Reporte Proyecto.- Detección de notas musicales tocadas por una flauta dulce en una grabación.

Alumnos:
Añorve Pons Germán Silvestre 2016630011
Monroy Martos Elioth 2016630258
Profesor: Gutierrez Aldana Eduardo
Materia: Teoría de Comunicaciones y Señales
Grupo: 3CM6

9 de diciembre de 2017

Índice

1. Introducción

La flauta es un instrumento de viento-madera en forma de tubo. Tiene 3 partes: base, cuerpo y cabeza. La flauta tiene ocho agujeros: siete delante y uno detrás. Estos agujeros se numeran, siendo el 0 el agujero de atrás y el 7 el de más abajo. A cada agujero le corresponde un dedo. Se tapan y se destapan con la yema de los dedos, pero sin apretar. Los agujeros 0, 1, 2 y 3 se tapan con la mano izquierda: el pulgar tapa el 0, el índice el 1, medio el 2, y anular el 3. El resto de agujeros se tapan con la mano derecha: el índice el 4, medio el 5, anular el 6 y meñique el 7. El dedo pulgar derecho se coloca entre los agujeros 4 y 5 por detrás de la flauta. Aunque no tapa ningún agujero, sirve para soportar el peso de la flauta. Se puede mostrar en la siguiente imagen.

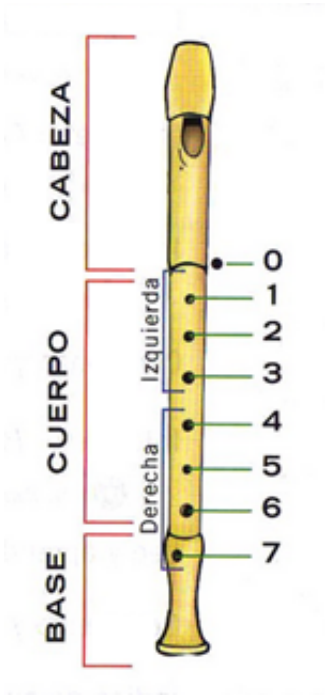


Figura 1: Composición de una flauta dulce

Para poder tocar las notas musicales en la flauta se deben colocar los dedos en los distintos agujeros de la manera en que se muestran en la siguiente

imagen.

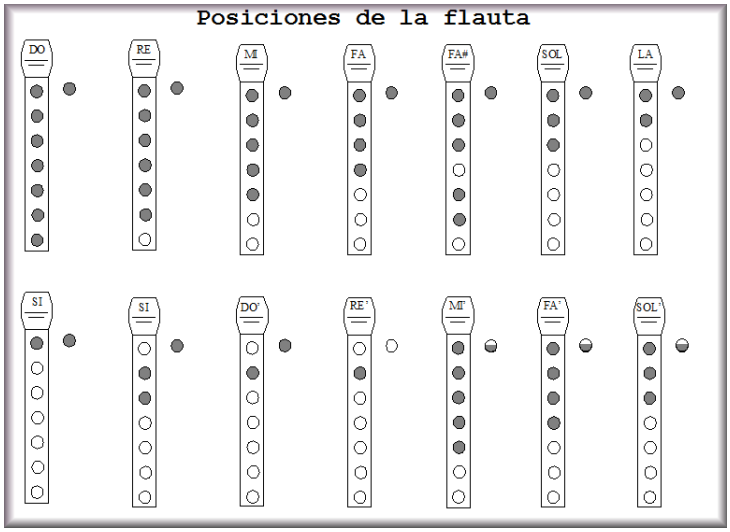


Figura 2: Posiciones de la flauta para tocar diversas notas

Los agujeros grises mostrados en la figura son los agujeros que deben ser tapados por los dedos y los agujeros blancos son los que no se deben tapar para poder hacer sonar esa nota musical. Las notas que tienen un apostrofe son notas más agudas, es decir, están en la segunda octava que puede tocar la flauta. Las notas musicales son 7: Do, Re, Mi, Fa, Sol, La y Si. Entre ellas hay notas que se consiguen ya sea con sostenidos o bemoles. Si quieren notas más graves o agudas que las 7 anteriores o sus intermedias, utilizo el nombre de las mismas notas para los tonos más altos o más graves. Por supuesto que no será la misma nota en la práctica, al escuchar el tono; pero se utilizarán los mismos nombres diciendo que la nota está en otra octava. De este modo tenemos notas de Do a Si, en la siguiente octava nuevamente de Do a Si, luego la octava siguiente de Do a Si, y así sucesivamente todo dentro del rango humanamente audible. En cuanto a las propiedades del sonido, no es en vano que la octava de una nota tenga el mismo nombre. La relación entre una nota y la siguiente del mismo nombre en frecuencia de vibraciones, es decir, la siguiente nota del mismo nombre, en la siguiente octava, vibra exactamente el doble de veces que la anterior, es decir, si un “La” estándar de altura media es producto de 440 vibraciones por segundo, el “La” de la

