PRÁCTICA 4

TRANSFORMADA DISCRETA DE FOURIER INVERSA

Teoría de comunicaciones y señales

Profesor: Gutiérrez Aldana Eduardo

Alumno:

Amador Nava Miguel Ángel

Grupo: 3CM8

Índice

Introducción

Se implementará la Transformada Discreta de Fourier Inversa (IDFT) en lenguaje C, con la cual se pasará del dominio de la frecuencia al del tiempo, haciendo el análisis de esta.

Objetivos

Generales: Implementación de la IDFT

Específicos:

- Comprobar que se genera una función en el tiempo con la misma frecuencia indicada en una señal en el dominio de la frecuencia.

Software y equipo utilizado

- Ubuntu 17.10
- Wine 1.7
- GoldWave 4.26
- Sublime Text 3

Marco teórico

Transformada Discreta de Fourier Inversa

La transformada Discreta de Fourier Inversa o IDFT (del inglés, Inverse Discrete Fourier Transform) está definida por la ecuación 1.

$$X_n = \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1$$

Ecuación 1 IDFT

Algunos autores al sacar la Transformada Discreta de Fourier (DFT) no sacan el promedio por lo cuál es necesario hacerlo en la inversa como en la ecuación 2.

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1$$

Ecuación 2 IDFT con cálculo del promedio

También existen autores que sacan la raíz del número de muestras totales al que se le aplicará la DFT y posteriormente la IDFT y en ambos pasos se distribuye el calculo del promedio.

$$X_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1$$

Ecuación 3 IDFT con cálculo de la raíz del promedio.

En este caso se saco el promedio en la DFT por lo cual su usará la ecuacion 1. Y usando la fórmula de Euler obtenemos la ecuación 4.

$$X_n = \sum_{k=0}^{N-1} X_k \left[cos\left(\frac{2\pi}{N} kn\right) + jsen\left(\frac{2\pi}{N} kn\right) \right]$$

Ecuación 4

Planteamiento del problema

Se pide realizar la Transformada Discreta de Fourier Inversa y guardar la parte real en el canal 1 y la parte imaginaria en el canal 2.

Para probar el programa se usará **las salidas de la DFT** de los archivos:

1. Cosenos
   - Archivos monoaurales de 2000 muestras por segundo de 0.016s de duración.
   - $Cos(2 * \Pi * \mathrm{f} * \mathrm{t} * \mathrm{n})$
   - $f = 62.5$
   - $n = 0,1,2, \dots ,32$

2. Senos
   - Archivos monoaurales de 2000 muestras por segundo de 0.016s de duración.
   - $Sin(2 * \Pi * \mathrm{f} * \mathrm{t} * \mathrm{n})$
   - $f = 62.5$
   - $n = 0,1,2, \dots ,32$

3. Suma de senos y cosenos
   - Archivo monoaural de 2000 muestras por segundo de 0.016s de duración.
   - $(Cos(2 * \Pi * \mathrm{f} * \mathrm{t} * 2) + Sin(2 * \Pi * \mathrm{f} * \mathrm{t} * 4))/2$
   - $f = 62.5$

Análisis Teórico

Transformada Discreta de Fourier Inversa

Como se pide que en el canal 1 se envíen los valores reales y en el canal 2 los valores imaginarios, de la ecuación 4 deducimos:

Canal 1:

$$X_n = \sum_{k=0}^{N-1} X_k cos\left(\frac{2\pi}{N} kn\right) \quad n = 0, \dots, N-1$$

Canal 2:

$$X_n = \sum_{n=0}^{N-1} X_k sen\left(\frac{2\pi}{N} kn\right) \quad n = 0, \dots, N-1$$

```
//IDFT
double C=2*PI/numMuestrasAudio;

for(int n=0;n<numMuestrasAudio;n++){
      muestrasAudioSalida[2*n]=0;
      muestrasAudioSalida[2*n+1]=0;

      for(int k=0;k<numMuestrasAudio;k++){      //Parte Real
      muestrasAudioSalida[2*n]+=((muestrasAudioCanal1[k]*cos(C*k*n))-
(muestrasAudioCanal2[k]*sin(C*k*n)));
      }

      for(int k=0;k<numMuestrasAudio;k++){      //Parte Imaginaria
      muestrasAudioSalida[2*n+1]+=((muestrasAudioCanal1[k]*sin(C*k*n))+(m
uestrasAudioCanal2[k]*cos(C*k*n)));
      }

}
```

