UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – INSTRUMENTACIÓN ACÚSTICA

CONTROL AUTOMATIZADO DE LA FRECUENCIA DE VIBRACIÓN EN UN SISTEMA PARA MEDIR CUERDAS DE INSTRUMENTOS MUSICALES

TESIS

PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

CARLOS ALBERTO PAZ MEDINA

TUTOR PRINCIPAL

FELIPE ORDUÑA BUSTAMANTE

INSTITUTO DE CIENCIAS APLICADAS Y TECNOLOGÍA (ICAT)

MÉXICO, D. F. DICIEMBRE 2022

RESUMEN

En la práctica musical y en la acústica musical es de interés estudiar el comportamiento de las cuerdas de los instrumentos musicales. En particular las que están hechas de nylon y otros materiales viscoelásticos similares en las que se ha observado que al ser montadas por primera vez en un instrumento musical pasan por un periodo de ajuste interno de "relajación" durante un periodo de tiempo en el que afecta la frecuencia de vibración. Esta cuestión también resulta de interés para la industria de la fabricación de cuerdas para instrumentos musicales y el objetivo de estas de ofrecer la mejor calidad posible a los consumidores. En esta tesis se propone ampliar el sistema experimental que se ha desarrollado en el Grupo de Acústica y Vibraciones del ICAT, con un mecanismo nuevo que sea capaz de ajustar la frecuencia de vibración de las cuerdas de manera automática, y así poder llevar un registro la frecuencia de vibración en cuerdas nuevas de nylon que permitirá a futuro estudiar el proceso de relajación de este material en el laboratorio. Dicho estudio no forma parte de los objetivos de esta tesis. Este proyecto parte del sistema experimental que ya existe en el laboratorio, en el que ya se cuenta con un método para medir y registrar pulsaciones en una cuerda tensada. El objetivo de esta tesis es colocar un motor de pasos en un extremo de la cuerda, controlado por medio de un microcontrolador *Arduino Nano*. Esto permite mover el mecanismo de afinación de la cuerda para que aumente o disminuya la tensión de la cuerda, hasta que la frecuencia fundamental de vibración de la cuerda se iguale con una frecuencia deseada que se establezca para el experimento.

ÍNDICE GENERAL.

a. Implementación método Karplus-Strong
b. Método de afinación automática con método de Karplus-Strong

6. Desarrollo experimental

    a. Descripción de sistema experimental
    b. Proceso de afinación automática en el sistema experimental
    c. Medición de cuerdas nuevas de nylon
    d. Resultados experimentales
        i. Datos de las cuerdas a estudiar.
        ii. Análisis de resultados.
    e. Evaluación

7. Conclusiones
8. Referencias
9. Anexos

1. INTRODUCCIÓN

Dentro del estudio de instrumentos musicales de cuerda pulsada, existen varias referencias del estudio de la radiación del sonido que se produce, pero caracterizada por el mecanismo de resonancia de los instrumentos, que es la superficie grande que amplifica las vibraciones hechas por las cuerdas [13], pero existe muy poca información acerca del comportamiento de las cuerdas, sobre todo las cuerdas nuevas, que al ser montadas en un instrumento, tienen un período de ajuste, en el que la frecuencia fundamental varía.

Objetivos

El principal objetivo de este trabajo es agregar al sistema experimental ya desarrollado en [11], la posibilidad de automatizar el proceso de afinación de cuerdas de nylon y que esto permita estudiar los *tiempos de relajación* por los que pasan las cuerdas nuevas montadas en el sistema.

Motivación

Para alcanzar este objetivo, se hará uso de un proyecto anterior en el que se mide la frecuencia fundamental de una cuerda a tensión contante, ahora con la implementación de un sistema mecánico de un motor de pasos adherido a una perilla de afinación de guitarra eléctrica convencional y la lógica de un microcontrolador para hacer el análisis y las operaciones de control necesarias para hacer los ajustes físicos pertinentes.

Durante el primer año de trabajo en el proyecto no se tuvo acceso al laboratorio debido a la contingencia sanitaria de COVID-19, durante ese tiempo y por la necesidad de avanzar en

el proyecto se hizo uso de las herramientas de simulación usando el método de síntesis de Karplus-Strong para simular la vibración de una cuerda y así tener una idea de los parámetros que se necesitan tomar en cuenta para modificar la afinación de la cuerda y también para hacer las consideraciones correspondientes a la hora de la elección de las características de la parte mecánica, en conctreto del motor a usar.

Hipótesis

En el caso particular de las cuerdas de guitarra hechas de nylon, al ser montadas por primera vez en el instrumento, no se puede asegurar la correcta afinación de dichas cuerdas, ya que pasan por un tiempo de “relajación”, que en este trabajo se tratará de medir si esto es debido al comportamiento viscoelástico propio de las cuerdas o a algún otro factor como lo puede ser la temperatura o humedad a la que la cuerda se expone.

2. ANTECEDENTES

Soluciones comerciales de afinación automática de cuerdas

Existen varios dispositivos que se han propuesto para automatizar la afinación de instrumentos de cuerda pulsada, dentro de los que han pasado a la parte comercial se encuentran los siguientes:

Roadie Tuner: es un dispositivo pequeño que se coloca en el clavijero del instrumento y registra la vibración de la cuerda, analiza la afinación y mueve automáticamente la perilla hasta lograr una afinación correcta. Las especificaciones del producto indican un intervalo de detección de frecuencias desde 27.5 Hz hasta 668.84 Hz, y es capaz de sincronizar vía Bluetooth con un teléfono inteligente para configurar la afinación deseada.

Figura 1.- Imagen comercial demostrativa del Rodie Tuner

Tronical: Un sistema de tres componentes que se adaptan a casi cualquier modelo comercial de guitarras acústicas y eléctricas. El primer componente es un sistema de clavijeros

motorizados que en la compañía llama *RoboHeads*, el segundo elemento es un dispositivo cuadrado donde se hace el procesamiento de señal, con una pantalla que sirve como interfaz visual con el usuario. El tercer y último componente del sistema es una base por donde pasan los engranajes del sistema mecánico.

Esta alternativa ofrece una instalación rápida y mantiene la integridad del instrumento. Las especificaciones que ofrece el fabricante señalan que el sistema tarda aproximadamente 5 segundos en afinar el instrumento, ofrece almacenamiento de hasta 36 combinaciones de afinación diferentes, además de que ofrece dos modos de operación dependiendo de las condiciones donde se encuentre el dispositivo; la primera, afinando todas las cuerdas del instrumento al mismo tiempo si se encuentra en un ambiente silencioso, y la segunda, el modo de afinación cuerda por cuerda para ambientes ruidosos.

Figura 2.- Imagen comercial demostrativa del dispositivo Tronical

Aunque ambos ejemplos de soluciones comerciales son capaces de mantener una cuerda de guitarra en una frecuencia deseada, el objetivo es diferente al que busca esta tesis, ya que no guardan datos que puedan ser analizados para caracterizar las cuerdas. Éstas soluciones están más enfocadas en obtener resultados de manera más rápida que la que se propone en este trabajo, ya que están pensados para usuarios finales (músicos ejecutantes), mientras que el sistema que se propone está más enfocado en el estudio y caracterización de las cuerdas. Desafortunadamente los fabricantes no comparten la información de cómo es el proceso de afinación de sus dispositivos de forma detallada, por lo que no existen datos públicos de la precisión de estos dispositivos. En la sección de referencias se encuentran los enlaces a las hojas de datos de ambos dispositivos, los cuales solo dan simple

información de la operación de los dispositivos, pero no de cómo es la manera en que estos operan, es posible conjeturar que ambos dispositivos funcionan con alguna clase de sensor de vibraciones, ya que pueden ser utilizados en instrumentos como guitarras eléctricas, las cuales, no tienen un sistema de amplificación acústica, por lo que es poco probable que estos dispositivos de afinación automática funcionen con un sensor de presión sonora (micrófono).

Sistema de laboratorio para medir cuerdas de instrumentos musicales

Como trabajo previo a este proyecto, el grupo de Acústica y Vibraciones del ICAT desarrolló un sistema experimental para medir características en cuerdas de instrumentos musicales tales como: afinación, armonicidad, brillantez tímbrica, etc. El cuál consiste en un sistema que permite someter una cuerda a tensión constante con el uso de una pesa, pulsarla de manera automatizada, registrar las señales de fuerza vibratoria y poder procesarlas de manera digital para obtener el espectrograma, el espectro vibratorio, las frecuencias de vibración, entre otros parámetros [1].

3. MARCO TEÓRICO

Ecuación de onda en cuerdas

Las ondas son de suma importancia en gran parte de la física, ya que transportan energía, momento y momento angular. Las ondas mecánicas, son las que se viajan en medios tangibles y se rigen con las ecuaciones derivadas de las leyes de la mecánica.

En una cuerda, la onda se forma al aplicar una fuerza, que hace que la cuerda cambie su posición formando una nueva curva, que va cambiando a lo largo del tiempo.

Para describir dicho movimiento, se deben usar las leyes de conservación de movimiento, que describe cómo se mueve el objeto en una situación en particular. La ecuación de onda, es una ecuación diferencial, lo que significa que marca la relación entre una o más derivadas, y una función y ciertas constantes externas.

La ecuación de onda especifica cómo la función *y* que representa el desplazamiento vertical de la cuerda, cambia en el medio, de la siguiente manera:

$$\frac{\partial^2 y}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 y}{\partial t^2} \qquad \text{E 3.1}$$

Donde *c* representa la velocidad de propagación de onda.

Es necesario que la ecuación de onda se complemente con “condiciones de frontera”, que permiten describir trayectorias u ondas explícitas.

Solución de D’Alembert para la ecuación de onda

La ecuación de movimiento de una cuerda se representa con la ecuación de onda en una sola dimensión. La solución general de la ecuación de onda puede expresarse en una forma

particular, conocida como *solución de d'Alembert,* esta solución es particularmente adecuada para representar estados dinámicos de un sistema, como cuerdas o barras [2].

Para el desarrollo consideremos la ecuación de onda para una dimensión continua que esto nos ayuda a simplificar la situación, ya que, en un sistema continuo infinito, tenemos ondas viajeras de cualquier frecuencia.

La ecuación de onda (E 3.1), puede reescribirse en su forma diferencial de la siguiente manera:

$$\left(\frac{\partial}{\partial t} + c\,\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} - c\,\frac{\partial}{\partial x}\right) f = \left(\frac{\partial}{\partial t} - c\,\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} + c\,\frac{\partial}{\partial x}\right) y = 0 \quad \text{E 3.2}$$

Claramente podemos notar que tenemos solución cuando se cumple:

$$\left(\frac{\partial}{\partial t} + c\,\frac{\partial}{\partial x}\right) y = 0 \quad \text{E 3.3}$$

O

$$\left(\frac{\partial}{\partial t} - c\,\frac{\partial}{\partial x}\right) y = 0 \quad \text{E 3.4}$$

Proponemos la función:

$$y_+(x,t) = f(z)|_{z=x-ct} = f(x-ct) \quad \text{E 3.5}$$

Donde $f(z)$es cualquier función arbitratia de *z,* que permite predecir la posición de la cuerda con respecto a la velocidad de propagación *c.* Usando regla de la cadena, aplicamos diferencial con respecto a la variable del tiempo, y a la espacial para obtener respectivamente:

$$y_{+,t} = -cF'(x-ct) \quad \text{E 3.6}$$

Y

$$y_{+,x} = F'(x-ct) \quad \text{E 3.7}$$

Se puede notar que la función $f(x-ct)$ representa una forma de onda viajando en la dirección positiva del eje *x* y a esta solución la llamaremos *onda viajera positiva*.

Ahora consideramos:

$$y_{-}(x,t) = g(z)|_{z=x+ct} = g(x+ct) \qquad \text{E 3.8}$$

Realizando el mismo procedimiento podemos concluir que $g(x+ct)$ es una onda viajera en el sentido negativo del eje *x* y nos referiremos a ella como *onda viajera negativa*. Por lo tanto, la solución completa de la ecuación de onda está dada por:

$$y(x,t) = f(x-ct) + g(x+ct) \qquad \text{E 3.9}$$

A esta solución se le conoce como solución de d'Alembert. Se puede comprobar que cualquier solución puede descomponerse en dos ondas viajeras en direcciones opuestas [2].

La solución de d'Alembert también se puede expresar de la siguiente forma alternativa:

$$y(x,t) = f(t-x/c) + g(t+x/c) \qquad \text{E 3.10}$$

Redefiniendo *f* y *g* ahora como funciones arbitrarias del tiempo *t*.

Solución de D'Alembert para dos extremos fijos

Al definir las condiciones de frontera como ambos extremos fijos, tenemos algo como en la siguiente figura:

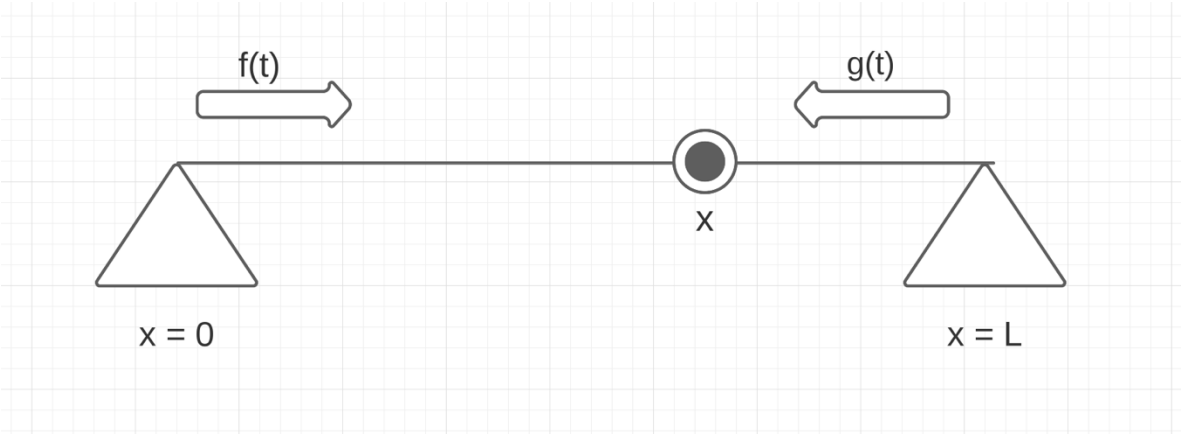


Figura 3.- Sistema de cuerda con dos extremos fijos

La ecuación 3.9 también se puede expresar como:

$$y(x,t) = f(t - {}^{x}/_{c}) + g(t - {}^{(L-x)}/_{c}) \qquad \text{E 3.11}$$

$$y(x,t) = f(t - {}^{x}/_{c}) + g(t - {}^{L}/_{c} + {}^{x}/_{c}) \qquad \text{E 3.12}$$

Ahora haciendo el análisis en los extremos, se tiene:

$$y(0,t) = 0 \rightarrow f(t) + g(t - {}^{L}/_{c}) = 0 \qquad \text{E 3.13}$$

$$y(L,t) = 0 \rightarrow f(t - {}^{L}/_{c}) + g(t) = 0 \qquad \text{E 3.14}$$

De donde se obtiene:

$$f(t) = -g(t - {}^{L}/_{c}) \qquad \text{E 3.15}$$

$$g(t) = -f(t - {}^{L}/_{c}) \qquad \text{E 3.16}$$

Considerando ${}^{L}/_{c}$ como un retardo de la onda original, se puede reescribir E 3.15 como:

$$g(t - {}^{L}/_{c}) = -f(t - {}^{2L}/_{c}) \qquad \text{E 3.17}$$

Sustituyendo en E 3.15, se tiene:

$$f(t) = -[-f(t - {}^{2L}/_{c})] \rightarrow f(t - {}^{2L}/_{c}) \qquad \text{E 3.18}$$

Y por lo tanto:

$$g(t) = g(t - {}^{2L}/_{c}) \qquad \text{E 3.19}$$

Por lo que se puede expresar E 3.11 de la siguiente manera:

$$y(x,t) = y(x, t - {}^{2L}/_{c}) \qquad \text{E 3.20}$$

Para tener una simulación más cercana al caso real, se puede agregar un valor de coeficiente de reflexión que combina la reflexión en ambos extremos $x = 0$ y $x = L$ en la ecuación E 3.20 de la siguiente forma:

$$y(x,t) = R * \ y(x, t - {}^{2L}/_{c}) \qquad \text{E 3.21}$$

Siendo *R* el valor de coeficiente de reflexión con valores entre 0 y 1.

La ecuación E 3.21, siendo usada de referencia para el desarrollo del algoritmo de Karplus-Strong.

Algoritmo de Karplus-Strong

Existen diferentes formas de lograr síntesis de instrumentos musicales de forma digital, pero la mayoría de ellas requieren de ciertas especificaciones tanto en hardware como en software que llegan a hacer muy costoso el proceso, es por eso que Kevin Karplus y Alex Strong idearon un nuevo algoritmo que reduce considerablemente los requerimientos técnicos para lograr un resultado placentero al oído humano y que además podría correr en básicamente cualquier equipo de cómputo.

Este algoritmo simula impulsos de afinación precisa haciendo uso de señales como una colección de ruido. La señal es alimentada a través de una línea de retraso donde la longitud depende de la frecuencia de la nota deseada. La señal retrasada es mandada a través de un filtro pasa bajas que suaviza todas las demás frecuencias ajenas a la frecuencia del tono deseado.

El algoritmo está basado en un modelo anterior conocido como síntesis de tabla de ondas (*wavetable synthesis* en inglés). Que consiste en repetir un número de muestras, que al igualar la longitud de la muestra retrasada con la frecuencia fundamental deseada. La retroalimentación de la señal de retraso únicamente amplifica la fundamental y sus armónicos. Por lo que la salida del algoritmo nos da el sonido simulado de una cuerda pulsada con un tiempo corto, como si fuera pulsada. [3]

Teniendo Y_n como el valor de la n-ésima muestra, se puede escribir un algoritmo de la forma:

$$Y_n = R * Y_{n-p}$$

Donde el parámetro *p* es la longitud de la tabla de datos o también se le conoce como el parámetro de periodicidad, que representa la cantidad de memoria que se necesita y el periodo del tono (en muestras).

La más simple modificación hecha por Alex Strong, es promediar dos muestras consecutivas, matemáticamente expresado como:

$$Y_n = \frac{R}{2}\left(Y_{n-p} + Y_{n-p-1}\right)$$

Resulta que, al realizar esta operación, se produce un decaimiento lento de la forma de onda. El tono resultante de este algoritmo tiene un timbre que corresponde a un periodo de $p + \frac{1}{2}$ muestras, y suena como el decaimiento de una cuerda pulsada. Ya que solo se necesita sumar y cambiar, el algoritmo es rápido y fácil de implementar en microprocesadores.

Para lograr el sonido de una cuerda pulsada, es deseable iniciar la nota con gran cantidad de armónicos altos y para lograrlo la tabla de ondas es llenada con valores aleatorios al inicio de cada nota. Como las muestras de la tabla se repiten, la aleatoriedad no produce ruido no deseado. Con el algoritmo de decaimiento en el que se retrasa un cierto número de muestras *p* la misma tabla, los armónicos altos decaen rápidamente, produciendo un sonido muy similar al de una guitarra. En la figura se muestra el flujo de señal del algoritmo.