siguiente Octava es físicamente algo vibrando 880 veces cada segundo, y el siguiente "La" producto de 1760 vibraciones por segundo, esta vibraciones son los Hertz (Hz). En la siguiente tabla se muestran las frecuencias que se pueden alcanzar según las notas musicales, aunque la flauta únicamente alcanza las octavas 4 y 5, es decir, que la frecuencia mas chica que la flauta puede generar es de 261.63 Hz y la nota mas aguda que puede generar es de 987.77 Hz.

Frecuencia (en Hertzios) de las notas musicales

	x								
		0	1	2	3	4	5	6	7	8
n=1	do		32.7	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01
n=2	do#		34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92
n=3	re		36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64
n=4	re#		38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03
n=5	mi		41.2	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04
n=6	fa	21.826	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65
n=7	fa#	23.125	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91
n=8	sol	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93
n=9	sol#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	
n=10	la	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	
n=11	la#	29.14	58.27	116.54	233.08	466.00	932.33	1864.66	3729.31	
n=12	si	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	

Figura 3: Frecuencias de las notas musicales

La transformada de Fourier Una transformada de Fourier es una operación matemática que transforma una señal de dominio de tiempo a dominio de frecuencia y viceversa. Estamos acostumbrados a señales con dominio de tiempo en la vida cotidiana. En el dominio de tiempo, la señal se expresa con respecto al tiempo. En el dominio de frecuencia, una señal es expresada con respecto a la frecuencia.

Una DFT (Transformada de Fourier Discreta - por sus siglas en inglés) es el nombre dado a la transformada de Fourier cuando se aplica a una señal digital (discreta) en vez de una análoga (continua). Una FFT (Transformada Rápida de Fourier) es una versión más rápida de la DFT que puede ser aplicada cuando el número de mustras de la señal es una potencia de dos. Un cálculo de FFT toma aproximadamente N * log2(N) operaciones, mientras que DFT toma aproximadamente N2 operaciones, así es que la FFT es significativamente más rápida.

La transformada discreta de Fourier (DFT) de una señal x[n] definidia en el rango 0<=n<=N-1 se define como:

$x[k] = \sum x[n] * e^{-j\frac{2*\Pi}{N}*k*n}$; 0<=j<=N-1

Funcionamiento del Proyecto Este proyecto consiste en un programa que detecte las notas musicales que son tocadas en una flauta dulce, el sonido que ésta produce será grabado desde la computadora un tiempo N (el tiempo de la canción que se desee tocar). al terminar la grabación, el programa se ejecuta para decir que nota fue tocada cada .064 segundos.
Esto es posible utilizando la FFT y evaluando la amplitud obtenida en frecuencia de los valores arrojados por la transformada.
La decisión de evaluar cada .064 segundos se debe a la frecuencia de muestreo escogida para realizar la grabación (4000 muestras/s, este valor fue obtenido por el teorema del muestreo) y la cantidad de muestras que se deseaban obtener. Al trabajar con la FFT era necesario que este total de muestras fuera una potencia de dos. Al dividir la grabación en fragmentos de .064 segundos se obtiene un total de 256 muestras, lo cual es suficiente para realizar el análisis con la resolución necesaria y empleando el menor número de muestras posible.

2. Código

El proyecto fue implementado en dos módulos.
El primero desarrollado en python el cual consiste en la grabación de audio y el segundo desarrollado en c el cual es el encargado de realizar el analisis en frecuencia del archivo de grabación generado por el primer módulo. A continuación se presenta el código de ambos.
record.py:

```
import pyaudio
import wave
from subprocess import call
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 4000
RECORD_SECONDS = 10
WAVE_OUTPUT_FILENAME = "output.wav"
p = pyaudio.PyAudio()
stream = p.open(format=FORMAT,
channels=CHANNELS,
rate=RATE,
input=True,
frames_per_buffer=CHUNK)
print("* recording")
frames = []
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
  data = stream.read(CHUNK)
  frames.append(data)

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()

call(["flauta.exe", "output.wav"])
```