Resultados

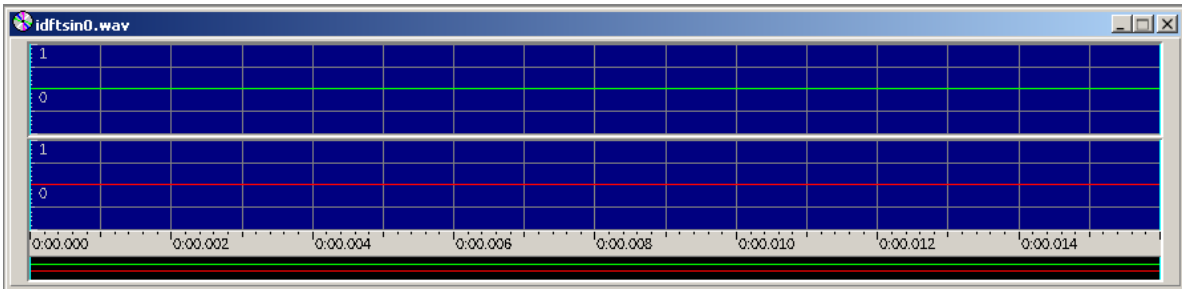

Ilustración 1 Seno de 0 Hz

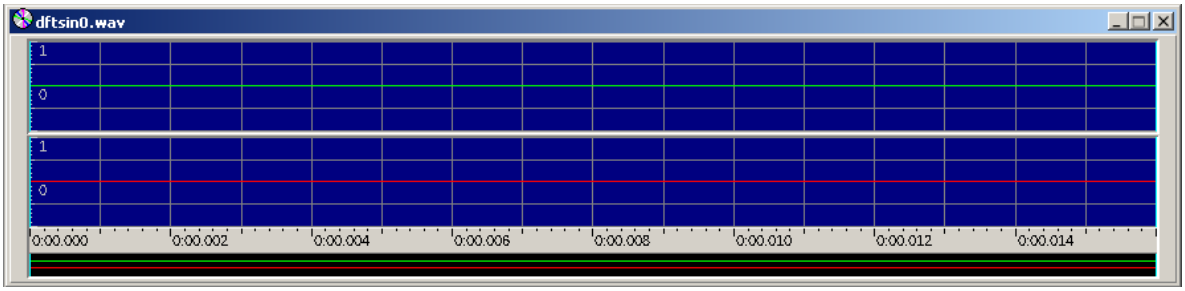

Ilustración 2 DFT del seno de 0 Hz

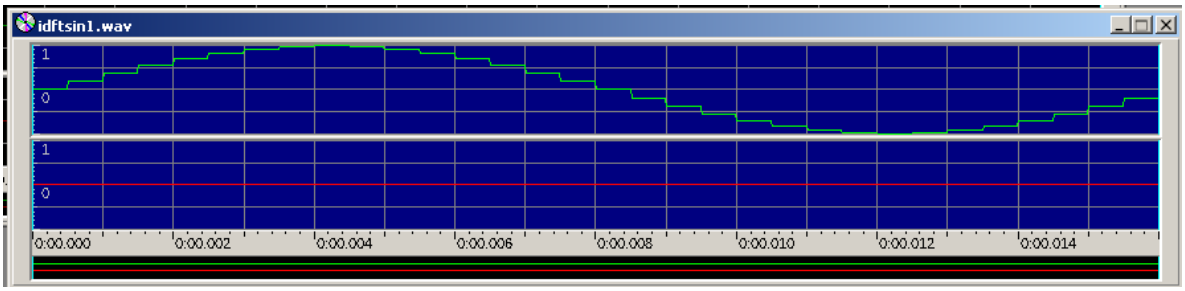

Ilustración 3 Seno de 62.5 Hz

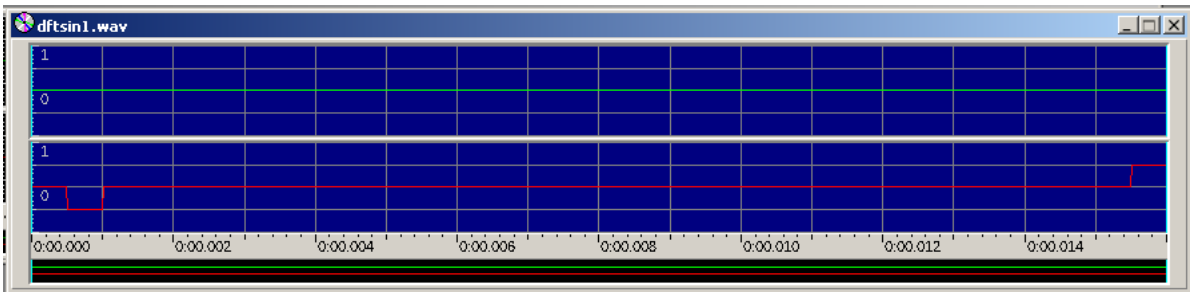

Ilustración 4 DFT del Seno de 62.5 Hz

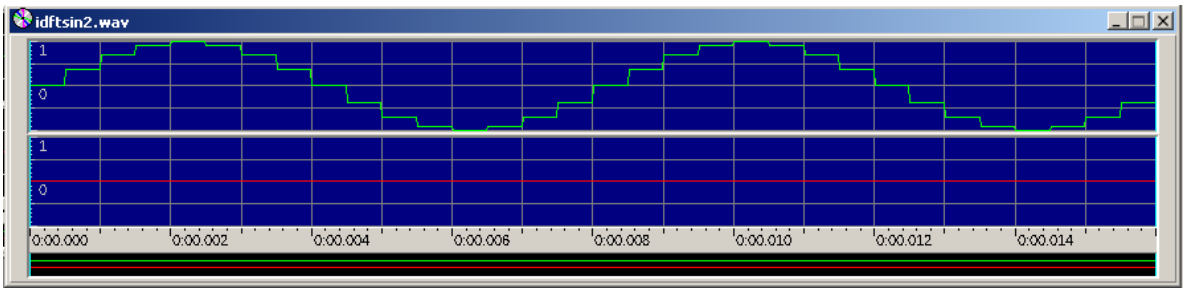

Ilustración 5 Seno de 125 Hz

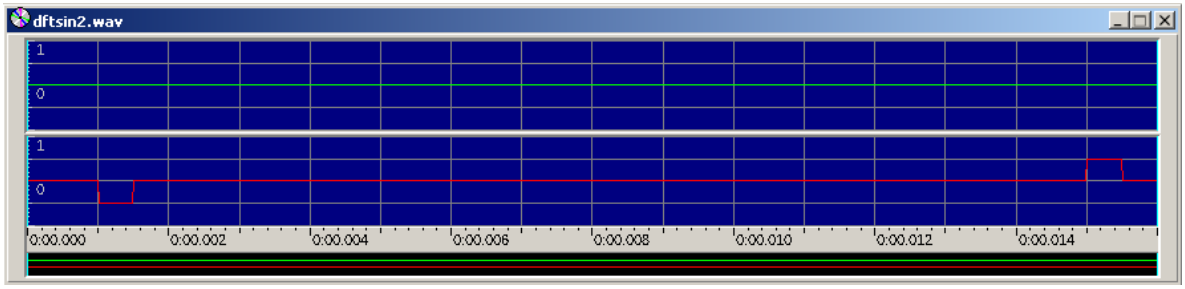

Ilustración 6 DFT del seno de 125 Hz

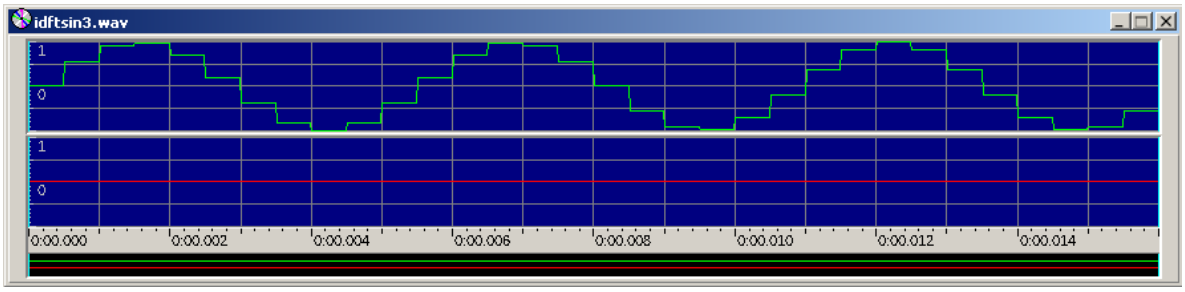

Ilustración 7 Seno de 187.5 Hz

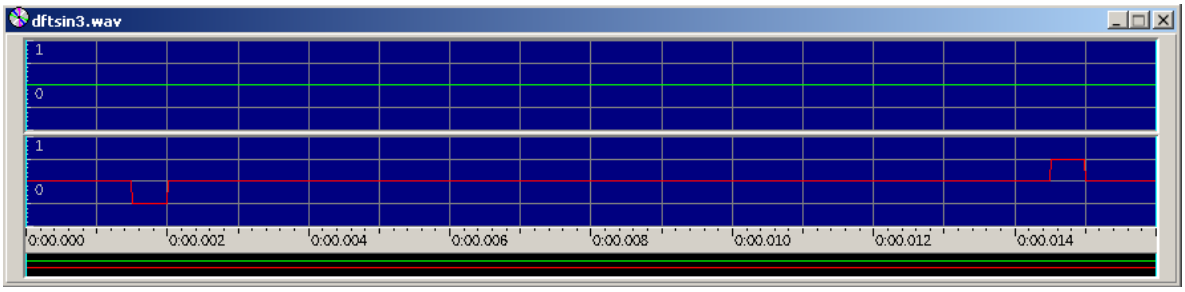

Ilustración 8 DFT del Seno de 187.5 Hz

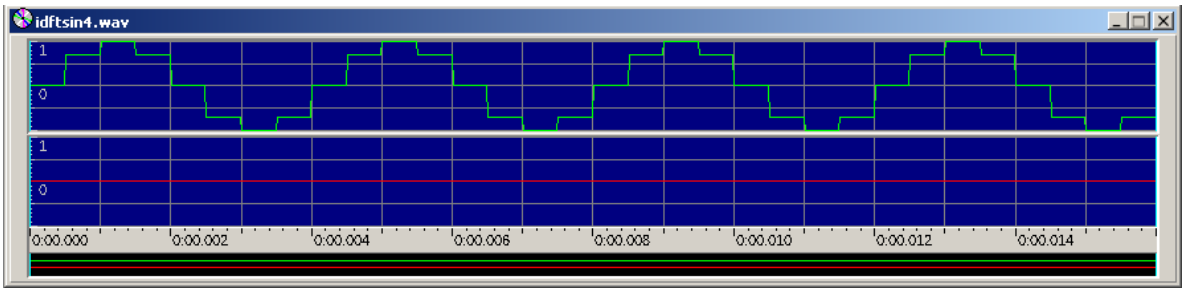

Ilustración 9 Seno de 250 Hz

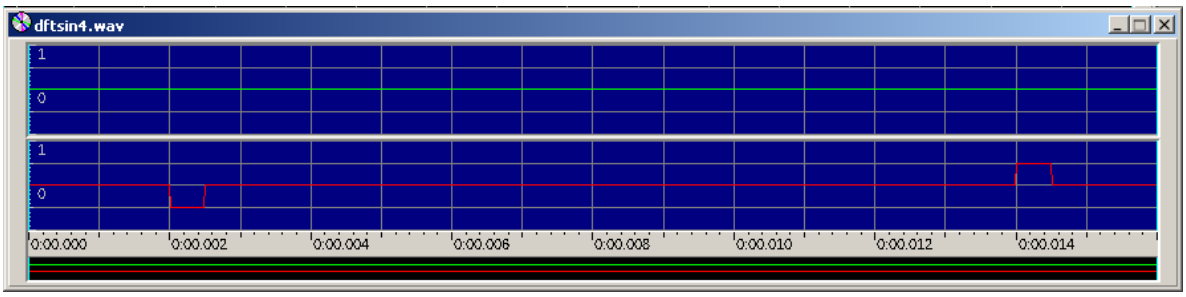

Ilustración 10 DFT del Seno de 250Hz

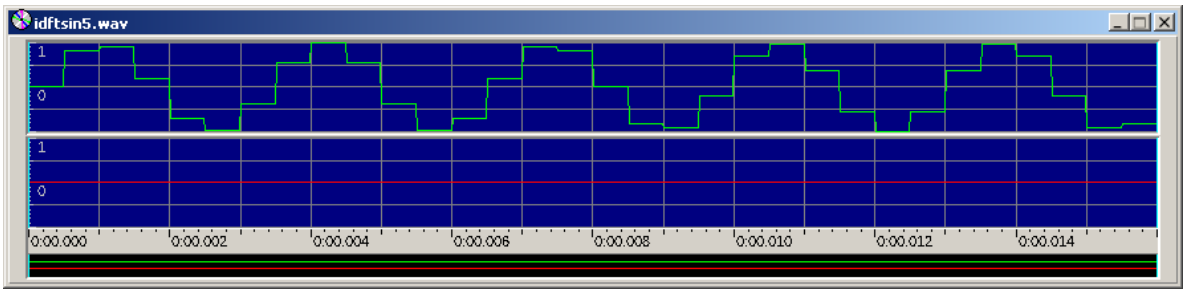

Ilustración 11 Seno de 312.5 Hz

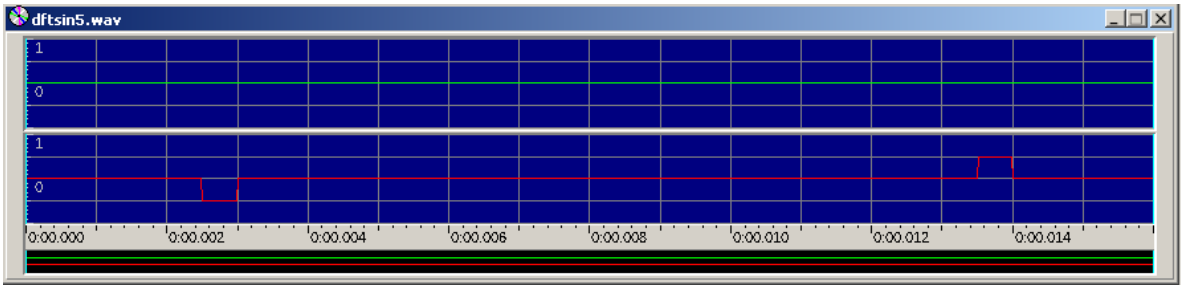

Ilustración 12 DFT del Seno 312.5 Hz

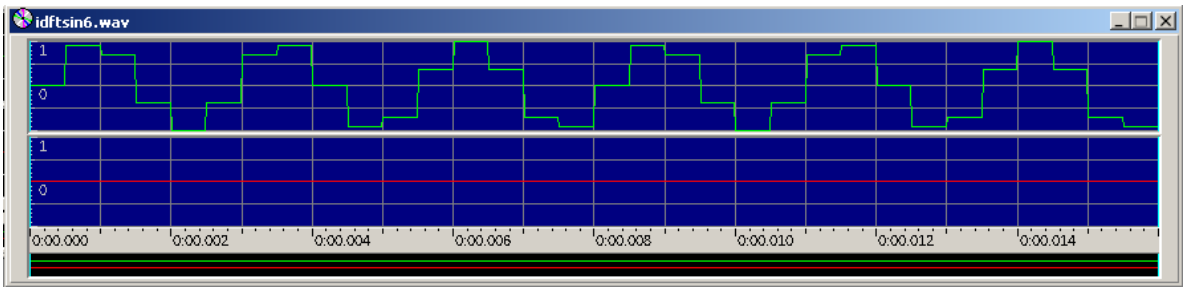

Ilustración 13 Seno de 375 Hz

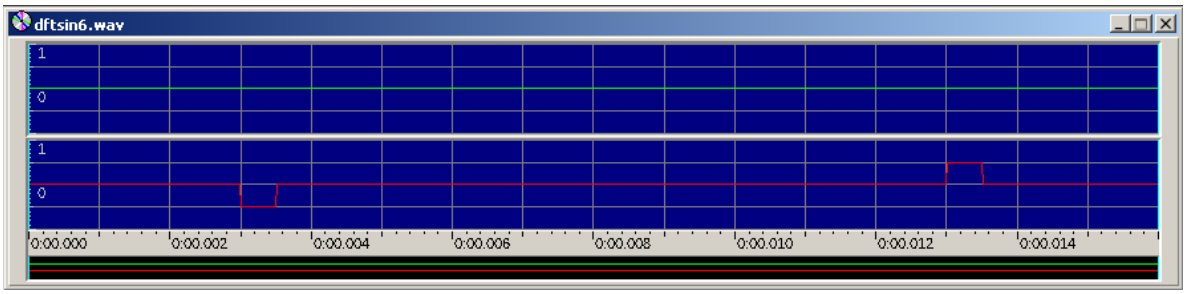

Ilustración 14 DFT del Seno de 375 Hz

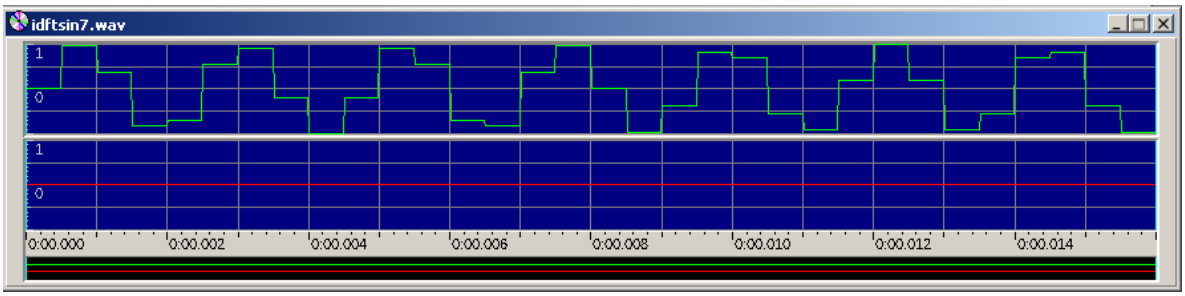

Ilustración 15 Seno de 437.5 Hz

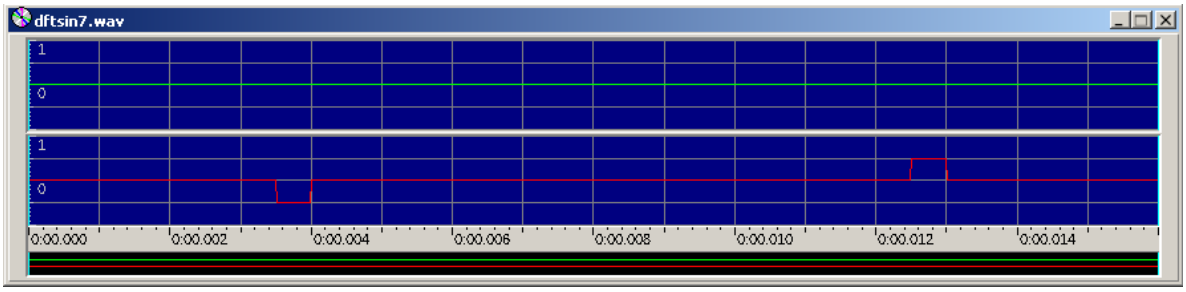

Ilustración 16 DFT del Seno de 437.5 Hz

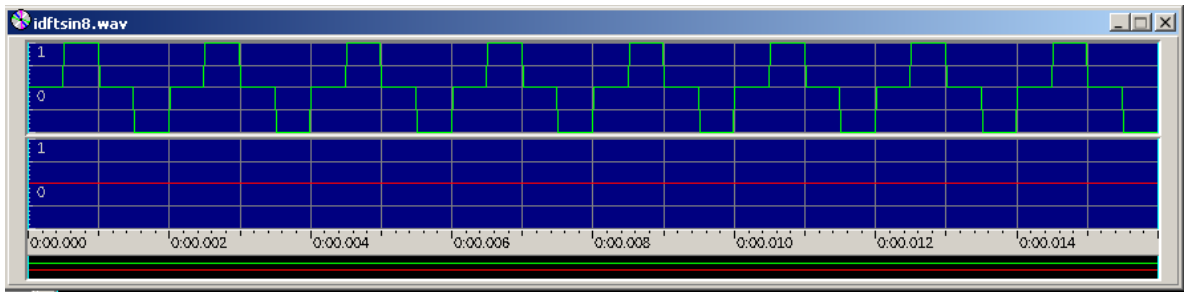

Ilustración 17 Seno de 500 Hz

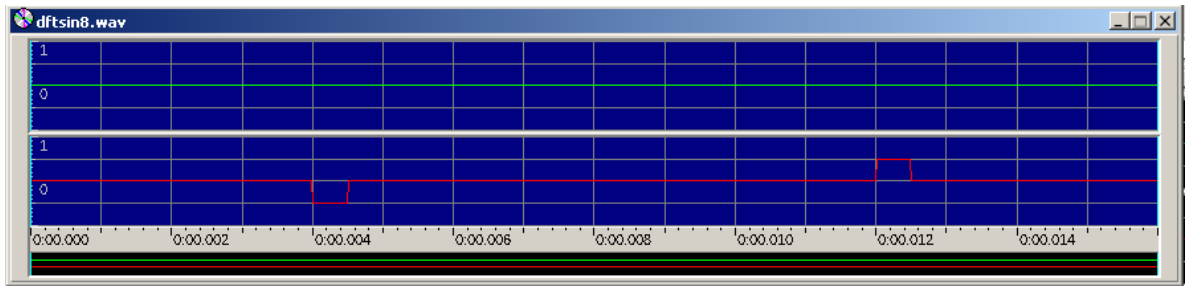

Ilustración 18 DFT del Seno de 500 Hz

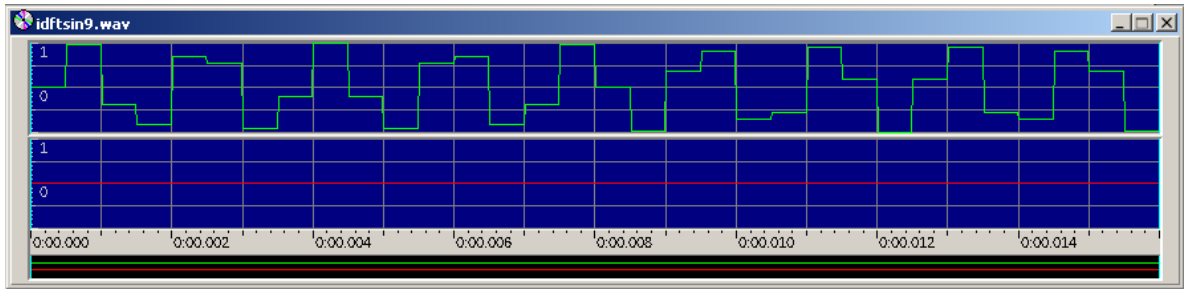

Ilustración 19 seno de 562.5 Hz