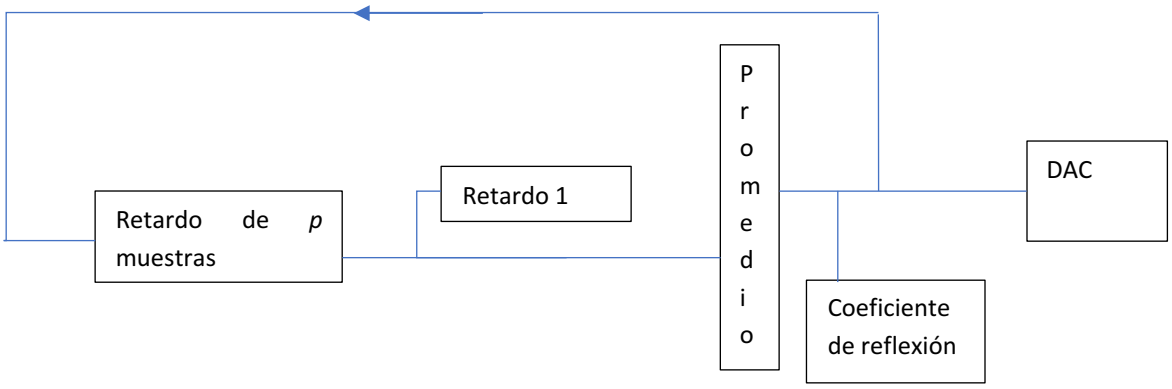


Figura 4.- Diagrama de flujo de señal del algoritmo propuesto por Karplus-Strong.

Existen algunas limitaciones que hay que considerar en el parámetro de periodicidad *p.* Si *p* es un valor pequeño, la variación entre las diferentes condiciones iniciales será relativamente largo, resultando en un pobre control de la amplitud. También, como el timbre de la nota es determinado por *p* y este debe ser un valor entero, así que cuando *p* es un valor grande las frecuencias disponibles están muy cerca unas de otras, pero cuando *p* es un valor pequeño, estas frecuencias están apartadas unas de otras. Si se quiere cubrir todo el rango de frecuencias de una guitarra (hasta alrededor de los 880 Hz), estas restricciones requieren frecuencias de muestreo de al menos 28.6 KHz [4].

Intervalo musical y percepción

El tono se define como un sonido periódico que provoca una sensación de altura musical. Los tonos pueden ser puros (variaciones senoidales en presión de aire a una sola frecuencia) o complejos. Los tonos complejos se pueden dividir en armónicos (periódicos, con un rango de repetición conocido como frecuencia fundamental, y están formados por sumas de senoidales con frecuencias que son múltiplos enteros o armónicos de la frecuencia fundamental), e inarmónicos (sumas de senoidales que no son múltiplos enteros de la frecuencia fundamental) [5].

Los intervalos musicales a menudo son expresados en cents o centésimos de tono en un sistema conocido como *octava de igual temperamento,* en el que la diferencia entre una frecuencia fundamental (*f)* y su octava (*2f),* es de 1200 cents [6].

El hecho de que una octava sea igual a 1200 cents conduce a las relaciones de potencias de 2, y que si se busca la diferencia en cents entre dos notas que forman un intervalo musical, se puede calcular de la siguiente manera:

$$\frac{f_2}{f_1} = 2^{\frac{c}{1200}}$$

Siendo *c* el número de cents, y f_1 y f_2 las frecuencias que forman el intervalo.

Despejando *c* de la ecuación anterior, tenemos:

$$c = 1200 * \log_2 \frac{f_2}{f_1}$$

El grado de sensibilidad que existe en la percepción de la altura frecuencial o capacidad de resolución de frecuencia depende de factores como frecuencia, cambios de intensidad y duración de los tonos en cuestión, y varia de persona a persona y de acuerdo con el estudio realizado por Zwicker, Flottorp y Stevens (1957), para una frecuencia de 2000 Hz, un cambio de 10 Hz (u 8.63 cents) ya es perceptible por el oído humano promedio [7].

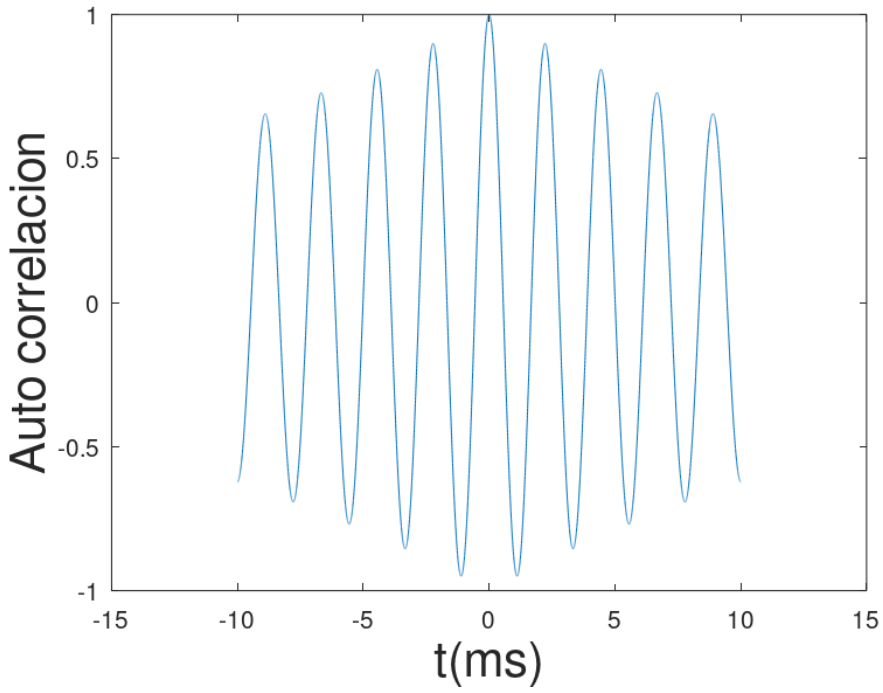
4. MÉTODO DE AFINACIÓN AUTOMÁTICA

Método de estimación de frecuencia fundamental

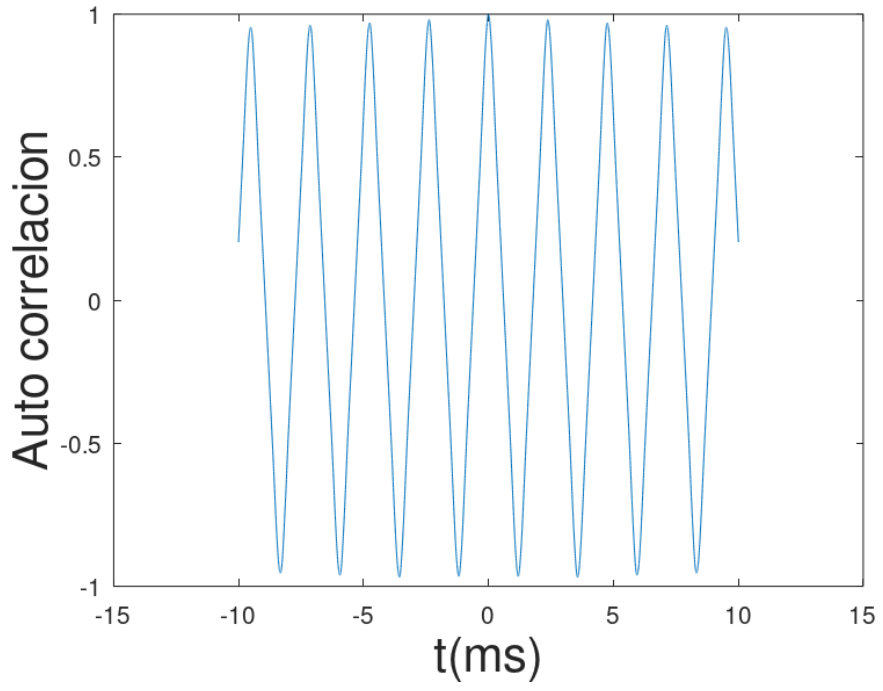
Para determinar la frecuencia fundamental de la cuerda se usa un script desarrollado en *Octave versión 4.0.3* que nos permite calcular la frecuencia fundamental a través del método de autocorrelación, el punto de simetría, que es donde la función de autocorrelación es igual a 1, como se puede ver en la sección de anexos y cómo podemos observar en la siguiente figura:

```
fmin = fminmax(1);
fmax = fminmax(2);
N = length(y);
minlag = 1 + floor(fs/fmax);
maxlag = 1 + ceil(fs/fmin);
tau_ms = (-maxlag:maxlag).'*1000/fs;
tau_2 = (0:maxlag).'*1000/fs;
y1_corr = xcorr(mean(y,2), maxlag, 'coeff');
figure;
plot(tau_ms,y1_corr);
xlabel('t(ms)', 'fontsize',20);
ylabel('Auto correlacion','fontsize',20);
```

Figura 5. A.- Extracto de código donde se hace el cálculo de autocorrelación normalizada con un valor máximo en 1 a t = 0.


Auto correlacion
t(ms)
1
0.5
0
-0.5
-1
-15
-10
-5
0
5
10
15

(a)

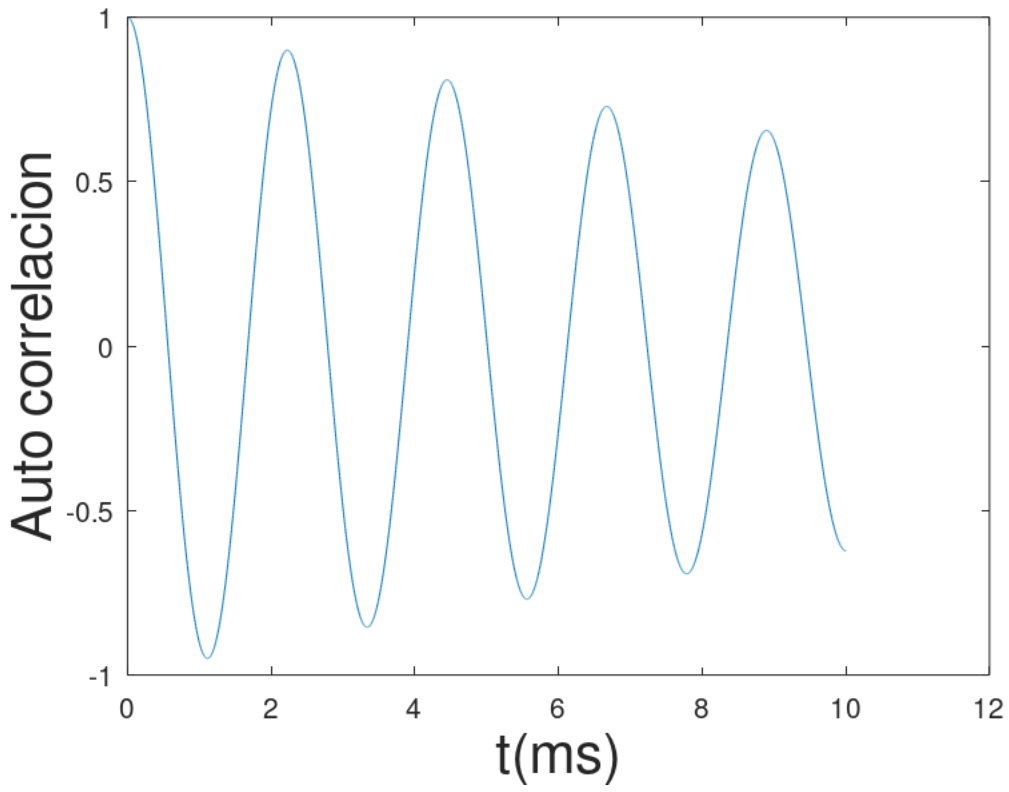

Auto correlacion
t(ms)
1
0.5
0
-0.5
-1
-15
-10
-5
0
5
10
15

(b)

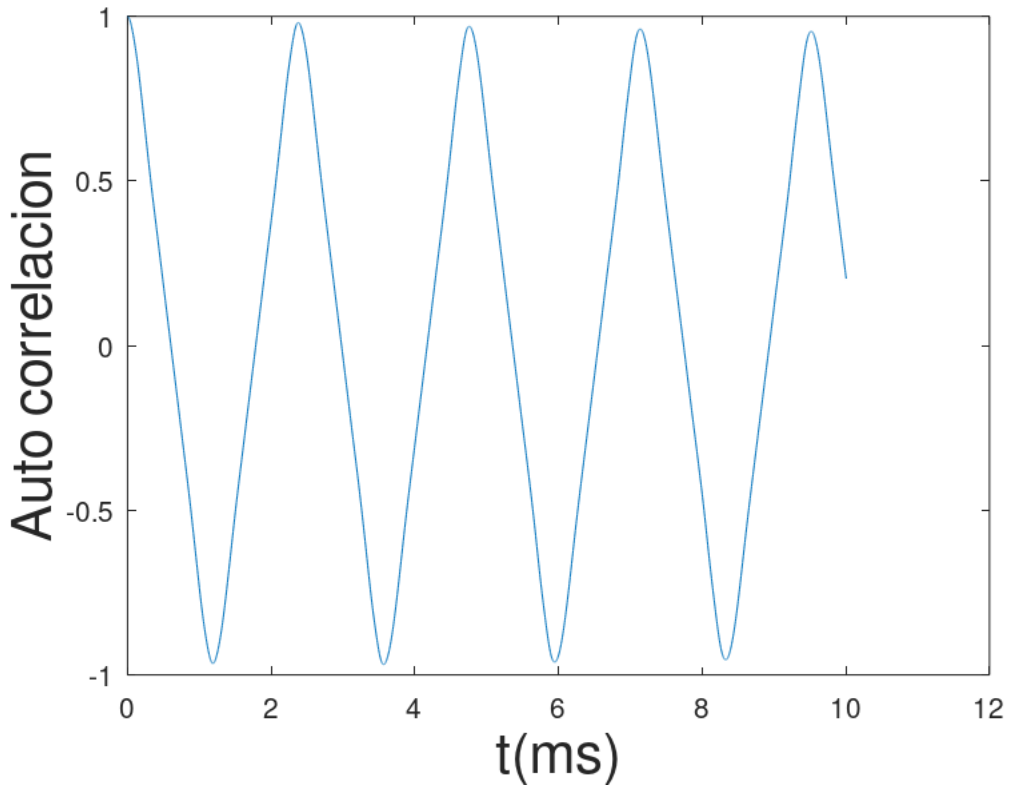
Figura 6.- Gráficas de autocorrelación normalizada con el valor máximo en t = 0, (a) es el cálculo de la simulación con un *R* = 0.90, y (b) es el cálculo de una medición experimental.

```
y_corr = ifftshift(y1_corr); %% Start at zero lag
figure;
plot(tau_2, y_corr(1:maxlag+1));
xlabel('t(ms)', 'fontsize',20);
ylabel('Auto correlacion','fontsize',20);
[~,bestlag] = max(y_corr(minlag:maxlag));
bestlag = bestlag + minlag - 1;
lags = [bestlag-1 bestlag bestlag+1].';
bestlag = sum(lags.*y_corr(lags)) / sum(y_corr(lags));
F0 = fs / (bestlag - 1);
```

Figura 5. B.- Fragmento del código donde se analiza la parte positiva de la correlación y se busca el primer máximo de periodicidad para determinar la frecuencia fundamental.



(a)



(b)

Figura 7.- Gráficas de autocorrelación positiva, (a) es el cálculo de la simulación con un *R* = 0.90, y (b) es el cálculo de una medición experimental.

Donde con la variable llamada *bestlag* se busca la posición en que la función tenga la mayor autocorrelación con corrimiento de tiempo no nulo.

Este método no es tan exacto y genera un error al momento de hacer los cálculos para el proceso de ajuste mecánico, por lo que es necesario usar un método para perfeccionar la búsqueda de la frecuencia fundamental en el que el método de autocorrelación nos da un rango de valores posibles en los que podría estar localizada la frecuencia fundamental. Hacemos uso de un nuevo script (ver en anexos) basado en el espectro de la señal para encontrar el mayor punto energético que corresponde con la frecuencia fundamental, con mayor precisión.

Proceso de ajuste mecánico

Una vez teniendo el valor de frecuencia fundamental, se necesita obtener una relación entre la diferencia de frecuencia existente entre la frecuencia fundamental a la hora de hacer la medición y la frecuencia deseada, y la cantidad de cuerda que necesita ser retirada o adicionada a la longitud vibrante para poder alcanzar dicha frecuencia deseada.

Partimos por definir la frecuencia en términos de la velocidad de propagación y la longitud vibrante en una cuerda (E 4.8).

Como siguiente paso, definimos ahora la velocidad de propagación en términos de la tensión sobre la cuerda y la densidad lineal del material, de la siguiente forma:

$$V = \sqrt{\frac{T}{\rho_L}} \qquad \text{E 5.1}$$

Con T = Tensión aplicada y $\rho\__L$= Densidad lineal del material.

De acuerdo a la descripción de un proceso elástico convencional, estas se pueden expresar de la siguiente forma:

$$T = Y * A * \mathrm{s} \qquad \text{E 5.2}$$

$$s = \frac{L_{ext} - L_0}{L_0} \qquad \text{E 5.3}$$

$$\rho_L = \rho * A \qquad \text{E 5.4}$$

donde Y es el módulo de rigidez longitudinal (módulo de Young), ρ es la densidad volumétrica de masa, A es el área de la sección transversal de la cuerda y s es la deformación relativa neta desde la longitud sin tensión L_0, hasta la longitud extendida L_{ext}. Suponemos

que $L < L_0$; es decir, la longitud vibrante L de la cuerda es menor que la longitud sin tensión L_0 y menor que la longitud extendida L_{ext}. La velocidad de propagación se puede expresar también en la forma:

$$V = ((E / \rho)\, s)^{1/2} \qquad \text{E5.5}$$

Esta es independiente del área de la sección transversal A y se relaciona con la velocidad de ondas compresionales longitudinales en el material de la cuerda: $c_l = (E / \rho)^{1/2}$. En estos términos, la deformación s se puede escribir como una relación cuadrática de velocidades, en la forma:

$$s = (V / c_l)^2 \qquad \text{E 5.6}$$

$$= (2\, L\, f)^2 / (E / \rho) \qquad \text{E 5.7}$$

Esto relaciona la deformación s con la frecuencia f.

En una cuerda con longitud vibrante L fija, la frecuencia de vibración $f = V / 2\, L$, cambia

en una cantidad df a: $f + df = (V + dV) / 2\, L$, en función de un cambio dV en la velocidad de propagación. Los cambios relativos son por lo tanto iguales:

$$\mathrm{d}f / f = \mathrm{d}V / V \qquad \text{E 5.8}$$

De manera similar, se puede expresar el cambio relativo de velocidad dV / V en términos del cambio relativo en la deformación neta ds / s, en la siguiente forma:

$$\mathrm{d}V / V = (1/2)\, \mathrm{d}s / s \qquad \text{E 5.9}$$

Si la longitud extendida L_{ext}, cambia a $L_{ext} + dx$ en una cantidad dx, entonces la deformación neta $s = (L_{ext} - L_0) / L_0$, cambia a: $s + ds = s + dx / L_0$, se obtiene entonces que:

$$\mathrm{d}s / s = \mathrm{d}x / s\, L_0 \qquad \text{E 5.10}$$

Finalmente, se establece la siguiente relación entre el cambio de longitud extendida dx y el cambio de frecuencia df:

$$dx / L_0 = 2\ s\ df / f$$ E 5.11

$$dx / \Delta L = 2\ df / f$$ E 5.12

en donde *2s = 2 (2Lf)2 / (E/ρ) = 2T / EA*, es el factor de afinación que indica el estiramiento relativo *dx / L_0* que es necesario para modificar la frecuencia en *df / f*.