funciones.h:

```
#ifndef __FUNCIONES_H__
#define __FUNCIONES_H__
  //Librerías de C
  #include <stdio.h>
  #include <stdlib.h>
  #include <math.h>
  #include <string.h>
  //Librería que contiene los máximos y mínimos de los
    diferentes tipos de datos en c
  #include <limits.h>
  //Libreria para conocer tiempo de ejecución
  #include <time.h>
  //Metodos
  void leerCabeceras(char**);
  void escribirArchivo(short*,short*);
  void leerMuestras(short*);
  void leerMuestras2Canales(short*,short*);
  void convertirFloat(short*,float*,float*,int);
  void convertirShort(short*,short*,float*,float*,int);
  //Cabeceras
  int chunkid;
  int chunksize;
  int format;
  int subchunk1id;
  int subchunk1size;
  short audioformat;
  short numchannels;
  int samplerate;
  int byterate;
  short blockalign;
  short bitspersample;
  int subchunk2id;
  int subchunk2size;
  //Archivo
  FILE* entrada;
  FILE* salida;
  //Variables para muestras
  short muestra;
  int aux_conteo=0;
  int total_muestras_originales;
  int total_muestras;
  int hayheaders=0;
  short headers[37];
  //Métodos TDF
```

```
#define PI acos(-1.0)//Defino la constante PI
void calcularFFT(short*,int);
void calcularFFTI(short*,short*);
int calcularNuevoNumeroMuestras(int);
void obtenerNota(short*,int);
//Inversión de bits
#define SWAP(x,y) do {typeof(x) _x = x;typeof(y) _y = y;x = _y
  ;y = _x;} while(0)
//Variables para obtener tiempo de ejecución
clock_t inicio, final;
double total;
//Notas
float duracion;
int aux1;
int amp=3000;
int bandera=0;
//Arreglos de frecuencias
int f_do[3] = {262,523,1046};
int f_doG[3] = {277,554,1108};
int f_re[3] = {294,587,1174};
int f_reG[3] = {311,622,1244};
int f_mi[3] = {330,659,1318};
int f_fa[3] = {349,698,1396};
int f_faG[3] = {370,740,1480};
int f_sol[3] = {392,784,1568};
int f_solG[3] = {416,831,1662};
int f_la[3] = {440,880,1760};
int f_laG[3] = {466,932,1864};
int f_si[3] = {494,988,1976};
//Métodos detectar nota
void esDo3(short*);
void esDoG3(short*);
void esRe3(short*);
void esReG3(short*);
void esMi3(short*);
void esFa3(short*);
void esFaG3(short*);
void esSol3(short*);
void esSolG3(short*);
void esLa3(short*);
void esLaG3(short*);
void esSi3(short*);
void esDo4(short*);
void esDoG4(short*);
void esRe4(short*);
```

```
  void esReG4(short*);
  void esMi4(short*);
  void esFa4(short*);
  void esFaG4(short*);
  void esSol4(short*);
  void esSolG4(short*);
  void esLa4(short*);
  void esLaG4(short*);
  void esSi4(short*);
  void esDo5(short*);
  void esDoG5(short*);
  void esRe5(short*);
  void esReG5(short*);
  void esMi5(short*);
  void esFa5(short*);
  void esFaG5(short*);
  void esSol5(short*);
  void esSolG5(short*);
  void esLa5(short*);
  void esLaG5(short*);
  void esSi5(short*);
#endif
```

