INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

ACADEMIA DE ACÚSTICA

Construcción de una flauta y su caracterización empleando análisis en frecuencia discreto

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA:

Mario Jiménez Hernández

Director de Tesis: Ing. José de Jesús Negrete Redondo.

Codirector de Tesis: Dr. Maximino Peña Guerrero.

México D.F. 30 de Enero 2007

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA
UNIDAD PROFESIONAL " ADOLFO LOPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TITULO DE **INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**

POR LA OPCION DE TITULACION **TESIS Y EXAMEN ORAL INDIVIDUAL**

DEBERA(N) DESARROLLAR **C. MARIO JIMÉNEZ HERNÁNDEZ**

"CONSTRUCCIÓN DE UNA FLAUTA Y SU CARACTERIZACIÓN EMPLEANDO ANÁLISIS EN FRECUENCIA DISCRETO"

DISEÑAR Y CONSTRUIR UNA FLAUTA A PARTIR DEL MODELO DE RESONADOR HELMHOLTZ, CON LA INFORMACIÓN CONTENIDA EN ARCHIVOS CON FORMATO WAV DE SUS NOTAS MUSICALES CARACTERIZARLA UTILIZANDO LA TRANSFORMADA DE FOURIER DISCRETA MEDIANTE UN SOFTWARE DESARROLLADO EN LENGUAJE C#

- INTRODUCCIÓN
- ANTECEDENTES
  - ELEMENTOS ACÚSTICOS DE LOS INSTRUMENTOS MUSICALES
  - PROCESAMIENTO DIGITAL DE SEÑALES ACÚSTICAS
- PROPUESTA DE SOLUCIÓN
  - INSTRUMENTO MUSICAL DESARROLLADO
  - SOFTWARE DESARROLLADO
- PRUEBAS Y RESULTADOS
- CONCLUSIONES Y TRABAJOS FUTUROS
- BIBLIOGRAFÍA

MÉXICO D.F. A 21 DE NOVIEMBRE DE 2006.

A S E S O R E S

ING. JOSE DE JESÚS NEGRETE REDONDO

DR. MAXIMINO PEÑA GUERRERO

ING. MARCO ANTONIO CERECEDO DÍAZ
JEFE DEL DEPARTAMENTO ACADÉMICO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA

Trabajo de tesis que forma parte de los resultados obtenidos en nuestro proyecto de investigación: *KALC: Biblioteca de Componentes de Software Para Prácticas de Laboratorio de Acústica Musical*, número de registro 20060550 asignado por la Secretaría de Investigación y Posgrado del Instituto Politécnico Nacional. Dicho proyecto fue realizado durante el año del 2006 dentro las instalaciones del Laboratorio de Acústica, y dirigido por el Dr. Maximino Peña Guerrero.

RESUMEN

Los instrumentos musicales han surgido a lo largo de la historia del hombre y siempre han sido construidos de manera empírica por medio de técnicas artesanales obtenidas a través de la experiencia, estas técnicas son transmitidas de generación en generación por los constructores de instrumentos musicales. El no tener un método formal para la construcción de instrumentos musicales ocasiona que al construir dos o más instrumentos en los que se requiere una similitud de sus características acústicas, se encuentren diferencias notables en su afinación y en el número de armónicos. Esto tiene mayor relevancia en el caso de instrumentos de viento construidos en una sola pieza debido a que no se puede modificar su afinación una vez terminada su construcción.

Esta tesis presenta la construcción de un instrumento musical de viento similar a una quena sudamericana afinado en la escala de G. Se modela utilizando la teoría de sistemas de vibración en tubos, y la teoría de resonadores Helmholtz para calcular el tamaño y la posición de los orificios que generan las diferentes notas musicales del instrumento, logrando con esto obtener el formalismo matemático que se requiere para su construcción. Se diseñó y construyó un instrumento, el cual se comparó con una quena artesanal, una flauta dulce y una tarka de madera de construcción rústica, esto permite distinguir sus componentes espectrales y hacer una comparación del timbre para cada instrumento.

Los resultados obtenidos se evaluaron implementando un programa en lenguaje C#, el cual consta de dos partes, la primera realiza el cálculo del tamaño y la posición de los orificios del instrumento, estos valores son empleados para su construcción; la segunda utiliza archivos de audio en formato WAV previamente capturados, uno para cada nota del instrumento, se realiza un análisis espectral en tiempo discreto para cada archivo utilizando la Transformada de Fourier Discreta, el resultado se compara con el espectro de un armónico generado digitalmente, esto permite verificar la afinación para cada nota. Se observa el espectro en frecuencia de $C5$ en cada instrumento analizando el timbre de la flauta con respecto a los otros instrumentos.

Agradecimientos

A mis amigos y compañeros que conocí durante la carrera... ...por las vivencias y momentos que pasamos juntos.

A mis amigos y amigas de los Talleres Culturales del IPN... ...porque me enseñaron que las artes, en especial la música, son el alimento del alma y nos enriquecen como seres humanos.

A mis profesores y profesoras que me formaron en mi estancia académica en la ESIME... ...porque sus enseñanzas me sirven para afrontar mi vida en el aspecto personal y laboral.

A mis profesores de la Academia de Acústica y mis asesores de tesis... ...por enseñarme que el conocimiento científico y humanístico están juntos.

A mi Madre y Padre... ...por darme la vida, preocuparse por mi educación, darme el apoyo moral y económico para poder terminar mi carrera.

A Dios... ...por permitirme estar aquí, y tener una búsqueda constante de conocimiento.

Mario Jiménez Hernández

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este capítulo describe un resumen del contenido de este trabajo de tesis, la escritura se realizó empleando el sistema LaTex [19, p:1] de composición de textos científicos, el problema propuesto se resuelve en base a una metodología científica de trabajo [3, p:43]. Primero se presentan los antecedentes acústicos necesarios para el diseño y construcción del instrumento musical propuesto, y despues se presentan los antecedentes de procesamiento digital de señales acústicas utilizados en la caracterización del instrumento. Como elementos acústicos, se presenta el análisis matemático de la escala musical de igual temperamento, esta escala se utiliza en la construcción de instrumentos temperados como el instrumento propuesto. A continuación se describen las ecuaciones para calcular la velocidad del sonido considerando el aire como medio de propagación para cualquier temperatura ambiente. Se presentan los fundamentos de la vibración en tubos y del resonador de Helmholtz con las ecuaciones para calcular la frecuencia de resonancia en estos sistemas.

Como antecedentes de procesamiento digital de señales acústicas, se presenta el teorema de muestreo de Nyquist, el cual determina la frecuencia de muestreo mínima con la que se deben capturar muestras de una señal analógica para que no exista pérdida de información. Se muestra el proceso de ventaneo utilizado en el procesamiento digital de señales, se describen las principales ventanas y sus características, en esta tesis se emplea la ventana de Hamming por ser la más adecuada para el

procesamiento de señales de audio. Se fundamenta la Transformada de Fourier Discreta junto con su algoritmo, esta herramienta matemática permite conocer el espectro en frecuencia de una señal digitalizada. Además, se presenta el formato WAV para almacenamiento de audio digital, el contenido de la estructura de su cabecera y la forma en que se almacenan las muestras.

De una manera detallada, se presenta el diseño del instrumento propuesto y se describe el software desarrollado para su caracterización. El instrumento propuesto es una flauta similar a una quena sudamericana, para su diseño se elige la escala de igual temperamento, la clave de afinación en G y la ejecución de una octava desde $G4$ hasta $G5$. Se diseña utilizando los elementos acústicos descritos, se usa la ecuación de resonancia en tubos para calcular su longitud efectiva, y con el modelo del resonador de Helmholtz se calcula la posición y el diámetro de los orificios a lo largo del instrumento que generan sus notas musicales. Se utiliza como material de construcción el PVC, empleando medidas industriales estándares con el objetivo de que sus dimensiones y el diámetro de los orificios sean reproducibles, a diferencia de un proceso artesanal en el cual las medidas varían entre instrumentos de características similares.

Se diseñó y se implementó un programa en lenguaje C# utilizando el compilador Visual Studio .NET 7.1 y la herramienta .NET Framework 1.1, el programa está dividido en dos partes. La primera muestra las características de dos archivos WAV, en la interfaz gráfica se muestran la ruta y el nombre de los archivos, el contenido de sus cabeceras, las gráficas de sus primeras 512 muestras, se genera una ventana de Hamming mediante un algoritmo y se convoluciona en tiempo con las muestras de audio de los archivos, de las muestras resultantes se obtienen sus espectros en frecuencia utilizando la Transformada de Fourier Discreta y se gráfican en la interfaz.

La segunda parte del software realiza los cálculos del diseño del instrumento en forma de algoritmo, se introducen las características del material de construcción, el tamaño de los orificios para generar las notas y la frecuencia límite de trabajo. Con los datos introducidos, el programa determina la longitud del instrumento y la posición

para cada orificio a lo largo del instrumento. En la interfaz gráfica se muestran los valores obtenidos en los cálculos, los datos de diseño elegidos y el dibujo del instrumento.

A continuación se presentan los resultados obtenidos en las pruebas de caracterización del instrumento utilizando el software desarrollado. Se almacenan previamente archivos WAV de cada nota que ejecuta el instrumento, y con un generador de armónicos en formato WAV se generan notas digitalmente con el valor de las frecuencias obtenidas en el diseño. Se verifica la afinación del instrumento al comparar el espectro en frecuencia de las notas que ejecuta con el de las notas generadas digitalmente.

El timbre del instrumento elaborado se compara con el de tres instrumentos de características musicales similares, los cuales son una quena sudamericana construida en madera de cedro, una flauta dulce fabricada en plástico y una tarka sudamericana construida en madera de pino. Previamente se almacena en archivos WAV la nota $C5$ generada por cada instrumento, con el software desarrollado se compara el espectro en frecuencia del instrumento elaborado con el de los demás instrumentos.

El resto del contenido de la tesis es el siguiente, en el capítulo dos se presentan los elementos acústicos necesarios para la construcción del instrumento propuesto y los de procesamiento digital de señales acústicas utilizados en su caracterización, en el capítulo tres se muestra el diseño del instrumento propuesto basándose en los elementos acústicos del capítulo dos, y se describe el programa elaborado en lenguaje C# para la caracterización y el diseño del instrumento empleando los elementos de procesamiento digital de señales acústicas del capítulo dos, en el capítulo cuatro se presentan las pruebas realizadas en la caracterización del instrumento y los resultados que se obtuvieron al comparar su timbre con el de instrumentos de características físicas y musicales similares, en el capitulo cinco se presentan las concluciones finales asi como algunas sugerencias para la mejora de este proyecto de tesis, por último se amplia el texto con un anexo que contiene un listado del codigo fuente del software desarrollado.

Capítulo 2

Antecedentes

Este capítulo esta dividido en dos secciones, la primera corresponde a los antecedentes acústicos que permiten la construcción del instrumento musical desarrollado en esta tesis, y la segunda corresponde a los antecedentes de procesamiento digital que permiten realizar el análisis espectral de intervalos de notas capturadas en formato WAV para la caracterización del instrumento musical.

2.1. Elementos acústicos de los instrumentos musicales

Un instrumento musical tiene como características acústicas el intervalo de octavas que puede ejecutar, el tipo de escala elegida en su construcción y la clave de afinación elegida en su ejecución, estos elementos determinan el valor de la frecuencia de las notas del instrumento; es necesario conocer la velocidad del sonido a la temperatura ambiente del lugar en el que se va a utilizar el instrumento para que los cálculos realizados tengan un alto grado de exactitud.

La flauta que se desarrolló en este trabajo de tesis, utiliza la teoría de vibraciones en tubos abiertos para calcular su frecuencia de resonancia fundamental y la teoría de resonadores de Helmholtz para posicionar los orificios a lo largo del instrumento. Al cubrir total o parcialmente estos orificios se varia la cantidad de flujo de aire y se obtiene una variación en la frecuencia de resonancia, generando las notas

musicales del instrumento. Una vez que se tienen estos elementos se realiza el cálculo y la construcción del instrumento musical con las características elegidas.

2.1.1. Escala de afinación

Los instrumentos musicales pueden ser clasificados de diferentes maneras, una de estas, se determina por la forma en como se produce el sonido en un instrumento, teniendo entonces tres clases de instrumentos, los de percusión (golpear el instrumento), los de aliento (soplar el instrumento) y los de cuerda (rasgar o arpegear el instrumento); otra forma de clasificarlos es por medio del tipo de escala musical que emplean, esta puede ser la escala de entonación justa o la escala de igual temperamento [9, p:39].

La escala de entonación justa permite poder distinguir exactamente entre un bemol y un sostenido, un ejemplo de un instrumento que emplea esta escala es el violín, la escala de igual temperamento unifica los bemoles y los sostenidos en una sola nota como es el caso del piano.

Un instrumento musical de viento construido en una sola pieza, emplea la escala de igual temperamento debido a que no puede existir una variación en su forma que permita modificar la frecuencia de las notas que proporciona, un ejemplo de este caso es una flauta dulce; en cambio en un violín la cuerda puede ser rasgada a cualquier distancia a lo largo de su longitud, debido a esto se dice que los violinistas afinan a oído, lo que representa un alto grado de dificultad ya que no tienen un patrón fijo dentro del instrumento que proporcione una nota exacta.

Toda la música occidental esta compuesta por 7 notas musicales fundamentales y sus alteraciones para ciertas notas, estas alteraciones corresponden a 5 notas en la escala de igual temperamento y 10 notas para la escala de entonación justa; a todo este intervalo de frecuencias se le denomina una octava [9, p:38].

Una octava tiene la característica de duplicar la frecuencia fundamental de la nota base, por ejemplo una nota A con un valor de 440 Hz tiene una octava ascendente en 880 Hz y una octava descendente en 220 Hz [5, p:16]. En la escala de igual

Tabla 2.1: Escala de igual temperamento

Intervalos de una Octava	Razón de Frecuencia
Unitono	1:1
Semitono o Segunda Menor	1.059463094:1
Tono Entero o Segunda Mayor	1.122462048:1
Tercera Menor	1.189207115:1
Tercera Mayor	1.259921050:1
Cuarta Perfecta	1.334839854:1
Cuarta Aumentada y Quinta Disminuida	1.414213562:1
Quinta Perfecta	1.498307077:1
Sexta Menor	1.587401052:1
Sexta Mayor	1.681792831:1
Séptima Menor	1.781797436:1
Séptima Mayor	1.887748625:1
Octava	2:1

temperamento (que se emplea en la construcción del instrumento musical propuesto) una octava esta dividida en 12 fracciones iguales [9, p:46] empleando la ecuación:

$$I_n = 2^{\frac{n}{12}}$$

Donde:

I_n = Número de intervalo de una octava, *adimensional*.

Para obtener los intervalos a partir de la fórmula anterior se procede de la siguiente manera, el primer intervalo resulta del cálculo de $2^{1/12} = 1{,}059463094$, el segundo intervalo será $2^{2/12} = 1{,}122462048$, y así consecutivamente hasta completar la octava; los valores para los 12 intervalos se muestran en la Tabla 2.1.

Una característica especial de cada instrumento musical es la clave en la que está afinado, existen 15 claves mayores y 15 claves menores en la escala de igual temperamento [9, p:49]; un instrumento de viento diseñado en la escala de igual temperamento está afinado solo para una clave. Para obtener los intervalos de la escala mayor en la escala de igual temperamento se sigue la estructura base de la Tabla 2.2 [9, p:49].

Dos líneas entre dos notas musicales representan un intervalo de un tono entero

Tabla 2.2: Intervalos de la escala mayor en la escala de igual temperamento

1 || 2 || 3 | 4 || 5 || 6 || 7 | 8

Tabla 2.3: Claves de la escala mayor en la escala de igual temperamento

Clave	Notas y sus Alteraciones
C	$C \,\|\|\, D \,\|\|\, E \mid F \,\|\|\, G \,\|\|\, A \,\|\|\, B \mid C$
D	$C\# \mid D \,\|\|\, E \,\|\|\, F\# \mid G \,\|\|\, A \,\|\|\, B \,\|\|\, C\#$
E	$C\# \,\|\|\, D\# \mid E \,\|\|\, F\# \,\|\|\, G\# \mid A \,\|\|\, B \,\|\|\, C\#$
F#	$C\# \,\|\|\, D\# \,\|\|\, E\# \mid F\# \,\|\|\, G\# \,\|\|\, A\# \mid B \,\|\|\, C\#$
G	$C \,\|\|\, D \,\|\|\, E \,\|\|\, F\# \mid G \,\|\|\, A \,\|\|\, B \mid C$
A	$C\# \mid D \,\|\|\, E \,\|\|\, F\# \,\|\|\, G\# \mid A \,\|\|\, B \,\|\|\, C\#$
B	$C\# \,\|\|\, D\# \mid E \,\|\|\, F\# \,\|\|\, G\# \,\|\|\, A\# \mid B \,\|\|\, C\#$
C#	$C\# \,\|\|\, D\# \,\|\|\, E\# \mid F\# \,\|\|\, G\# \,\|\|\, A\# \,\|\|\, B\# \mid C\#$
Db	$C \mid Db \,\|\|\, Eb \,\|\|\, F \mid Gb \,\|\|\, Ab \,\|\|\, Bb \,\|\|\, C$
Eb	$C \,\|\|\, D \mid Eb \,\|\|\, F \,\|\|\, G \mid Ab \,\|\|\, Bb \,\|\|\, C$
F	$C \,\|\|\, D \,\|\|\, E \mid F \,\|\|\, G \,\|\|\, A \mid Bb \,\|\|\, C$
Gb	$Cb \,\|\|\, Db \,\|\|\, Eb \,\|\|\, F \mid Gb \,\|\|\, Ab \,\|\|\, Bb \mid Cb$
Ab	$C \mid Db \,\|\|\, Eb \,\|\|\, F \,\|\|\, G \mid Ab \,\|\|\, Bb \,\|\|\, C$
Bb	$C \,\|\|\, D \mid Eb \,\|\|\, F \,\|\|\, G \,\|\|\, A \mid Bb \,\|\|\, C$
Cb	$Cb \,\|\|\, Db \,\|\|\, Eb \mid Fb \,\|\|\, Gb \,\|\|\, Ab \,\|\|\, Bb \mid Cb$

y una línea indica un intervalo de un semitono. La escala mayor comienza con la clave de C y la escala menor comienza con la clave de A. Los intervalos de las escalas se construyen a partir de la estructura de la Tabla 2.2. Para el caso de la escala de C se tiene $C \,||\, D \,||\, E \mid F \,||\, G \,||\, A \,||\, B \mid C$, para obtener la siguiente escala se recorre la estructura y se colocan las alteraciones correspondientes en cada nota, todas las escalas mayores en la escala de igual temperamento se muestran en la Tabla 2.3.

Dependiendo del instrumento se elige una clave, los instrumentos folklóricos de las culturas americanas antiguas y actuales emplean la clave de G para la composición de la música tradicional, a diferencia de la música occidental clásica y moderna que puede emplear diferentes claves en su composición.

2.1.2. Velocidad de propagación del sonido

Podemos definir el sonido como el fenómeno mecánico acústico que permite a los humanos percibir su medio ambiente por medio de la percepción de ondas mecánicas, a esta acción se le denomina sentido del oído [20, p:11]. La música es una combinación de frecuencias de ondas mecánicas generadas de manera armónica, al momento de percibirse producen una sensación agradable en el sistema auditivo humano.

El sonido es producto de las vibraciones mecánicas de las moléculas, su frecuencia varia en un intervalo estadístico entre 20 Hz y 20 kHz [20, p:152], este intervalo puede variar entre diferentes personas debido a su fisiología o condiciones particulares. La respuesta del oído no es lineal [20, p:151], este responde a un umbral de audición que comienza desde una presión acústica de 0 dB hasta 80 dB sin presentar dolor o daño al sistema auditivo, una vez que se sobrepasan los 80 dB se pone en riesgo de daño el mecanismo de audición.

Una propiedad del sonido es que su velocidad de transmisión es independiente de las frecuencias que contiene, esta velocidad únicamente es afectada por las condiciones del medio de propagación (temperatura, densidad) [15, p:47], el sonido se puede propagar en medios sólidos, líquidos y gaseosos, sin embargo el sistema auditivo evolucionó para percibir las vibraciones de moléculas en el aire de la atmósfera. El valor de la velocidad del sonido en el aire a 0° [18, p:39] esta definido por la siguiente ecuación:

$$c_0 = \sqrt{\frac{\gamma p_0}{\rho}}$$

Donde:

c_0 = Velocidad del sonido a °0, en m/s.