5. SIMULACIÓN

Debido a que durante el primer año del desarrollo de esta tesis no se tuvo acceso a las instalaciones del ICAT, se optó por comenzar el proyecto con pruebas simuladas a través de métodos numéricos programados en el software Matlab, donde se utilizó el algoritmo de Karplus-Strong (basado en la solución de D'Alembert de la ecuación de onda) para generar la síntesis de audio más parecida al sonido de una cuerda pulsada.

Cabe resaltar que el desarrollo de la simulación no tiene como objetivo comparar los resultados con la implementación del experimento real, sino que se busca que nos dio una idea de la metodología que se necesita para poder desarrollar el experimento en el laboratorio a la hora que las condiciones sanitarias lo permitieran.

Implementación método Karplus-Strong

Existen diferentes formas de implementar el algoritmo de Karplus-Strong, en este trabajo se utilizó el método de *contador decreciente* que consiste en una tabla de ondas que se almacena de atrás hacia adelante en la memoria del dispositivo, con un puntero en el valor actual. Como cada muestra es parte de la salida, el puntero se decrementa para obtener el siguiente valor. Cuando el puntero llega a un valor menor al del fondo de la tabla, se vuelve al principio de la tabla. Que se asemeja a la solución de D'Alembert, simulando la onda viajera en ambos sentidos (E 3.9).

En este caso, se realizaron simulaciones en Matlab para lo cual se utilizó la función:

$$A(t) = \sin(\omega\, t) = \sin\big(2 * \pi * f * t(D)\big) \qquad \text{E 5.1}$$

Donde el vector de tiempo se rige en función del retardo visto en el planteamiento del algoritmo y dentro de la simulación la definimos inicialmente como:

$$D = \frac{f_s}{f_i} \qquad \text{E 5.2}$$

Teniendo f_s como la frecuencia de muestreo y f_i siendo la frecuencia inicial de la simulación. Lo que nos da como resultado una forma de onda como en la figura

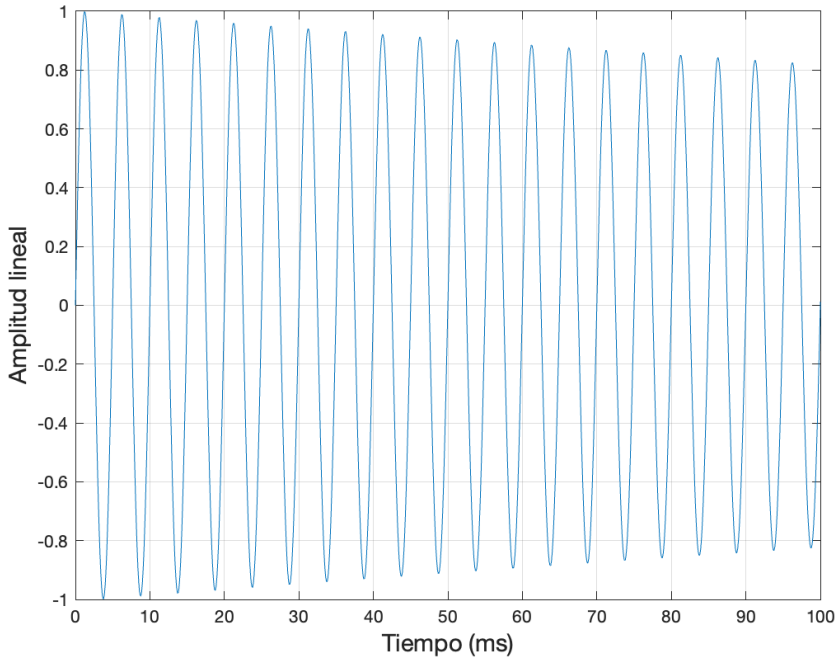


Figura 8.- Forma de onda resultante de la implementación del algoritmo de Karplus-Strong

Para el análisis de la frecuencia se usó una herramienta de Matlab conocida como *pspectrum* que nos devuelve las frecuencias correspondientes a los estimados espectrales de la muestra, los cuales podemos graficar de la siguiente forma:

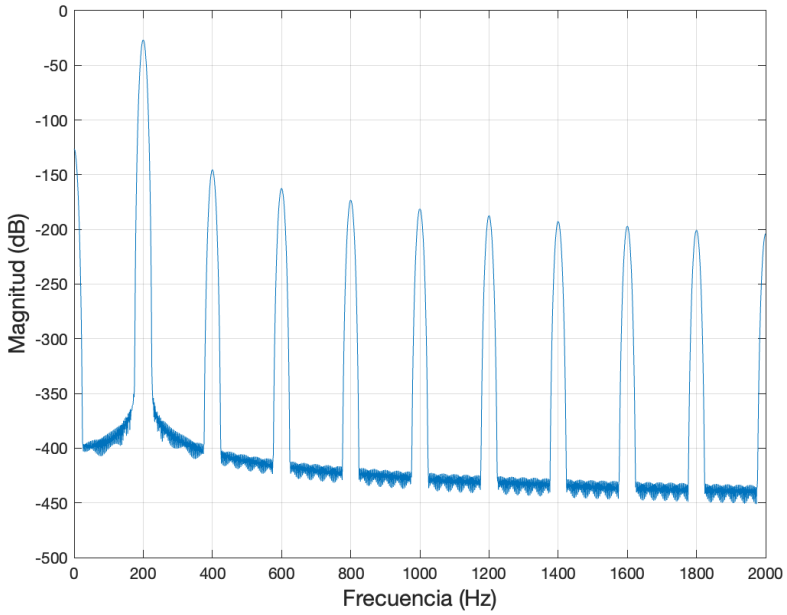


Figura 9.- Gráfica de espectro de la simulación del método de Karplus-Strong

El desarrollo completo del algoritmo en Matlab se encuentra disponible en la sección de anexos.

Método de afinación automática con método de Karplus-Strong

Dentro de esta primera simulación también se implementó un primer acercamiento a la afinación automática, ya que se da un parámetro de frecuencia inicial, así como una frecuencia deseada. La variable que cambia la frecuencia es el retardo, lo que permite realizar iteraciones actualizando el valor del retardo. De la ecuación E 5.2 se puede reescribir como:

$$f * D = f_s \qquad \text{E 5.3}$$

Aplicando logaritmo en ambos lados de la ecuación, se puede reescribir:

$$\ln(f) + \ln(D) = \ln(f_s) \qquad \text{E 5.4}$$

Derivando ambos lados de la ecuación, se obtiene:

$$\frac{df}{f} + \frac{dD}{D} = 0 \qquad \text{E 5.5}$$

$$\frac{dD}{D} = -\frac{df}{f} \qquad \text{E 5.6}$$

Volviendo a usar la ecuación E 5.2 y reordenando la ecuación, se obtiene:

$$D = D - \left[\left(\frac{f_s}{{f_f}^2}\right) * (f_d - f_f)\right] \qquad \text{E 5.7}$$

Donde f_f es la frecuencia fundamental en el momento del análisis y f_d es la frecuencia deseada. Además de que definimos una condición para que la estimación sea lo suficientemente correcta, y no se quede en un ciclo infinito se utiliza el valor mínimo de cambio de retardo, que, al ser un valor discreto, el mínimo cambio que puede tener es 1, dicha condición está definida como:

$$1 = \left(\frac{f_s}{{f_f}^2}\right) \Delta f \qquad \text{E 5.8}$$

Despejando Δf de E 5.8, se obtiene:

$$\Delta f = \left(\frac{{f_f}^2}{f_s}\right) \qquad \text{E 5.9}$$

De la ecuación anterior podemos definir una resolución frecuencial con la que el sistema experimental va a operar.

$$\frac{f_d + \Delta f}{f_d} = 1 + \frac{\Delta f}{f_d} \qquad \text{E 5.10}$$

Sustituyendo E 4.4 en E 4.5:

$$1 + \frac{\left(f_d^2 / f_s\right)}{f_d}$$

$$= 1 + \frac{f_d}{f_s} \qquad \text{E 5.11}$$

Con esta forma de la ecuación anterior, se puede observar que la precisión depende directamente de la frecuencia de muestreo y además se tiene mayor precisión buscando frecuencias bajas.

El siguiente paso es considerar la parte mecánica del experimento. Simulando la parte del motor a pasos que va a hacer el ajuste de la frecuencia fundamental en la cuerda, por lo que se necesita hacer un análisis y encontrar la relación que existe entre el movimiento del motor y la cantidad de cuerda que se retira o añade a la parte vibrante del experimento y cómo todo esto impacta en la frecuencia que produce la cuerda al ser pulsada. Donde al ser todavía una simulación, tenemos que definir parámetros de la cuerda como densidad lineal, calibre de la cuerda, frecuencia inicial, tensión aplicada y módulo de Young del nylon. El script desarrollado en Matlab, realiza el mismo proceso que el sistema experimental, en el que se calcula la frecuencia fundamental a través de la función de correlación nativa de Matlab y una vez que se obtiene, se realiza el cálculo del cambio de longitud de la cuerda, derivada del movimiento del motor, denotada por la variable *dx,* que como se puede observar en la sección de anexos, se define como:

$$dx = \frac{df}{f_{deseada}} \frac{2\,L\,T}{S\,Y} \qquad \text{E 5.12}$$

Donde *df* es la diferencia entre la frecuencia deseada y la frecuencia medida, *S* es el área de la sección transversal de la cuerda, *T* la tensión ejercida sobre la cuerda y *Y* el valor del módulo de Young del nylon.

Al ser un proceso de simulación y no tener parte física qué mover, aquí lo que se necesita para cambiar la frecuencia fundamental de la cuerda sintética es actualizar el valor de retardo que podemos expresar el valor de retardo mediante la ecuación E 5.7.

6. DESARROLLO EXPERIMENTAL

Para los experimentos realizados se eligieron cuerdas de guitarra acústica de nylon de la marca *La Valenciana* y se eligió trabajar con las tres primeras cuerdas que son homogéneas, ya que un recubrimiento o *entorchado* adicional.

Descripción del sistema experimental

En la figura 10, tenemos una imagen del sistema experimental que consiste en una cuerda fija en dos extremos, de un lado se encuentra fija por una porta broca.

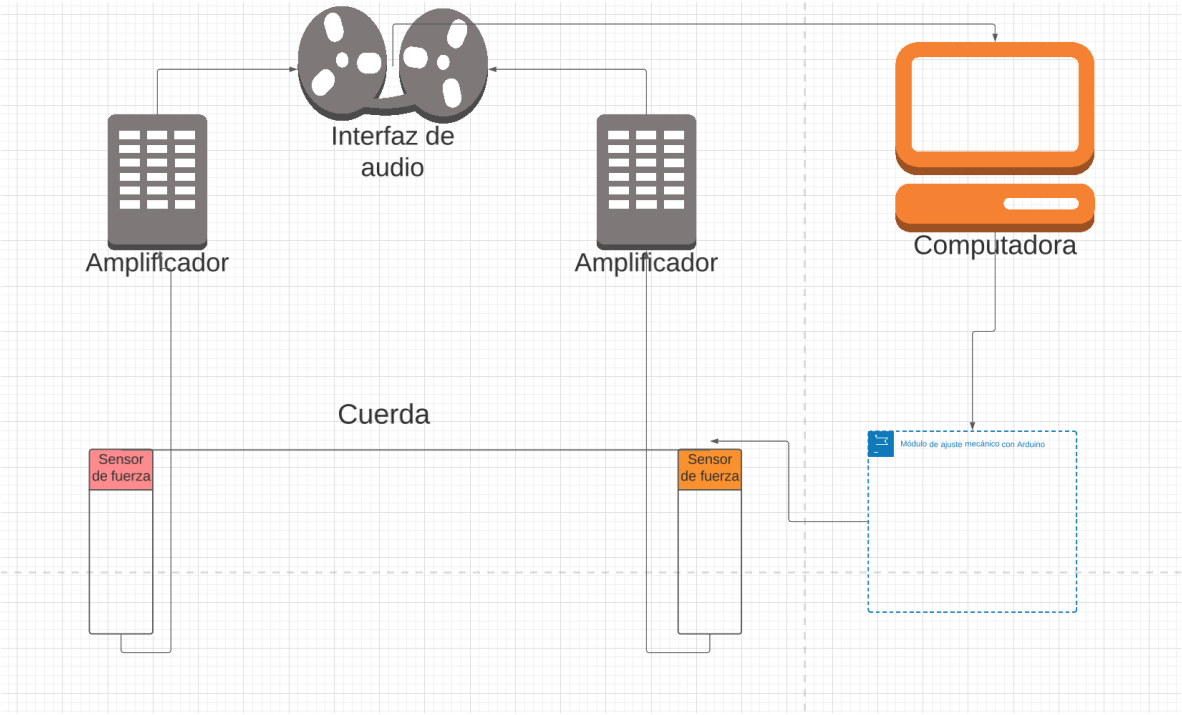
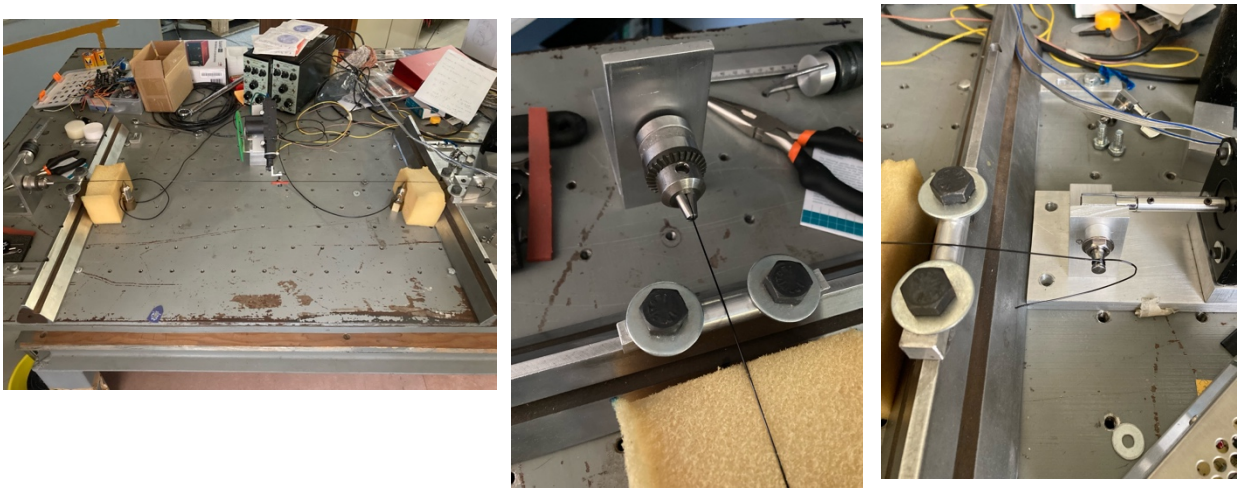


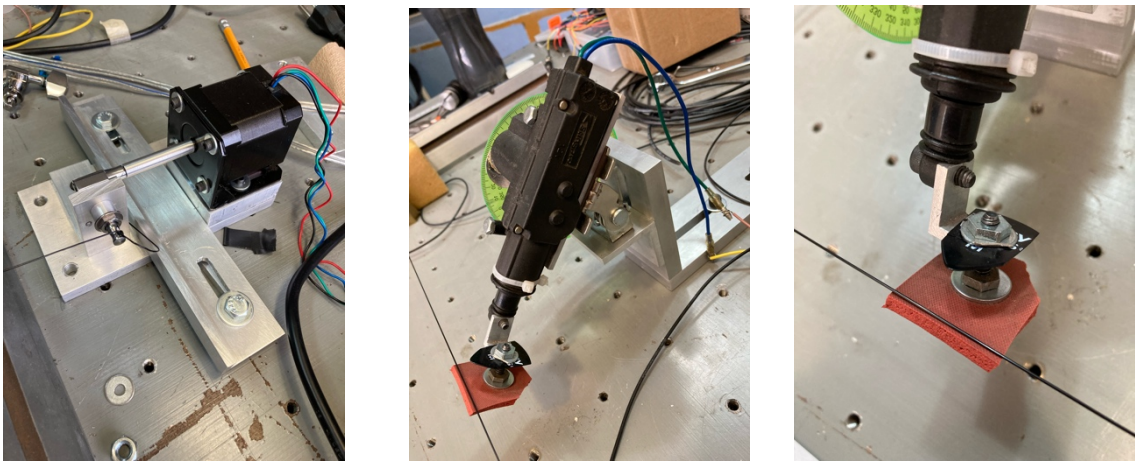
Figura 10.- Diagrama de sistema experimental.



(a) (b) (c)

Figura 11.- a) Fotografía del montaje completo del experimento. b) Acercamiento al porta broca que mantiene fijo un extremo de la cuerda. c) Acercamiento a la perilla de guitarra eléctrica al otro extremo de la cuerda

Y por el otro extremo, la cuerda está sujeta por una perilla de guitarra eléctrica que permitirá hacer el cambio en la longitud vibrante en la cuerda.



(a) (b) (c)

Figura 12.- a) Acercamiento al dispositivo diseñado para tensar o destensar la cuerda. b) Acercamiento al dispositivo pulsador de la cuerda. c) Acercamiento al acoplamiento de la plumilla que ejecuta la pulsación en la cuerda

Para la parte de la pulsación de la cuerda, se adaptó un actuador del seguro de una puerta de auto con un ángulo al cual se le colocó un tornillo con una plumilla de 0.71mm que pone a vibrar la cuerda al momento que el actuador se expande y un amortiguador de espuma para silenciar de manera inmediata cuando el actuador se retrae.

Para poder hacer el proceso de tensar o destensar la cuerda de manera automática se adaptó un motor a pasos junto con una base a la perilla de guitarra como podemos observar en la figura.

Los elementos que implican movimiento (actuador y motor de pasos) son controlados por un microcontrolador *Arduino Nano* que además de mandar pulsos de movimiento, también tiene conectado un sensor de temperatura y humedad, datos que podrían ser útiles para el análisis de los resultados. A este microcontrolador se le cargó el programa incluido en los anexos y se hizo la conexión que se muestra en la figura.