flauta.c:

```
#include"funciones.h"
int main(int argc, char *argv[]){
  //Leo las cabeceras
  leerCabeceras(argv);
  //Defino variables
  total_muestras_originales=subchunk2size/blockalign;
  printf("Total muestras originales:%d\n",
    total_muestras_originales);
  //Necesitamos que el total de muestras sea una potencia de 2
  total_muestras=calcularNuevoNumeroMuestras(
    total_muestras_originales);
  printf("Nuevo total de muestras:%d\n", total_muestras);
  short *muestras=(short *)malloc(total_muestras * sizeof(short)
    );
  //Leo las muestras
  int t=256;
  leerMuestras(muestras);
  short *aux_muestras=(short *)malloc(t * sizeof(short));
  int i;
  int j;
  for (i = 0; i < total_muestras/t; i++){
```

```
19      for(j=0;j<t;j++){
20        aux_muestras[j]=muestras[aux_conteo];
21        aux_conteo+=1;
22      }
23      calcularFFT(aux_muestras,t);
24    }
25  }
26  void leerCabeceras(char ** argv){
27    entrada = fopen(argv[1], "rb");
28    if(!entrada) {
29      perror("\nFile opening failed");
30      exit(0);
31    }
32    fread(&chunkid,sizeof(int),1,entrada);
33    fread(&chunksize,sizeof(int),1,entrada);
34    fread(&format,sizeof(int),1,entrada);
35    fread(&subchunk1id,sizeof(int),1,entrada);
36    fread(&subchunk1size,sizeof(int),1,entrada);
37    fread(&audioformat,sizeof(short),1,entrada);
38    fread(&numchannels,sizeof(short),1,entrada);
39    fread(&samplerate,sizeof(int),1,entrada);
40    fread(&byterate,sizeof(int),1,entrada);
41    fread(&blockalign,sizeof(short),1,entrada);
42    fread(&bitspersample,sizeof(short),1,entrada);
43    fread(&subchunk2id,sizeof(int),1,entrada);
44    fread(&subchunk2size,sizeof(int),1,entrada);
45  }
46  void leerMuestras(short *muestras){
47    int i=0;
48    while (feof(entrada) == 0){
49      if(i<total_muestras_originales){
50        fread(&muestra,sizeof(short),1,entrada);
51        muestras[i]=muestra;
52        i++;
53      }else{
54        hayheaders=1;
55        fread(&headers,sizeof(short),37,entrada);
56        break;
57      }
58    }
59    //Ajuste por si las muestras originales no fueron potencia de
      dos
60    if(total_muestras_originales<total_muestras){
61      for (i = total_muestras_originales; i < total_muestras; i++)
      {
```

```
62        muestras[i]=0;
63      }
64    }
65    fclose(entrada);
66  }
67  void escribirArchivo(short* muestrasRe,short* muestrasIm){
68    //Escribo el archivo
69    fwrite(&chunkid,sizeof(int),1,salida);
70    fwrite(&chunksize,sizeof(int),1,salida);
71    fwrite(&format,sizeof(int),1,salida);
72    fwrite(&subchunk1id,sizeof(int),1,salida);
73    fwrite(&subchunk1size,sizeof(int),1,salida);
74    fwrite(&audioformat,sizeof(short),1,salida);
75    fwrite(&numchannels,sizeof(short),1,salida);
76    fwrite(&samplerate,sizeof(int),1,salida);
77    fwrite(&byterate,sizeof(int),1,salida);
78    fwrite(&blockalign,sizeof(short),1,salida);
79    fwrite(&bitspersample,sizeof(short),1,salida);
80    fwrite(&subchunk2id,sizeof(int),1,salida);
81    fwrite(&subchunk2size,sizeof(int),1,salida);
82    //Ahora escribo las muestras
83    int i=0;
84    for(i=0;i<total_muestras;i++){
85      fwrite(&muestrasRe[i],sizeof(short),1,salida);
86      fwrite(&muestrasIm[i],sizeof(short),1,salida);
87    }
88    //Y por último los headers de goldwave
89    if(hayheaders){
90      for(i=0;i<37;i++){
91        fwrite(&headers[i],sizeof(short),1,salida);
92      }
93    }
94    fclose(salida);
95  }
96  void calcularFFT(short *muestras_recibidas, int
      total_muestras_recibidas){
97    //Aquí va el algoritmo para la FFT
98    float *Xre=(float *)malloc(total_muestras_recibidas * sizeof(
      float));
99    float *Xim=(float *)malloc(total_muestras_recibidas * sizeof(
      float));
100   int i;
101   //Convierto las muestras de short a float
102   convertirFloat(muestras_recibidas, Xre, Xim,
      total_muestras_recibidas);
```

```
103   //FFT
104   int j, k, fk, m, n, ce, c, w;
105   float arg, seno, coseno, tempr, tempi;
106   //Bit reversal
107   m=log((float)total_muestras_recibidas)/log(2.0);
108   j=w=0;
109   for (i = 0; i < total_muestras_recibidas; i++){
110     if (j>i){
111       SWAP(Xre[i],Xre[j]);
112       SWAP(Xim[i],Xim[j]);
113     }
114     w=total_muestras_recibidas/2;
115     while(w>=2 && j>=w){
116       j-=w;
117       w>>=1;
118     }
119     j+=w;
120   }
121   ce=m;
122   c=0;
123   //Mariposas
124   for (i = 0; i < m; i++) {
125     for(j = 0; j < (int)pow(2,ce-1); j++){
126       n = (int)pow(2,i);
127       for(k = 0; k < n; k++){
128         fk=k*(int)pow(2,ce-1);
129         coseno=cos((-1)*2*PI*fk/total_muestras_recibidas);
130         seno=sin((-1)*2*PI*fk/total_muestras_recibidas);
131         tempr=Xre[c+n];
132         Xre[c+n]=(Xre[c+n]*coseno) - (Xim[c+n]*seno);
133         Xim[c+n]=(Xim[c+n]*coseno) + (tempr*seno);
134         tempr=(Xre[c]+Xre[c+n])/2;
135         tempi=(Xim[c]+Xim[c+n])/2;
136         Xre[c+n]=(Xre[c]-Xre[c+n])/2;
137         Xim[c+n]=(Xim[c]-Xim[c+n])/2;
138         Xre[c]=tempr;
139         Xim[c]=tempi;
140         c++;