γ = Razón de calor específico a °0, *adimensional* (1,402).

p_0 = Presión atmosférica a °0, en *pascales* ($1{,}013 \cdot 10^5$).

ρ = Densidad del aire a °0, en kg/m^3 (1,293).

Resolviendo la ecuación para los valores establecidos, la velocidad del sonido es

331.4205755 m/s. Cuando existe una variación de temperatura en el aire [7, p:152], la velocidad del sonido se puede aproximar por medio de la siguiente ecuación:

$$c = c_0\sqrt{1 + \frac{T}{273}}$$

Donde:

c = Velocidad del sonido a temperatura ambiente, en m/s.

c_0 = Velocidad del sonido a 0°, en m/s.

T = Temperatura del aire, en °C.

Se considera la temperatura ambiente de 20°C como un promedio estadístico de la variación de la temperatura a lo largo del día. La velocidad del sonido con este valor de temperatura es de 343.3453734 m/s.

2.1.3. Resonancia en tubos

Los instrumentos musicales pueden ser modelados matemáticamente por medio de la teoría de vibración en tubos (instrumentos de aliento), vibración en cuerdas (instrumentos de cuerdas) y vibración en membranas o barras(instrumentos de percusión). La resonancia en tubos permite generar notas musicales con frecuencias específicas, la ventaja de este tipo de sistemas es que no pierden su afinación con el tiempo como sucede con las cuerdas estiradas en los instrumentos de cuerdas, la desventaja es que si al ser construido el instrumento se elabora de manera errónea, queda desafinado sin poderse realizar alguna corrección.

Una onda que se propaga a lo largo de un tubo circular que cumple la condición de que su longitud de onda sea mucho mayor que el diámetro de su abertura, se propaga como un frente de onda plano a lo largo del tubo [15, p:51]. Esto permite poder modelar de manera exacta la frecuencia de resonancia del tubo en función de su longitud efectiva. La longitud efectiva se obtiene por medio de una corrección que depende de la existencia o ausencia de una pestaña en el extremo del tubo [7, p:270]. En caso de que exista una pestaña, la longitud efectiva se determina por la ecuación:

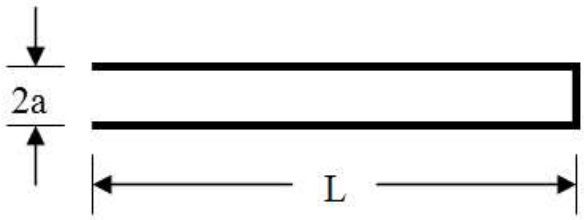

Figura 2.1: Tubo abierto en un extremo

$$Lef = L + 0{,}85a$$

Donde:

Lef = Longitud efectiva, en m.

L = Longitud del tubo, en m.

a = Radio de la abertura del tubo, en m.

En la ecuación anterior el valor del radio de la abertura del tubo incluye el valor de la pestaña. En el caso de la ausencia de una pestaña, se tiene la ecuación:

$$Lef = L + 0{,}6a$$

La Figura 2.1 muestra las características de un tubo abierto en un extremo sin pestaña.

La frecuencia de resonancia en un tubo abierto por un extremo [7, p:269] se determina por la siguiente ecuación usando el valor de $n = 1$ para calcular el valor del armónico fundamental.

$$fn = \frac{2n-1}{4}\frac{c}{Lef}$$

Donde:

fn = Frecuencia de resonancia, en Hz.

n = Número de armónico, *adimensional*.

c = Velocidad del sonido a temperatura ambiente, en m/s.

Lef = Longitud efectiva, en m.

En el caso de que se quiera obtener un armónico de la frecuencia fundamental se cambia el valor de n por el armónico correspondiente. La frecuencia de resonancia en un tubo abierto en ambos extremos [9, p:83] se determina por la misma ecuación anterior cambiando únicamente el valor en el denominador de 4 por 2, como se observa en la siguiente ecuación:

$$fn = \frac{2n-1}{2}\frac{c}{Lef}$$

Con la primera ecuación se diseñan los instrumentos folklóricos sudamericanos llamados Zampoñas, estos consisten en 15 tubos cerrados por un extremo cuyas frecuencias fundamentales corresponden a dos octavas en forma ascendente. La frecuencia fundamental de cualquier flauta compuesta de un solo tubo como las quenas sudamericanas, se calcula a partir de la ecuación anterior.

2.1.4. Resonador de Helmholtz

Un resonador de Helmholtz consiste de un recipiente cerrado con un volumen fijo, y un orificio de área constante con un cuello de cierta longitud, por esta abertura se proporciona un flujo de aire que pone en resonancia al sistema [5, p:43]. Para que pueda existir resonancia la longitud de onda debe ser mucho mayor que la raíz cuadrada del área de la abertura [7, p:297], esto se muestra en la siguiente ecuación:

$$\lambda >> S^{1/2}$$

Donde:

λ = Longitud de onda, en m.

S = Área del orificio, en m^2.

En la Figura 2.2 se muestra la estructura de un resonador de Helmholtz.

Al igual que como sucede en el cálculo de la resonancia en tubos, la longitud

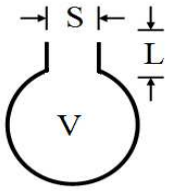

Figura 2.2: Resonador de Helmholtz

efectiva del cuello de la abertura requiere de una corrección si tiene en su extremo una pestaña [7, p:298], para este caso la longitud efectiva se calcula a partir de la ecuación:

$$L' = L + 2(0{,}85a)$$

En el caso de que no tenga pestaña se emplea la ecuación:

$$L' = L + (0{,}85 + 0{,}6)a$$

En ambas fórmulas se tiene que:

L' = Longitud efectiva, en m.

L = Longitud del cuello, en m.

a = Radio del orificio, en m.

La fórmula para calcular la frecuencia de resonancia del resonador [7, p:299] es:

$$\omega_0 = c\sqrt{\frac{S}{L'V}}$$

Donde:

ω_0 = Frecuencia de resonancia, en Hz.

c = Velocidad del sonido a temperatura ambiente, en m/s.

S = Área del orificio, en m^2.

L' = Longitud efectiva, en m.

V = Volumen del resonador, en m^3.

Cualquiera de los instrumentos musicales de viento construidos en una sola pieza como las flautas dulces, emplean la teoría de resonadores de Helmholtz para calcular la posición exacta de los orificios a lo largo del tubo que forma el instrumento.

2.2. Procesamiento digital de señales acústicas

Una vez elegidas las características acústicas que permiten construir el instrumento, es necesario capturar y almacenar los sonidos emitidos por el instrumento para su análisis en el dominio de la frecuencia por medio de la Transformada de Fourier, para esto se emplea el formato de audio digital WAV por ser un formato donde las muestras almacenadas se encuentran en forma directa a diferencia de los otros formatos que contienen algoritmos de compresión, la señal tiene que ser capturada con una frecuencia de muestreo que permita obtener el espectro sin perdidas de información.

Debido a que la señal capturada puede tener una duración muy grande en el dominio del tiempo se realiza su análisis en solo una porción de todo el intervalo de muestras, para hacer esto, se emplea el proceso de ventaneo, se debe elegir una ventana que permita la menor pérdida de información durante el procesamiento digital.

Una vez elegidas todas las características, el espectro de la señal se puede representar de manera gráfica en una computadora por medio de un software desarrollado para este propósito específico.

2.2.1. Muestreo de señales en tiempo discreto

Actualmente, el uso de computadoras digitales en las que se pueden implementar algoritmos para la manipulación de señales con velocidades de procesamiento elevadas, permite que procesos analógicos empleados en la obtención de características de señales sean implementados en forma digital a partir de señales muestreadas en forma discreta.

Para poder convertir las señales analógicas a señales digitales, se deben cap-

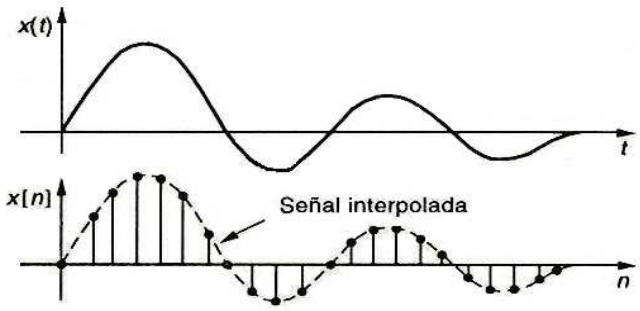

Figura 2.3: Muestreo digital de una señal analógica

turar muestras de la señal analógica en espacios iguales y continuos de tiempo, de tal manera que la forma de onda de la señal original se pueda recuperar mediante interpolación [8, p:17]. La elección de la frecuencia de muestreo para una señal dada depende de que tan rápido cambia la señal con el tiempo. La Figura 2.3 representa el muestreo de una señal analógica en una señal digital discreta.

Para lograr esto, se utiliza el criterio de Nyquist, que establece que para que en una señal no exista pérdida de información, esta debe ser muestreada al doble de su frecuencia máxima de trabajo [14, p:29]. Un ejemplo claro, es la frecuencia de muestreo del audio digital, debido a que la frecuencia máxima que escucha un humano en promedio es de 20 kHz, bastará con tomar muestras al doble, el estándar establecido es de 44,100 muestras por segundo.

Si no se realiza el muestreo utilizando este criterio y se hace con una frecuencia menor, se produce el fenómeno de aliasing [14, p:27], el cual proporciona una representación errónea de la señal. Una vez capturada la señal con este error no se puede reconstruir o corregir de ninguna manera. En la Figura 2.4 se presenta el muestreo de la señal analógica de la Figura 2.3 con una frecuencia de muestreo menor a la requerida, lo que produce alisamiento en la señal reconstruida al aplicarle interpolación a sus muestras.

La expresión del teorema de Nyquist es la siguiente:

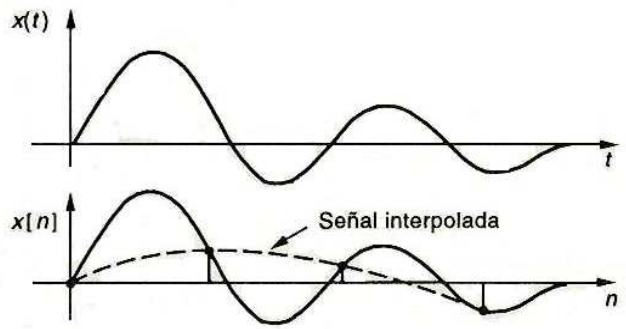

Figura 2.4: Aliasing digital de una señal analógica

$$V_m = 2f_M$$

Donde:

V_m = Velocidad de muestreo, en *muestras por segundo.*

f_M = Frecuencia máxima de trabajo, en Hz.

Los sonidos acústicos que proporciona un instrumento musical dependen del intervalo de octavas que puede producir, y la cantidad de armónicos que acompañan a la frecuencia fundamental. En el caso de esta tesis, las muestras capturadas están en formato monofonico con una frecuencia de muestreo de 22,050 htz, debido a que el instrumento construido no genera frecuencias superiores.

2.2.2. Tipos de Ventanas

Una señal puede tener una duración finita en un intervalo de tiempo muy grande, sin embargo un análisis de la señal en todas sus muestras resultaría inadecuado. Para evitar este problema se utilizan intervalos de muestras de la señal denominados ventanas. El mecanismo de ventaneo, consiste en dividir la señal a analizar en intervalos de duración finitos, los cuales son tomados de manera consecutiva a lo largo de la señal [10, p:239].

Las muestras de la señal que se encuentran en los extremos de una ventana

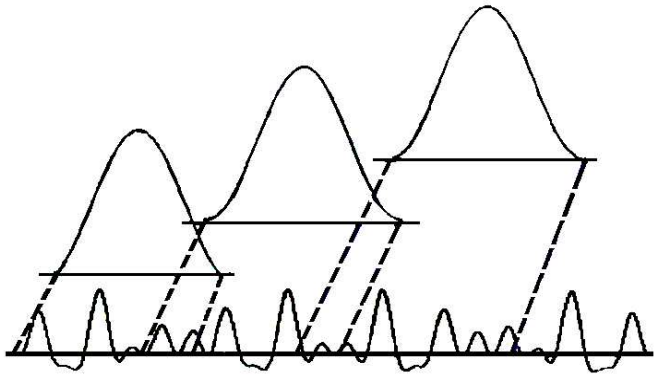

Figura 2.5: Solapamiento de ventanas en una señal

tienen menor peso que las que se encuentran en medio, esto permite que las características de las muestras en los extremos no causen una mala interpretación de lo que ocurre en la parte central dentro de un proceso específico [6, p:486].

Las ventanas pueden ser colocadas de manera que existan solapamientos en sus extremos analizando únicamente los intervalos centrales [1, p:142], logrando con esto una mejor calidad en los resultados, en la Figura 2.5 se muestra el solapamiento de una señal utilizando ventanas tipo Hamming.

La duración de una ventana determina la cantidad de cambios que se pueden obtener, para una duración temporal larga se omiten los cambios locales producidos en la señal, mientras que con una duración demasiado corta se reflejan los cambios puntuales y se reduce la resolución espectral. Al utilizar una ventana se hace que se convolucione la Transformada de Fourier de la señal con la Transformada de Fourier de la ventana, por esto se debe de elegir la ventana que produzca menor distorsión durante el proceso que se realiza [11, p:468]. La Tabla 2.4 muestra las ventanas utilizadas en el tratamiento de señales discretas.

Como se puede ver en la Tabla 2.4, la ventana rectangular permite trabajar con un intervalo de la señal de manera natural sin realizar ningún proceso sobre las muestras, sin embargo tiene la desventaja de que al convolucionar el espectro de su Trans-

Tabla 2.4: Tipos de Ventanas

Ventana	$w(n)$, $0 \leq n \leq M-1$
Bartlett	$1 - \frac{2\lvert n - \frac{M-1}{2} \rvert}{M-1}$
Blackman	$0{,}42 - 0{,}45cos\frac{2\pi n}{M-1} + 0{,}08cos\frac{4\pi n}{M-1}$
Hamming	$0{,}54 - 0{,}46cos\frac{2\pi n}{M-1}$
Hanning	$\frac{1}{2}(1 - cos\frac{2\pi n}{M-1})$
Rectangular	$1 = \lvert n \rvert \leq \frac{T_0}{2}$

Tabla 2.5: Características de Ventanas

Ventana	Ancho del lóbulo	Pico del Lóbulo (dB)
Bartlett	$8\pi/M$	-27
Blackman	$12\pi/M$	-58
Hamming	$8\pi/M$	-43
Hanning	$8\pi/M$	-32
Rectangular	$4\pi/M$	-58

formada de Fourier con las muestras de una señal genera distorsión en el resultado, otros tipos de ventanas producen menos distorsión.

En esta tesis se elige la ventana de Hamming porque al convolucionar el espectro de su Transformada de Fourier con las señales de prueba no distorsiona los resultados dentro del lóbulo principal de trabajo [4, p:361], en la Tabla 2.5 se muestran las características de las ventanas de la Tabla 2.4.

2.2.3. Transformada de Fourier Discreta

Una herramienta matemática que permite representar una señal dada en el dominio del tiempo en sus componentes armónicos en el dominio de la frecuencia, es la Transformada de Fourier. La señal original se descompone en señales ortogonales de senos y cósenos con amplitudes respectivas para cada uno de los componentes, esto permite conocer el espectro en frecuencia de cualquier señal dentro de un intervalo de frecuencias previamente definido.

El equivalente de la Transformada de Fourier en tiempo continuo para el análi-

sis en el dominio del tiempo discreto es la Transformada de Fourier Discreta (TFD). Igual que su contraparte permite tener una representación en el dominio de la frecuencia de una señal muestreada en forma discreta [13, p:31], la TFD se representa por la siguiente ecuación:

$$F(\frac{n}{NT}) = \frac{1}{N}\sum_{k=0}^{N-1} m(kT)e^{-}j\frac{2\pi nk}{N}$$

Donde:

N = Número de muestras que componen la ventana, *adimensional*.

T = Periodo de muestreo, en s.

n = Índice de la frecuencia que se desea calcular, $n = 0, 1, ...N-1$, *adimensional*.

$m(KT)$ = Valor de la muestra tomada en el instante KT, *adimensional*.

Para implementar la ecuación anterior en un algoritmo computacional se descompone la parte exponencial en senos y cosenos por medio de la identidad de Euler [2, p:524] de la siguiente forma:

$$e^{-}j\frac{2\pi nk}{N} = cos\frac{-2\pi nk}{N} + jsen\frac{-2\pi nk}{N}$$

El algoritmo de la Transformada de Fourier Discreta calcula para n frecuencias el valor del módulo (n) que representa los coeficientes en frecuencia [1, p:185], estos valores se almacenan en un arreglo (array) para después recuperarlos, el algoritmo se muestra a continuación:

```
pi = 3.1416
BUCLE n = 0 A (N/2)-1
  COMIENZO
  SumaParteReal = 0
  SumaParteImag = 0
  BUCLE k = 0 A N-1
    COMIENZO
    SumaParteReal = SumaParteReal + m(k)*COS(-2*pi*n*k/N)
    SumaParteImag = SumaParteImag + m(k)*SEN(-2*pi*n*k/N)
  FIN
  módulo(n) = RAIZ CUADRADA(sumaParteReal*SumaParteReal
                            + SumaParteImag*SumaParteImag)
FIN
```

2.2.4. Formato de audio digital WAV

Para almacenar audio digital, el formato más común es el WAV (Waveform Audio File Format), creado por la empresa Microsoft e implementado originalmente para el sistema operativo Windows. Es un subconjunto de especificaciones RIFF (Resourcen Interchange File Format) para formatos multimedia, este formato almacena segmentos (chunks) de información multimedia conteniendo su descripción, su formato y su lista de reproducción [1, p:207]. El formato WAV se almacena dentro de un archivo con formato RIFF en el cual se definen los segmentos que contiene.

Todos los archivos RIFF llevan una cabecera de 8 bytes, los primeros 4 bytes forman un campo que identifica al archivo, teniendo en su contenido el identificador RIFF; los otros 4 bytes especifican la longitud de los datos a partir de la cabecera (la longitud total del archivo menos 8). Después de la cabecera RIFF hay 4 bytes que identifican el tipo de datos que contiene el archivo, para el caso de un archivo WAV estos 4 bytes contienen el identificador WAVE.

Del conjunto de segmentos que definen el formato WAV dos son obligatorios, el segmento de formato y el segmento de datos. El segmento de formato debe aparecer antes que el segmento de datos. Los componentes que forman la cabecera de un archivo WAV se muestran en la Tabla 2.6.

Al final del segmento del contenido de la cabecera, comienza el área de datos donde se almacena la información de audio, el almacenamiento de los datos en un archivo WAV se realiza sin hacer algoritmos de comprensión a diferencia de otros formatos como el MP3. La estructura de la secuencia de datos depende de la frecuencia de muestreo, el número de canales y el número de bytes por muestra en que fueron capturados.

Cuando un archivo WAV se encuentra en formato monofonico las muestras se almacenan en forma consecutiva, si el archivo esta en formato estereo las muestras se almacenan de forma alternada, teniendo una muestra por cada canal [1, p:208]. Si ca-

Tabla 2.6: Cabecera de un archivo WAV

Bytes	Contenido	Descripción
00-03	RIFF	Cabecera
04-07	—	Tamaño
08-11	WAVE	Formato
12-15	fmt	Extensión
16-19	16	Formato 1
20-21	1	Formato 2
22-23	1,2	Número de canales
24-27	—	Frecuencia de Muestreo
28-31	—	Bytes por segundo
32-33	1,2,4	Bytes por captura
34-35	8,16	Bits por muestras
36-39	data	Nombre del segmento
40-43	—	Número de muestras
resto	—	Muestras

da muestra es de 8 bits entonces representa un byte de información y si es de 16 bits dos byte.