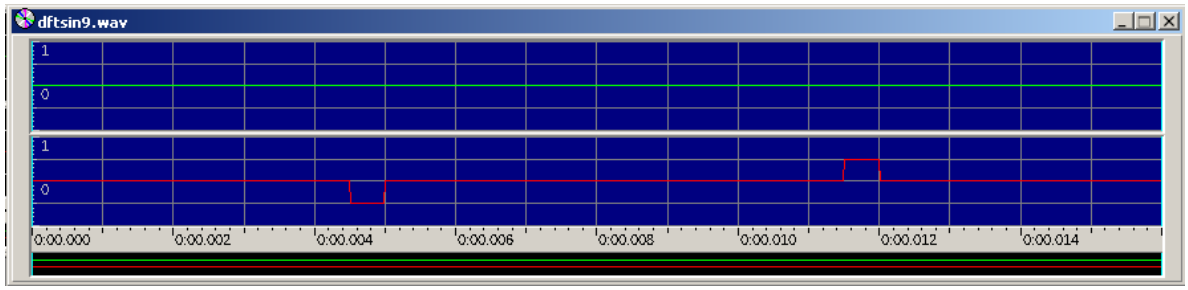

Ilustración 20 DFT del Seno de 562.5 Hz

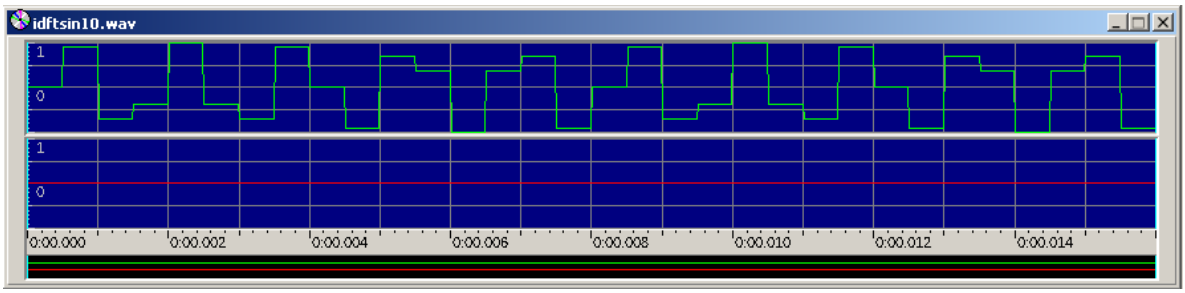

Ilustración 21 Seno de 650 Hz

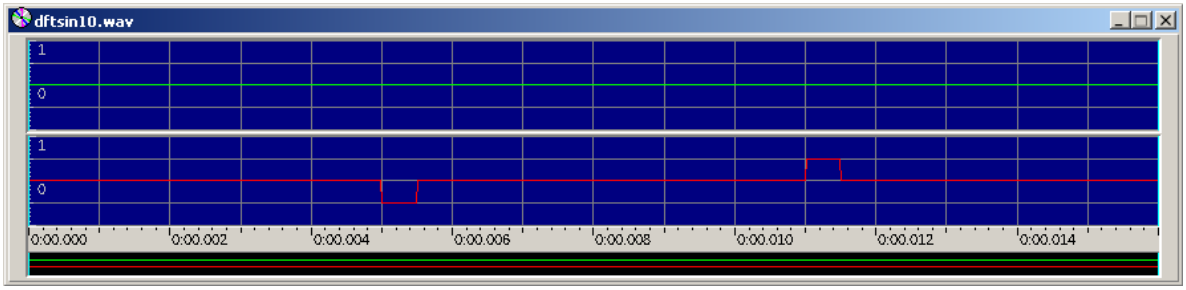

Ilustración 22 DFT del Seno de 650 Hz

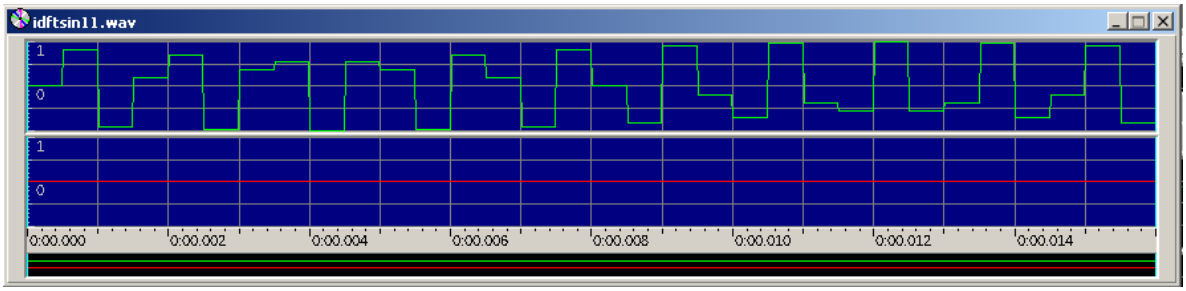

Ilustración 23 Seno de 687.5 Hz

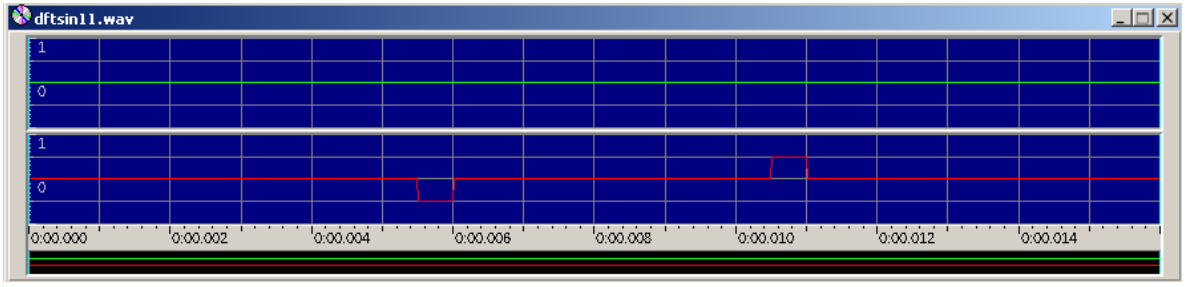

Ilustración 24 DFT del Seno de 687.5 Hz

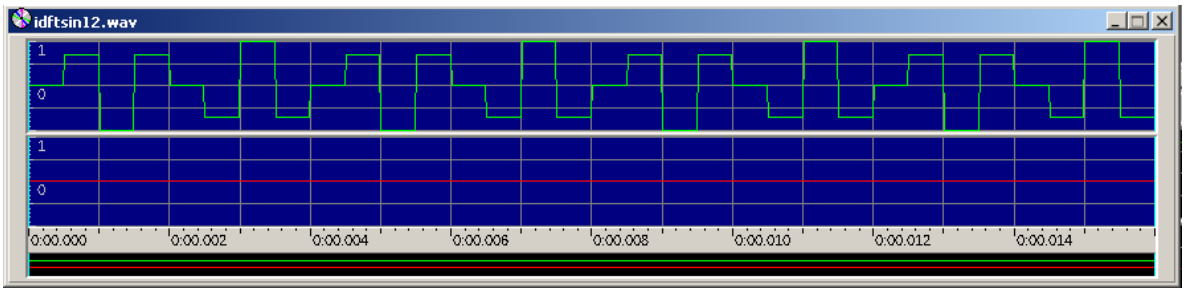

Ilustración 25 Seno de 750 Hz

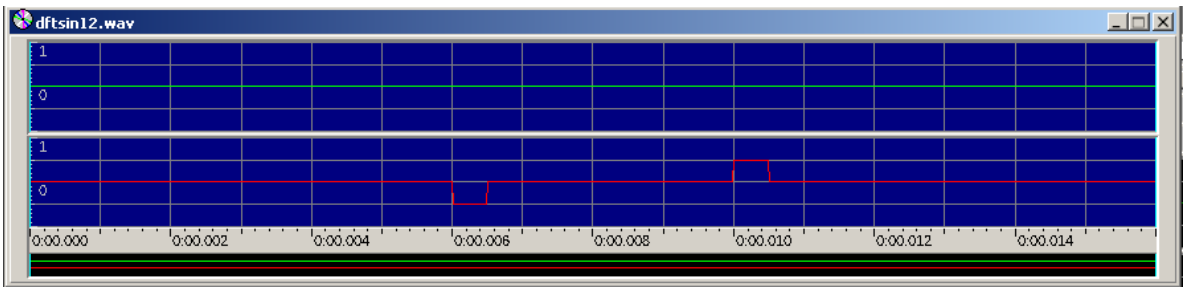

Ilustración 26 DFT del Seno de 750 Hz

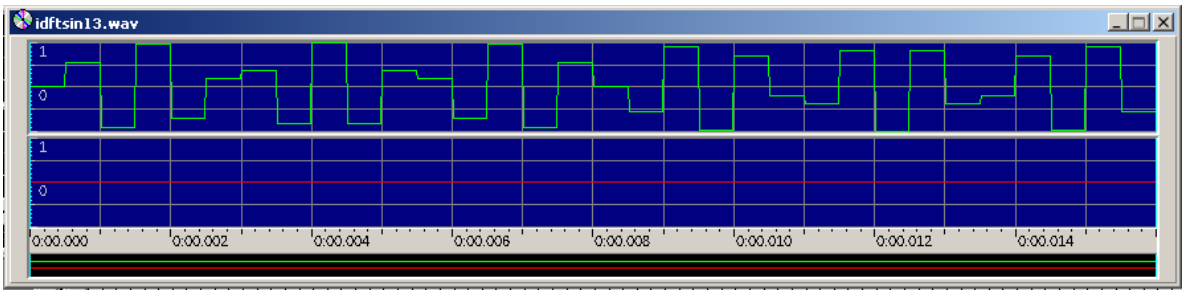

Ilustración 27 Seno de 812.5 Hz

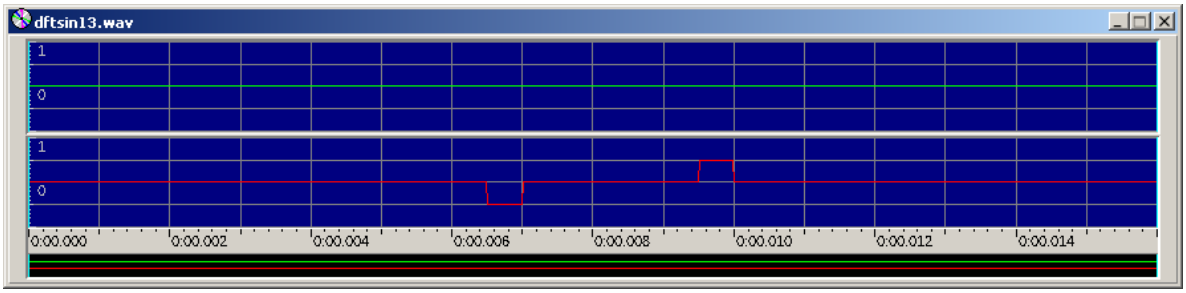

Ilustración 28 DFT del Seno de 812.5 Hz

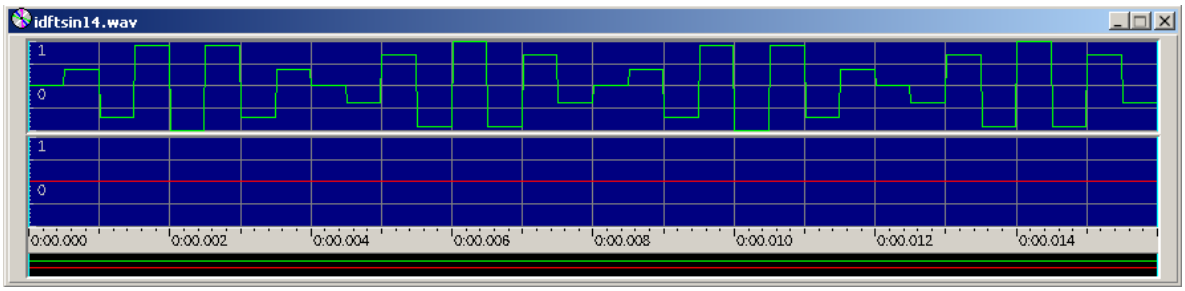

Ilustración 29 Seno de 875 Hz

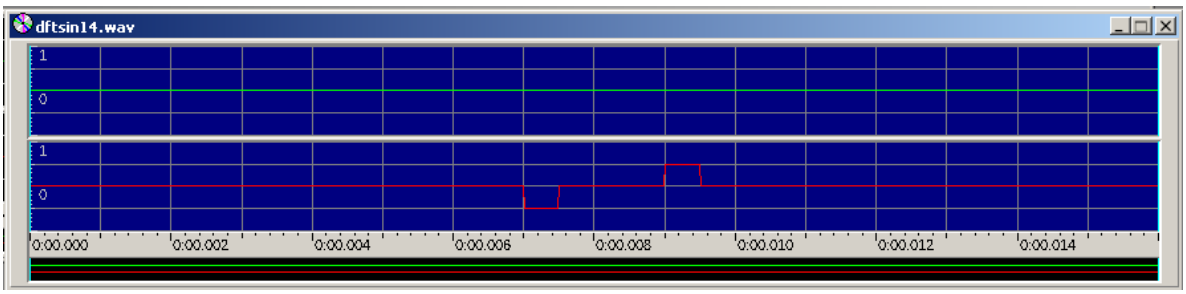

Ilustración 30 DFT del Seno de 875 Hz

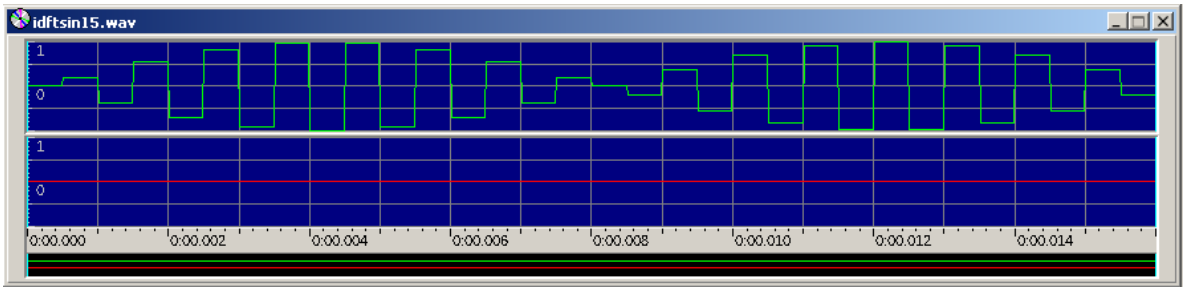

Ilustración 31 Seno de 937.5

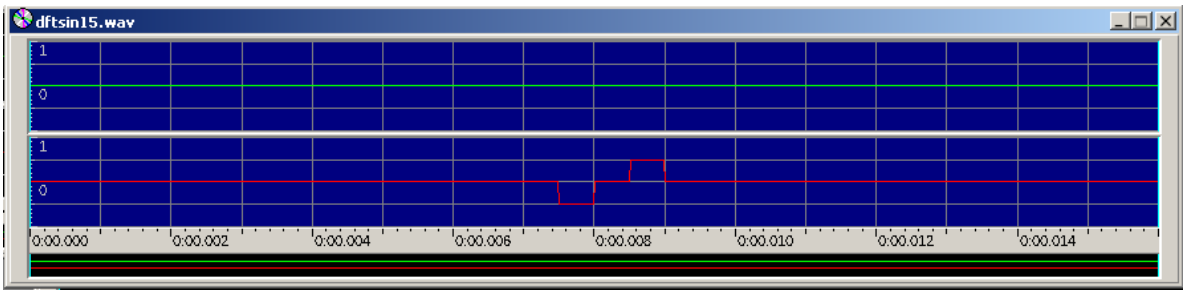

Ilustración 32 DFT del Seno de 937.5

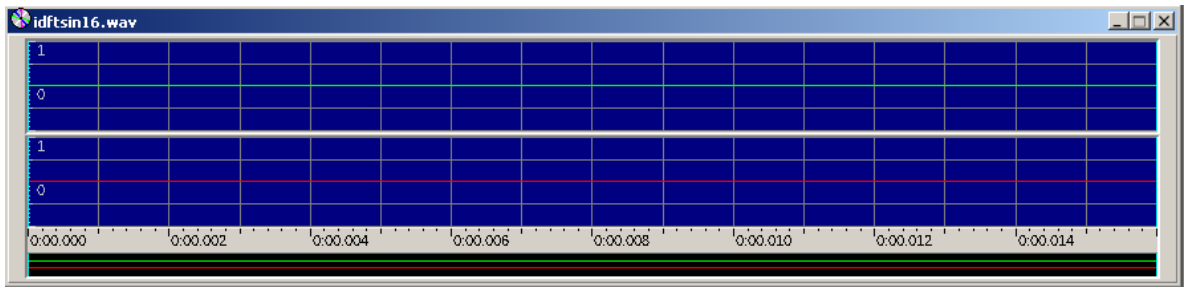

Ilustración 33 Seno de 1000 Hz

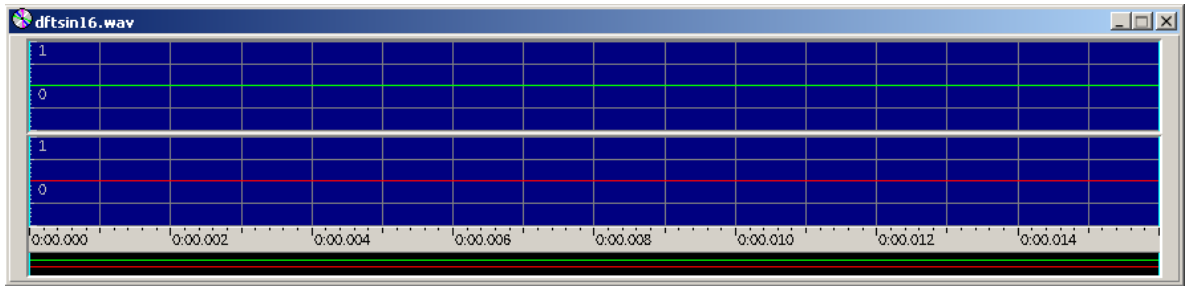

Ilustración 34 DFT del Seno de 1000 Hz

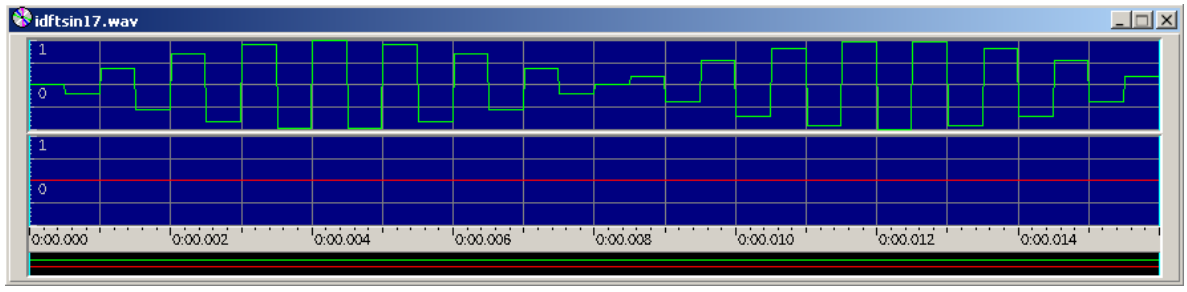

Ilustración 35 Seno de 1062.5 Hz

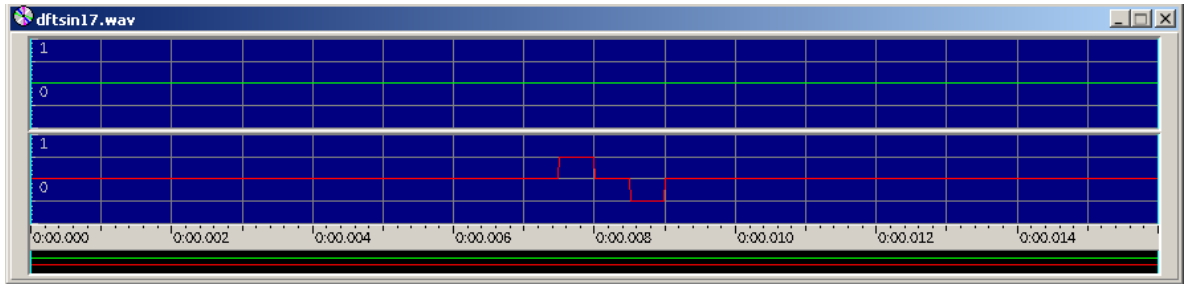

Ilustración 36 DFT del Seno de 1062.5 Hz

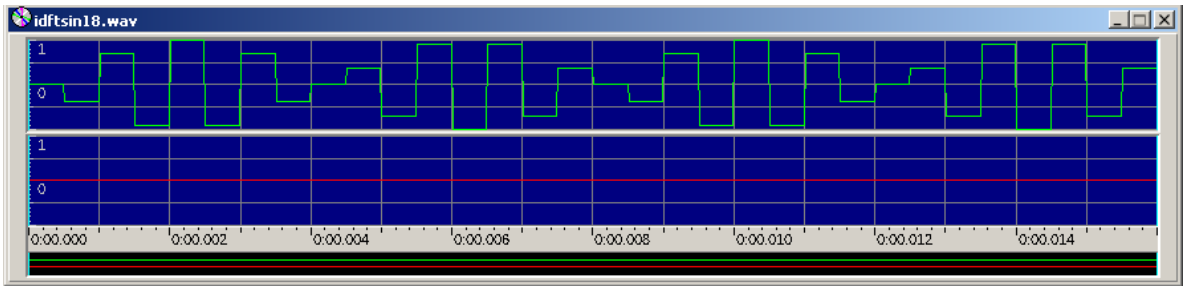

Ilustración 37 Seno de 1125 Hz

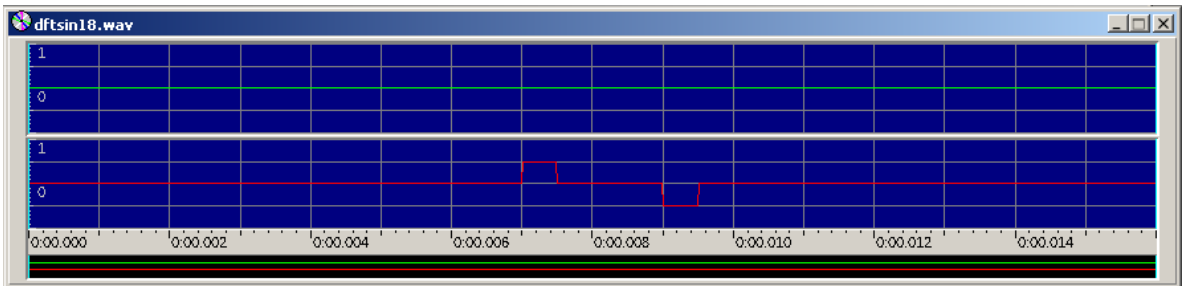

Ilustración 38 DFT del Seno de 1125 Hz

