A.17.

```
estado='Filtrando ...';
if aux==1
    %Definir por el usuario caso,caso2,A,fs,fc,N
    ti=floor(min(A(:,1))); nt=ti*fs;
    ti=abs((ti*fs+1)-nt); tf=abs(tf*fs-nt);
    A=A(ti:tf,:); [nx,ny]=size(A);
    B=zeros(nx,ny); B(:,1)=A(:,1);
if caso==1
    B(:,N+1)=highpass(A(:,N+1),fc,fs);
end
if caso2==1
    B(:,N+1)=lowpass(A(:,N+1),fc,fs);
end
end
```

Código A.17: Filtrar muestra.

Bibliografía

[1] James P Sethna, Karin A Dahmen y Christopher R Myers. «Crackling noise». En: *Macmillan Magazines Ltd* 410 (2001), págs. 242-250.

[2] Angel Garcimartin, Alessio Guarino, Ludovic Bellon y Sergio Ciliberto. «Statistical properties of fracture precursors». En: *Physical Review Letters* 79.17 (1997), pág. 3202.

[3] Aghil Abed Zadeh. «Crackling Noise in a Granular Stick-Slip Experiment». Tesis doct. Duke University, 2019.

[4] Paul A Houle y James P Sethna. «Acoustic emission from crumpling paper». En: *Physical Review E* 54.1 (1996), pág. 278.

[5] I Campos-Silva, AS Balankin, AH Sierra, N Lopez-Perrusquia, R Escobar-Galindo y D Morales-Matamoros. «Characterization of rough interfaces obtained by boriding». En: *Applied Surface Science* 255.5 (2008), págs. 2596-2602.

[6] Eric M Kramer y Alexander E Lobkovsky. «Universal power law in the noise from a crumpled elastic sheet». En: *Physical Review E* 53.2 (1996), pág. 1465.

[7] RS Mendes, LC Malacarne, RPB Santos, HV Ribeiro y S Picoli Jr. «Earthquake-like patterns of acoustic emission in crumpled plastic sheets». En: *EPL (Europhysics Letters)* 92.2 (2010), pág. 29001.

[8] Benoit B Mandelbrot. *La geometría fractal de la naturaleza*. Barcelona, España: Tusquets Editores S.A., 1997.

[9] A-L Barabási y Harry Eugene Stanley. *Fractal concepts in surface growth*. Cambridge University Press, 1995.

[10] Paul Meakin. *Fractals, scaling and growth far from equilibrium*. Vol. 5. Cambridge University Press, 1998.

[11] Christoph Bandt, Siegfried Graf y Martina Zähle. *Fractal geometry and stochastics*. Springer, 1995.

[12] Manus J Donahue III. «An introduction to mathematical Chaos theory and fractal geometry». En: *Retrieved September* 20 (1997), pág. 2001.

[13] Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 1990.

[14] Heinz-Otto Peitgen, Hartmut Jürgens y Dietmar Saupe. *Chaos and fractals: new frontiers of science*. Springer Science & Business Media, 2004.

[15] T Gregory Dewey. *Fractals in molecular biophysics*. Oxford University Press, 1997.

[16] Kenneth J Falconer y KJ Falconer. *Techniques in fractal geometry*. Vol. 3. Wiley Chichester (W. Sx.), 1997.

[17] Edgar E Peters. *Fractal market analysis: applying chaos theory to investment and economics*. Vol. 24. John Wiley & Sons, 1994.

[18] Paul Meakin. *Fractals, scaling and growth far from equilibrium*. Vol. 5. Cambridge University Press, 1998.

[19] D Morales, F Castrejón y A Balankin. *Geometría fractal: auto-similaridad*.

[20] Rodrigo Alfredo Valenzuela Melivilu et al. «Multifractalidad en series de tiempo de pm2. 5 sobre Santiago de Chile». En: (2016).

[21] J Brew. *Aloe polyphylla, Recuperado 4 de marzo de 2024*. 2 November 2008. URL: https://commons.wikimedia.org/wiki/File:Spiral_aloe.jpg#mw-jump-to-license.

[22] A. Abrahami. *Cepaea nemoralis, Recuperado 4 de marzo de 2024*. 21 de noviembre de 2008. URL: https://es.wikipedia.org/wiki/Cepaea_nemoralis.