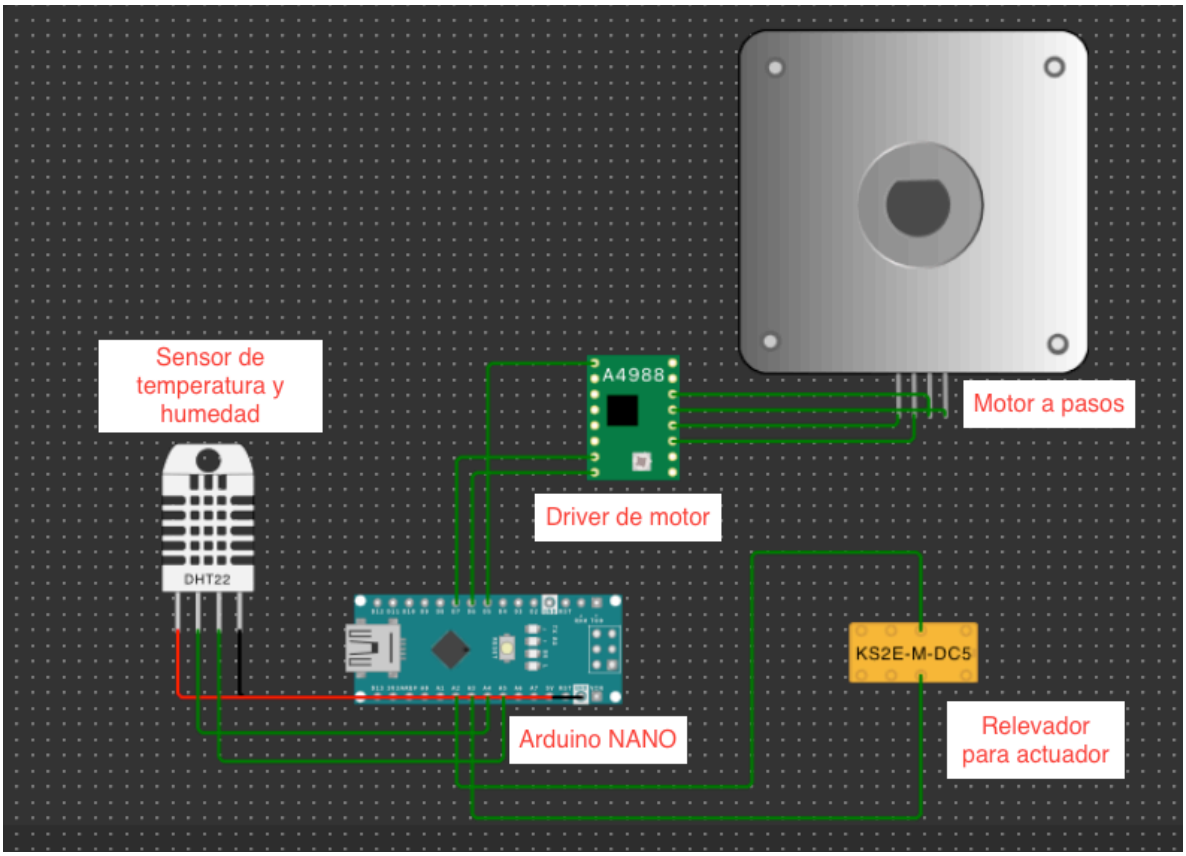
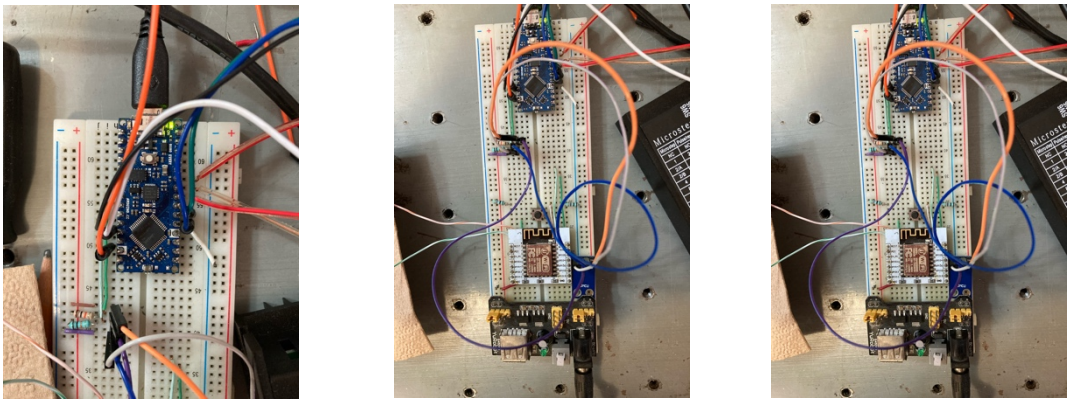


Figura 13.- Diagrama de conexión de Arduino Nano que controla el experimento.



(a) (b) (c)

Figura 14.- a) Acercamiento al Arduino Nano b,c) Acercamiento a la conexión completa del Arduino en protoboard.

El motor a pasos de la marca *Quimat Motor* modelo *17HS19-2004S1*, utilizado para el experimento, requiere un driver de la marca *LMEX* modelo *TB6600*, el cual se encarga de mover el embobinado interno del motor dando dirección y distancia del micropaso de giro, para la realización de este experimento se configuró el motor con un micropaso de 200 pasos por giro completo.

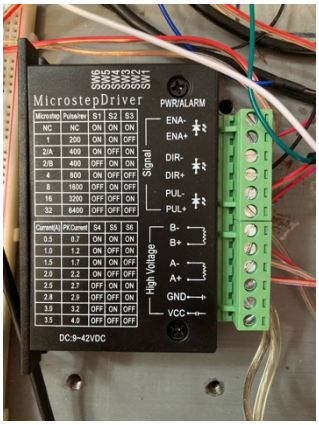


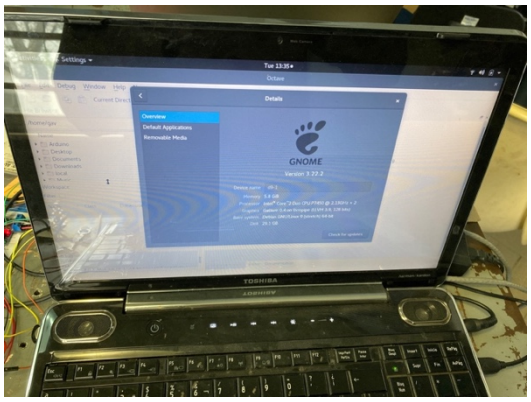
Figura 15.- Fotografía de driver para motor a pasos.

Para la captura de los datos, se colocaron dos cabezas de impedancia marca *Brüel & Kjaer* modelo *Type 8001* los cuales son capaces de medir fuerza. Es necesario pasar la señal por un primer proceso de amplificación con un amplificador marca *Brüel & Kjaer* modelo *Type 2635.* Después de este proceso, la señal pasa a una interfaz de audio marca *Focusrite*

modelo *Scarlett 4i4 de 3ra Generación*, que nos permite hacer la conversión análoga/digital de la señal para poder mandarla a través de cable USB directamente a la computadora marca *Toshiba* modelo *Satellite* con un sistema operativo *Debian GBU/Linux 9,* donde se hace el análisis y el post-procesamiento.

(a)

(b)



(c)

Figura 16.- a) Fotografía de preamplificador de señal *Brüel & Kjaer* modelo *Type 2635.* b) Fotografía de interfaz de audio *Scarlett 2i2 3ra Gen.* (c) Fotografía de laptop utilizada para desarrollar el experimento con la pantalla de información de sus características.

Proceso de afinación automática en sistema experimental

Después de tener el calculo de la frecuencia fundamental como se expuso en la sección 4.b, se debe hacer una relación entre *dx* y el número de pasos necesarios y la dirección que debe tomar el motor para lograr la frecuencia deseada. Se comienza por:

$$\theta_{cil} = \frac{dx}{r_{cil}} * \left(\frac{180}{\pi}\right) \qquad \text{E 6.1}$$

Con θ_{cil} como el ángulo de giro del cilindro y r_{cil}como el radio del cilindro.

Como siguiente paso, se definió el número de vueltas necesarias que debe dar la perilla del afinador para que el cilindro diera una vuelta completa, que fueron 14 giros de la perilla, por lo que podemos definir:

$$\theta_{per} = 14 * \theta_{cil} \qquad \text{E 6.2}$$

Con θ_{per} siendo el ángulo de la perilla.

El último paso para la afinación de la cuerda es poder mandar el dato del número de pasos necesarios y la dirección en la que el motor debe moverse para conseguir llegar a la frecuencia deseada. Para esto, debemos conocer el ángulo con el que gira el motor con cada paso que da, sabiendo que la configuración que le dimos al motor fue de 200 pasos por giro completo, entonces podemos definir:

$$\theta_{mot} = \frac{360}{200} \qquad \text{E 6.3}$$

Donde θ_{mot} son los grados de giro del motor por cada paso realizado.

El número de pasos necesarios que debe dar el motor para alcanzar la frecuencia deseada se calcula de la siguiente manera:

$$Pasos\ motor = \frac{\theta_{per}}{\theta_{mot}} \qquad \text{E 6.4}$$

Para poder mandar este dato al motor, necesitamos hacer uso del microcontrolador Arduino Nano y poder tener comunicación retroalimentada entre su interfaz y el script que realiza los cálculos en Octave, para lo que utilizamos la librería *serial* de Octave y un código en el *Arduino IDE versión 1.8.16* para realizar dicha comunicación.

Dentro del código en Arduino, se integró la parte del pulsado de la cuerda, los pasos del motor y la toma de datos de temperatura y humedad.

En la parte de anexos se encuentra el código completo utilizado, pero en esta sección se expone cada una de las tres partes controladas por el Arduino Nano.

Para la parte del pulsado de la cuerda, definimos la función *actuator* la cuál al recibir un valor mayor a 0 manda un pulso *HIGH* al actuador haciendo sonar la cuerda y si recibe un pulso menor a 0 se pone en estado *LOW* lo que regresa el actuador y amortigua la vibración de la cuerda como podemos ver en el siguiente fragmento del código:

```
void actuator () {

    delay(abs(pulse));
    if (pulse > 0) {
      if (verbose_mode) {
        Serial.println("Hit");
      }
      digitalWrite (HIT, HIGH);
      delay(100);
      digitalWrite (HIT, LOW);
    } else {
      if (verbose_mode) {
        Serial.println("Recover");
      }
      digitalWrite (RECOVER, LOW);
      delay(100);
      digitalWrite (RECOVER, HIGH);
    }
}
```

Figura 17.- Fragmento de código para la activación del actuador, desarrollado en el IDE de Arduino.

Ahora la parte de la comunicación con el motor de pasos, de igual manera que con el actuador, se creó una función que recibe el valor del número de pasos necesarios para lograr la afinación deseada y la dirección de giro de acuerdo al signo de dicho valor, siendo un valor positivo indicación de que el giro sigue el movimiento de las manecillas del reloj, y un valor negativo el movimiento inverso, así como lo podemos ver en el siguiente fragmento de código:

```
void drive_motor () {
  // Ejecutar pasos según variable steps
  if (steps > 0)
  {
    digitalWrite(DIR,LOW);
    for (int i=0; i<steps; i++)    //Forward N steps
    {
      digitalWrite(PUL,HIGH);
      delayMicroseconds(400);
      digitalWrite(PUL,LOW);
      delayMicroseconds(400);
      //delay(100);
    }
  }
  else
  {
    steps = -steps; // Toma valor positivo de steps
    digitalWrite(DIR,HIGH);
    for (int i=0; i<steps; i++)   //Backward N steps
    {
      digitalWrite(PUL,HIGH);
      delayMicroseconds(400);
      digitalWrite(PUL,LOW);
      delayMicroseconds(400);
    }
  }
}
```

Figura 18.- Fragmento de código para el control del motor a pasos desarrollado en el IDE de Arduino

Por último, la parte de la toma de datos de temperatura y humedad se creó una función que al ser llamada, nos devuelve el registro de fecha y hora junto con un valor de temperatura expresada en grados Centígrados y un valor de humedad en porcentaje como podemos ver en la siguiente imagen:

```
void get_th() {
  Wire.beginTransmission(HDC);
  Wire.write(0x00);
  Wire.endTransmission(true);
  delay(200);

  Wire.requestFrom((uint8_t)HDC,(size_t)4,true); //It asks for 4 bytes, 2 bytes per register

  if(Wire.available()){

    temvalmsb = Wire.read();
    temvallsb = Wire.read();
    humvalmsb = Wire.read();
    humvallsb = Wire.read();

    //TEMPERATURE
    temrawvalue = temvalmsb <<8| temvallsb; //Locates each byte in its proper place
    temperature = ((float)temrawvalue/(float)ratio)*165.0-40.0;

    //HUMIDITY
    humrawvalue = humvalmsb <<8| humvallsb;
    humidity = ((float)humrawvalue/(float)ratio)*100.0;

    char buffer1[10];
    char buffer2[10];

    dtostrf(temperature, 6, 2, buffer1);
    dtostrf(humidity, 6, 2, buffer2);

    String str1 = String(buffer1);
    String str2 = String(buffer2);

    str1.trim();
    str2.trim();

    Serial.println(str1 + "," + str2);
```

Figura 19.- Fragmento de código para la toma de lectura de temperatura y humedad desarrollado en el IDE de Arduino.

El experimento consiste en hacer una medición cada minuto en el que se graba un archivo de audio durante los primeros 30 segundos de la medición. El audio es analizado en la segunda parte de la medición para encontrar la frecuencia fundamental de la cuerda en ese momento, con los métodos descritos anteriormente y poder realizar los ajustes necesarios para llegar a la frecuencia deseada, mandando las instrucciones necesarias al motor.

Tomando también lectura de temperatura y humedad del sensor destinado a estas mediciones. Guardando los datos de temperatura, humedad, frecuencia fundamental, ángulo de giro del cilindro del afinador, fecha y hora, en archivos de texto y así poder hacer el análisis de los resultados obtenidos. Dicho procedimiento se realizó 1,200 veces por cada cuerda, además de que se utilizó el script de post-procesamiento utilizado en el trabajo *Sistema de laboratorio para medir cuerdas de instrumentos musicales* para obtener espectrogramas y espectros de amplitud de vibración, el código completo se encuentra disponible en la sección de anexos.

A continuación, se muestran los resultados obtenidos con cada una de las cuerdas trabajadas.

DATOS DE LAS CUERDAS A ESTUDIAR.

Los datos significativos que necesitamos saber de las cuerdas a estudiar son el calibre de cada una de ellas y la frecuencia fundamental deseada en cada una de ellas, ya que están diseñadas para convencionalmente producir una nota en específico cuando están montadas en el instrumento, en este caso, una guitarra acústica. Los datos vienen presentados en la siguiente tabla:

Tabla 1.- Características de las tres cuerdas nuevas de guitarra clásica marca La Valenciana, hechas de Nylon liso, sin entorchado, utilizadas en los experimentos. Se indica la frecuencia de vibración nominal para el tiro de 65 cm, convencional de la guitarra clásica, y para el tiro de L=51 cm usado en los experimentos. Se indican también las frecuencias deseadas f_d.

	Cuerda 1	Cuerda 2	Cuerda 3
Modelo	Cristal	Titanio	Cristal

Calibre (pulg)	0.028	0.032	0.040
Calibre (mm)	0.711	0.813	1.016
Nota musical	Mi 4	Si 3	Sol 3
f_{nom} (Hz) (65 cm)	329.6	246.9	196.0
f_{nom} (Hz) (51 cm)	420.1	314.7	249.8
f_d	420.0	320.0	250.0

(a) (b) (c)

Figura 20.- a) Cuerda primera. b) Cuerda segunda. c) Cuerda tercera

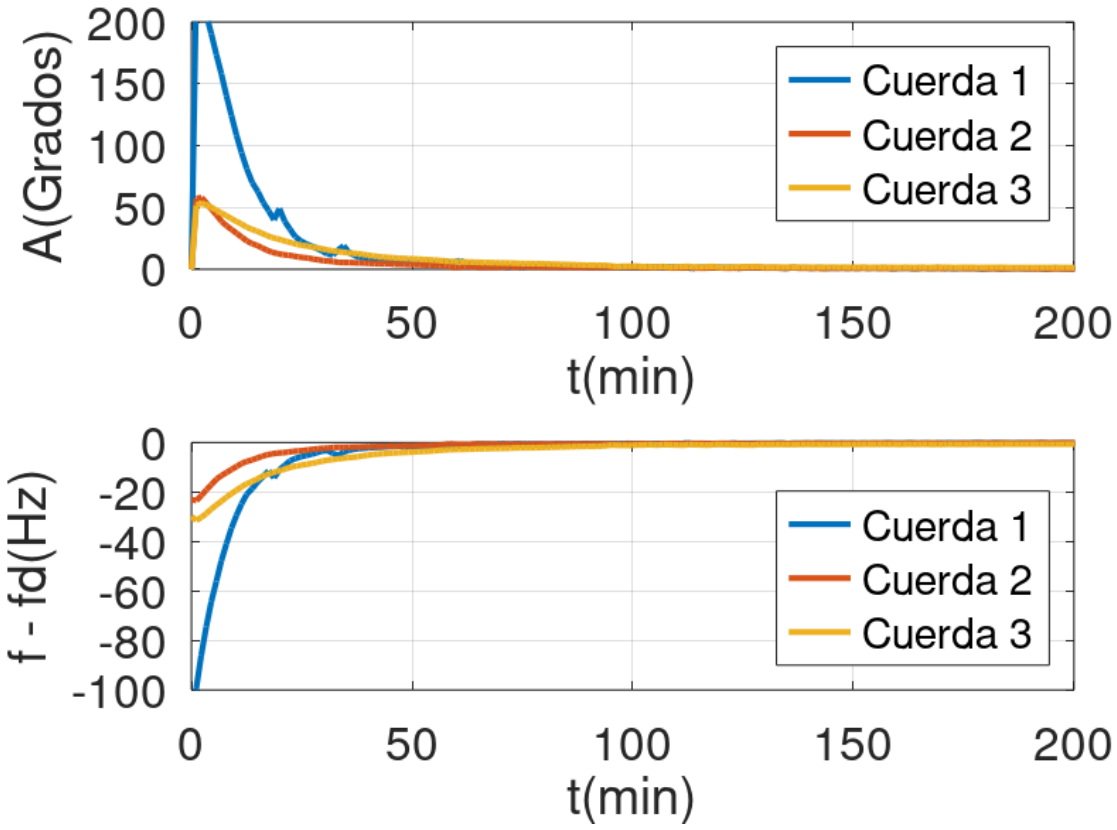
ANÁLISIS DE RESULTADOS

Después de realizar las mediciones y el procesamiento de los datos se obtuvieron los siguientes resultados.

Para empezar, tenemos gráficos que muestran la relación que existe entre la frecuencia fundamental de las cuerdas contra el ángulo de giro del cilindro del afinador y la longitud de cuerda retirada o aumentada (*dx*), todos en función del tiempo, lo cual nos permite

comprobar el tiempo que tarda en estabilizarse la cuerda en cada caso, y si después de eso, la cuerda mantiene la afinación constante o aún presenta variaciones considerables.

(a)



(b)

Figura 21.- (a) Giro del perno de afinación *A* en grados y (b) diferencia de frecuencia $f - f_d$ de las cuerdas 1, 2 y 3 en los primeros 200 minutos (3:20 horas) del experimento.

En esta gráfica podemos observar que después de que se llega a la afinación deseada, la variación ya no es considerable en lo que resta del experimento y la afinación se mantiene prácticamente constante. Adicional para tener un apoyo visual más claro, también se realizaron gráficas del diferencial de frecuencia $dF = f_{deseada} - f_{medida}$ con una escala en centésimas de semitono.