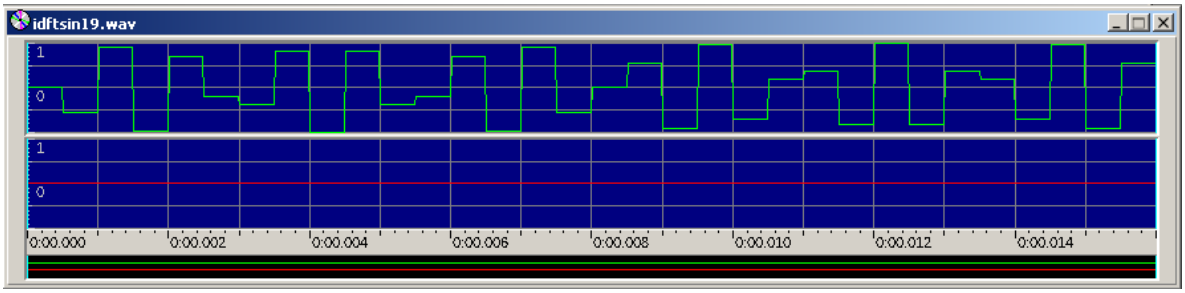

Ilustración 39 Seno de 1187.5 Hz

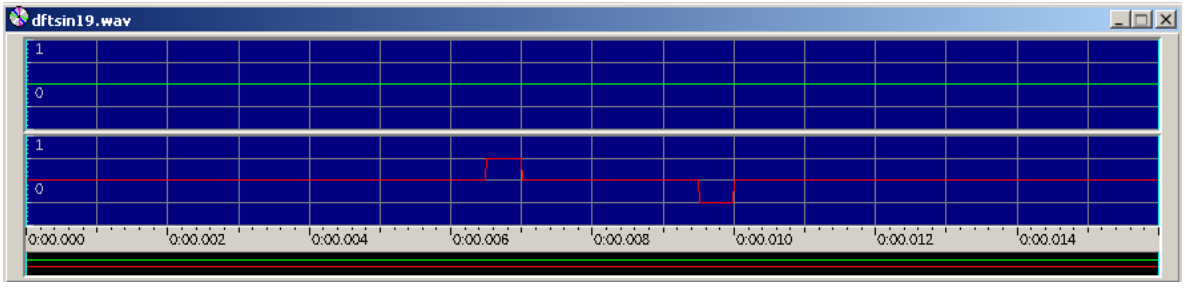

Ilustración 40 DFT Seno de 1187.5 Hz

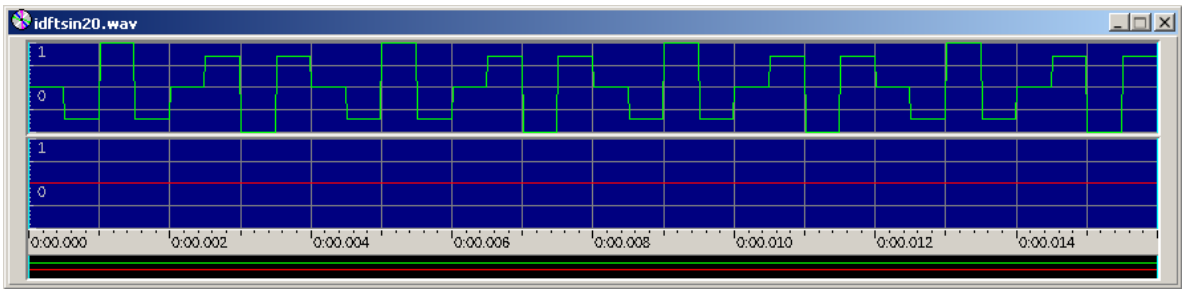

Ilustración 41 Seno de 1250 Hz

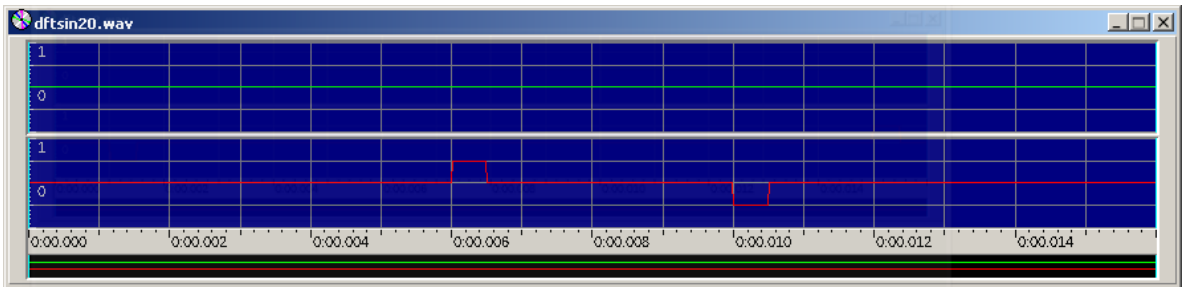

Ilustración 42 DFT del Seno de 1250 Hz

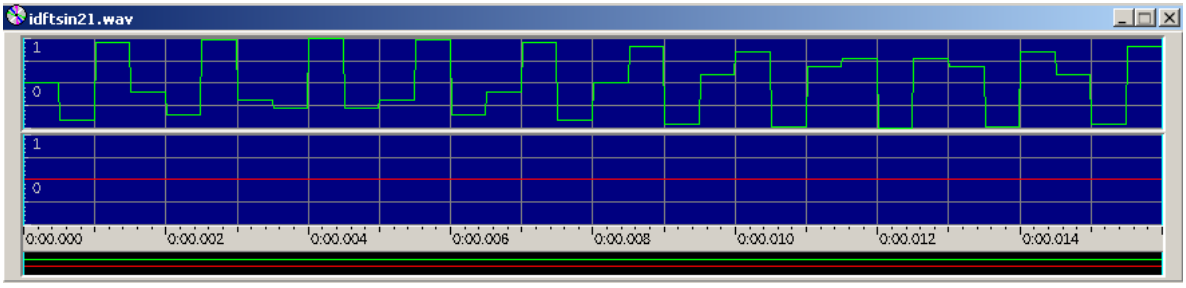

Ilustración 43 Seno de 1312.5 Hz

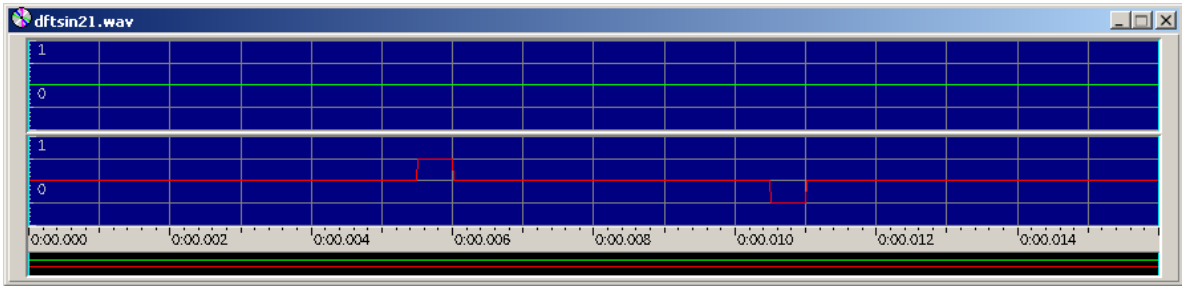

Ilustración 44 DFT del Seno de 1312.5 Hz

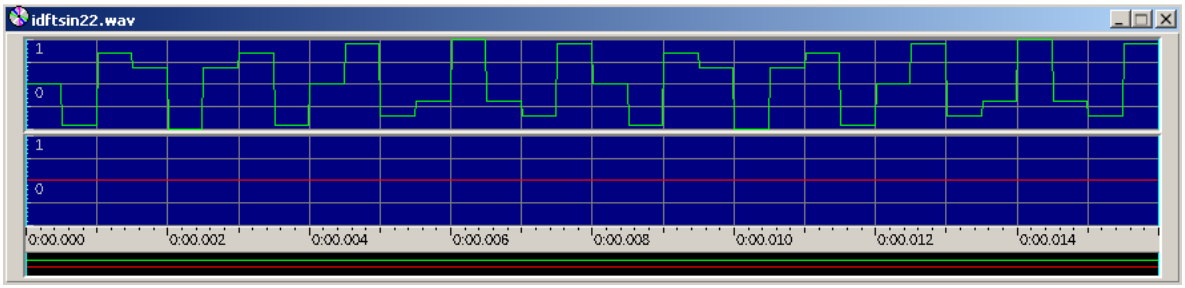

Ilustración 45 Seno de 1375 Hz

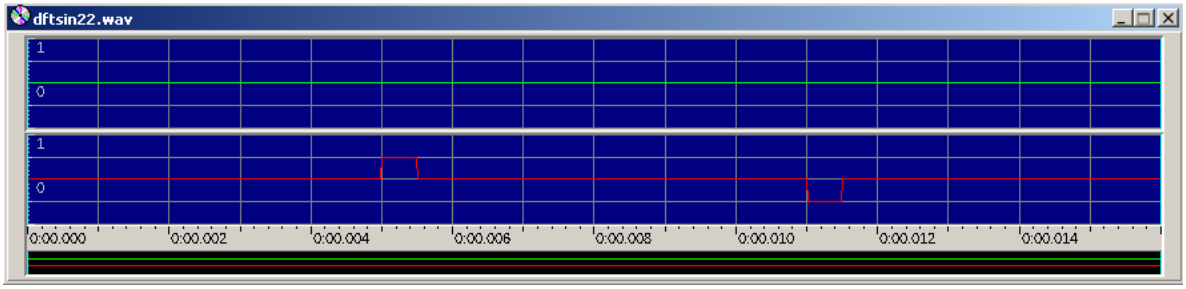

Ilustración 46 DFT del Seno de 1375 Hz

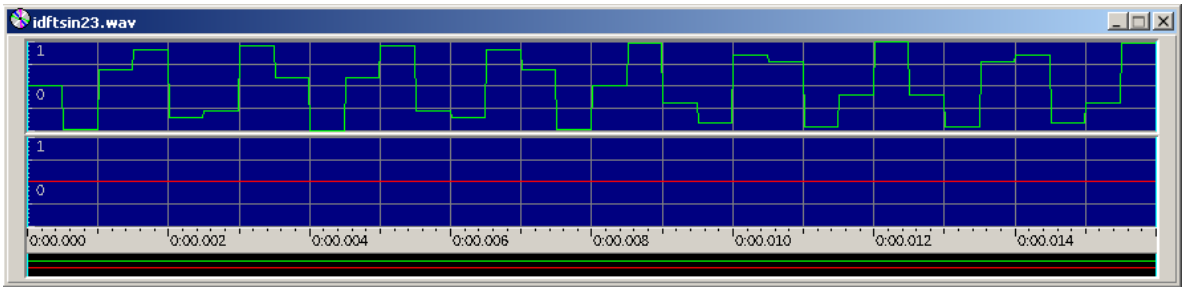

Ilustración 47 Seno de 1437.5 Hz

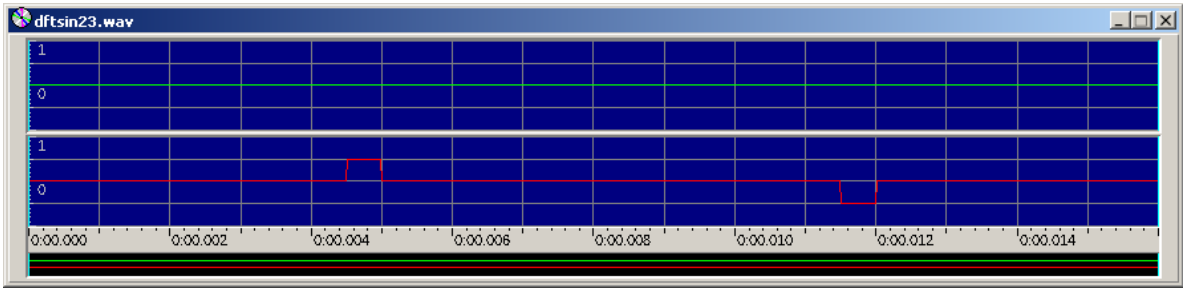

Ilustración 48 DFT del Seno de 1437.5 Hz

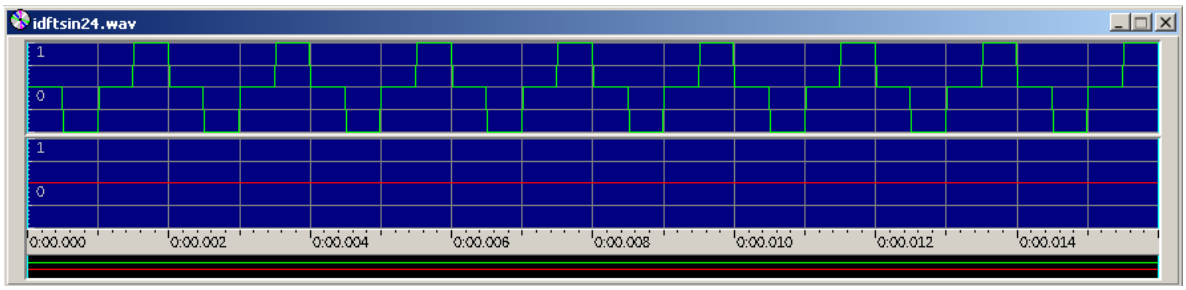

Ilustración 49 Seno de 1500 Hz

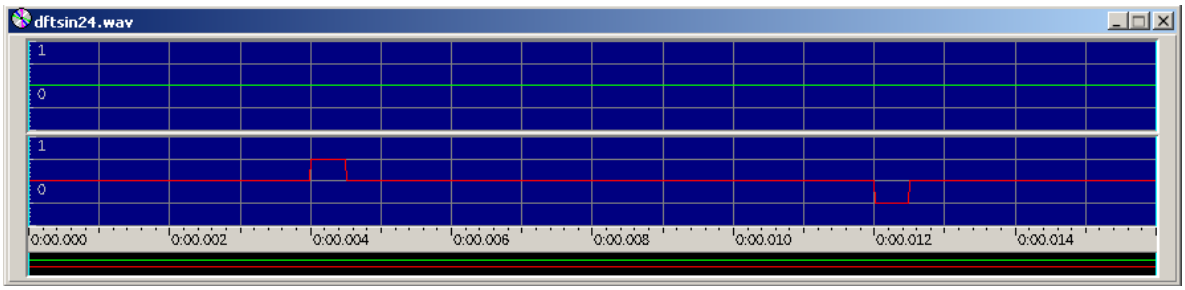

Ilustración 50 DFT del Seno de 1500 Hz

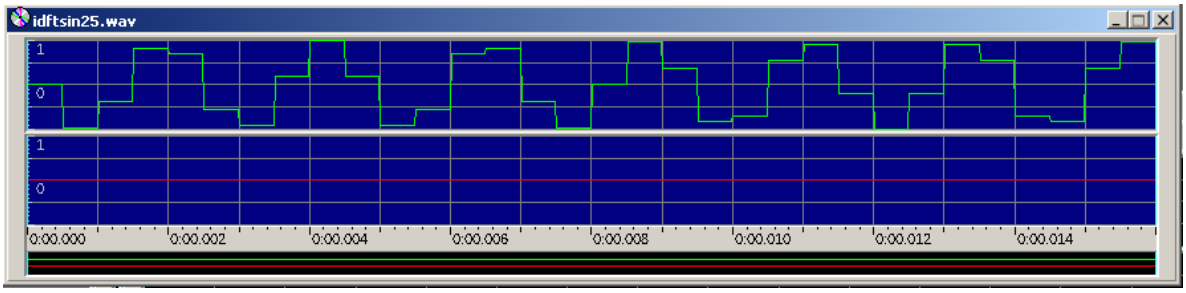

Ilustración 51 Seno de 1562.5 Hz

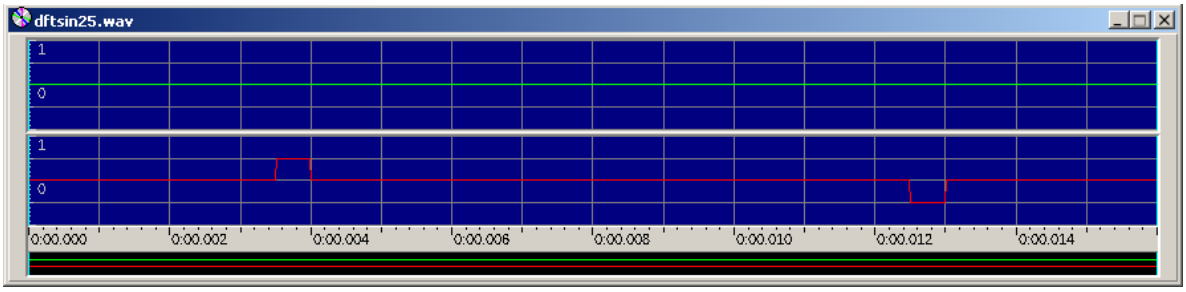

Ilustración 52 DFT del Seno de 1562.5 Hz

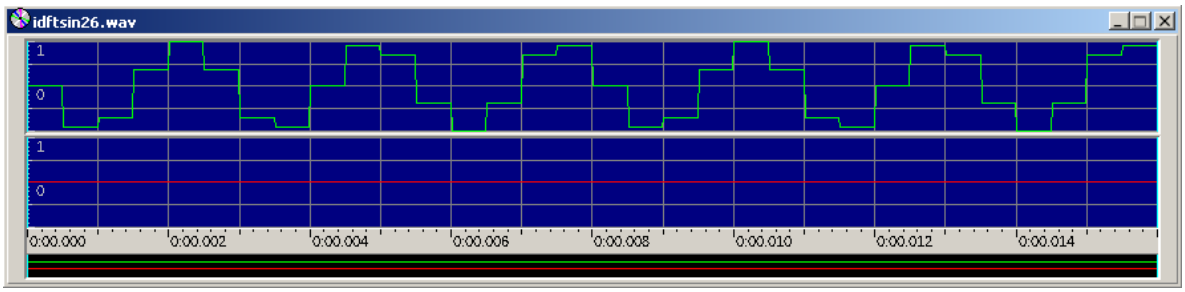

Ilustración 53 Seno de 1625 Hz

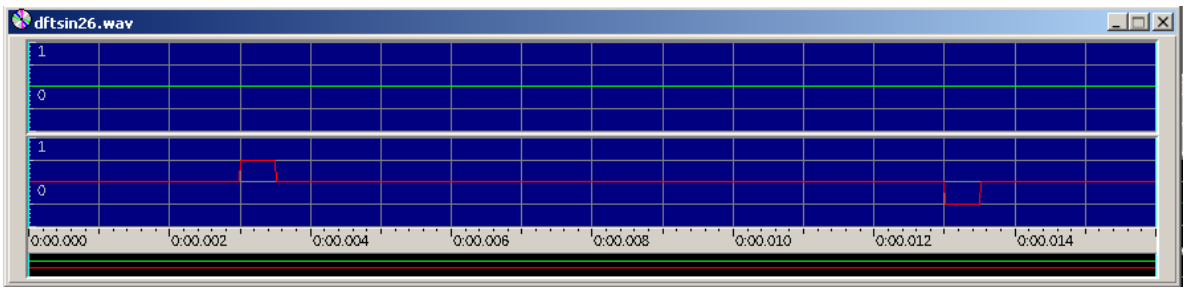

Ilustración 54 DFT del Seno de 1625 Hz

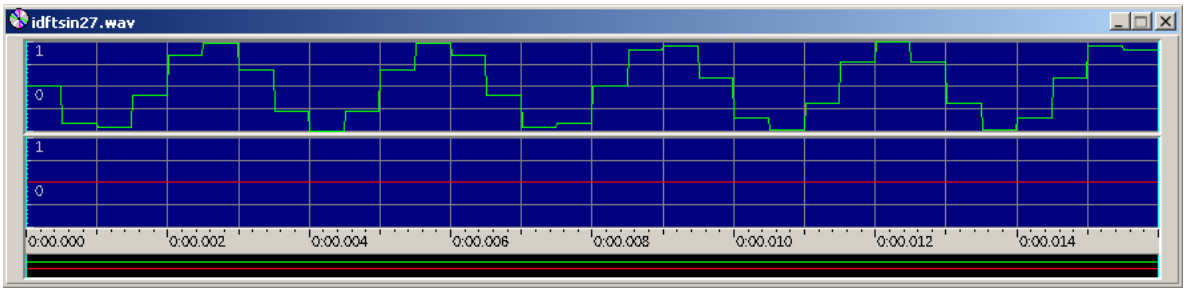

Ilustración 55 Seno de 1687.5 Hz

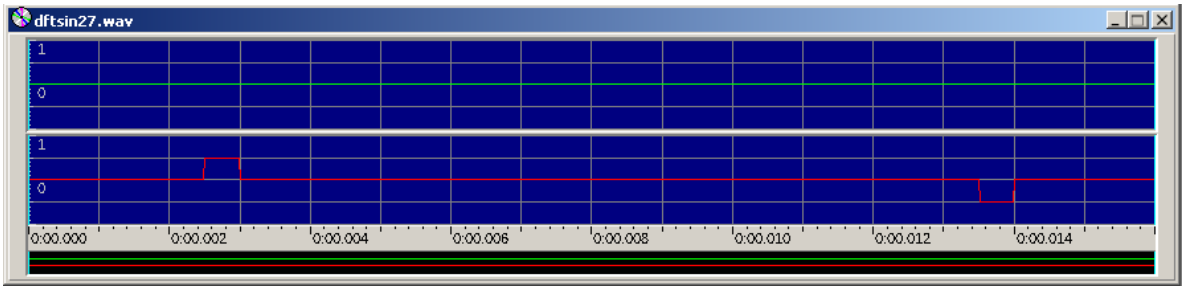

Ilustración 56 DFT del Seno de 1687.5 Hz

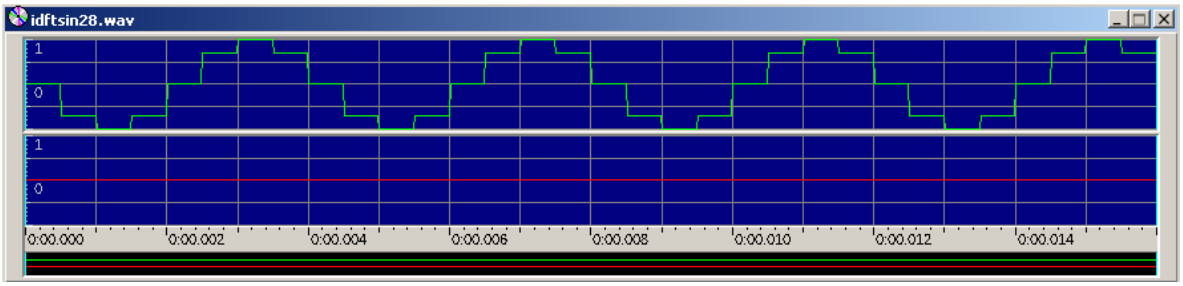