141       }
142       c += n;
143     }
144     c = 0;
145     ce -= 1;
146   }
147   short *Reales=(short *)malloc(total_muestras_recibidas *
```

```
    sizeof(short));
148   short *Imaginarias=(short *)malloc(total_muestras_recibidas *
      sizeof(short));
149   //Regreso las muestras calculadas a short
150   convertirShort(Reales,Imaginarias,Xre,Xim,
      total_muestras_recibidas);
151   obtenerNota(Reales,total_muestras_recibidas);
152 }
153 int calcularNuevoNumeroMuestras(int total){
154   if ((total & (total-1))==0){
155     puts("Ya es potencia de 2");
156   }else{
157     puts("No es potencia de 2");
158     int i;
159     i=(int)ceil((float)log(total_muestras_originales)/(float)log
      (2));
160     printf("i:%d\n", i);
161     total=pow(2,i);
162   }
163   return total;
164 }
165 void convertirFloat(short *muestras, float *Xre, float *Xim, int
      total_muestras_recibidas){
166   int i;
167   for (i = 0; i < total_muestras_recibidas; i++){
168     Xre[i]=(float)muestras[i]/(float)(SHRT_MAX);
169     Xim[i]=0.0;
170   }
171 }
172 void convertirShort(short *Reales, short *Imaginarias, float *
      Xre, float *Xim, int total_muestras_recibidas){
173   int i;
174   for (i = 0; i < total_muestras_recibidas; i++){
175     Reales[i]=Xre[i]*(SHRT_MAX);
176     Imaginarias[i]=Xim[i]*(SHRT_MAX);
177   }
178 }
179 void obtenerNota(short *Xre,int muestras_recibidas){
180   //Obtengo la duración del archivo
181   duracion=(float)muestras_recibidas/(float)samplerate;
182   //printf("Duracion del archivo: %f\n", duracion);
183   int i;
184   for(i=0;i<muestras_recibidas/2;i++){
185     if(Xre[i]>amp)
186     printf("Xre[%d] = %d\n", i, Xre[i]);
```

```
187   }
188   //Aquí reviso que notas fueron identificadas
189   esDo3(Xre);
190   esDoG3(Xre);
191   esRe3(Xre);
192   esReG3(Xre);
193   esMi3(Xre);
194   esFa3(Xre);
195   esFaG3(Xre);
196   esSol3(Xre);
197   esSolG3(Xre);
198   esLa3(Xre);
199   esLaG3(Xre);
200   esSi3(Xre);
201   esDo4(Xre);
202   esDoG4(Xre);
203   esRe4(Xre);
204   esReG4(Xre);
205   esMi4(Xre);
206   esFa4(Xre);
207   esFaG4(Xre);
208   esSol4(Xre);
209   esSolG4(Xre);
210   esLa4(Xre);
211   esLaG4(Xre);
212   esSi4(Xre);
213   esDo5(Xre);
214   esDoG5(Xre);
215   esRe5(Xre);
216   esReG5(Xre);
217   esMi5(Xre);
218   esFa5(Xre);
219   esFaG5(Xre);
220   esSol5(Xre);
221   esSolG5(Xre);
222   esLa5(Xre);
223   esLaG5(Xre);
224   esSi5(Xre);
225 }
226 void esDo3(short *Xre){
227   aux1=round((float)f_do[0]*duracion);
228   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
229     puts("La nota corresponde a un do3");
230     bandera=1;
231   }
```

```
232 }
233 void esDoG3(short *Xre){
234   aux1=round((float)f_doG[0]*duracion);
235   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
236     puts("La nota corresponde a un do#3");
237     bandera=1;
238   }
239 }
240 void esRe3(short *Xre){
241   aux1=round((float)f_re[0]*duracion);
242   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
243     puts("La nota corresponde a un re3");
244     bandera=1;
245   }
246 }
247 void esReG3(short *Xre){
248   aux1=round((float)f_reG[0]*duracion);
249   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
250     puts("La nota corresponde a un re#3");
251     bandera=1;
252   }
253 }
254 void esMi3(short *Xre){
255   aux1=round((float)f_mi[0]*duracion);
256   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
257     puts("La nota corresponde a un mi3");
258     bandera=1;
259   }
260 }
261 void esFa3(short *Xre){
262   aux1=round((float)f_fa[0]*duracion);
263   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
264     puts("La nota corresponde a un fa3");
265     bandera=1;
266   }
267 }
268 void esFaG3(short *Xre){
269   aux1=round((float)f_faG[0]*duracion);
270   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
271     puts("La nota corresponde a un fa#3");
272     bandera=1;
273   }
274 }
275 void esSol3(short *Xre){
276   aux1=round((float)f_sol[0]*duracion);
```

```
277   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
278     puts("La nota corresponde a un sol3");
279     bandera=1;
280   }
281 }
282 void esSolG3(short *Xre){
283   aux1=round((float)f_solG[0]*duracion);
284   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
285     puts("La nota corresponde a un sol#3");
286     bandera=1;
287   }
288 }
289 void esLa3(short *Xre){
290   aux1=round((float)f_la[0]*duracion);
291   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
292     puts("La nota corresponde a un la3");
293     bandera=1;
294   }
295 }
296 void esLaG3(short *Xre){
297   aux1=round((float)f_laG[0]*duracion);
298   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
299     puts("La nota corresponde a un la#3");
300     bandera=1;
301   }
302 }
303 void esSi3(short *Xre){
304   aux1=round((float)f_si[0]*duracion);
305   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
306     puts("La nota corresponde a un si3");
307     bandera=1;
308   }
309 }
310 void esDo4(short *Xre){
311   aux1=round((float)f_do[1]*duracion);
312   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