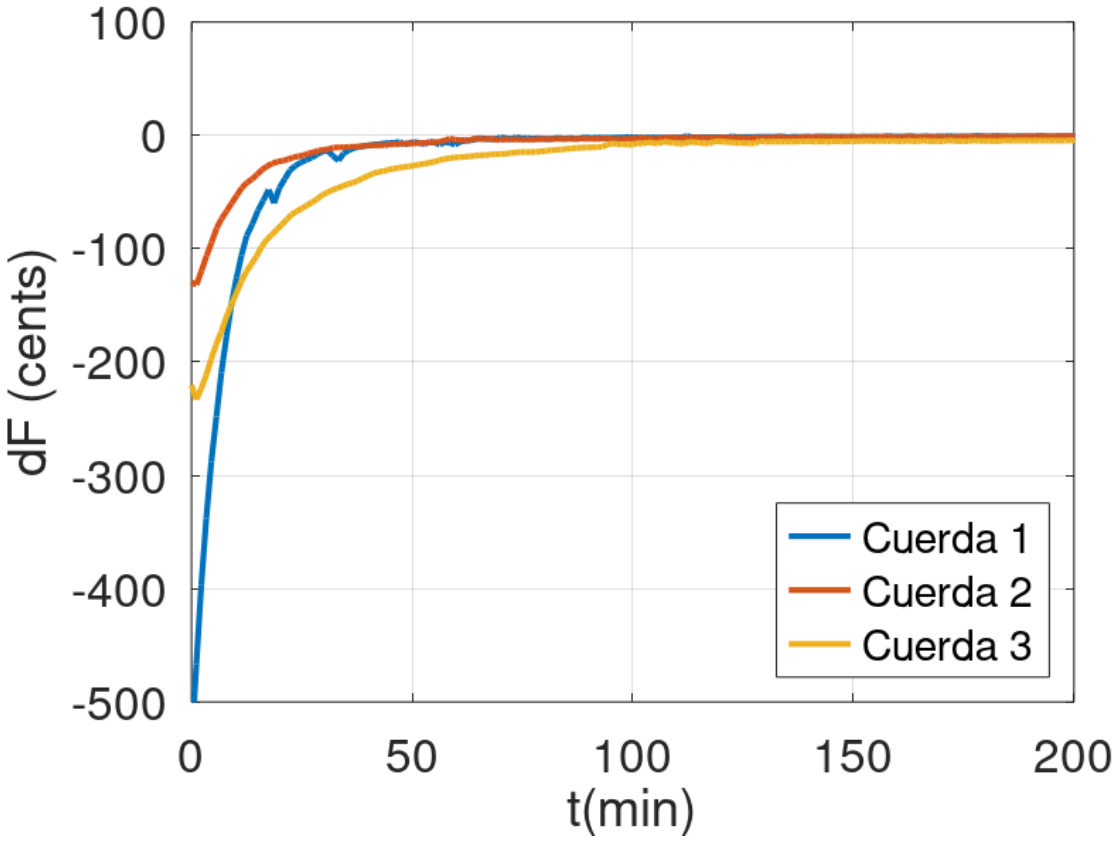


Figura 22.- Diferencia de afinación en cents (centésimas de semitono) respecto a la afinación deseada en los primeros 200 minutos (3:20 horas) del experimento.

De estas gráficas podemos notar que entre mayor es el calibre de la cuerda, mayor es el tiempo de ajuste que necesita para poder estabilizarse en la frecuencia a la que deseamos

que esté afinada. Mientras que en la siguiente gráfica podemos ver a detalle que la afinación no tiene un ajuste constante, sino que va teniendo pequeñas variaciones aunque ya se encuentre dentro del rango de precisión aceptada.

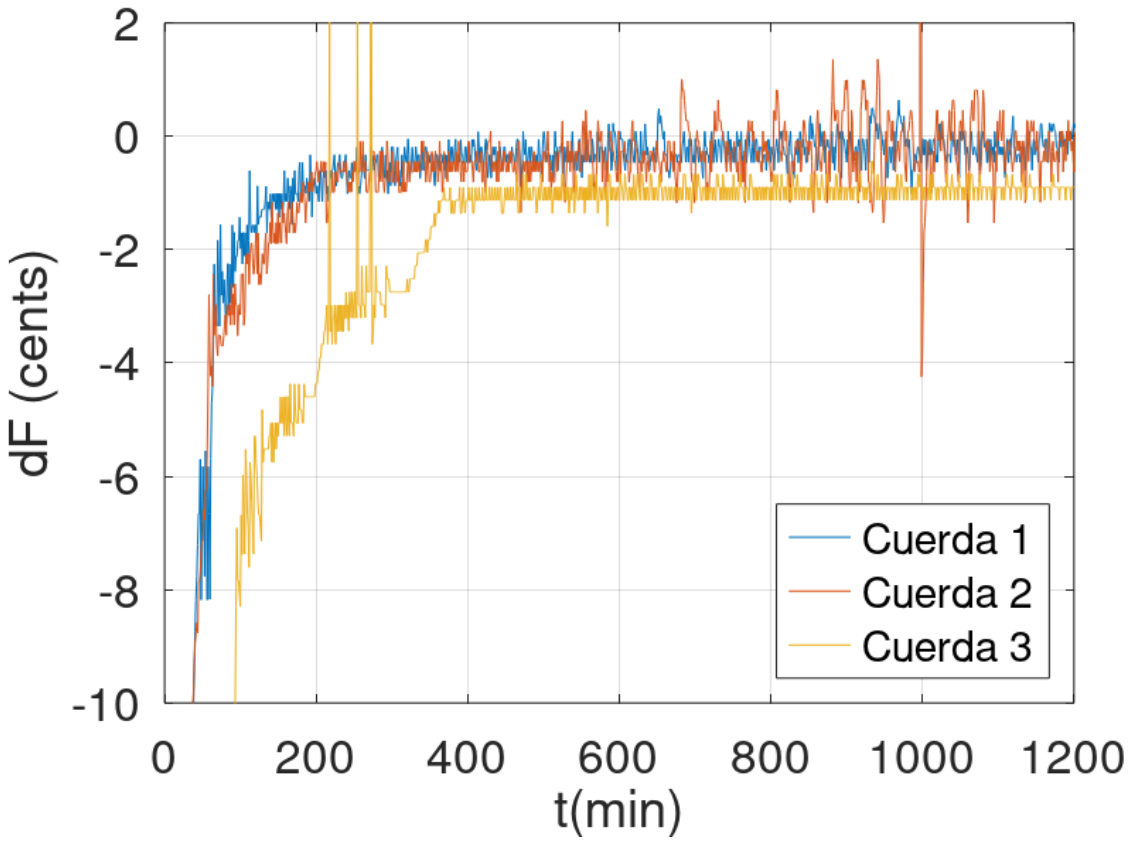


Figura 23.- Diferencia de afinación en cents (centésimas de semitono) respecto a la afinación deseada durante 1200 minutos (20 horas), escala vertical en detalle.

Como parte del análisis, un punto de interés es tratar de establecer una relación entre $\frac{f_{medida}}{f_{deseada}}$ y *dx/L* y observar si los comportamientos son similares en todas las cuerdas o

difieren.

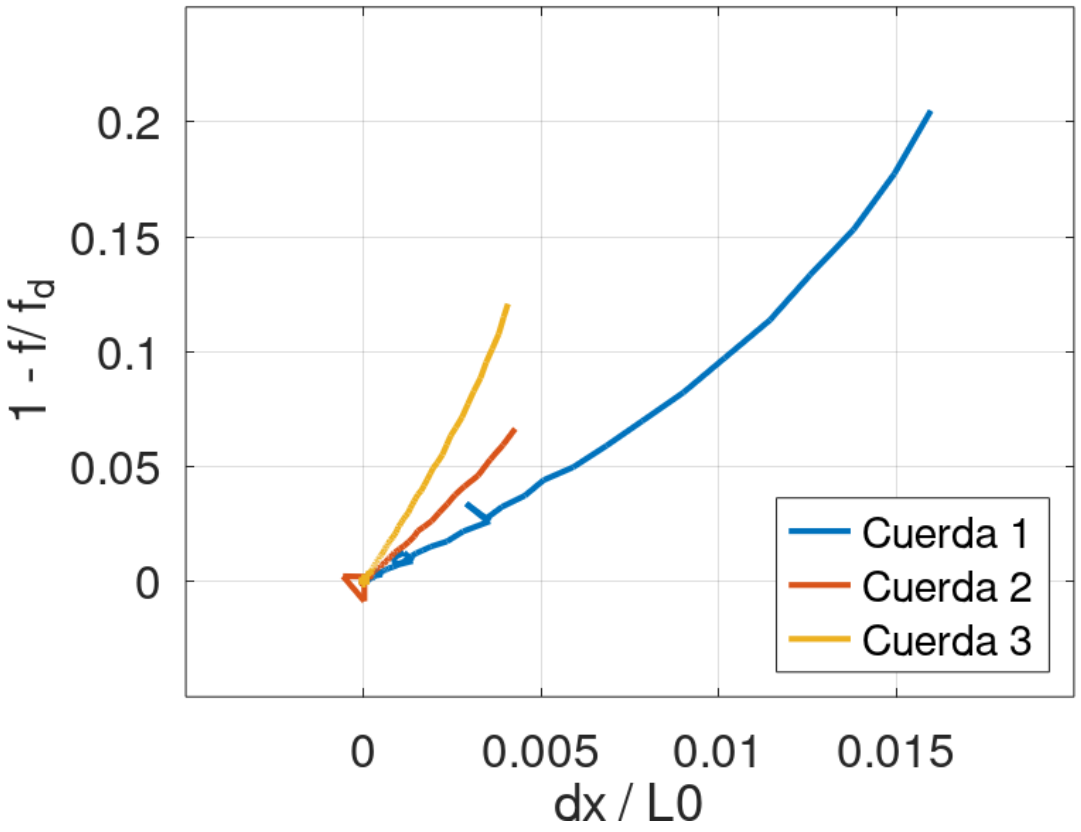


Figura 24.- Relación entre la diferencia de frecuencia relativa $1 - f / f_d$ y el estiramiento dx / L_0.

Como podemos observar, cerca del origen el comportamiento entre las variables tiende a una línea recta, con diferentes pendientes, por lo cual podríamos asociar los valores de la pendiente con el tiempo de estiramiento inicial de cada cuerda.

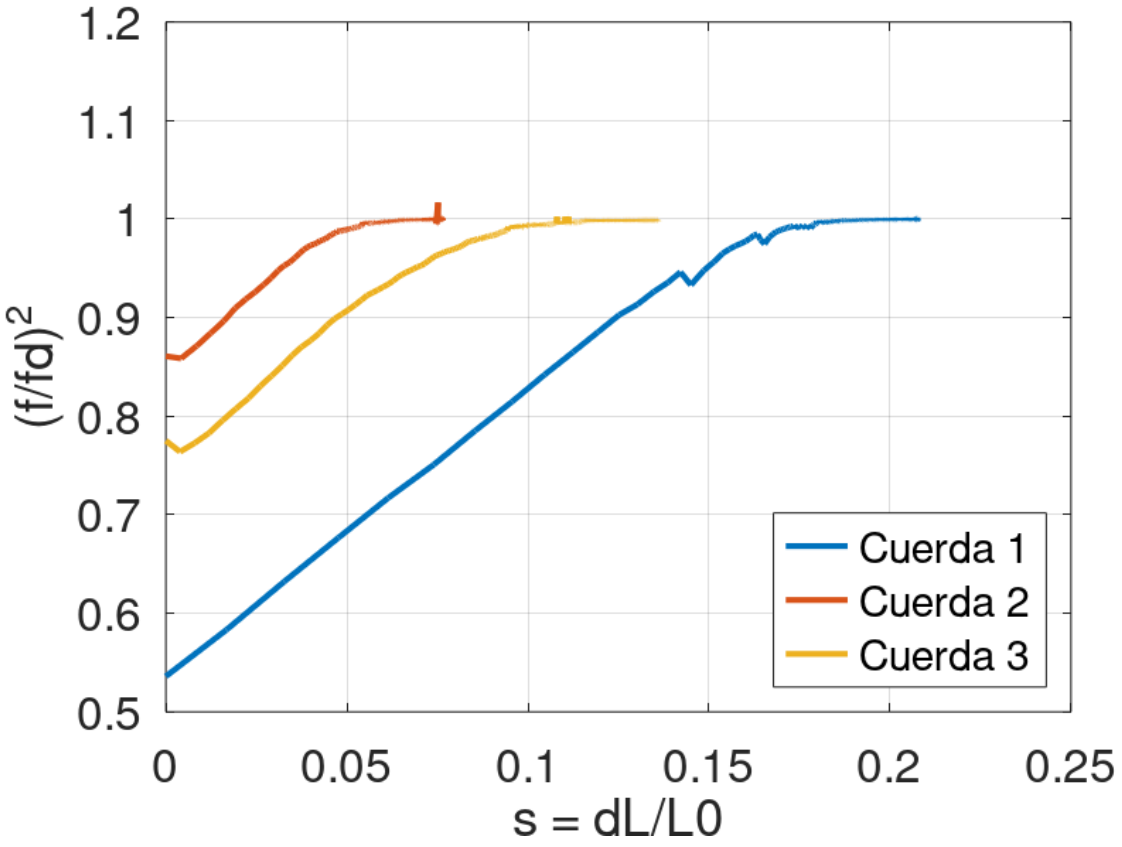


Figura 25.- Frecuencia relativa al cuadrado (f / fd)^2 vs. La deformación relativa: s = (L_{ext} – L_0) / L0.

La figura 25 muestra la frecuencia relativa al cuadrado $(\frac{f}{f_d})^2$ contra la deformación relativa $s = \frac{\Delta L}{L_0}$.

Un último parámetro de interés es el cambio en la elongación de cada una de las cuerdas en comparación con las demás.

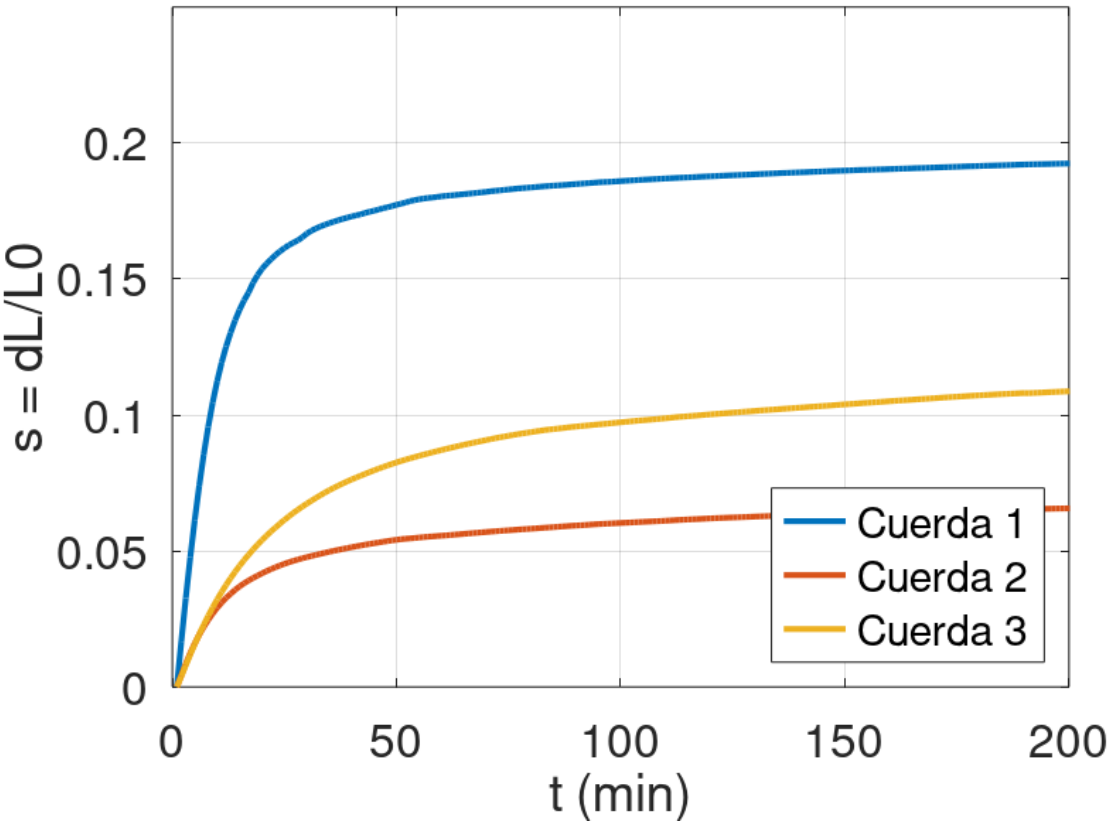


Figura 26.- Evolución inicial de la deformación relativa de las cuerdas: $s = (L\text{ext} - L_0) / L_0$.

Tabla 2. Descriptores del proceso de afinación de tres cuerdas de nylon: primera, segunda y tercera. Los valores terminales de afinación y tensión son el promedio de las 14 horas finales, sin considerar las 6 horas iniciales del experimento.

	Cuerda 1	Cuerda 2	Cuerda 3
Afinación terminal (cents)	-0.18	-0.18	0.94
Tensión terminal (newtons)	109.3	82.9	79.0
Estiramiento total (mm)	167.2	61.0	110.0
Deformación relativa a L_0	0.208	0.076	0.136
Factor lineal (pendiente) $dx/L_0 \ vs. 1 - f/f_d$	0.128	0.075	0.044
Factor lineal (pendiente) $s \ vs. (f/f_d)^2$	0.353	0.305	0.302

La Tabla 2 indica los descriptores del proceso de afinación de las tres cuerdas de Nylon: primera, segunda y tercera. Los valores terminales de afinación y tensión son el promedio de las 14 horas finales, sin considerar las 6 horas iniciales del experimento.

Como podemos observar de la Tabla 2 y de la ecuación E 5.9, la afinación terminal y la resolución frecuencial a la que llegamos con el parámetro de frecuencia de muestreo de 192 kHz que usamos, la precisión que obtuvimos en las mediciones no varía más allá de 2 cents, por lo que recuperando la información descrita en el capítulo 3, los humanos

promedio son capaces de percibir cambios de frecuencia alrededor de los 8 cents, por lo que se puede decir que el experimento cumple en cuestión de precisión.

Recuperando el trabajo de post-procesamiento del trabajo anterior de *Sistema de laboratorio para medir cuerdas de instrumentos musicales* obtuvimos también espectrogramas y espectros vibratorios de cada uno de los experimentos.

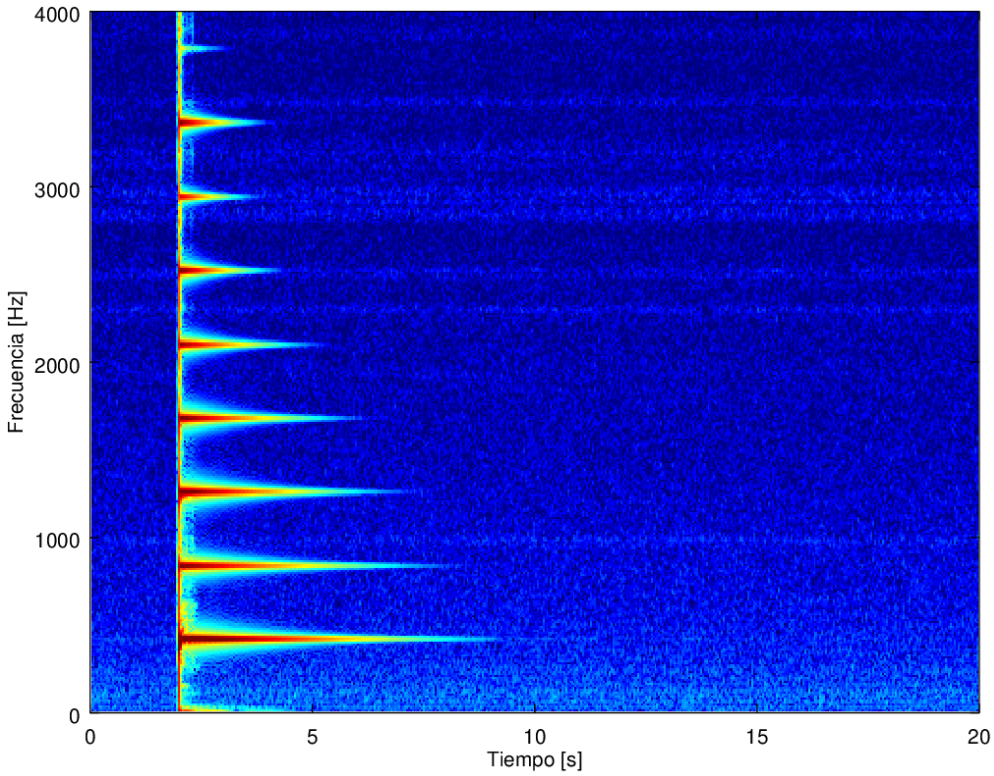


Figura 27.- Espectrograma de la cuerda primera en el minuto 401 del experimento.