Ilustración 57 Seno de 1750 Hz

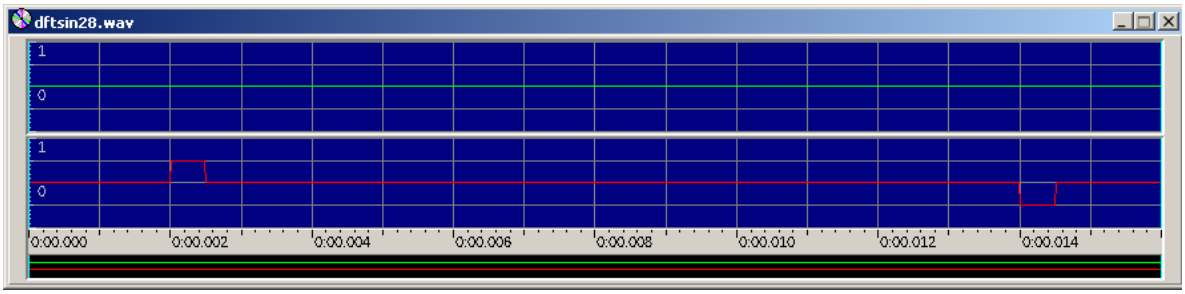

Ilustración 58 DFT del Seno de 1750 Hz

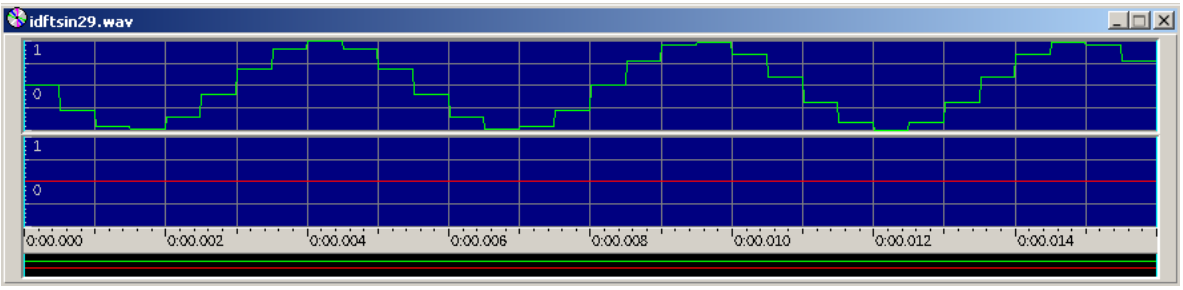

Ilustración 59 Seno de 1812.5 Hz

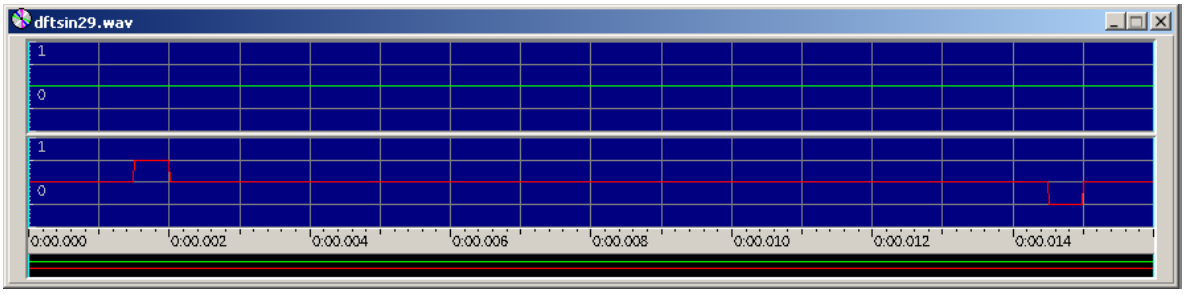

Ilustración 60 DFT Seno de 1812.5 Hz

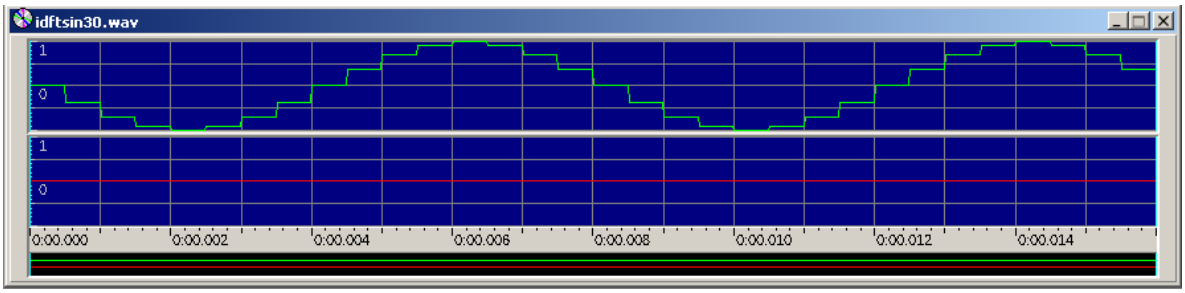

Ilustración 61 Seno de 1875 Hz

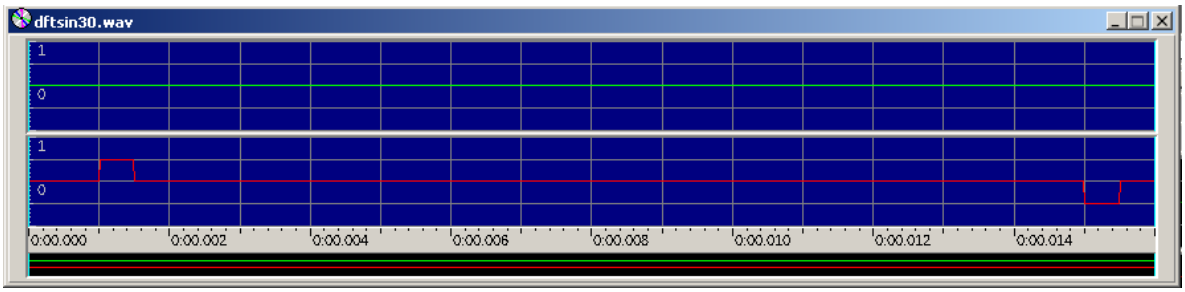

Ilustración 62 DFT del Seno de 1875 Hz

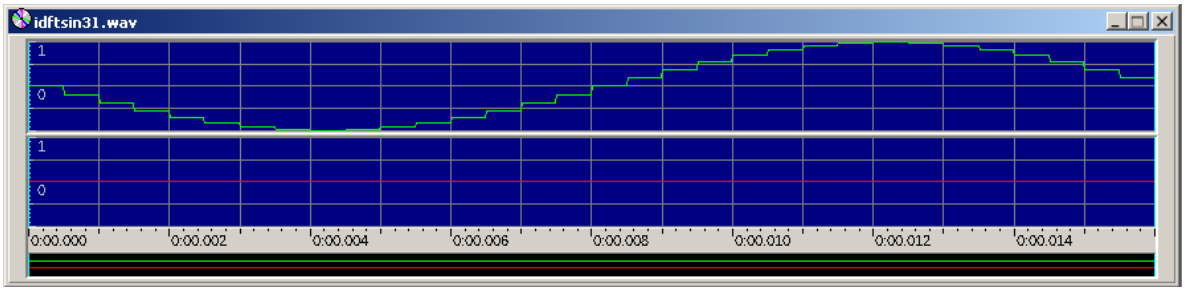

Ilustración 63 Seno de 1937.5 Hz

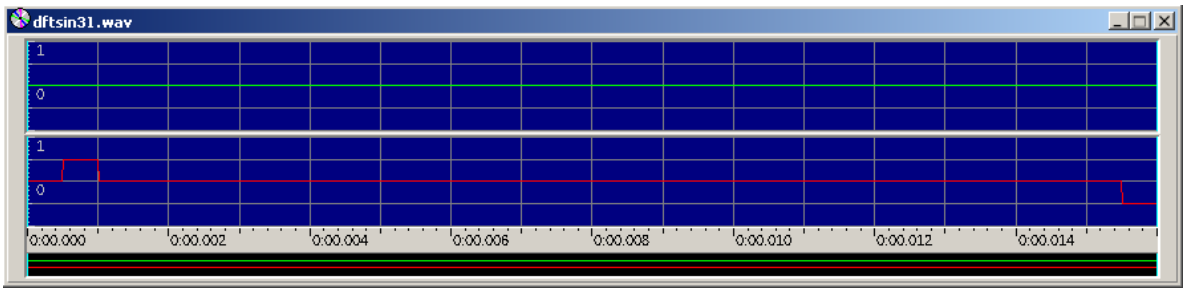

Ilustración 64 DFT del Seno de 1937.5 Hz

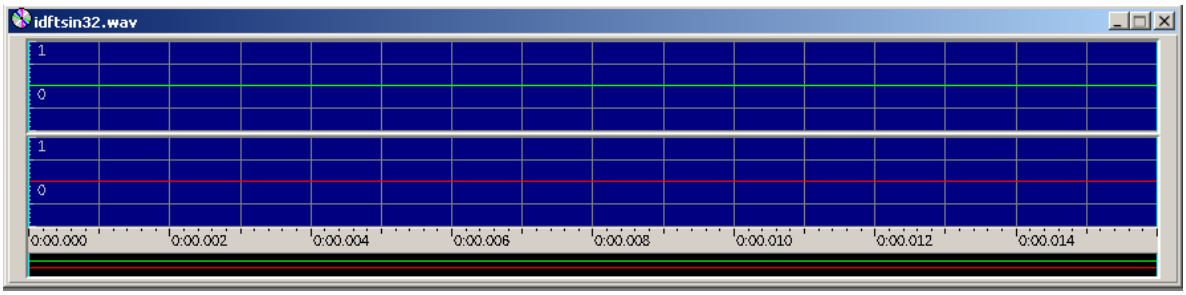

Ilustración 65 Seno de 2000 Hz

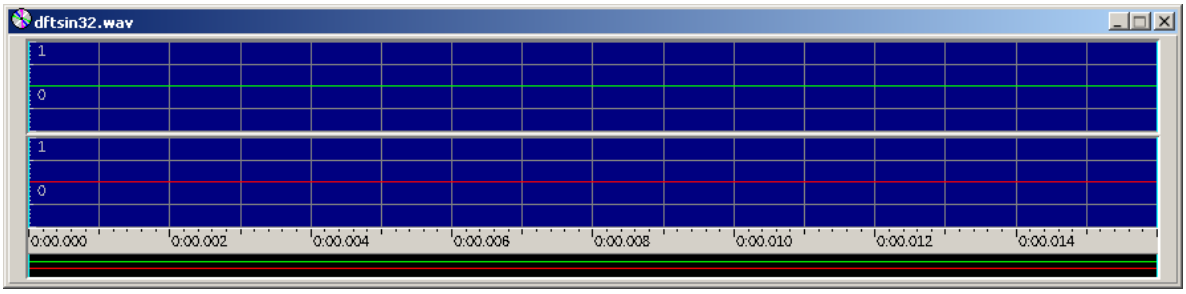

Ilustración 66 DFT del Seno de 2000 Hz

Cosenos

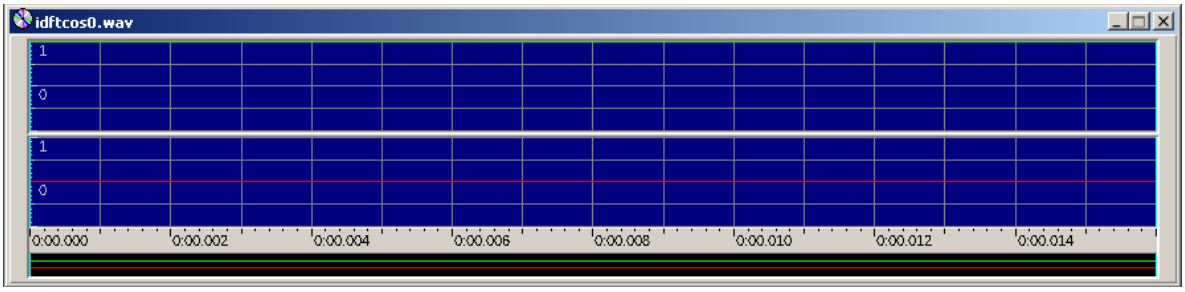

Ilustración 67 Coseno de 0 Hz

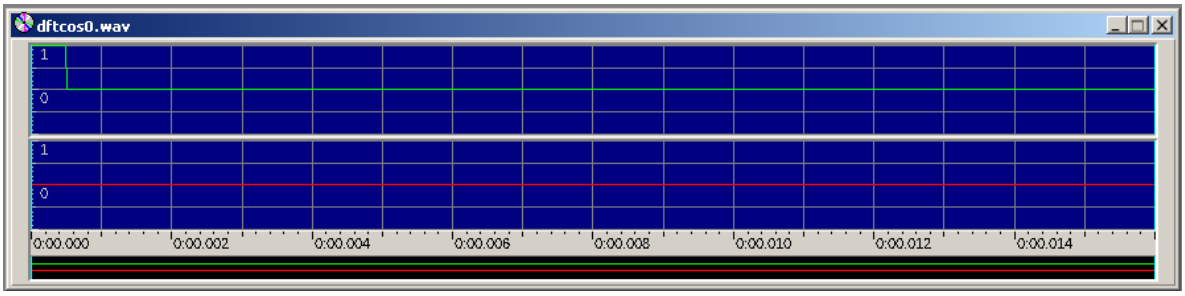

Ilustración 68 DFT del Coseno de 0 Hz

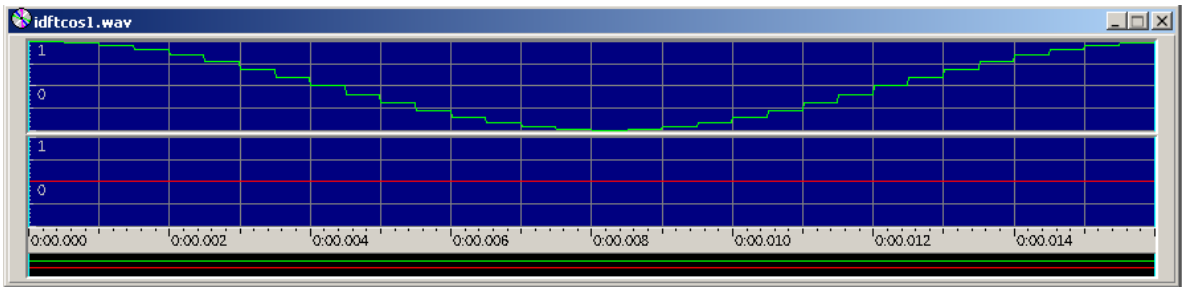

Ilustración 69 Seno de 62.5 Hz

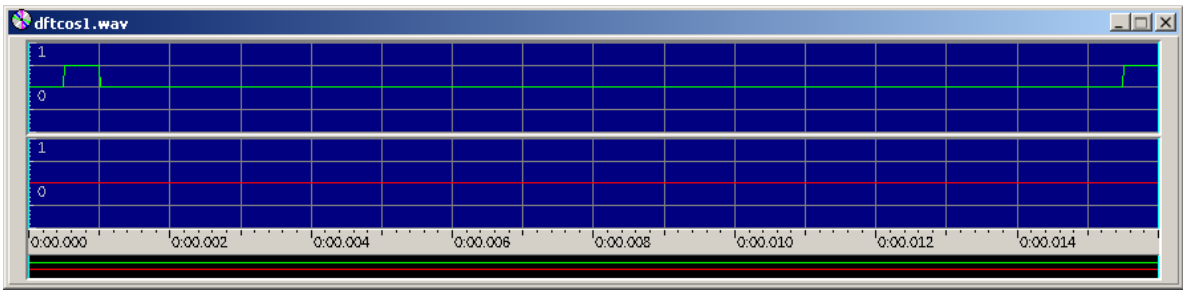

Ilustración 70 DFT del Seno de 62.5 Hz

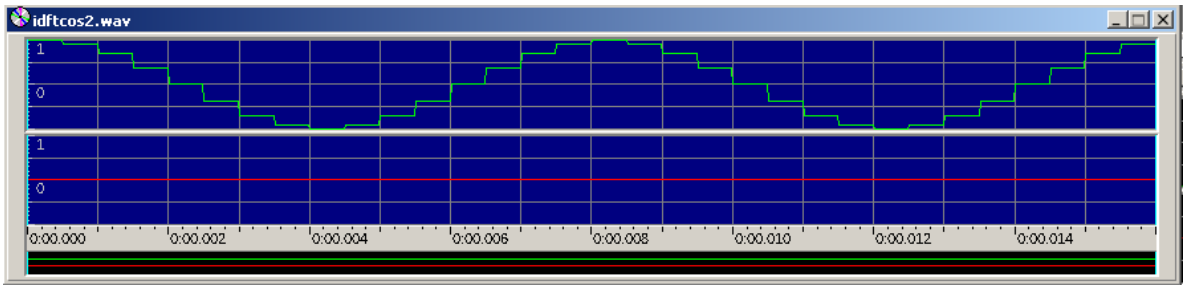

Ilustración 71 Coseno de 125 Hz

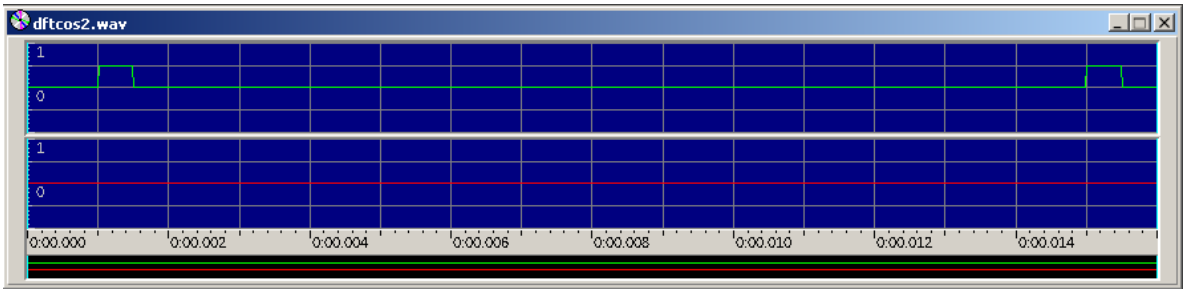

Ilustración 72 DFT del Coseno de 125 Hz

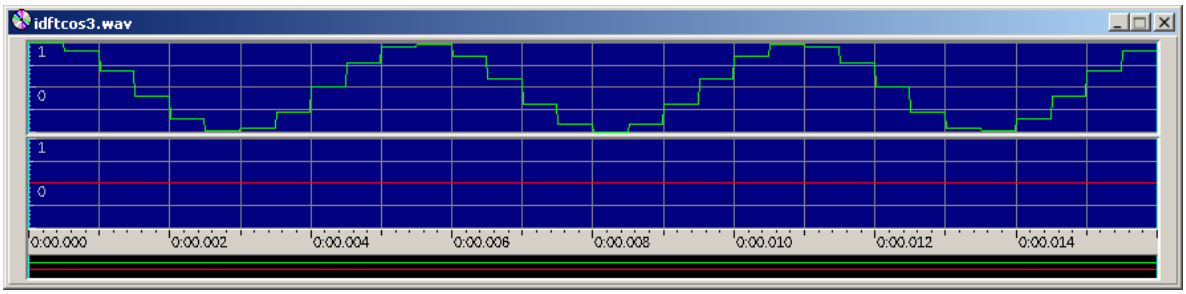

Ilustración 73 Coseno de 187.5 Hz

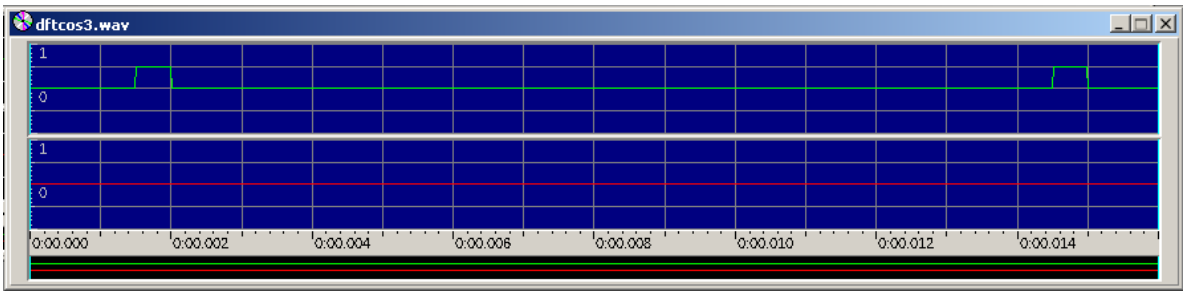

Ilustración 74 DFT del Coseno de 187.5 Hz

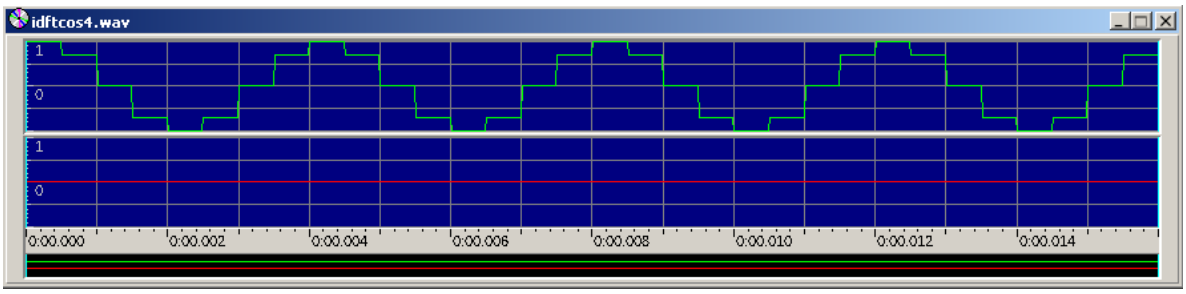

Ilustración 75 Coseno de 250 Hz

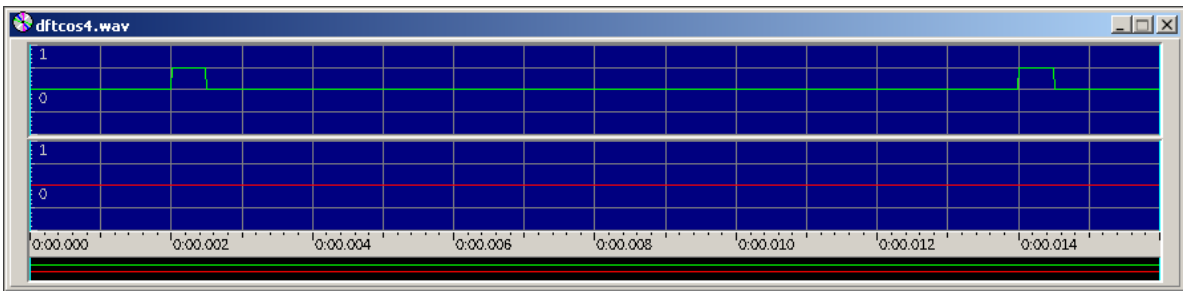