2.3. Resumen

En este capítulo se presentaron los elementos acústicos necesarios para la construcción del instrumento musical de viento a caracterizar. Se fundamentó el empleo de la escala de igual temperamento y su construcción matemática, se calculó la velocidad del sonido para la temperatura ambiente del aire (que es el medio empleado para la transmisión del sonido), se presentó el método para determinar la frecuencia de resonancia en tubos y por último se hizo el análisis del resonador de Helmholtz, con estos modelos se calcula la longitud del instrumento, la posición y el tamaño de los orificios a lo largo del instrumento, logrando con esto poder variar la frecuencia de resonancia en el instrumento y obtener las notas musicales.

Como primer elemento de procesamiento digital se presentó el muestreo digital de señales acústicas, el cual tiene su base en el teorema de muestreo de Nyquist, que

fundamenta el valor de la frecuencia de muestreo a la cual deben ser capturadas las señales para no tener pérdida de información, se presentaron las ventanas utilizadas en el tratamiento digital de señales con sus características, siendo la ventana de Hamming la mas eficiente en el tratamiento de señales de audio, se fundamentó el uso de la Transformada de Fourier Discreta para ser utilizada como herramienta en la obtención del espectro en frecuencia de señales en forma discreta, por último se mostraron las características del formato WAV utilizado para la manipulación de audio digital.

Capítulo 3

Propuesta de solución

Este capítulo esta dividido en dos secciones, la primera presenta los cálculos de diseño realizados para la construcción del instrumento musical propuesto, se obtiene la longitud del instrumento, la posición y el tamaño de los orificios que generan las notas musicales, posteriormente se muestra el instrumento construido en PVC. En la segunda sección, se muestra el diseño del software desarrollado para la obtención del espectro en frecuencia de una señal de audio previamente capturada en formato WAV por medio de la Transformada de Fourier Discreta, y el software para calcular la posición de los orificios que generan las notas musicales a partir de las características físicas del material elegido para la construcción del instrumento.

3.1. Instrumento musical desarrollado

En esta tesis se propone el diseño y construcción de un instrumento musical del tipo de los alientos, similar a una quena sudamericana. La quena es un instrumento folklórico formado por un tubo abierto en ambos extremos con siete orificios a lo largo del tubo, seis en la parte superior y uno en la parte inferior. Al ser tapados los orificios de manera total o parcial generan las notas musicales, el diámetro y la posición de los orificios determinan la frecuencia de resonancia del instrumento en la ejecución de una nota.

Los radios de los orificios que generan las notas musicales en una quena son di-

Tabla 3.1: Clave de G en la escala de igual temperamento

$C \mid\mid D \mid\mid E \mid\mid F\# \mid G \mid\mid A \mid\mid B \mid C$

ferentes, variando para cada uno el área donde realiza el flujo de aire, esto se hace con el fin de tener los orificios en posiciones equidistantes, permitiendo al músico tener una ejecución más ergonómica. En el diseño del instrumento propuesto se utiliza para todos los orificios un mismo radio, con el objetivo de emplear medidas industriales estándares, tanto en las brocas para su perforación como en el uso de tubos de PVC con diámetros definidos, teniendo entonces condiciones físicas reproducibles, a diferencia de un trabajo artesanal.

En un extremo de la parte superior de la quena, se encuentra una pequeña abertura que permite la entrada de aire hacia el instrumento, este pequeño orificio tiene la función de boquilla del instrumento, el músico debe introducir un flujo de aire constante a través de esta abertura y al mismo tiempo tapar este extremo del instrumento con la parte inferior de sus labios, a este proceso se le denomina *emboquillar*. Una quena esta afinada en la clave de G y se construye en madera, bambú, totora o plástico, su intervalo de frecuencias de resonancia abarca una octava, desde $G4$ hasta $G5$, para la construcción del instrumento propuesto se utiliza la misma clave de afinación y el mismo intervalo de frecuencias.

3.1.1. Características musicales

Para la construcción del instrumento se emplea la escala de igual temperamento, afinándolo en la clave de G. Esta clave solo tiene una alteración de un semitono correspondiente a $F\#$, sus intervalos se observan en la Tabla 3.1.

El intervalo de las notas que proporciona el instrumento se encuentra entre $G4$ y $G5$, que corresponden a una octava. Para calcular los valores de las notas se utilizan las razones de frecuencia de una octava obtenidas en la Tabla 2.1. Los cálculos

Tabla 3.2: Frecuencias de las notas del instrumento propuesto

Nota	Frecuencia (Hz)
G4	391.995436
A4	440
B4	493.8833013
C5	523.2511306
D5	587.3295358
E5	659.2551138
F#5	698.4564629
G5	783.990872

se realizan usando $G4 = 391.995436$ Hz como frecuencia fundamental, a continuación se muestra el cálculo para $A4$.

$$A4 = G4 \cdot 1{,}122464048 = 440Hz$$

De la misma forma, se calculan los valores de las frecuencias de las notas restantes que proporciona el instrumento, la Tabla 3.2 muestra los valores obtenidos.

3.1.2. Diseño y construcción del instrumento

Se utilizó como material de construcción el PVC porque permite una fácil manipulación de corte y perforación. La medida del radio del tubo es de 1/2"(0.0127 m), el radio de los orificios que generan los notas musicales es de 1/32"(0.00555625 m) y tiene un espesor de 1/16"(0.0015875 m), se eligen estos valores por ser los más aproximados al tamaño promedio de las quenas sudamericanas.

La longitud del tubo con el que se construye el instrumento musical determina su frecuencia de resonancia fundamental, esta se calcula a partir de la longitud efectiva. Para calcular la longitud efectiva se emplea la ecuación de resonancia en tubos abiertos en ambos extremos (Sección 2.1.3). Se usa la nota más grave como el valor de frecuencia límite que genera el instrumento, en este caso es $G4 = 391.995436$ Hz, el cálculo de la longitud efectiva se muestra a continuación:

$$Lef = \frac{c}{2 \cdot G4} = \frac{343{,}3453734 m/s}{2 \cdot 391{,}995436 Hz} = 0{,}437945626 m$$

Donde:

Lef = Longitud efectiva del tubo de PVC, en m.

c = Velocidad del sonido a temperatura ambiente, en m/s.

$G4$ = Frecuencia de resonancia límite del instrumento, en Hz.

Como el PVC puede ser considerado como un tubo uniforme se hace la corrección de la longitud efectiva sin presencia de pestaña. Con el valor obtenido en la ecuación anterior, se calcula el valor de la longitud real del instrumento a partir de la ecuación de corrección en un tubo abierto sin pestaña (Sección 2.1.3), el cálculo de la longitud real es:

$$L = Lef - 0{,}6a = 0{,}437945626 m - 0{,}6(0{,}0127 m) = 0{,}430325626 m$$

Donde:

L = Longitud real del instrumento, en m.

Lef = Longitud efectiva del tubo de PVC, en m.

a = Radio del tubo de PVC, en m.

Los instrumentos de viento como las flautas, son sistemas acústicos que trabajan como un resonador de Helmholtz en cada orificio del instrumento, al tapar cada uno se controla la salida de flujo de aire del resonador, obteniendo las diferentes frecuencias de resonancia que generan las notas musicales. Para obtener la posición de los orificios a lo largo del instrumento se calculan los volúmenes necesarios de los resonadores que generan las frecuencias para cada una de las notas musicales. Se puede considerar al instrumento como un cilindro, cuyo volumen se determina por el área del tubo y el valor de la longitud requerida para cada resonador.

La primera nota musical que genera el instrumento por medio de un orificio es $A4 = 440$ Hz, para calcular la posición del orificio a lo largo del tubo de PVC se cal-

cula el volumen necesario del resonador que genera esta frecuencia (Sección 2.1.4), la ecuación para obtener el volumen se muestra a continuación:

$$V = \frac{Sc^2}{L'\omega_0^2}$$

Donde:

V = Volumen del resonador que genera la nota musical, en m^3.

S = Área del orificio que genera la nota, en m^2.

c = Velocidad del sonido a temperatura ambiente, en m/s.

L' = Espesor efectivo del tubo empleado, en m.

ω_0 = Frecuencia en radianes de la nota a generar, en Hz.

El área del orificio se calcula a partir de la fórmula del área de un circulo utilizando el valor del radio del orificio propuesto, a continuación se muestra el cálculo con los valores elegidos:

$$S = \pi r^2 = 3{,}1416 \cdot 30{,}87191406^{-6} m^2 = 96{,}98697842^{-6} m^2$$

Donde:

S = Área del orificio que genera la nota, en m^2.

r = Radio del orificio que genera la nota, en m.

El valor de la velocidad del sonido a temperatura ambiente de 20° (Sección 2.1.2) al cuadrado es:

$$c^2 = 343{,}3453734^2 m/s = 117886{,}0454 m^2/s^2$$

Se puede considerar que el valor de la longitud efectiva del cuello del resonador es equivalente al espesor efectivo del tubo de PVC, este se calcula sin presencia de pestaña (Sección 2.1.4), utilizando el espesor del tubo de PVC y el valor del radio del orificio que genera la nota musical, el valor del espesor efectivo es:

$$L' = L + (0{,}85 + 0{,}6)a = 1{,}5875^{-3}m + (1{,}45)5{,}55625^{-3}m = 9{,}6440625^{-3}m$$

Donde:

L' = Espesor efectivo del tubo de PVC, en m.

L = Espesor del tubo de PVC, en m.

a = Radio del orificio que genera la nota, en m.

La frecuencia de resonancia tiene que estar expresada en radianes, por lo tanto el cuadrado de la frecuencia es:

$$\omega_0^2 = 4\pi^2 f^2 = 4 \cdot 3{,}1416^2 \cdot 440^2 Hz = 7643021{,}648 Hz^2$$

Donde:

ω_0 = Frecuencia en radianes de la nota a generar ($A4$), en Hz.

f = Frecuencia de la nota a generar ($A4$), en Hz.

Con los resultados anteriores se procede a calcular el volumen del resonador que genera la frecuencia de $A4 = 440$ Hz y que corresponde al primer orificio, el valor del volumen obtenido es:

$$V = \frac{96{,}98697842^{-6} m^2 \cdot 117886{,}0454 m^2/s^2}{9{,}6440625^{-3} m \cdot 7643021{,}648 Hz^2} = 155{,}1139018^{-6} m^3$$

Por medio de la fórmula del volumen de un cilindro se calcula la posición a lo largo del tubo (altura del cilindro) en que se coloca el orificio que genera la nota, para el caso de $A4$ se tiene:

$$h = \frac{V}{\pi r^2} = 0{,}3061212 m$$

Donde:

h = Posición del orificio a lo largo del tubo de PVC, en m.

V = Volumen del resonador, en m^3.

Tabla 3.3: Posición de los orificios de las notas del instrumento propuesto

Nota	Distancia (m)
A4	0.3061212
B4	0.2429686
C5	0.2164603
D5	0.1718047
E5	0.1363615
F#5	0.1082302
G5	0.09642214

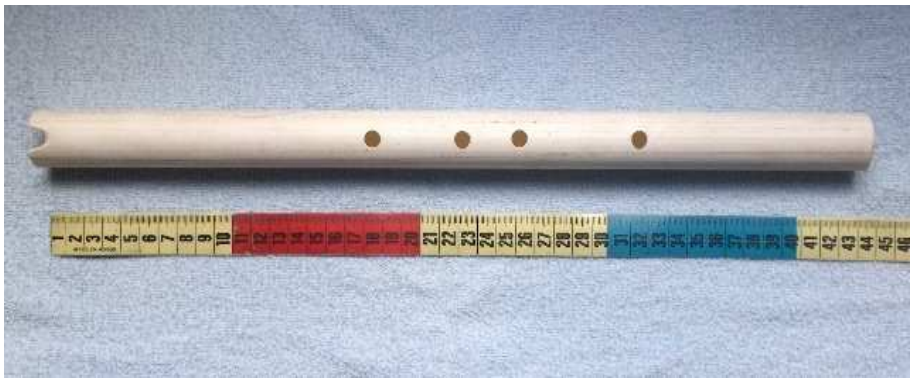

Figura 3.1: Flauta en G

r = Radio del tubo de PVC, en m.

Este procedimiento se realiza para cada valor de frecuencia de cada una de las notas generadas por el instrumento, obteniendo el valor de la posición de cada orificio a lo largo del tubo de PVC, la Tabla 3.3 muestra los valores obtenidos para cada una de las notas.

Debido a que las distancias entre orificios consecutivos no son equidistantes se dificulta la ejecución de las notas por parte del músico, por tal motivo solo se perforan los primeros cuatro orificios en el tubo de PVC, teniendo entonces cinco notas musicales: $G4$ (Frecuencia de resonancia fundamental del tubo), $A4$ (Primer orificio), $B4$ (Segundo orificio), $C5$ (Tercer orificio) y $D5$ (Cuarto orificio), el instrumento construido se muestra en la Figura 3.1.

3.2. Software desarrollado

Para realizar el software que se presenta en este trabajo de tesis se uso el lenguaje de programación C#. Este lenguaje fue desarrollado por Microsoft a finales de los años 90 formando parte de la estrategia general de .NET, se considera un lenguaje orientado a componentes debido a su compatibilidad con la escritura de componentes de software, sus características son las propiedades, los métodos y los eventos [17, p:8]. El compilador utilizado fue Visual Studio.NET 7.1 y la herramienta .NET Framework 1.1.

Se decidió utilizar el lenguaje C# porque es compatible con la programación distribuida, utiliza código intermedio que le proporciona portabilidad, permite interoperabilidad entre lenguajes logrando la creación de grandes sistemas distribuidos de software y tiene una completa integración con la plataforma Windows.

El software desarrollado en esta tesis se divide en dos partes, la primera obtiene el espectro en frecuencia de las primeras 512 muestras de dos archivos WAV previamente almacenados. Las muestras se almacenan en memoria, se genera un filtro Ventana de Hamming que se convoluciona en tiempo con las muestras almacenadas, a los valores resultantes se les aplica la Transformada de Fourier Discreta. En la interfaz gráfica se muestran los indicadores de las cabeceras de los dos archivos, las graficas de sus primeras 512 muestras y sus correspondientes espectros en frecuencia. La información para cada archivo se presenta con diferentes colores, en azul para el primero y rojo para el segundo. El software se diseño para utilizar archivos WAV en formato monofonico, con muestras de 8 bits a cualquier frecuencia de muestreo.

La segunda parte del software calcula las características físicas del instrumento musical propuesto, se introducen los valores del radio y el espesor del tubo de PVC en el que se construye la flauta, y el valor del radio de los orificios que proporcionan las notas musicales. El programa determina los 12 intervalos de la octava del instrumento, la frecuencia de cada nota musical y la posición de cada orificio a lo largo del tubo,

estos valores se presentan en una tabla dividida en cinco columnas, las cuales muestran el nombre de los intervalos de una octava, la razón para cada intervalo, el valor de la frecuencia de las notas musicales, el nombre de las notas y la posición de los orificios que generan las notas. En la parte inferior de la tabla se dibuja el instrumento con los orificios que generan las notas musicales, cada uno con el símbolo de su nota, en el lado izquierdo del instrumento se dibuja la boquilla, por último, abajo del dibujo se muestran las características físicas del material con las que se realizan los cálculos.

3.2.1. Espectro en frecuencia de dos archivos WAV

La Interfas gráfica está desarrollada en base a una aplicación tipo formulario, dividida en dos áreas, la primera muestra la ruta, el nombre y los elementos que forman la cabecera de cada uno de los dos archivos (Tabla 2.6), la segunda se divide en cuatro secciones: las dos primeras muestran las gráficas de las primeras 512 muestras de las señales de audio contenidas en los archivos, y las dos restantes muestran las gráficas de los espectros en frecuencia de las muestras almacenadas utilizando la Transformada de Fourier Discreta (TFD). A continuación se describe el código implementado en la realización de esta parte del software, el código fuente se encuentra en el apéndice A.

Al principio del programa se indica al compilador que se utilizan los espacios de nombres $System$, $System.IO$, $System.Drawing$, $System.Windows.Forms$ y $System.Drawing.Drawing2D$, por medio de la instrucción *using*. Estos son espacios de nombres reservados para elementos que se asocian con la biblioteca de clases de .NET Framework [17, p:19], la especificación se realiza de la siguiente forma:

```
000  using System;
002  using System.Drawing;
003  using System.Windows.Forms;
```

Cualquier programa realizado en C# tiene todo su contenido dentro de una clase, en la instrucción *public class* $Transformada : Form$ se declara la clase prin-

cipal del programa, se usa la palabra *class* para declarar la definición de una nueva clase con el nombre de $Transformada$. La definición de una clase comienza con una llave de apertura y finaliza con una llave de cierre, todos los elementos que se encuentran dentro de las dos llaves, son los elementos de la clase.

Mediante la palabra $Form$ se le indica al compilador el uso de un constructor que permite crear lo que se denomina una ventana en ambiente Windows, en .NET Framework se le denomina formulario [17, p:729]. Un formulario contiene una barra de titulo con el nombre de la aplicación y un área de trabajo dentro del formulario denominada área del cliente, los dos puntos y la palabra $Form$ delante de la declaración de la clase $Transformada$ le proporcionan las características de la clase $Form$ a la clase $Transformada$.

Después se coloca la instrucción *public static void* $Main()$, esta inicia el método $Main()$ que establece el punto de inicio de cualquier programa en lenguaje C#. La palabra *public* es un especificador de acceso, cuando un miembro de una clase va precedido por *public* significa que se puede tener acceso a el mediante código fuera de la clase en la que se ha declarado. El método $Main()$ se declara como *public* ya que es llamado por código fuera de su clase dentro del sistema operativo.

Mediante la palabra *static* se llama a $Main()$ antes de que se haya creado un objeto de su clase, esto es necesario ya que $Main()$ se llama al iniciar el programa. La palabra *void* indica al compilador que $Main()$ no devuelve ningún valor, los paréntesis vacíos después de $Main$ indican que no se pasa ningún argumento al método, las llaves de cierre y apertura indican el campo de instrucciones dentro de $Main()$.

Con la instrucción $Application.Run(newTransformada())$ se indica que un objeto nuevo llamado $Transformada$ se pasa como argumento al método Run que se encuentra dentro de la clase *Application*, esto inicia la ejecución de la ventana del Formulario. La clase *Application* esta definida en $System.Windows.Forms$, el código de las instrucciones mencionadas es el siguiente:

```
005  public class Transformada:Form {
006    public static void Main() {
007      Application.Run(new Transformada());
009    }
     ...CONTENIDO DEL PROGRAMA...
388  }
```

En el siguiente bloque de código se declara un constructor de la clase $Transformada$ precedido de la palabra *public* que le indica al compilador que sus componentes están disponibles para otras partes de código del programa. Dentro de este constructor se proporcionan las características del formulario. La palabra *this* permite indicar las características del objeto que ha sido declarado por el constructor.

Mediante la instrucción $this.Text$ se asigna el texto que se visualiza sobre la barra de titulo del formulario, la cadena del texto tiene que estar incluida entre comillas. La instrucción $this.size$ asigna el número de pixeles del ancho y alto del formulario, estos valores se incluyen en una estructura llamada *size*, que se genera con la instrucción $newsize(Ancho, Alto)$, esta instrucción se encuentra en el campo $System.Drawing$. La instrucción $this.Paint$ indica al compilador que se va a usar el evento $Paint$ que asigna las características de la presentación de gráficos dentro del formulario, el evento $Paint$ forma parte de la clase $Form$.

Con el signo += se instala un controlador de eventos adjuntando el método $Dibujar\_Graficas$ que permite el control de un evento tipo $Paint$. La palabra $PaintEvenHandler$ es un delegado que se define en el espacio de nombres $System.Windows.Forms$, con esta instrucción se define en la clase un método que tiene los mismos argumentos y tipos que el delegado, su argumento hace referencia al objeto sobre el que se aplica el evento $Paint$ que en este caso es el objeto $Transformada$. Después de crear el formulario en $Main()$, el método $Dibujar\_Graficas$ se adjunta al evento $Paint$ del Formulario, el código de las ins-

trucciones descritas es el siguiente:

```
009  public Transformada() {
010    this.Text = "Transformada de Fourier";
011    this.Size = new Size(650,510);
012    this.Paint += new PaintEventHandler(Dibujar_Graficas);
013  }
```

Interfaz gráfica

Dentro del siguiente bloque de código se encuentran todas las instrucciones que generan la interfaz de la presentación gráfica de los archivos WAV, la primera instrucción muestra dos argumentos del método $Dibujar\_Graficas$ pertenecientes al controlador de eventos $Paint$. El primero es *object sender* el cual hace referencia al objeto sobre el que se aplica el evento $Paint$ que es $Transformada$.