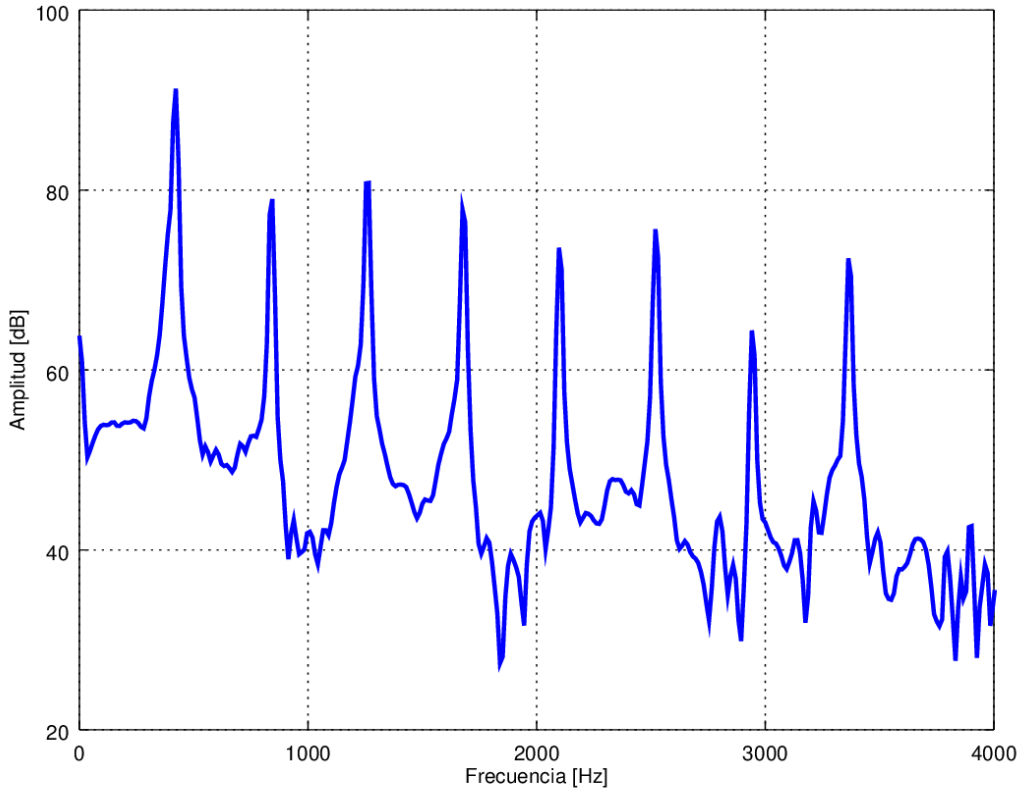


Figura 28.- Espectro vibratorio de la cuerda primera en el minuto 401 del experimento

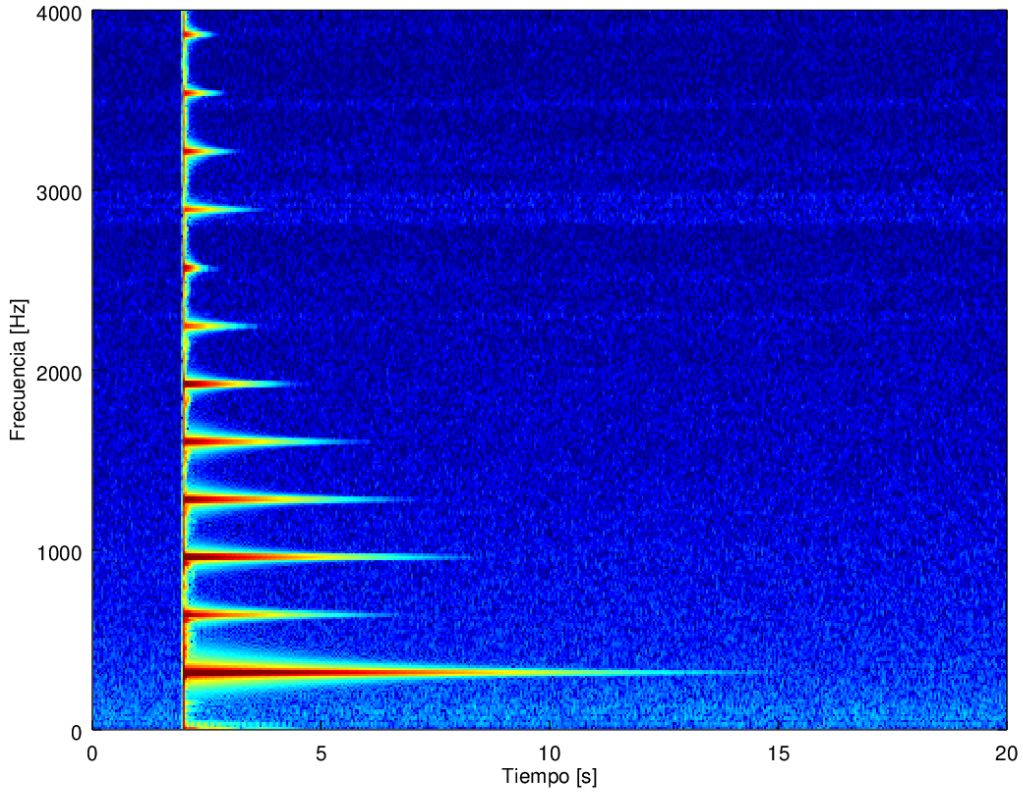


Figura 29.- Espectograma de la cuerda segunda en el minuto 403 del experimento.

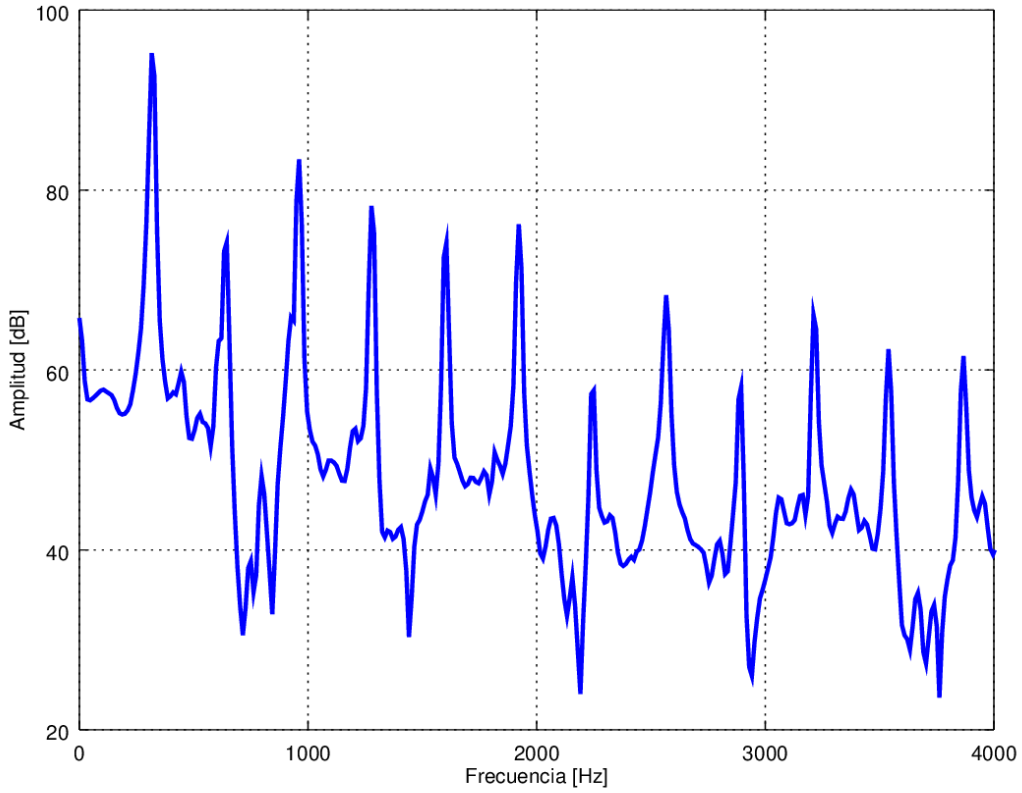


Figura 30.- Espectro vibratorio de la cuerda segunda en el minuto 403 del experimento

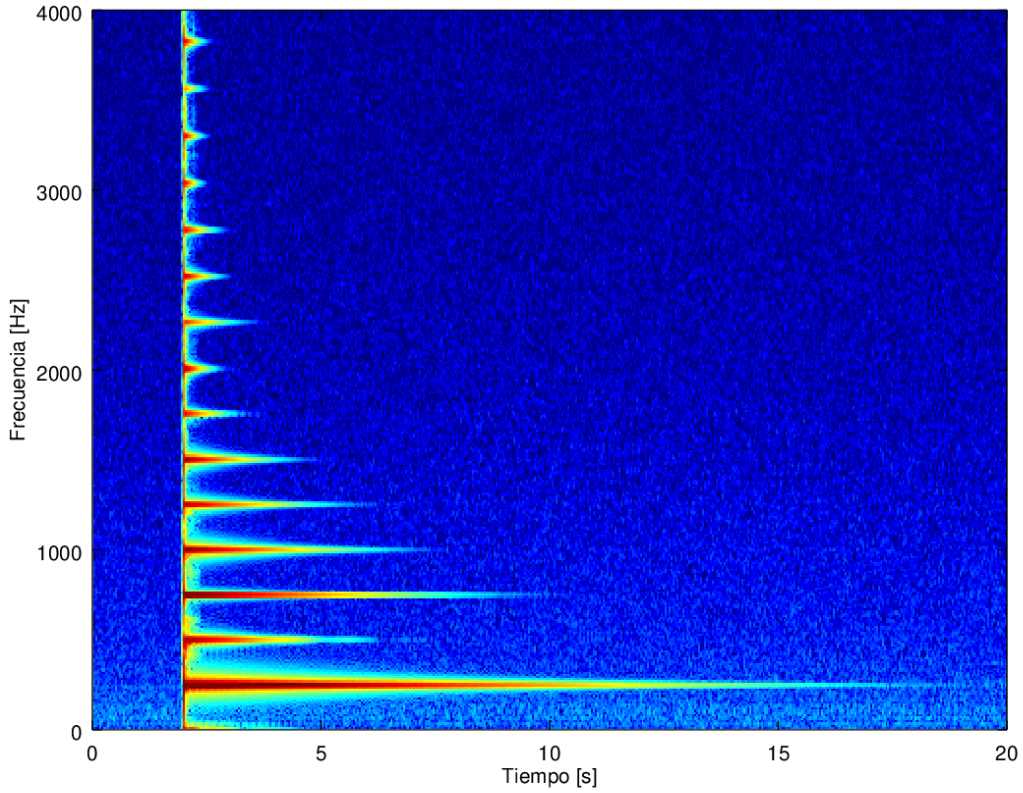


Figura 31.- Espectograma de la cuerda tercera en el minuto 404 del experimento.

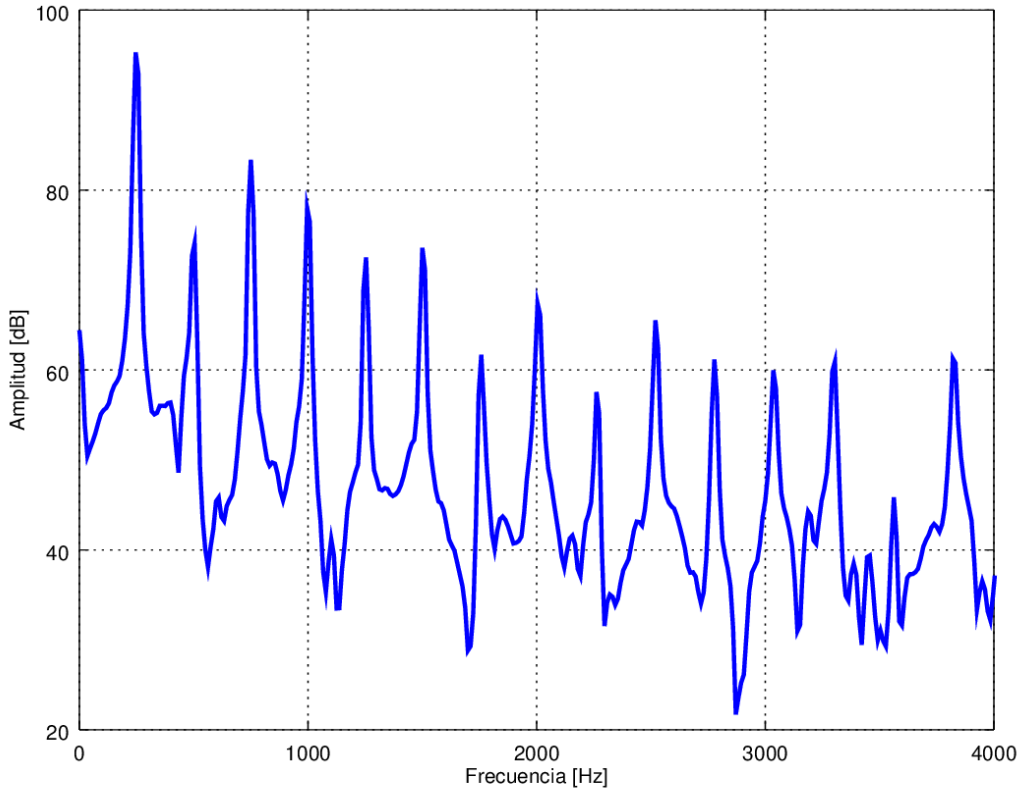


Figura 32.- Espectro vibratorio de la cuerda tercera en el minuto 404 del experimento

Se realizaron gráficas de temperatura y humedad para ver el efecto que pudieran o no tener dichos parámetros en el proceso de afinación de la cuerda, pero al hacer el análisis de los resultados, no se encuentra algo que pueda sugerir que sean variables considerables en el caso estudiado.

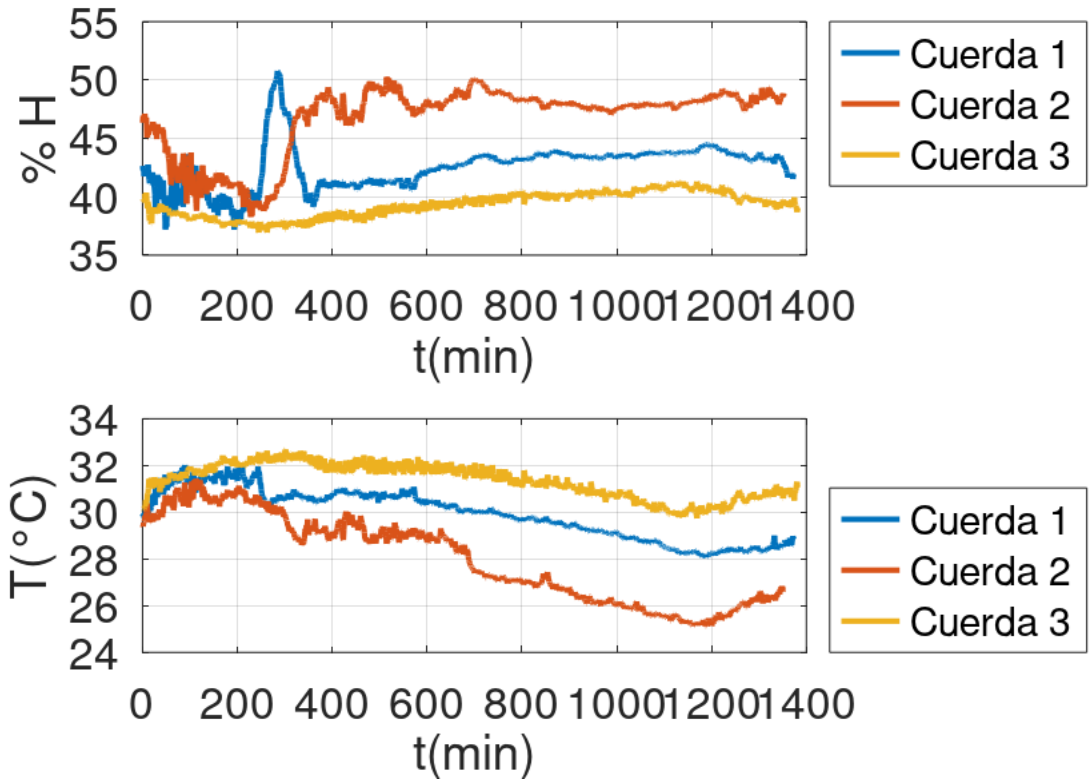


Figura 33.- Gráficas de temperatura y humedad de los tres experimentos

Evaluación

De las mediciones hechas podemos observar que las cuerdas pasan por un período de ajuste que depende del calibre de la cuerda, siendo a mayor grosor de calibre, mayor el tiempo que tarda en ajustarse a la frecuencia deseada.

7. CONCLUSIONES.

Este trabajo continuó con el estudio realizado por el grupo de Acústica y Vibraciones del ICAT, donde buscamos estudiar el comportamiento de las cuerdas de nylon en instrumentos musicales y este período de ajuste por el que pasan dichas cuerdas, ya que se considera un parámetro importante a saber tanto por los fabricantes como por los usuarios finales.

Para poder conocer mejor este proceso se desarrolló un sistema experimental de afinación automática que analiza la frecuencia fundamental de la cuerda y a partir de ese dato realiza los cálculos necesarios para mover un dispositivo que tensa o destensa la cuerda cambiando su frecuencia hasta llegar a la frecuencia deseada. Que como se pudo ver en la sección de resultados, el tiempo que tarda en estabilizarse la frecuencia fundamental en la cuerda, depende del calibre de la cuerda, ya que a mayor diámetro, es mayor el período de ajuste que necesita para que la frecuencia fundamental se mantenga estable. Como se puede ver en la parte de resultados, se piensa que dicha relación se puede asociar a una constante numérica de acuerdo con el calibre de la cuerda.

Como recomendaciones para trabajos a futuro, se propone estudiar otro tipo de cuerdas, de otros instrumentos, de diferentes materiales, cambiar la longitud vibrante. Para mejorar el sistema experimental se podría cambiar el microcontrolador y la computadora por equipos con mayor poder de procesamiento que permita acortar los tiempos de espera entre los procesos que se están realizando.

8. REFERENCIAS.

[1] Orduña F., Velasco R., Pérez A., Dorantes R., Pérez S. J. (2021). Sistema de laboratorio para medir cuerdas de instrumentos musicales. *SOMI XXXIV Congreso de Instrumentación.*

[2] Sirota, L., & Halevi, Y. (2013). Extended D'Alembert solution of finite length second order flexible structures with damped boundaries. *Mechanical Systems and Signal Processing, 39*(1–2), 47–58. https://doi.org/10.1016/j.ymssp.2012.01.006

[3] Sullivan, Charles R. "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback." Computer Music Journal 14, no. 3 (1990): 26–37. https://doi.org/10.2307/3679957.