Ilustración 76 DFT del Coseno de 250 Hz

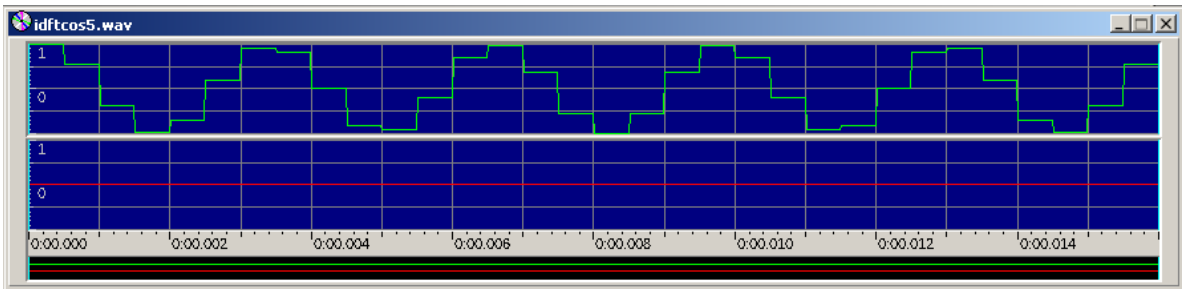

Ilustración 77 Coseno de 312.5 Hz

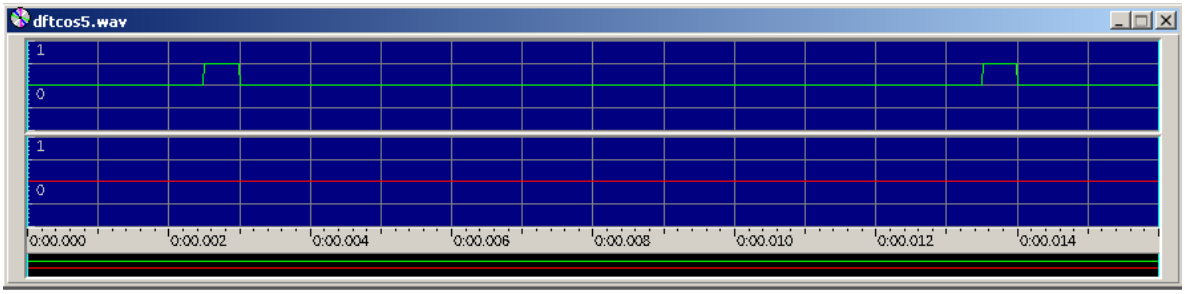

Ilustración 78 DFT del Coseno de 312.5 Hz

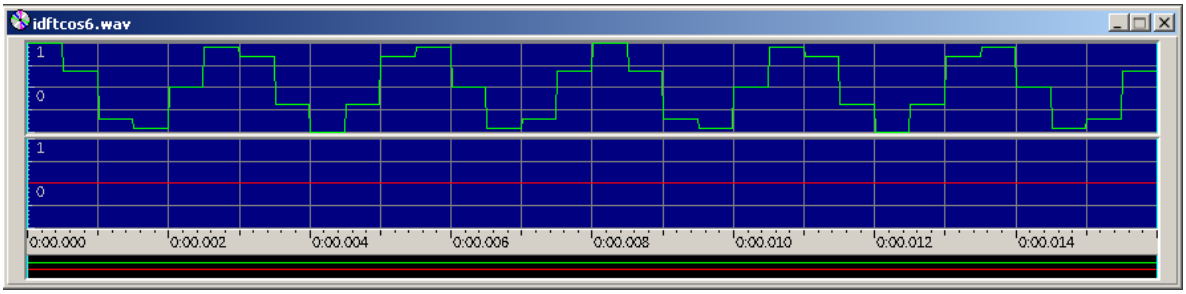

Ilustración 79 Coseno de 375 Hz

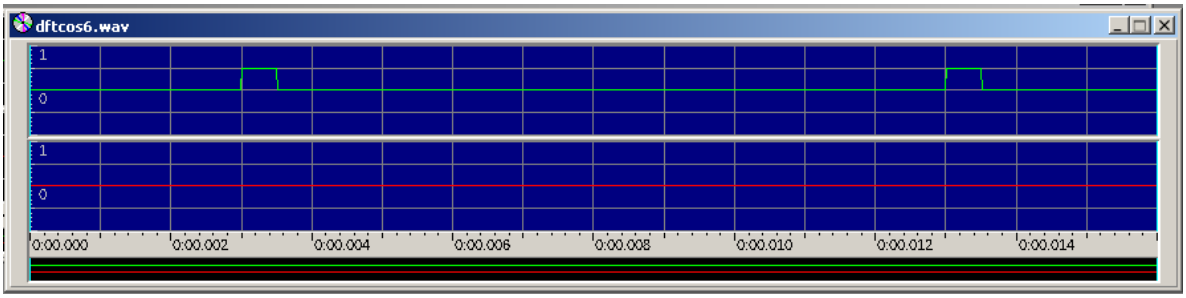

Ilustración 80 DFT Coseno de 375 Hz

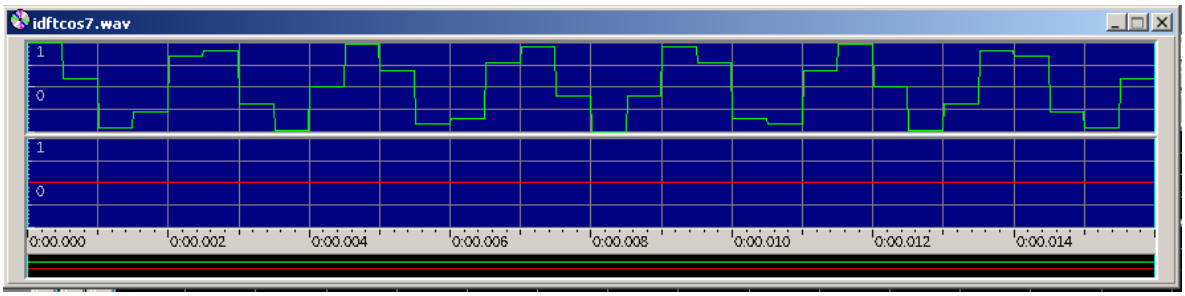

Ilustración 81 Coseno de 437.5 Hz

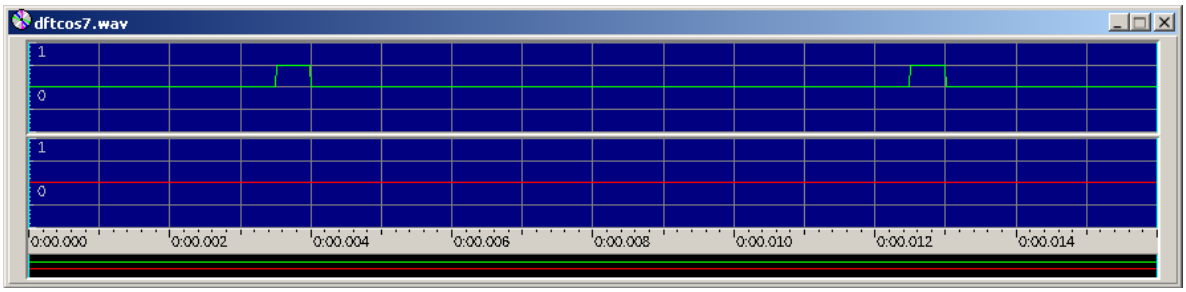

Ilustración 82 DFT del Coseno de 437.5 Hz

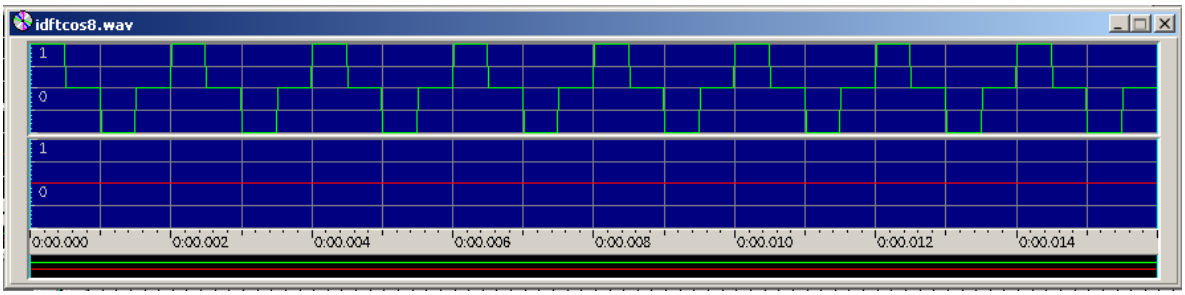

Ilustración 83 Coseno de 500 Hz

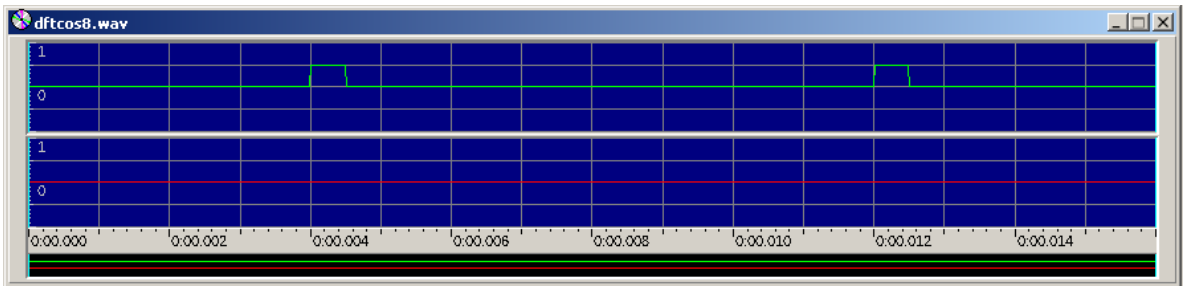

Ilustración 84 DFT del Coseno de 500 Hz

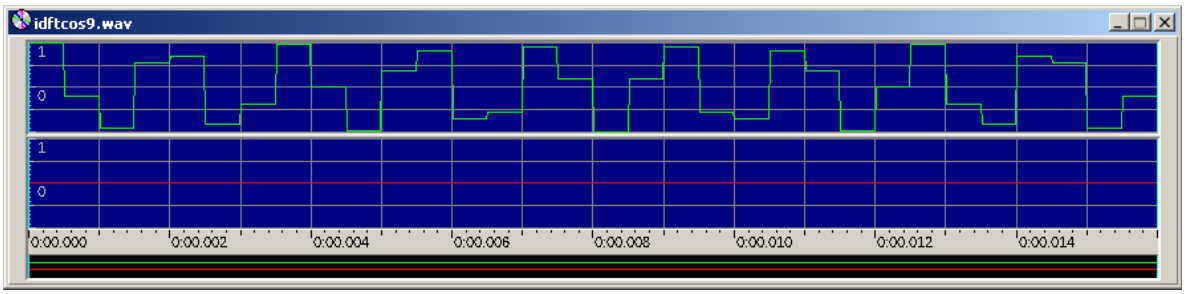

Ilustración 85 Coseno de 562.5 Hz

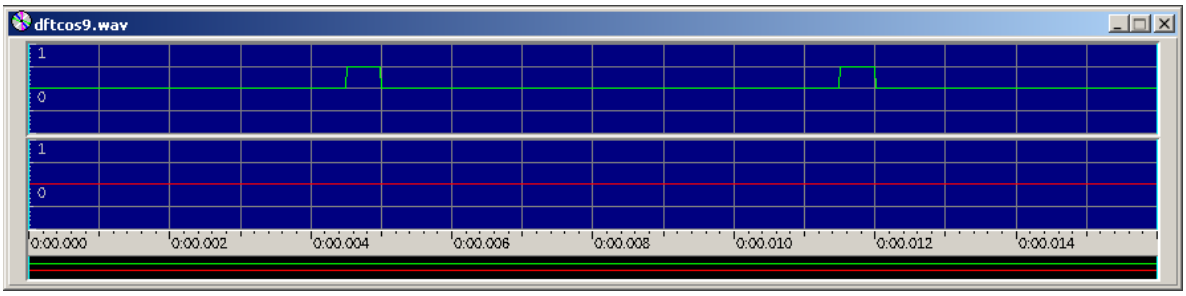

Ilustración 86 DFT del Coseno de 562.5 Hz

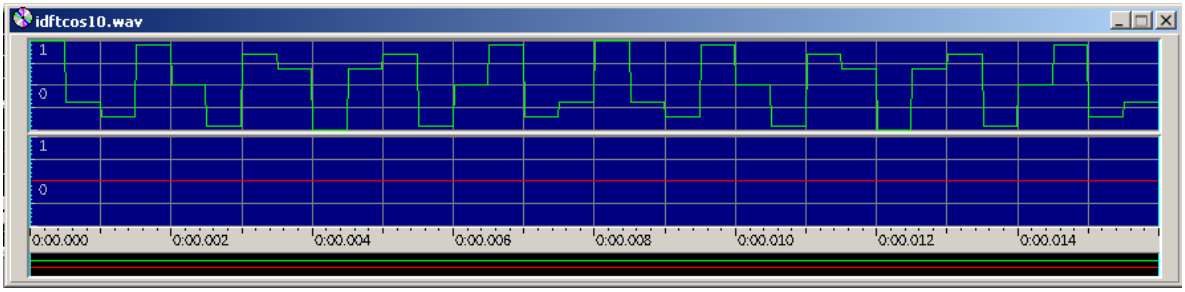

Ilustración 87 Coseno de 650 Hz

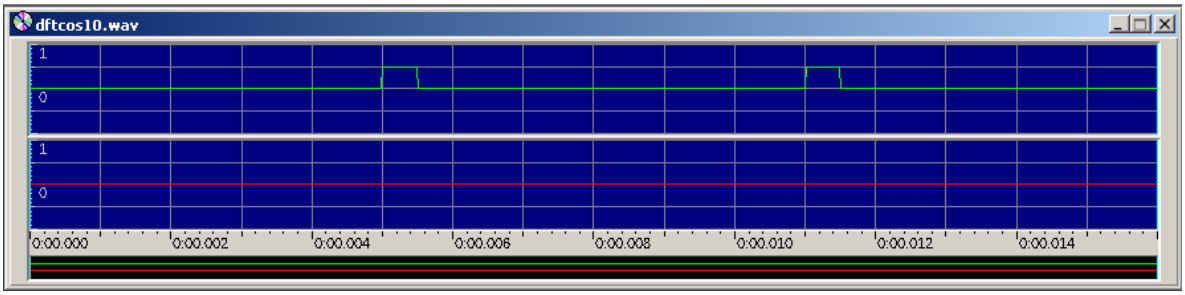

Ilustración 88 DFT del Coseno de 650 Hz

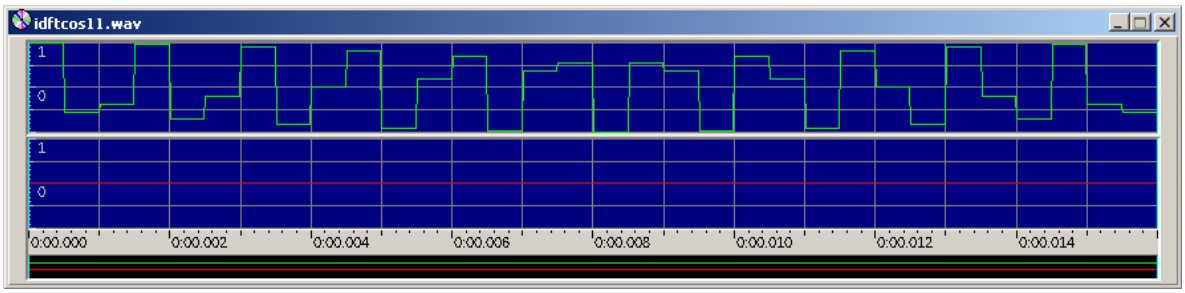

Ilustración 89 Coseno de 687.5 Hz

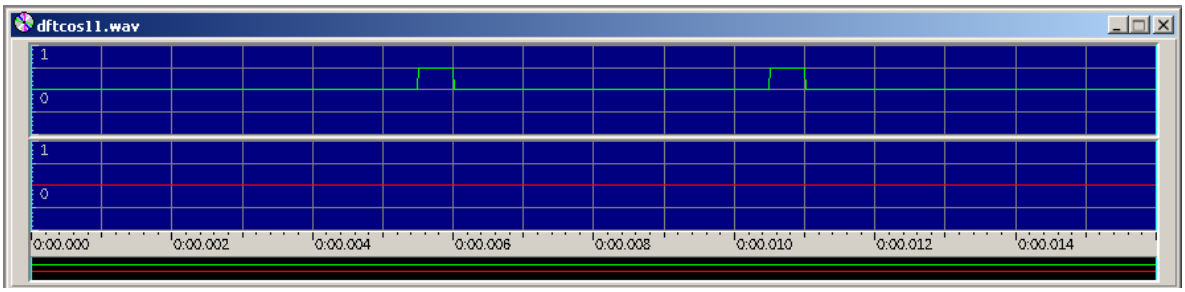

Ilustración 90 DFT del Coseno de 687.5 Hz

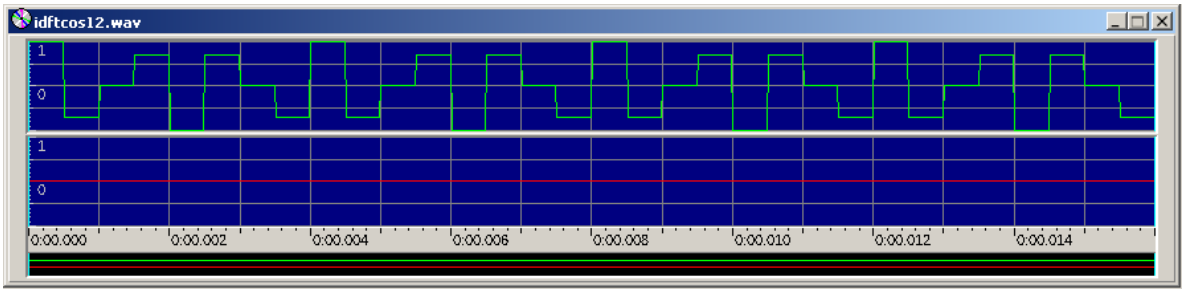

Ilustración 91 Coseno de 750 Hz

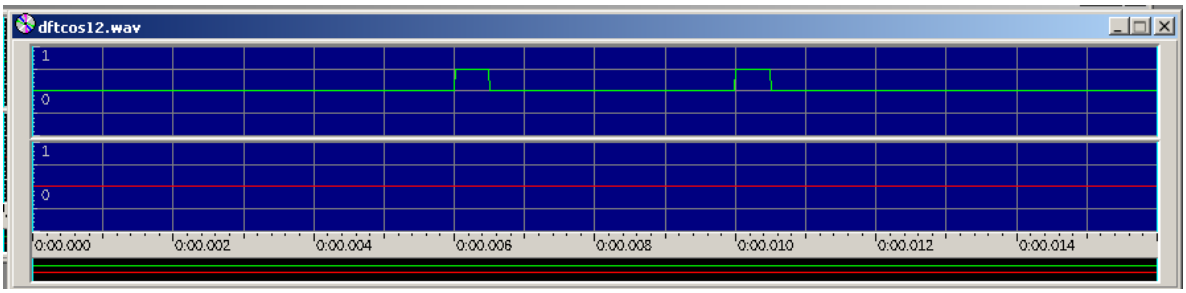

Ilustración 92 DFT del Coseno de 750 Hz

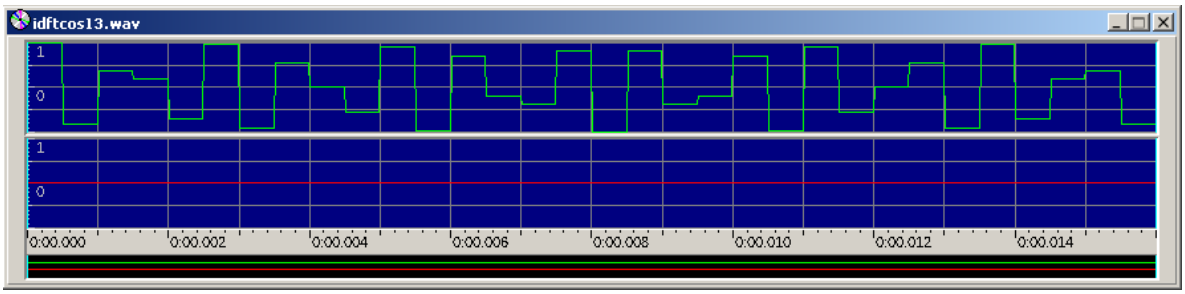

Ilustración 93 Coseno de 812.5 Hz

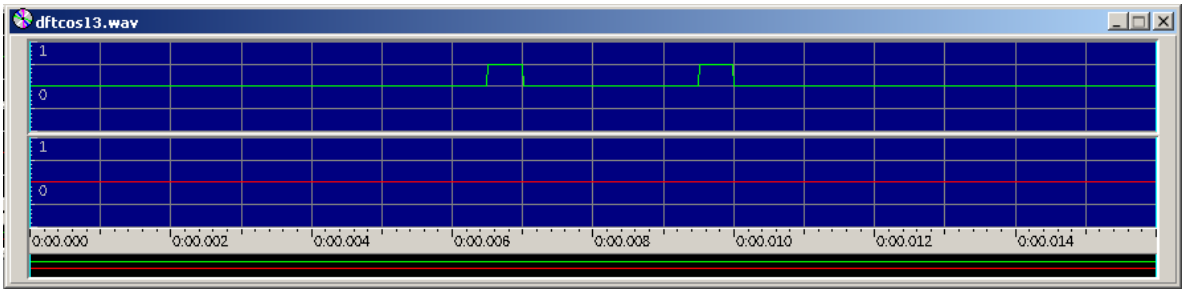

Ilustración 94 DFT del Coseno de 812.5 Hz