Mediante el segundo argumento se hace referencia a una clase de tipo $PaintEventArgs$ abreviada con la letra e, la cual define los métodos dentro del espacio de nombres $System.Windows.Forms$. La instrucción $Graphics\ g = e.Graphics$ le indica al compilador que se utilizará la clase $Graphics$ que se encuentra definida en el espacio de nombres $System.Drawing.Graphics$, la cual se utiliza para dibujar gráficos y texto en los formularios. La letra g se asigna como un nombre para los objetos gráficos, a continuación se presenta el código de las instrucciones mencionadas:

```
014  public void Dibujar_Graficas(object sender, PaintEventArgs e) {
017    Graphics g = e.Graphics;
       ...INSTRUCCIONES PARA EL DIBUJO DE LAS AREAS DE TRABAJO...
387  }
```

Todas las instrucciones empleadas en el dibujo de la interfaz gráfica y la presentación gráfica de los archivos WAV se encuentran en los campos $System.Drawing$

y $System.Drawing.Drawing2D$ [12, p:71]. Las áreas de trabajo de la interfaz gráfica están delimitadas por cuatro rectángulos de color azul. Para dibujar un rectángulo se emplea la instrucción $g.DrawRectangle(new\ Pen(Color.Tipo), Cord\ X, Cord\ Y, Ancho, Alto)$. La instrucción $new\ pen(Color.Tipo)$ crea un objeto gráfico con el color especificado, el objeto se genera con un ancho de un píxel si no se establece un valor dentro del argumento. Los argumentos $Cord\ X$ y $Cord\ Y$ son las coordenadas de inicio del rectángulo dentro del formulario, los argumentos *Ancho* y *Alto* son el valor del número de pixeles para cada lado del rectángulo.

Dentro de los rectángulos primero, segundo y cuarto se delimitan las áreas de información de los archivos WAV. El primero se encuentra en el lado izquierdo del formulario, dentro de su área se colocan los indicadores de la cabecera del primer archivo WAV, estos se colocan en la misma secuencia como se encuentran dentro de la estructura de la cabecera del archivo (Tabla 2.6), se reserva un espacio vació para cada indicador en el cual se muestra el dato obtenido. El cuarto rectángulo esta ubicado en el otro extremo y muestra los indicadores de la cabecera para el segundo archivo.

En el interior del segundo rectángulo se muestra la ruta y el nombre de los dos archivos, en esta área se visualizan los mensajes de error proporcionados por el sistema de apertura y cierre de archivos, este rectángulo se dibuja en la parte superior del formulario. El tercer rectángulo se divide en cuatro secciones donde se muestran los procesos digitales que se realizan en las muestras de audio de los archivos WAV.

Cada nombre del identificador de la cabecera de un archivo WAV se visualiza por una cadena de caracteres, para mostrarla en el formulario se utiliza la instrucción $g.DrawString(cadena, Tipo\_de\_letra, Brushes.Color, Cord\ X, Cord\ Y)$, el argumento *cadena* se coloca entre comillas dobles, los tipos de letras empleados son $Font$ y $Verdana$, el argumento $Brushes.Color$ proporciona el estilo de textura y el color que tendrá el texto de la cadena, los argumentos $Cord\ X$ y $Cord\ Y$ indican las coordenadas de inicio de impresión de la cadena sobre el formulario. Una parte del código que genera los instrucciones mencionadas se muestra a continuación:

```
019  g.DrawRectangle(new Pen(Color.Blue),128,0,511,95);
061  g.DrawString("DATOS DEL ARCHIVO",Font,Brushes.Blue,5,10);
084  g.DrawString("Formato:",Font,Brushes.Black,5,100);
```

Dos polígonos en forma de rectángulos de color blanco, muestran la información del nombre y la ruta de los archivos WAV utilizados. Para generar un polígono se usa la instrucción $g.FillRectangle(new\ SolidBrush(Color.Tipo), CordX, CordY, Ancho, Alto)$. La instrucción $new\ SolidBrush$ genera un objeto gráfico con un estilo de relleno sólido del color especificado. Los argumentos siguientes se establecen de la misma forma que en la instrucción $g.DrawRectangle$, la instrucción que genera el polígono para la ruta y nombre del primer archivo se muestra a continuación:

```
040  g.FillRectangle(new SolidBrush (Color.White),140,30,170,30);
```

De la misma forma mencionada anteriormente se generan los rectángulos restantes de las áreas de trabajo y se colocan las cadenas de información de los archivos. El área donde se realiza la presentación gráfica de los archivos WAV esta dividida en cuatro secciones, la división se realiza por medio de tres líneas espaciadas de manera equidistante, las líneas tienen el mismo ancho y el mismo color que los rectángulos dibujados. La instrucción que genera una línea es $g.DrawLine(new\ Pen(Color.Estilo), Inicio\,X, Inicio\,Y, Fin\,X, Fin\,Y)$, la instrucción $new\ pen(Color.Estilo)$ genera el objeto gráfico del color establecido, los argumentos restantes son las coordenadas de inicio y fin de la línea. Si no se coloca un tipo de estilo de línea el compilador coloca un valor estándar.

Una referencia de cruce por cero se coloca en las dos primeras secciones por medio de una línea punteada de color negro colocada horizontalmente en la mitad de cada sección. Para generar la línea con un nuevo estilo se asigna el valor de $DashStyle$ al objeto gráfico. Las instrucciones siguientes muestran el uso de $g.DrawLine$ y un cambio de estilo de línea.

```
023  g.DrawLine(new Pen(Color.Blue,0),128,223,639,223);
027  Pen penline = new Pen (Color.Black, 1);
028  penline.DashStyle = DashStyle.Dash;
029  g.DrawLine(penline,129,159,639,159);
```

Las cuatro secciones que muestran la información de las señales de los archivos WAV se identifican con cadenas de caracteres en forma vertical, cada una muestra el contenido de cada sección y se identifican por medio del color correspondiente para cada archivo. La cadena para las primeras dos secciones es *Senal Original*, y de las dos restantes es *Armonicos TFD Hz*. Para representar en forma vertical las cadenas, se rota el estilo de trabajo del formulario en -90 grados [12, p:220], cualquier instrucción siguiente que ejecute una acción de dibujo trabaja en el ángulo especificado. Una vez que se colocan las cadenas en el lugar determinado se regresa el estilo de trabajo del formulario a su posición original. La instrucción que permite realizar esta acción es $g.RotateTransform(angulo)$, donde el argumento *angulo* determina el valor de giro del formulario, el código de las instrucciones mencionadas se muestra a continuación:

```
032  g.RotateTransform(-90);
033    g.DrawString("Senal Original",Font,Brushes.Red,
                    new Rectangle(-195, 113, 80, 40));
037  g.RotateTransform(90);
```

La instrucción *new Rectangle* establece un área delimitada por cuatro coordenadas en la que aparece el texto con la rotación especificada por el ángulo. El código implementado por las instrucciones en conjunto descritas hasta este punto constituyen la interfas gráfica, esta se muestra en la Figura 3.2.

Apertura de un Archivo WAV

El sistema de entrada y salida del lenguaje C# esta diseñado bajo una jerarquía de clases definidas en .NET Framework. Todas las instrucciones de entrada y salida

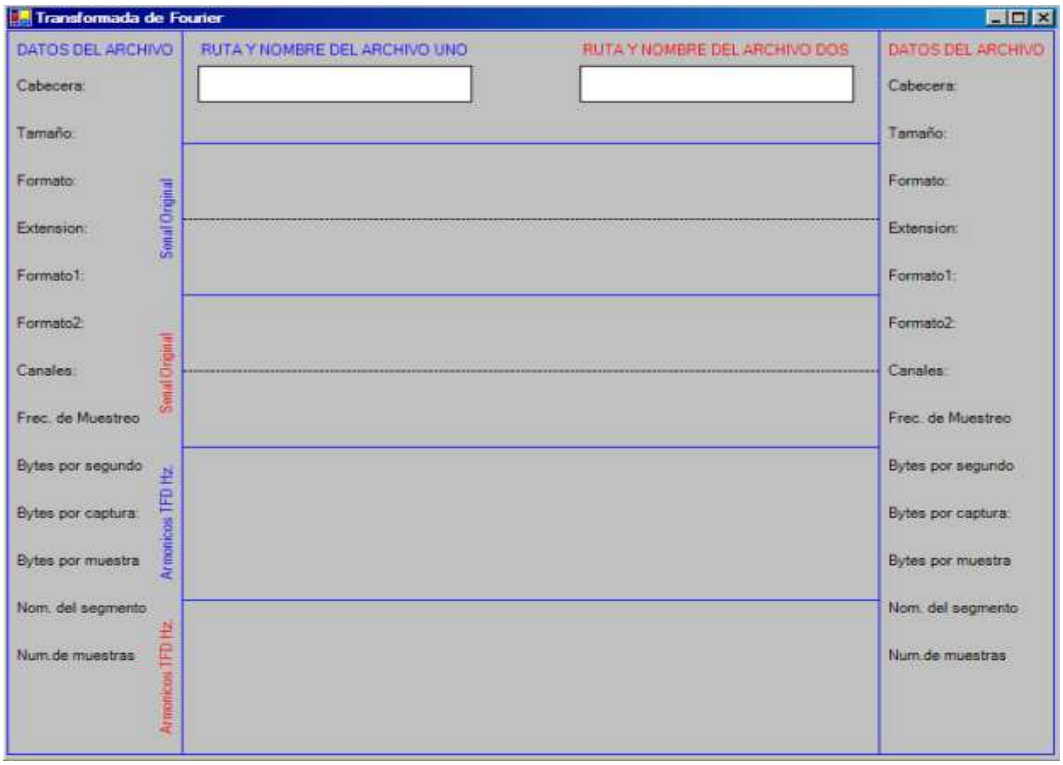

Figura 3.2: Interfaz gráfica

utilizan el espacio de nombres *System*. El lenguaje C# ejecuta instrucciones de entrada y salida mediante secuencias, todos los elementos de entrada y salida en C# operan en bytes [17, p:383]. Si las muestras de un archivo son un conjunto de caracteres ASCII, es necesario realizar una conversión entre el tipo *char* y el tipo *byte* debido a que *char* es un tipo de 16 bits y *byte* es un tipo de 8 bits, esto se realiza omitiendo el byte de orden superior del valor *char*.

Para abrir un archivo WAV se utilizan los métodos integrados dentro de la clase *System.IO*, los resultados de los procesos matemáticos realizados sobre las muestras se almacenan en arrays lineales, una ves almacenadas las muestras de los resultados, se grafican utilizando los métodos de la clase *System.Drawing.Drawing2D*.

Para poder obtener una secuencia de bytes de un archivo se crea un objeto del tipo *FileStream* con el nombre de *archivo*, el método que especifica la acción es *new FileStream*(*nombre*, *FileMode.Modo*), el argumento *nombre* especifica la ruta y el

nombre del objeto, en este caso del archivo WAV que se va a manipular, y *Modo* indica la acción que se realiza sobre el archivo, en este caso es *Open* que indica la lectura del archivo.

Si al momento de abrir un archivo se produce un error se inicia una excepción, si el archivo no existe se inicia $FileNotFoundException$ y a continuación se muestra el mensaje *Error en la ruta especificada*, delante del mensaje se muestra la ruta de búsqueda del compilador. Para cualquier otro tipo de error durante el proceso, se muestra el mensaje *Error al abrir el archivo*, ambas cadenas se implementan utilizando el comando $g.DrawString$ mostrado en la sección anterior. Para controlar las dos excepciones que se generan, se emplea un bloque *try/catch*, las instrucciones del código de apertura de un archivo con sus excepciones es el siguiente:

```
045  FileStream archivo;
046  try {
047    archivo = new FileStream(nombre,FileMode.Open);
048  }
049  catch (FileNotFoundException exc) {
050    g.DrawString("Error en la ruta especificada",Font,
                  Brushes.Red,140,70);
051    return;
052  }
053  catch {
054    g.DrawString("Error al abrir el archivo",Font,
                  Brushes.Black,152,70);
055  return;
056  }
```

Para almacenar las muestras se crea un array tipo $float$ llamado $MuestrasWAV$ con un tamaño de memoria para 1068 muestras. Las primeras 44 for-

man la cabecera del archivo y las 1024 restantes son muestras de audio del archivo WAV. Se utiliza el método *ReadByte*() que lee un único byte desde un archivo cada vez que se llama y lo devuelve como un valor entero. Debido a que las muestras que forman la cabecera son del tipo *char* y los resultados de los procesos matemáticos en la obtención de la información del archivo WAV son del tipo *int*, se realiza una conversión de tipos.

Para asignar las muestras que se obtienen del archivo en cada índice del array $MuestrasWAV$ se emplea un ciclo for. Las muestras obtenidas del archivo WAV son de 8 bits y por lo tanto forman un intervalo de 256 niveles [16, p:25], para tener un cero como referencia se resta el valor de 128 a cada muestra, el código que realiza lo anterior se muestra a continuación:

```
043  int i;
044  float [] MuestrasWAV = new float [1068];
057  for(i=0; i<1068; i++) {
058    MuestrasWAV[i]=archivo.ReadByte()-128;
059  }
```

Los identificadores de un archivo WAV del tipo caracter son *cabacera*, $formato$, *extension* y *nombre del segmento* (Tabla 2.6), sus elementos que los conforman se encuentran almacenados dentro de cada uno de los índices correspondientes dentro del array $MuestrasWAV$, para recuperarlos se asigna el valor de cada índice a una variable tipo *byte* con el mismo nombre que el identificador.

El proceso se realiza por medio de un ciclo for que recupera únicamente las muestras requeridas dentro del array $MuestrasWAV$. Cada valor que se recupera en un ciclo del bucle for, se muestra en la interfaz gráfica por medio de la instrucción $g.DrawString$ en el área respectiva de cada identificador. Se utiliza una conversión de tipo *char* dentro del argumento de cadena del método $g.DrawString$ para que el valor se muestre como un caracter. De la misma forma se obtienen los identificadores

restantes de tipo caracter, el código que muestra el identificador *cabecera* es el siguiente:

```
064  for(i=0;i<4;i++) {
065    byte Cabecera;
066    Cabecera=(byte) (MuestrasWAV[i]+128);
067    g.DrawString(""+(char)Cabecera,new Font("Verdana",8),
                    Brushes.Blue,62+(i*12),80);
068  }
```

Los identificadores de un archivo WAV del tipo numérico son *tamano*, $formato$ 1, $formato$ 2, *numero de canales*, $frecuencia\ de\ muestreo$, *bytes por segundo*, *bytes por captura*, *bytes por muestras* y *numero de muestras* (Tabla 2.6). Para mostrar el contenido de los identificadores numéricos se tienen que adjuntar los valores de cada byte individual que componen el campo del identificador en forma binaria, juntos representan el valor total del campo [16, p:30]. Su representación decimal muestra el contenido de cada grupo de 8 bits.

Un valor como 11,025 se representa en binario como 10 1011, 0001 0001, sus componentes binarios se dividen en 2 bytes. El primer byte es el valor binario de 0001 0001 el cual en su forma decimal es 17, el segundo byte es el valor binario 10 1011, el cual en su forma decimal es 43. Mediante un proceso similar a la obtención de los identificadores de caracteres, se obtienen las representaciones decimales de cada byte que componen un dato numérico.

Para representar el valor de los bytes obtenidos de manera conjunta, una vez que se obtienen del array $MuestrasWAV$, es necesario sumar el valor decimal del primer byte con el valor del segundo byte recorrido 8 posiciones a la derecha en su forma binaria. Como el primer bit del segundo byte vale 256, el segundo 512, y así susecivamente hasta llegar a 32,768 es necesario sumar el valor de cada bit que compone el segundo byte en una variable.

Para poder conocer el valor de cada bit se divide la cantidad del valor entero del segundo byte entre dos, se determina si existe un acarreo de 1 o 0, si existe el acarreo se suma el valor correspondiente para cada bit a una variable que contendrá la suma de todos los valores de cada bit [16, p:26]. Este proceso se hace ocho veces por medio de un ciclo for, cada ciclo del bucle representa un bit del segundo byte, el valor que se suma en cada acarreo se genera por una multiplicación acumulativa del valor 256 por 2, teniendo así el valor de cada bit en cada ciclo.

Se realiza una corrección en el caso de que en la ultima división existe un acarreo, esto se verifica obteniendo el resto de la división del valor del segundo byte entre dos, si hay acarreo se aplica la corrección y se suma el valor de 256 para complementar el valor exacto del identificador. El bloque de código que se muestra a continuación implementa el proceso descrito para calcular el valor entero del identificador del tamaño del archivo.

```
071  int Sumat=256,Condiciont=(int)(MuestrasWAV[5]+128),Bit2t=0,
         Tamaño=0,Correcciont=0;
072  float Restot=0;
073  for(i=0;i<8;i++) {
074    { Sumat=Sumat*2;
075      Restot=MuestrasWAV[5]%2;
076      Condiciont=Condiciont/2;
077      if ((Condiciont%2)==1) {Bit2t=Bit2t+Sumat;}
078     }
079  }
080  if(Restot==1) {Correcciont=256;}
081  Tamaño=Bit2t+(int)(MuestrasWAV[4]+128)+Correcciont;
082  g.DrawString(""+Tamaño,Font,Brushes.Blue,60,100);
```

De la misma forma descrita anteriormente se obtienen el resto de los identifi-

cadores de los dos archivos WAV y se colocan en su posición correspondiente dentro de la interfaz gráfica.

Para graficar las muestras del archivo WAV se declaran cuatro variables del tipo float, que representan los valores de las coordenadas de dos muestras consecutivas enlazadas por una línea. Las coordenadas de la primera muestra son $x0$ y $y0$, y las coordenadas de la segunda son x y y, para dibujar la línea que las une se utiliza la función $g.DrawLine$, las coordenadas de la primera muestra recuperada indican el comienzo de la línea y las de la segunda el final de la línea. Se comienzan a graficar en las coordenadas de referencia iniciales localizadas a la mitad de las secciones *Senal Original* para cada archivo dentro de la interfaz gráfica.

Mediante un ciclo for se incrementa el valor en el eje x en una unidad, y al mismo tiempo se recupera el valor de la muestra almacenada en el array $MuestrasWAV$ el cual corresponde al valor en el eje y. Un archivo WAV se gráfica a parir de la muestra numero 45 ya que los datos desde esta muestra en adelante son únicamente muestras de audio del archivo, sin embargo existen archivos WAV que contienen información adicional de su cabecera hasta la muestra 57, por esta razón, en el software desarrollado las muestras de los archivos se comienzan a graficar a partir de la muestra 58.