[23] H. Zell. *Hippocampus barbouri, Recuperado 4 de marzo de 2024*. 14 de febrero de 2010. URL: https://es.wikipedia.org/wiki/Hippocampus_barbouri.

[24] NASA/GSFC. *Borrasca, Recuperado 4 de marzo de 2024*. 16 de diciembre de 2009. URL: https://es.wikipedia.org/wiki/Borrasca.

[25] Juan Bibiloni. *Dicksonia antarctica, un helecho del Terciario, Recuperado 4 de marzo de 2024*. 30 de junio de 2012. URL: https://jardin-mundani.blogspot.com/2012/06/dicksonia-antarctica.html.

[26] Peter Woodard. *Dicksoniayoungiae, Farm Cove, Recuperado 4 de marzo de 2024*. 21 de febrero 2010. URL: https://jardin-mundani.blogspot.com/2012/06/dicksonia-antarctica.html.

[27] Remsemillas. *Semillas de Brócoli Romanesco, Recuperado 4 de marzo de 2024*. 2020. URL: https://www.remsemillas.com/product-page/semillas-de-br%C3%B3coli-romanesco.

[28] Freepik/8photo. *Cuál es la parte más nutritiva y saludable del brócoli: el tronco, las flores o las hojas, Recuperado 4 de marzo de 2024*. 28 Febrero 2024. URL: https://www.directoalpaladar.com/ingredientes-y-alimentos/cual-parte-nutritiva-saludable-brocoli-tronco-flores-hojas.

[29] Wilson Bentley. *Studies among the Snow Crystals, Recuperado 4 de marzo de 2024*. 1902. URL: https://es.wikipedia.org/wiki/Wilson_Bentley#/media/Archivo:SnowflakesWilsonBentley.jpg.

[30] Antonio Nicolás Ochaíta. *Melipona: la abeja sin aguijón sagrada de los mayas, Recuperado 4 de marzo de 2024*. 7 Febrero 2021. URL: https://laplazuela.net/index.php/naturaleza/13068-melipona-la-abeja-sin-aguijon-sagrada-de-los-mayas.

[31] JMPerez. *Screenshot taken from NASA World Wind of the Gran Canaria Island., Recuperado 4 de marzo de 2024*. 27 abril 2006. URL: https://es.m.wikipedia.org/wiki/Archivo:Gran_Canaria_NWW.jpg.

[32] Ángel Navarro. *Mosca de la fruta, Recuperado 4 de marzo de 2024*. URL: https://www.flickr.com/photos/160931598@N03/44185145231/.

[33] Jin Li, Xian Zhang, Jingtian Tang, Jin Cai y Xiaoqiong Liu. «Audio magnetotelluric signal-noise identification and separation based on multifractal spectrum and matching pursuit». En: *Fractals* 27.01 (2019), pág. 1940007.

[34] STEREN. *Manual de instrucciones 0817A*. 2021. URL: https://descargas.steren.com.mx/MOV-034-instr.pdf.

[35] STEREN. *Catalogo 2021*. 2021. URL: https://www.steren.com.mx/.

[36] Melexis INSPIRED ENGINEERING. *MLX90614 family Datasheet Single and Dual Zone Infra Red Thermometer in TO-39*. 2019. URL: www.melexis.com.

[37] JIMBLOM. *MLX90614 IR Thermometer Hookup Guide*. 29 Octubre 2015. URL: https://learn.sparkfun.com/.

[38] ON Semiconductor. *MOC3010M, MOC3011M, MOC3012M, MOC3020M, MOC3021M, MOC3022M, MOC3023M 6-Pin DIP Random-Phase Triac Driver Output Optocoupler (250/400 V Peak)*. 2019. URL: http://www.onsemi.com/.

[39] ST SGS-TOMSON MICROELECTRONICS. *BTA08 S/A SENSITIVE GATE TRIACS*. 1995. URL: https://www.alldatasheet.com/.

[40] AG Electrónica S.A. de C.V. *OKY3001: Módulo de lectura y escritura de tarjeta SD*. 2018. URL: http://www.agelectronica.com/.

[41] UNIT ELECTRONICS. *Display LCD 16×2 Fondo Azul / Amarillo con I2C, Recuperado 25 de Agosto de 2021*. URL: https://uelectronics.com/.

[42] Julio César García Guillén. «Diseño y construcción del sistema de adquisicion de datos y control de un deshidratador de charolas giratorias». Tesis de ingenieria. Universidad Tecnológica de la Mixteca, 2019.