313     puts("La nota corresponde a un do4");
314     bandera=1;
315   }
316 }
317 void esDoG4(short *Xre){
318   aux1=round((float)f_doG[0]*duracion);
319   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
320     puts("La nota corresponde a un do#4");
321     bandera=1;
```

```
322   }
323 }
324 void esRe4(short *Xre){
325   aux1=round((float)f_re[1]*duracion);
326   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
327     puts("La nota corresponde a un re4");
328     bandera=1;
329   }
330 }
331 void esReG4(short *Xre){
332   aux1=round((float)f_re[1]*duracion);
333   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
334     puts("La nota corresponde a un re#4");
335     bandera=1;
336   }
337 }
338 void esMi4(short *Xre){
339   aux1=round((float)f_mi[1]*duracion);
340   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
341     puts("La nota corresponde a un mi4");
342     bandera=1;
343   }
344 }
345 void esFa4(short *Xre){
346   aux1=round((float)f_fa[1]*duracion);
347   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
348     puts("La nota corresponde a un fa4");
349     bandera=1;
350   }
351 }
352 void esFaG4(short *Xre){
353   aux1=round((float)f_faG[1]*duracion);
354   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
355     puts("La nota corresponde a un fa#4");
356     bandera=1;
357   }
358 }
359 void esSol4(short *Xre){
360   aux1=round((float)f_sol[1]*duracion);
361   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
362     puts("La nota corresponde a un sol4");
363     bandera=1;
364   }
365 }
366 void esSolG4(short *Xre){
```

```
367   aux1=round((float)f_solG[1]*duracion);
368   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
369     puts("La nota corresponde a un sol#4");
370     bandera=1;
371   }
372 }
373 void esLa4(short *Xre){
374   aux1=round((float)f_la[1]*duracion);
375   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
376     puts("La nota corresponde a un la4");
377     bandera=1;
378   }
379 }
380 void esLaG4(short *Xre){
381   aux1=round((float)f_laG[1]*duracion);
382   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
383     puts("La nota corresponde a un la#4");
384     bandera=1;
385   }
386 }
387 void esSi4(short *Xre){
388   aux1=round((float)f_si[1]*duracion);
389   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
390     puts("La nota corresponde a un si4");
391     bandera=1;
392   }
393 }
394 void esDo5(short *Xre){
395   aux1=round((float)f_do[2]*duracion);
396   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
397     puts("La nota corresponde a un do5");
398     bandera=1;
399   }
400 }
401 void esDoG5(short *Xre){
402   aux1=round((float)f_doG[0]*duracion);
403   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
404     puts("La nota corresponde a un do#5");
405     bandera=1;
406   }
407 }
408 void esRe5(short *Xre){
409   aux1=round((float)f_re[2]*duracion);
410   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
411     puts("La nota corresponde a un re5");
```

```
412       bandera=1;
413     }
414   }
415   void esReG5(short *Xre){
416     aux1=round((float)f_reG[2]*duracion);
417     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
418       puts("La nota corresponde a un re#5");
419       bandera=1;
420     }
421   }
422   void esMi5(short *Xre){
423     aux1=round((float)f_mi[2]*duracion);
424     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
425       puts("La nota corresponde a un mi5");
426       bandera=1;
427     }
428   }
429   void esFa5(short *Xre){
430     aux1=round((float)f_fa[2]*duracion);
431     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
432       puts("La nota corresponde a un fa5");
433       bandera=1;
434     }
435   }
436   void esFaG5(short *Xre){
437     aux1=round((float)f_faG[2]*duracion);
438     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
439       puts("La nota corresponde a un fa#5");
440       bandera=1;
441     }
442   }
443   void esSol5(short *Xre){
444     aux1=round((float)f_sol[2]*duracion);
445     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
446       puts("La nota corresponde a un sol5");
447       bandera=1;
448     }
449   }
450   void esSolG5(short *Xre){
451     aux1=round((float)f_solG[2]*duracion);
452     if(Xre[aux1]>amp || Xre[aux1+1]>amp){
453       puts("La nota corresponde a un sol#5");
454       bandera=1;
455     }
456   }
```

```
457 void esLa5(short *Xre){
458   aux1=round((float)f_la[2]*duracion);
459   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
460     puts("La nota corresponde a un la5");
461     bandera=1;
462   }
463 }
464 void esLaG5(short *Xre){
465   aux1=round((float)f_laG[2]*duracion);
466   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
467     puts("La nota corresponde a un la#5");
468     bandera=1;
469   }
470 }
471 void esSi5(short *Xre){
472   aux1=round((float)f_si[2]*duracion);
473   if(Xre[aux1]>amp || Xre[aux1+1]>amp){
474     puts("La nota corresponde a un si5");
475     bandera=1;
476   }
477 }
```