Los archivos WAV con los que se trabaja en esta tesis tienen muestras de 8 bits, por lo tanto están compuestos de 256 niveles en el eje y, debido a que el espacio reservado para graficar las muestras dentro de la interfaz solo tiene 128 pixeles, el valor de las muestras se divide entre dos, el código que realiza la graficación del primer archivo se muestra a continuación:

```
182  float x0,y0,x,y;
183  x=128; y=159;
184  for(i=58; i<570; i++){
185    x0=x;
186    x=128+(i-58);
```

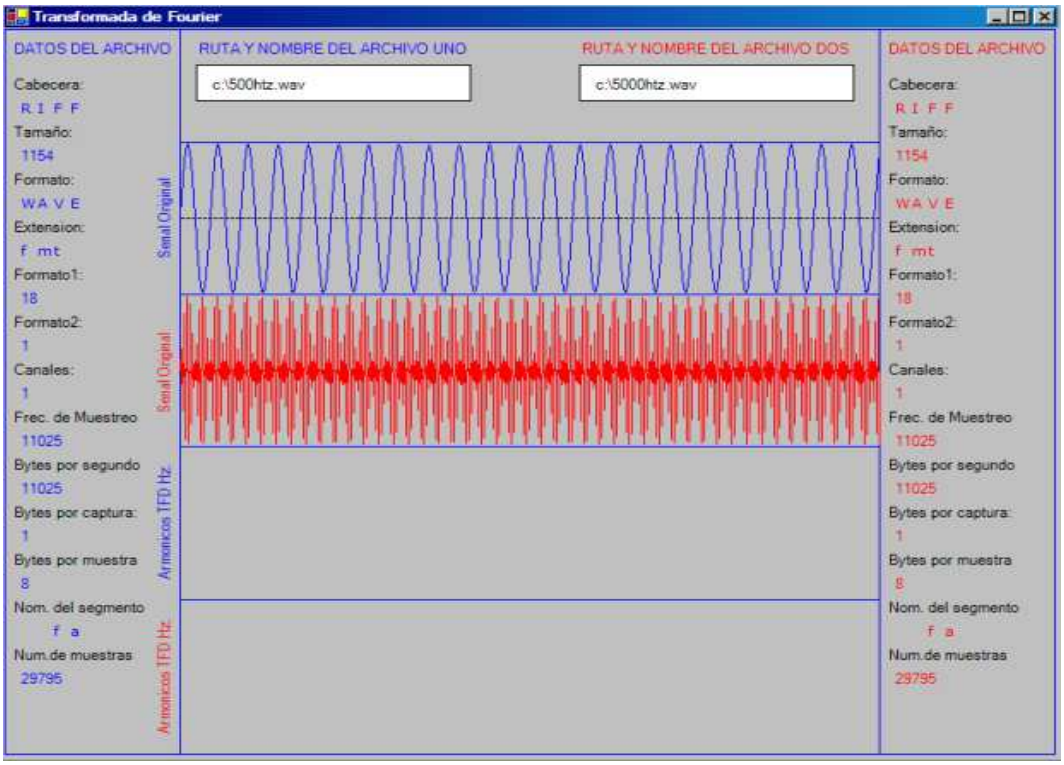

Figura 3.3: Indicadores y gráficas de dos archivos WAV

```
187     y0=y;
188     y=159-((MuestrasWAV[i])/2);
189     g.DrawLine(new Pen(Color.Blue,0),x0,y0,x,y);
190  }
```

La Figura 3.3 muestra el resultado del código generado para abrir dos archivos WAV, el primero contiene un armónico de 500 Hz, y el segundo un armónico de 5000 Hz. En la figura se observan los indicadores obtenidos de las cabeceras de los dos archivos y las graficas de sus muestras en las secciones *Senal Original* respectivamente para cada archivo dentro la interfaz gráfica.

Ventana de Hamming

Las muestras almacenadas de los archivos WAV, se convolucionan con un filtro ventana de Hamming. La ventana de Hamming se genera a partir de la ecuación

definida en la Tabla 2.4, el número de muestras que componen la ventana es 512 por ser el mismo número de muestras que se grafican del archivo WAV. Las muestras de la ventana se guardan en un array tipo $float$ con el nombre de $Ventana$ mediante un ciclo for, los valores de la ventana están comprendidos en el intervalo entre 0 y 1, el código que genera y almacena las muestras de la ventana de Hamming se muestra a continuación:

```
192  float[] Ventana = new float[1068];
193  for(i=58; i<570; i++) {
194    Ventana[i]=(float)(.54-(.46*Math.Cos((2*3.14*(i-58))/512)));
195  }
```

Las muestras almacenadas de la ventana de Hamming se convolucionan en tiempo con las muestras de la señal del archivo WAV, esto se logra multiplicando el valor del contenido dentro de cada índice del array $MuestrasWAV$ por el valor contenido en el índice correspondiente del array $Ventana$, el resultado de cada muestra de la multiplicación se almacena en un array tipo $float$ llamado $Muestras$, el código que realiza la convolución de las muestras es el siguiente:

```
197  float[] Muestras = new float[1068];
198  for(i=58; i<570; i++){
199    Muestras[i]=Ventana[i]*MuestrasWAV[i];
200  }
```

Transformada de Fourier Discreta

La obtención de la Transformada de Fourier Discreta (TFD) de las muestras almacenadas en el array $Muestras$ proporciona el espectro en frecuencia de la señal (Sección 2.2.3), las muestras que se obtienen son almacenadas en un array tipo $float$ con el nombre de $Modulo$. Se crean las variables $Real$ y $Imag$ de tipo $float$, la variable $Real$ es la parte real del componente y la variable $Imag$ es la parte imaginaria.

Todas las funciones matemáticas empleadas se encuentran dentro de la clase *Math* la cual pertenece al espacio de nombres *System* [17, p:523]. Los métodos definidos por esta clase son del tipo *static* y los valores de los ángulos están dados en radianes, los valores que proporcionan los métodos de la clase *Math* son del tipo *double*, por eso es necesario realizar una conversión de tipo a *float* para cada bloque de instrucciones. Los métodos utilizados en el algoritmo de la TFD son *Math.PI* que proporciona el valor de Pi, *Math.Cos* el valor del coseno de su argumento, *Math.Sin* el valor del seno de su argumento y *Math.Sqrt* el valor de la raíz cuadrada de su argumento.

El proceso se realiza para un intervalo de 512 muestras, para graficar cada muestra del resultado de cada paso del algoritmo de la TFD se utiliza la función *g.FillRectangle* (*new SolidBrush*(*Color.Tipo*), *Cord X*, *Cord Y*, *Ancho*, *Alto*) que dibuja un rectángulo, el objeto gráfico que se crea en la instrucción *new SolidBrush* indica la forma del estilo con que se dibuja su área, el argumento *Color.Tipo* dentro de los paréntesis establece el color del objeto gráfico, los argumentos *Cord X* y *Cord Y* indican las coordenadas de inicio del objeto gráfico, los argumentos *Ancho* y *Alto* indican el número de pixeles correspondientes para cada valor.

Esta instrucción se usa para que el espectro en frecuencia se dibuje de manera continua, proporcionando un ancho a cada rectángulo que permita cubrir todos los pixeles entre muestras consecutivas. Una línea de color azul se utiliza como eje de referencia horizontal en la gráfica del espectro, el código que realiza el algoritmo de la TFD en las muestras del array *Muestras* es el siguiente:

```
202  float Real,Imag;
203  float [] Modulo = new float[1068];
204  int N=512,n,k;
205  for(n=58; n<314; n++){
206    Real=0;
```

```
207     Imag=0;
208     for(k=58; k<570; k++){
209       Real=Real+(MuestrasWAV[k]*(float)
                    Math.Cos((-2*Math.PI*(n-58)*(k-58))/N));
210       Imag=Imag+(MuestrasWAV[k]*(float)
                    Math.Sin((-2*Math.PI*(n-58)*(k-58))/N));
211     }
212     Modulo[n]=(float)(Math.Sqrt((Real*Real)+(Imag*Imag)));
213     g.DrawLine(new Pen(Color.Blue,0),128,460,639,460);
214     g.FillRectangle(new SolidBrush(Color.Blue),128+((n-58)*2),
                        460-(Modulo[n]/285),2,Modulo[n]/285);
215   }
```

Para indicar el valor de las frecuencias en la gráfica de la TFD, se coloca una escala de valores numéricos en intervalos igualmente espaciados en forma de cadenas de caracteres debajo de la línea de referencia. El valor para cada intervalo se calcula dividiendo la mitad de la frecuencia de muestreo del archivo WAV que se esta analizando en diez intervalos iguales, se usa la mitad de la frecuencia porque el espectro de la TFD únicamente representa el intervalo de la mitad de la frecuencia de muestreo, siendo este valor la resolución máxima de la TFD. La posición de las cadenas de caracteres en la interfaz gráfica, se obtiene al dividir el número total de pixeles del eje horizontal entre diez intervalos, tomando cada píxel resultante como el valor de la coordenada x que se utiliza en una instrucción $g.DrawString$ se colocan los valores en la interfaz.

Los valores de los intervalos de frecuencia y su posición en el eje x se calculan mediante un ciclo for, estos son asignados a una variable tipo $float$ de nombre eje, la misma variable de iteración del bucle incrementa el valor de la coordenada x que asigna el número del píxel en el eje x, esto se logra al multiplicar el valor de i por 50, teniendo así los valores para los 10 intervalos, el código de las instrucciones descritas

es el siguiente:

```
216  float eje;
217  for(i=0;i<10;i++){
218    eje=(int)(i*((Frecmues/2)/10));
219    g.DrawString(""+eje,Font,Brushes.Black,129+(i*50),465);
220  }
```

En la Figura 3.4 se presentan las gráficas de los espectros en frecuencia de los archivos WAV de la Figura 3.3, estos se grafican en colores iguales a los indicadores y gráficas de cada archivo. Cuando se terminan de obtener los datos de los archivos y se finaliza el procesamiento digital, se indica al compilador que cierre los archivos mediante la instrucción *archivo.Close*().

```
archivo.Close();
```

3.2.2. Características físicas de una flauta

El software desarrollado en esta sección es una herramienta computacional que calcula el valor de la longitud a la cual se colocan los orificios que generan las notas musicales en el instrumento propuesto, el programa es una aplicación tipo formulario similar a la de la sección anterior. Su interfaz gráfica se encuentra dividida en dos áreas de trabajo, la primera muestra una tabla con cinco columnas, las cuales contienen el nombre de los intervalos de una octava, la razón para cada intervalo, la frecuencia de cada nota, el nombre de la nota y la posición desde la boquilla hasta la posición de los orificios que generan las notas musicales. En la segunda, se dibuja el instrumento con sus respectivos orificios y se presentan las características físicas del material utilizado para su construcción.

Se utilizan los mismos espacios de nombres que en la aplicación anterior, el nombre de la clase principal es *Flauta*, el programa genera una aplicación tipo formulario empleando las mismas librerías dinámicas y funciones de dibujo mencionadas en

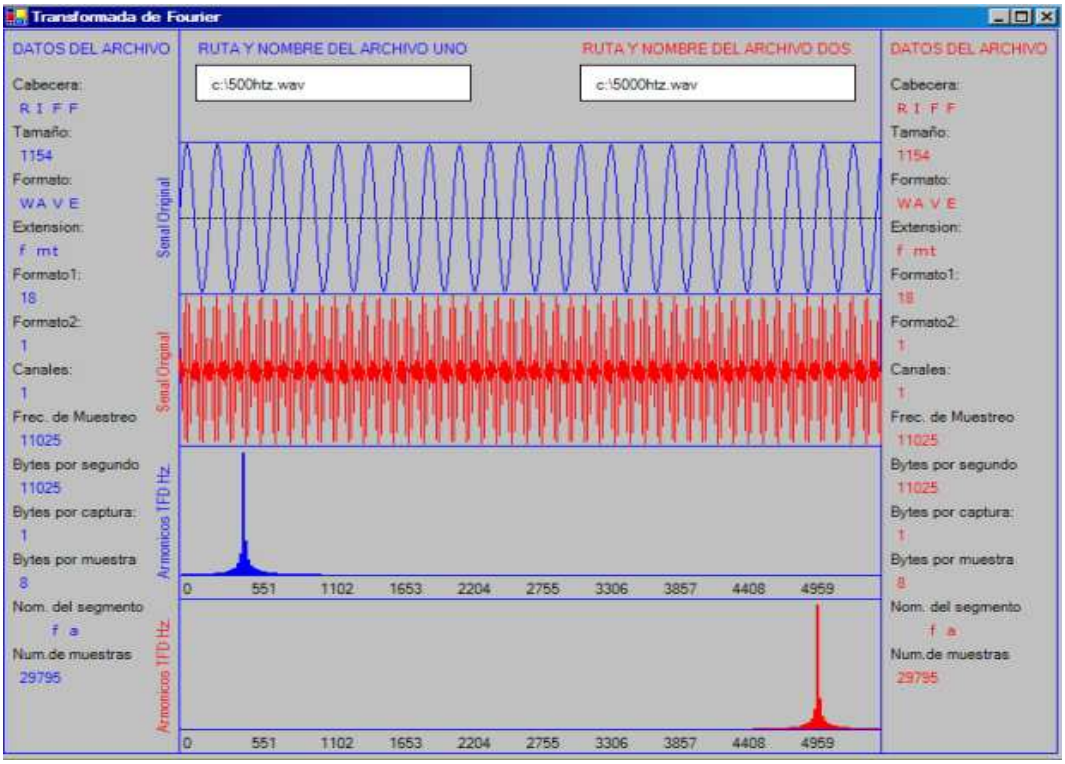

Figura 3.4: Espectros en frecuencia de los archivos de la Figura 3.3

la sección 3.2.1, las primeras 15 líneas de instrucciones de código son iguales a la aplicación anterior y realizan las mismas acciones, solo varia el titulo del formulario que es *Flauta en G* y sus dimensiones que son 550 pixeles de largo por 490 pixeles de alto, las siguientes instrucciones generan la aplicación.

```
000  using System;
001  using System.IO;
002  using System.Drawing;
003  using System.Windows.Forms;
004  using System.Drawing.Drawing2D;
005  public class Flauta:Form{
006    public static void Main(){
007      Application.Run(new Flauta());
008    }
```

```
009    public Flauta(){
010      this.Text = "Flauta en G";
011      this.Size = new Size(570,490);
012      this.Paint += new PaintEventHandler(Draw_Graphics);
013    }
014    public void Draw_Graphics(object sender,PaintEventArgs e){
015      Graphics g = e.Graphics;
         ...CONTENIDO DEL PROGRAMA...
111    }
112  }
```

Las características físicas del material con que se construye el instrumento se declaran al principio del programa como variables tipo *double* y son utilizadas durante la ejecución a lo largo del programa, estas son la velocidad del viento a la temperatura ambiente (c), el radio del tubo donde se construye el instrumento (RT), el radio de los orificios que generan las notas musicales ($RadOri$), el espesor del tubo (L) y la frecuencia limite de resonancia del instrumento ($G4$). Los valores de las características físicas se muestran en la parte inferior del formulario mediante la instrucción $g.DrawString()$, a continuación se muestra parte del código que genera lo descrito.

```
017  double RadOri,c,L,G4,RT;
018  c = 343.3453734;           //Velocidad del sonido a 20 grados
019  RT=0.0127;                 //Radio del tubo
020  RadOri = 0.00555625;       //Radio del orificio
021  L = 0.0015875;             //Espesor del tubo
022  G4=391.995436;             //Frecuencia limite
023  g.DrawString("Radio del Tubo:",Font,Brushes.Black,5,420);
024  g.DrawString(""+RT+" m",Font,Brushes.Blue,90,420);
```

La información del diseño se muestra en cinco columnas, la primera tiene como

titulo $INTERVALOS\ DE\ UNA\ OCTAVA$, esta muestra los nombres de los intervalos de una octava (Tabla 2.1), se colocan utilizando la instrucción $g.DrawString()$, parte del código que genera la columna es el siguiente.

```
053  g.DrawString("INTERVALOS DE UNA OCTAVA",Font,Brushes.Blue,5,10);
054  g.DrawString("Unitono",Font,Brushes.Black,5,40);
055  g.DrawString("Semitono o Segunda Menor",Font,Brushes.Black,5,60);
```

La segunda columna tiene como titulo $RAZON$, muestra la razón de frecuencia para cada nota de una octava en la escala de igual temperamento (Tabla 2.1), los 12 intervalos se calculan mediante un ciclo for, se declara la variable In de tipo *double* en la cual se almacena el valor del intervalo, se utiliza la ecuación que calcula los intervalos de la escala de igual temperamento (Sección 2.1.1). Para implementar el algoritmo de la ecuación se utiliza la instrucción $Math.Pow(base, exponente)$, cada uno de los intervalos proporciona el valor de cada razón, estos valores se muestran en la segunda columna en forma de cadenas de caracteres, a continuación se muestra el código que realiza lo descrito.

```
037  float i;
038  double In,frec,pi=3.141592654,S,c2,Lef,w2,V,h;
039  for(i=0; i<13; i++){
040    In = Math.Pow(2.0,i/12.0);
048    g.DrawString(""+(float)In,Font,Brushes.Blue,210,40+(i*20));
051  }
```

Con el valor almacenado en la variable In, se calcula el valor de la frecuencia de cada nota musical del instrumento al multiplicar cada valor del intervalo calculado por la frecuencia limite $G4$. Se emplea el bucle for utilizado para calcular la razón de cada intervalo de la octava, el valor obtenido se almacena en la variable $frec$ de tipo *double* y se muestra en forma de cadena de caracteres en la tercera columna, esta tiene el titulo de $FRECUENCIA$, el código siguiente implementa lo descrito.

```
039  for(i=0; i<13; i++){
041    frec=(float)(G4*In);
049    g.DrawString(""+(float)frec,Font,Brushes.Blue,290,40+(i*20));
051  }
```

Para calcular la posición de los orificios se obtiene el volumen del resonador que genera la frecuencia de la nota requerida, para esto se utiliza el bucle for que calcula el valor de la razón y la frecuencia para cada nota. Con la ecuación del volumen de un cilindro se determina la posición en la cual tiene que ser perforado el orificio para generar la nota musical correspondiente de cada frecuencia, esto se hace utilizando el procedimiento de la sección 3.1.2.

Las variables de la ecuación para calcular el volumen se declaran de tipo *double*, la velocidad del sonido al cuadrado es $c2$, el área del orificio es S, la frecuencia en radianes al cuadrado es $w2$, el volumen del resonador es V y el valor de la posición del orificio es h, el valor de la posición se muestra en la quinta columna en forma similar a las columnas anteriores, esta tiene el titulo de $POSICION$. Para calcular las variables descritas se utilizan las variables de las características físicas del material empleado declaradas al principio del programa, las líneas de código siguientes muestran lo descrito.

```
038  double In,frec,pi=3.141592654,S,c2,Lef,w2,V,h;
039  for(i=0; i<13; i++){
042    S = pi*RadOri*RadOri;
043    c2 = c*c;
044    Lef = L+(1.45*RadOri);
045    w2 = (2*pi*frec)*(2*pi*frec);
046    V = (S*c2)/(Lef*w2);
047    h = V/(pi*RT*RT);
050    g.DrawString(""+(float)h,Font,Brushes.Blue,440,40+(i*20));
```

```
051  }
```

La cuarta columna tiene el titulo de $ORIFICIO$, muestra las notas que genera cada orificio del instrumento en notación musical, parte del código que genera la columna se muestra a continuación.

```
069  g.DrawString("ORIFICIO",Font,Brushes.Black,370,10);
070  g.DrawString("G4",Font,Brushes.Blue,370,40);
071  g.DrawString("Ausente",Font,Brushes.Blue,370,60);
072  g.DrawString("A4",Font,Brushes.Blue,370,80);
```

El área donde se colocan las cinco columnas con la información del diseño del instrumento esta delimitada por un rectángulo en color azul, este se genera mediante la instrucción $g.DrawRectangle()$. En la parte inferior del formulario, se dibuja un esquema del instrumento con la posición de cada uno de los orificios con el nombre de la nota que genera y la boquilla del instrumento. Las instrucciones de dibujo utilizadas son $g.DrawRectangle()$ y $g.DrawEllipse()$, el bloque de código siguiente muestra una parte de las instrucciones mencionadas.

```
090  g.DrawRectangle(new Pen(Color.Blue),0,0,540,300);
093  g.DrawRectangle(new Pen(Color.Black),60,370,450,30);
097  g.DrawEllipse(new Pen(Color.Black),215,375,20,20);
098  g.DrawString("G5",Font,Brushes.Blue,215,350);
```

La longitud del instrumento se calcula por medio de la ecuación de resonancia en tubos abiertos por ambos extremos (Sección 2.1.3), se calcula la longitud efectiva y con ese valor se hace la corrección sin presencia de pestaña para obtener la longitud real del instrumento. Se declaran las variables $LefecT$ y $LongT$ respectivamente para cada variable mencionada, con los datos del material utilizados se calcula la longitud real (Sección 3.1.2), el valor obtenido se coloca en forma de una cadena de caracteres en la parte inferior del dibujo del instrumento, el código siguiente muestra lo descrito.

Figura 3.5: Diseño de la Flauta en G

```
084  float LongT,LefecT;
085  LefecT=(float)(c/(2.0*G4));
086  LongT=(float)(LefecT-(0.6*RT));
087  g.DrawString("Longitud Total:",Font,Brushes.Black,400,420);
088  g.DrawString(""+LongT,Font,Brushes.Blue,480,420);
```

Todas las instrucciones en conjunto antes descritas generan la interfaz gráfica del programa, esta se muestra en la Figura 3.5.