[4] Karplus, K., & Strong, A. (1983). Digital Synthesis of Plucked-String and Drum Timbres. *Computer Music Journal*, *7*(2), 43–55. https://doi.org/10.2307/3680062

[5] Oxenham, Andrew J. 2013. "1 - The Perception of Musical Tones." The Psychology of Music, January, 1–33. doi:10.1016/B978-0-12-381460-9.00001-8.

[6] David Sulzer. 2021. *Music, Math, and Mind : The Physics and Neuroscience of Music*. New York: Columbia University Press.

[7] Roederer, Juan G. 2008. The Physics and Psychophysics of Music : An Introduction. 4th ed. Springer.

[8] Hagedorn, P., & DasGupta, A. (2007). *Vibrations and waves in continuous mechanical systems*, 69-112. John Wiley & Sons.

[9] Karjalainen, M., Välimäki, V., & Tolonen, T. (1998). Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and beyond. *Computer Music Journal*, *22*(3), 17–32. https://doi.org/10.2307/3681155

[10] Polievkt Perov, Nataliia Perova-Mello, & Walter H. Johnson. (2015). The physics of guitar string vibrations. *American Journal of Physics*, *84*, 38–43.

[11] Jared Rafferty, Adam R. V. Davenport, Robert D. Polak, & Andrew Fischer. (2018). Determining Young's modulus by measuring guitar string frequency. *The Physics Teacher*, *56*, 122–123.

[12] P. M. Vilela, R. A. Moscoso, *and* D. Thompson , "What every musician knows about viscoelastic behavior", American Journal of Physics 65, 1000-1003 (1997)https://doi.org/10.1119/1.18693

[13] Wheeler, G. F., & Crummett, W. P. (1987). The vibrating string controversy. *American Journal of Physics*, *55*(1), 33–37. https://doi.org/10.1119/1.15311

[14] Semeria, M. (2015). Los Tres Teoremas. Fourier - Nyquist - Shannon. *Documentos de Trabajo*, *582*, 1–14.

[15] Lynch-Aird, N., & Woodhouse, J. (2018). Frequency Measurement of Musical Instrument Strings Using Piezoelectric Transducers. *Vibration*, *1*(1), 3–19. MDPI AG. http://dx.doi.org/10.3390/vibration1010002

[16] PENTTINEN, H., & VÄLIMÄKI, V. (2004). A time-domain approach to estimating the plucking point of guitar tones obtained with an under-saddle pickup. *Musical Acoustics*, *65*(12), 1207–1220.

[17] Thomas Rossing. (2014). *Springer Handbook of Acoustics*. Vol. 2nd edition. Dordrecht: Springer.

[18] Broersen, Petrus M.T. 2006. Automatic Autocorrelation and Spectral Analysis. Springer.

[19] Webb, Robert H.. (1997). Elementary Wave Optics - 2.1 Physical Description. Dover Publications. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt00B0AUH4/elementary-wave-optics/physical-description

[20] Orduña, Felipe, Paz, Carlos, Pérez, Antonio, Velasco, Roberto. *"Control automatizado de la frecuencia de vibración en un sistema de laboratorio para medir cuerdas de*

instrumentos musicales". SOMI 36 Congreso de la Sociedad Mexicana de Instrumentación, 26 al 28 de octubre de 2022, Ciudad de México.

[21] Roadie 3 Manual. (s. f.). Roadie Music. Recuperado 25 de septiembre de 2022, de https://bandindustrieshelp.freshdesk.com/support/solutions/articles/67000670223-roadie-3-manual

[22] Downloads. (2022, 27 julio). guitar tuner professional Tronical. full automatic guitar tuner that fits almost any guitar like Fender, Gibson, Ibanez Yamaha. Recuperado 25 de septiembre de 2022, de https://www.tronicaltune.com/downloads/?v=796834e7a283

9. ANEXOS.

1.- Desarrollo de algoritmo de Karplus-Strong con afinación automática variando únicamente el valor de retardo.

```
clear all;
close all;
clc
f_i = 250; %Frecuencia inicial
f_d = 200; %Frecuencia deseada
fund = f_i; %Frecuencia fundamental
fs = 10000; %Frecuencia de muestreo
fmin = 50;
fmax = 1000;
D = round(fs/f_i) %Retardo
T = 2; %Duracion
R = 0.99; %Atenuacion
Amp = 0.9; %Ganancia
N = round(T*fs);
t = (0:N-1).'*1/fs; %Vector de tiempo
A = zeros(N,1);
del_f = fund^2 / fs %delta de frecuencia

while abs(fund - f_d) > del_f

    D = D - round((fs / fund^2) * (f_d - fund)) %Actualizacion de retardo
    A(1:D) = sin(2*pi*(fs/D)*t(1:D)); %Construccion del vector a sintetizar
```

```
    for n = D+1:N
        A(n) = R * A(n - D);
    end

    %Cálculo de frecuencia fundamental por correlación
    nmin = round(fs/fmax);
    nmax = round(fs/fmin);
    [c, A_Corr] = xcorr(A, nmax);
    [cmax, cind] = max(c (nmax+1+nmin:nmax+1+nmax));
    cind = cind + nmin - 1;
    fund = fs / cind

    %Cálculo del espectro
    [p,f] = pspectrum(A,fs);
    p = 20*log10(p);
end

figure;
plot(t*1000,A);
xlim([0 100]);
grid;
ylabel('Amplitud lineal', FontSize= 14)
xlabel('Tiempo (ms)', FontSize= 14)

figure;
plot(f, p);
xlim([0 2000])
grid;
xlabel('Frecuencia (Hz)', FontSize = 14)
ylabel('Magnitud (dB)', FontSize= 14)
```

```
sound(Amp*A, fs);
```

2.- Script de simulación de método de afinación automática en Matlab:

```
clear all
close all

clc
%Programa para ajustar la frecuencia de una cuerda de nylon de 50cm de
%longitud vibrante. Tomando en cuenta el sistema de afinación mecánico.

L = 0.5; %Londitud vibrante de la cuerda
rho_l = 7.47e-4; %Densidad lineal de la cuerda de nylon
%rho_l = 523.25e-3; %Densidad lineal de la cuerda de nylon
T_0 = 77.47; %Tensión inicial del la cuerda en Newtons
Y = 2.85e9; %Valor del módulo de Young del nylon
r =  4.55e-4; %Radio de la sección transversal de la cuerda
S = r^2 * pi;%Cálculo del área de la sección transversal de la cuerda
E = S*Y;
fs = 44100; %Frecuencia de muestreo
dur = 2; %Duracion del impulso
fmin = 50;
fmax = 1000;
T = T_0;
d_cil = 0.01; %Diametro del cilindro de afinación en metros
r_cil = d_cil/2; %Radio del cilindro de afinación en metros
theta_motor = 1.8; %Grados de giro del motor
```

```
N = round(dur*fs);

Amp = 0.7; %Valor de amplitud

t = (0:N-1).'*1/fs; %Vector de tiempo

v_0 = sqrt(T_0/rho_l); %Calculo de la velocidad inicial del sistema

R= 0.99; %Atenuación

f_medida = v_0 / (2*L); %Frecuencia de la cuerda en condiciones iniciales.

f_deseada = 150; %Frecuencia deseada en Hz %Frecuencia medida experimentalmente

D = round(fs/f_medida); %Calculo de retardo

A = zeros(N,1);

del_f = (f_deseada^2 / fs) * (2/3); %Precisión de afinación


counter = 0;


 while true


    A(1:D) = sin(2*pi*f_medida*t(1:D));

    for n = D+1:N

        A(n) = R * A(n - D);

    end


%Análisis del espectro

[p,f] = pspectrum(A,fs);


%Cálculo de frecuencia fundamental por correlación

nmin = round(fs/fmax);

nmax = round(fs/fmin);

[c, A_Corr] = xcorr(A, nmax);

[cmax, cind] = max(c (nmax+1+nmin:nmax+1+nmax));

cind = cind + nmin - 1;

f_medida = fs / cind
```

```
% Probar si la cuerda está afinada
if abs(f_medida - f_deseada) < del_f || counter > 1000
    break;
end

% Actualización de los parámetros
df = f_deseada - f_medida;
dx = (df/f_deseada)*((2*L*T)/(E+T)); %Longitud del arco del cilindro
while abs(dx) > L/100
    dx = dx/10;
end
theta_cil = (dx/r_cil)* (180/pi); %Angulo de giro del cilindro en grados
theta_per = 14 * theta_cil; %Grados de giro de perilla para obtener
afinación deseada

pasos_motor = round(theta_per/theta_motor)

dT = E * (dx/L);
T = T + dT;

drho = -rho_l * (dx/L);
rho_l = rho_l + drho;

v = sqrt(T/rho_l);
D = round(2*L*fs / v);
end
figure;
plot(t,A);
figure;
plot(f,p);
```

```
sound(Amp*A, fs);
```

3.- Script con método de afinación automática con los datos para los tres calibres de cuerdas estudiadas.

```
clear all
close all

clc


 if exist('OCTAVE_VERSION')
    pkg load signal
  end;

if exist('OCTAVE_VERSION')
    pkg load instrument-control
  end;

  if exist('OCTAVE_VERSION')
    pkg load arduino
  end;
%Programa para ajustar la frecuencia de una cuerda de nylon de 50cm de
%longitud vibrante. Tomando en cuenta el sistema de afinación mecánico.

data_dir = '/home/gav/results_real/auto-005';
audio_dir = [data_dir filesep 'audio-frames'];
mkdir(data_dir);
mkdir(audio_dir);
```

```
L = 0.51; %Longitud vibrante de la cuerda

L0 = 0.805; %Longitud efectiva de la cuerda

%d_cil = 0.194*0.0254; %Diametro interno del cilindro de afinación en metros

d_cil = 0.235*0.0254; %Diametro externo del cilindro de afinación en metros

cal_cuerda = 0.0280*0.0254; %Calibre de la cuerda 1ra

%cal_cuerda = 0.0320*0.0254; %Calibre de la cuerda 2da

%cal_cuerda = 0.040*0.0254; %Calibre de la cuerda 3ra

rho_v = 1500; %Densidad volumetrica en kg/m3 del nylon

Y = 2.85e9; %Valor del módulo de Young del nylon

f_deseada = 420; %Frecuencia deseada en Hz %Frecuencia medida experimentalmente para 1ra cuerda

%f_deseada = 320; %Frecuencia deseada en Hz %Frecuencia medida experimentalmente para 2da cuerda

%f_deseada = 250; %Frecuencia deseada en Hz %Frecuencia medida experimentalmente para 3da cuerda

mu = 0.125; %% Factor de adaptacion "lenta"


fmin = 50;

fmax = 1000;

fminmax = [100 1000]; %Frecuencias de corte de correlacion


d_cil = d_cil + cal_cuerda;

r_cil = d_cil / 2; %Radio del cilindro de afinación en metros

theta_motor = 360 / 200; %Grados de giro del motor

r =  cal_cuerda / 2; %Radio de la sección transversal de la cuerda

S = r^2 * pi;%Cálculo del área de la sección transversal de la cuerda

rho_l = rho_v * S; %Densidad lineal de la cuerda de nylon

E = S*Y;

fs = 192000; %Frecuencia de muestreo



%Inicio de control serial de arduinos
```

```
disp('Opening Arduino serial device ...');

fflush(stdout);

nano= serial('/dev/ttyACM0',9600);

pause(5); %% Pausa para comunicacion con el Arduino


%Parametros de grabacion de audio

PAUSE_GENERAL = 30;

PAUSE_TH = 2;

PAUSE_ST = 2;

PAUSE_ARD = 1;

PRE_RECORD = 1;

RECORD = 30;

PAUSE_REAL = PAUSE_GENERAL - (PAUSE_ARD + PAUSE_ARD + PAUSE_ST + PAUSE_TH);



del_f = (f_deseada^2 / fs); %Precisión de afinación

dx = 0;

theta_cil = 0;


%Creacion de archivo de temperatura y humedad


th_f = fopen([data_dir filesep 'th.csv'], 'a');

fprintf(th_f,'%s\n%s\n',    '#INFO:COLUMNS:    datetime:date    and    time
[%Y-%m-%d_%H:%M:%S],    Temp:temperature    [C],    Hum:relative    humidity
[%]','datetime,Temp,Hum');

fclose(th_f);


counter = 0;


for counter = 0:1199
```

```
    y = string_record(nano, PRE_RECORD, RECORD);
    timestamp = strrep(datestr(now,31),' ','_');
    tag = sprintf('_%06d', counter);
    audiowrite([audio_dir filesep 'lectura' tag '.wav'],y,fs);
    fclose(fopen([audio_dir filesep 'data' tag '.flag'],'w'));
    f_deseada
    f_medida = fundamental(y,fs,fminmax);
    S = 20*log10(abs(fft(mean(y,2))));
    N = length(S);
    f = (0:N-1).' * fs/N;
    f_medida = harmonic_peaks(f, S, 1, f_medida, 5* del_f)

      %Toma de lectura de temperatura y humedad
      pause(PAUSE_TH);
      srl_write(nano,['get:th']);
      pause(PAUSE_ARD);
      th = srl_read(nano, 13);
      th = char(th(1:11))
      th_f = fopen([data_dir filesep 'th.csv'], 'a');
      fprintf(th_f,'%s,%s\n',timestamp,th);
      fclose(th_f);
      % Actualización de los parámetros
      df = f_deseada - f_medida;
      T = rho_l * (2*L*f_medida)^2



      data_f = fopen([data_dir filesep 'data_file.dat'], 'a');
      fprintf(data_f, '%s %f %f %f %f\n',timestamp, f_medida, theta_cil, T,
dx);
      fclose(data_f);
```

```
% Probar si la cuerda está afinada
if abs(f_medida - f_deseada) < del_f || counter > 1000
    disp('Cuerda afinada');
    %break;
end;

dx = (df/f_deseada)*((2*L0*T)/(E+T)); %Longitud del arco del cilindro
while abs(dx) > L0/20
    disp('Reduciendo giro');
    dx = sign(dx) * L0/20;
end;
theta_cil = (dx/r_cil)* (180/pi); %Angulo de giro del cilindro en
grados
theta_per = 14 * theta_cil; %Grados de giro de perilla para obtener
afinación deseada
pasos_motor = round(mu*theta_per/theta_motor)
pause(PAUSE_ST);
%Activacion del motor
srl_write(nano,['set:steps:' int2str(pasos_motor)]);
pause(PAUSE_ARD);

if 0
dT = E * (dx/L);
T = T + dT;
drho = -rho_l * (dx/L);
rho_l = rho_l + drho;
v = sqrt(T/rho_l);
end;
counter = counter + 1
pause(PAUSE_REAL);
```

```
endfor;
```

4.- Script desarrollado en Octave para la grabación de archivos de audio en formato .wav.

```
function ...
  [y fs] = string_record(pulse_device, pulse_delay, ...
                         time_seconds, audio_dev_name, ...
                         input_ch, fs, nbits)
  %%[y fs] = string_record(pulse_device, pulse_delay, ...
  %%                       time_seconds, audio_dev_name, ...
  %%                       input_ch, fs, nbits)
  %%
  %%Record audio input from string test system in Acústica-ICAT.
  %%
  %%Accepts:
  %%pulse_device   Serial port of pulse device, default:
serial('/dev/ttyACM1',9600);
  %%pulse_delay    Delay in seconds before pulse on/off: default: [1 1].
  %%time_seconds   Recording time in seconds, default: 5.
  %%audio_dev_name Audio device name, default = 'Scarlett 4i4 USB: Audio
(hw:1,0) (ALSA)'.
  %%input_ch       Audio input channel, single or an array, default: [1
2].
  %%sampling_freq  Sampling rate per second, default = 48000.0.
  %%nbits          Bits per sample, default = 16.
  %%
  %%Returns:
  %%y              recorded audio input.
  %%fs             sampling rate per second.

  if exist('time_seconds') ~= 1
    time_seconds = 5;
  end;

  if exist('pulse_device') ~= 1
    pulse_device = serial('/dev/ttyACM0', 9600);
  end;

  if exist('pulse_delay') ~= 1
    pulse_delay = [2 2];
  end;

  if exist('audio_dev_name') ~= 1
    %%audio_dev_name = 'default (ALSA)';
    audio_dev_name = 'Scarlett 4i4 USB: Audio (hw:1,0) (ALSA)';
  end;

  if ischar(audio_dev_name)
    audio_dev_name = {audio_dev_name}; %% convert str to cell str
  end;

  if exist('input_ch') ~= 1
    input_ch = [1 2];
```

```
  end;

  if exist('fs') ~= 1
    fs = 192000;
  end;

  if exist('nbits') ~= 1
    nbits = 16;
  end;

  %% pulse_device = serial(pulse_device, 9600);
  num_inputs = length(input_ch);
  max_inputs = max(input_ch);
  input_dev = audiodevinfo(1, audio_dev_name{1});
  input_dev_name = audiodevinfo(1, input_dev)
  audio_in = audiorecorder(fs, nbits, max_inputs, input_dev);
  record(audio_in, time_seconds);
  fprintf('  string_record: %d seconds left\r', round(time_seconds));
  fflush(stdout);
  time_seconds = time_seconds - pulse_delay(1);
  pause(pulse_delay(1));
  srl_write(pulse_device,['set:pulse:1']); %% String pulse ON
  while isrecording(audio_in)
    fprintf('  string_record: %d seconds left\r', round(time_seconds));
    fflush(stdout);
    time_seconds = time_seconds - 1;
    pause(1);
  end;
  fprintf('  string_record: %d seconds left\r', round(time_seconds));
  fflush(stdout);
  fprintf('\n');
  stop(audio_in);
  y = getaudiodata(audio_in);
  y = y(:, input_ch);
  input_peak_values = max(abs(y));
  disp('Input peak values [%]:');
  disp(input_peak_values * 100);
  pause(pulse_delay(end));
  srl_write(pulse_device, ['set:pulse:0']); %% String pulse OFF

%%Felipe Orduña <felipe.orduna@ccadet.unam.mx>

%% Local Variables:
%% mode: octave
%% indent-tabs-mode: nil
%% End:
```

5.- Script desarrollado en Octave para la determinación de frecuencia fundamental a través del método de autocorrelación.

```
function ...
    F0 = fundamental(y,fs,fminmax)
  %%
  %%Estimate the fundamental frequency F0 in hertz of time signal y,
  %%downmixed to mono if necessary.  fs, sampling rate per second.
  %%fminmax, optional fundamental frequency range, default: [100 1000].

  if exist('OCTAVE_VERSION')
    pkg load signal;
  end;

  if exist('fminmax') ~= 1
    fminmax = [100 1000];
  end;

  fmin = fminmax(1);
  fmax = fminmax(2);
  N = length(y);
  minlag = 1 + floor(fs/fmax);
  maxlag = 1 + ceil(fs/fmin);
  tau_ms = (-maxlag:maxlag).'*1000/fs;
  tau_2 = (0:maxlag).'*1000/fs;
  y1_corr = xcorr(mean(y,2), maxlag, 'coeff');
  figure;
  plot(tau_ms,y1_corr);
  xlabel('t(ms)', 'fontsize',20);
  ylabel('Auto correlacion','fontsize',20);
  y_corr = ifftshift(y1_corr); %% Start at zero lag
```

```
  figure;
  plot(tau_2, y_corr(1:maxlag+1));
  xlabel('t(ms)', 'fontsize',20);
  ylabel('Auto correlacion','fontsize',20);
  [~,bestlag] = max(y_corr(minlag:maxlag));
  bestlag = bestlag + minlag - 1;
  lags = [bestlag-1 bestlag bestlag+1].';
  bestlag = sum(lags.*y_corr(lags)) / sum(y_corr(lags));
  F0 = fs / (bestlag - 1);

%% Felipe Orduña <felipe.orduna@icat.unam.mx>
%% [http://www.academicos.ccadet.unam.mx/felipe.orduna]

%% Local Variables:
%% mode: octave
%% comment-column: 0
%% indent-tabs-mode: nil
%% End:
```