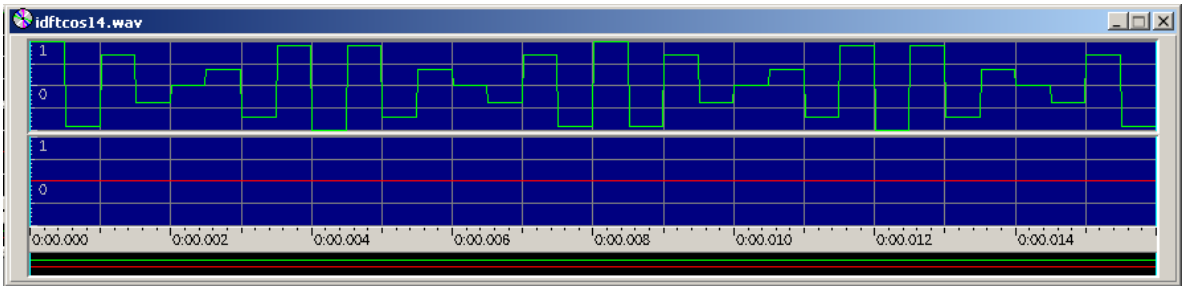

Ilustración 95 Coseno de 875 Hz

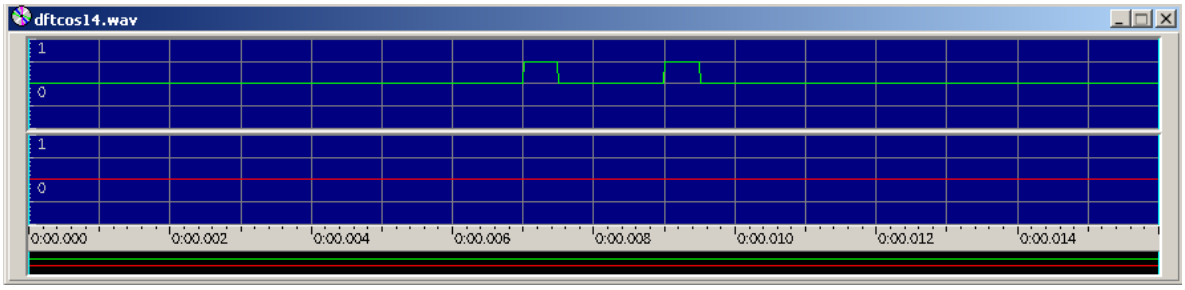

Ilustración 96 DFT del Coseno de 875 Hz

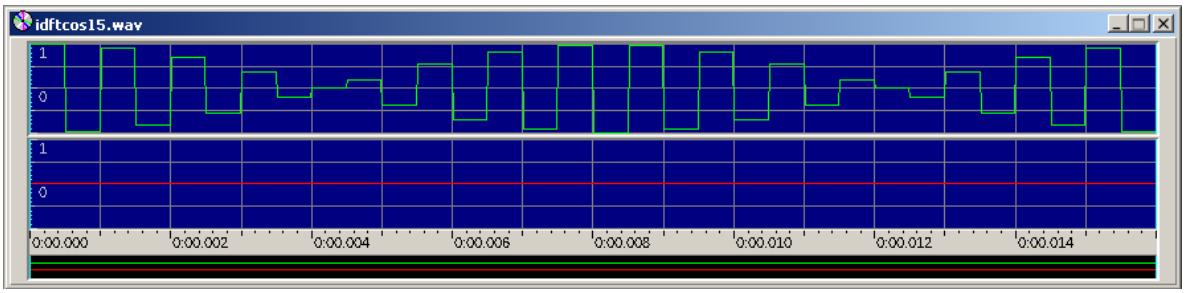

Ilustración 97 Coseno de 937.5

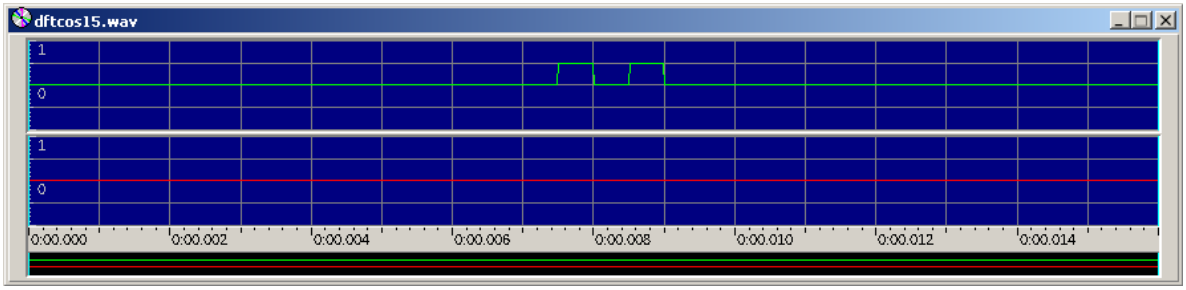

Ilustración 98 DFT del Coseno de 937.5

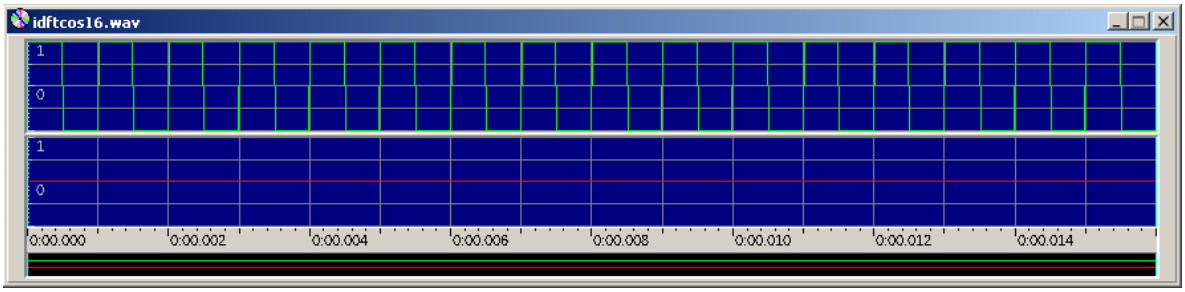

Ilustración 99 Coseno de 1000 Hz

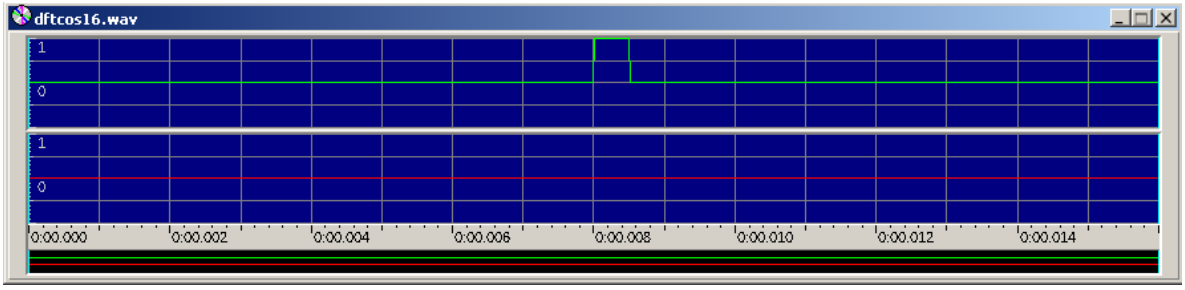

Ilustración 100 DFT del Coseno de 1000 Hz

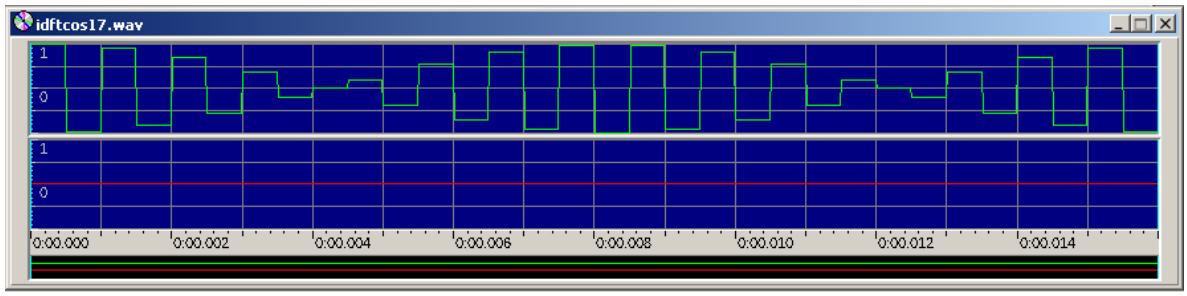

Ilustración 101 Coseno de 1062.5 Hz

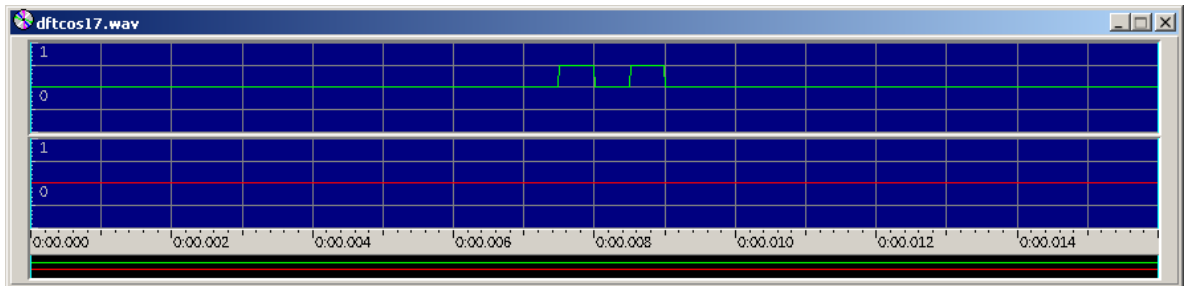

Ilustración 102 DFT del Coseno de 1062.5 Hz

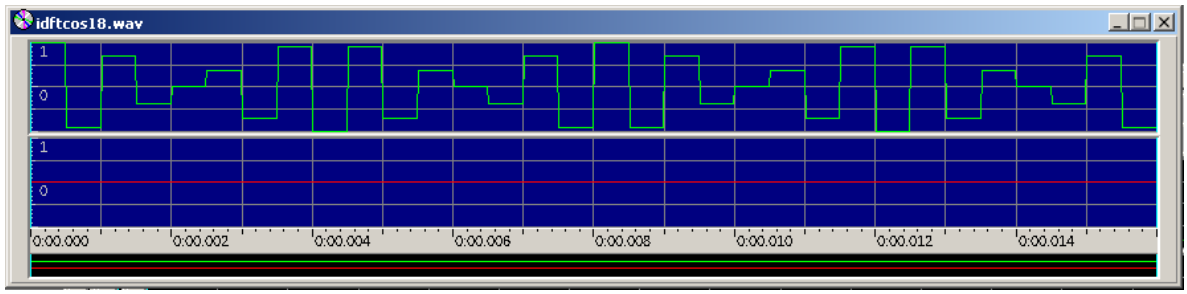

Ilustración 103 Coseno de 1125 Hz

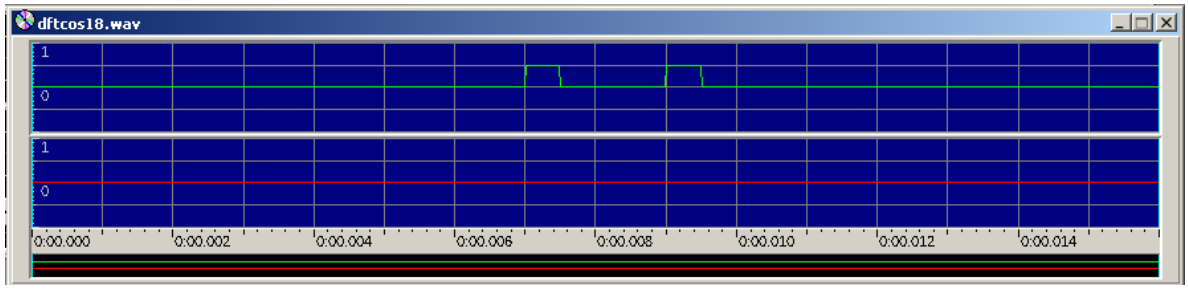

Ilustración 104 DFT del Coseno de 1125 Hz

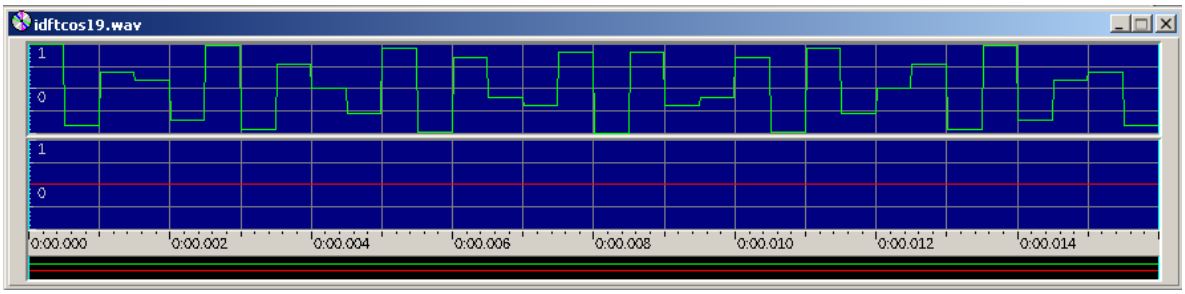

Ilustración 105 Coseno de 1187.5 Hz

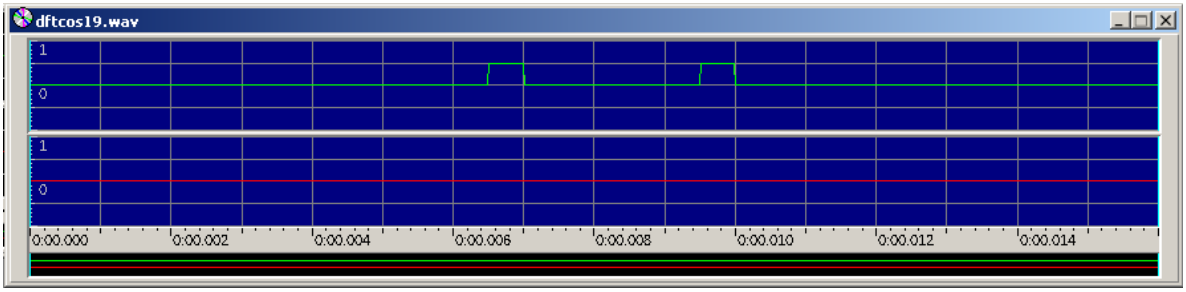

Ilustración 106 DFT del Coseno de 1187.5 Hz

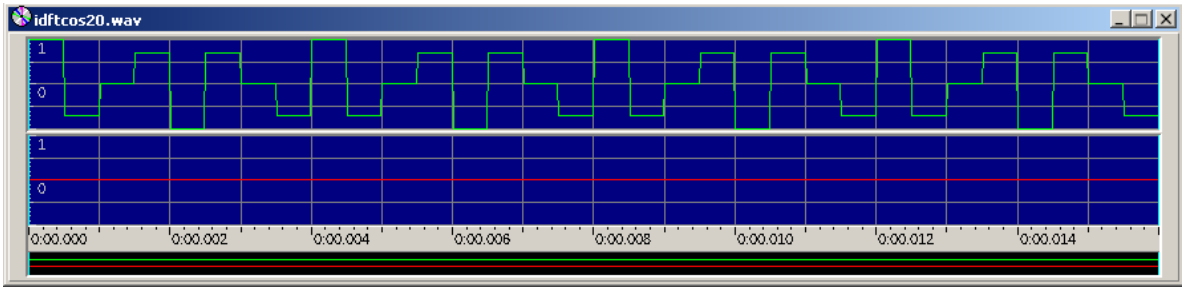

Ilustración 107 Coseno de 1250 Hz

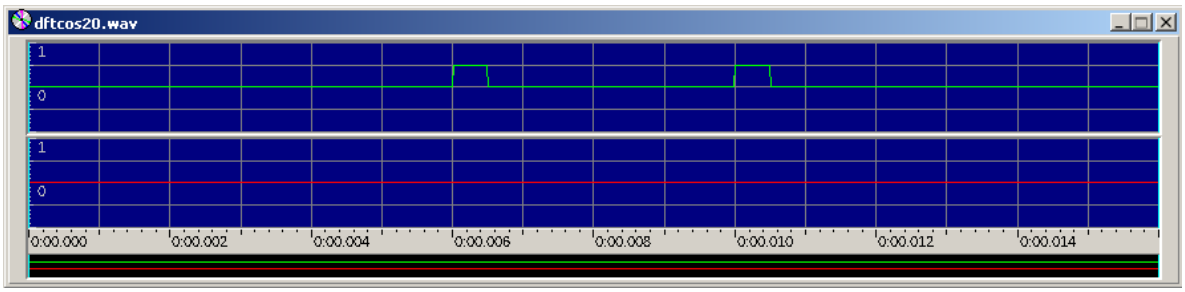

Ilustración 108 DFT del Coseno de 1250 Hz

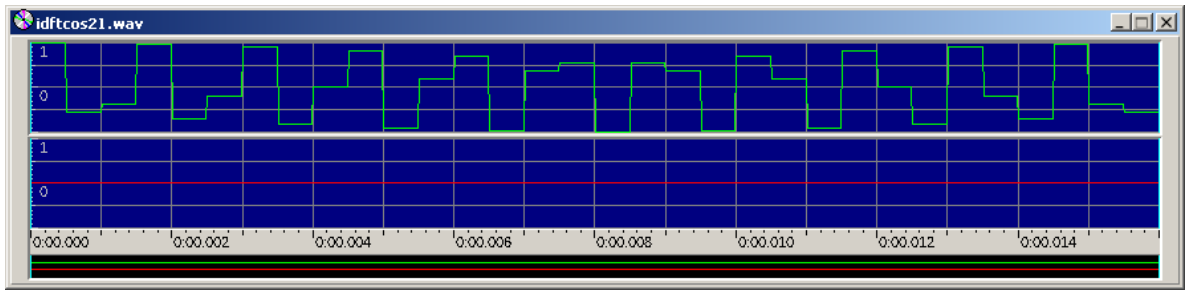

Ilustración 109 Coseno de 1312.5 Hz

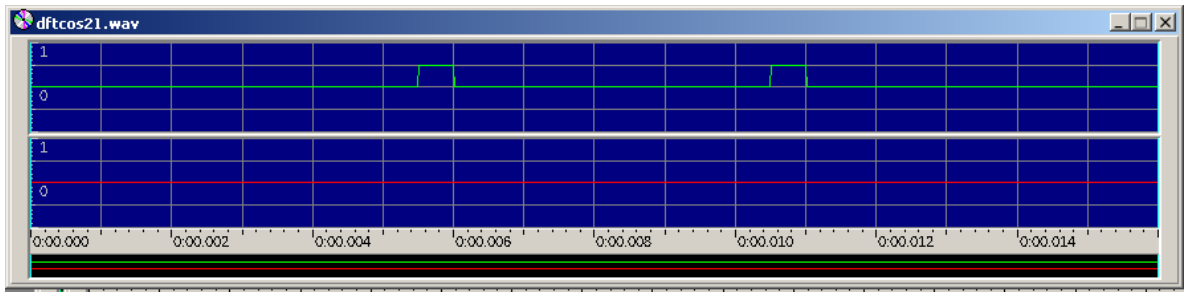

Ilustración 110 DFT del Coseno de 1312.5 Hz

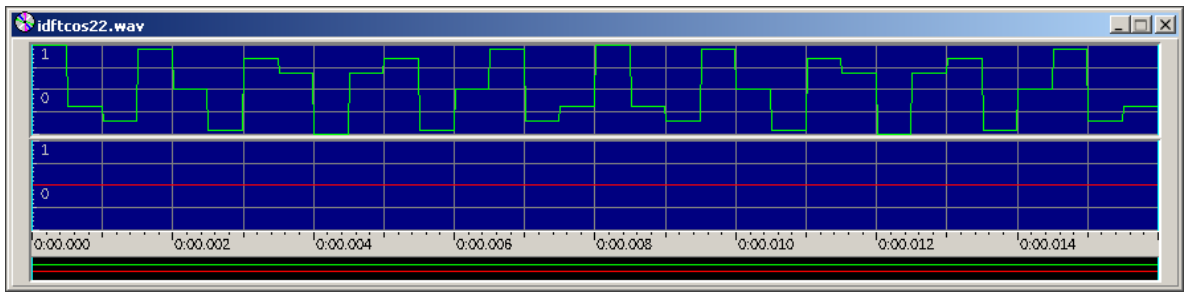

Ilustración 111 Coseno de 1375 Hz

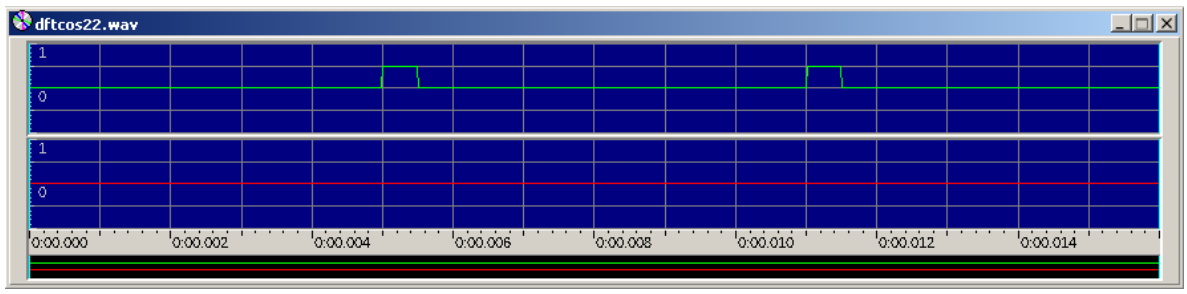

Ilustración 112 DFT del Coseno de 1375 Hz