3.3. Resumen

Este capitulo esta dividido en dos secciones. La primera describe el diseño del instrumento propuesto, primero se elige la octava que ejecuta, la clave de afinación y el material de construcción. Con los datos seleccionados se obtiene la longitud del instrumento a partir de la ecuación de resonancia en tubos, se calcula el volumen de los resonadores Helmholtz para cada nota y la posición de los orificios a lo largo del instrumento, por último se muestra el instrumento elaborado.

La segunda sección muestra en dos apartados el software desarrollado en lenguaje C# para la caracterización y el diseño del instrumento. El primero describe el programa que obtiene la información contenida en dos archivos WAV, esta diseñado para trabajar con archivos de 8 bits con cualquier frecuencia de muestreo. La interfaz gráfica muestra el contenido de sus cabeceras, las graficas sus primeras 512 muestras y utiliza un filtro ventana de Hamming para graficar los espectros en frecuencia, empleando la Transformada de Fourier Discreta.

En el segundo apartado se describe el programa que obtiene las dimensiones del instrumento a partir de las características del material elegido para su construcción, calcula la longitud del instrumento y la posición de los orificios que generan las notas. Los resultados obtenidos se muestran en una interfaz con cinco columnas, que contienen los intervalos de la octava, la razón para cada intervalo, las frecuencias para cada nota, su notación musical y la posición de los orificios a lo largo del instrumento, en la parte inferior se dibuja el instrumento y se muestran las características del material utilizado en su construcción.

Capítulo 4

Pruebas y Resultados

Para caracterizar el instrumento se hace la comparación del espectro en frecuencia de las notas que genera con el espectro de notas generadas digitalmente cuyos valores de frecuencia coinciden con los calculados en su diseño. Se utiliza un generador de señales digitales en formato WAV para generar las notas de la Tabla 3.2, las notas que genera el instrumento elaborado se capturan previamente en archivos con formato WAV. Por medio del software desarrollado se comparan las características de los dos archivos, el contenido de sus cabeceras, las gráficas de sus primeras 512 muestras y las gráficas de sus espectros en frecuencia. Al comparar los espectros de ambas notas dentro del intervalo de frecuencias definido, se verifica la afinación del instrumento observando la posición del espectro de la nota ideal con respecto a la posición del espectro de la nota generada por el instrumento.

Un instrumento musical genera en cada nota un armónico principal y un número indefinido de armónicos secundarios, la suma de todos los armónicos constituye el timbre del instrumento, el timbre está determinado por el tamaño, la forma y el material de construcción de cada instrumento. El timbre del instrumento elaborado se compara con el de tres instrumentos musicales de viento similares, el primero es una quena sudamericana de fabricación artesanal construida en madera de cedro, el segundo es una flauta dulce de fabricación industrial elaborada en plástico y el tercero es una tarka sudamericana de construcción rustica construida en madera de pino. Los ins-

trumentos descritos están afinados en claves distintas y ejecutan diferentes octavas, pero coinciden en la nota $C5$. Se emplean archivos WAV previamente capturados de la nota $C5$ generada por los cuatro instrumentos para comparar el timbre del instrumento elaborado en relación a los otros.

4.1. Pruebas de caracterización

En la Figura 4.1 se muestra la caracterización para la nota $G4$ del instrumento, en el lado izquierdo del formulario se observa el contenido de la cabecera del archivo de la nota generada digitalmente, de forma similar en el lado derecho del formulario se muestra el contenido de la cabecera del archivo de la nota generada por el instrumento. En la parte superior central del formulario se describe la ruta de almacenamiento y el nombre para cada uno de los archivos. La información de sus muestras y sus espectros se grafican en los cuadros de la parte central del formulario, en el primer cuadro están las muestras de la nota generada digitalmente y en el tercero la gráfica de su espectro en frecuencia, el segundo cuadro contiene las muestras de la nota generada por el instrumento y el cuarto su espectro en frecuencia. De forma similar se hace la caracterización para las notas restantes, en la Figura 4.2 para $A4$, en la Figura 4.3 para $B4$, en la Figura 4.4 para $C5$ y en la Figura 4.5 para $D5$.

En la caracterización para cada nota se observa que el armónico principal de su espectro en frecuencia corresponde en forma aproximada al armónico de la nota generada digitalmente, el valor del error para cada nota se muestra en la Tabla 4.1. Las posibles fuentes del error obtenido pueden ser que el procedimiento de cálculo no es el adecuado, el micrófono utilizado para capturar las señales de prueba no tiene las condiciones mecánico acústicas apropiadas, la ejecución de la nota musical por el músico no es adecuada y que el instrumento no esta elaborado con las medidas exactas obtenidas en el cálculo. Se observan en las gráficas armónicos de menores magnitudes complementarios al armónico principal tanto en frecuencias inferiores como superiores, estos representan el timbre característico del instrumento.

Tabla 4.1: Error en las notas del instrumento elaborado

Nota	Error (%)
G4	5.6
A4	0.0
B4	4.3
C5	8.3
D5	11.1

4.2. Resultados espectrales

En la Figura 4.6 se muestra la imagen de la quena sudamericana que se utiliza para comparar su timbre característico con el del instrumento elaborado, en la Figura 4.7 se presentan los resultados obtenidos al evaluar $C5$ en ambos instrumentos, el primer archivo contiene la información de la nota del instrumento elaborado y el segundo el de la quena, el resultado muestra una similitud en sus componentes armónicos de sus espectros en frecuencia.

La Figura 4.8 muestra la imagen de la flauta dulce utilizada para la comparación de su timbre característico con el del instrumento elaborado y la Figura 4.9 la comparación del espectro en frecuencia de la notas $C5$ de ambos instrumentos. Los resultados muestran que la nota generada por la flauta tiene un componente espectral formado por cuatro armónicos principales, de los cuales el segundo coincide con el armónico del instrumento elaborado.

La Figura 4.10 muestra la imagen de la tarka sudamericana empleada para comparar su timbre característico con el del instrumento elaborado y la Figura 4.11 la comparación de los espectros de la nota $C5$ de ambos instrumentos, debido a las características físicas de la tarka la nota C que genera se encuentra una octava abajo de la nota del instrumento elaborado. El resultado verifica que el armónico principal del espectro de la tarka se encuentra una octava por debajo del armónico principal del instrumento elaborado, y que el segundo armónico mas representativo del componente espectral de la tarka se encuentra a la misma frecuencia que el armónico

principal del instrumento desarrollado.

4.3. Resumen

Este capítulo esta divido en dos secciones. La primera muestra los resultados de las pruebas de caracterización realizadas al instrumento desarrollado en este proyecto de tesis, se compara el espectro en frecuencia de las notas generadas por el instrumento con el espectro de las notas construidas en forma digital cuyos valores de frecuencia coinciden con los cálculos de diseño del instrumento utilizando el software desarrollado, con esta prueba se verifica la afinación de las notas y se obtiene el error para cada una.

En la segunda sección se presenta la comparación del timbre del instrumento desarrollado con el timbre de tres instrumentos musicales de viento similares. Se utilizan una quena sudamericana, una flauta dulce y una tarka sudamericana. Las notas $C5$ que generan los cuatro instrumentos coinciden en su valor de frecuencia, estas notas se capturan previamente en formato WAV y se comparan espectralmente con el software desarrollado. El espectro en frecuencia de las notas analizadas representa el timbre característico de cada instrumento musical.

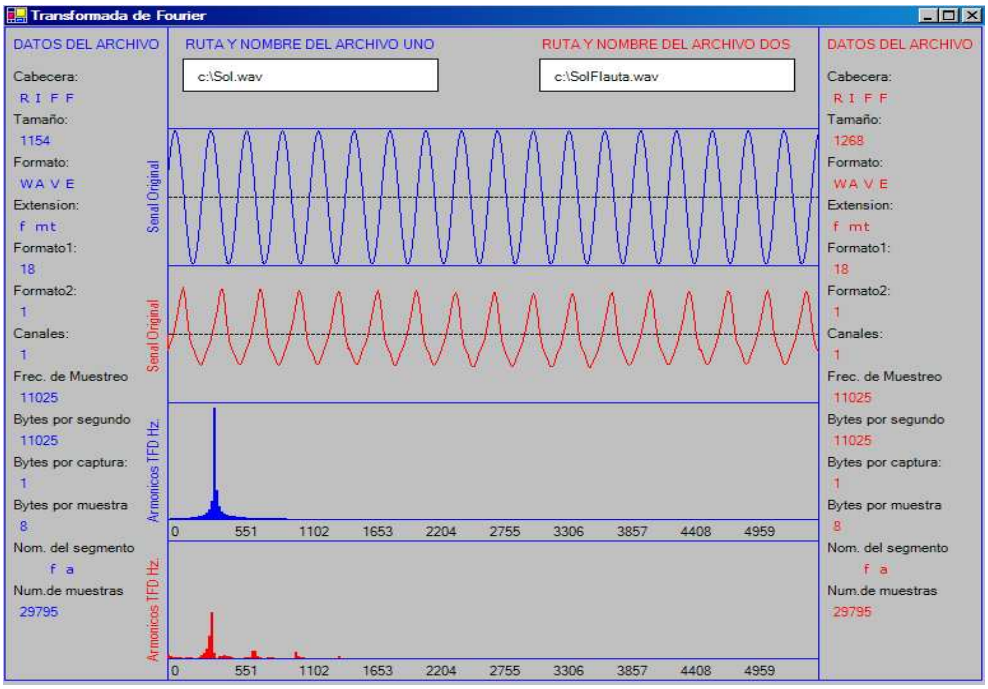

Figura 4.1: Caracterización de G4

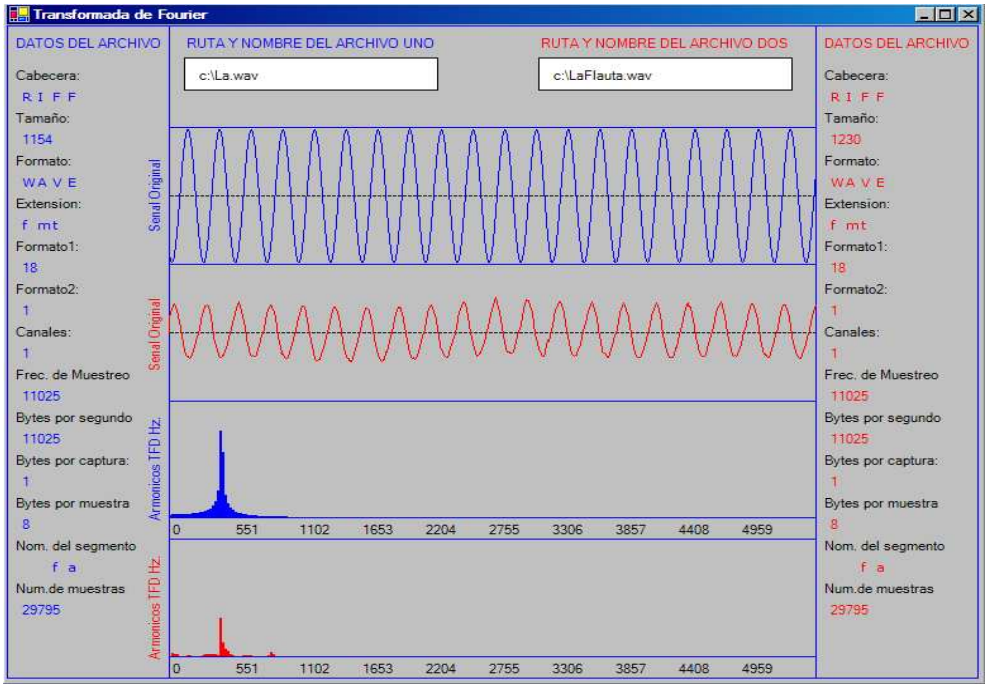

Figura 4.2: Caracterización de A4

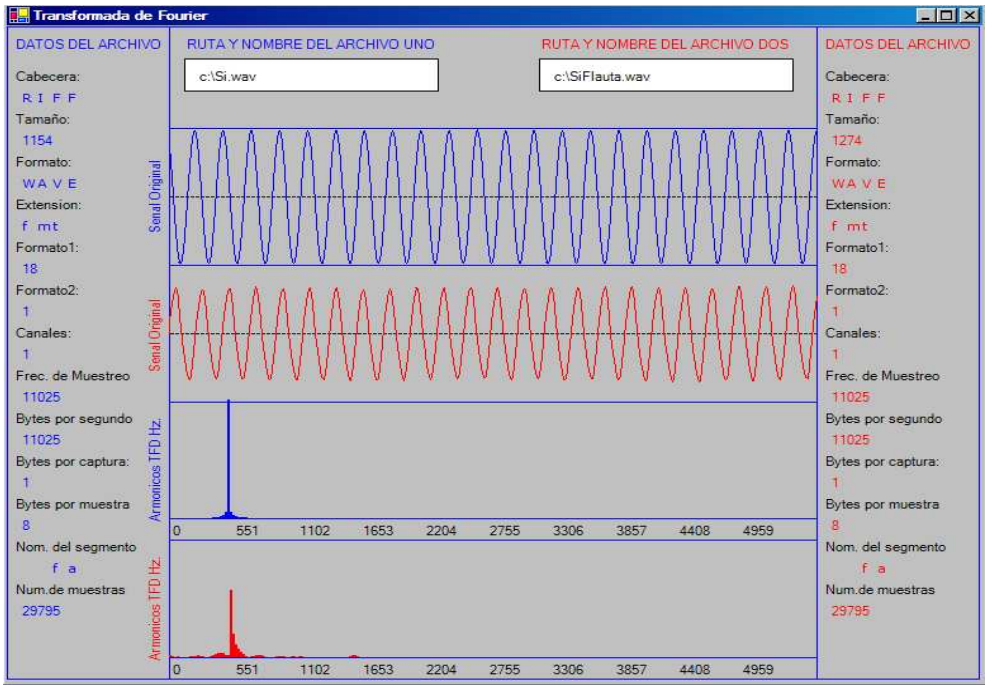

Figura 4.3: Caracterización de B4

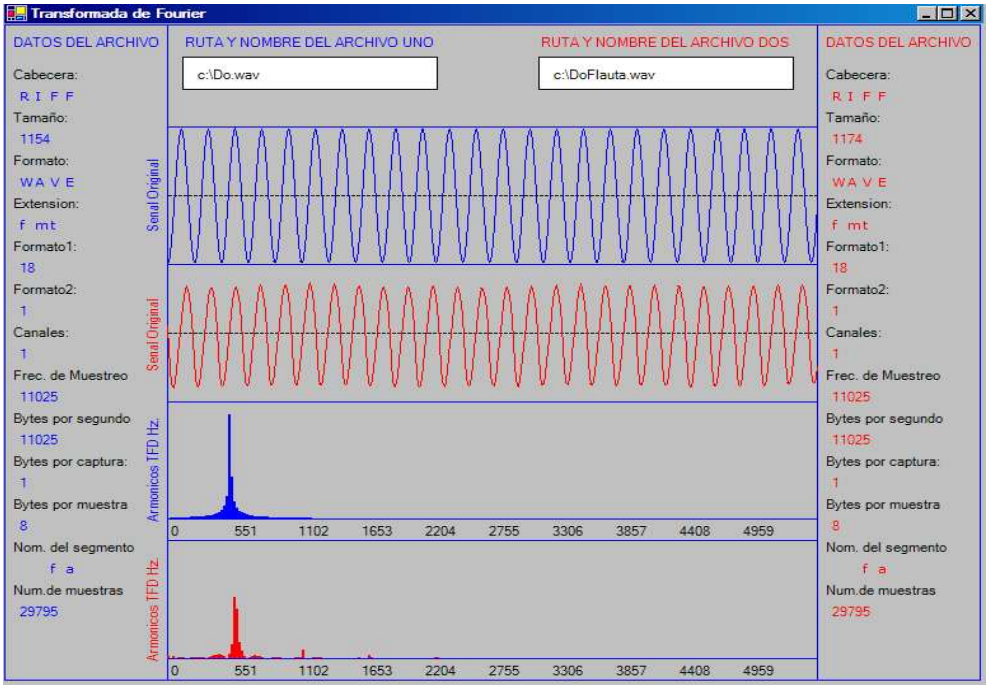

Figura 4.4: Caracterización de C5

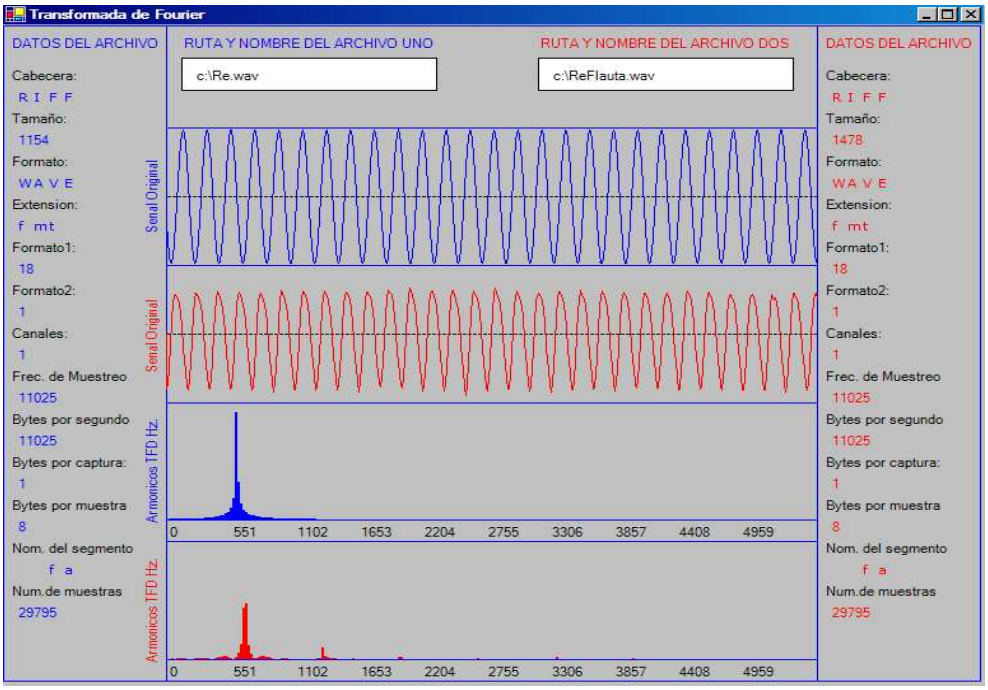

Figura 4.5: Caracterización de D5

Figura 4.6: Quena sudamericana

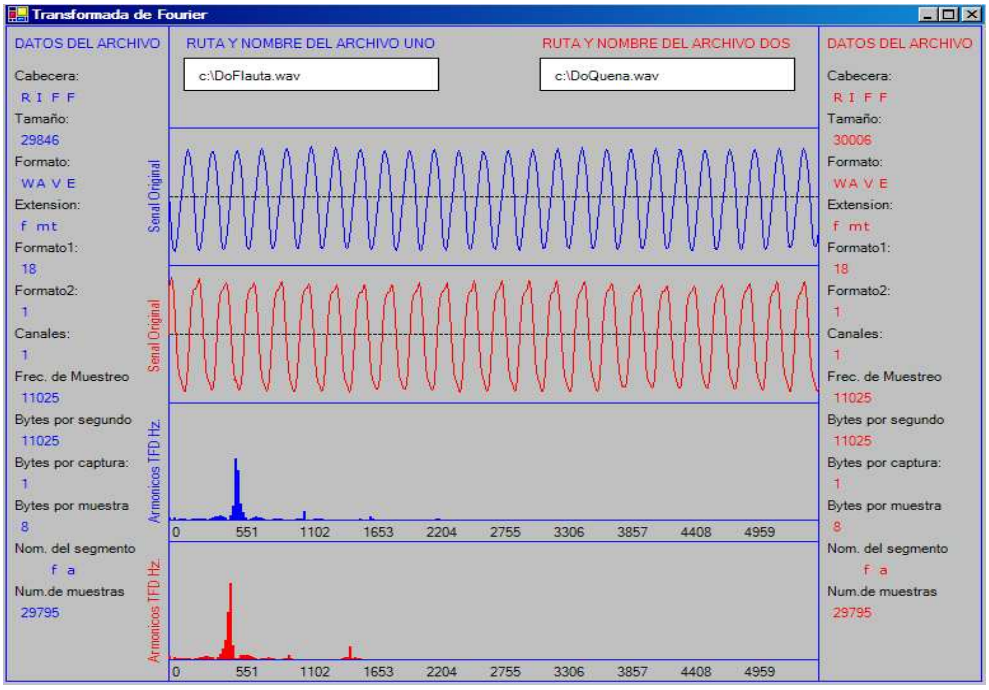

Figura 4.7: Comparación espectral de C5, entre el instrumento elaborado y la quena sudamericana de la Figura 4.6