6.- Script para la estimación de la frecuencia fundamental con el uso del espectro.

```
function ...
  [f1 centroid weight] = harmonic_peaks(f, S, N_harm, f0, df)
% [f1 centroid weight] = harmonic_peaks(f, S, N_harm, f0, df)

  N = length(f);
%  N_harm = 8;

  if exist('df') ~= 1
    df = f0;
```

```
  end;
  if length(df) == 1
    bw = df * ones(N_harm, 1);
  end;

  MAX_ITER = 100;
  %%while 1
  for iter=1:MAX_ITER
    f_harm = f0 * (1:N_harm)';
    weight = zeros(N_harm,1);
    centroid = zeros(N_harm,1);
    if ischar(df)
      if strcmp(df, 'ERB') % Equivalent Rectangular Bandwidth
        bw = 24.7 * (0.00437 * f_harm + 1);
      elseif df(end) == '%' % Percent relative bandwidth
        bw = str2num(df(1:end-1)) * f_harm;
      end;
    end;
    for n=1:N_harm
      n1 = round(1 + (N - 1) * (f_harm(n) - bw(n)/2 - f(1)) / (f(N) -
f(1)));
      n2 = round(1 + (N - 1) * (f_harm(n) + bw(n)/2 - f(1)) / (f(N) -
f(1)));
      %  window = ones(n2 - n1 + 1, 1);
      window = hanning(n2 - n1 + 1);
      S2 = window .* S(n1:n2) / (n2 - n1 + 1);
      weight(n) = sum(S2);
      centroid(n) = sum(S2 .* f(n1:n2)) / weight(n);
      centroid(n) = centroid(n) / n;
    end
    f1 = sum(weight .* centroid) / sum(weight);
```

```
    if abs(1 - f0/f1) < 1e-10 %% eps
      break;
    end;
    f0 = f1;
  end;
  if iter == MAX_ITER
    warning(['Sin convergencia con MAX_ITER=' int2str(MAX_ITER)]);
  end;

%Felipe Orduña <felipe.orduna@icat.unam.mx>
%[http://www.academicos.ccadet.unam.mx/felipe.orduna]

% Local Variables:
% mode: octave
% indent-tabs-mode: nil
% End:
```

7.- Programa de post procesamiento de datos obtenidos de las mediciones.

```
function ...
  %%smv(prefix, wait_for_measurement, frame_period)
  smv(prefix, wait_for_measurement, frame_period, fminmax)
%%smv(prefix, wait_for_measurement, frame_period)
%%smv.m - make spectrum/spectrogam movies of repeatedly plucked string.
%%
%%prefix indicates the working directory containing recorded WAV files
%%named 'lectura_%06d.wav' under the subdirectory prefix/audio-frames.
%%If working in the current directory, use a single dot: prefix = '.';
```

```
%%
%%wait_for_measurement[=3] wait these frame_periods for new experimental
%%data to become available before timing out.
%%
%%frame_period[=60 s] experimental frame period in seconds.

%%clear all;
%%close all;

%% Disable pager:
more off

if exist('OCTAVE_VERSION') == 5
  pkg load signal
end;

%%prefix = '000066'; % Work below this directory

if exist('prefix') ~= 1 % No prefix given, so ...
  prefix = '.'; % use the current working directory.
end;

if exist('wait_for_measurement') ~= 1 %% Concurrent with experiment
  wait_for_measurement = 3; %% Wait 3 frames for new experimental data.
end;

if exist('frame_period') ~= 1
  frame_period = 60; %% in seconds
end;
```

```
if exist('fminmax') ~= 1
  fminmax = [100 1000]; %% in hertz
end;

% Signal analysis parameters
block_size = 4096;
step_size = 2048;

% Plot limits
min_time = 0;
max_time = 20; % seconds
min_freq = 0;
max_freq = 4000; % hertz
min_dB = 20;
max_dB = 100; % decibels

% Harmonic search
N_harm = 10;
max_harm_index = round(block_size / 4);

%set(0, 'defaultfigurevisible', 'off');
audio_prefix = [prefix filesep 'audio-frames'];
data_prefix = [prefix filesep 'data-frames'];
movie_prefix = [prefix filesep 'movie-frames'];
mkdir(data_prefix);
mkdir(movie_prefix);

%%N = length(glob([audio_prefix filesep 'lectura_*.wav']));
%%frame_time = (0:N-1).' * frame_period;
%%for n=1:N
```

```
n = 0;
omit = false;
while 1
  n = n + 1;
%  filename = [prefix filesep 'lectura_' int2str(n-1) '.wav'];
  framename = sprintf('lectura_%06d.wav', n-1);
  filename = [audio_prefix filesep framename];
  % Check if this frame is already processed
  pp1_flag = [data_prefix filesep sprintf('pp1_%06d.flag',n-1)];
  % Some further processing needs at least one of this cases to be
processed.
  %%if ( isfile(pp1_flag) && n-1 ~= 0 )
  if !isempty(stat(pp1_flag)) %% && n-1 ~= 0 )
    %%disp(['Omitiendo ' num2str(n-1)]);
    fprintf('Omitiendo %d\r', n-1);
    omit = true;
    continue
  end
  if omit
    fprintf('\n');
    omit = false;
  end;
  %% break;
  % Loop until reading is possible
  data_flag = [audio_prefix filesep sprintf('data_%06d.flag',n-1)];
  data_read = false;
  waiting = 0;
  while !data_read
    %%if isfile(data_flag)
    if !isempty(stat(data_flag))
      [y,fs] = audioread(filename);
```

```
    nbits = audioinfo(filename).BitsPerSample;
    data_read = true;
  elseif wait_for_measurement > 0
    disp(['Esperando ' data_flag]);
    pause(frame_period);
    waiting = waiting + 1;
  end;
  if waiting >= wait_for_measurement
    break;
  end;
end
if !data_read
  if wait_for_measurement > 0
    warning(['No disponible ' data_flag]);
  end;
  break;
end;
disp(['Analizando ' num2str(n-1)])
scale = 2^nbits;
y = y * scale;
channels = size(y, 2);
for k=1:channels
  % Calculate spectrum and spectrogram
  [Y f YY t] = simple_spectrum(y(:,k), fs, block_size, step_size);
  f_search = f(1:max_harm_index);
  Y_search = Y(1:max_harm_index);
  %%f0 = f0_guess(f_search, Y_search); % Simple guess
  %%f0 = f0_guess(f_search, Y_search, N_harm); % Fancy guess
  f0 = fundamental(y,fs,fminmax);
  [ f1 harmonic_freqs harmonic_levels ] = harmonic_peaks(f_search, ...
```

```
                                        Y_search, N_harm, f0);
    if (n > 1)
      eval(['harmonic_freqs' int2str(k), ...
        '=load([data_prefix    filesep    ''harmonic_freqs'    int2str(k)
'.dat'']);']);
      eval(['harmonic_levels' int2str(k), ...
        '=load([data_prefix    filesep    ''harmonic_levels'    int2str(k)
'.dat'']);']);
    end;
    eval(['harmonic_freqs' int2str(k) '(n,:) = harmonic_freqs.'';']);
    eval(['harmonic_levels'          int2str(k)          '(n,:)          =
10*log10(harmonic_levels).'';']);
    save('-ascii', ...
      [data_prefix filesep 'harmonic_freqs' int2str(k) '.dat'], ...
      ['harmonic_freqs' int2str(k)]);
    save('-ascii', ...
      [data_prefix filesep 'harmonic_levels' int2str(k) '.dat'], ...
      ['harmonic_levels' int2str(k)]);
    %% Save frequency axis
    if (n == 1) && (k == 1)
      framename = 'frequency_axis.dat';
      save('-ascii', [data_prefix filesep framename], 'f');
    end;
    %% Save spectrum
    framename = sprintf('spectrum%d-%06d.dat', k, n - 1);
    save('-ascii', [data_prefix filesep framename], 'Y');
    % Plot spectrum
    figure;
    plot(f, 10*log10(Y), 'linewidth', 2); grid;
    axis([min_freq max_freq min_dB max_dB]);
    xlabel('Frecuencia [Hz]');
    ylabel('Amplitud [dB]');
```

```
    framename = sprintf('spectrum%d-%06d.png', k, n - 1);
    print([movie_prefix filesep framename], '-dpng');
    pause(1); %% Wait a little for graphic file to be produced...
    close;
    % Plot spectrogram
    figure;
    colorscale = size(colormap,1) / (20*log10(scale));
    image(t,f,10*log10(YY) * colorscale);
    set(gca,'ydir','normal');
    axis([min_time max_time min_freq max_freq]);
    xlabel('Tiempo [s]');
    ylabel('Frecuencia [Hz]');
    framename = sprintf('spectrogram%d-%06d.png', k, n - 1);
    print([movie_prefix filesep framename], '-dpng');
    pause(1); %% Wait a little for graphic file to be produced...
    close;
  end;
  fclose(fopen(pp1_flag, 'w'));
end;
%set(0, 'defaultfigurevisible', 'on');

for n=1:N_harm
  harm_tag{n} = [ 'F' int2str(n) ];
end;

%%timestamp = datestr(now(), 'yyyy-mm-dd_HHMMSS-');
timestamp = '';

if exist('channels') ~= 1
  channels = ...
```

```
  audioinfo([audio_prefix filesep 'lectura_000000.wav']).NumChannels;
end;


for k=1:channels
  tag = int2str(k);
%%  if 0
  moviename = [ prefix filesep timestamp 'spectrum' tag '.mpg' ];
  system(['ffmpeg -pix_fmt yuv420p -y ' ...
          '-i ' movie_prefix filesep 'spectrum' tag '-%06d.png ' ...
          moviename]);
  moviename = [ prefix filesep timestamp 'spectrogram' tag '.mpg' ];
  system(['ffmpeg -pix_fmt yuv420p -y ' ...
          '-i ' movie_prefix filesep 'spectrogram' tag '-%06d.png ' ...
          moviename]);
%%  end; %% if 0
  % Reload harmonic harmonic_freqss and applitudes
  framename = 'frequency_axis.dat';
  f = load([data_prefix filesep framename]);
  eval(['harmonic_freqs' int2str(k), ...
    '=load([data_prefix     filesep     ''harmonic_freqs'     int2str(k)
'.dat'']);']);
  eval(['harmonic_levels' int2str(k), ...
    '=load([data_prefix     filesep     ''harmonic_levels'     int2str(k)
'.dat'']);']);
  numframes = size(harmonic_freqs1,1);
  frame_time = (0:numframes-1).' * frame_period;
  %%
  figure;
  plot(frame_time/60, eval(['harmonic_freqs' tag]), 'linewidth', 2); grid;
  xlabel('Tiempo [min]');
  ylabel('Frecuencia [Hz]');
```

```
  legend(harm_tag);

  print([ prefix filesep timestamp 'harmonic_freqs' tag '.png' ], '-dpng');

  close;

  figure;

  plot(frame_time/60,  eval(['harmonic_levels'  tag]),  'linewidth',  2);
grid;

  xlabel('Tiempo [min]');

  ylabel('Amplitud [dB]');

  legend(harm_tag);

  print([  prefix  filesep  timestamp  'harmonic_levels'  tag  '.png'  ],  '-
dpng');

  close;

end;


return;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Felipe Orduña <felipe.orduna@icat.unam.mx>

%[http://www.academicos.ccadet.unam.mx/felipe.orduna]


% Local Variables:

% mode: octave

% indent-tabs-mode: nil

% End:
```

8.- Programa en IDE de Arduino para control de partes mecánicas y sensor de temperatura y humedad

```
#include <Wire.h>
#include <string.h>

#define HDC 0x40    //B1000000 MSB
#define ratio 65536 //2^16

#define HIT 2
#define RECOVER 3

// Measurement
uint16_t humrawvalue;  //Raw data value, MS byte = valmsb, LS byte = vallsb
uint16_t temrawvalue;  //Raw data value, MS byte = valmsb, LS byte = vallsb
uint8_t humvalmsb, humvallsb;  //Raw 8-bit byte of humidity
uint8_t temvalmsb, temvallsb;  //Raw 8-bit byte of temperature
float humidity, temperature;

int PUL=7; //Pin para la señal de pulso
int DIR=6; //define Direction pin
int EN=5; //define Enable Pin

// State
int verbose_mode = 0;
String mode   = "ondemand";
float  period = 10;
int  steps = 0; // Solicitud de pasos del motor
long   timer  = 0;
int state = 0; // Estado del actuador

// Serial communication
```

```
String serial_input = "";
String si_action = "";
String si_var = "";
String si_val = "";

// Valores de pulsador
//String data;
float pulse;

void setup() {
  Serial.begin(9600);
  while (!Serial) {;} // wait for serial to connect
  Wire.begin();
  delay(5000);
  if (verbose_mode) {
    Serial.println("#=== Temperature and Relative Humidity Sensor ===");
    Serial.println("#INFO:COLUMNS:  Temp:temperature  [C],  Hum:relative
humidity [%]");
    Serial.println("Temp,Hum");
  }
  timer = millis();

  pinMode (PUL, OUTPUT);
  pinMode (DIR, OUTPUT);
  pinMode (EN, OUTPUT);
  digitalWrite(EN,HIGH);

  pinMode (HIT, OUTPUT);
  pinMode (RECOVER, OUTPUT);
  digitalWrite (HIT, LOW);
  digitalWrite (RECOVER, HIGH);
```

```
}

void loop() {

  char buffer[100];

  if ( Serial.available() > 0 ) {
    serial_input = Serial.readString();
    serial_input.trim();

    // TODO: disregard upper/lower case in messages.

    // Parse serial input
    si_action = "";
    si_var = "";
    si_val = "";
    char sz[100];
    serial_input.toCharArray(sz, 100);
    char *p = sz;
    char *str;
    int c = 0;
    while ((str = strtok_r(p, ":", &p)) != NULL) {
      if ( c==0 ) {
        si_action = str;
      } else if (c==1) {
        si_var = str;
      } else if (c==2) {
        si_val = str;
      }
      c++;
```

```
    }


    if ( si_action == "set" ) {
      if ( si_var == "mode" ) {
        if ( si_val == "periodic" ||
             si_val == "ondemand" ) {
          mode = si_val;
          if (verbose_mode) {
            Serial.println("#INFO: mode set to " + mode);
          }
        } else {
          Serial.println("#ERROR: invalid mode " + si_val);
        }
      } else if (si_var == "period" ) {
        //char buffer[10];
        si_val.toCharArray(buffer, 10);
        period = atof(buffer);
        if (verbose_mode) {
          dtostrf(period, 4, 3, buffer);
          Serial.println("#INFO: period set to " + String(buffer) + "s");
        }
      } else if (si_var == "steps" ) {
        //char buffer[10];
        si_val.toCharArray(buffer, 10);
        steps = atoi(buffer);
        if (verbose_mode) {
          //dtostrf(steps, 4, 3, buffer);
          sprintf(buffer, "%d", steps);
          Serial.println("#INFO: motor set to " + String(buffer) + "
steps");
        }
```

```
      if (steps != 0)
      {
        drive_motor();
      }
    } else if (si_var == "pulse") {
      //char buffer[10];
      si_val.toCharArray(buffer, 10);
      pulse = atof(buffer);
      if (verbose_mode) {
        dtostrf(pulse, 4, 3, buffer);
        Serial.println("#INFO: pulse set to " + String(buffer));
      }
      actuator();
    } else if (si_var == "verbose") {
      si_val.toCharArray(buffer, 10);
      verbose_mode = atoi(buffer);
      if (verbose_mode) {
        //dtostrf(pulse, 4, 3, buffer);
        sprintf(buffer, "%d", verbose_mode);
        Serial.println("#INFO: verbose set to " + String(buffer));
      }
    } else {
      Serial.println("#ERROR: invalid var " + si_var);
    }
  } else if ( si_action == "get" ) {
    //if ( mode == "ondemand" && si_var == "th" ) {
    if ( si_var == "th" ) {
      get_th();
    } else if ( si_var == "mode" ) {
      Serial.println(mode);
```

```
      } else if ( si_var == "period" ) {
        Serial.println(period);
      } else if ( si_var == "name" ) {
        Serial.println("controller");
      } else if ( si_var == "verbose" ) {
        Serial.println(verbose_mode);
      } else {
        Serial.println("#ERROR: invalid var " + si_var);
      }
    } else {
      Serial.println("#ERROR: invalid action " + si_action);
    }
    if ( mode == "periodic" && millis() - timer >= int(period*1000) ) {
      get_th();
      timer = millis();
    }
  }
}


void get_th() {
  Wire.beginTransmission(HDC);
  Wire.write(0x00);
  Wire.endTransmission(true);
  delay(200);

  Wire.requestFrom((uint8_t)HDC,(size_t)4,true); //It asks for 4 bytes, 2
bytes per register

  if(Wire.available()){

    temvalmsb = Wire.read();
```

```
    temvallsb = Wire.read();
    humvalmsb = Wire.read();
    humvallsb = Wire.read();

    //TEMPERATURE
    temrawvalue = temvalmsb <<8| temvallsb; //Locates each byte in its
proper place
    temperature = ((float)temrawvalue/(float)ratio)*165.0-40.0;

    //HUMIDITY
    humrawvalue = humvalmsb <<8| humvallsb;
    humidity = ((float)humrawvalue/(float)ratio)*100.0;

    char buffer1[10];
    char buffer2[10];

    dtostrf(temperature, 6, 2, buffer1);
    dtostrf(humidity, 6, 2, buffer2);

    String str1 = String(buffer1);
    String str2 = String(buffer2);

    str1.trim();
    str2.trim();

    Serial.println(str1 + "," + str2);
  }
}

void drive_motor () {
  // Ejecutar pasos según variable steps
```

```
  if (steps > 0)
  {
    digitalWrite(DIR,LOW);
    for (int i=0; i<steps; i++)    //Forward N steps
    {
      digitalWrite(PUL,HIGH);
      delayMicroseconds(400);
      digitalWrite(PUL,LOW);
      delayMicroseconds(400);
      //delay(100);
    }
  }
  else
  {
    steps = -steps; // Toma valor positivo de steps
    digitalWrite(DIR,HIGH);
    for (int i=0; i<steps; i++)   //Backward N steps
    {
      digitalWrite(PUL,HIGH);
      delayMicroseconds(400);
      digitalWrite(PUL,LOW);
      delayMicroseconds(400);
    }
  }
}

void actuator () {

    delay(abs(pulse));
    if (pulse > 0) {
```

```
      if (verbose_mode) {
        Serial.println("Hit");
      }
      digitalWrite (HIT, HIGH);
      delay(100);
      digitalWrite (HIT, LOW);
    } else {
      if (verbose_mode) {
        Serial.println("Recover");
      }
      digitalWrite (RECOVER, LOW);
      delay(100);
      digitalWrite (RECOVER, HIGH);
    }
}
```