3. Prueba

Para comprobar el funcionamiento del proyecto, se realizo la siguiente grabación, la cual contiene una pequeña parte del Himno a la Alegría. Cuyas notas (de ese fragmento) son:
si4-si4-do5-re5-re5-do5
si4-la4-sol4-sol4-la4-si4-si4-la4-la4
si4-si4-do5-re5-re5-do5
El archivo wav con la grabación se aprecia en la Figura 4:

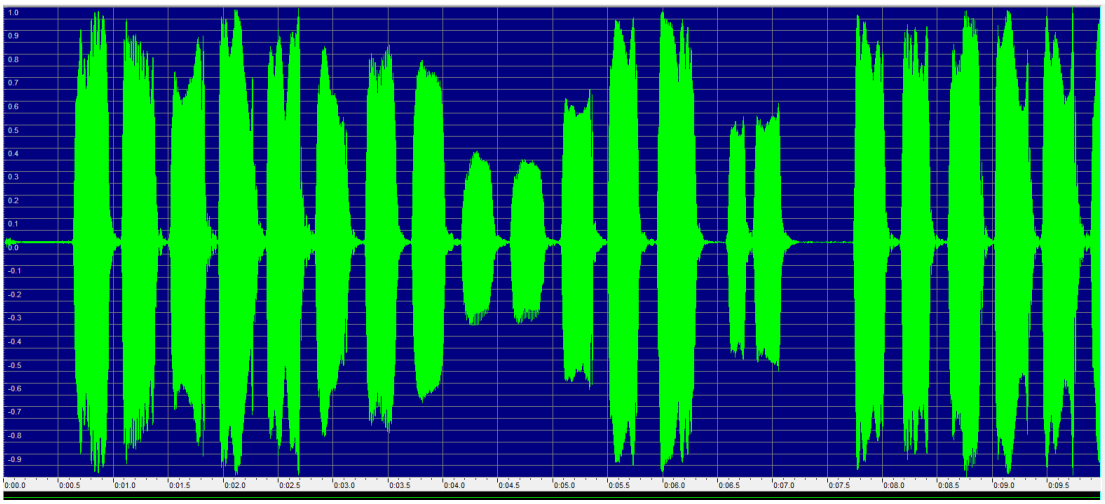
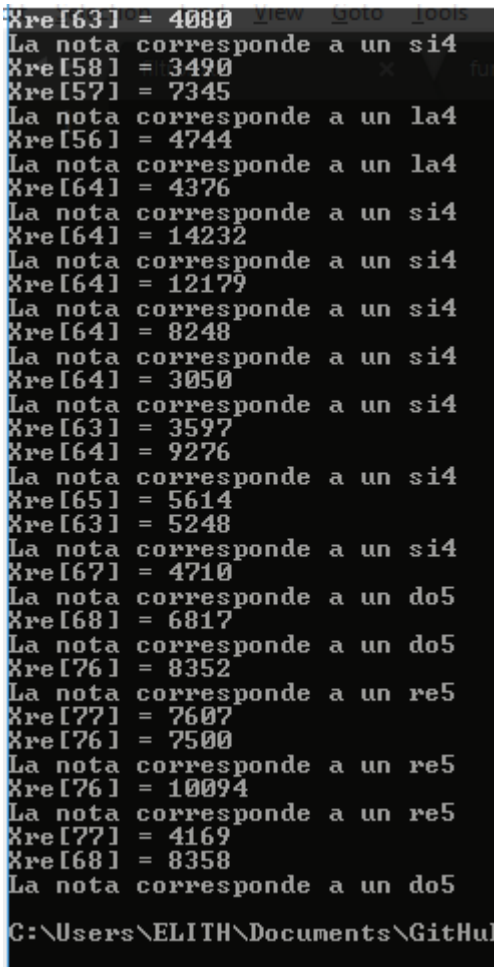