Figura 4.8: Flauta dulce

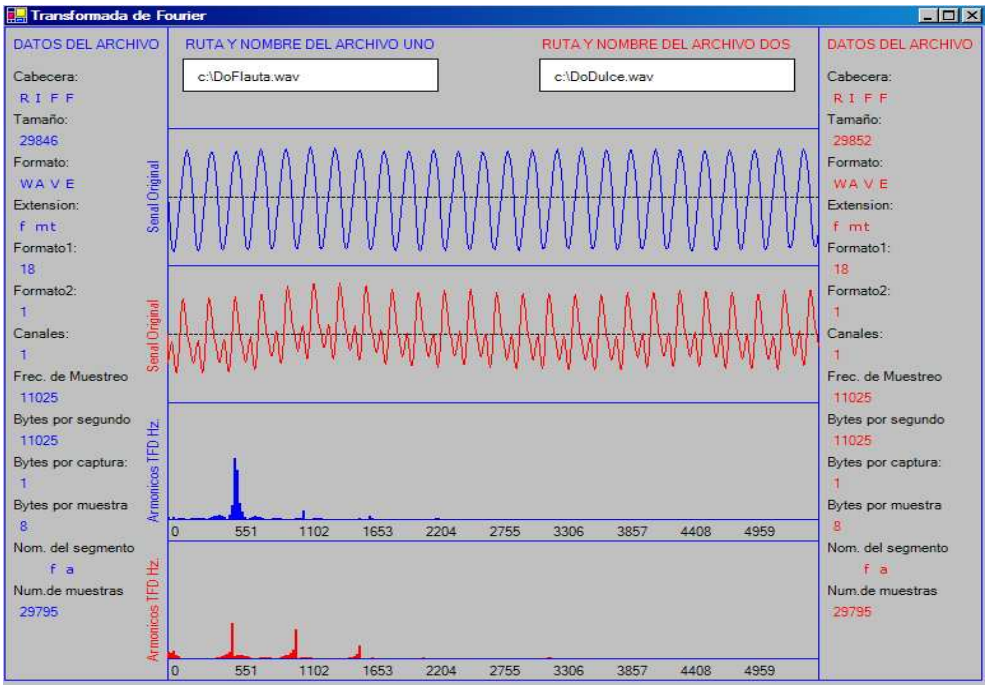

Figura 4.9: Comparación espectral de C5, entre el instrumento elaborado y la flauta dulce de la Figura 4.8

Figura 4.10: Tarka sudamericana

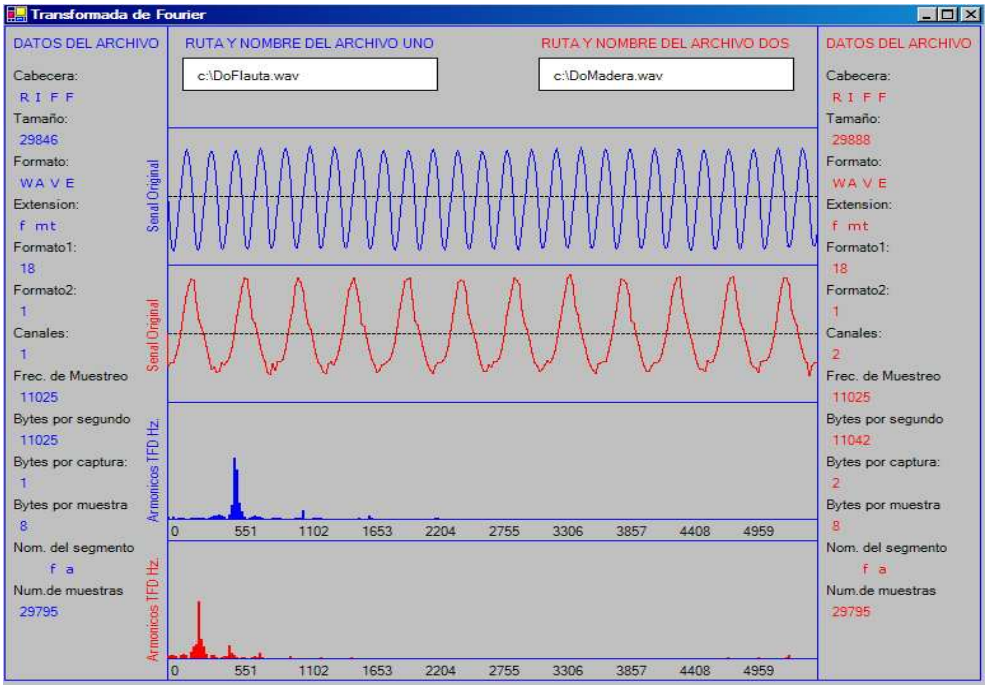

Figura 4.11: Comparación espectral de C5, entre el instrumento elaborado y la tarka sudamericana de la Figura 4.10

Capítulo 5

Conclusiones y Trabajos Futuros

El trabajo desarrollado en la presente tesis, presentó el método para diseñar y construir una flauta similar a una quena sudamericana, este método puede ser extrapolado a cualquier instrumento musical de viento realizando los ajustes para la forma, tamaño y características musicales del instrumento que se quiera analizar o construir.

Se eligió el PVC como material de construcción del instrumento, por permitir una fácil manipulación de corte y perforación. Se emplean medidas estándares industriales con el objeto de tener dimensiones reproducibles, a diferencia de un proceso artesanal donde la construcción varía para cada instrumento.

El proceso de diseño del instrumento se realizó en forma de algoritmo, al ingresar las características del material elegido se generan las posiciones de los orificios a lo largo del instrumento, esto permite elegir las dimensiones más apropiadas. Se desarrolló una herramienta computacional para conocer el espectro en frecuencia de las muestras de un archivo WAV utilizando la Transformada de Fourier Discreta,

El modelo acústico utilizado en el diseño del instrumento proporciona resultados satisfactorios, al comparar las notas generadas por el instrumento con las notas generadas digitalmente cuyos valores de frecuencia corresponden a los calculos teóricos, los errores obtenidos son menores al 10 %.

Se propone como un trabajo posterior realizar el cálculo del espectro en frecuencia usando el algoritmo de la Transformada Rápida de Fourier, y agregar al software

una herramienta para elegir el número de muestras en las que se realiza el proceso, obteniendo diferentes resoluciones espectrales.

Como herramientas complementarias al software se puede mostrar el tiempo de reproducción de las muestras, proporcionar un zoom para intervalos de la señal y generar la reproducción de las muestras seleccionadas.

Mediante el software desarrollado se propone analizar el comportamiento de diferentes tipos de ventanas empleadas en el procesamiento digital, cambiando la parte de la expresión matemática del algoritmo de la ventana de Hamming por la expresión de otra ventana.

El software desarrollado se puede utilizar para implementar procesos que determinen otras características en una señal de audio contenida en un archivo WAV, como pueden ser, el cálculo de la energía de la señal, la obtención de su espectro en frecuencia por predicción lineal, la convolución de señales, la identificación de formantes en procesos de reconocimiento de voz y la edición de audio.

Apéndice A

Código fuente del software desarrollado

Este apéndice se divide en dos secciones. La primera muestra el código fuente del programa que calcula la Transformada de Fourier Discreta de las muestras de dos archivos WAV previamente capturados, emplea una ventana de Hamming como filtro digital sobre las muestras. El programa esta diseñado para trabajar con archivos WAV en formato monofonico, con muestras de 8 bytes, capturados a cualquier frecuencia de muestreo. En el formulario desarrollado se presentan en forma gráfica las primeras 512 muestras de los archivos, sus espectros en frecuencia y los datos contenidos en sus cabeceras.

La segunda sección muestra el código fuente del programa que calcula las características físicas de diseño del instrumento musical desarrollado en este proyecto de tesis, su construcción se realizo en un tubo de PVC y la escala de afinación elegida es G. En el programa se ingresan, la frecuencia de trabajo limite, el valor del radio del tubo, el espesor del tubo y el radio de los orificios que generan las notas musicales. El programa calcula a partir de los datos especificados, los 12 intervalos que forman una octava, los valores de las frecuencias para cada nota, y la posición de los orificios a lo largo del tubo, estos valores determinan las características de los resonadores Helmholtz que generan las frecuencias de resonancia de las notas musicales del instrumento.

A.1. Programa: Transformada de Fourier Discreta de dos archivos WAV

```
000 using System;
001 using System.IO;
002 using System.Drawing;
003 using System.Windows.Forms;
004 using System.Drawing.Drawing2D;
005 public class Transformada:Form{
006   public static void Main(){
007     Application.Run(new Transformada());
008   }
009   public Transformada(){
010     this.Text = "Transformada de Fourier";
011     this.Size = new Size(778,640);
012     this.Paint += new PaintEventHandler(Dibujar_Graficas);
013   }
014   public void Dibujar_Graficas(object sender,PaintEventArgs e){
015     string nombre=Archivo1;
016     string nombre2=Archivo2;
017     Graphics g = e.Graphics;
018     //AREAS DE TRABAJO
019     g.DrawRectangle(new Pen(Color.Blue),128,0,511,95);
020     g.DrawRectangle(new Pen(Color.Blue),128,95,511,515);
021     g.DrawRectangle(new Pen(Color.Blue),0,0,128,610);
022     //LIMITADORES DE AREAS DE SEÑALES
023     g.DrawLine(new Pen(Color.Blue,0),128,223,639,223);
024     g.DrawLine(new Pen(Color.Blue,0),128,351,639,351);
025     g.DrawLine(new Pen(Color.Blue,0),128,480,639,480);
026     //REFERENCIA CERO DE LAS SENALES
027     Pen penline = new Pen(Color.Black,1);
028     penline.DashStyle = DashStyle.Dash;
029     g.DrawLine(penline,129,159,639,159);
030     g.DrawLine(penline,129,287,639,287);
031     //MENSAJES DE CONTENIDO DE LAS SENALES
032     g.RotateTransform(-90);
033     g.DrawString("Senal Original",Font,Brushes.Blue,
                    new Rectangle(-195,110,120,40));
034     g.DrawString("Senal Original",Font,Brushes.Red,
                    new Rectangle(-325,110,120,40));
035     g.DrawString("Armonicos TFD Hz.",Font,Brushes.Blue,
                    new Rectangle(-465,110,120,40));
036     g.DrawString("Armonicos TFD Hz.",Font,Brushes.Red,
                    new Rectangle(-595,110,120,40));
037     g.RotateTransform(90);
038     //CUADRO DE NOMBRE DE ARCHIVO
039     g.DrawString("RUTA Y NOMBRE DEL ARCHIVO UNO",Font,
```

```
                Brushes.Blue,140,10);
040     g.FillRectangle(new SolidBrush (Color.White),140,30,200,30);
041     g.DrawRectangle(new Pen(Color.Black),140,30,200,30);
042     g.DrawString(nombre,Font,Brushes.Black,150,40);
043     int i;
044     float [] MuestrasWAV = new float [1068];
045     FileStream archivo;
046     try{
047       archivo = new FileStream(nombre,FileMode.Open);
048     }
049     catch(FileNotFoundException exc){
050       g.DrawString("Error en la ruta especificada",Font,
                    Brushes.Blue,140,70);
051       return;
052     }
053     catch{
054       g.DrawString("Error al abrir el archivo",Font,
                    Brushes.Blue,152,70);
055       return;
056     }
057     for(i=0; i<1068; i++){
058       MuestrasWAV[i]=archivo.ReadByte()-128;
059     }
060     //IDENTIFICADORES DEL ARCHIVO WAV
061     g.DrawString("DATOS DEL ARCHIVO",Font,Brushes.Blue,5,10);
062     //CABECERA
063     g.DrawString("Cabecera:",Font,Brushes.Black,5,40);
064     for(i=0;i<4;i++){
065       byte Cabecera;
066       Cabecera=(byte)(MuestrasWAV[i]+128);
067       g.DrawString(""+(char)Cabecera,new Font("Verdana",8),
                    Brushes.Blue,10+(i*12),60);
068     }
069     //TAMAÑO
070     g.DrawString("Tamaño:",Font,Brushes.Black,5,80);
071     int Sumat=256,Condiciont=(int)(MuestrasWAV[5]+128),Bit2t=0,
            Tamaño=0,Correcciont=0;
072     float Restot=0;
073     for(i=0;i<8;i++){
074       { Sumat=Sumat*2;
075         Restot=MuestrasWAV[5]%2;
076         Condiciont=Condiciont/2;
077         if((Condiciont%2)==1){Bit2t=Bit2t+Sumat;}
078         }
079     }
080     if(Restot==1){Correcciont=256;}
081     Tamaño=Bit2t+(int)(MuestrasWAV[4]+128)+Correcciont;
```

```
082     g.DrawString(""+Tamaño,Font,Brushes.Blue,10,100);
083     //FORMATO
084     g.DrawString("Formato:",Font,Brushes.Black,5,120);
085     for(i=8;i<12;i++){
086       byte Formato;
087       Formato=(byte)(MuestrasWAV[i]+128);
088       g.DrawString(""+(char)Formato,new Font("Verdana",8),
                    Brushes.Blue,10-96+(i*12),140);
089     }
090     //EXTENSION
091     g.DrawString("Extension:",Font,Brushes.Black,5,160);
092     for(i=12;i<16;i++){
093       byte Extension;
094       Extension=(byte)(MuestrasWAV[i]+128);
095       g.DrawString(""+(char)Extension,new Font("Verdana",8),
                    Brushes.Blue,10-144+(i*12),180);
096     }
097     //FORMATO1
098     g.DrawString("Formato1:",Font,Brushes.Black,5,200);
099     for(i=16;i<20;i++){
100       byte Formato1;
101       Formato1=(byte)(MuestrasWAV[i]+128);
102       if(Formato1!=0){g.DrawString(""+Formato1,Font,
                                    Brushes.Blue,10-368+(i*23),220);}
103     }
104     //FORMATO2
105     g.DrawString("Formato2:",Font,Brushes.Black,5,240);
106     for(i=20;i<22;i++){
107       byte Formato2;
108       Formato2=(byte)(MuestrasWAV[i]+128);
109       if(Formato2!=0){g.DrawString(""+Formato2,Font,
                                    Brushes.Blue,10-240+(i*12),260);}
110     }
111     //CANALES
112     g.DrawString("Canales:",Font,Brushes.Black,5,280);
113     for(i=22;i<24;i++){
114       byte Canales;
115       Canales=(byte)(MuestrasWAV[i]+128);
116       if(Canales!=0){g.DrawString(""+Canales,Font,
                                   Brushes.Blue,10-264+(i*12),300);}
117     }
118     //FRECUENCIA DE MUESTREO
119     g.DrawString("Frec. de Muestreo",Font,Brushes.Black,5,320);
120     int Sumafm=256,Condicionfm=(int)(MuestrasWAV[25]+128),Bit2fm=0,
            Frecmues=0,Correccionfm=0;
121     float Restofm=0;
122     for(i=0;i<8;i++){
```

```
123       { Sumafm=Sumafm*2;
124         Restofm=MuestrasWAV[25]%2;
125         Condicionfm=Condicionfm/2;
126         if((Condicionfm%2)==1){Bit2fm=Bit2fm+Sumafm;}
127       }
128     }
129     if(Restofm==-1){Correccionfm=256;}
130     Frecmues=Bit2fm+(int)(MuestrasWAV[24]+128)+Correccionfm;
131     g.DrawString(""+Frecmues,Font,Brushes.Blue,10,340);
132     //BYTES POR SEGUNDO
133     g.DrawString("Bytes por segundo",Font,Brushes.Black,5,360);
134     int Sumabxs=256,Condicionbxs=(int)(MuestrasWAV[29]+128),Bit2bxs=0,
            Bytesxseg=0,Correccionbxs=0;
135     float Restobxs=0;
136     for(i=0;i<8;i++){
137       { Sumabxs=Sumabxs*2;
138         Restobxs=MuestrasWAV[29]%2;
139         Condicionbxs=Condicionbxs/2;
140         if((Condicionbxs%2)==1){Bit2bxs=Bit2bxs+Sumabxs;}
141       }
142     }
143     if(Restobxs==-1){Correccionbxs=256;}
144     Bytesxseg=Bit2bxs+(int)(MuestrasWAV[28]+128)+Correccionbxs;
145     g.DrawString(""+Bytesxseg,Font,Brushes.Blue,10,380);
146     //BYTES POR CAPTURA
147     g.DrawString("Bytes por captura:",Font,Brushes.Black,5,400);
148     for(i=32;i<34;i++){
149       byte Bytesxcap;
150       Bytesxcap=(byte)(MuestrasWAV[i]+128);
151       if(Bytesxcap!=0){g.DrawString(""+Bytesxcap,Font,
                                      Brushes.Blue,10-800+(i*25),420);}
152     }
153     //BYTES POR MUESTRA
154     g.DrawString("Bytes por muestra",Font,Brushes.Black,5,440);
155     for(i=34;i<36;i++){
156       byte Bytesxmues;
157       Bytesxmues=(byte)(MuestrasWAV[i]+128);
158       if(Bytesxmues!=0){g.DrawString(""+Bytesxmues,Font,
                                       Brushes.Blue,10-850+(i*25),460);}
159     }
160     //NOMBRE DEL SEGMENTO
161     g.DrawString("Nom. del segmento",Font,Brushes.Black,5,480);
162     for(i=36;i<40;i++){
163       byte Nomseg;
164       Nomseg=(byte)(MuestrasWAV[i]+128);
165       if(Nomseg>65&&Nomseg<122){g.DrawString(""+(char)Nomseg,
             new Font("Verdana",8),Brushes.Blue,10-432+(i*12),500);}
```

```
166     }
167     //NUMERO DE MUESTRAS
168     g.DrawString("Num.de muestras",Font,Brushes.Black,5,520);
169     int Sumanm=256,Condicionnm=(int)(MuestrasWAV[41]+128),Bit2nm=0,
            Nummues=0,Correccionnm=0;
170     float Restonm=0;
171     for(i=0;i<8;i++){
172       { Sumanm=Sumanm*2;
173         Restonm=MuestrasWAV[41]%2;
174         Condicionnm=Condicionnm/2;
175         if((Condicionnm%2)==1){Bit2nm=Bit2nm+Sumanm;}
176       }
177     }
178     if(Restonm==-1){Correccionnm=256;}
179     Nummues=Bit2nm+(int)(MuestrasWAV[40]+128)+Correccionnm;
180     g.DrawString(""+Nummues,Font,Brushes.Blue,10,540);
181     //GRAFICA DE LAS MUESTRAS
182     float x0,y0,x,y;
183     x=128; y=159;
184     for(i=58; i<570; i++){
185       x0=x;
186       x=128+(i-58);
187       y0=y;
188       y=159-((MuestrasWAV[i])/2);
189       g.DrawLine(new Pen(Color.Blue,0),x0,y0,x,y);
190     }
191     //CALCULO DE LA VENTANA
192     float[] Ventana = new float[1068];
193     for(i=58; i<570; i++){
194       Ventana[i]=(float)(.54-(.46*Math.Cos((2*3.14*(i-58))/512)));
195     }
196     //MUESTRAS ENVENTANADAS
197     float[] Muestras = new float[1068];
198     for(i=58; i<570; i++){
199       Muestras[i]=Ventana[i]*MuestrasWAV[i];
200     }
201     //TRANSFORMADA DE FOURIER
202     float Real,Imag;
203     float [] Modulo = new float[1068];
204     int N=512,n,k;
205     for(n=58; n<314; n++){
206       Real=0;
207       Imag=0;
208       for(k=58; k<570; k++){
209         Real=Real+(MuestrasWAV[k]*
                 (float)Math.Cos((-2*Math.PI*(n-58)*(k-58))/N));
210         Imag=Imag+(MuestrasWAV[k]*
```

```
                (float)Math.Sin((-2*Math.PI*(n-58)*(k-58))/N));
211       }
212       Modulo[n]=(float)(Math.Sqrt((Real*Real)+(Imag*Imag)));
213       g.DrawLine(new Pen(Color.Blue,0),128,460,639,460);
214       g.FillRectangle(new SolidBrush(Color.Blue),128+((n-58)*2),
                        460-(Modulo[n]/285),2,Modulo[n]/285);
215     }
216     float eje;
217     for(i=0;i<10;i++){
218       eje=(int)(i*((Frecmues/2)/10));
219       g.DrawString(""+eje,Font,Brushes.Black,129+(i*50),465);
220     }
221     archivo.Close();
222     //INFORMACION ARCHIVO DOS
223     g.DrawRectangle(new Pen(Color.Blue),639,0,128,610);
224     g.DrawString("RUTA Y NOMBRE DEL ARCHIVO DOS",Font,
                  Brushes.Red,420,10);
225     //CUADRO DE NOMBRE DE ARCHIVO DOS
226     g.FillRectangle(new SolidBrush (Color.White),420,30,200,30);
227     g.DrawRectangle(new Pen(Color.Black),420,30,200,30);
228     g.DrawString(nombre2,Font,Brushes.Black,430,40);
229     try{
230       archivo = new FileStream(nombre2,FileMode.Open);
231     }
232     catch(FileNotFoundException exc){
233       g.DrawString("No se encuentra el archivo en la ruta
                      especificada",Font,Brushes.Green,140,70);
234       return;
235     }
236     catch{
237       g.DrawString("No se puede abrir el archivo",Font,
                      Brushes.Green,152,70);
238       return;
239     }
240     //MUESTRAS ARCHIVO DOS
241     for(i=0; i<1068; i++){
242       MuestrasWAV[i]=archivo.ReadByte()-128;
243     }
244     //IDENTIFICADORES ARCHIVO DOS
245     g.DrawString("DATOS DEL ARCHIVO",Font,Brushes.Red,644,10);
246     //CABECERA DOS
247     g.DrawString("Cabecera:",Font,Brushes.Black,644,40);
248     for(i=0;i<4;i++){
249       byte Cabecera;
250       Cabecera=(byte)(MuestrasWAV[i]+128);
251       g.DrawString(""+(char)Cabecera,new Font("Verdana",8),
                      Brushes.Red,649+(i*12),60);
```

```
252     }
253     //TAMAÑO ARCHIVO DOS
254     g.DrawString("Tamaño:",Font,Brushes.Black,644,80);
255     for(i=0;i<8;i++){
256       { Sumat=Sumat*2;
257         Restot=MuestrasWAV[5]%2;
258         Condiciont=Condiciont/2;
259         if((Condiciont%2)==1){Bit2t=Bit2t+Sumat;}
260       }
261     }
262     if(Restot==1){Correcciont=256;}
263     Tamaño=Bit2t+(int)(MuestrasWAV[4]+128)+Correcciont;
264     g.DrawString(""+Tamaño,Font,Brushes.Red,649,100);
265     //FORMATO ARCHIVO DOS
266     g.DrawString("Formato:",Font,Brushes.Black,644,120);
267     for(i=8;i<12;i++){
268       byte Formato;
269       Formato=(byte)(MuestrasWAV[i]+128);
270       g.DrawString(""+(char)Formato,new Font("Verdana",8),
                    Brushes.Red,649-96+(i*12),140);
271     }
272     //EXTENSION ARCHIVO DOS
273     g.DrawString("Extension:",Font,Brushes.Black,644,160);
274     for(i=12;i<16;i++){
275       byte Extension;
276       Extension=(byte)(MuestrasWAV[i]+128);
277       g.DrawString(""+(char)Extension,new Font("Verdana",8),
                    Brushes.Red,649-144+(i*12),180);
278     }
279     //FORMATO1 ARCHIVO DOS
280     g.DrawString("Formato1:",Font,Brushes.Black,644,200);
281     for(i=16;i<20;i++){
282       byte Formato1;
283       Formato1=(byte)(MuestrasWAV[i]+128);
284       if(Formato1!=0){g.DrawString(""+Formato1,Font,
                                    Brushes.Red,649-368+(i*23),220);}
285     }
286     //FORMATO2 ARCHIVO DOS
287     g.DrawString("Formato2:",Font,Brushes.Black,644,240);
288     for(i=20;i<22;i++){
289       byte Formato2;
290       Formato2=(byte)(MuestrasWAV[i]+128);
291       if(Formato2!=0){g.DrawString(""+Formato2,Font,
                                    Brushes.Red,649-240+(i*12),260);}
292     }
293     //CANALES ARCHIVO DOS
294     g.DrawString("Canales:",Font,Brushes.Black,644,280);
```

```
295     for(i=22;i<24;i++){
296       byte Canales;
297       Canales=(byte)(MuestrasWAV[i]+128);
298       if(Canales!=0){g.DrawString(""+Canales,Font,
                                  Brushes.Red,649-264+(i*12),300);}
299     }
300     //FRECUENCIA DE MUESTREO ARCHIVO DOS
301     g.DrawString("Frec. de Muestreo",Font,Brushes.Black,644,320);
302     for(i=0;i<8;i++){
303       { Sumafm=Sumafm*2;
304         Restofm=MuestrasWAV[25]%2;
305         Condicionfm=Condicionfm/2;
306         if((Condicionfm%2)==1){Bit2fm=Bit2fm+Sumafm;}
307       }
308     }
309     if(Restofm==-1){Correccionfm=256;}
310     Frecmues=Bit2fm+(int)(MuestrasWAV[24]+128)+Correccionfm;
311     g.DrawString(""+Frecmues,Font,Brushes.Red,649,340);
312     //BYTES POR SEGUNDO ARCHIVO DOS
313     g.DrawString("Bytes por segundo",Font,Brushes.Black,644,360);
314     for(i=0;i<8;i++){
315       { Sumabxs=Sumabxs*2;
316         Restobxs=MuestrasWAV[29]%2;
317         Condicionbxs=Condicionbxs/2;
318         if((Condicionbxs%2)==1){Bit2bxs=Bit2bxs+Sumabxs;}
319       }
320     }
321     if(Restobxs==-1){Correccionbxs=256;}
322     Bytesxseg=Bit2bxs+(int)(MuestrasWAV[28]+128)+Correccionbxs;
323     g.DrawString(""+Bytesxseg,Font,Brushes.Red,649,380);
324     //BYTES POR CAPTURA ARCHIVO DOS
325     g.DrawString("Bytes por captura:",Font,Brushes.Black,644,400);
326     for(i=32;i<34;i++){
327     byte Bytesxcap;
328     Bytesxcap=(byte)(MuestrasWAV[i]+128);
329     if(Bytesxcap!=0){g.DrawString(""+Bytesxcap,Font,
                                  Brushes.Red,649-800+(i*25),420);}
330     }
331     //BYTES POR MUESTRA ARCHIVO DOS
332     g.DrawString("Bytes por muestra",Font,Brushes.Black,644,440);
333     for(i=34;i<36;i++){
334       byte Bytesxmues;
335       Bytesxmues=(byte)(MuestrasWAV[i]+128);
336       if(Bytesxmues!=0){g.DrawString(""+Bytesxmues,Font,
                                    Brushes.Red,649-850+(i*25),460);}
337     }
338     //NOMBRE DEL SEGMENTO ARCHIVO DOS
```

```
339     g.DrawString("Nom. del segmento",Font,Brushes.Black,644,480);
340     for(i=36;i<40;i++){
341       byte Nomseg;
342       Nomseg=(byte)(MuestrasWAV[i]+128);
343       if(Nomseg>65&&Nomseg<122){g.DrawString(""+(char)Nomseg,
              new Font("Verdana",8),Brushes.Red,649-432+(i*12),500);}
344     }
345     //NUMERO DE MUESTRAS ARCHIVO DOS
346     g.DrawString("Num.de muestras",Font,Brushes.Black,644,520);
347     for(i=0;i<8;i++){
348       { Sumanm=Sumanm*2;
349         Restonm=MuestrasWAV[41]%2;
350         Condicionnm=Condicionnm/2;
351         if((Condicionnm%2)==1){Bit2nm=Bit2nm+Sumanm;}
352       }
353     }
354     if(Restonm==-1){Correccionnm=256;}
355     Nummues=Bit2nm+(int)(MuestrasWAV[40]+128)+Correccionnm;
356     g.DrawString(""+Nummues,Font,Brushes.Red,649,540);
357     //GRAFICA DE LAS MUESTRAS ARCHIVO DOS
358     x=128; y=287;
359     for(i=58; i<570; i++){
360       x0=x;
361       x=128+(i-58);
362       y0=y;
363       y=287-((MuestrasWAV[i])/2);
364       g.DrawLine(new Pen(Color.Red,0),x0,y0,x,y);
365     }
366     //MUESTRAS ENVENTANEADAS ARCHIVO DOS
367     x=128; y=287;
368     for(i=58; i<570; i++){
369       Muestras[i]=Ventana[i]*MuestrasWAV[i];
370     }
371     //TRANSFORMADA DE FOURIER ARCHIVO DOS
372     for (n=58; n<314; n++){
373       Real=0;
374       Imag=0;
375       for (k=58; k<570; k++){
376         Real=Real+(MuestrasWAV[k]*
                   (float)Math.Cos((-2*Math.PI*(n-58)*(k-58))/N));
377         Imag=Imag+(MuestrasWAV[k]*
                   (float)Math.Sin((-2*Math.PI*(n-58)*(k-58))/N));
378       }
379       Modulo[n]=(float)(Math.Sqrt((Real*Real)+(Imag*Imag)));
380       g.DrawLine(new Pen(Color.Blue,0),128,590,639,590);
381       g.FillRectangle(new SolidBrush(Color.Red),128+((n-58)*2),
                        590-(Modulo[n]/285),2,Modulo[n]/285);
```

```
382     }
383     for(i=0;i<10;i++){
384       eje=(int)(i*((Frecmues/2)/10));
385       g.DrawString(""+eje,Font,Brushes.Black,129+(i*50),595);
386     }
387     archivo.Close();
388   }
389 }
```

A.2. Programa: Características físicas de una flauta en G

```
000  using System;
001  using System.IO;
002  using System.Drawing;
003  using System.Windows.Forms;
004  using System.Drawing.Drawing2D;
005  public class Flauta:Form{
006    public static void Main(){
007      Application.Run(new Flauta());
008    }
009    public Flauta(){
010      this.Text = "Flauta en G Mayor";
011      this.Size = new Size(550,490);
012      this.Paint += new PaintEventHandler(Draw_Graphics);
013    }
014    public void Draw_Graphics(object sender,PaintEventArgs e){
015      Graphics g = e.Graphics;
016      //CARACTERISTICAS FISICAS
017      double RadOri,c,L,G4,RT;
018      c = 343.3453734;      //Velocidad del sonido a 20 grados
019      RT=0.0127;            //Radio del tubo
020      RadOri = 0.00555625;  //Radio del orificio
021      L = 0.0015875;        //Espesor del tubo
022      G4=391.995436;        //Frecuencia limite
023      g.DrawString("Radio del Tubo:",Font,Brushes.Black,5,420);
024      g.DrawString(""+RT+" m",Font,Brushes.Blue,90,420);
025      g.DrawString("Radio del Orificio:",Font,Brushes.Black,5,440);
026      g.DrawString(""+RadOri+" m",Font,Brushes.Blue,100,440);
027      g.DrawString("Espesor del tubo:",Font,Brushes.Black,200,420);
028      g.DrawString(""+L+" m",Font,Brushes.Blue,295,420);
029      g.DrawString("Frecuencia Limite:",Font,Brushes.Black,200,440);
030      g.DrawString(""+G4+" m",Font,Brushes.Blue,300,440);
031      //CALCULO DE RAZON, FRECUENCIA Y POSICION
032      g.DrawString("RAZON",Font,Brushes.Black,210,10);
```

```
033     g.DrawString("FRECUENCIA",Font,Brushes.Black,280,10);
034     g.DrawString("( Hz )",Font,Brushes.Black,300,24);
035     g.DrawString("POSICION",Font,Brushes.Black,440,10);
036     g.DrawString("( m )",Font,Brushes.Black,455,24);
037     float i;
038     double In,frec,pi=3.141592654,S,c2,Lef,w2,V,h;
039     for(i=0; i<13; i++){
040       In = Math.Pow(2.0,i/12.0);
041       frec=(float)(G4*In);
042       S = pi*RadOri*RadOri;
043       c2 = c*c;
044       Lef = L+(1.45*RadOri);
045       w2 = (2*pi*frec)*(2*pi*frec);
046       V = (S*c2)/(Lef*w2);
047       h = V/(pi*RT*RT);
048       g.DrawString(""+(float)In,Font,Brushes.Blue,210,40+(i*20));
049       g.DrawString(""+(float)frec,Font,Brushes.Blue,290,40+(i*20));
050       g.DrawString(""+(float)h,Font,Brushes.Blue,440,40+(i*20));
051     }
052     //INDICADORES DE INTERVALOS
053     g.DrawString("INTERVALOS DE UNA OCTAVA",Font,Brushes.Blue,5,10);
054     g.DrawString("Unitono",Font,Brushes.Black,5,40);
055     g.DrawString("Semitono o Segunda Menor",Font,Brushes.Black,5,60);
056     g.DrawString("Tono entero o Segunda Mayor",Font,Brushes.Black,5,80);
057     g.DrawString("Tercera Menor",Font,Brushes.Black,5,100);
058     g.DrawString("Tercera Mayor",Font,Brushes.Black,5,120);
059     g.DrawString("Cuarta Perfecta",Font,Brushes.Black,5,140);
060     g.DrawString("Cuarta Aumentada y Quinta Disminuida",Font,
061                  Brushes.Black,5,160);
062     g.DrawString("Quinta Perfecta",Font,Brushes.Black,5,180);
063     g.DrawString("Sexta Menor",Font,Brushes.Black,5,200);
064     g.DrawString("Sexta Mayor",Font,Brushes.Black,5,220);
065     g.DrawString("Séptima Menor",Font,Brushes.Black,5,240);
066     g.DrawString("Séptima Mayor",Font,Brushes.Black,5,260);
067     g.DrawString("Octava",Font,Brushes.Black,5,280);
068     //NOMBRE DE LAS NOTAS
069     g.DrawString("ORIFICIO",Font,Brushes.Black,370,10);
070     g.DrawString("G4",Font,Brushes.Blue,370,40);
071     g.DrawString("Ausente",Font,Brushes.Blue,370,60);
072     g.DrawString("A4",Font,Brushes.Blue,370,80);
073     g.DrawString("Ausente",Font,Brushes.Blue,370,100);
074     g.DrawString("B4",Font,Brushes.Blue,370,120);
075     g.DrawString("C5",Font,Brushes.Blue,370,140);
076     g.DrawString("Ausente",Font,Brushes.Blue,370,160);
077     g.DrawString("D5",Font,Brushes.Blue,370,180);
078     g.DrawString("Ausente",Font,Brushes.Blue,370,200);
079     g.DrawString("E5",Font,Brushes.Blue,370,220);
```

```
080      g.DrawString("Ausente ",Font,Brushes.Blue,370,240);
081      g.DrawString("F#5",Font,Brushes.Blue,370,260);
082      g.DrawString("G5",Font,Brushes.Blue,370,280);
083      //LONGITUD TOTAL DEL INSTRUMENTO (UNITONO)
084      float LongT,LefecT;
085      LefecT=(float)(c/(2.0*G4));
086      LongT=(float)(LefecT-(0.6*RT));
087      g.DrawString("Longitud Total:",Font,Brushes.Black,400,420);
088      g.DrawString(""+LongT,Font,Brushes.Blue,480,420);
089      //TABLA DE DATO
090      g.DrawRectangle(new Pen(Color.Blue),0,0,540,300);
091      //DIBUJO DEL INSTRUMENTO
092      g.DrawString("CARACTERISTICAS FISICAS",Font,Brushes.Blue,5,310);
093      g.DrawRectangle(new Pen(Color.Black),60,370,450,30);
094      g.DrawString("G4",Font,Brushes.Blue,515,380);
095      g.DrawRectangle(new Pen(Color.Black),60,380,20,10);
096      g.DrawString("Boquilla",Font,Brushes.Blue,10,380);
097      g.DrawEllipse(new Pen(Color.Black),215,375,20,20);
098      g.DrawString("G5",Font,Brushes.Blue,215,350);
099      g.DrawEllipse(new Pen(Color.Black),255,375,20,20);
100      g.DrawString("F#5",Font,Brushes.Blue,255,350);
101      g.DrawEllipse(new Pen(Color.Black),295,375,20,20);
102      g.DrawString("E5",Font,Brushes.Blue,295,350);
103      g.DrawEllipse(new Pen(Color.Black),335,375,20,20);
104      g.DrawString("D5",Font,Brushes.Blue,335,350);
105      g.DrawEllipse(new Pen(Color.Black),375,375,20,20);
106      g.DrawString("C5",Font,Brushes.Blue,375,350);
107      g.DrawEllipse(new Pen(Color.Black),415,375,20,20);
108      g.DrawString("B4",Font,Brushes.Blue,415,350);
109      g.DrawEllipse(new Pen(Color.Black),455,375,20,20);
110      g.DrawString("A4",Font,Brushes.Blue,455,350);
111    }
112  }
```

Bibliografía

[1] Bernal Bermudez Jesús, Bobadilla Sancho Jesús, Gómez Vilda Pedro, *Reconocimiento de Voz y Fonética Acústica*, RA-MA, España, 2000. (p: 21)

[2] Chapra Steven C., Canale Raymond P., *Numerical Methods for Engineers: With Software and Programming Applications*, Fourth Edition, Mc Graw Hill, USA, 2002. (p: 23)

[3] Eco Humberto. *Como si fa una Tesi di Laurea*, Tascabili Bompiani, Italia, 1977. (p: 6)

[4] Hayes Monson H., *Digital Signal Processing*, Mc Graw Hill, USA, 1999. (p: 22)

[5] Helmholtz Hermann, *On the Sensations of Tone*, Fourth Edition, Dover, USA, 1954. (p: 10)

[6] Kalouptsidis Nicholas. *Signal Processing Systems: Theory and Design*, Wiley Interscience, USA, 1997. (p: 21)

[7] Kinsler Laurent E., Frey Austin R., Coppens Alan B., Sanders James V., *Fundamentals of Acoustics*, Third Edition, John Wiley & Sons Inc., USA, 1990. (p: 14)

[8] Meade M. L., Dillon C. R., *Signals and Systems: Models and Behaviour*, Second Edition, Chapman & Hall, Inglaterra, 1991. (p: 19)

[9] Olson Harry F., *Music, Physics and Engineering*, Second Edition, Dover, USA, 1967. (p: 10)

[10] Oppenheim Alan V., Schafer Ronald W., *Digital Signal Processing*, Prentice Hall, USA, 1975. (p: 20)

[11] Oppenheim Alan V., Schafer Ronald W., Buck John R., *Discrete Time Signal Processing*, Prentice Hall, USA, 1999. (p: 21)

[12] Petzold Charles, *Programming Microsoft Windows with C#*, Mc Graw Hill, USA, 2002. (p: 39)

[13] Proakis John G., Ingle Vinay K., *A Self-Study Guide for Digital Signal Processing*, Prentice Hall, USA, 2004. (p: 23)

[14] Proakis John G., Manolakis Dimitris G., *Digital Singal Processing: Principles, Algorithms and Applications*, Third Edition, Prentice Hall, USA, 1996. (p: 19)

[15] Rayleigh John William Strutt Baron, *The Theory of the Sound Vol I, Vol II*, Dover, USA, 1945. (p: 13)

[16] Rojas Ponce Alberto, *Ensamblador Basico*, Computec, México, 1995. (p: 44)

[17] Schildt Herbert, *C# The Complete Reference*, Mc Graw Hill, USA, 2002. (p: 34)

[18] Seto William W., *Acoustics*, Mc Graw Hill, Book Company, USA, 1971. (p: 13)

[19] Valiente Feruglio Gabriel. *Composición de Textos Cientificos con Latex*, Alfaomega, España, 2001. (p: 6)

[20] Yost William A., *Fundamentals of Hearing: An Introduction*, Fourth Edition, Academic Press, USA, 2000. (p: 13)