Figura 4: Wav generado por la grabación-Contiene un fragmento del Himno a la Alegría

Al ingresar esta grabación al programa desarrollado, se obtuvo la siguiente salida:

```
No es potencia de 2
i:16
Nuevo total de muestras:65536
Xre[64] = 8243
La nota corresponde a un si4
Xre[64] = 3216
La nota corresponde a un si4
Xre[63] = 3303
La nota corresponde a un si4
Xre[65] = 12713
Xre[63] = 5845
La nota corresponde a un si4
Xre[68] = 10520
La nota corresponde a un do5
Xre[68] = 8901
La nota corresponde a un do5
Xre[68] = 10753
La nota corresponde a un do5
Xre[68] = 5451
La nota corresponde a un do5
Xre[77] = 8695
Xre[76] = 10421
La nota corresponde a un re5
Xre[75] = 4184
La nota corresponde a un re5
Xre[75] = 3455
Xre[76] = 9680
La nota corresponde a un re5
Xre[76] = 7870
La nota corresponde a un re5
Xre[76] = 13018
La nota corresponde a un re5
Xre[73] = 3117
Xre[67] = 9230
La nota corresponde a un do5
Xre[67] = 3571
La nota corresponde a un do5
Xre[67] = 7359
La nota corresponde a un do5
Xre[66] = 4091
Xre[63] = 3129
La nota corresponde a un si4
Xre[63] = 7153
La nota corresponde a un si4
Xre[63] = 6830
La nota corresponde a un si4
Xre[63] = 5098
La nota corresponde a un si4
Xre[57] = 4041
La nota corresponde a un la4
Xre[56] = 10304
La nota corresponde a un la4
Xre[56] = 6290
La nota corresponde a un la4
Xre[50] = 4028
La nota corresponde a un sol4
Xre[50] = 3006
La nota corresponde a un sol4
Xre[56] = 3089
La nota corresponde a un la4
Xre[57] = 8017
La nota corresponde a un la4
Xre[57] = 9144
La nota corresponde a un la4
Xre[57] = 4107
La nota corresponde a un la4
Xre[56] = 3419
La nota corresponde a un la4
Xre[65] = 4007
Xre[64] = 8566
La nota corresponde a un si4
Xre[63] = 4792
```


Xre[63] = 4080
La nota corresponde a un si4
Xre[58] =3490
Xre[57] = 7345
La nota corresponde a un la4
Xre[56] = 4744
La nota corresponde a un la4
Xre[64] = 4376
La nota corresponde a un si4
Xre[64] = 14232
La nota corresponde a un si4
Xre[64] = 12179
La nota corresponde a un si4
Xre[64] = 8248
La nota corresponde a un si4
Xre[64] = 3050
La nota corresponde a un si4
Xre[63] = 3597
Xre[64] = 9276
La nota corresponde a un si4
Xre[65] = 5614
Xre[63] = 5248
La nota corresponde a un si4
Xre[67] = 4710
La nota corresponde a un do5
Xre[68] = 6817
La nota corresponde a un do5
Xre[76] = 8352
La nota corresponde a un re5
Xre[77] = 7607
Xre[76] = 7500
La nota corresponde a un re5
Xre[76] = 10094
La nota corresponde a un re5
Xre[77] = 4169
Xre[68] = 8358
La nota corresponde a un do5
C:\Users\ELITH\Documents\GitHu

4. Conclusiones

Como se pudo observar en la prueba realizada, el programa detectó correctamente todas las notas tocadas en la grabación, por lo cual podemos decir que el objetivo del proyecto se cumplió, al menos hasta el alcance establecido (detectar todas las notas tocadas por una flauta dulce en una grabación). Pero el trabajo realizado puede ser extendido en proyectos más complejos, los cuales permitan determinar que notas y por cuanto tiempo fueron tocadas, siendo útil para la generación de partituras o archivos midi a partir de una grabación contenida en un archivo wav. Además de que el proyecto puede ser extendido no solo para la detección de notas musicales tocadas por una flauta dulce, si no para más instrumentos de viento.
La parte más compleja de realizar del proyecto fue determinar la resolución necesaria para evitar el traslape de las notas, debido a que algunas notas tienen una frecuencia muy cercana a la de otras, y si no es determinada correctamente la resolución es posible que se generen traslapes entre las mismas y el programa puede confundir distintas notas o marcar que una nota que no fue tocada fue detectada.

Referencias

[1] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 2011.

[2] A. V. Oppenheim, *Tratamiento de señales en tiempo discreto*. Pearson, 2009.