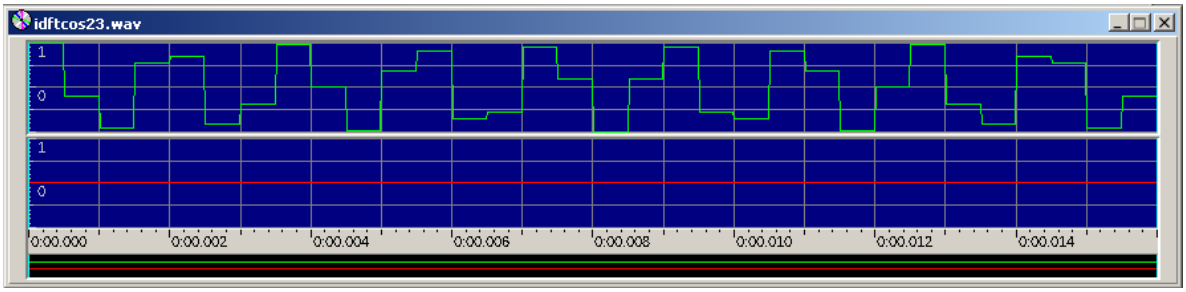

Ilustración 113 Coseno de 1437.5 Hz

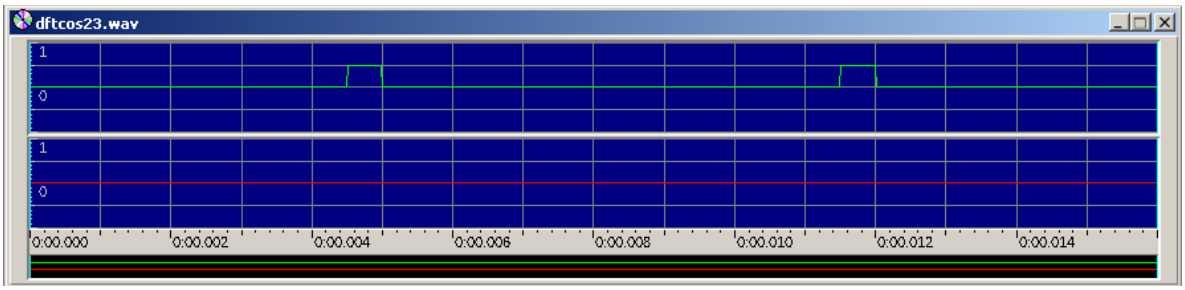

Ilustración 114 DFT del Coseno de 1437.5 Hz

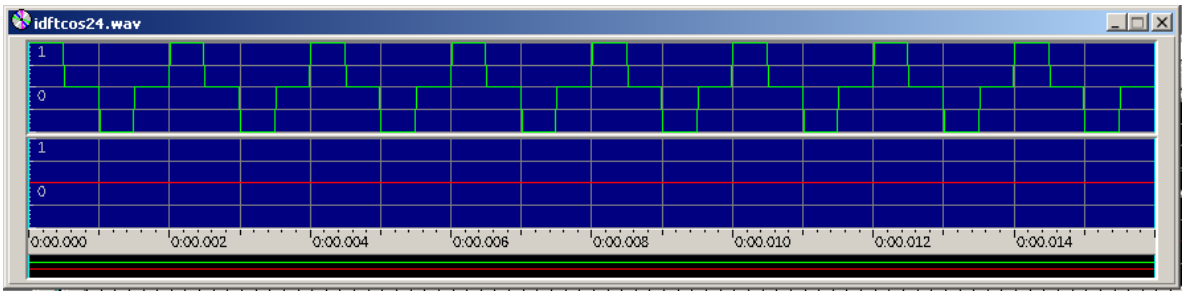

Ilustración 115 Coseno de 1500 Hz

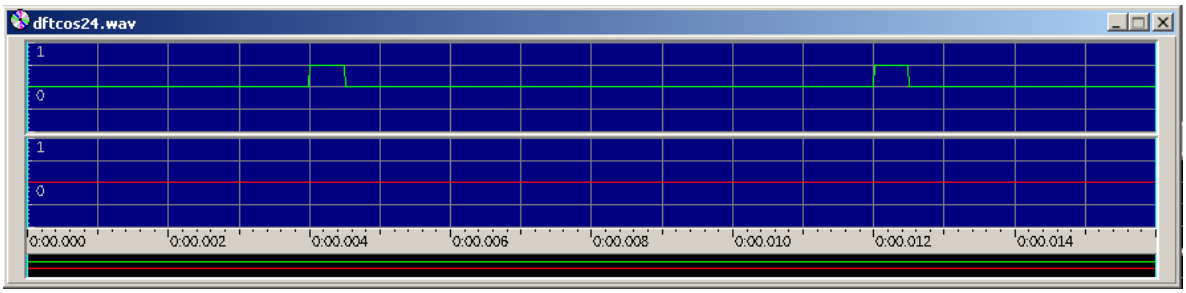

Ilustración 116 DFT del Coseno de 1500 Hz

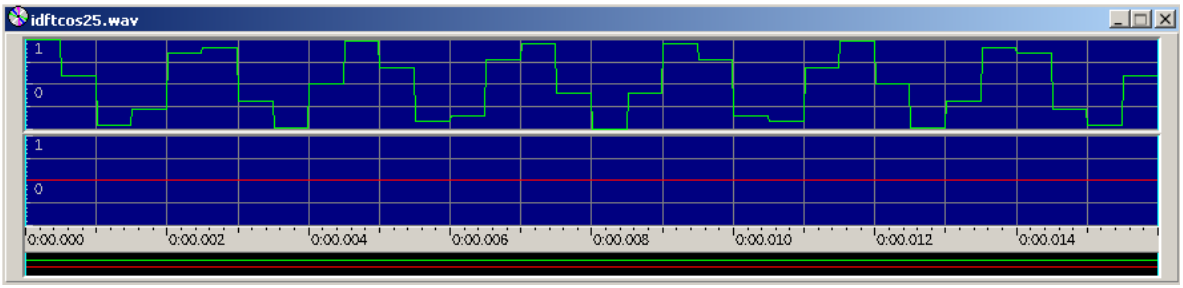

Ilustración 117 Coseno de 1562.5 Hz

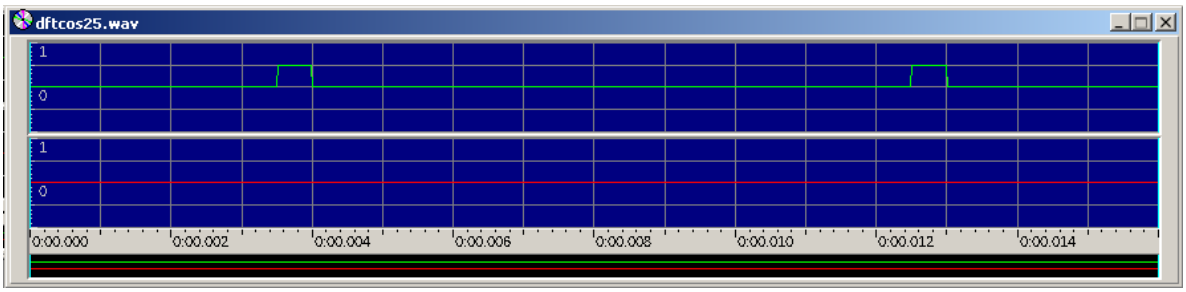

Ilustración 118 DFT del Coseno de 1562.5 Hz

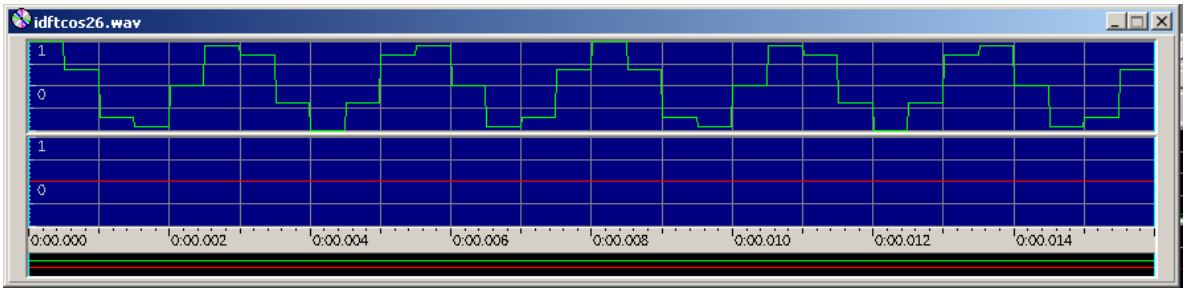

Ilustración 119 Coseno 1625 Hz

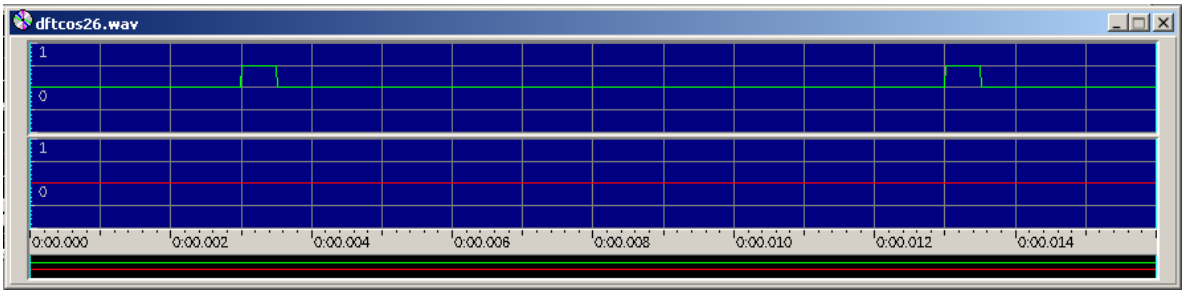

Ilustración 120 DFT del Coseno de 1625 Hz

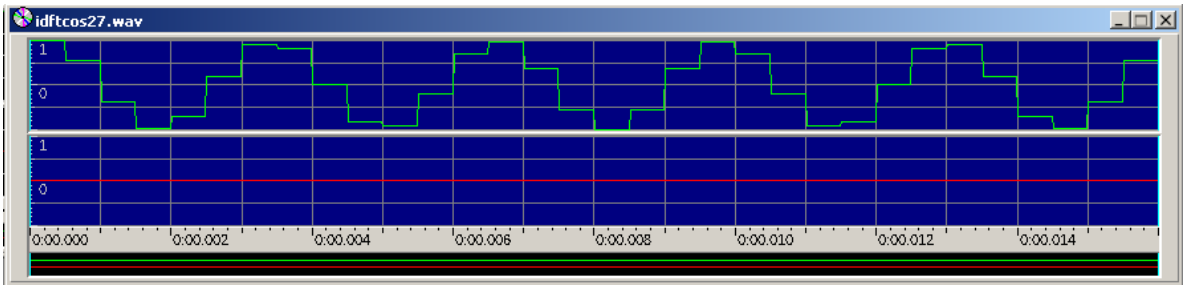

Ilustración 121 Coseno de 1687.5 Hz

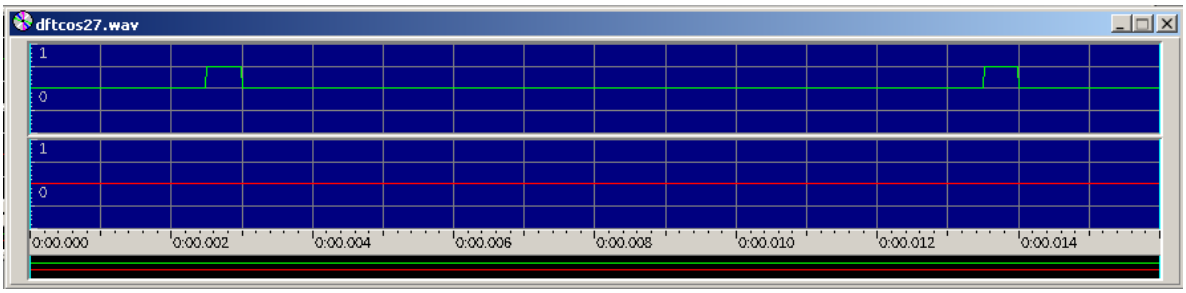

Ilustración 122 DFT del Coseno de 1687.5 Hz

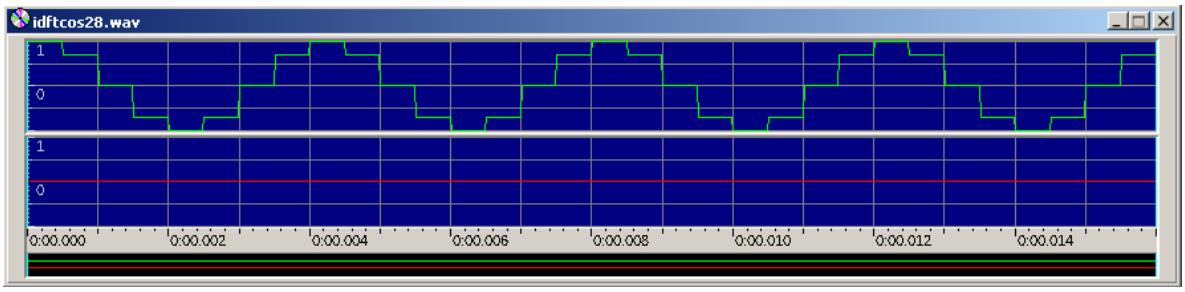

Ilustración 123 Coseno de 1750 Hz

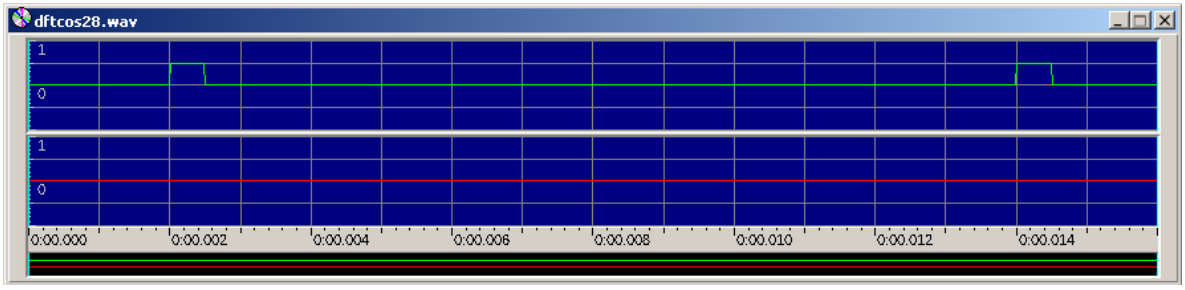

Ilustración 124 DFT del Coseno de 1750 Hz

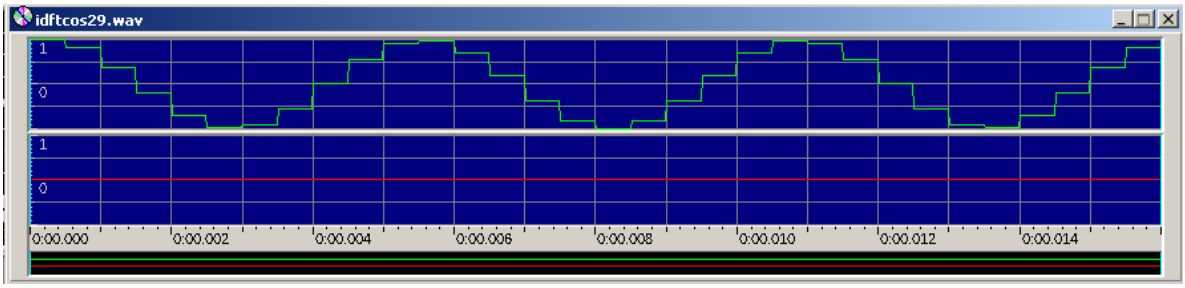

Ilustración 125 Coseno de 1812.5 Hz

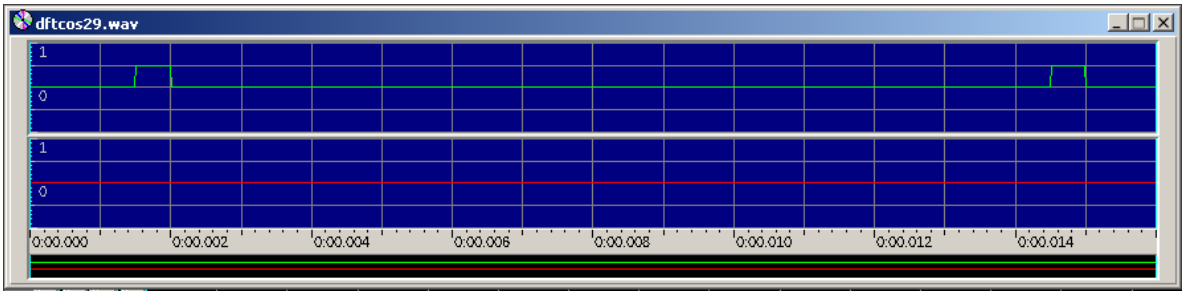

Ilustración 126 DFT del Coseno de 1812.5 Hz

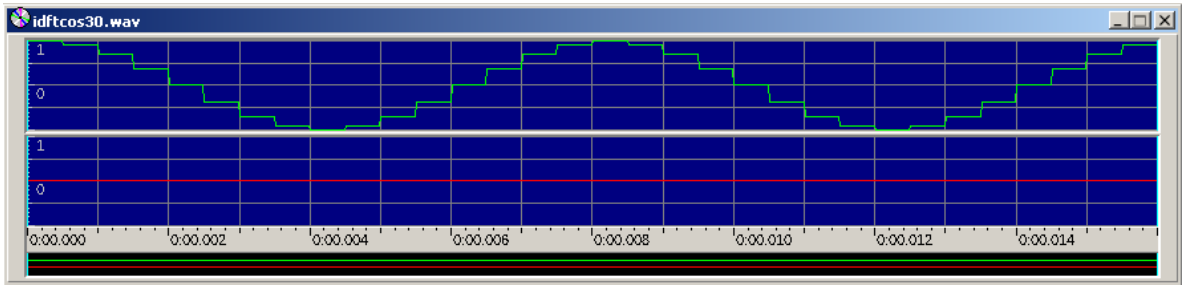

Ilustración 127 Coseno de 1875 Hz

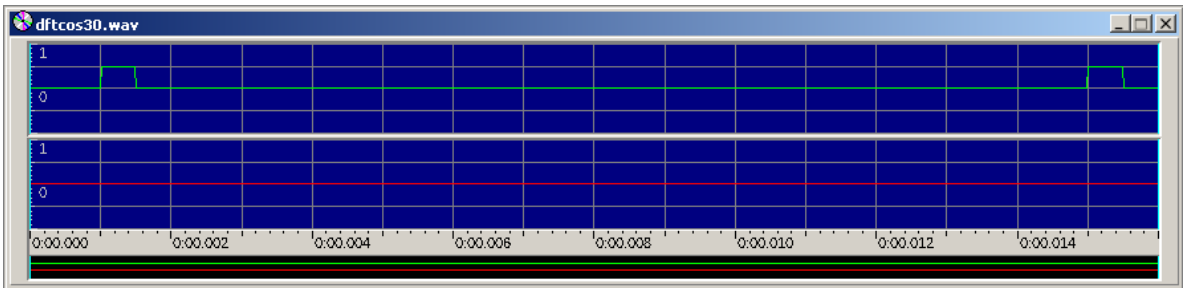

Ilustración 128 DFT del Coseno de 1875 Hz

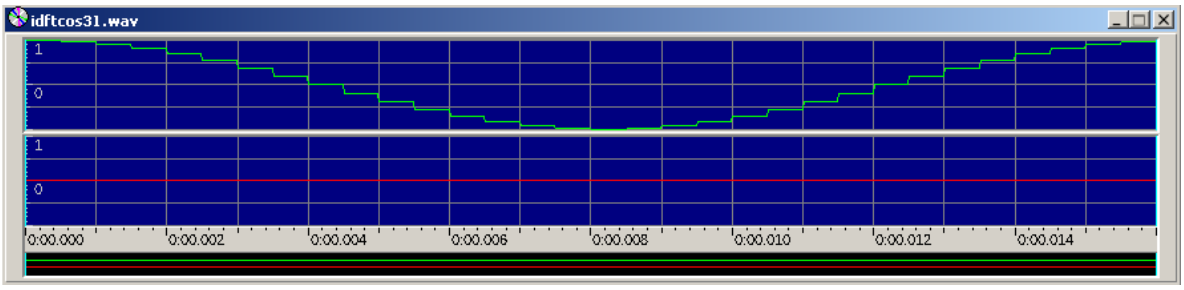

Ilustración 129 Coseno de 1937.5 Hz

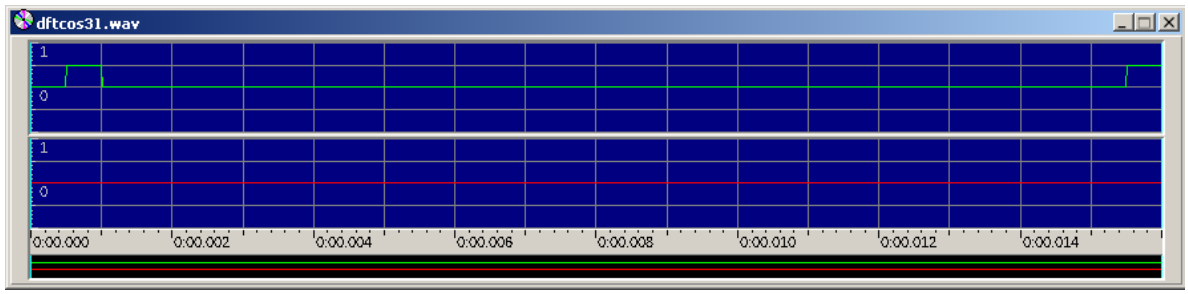

Ilustración 130 DFT del Coseno de 1937.5 Hz

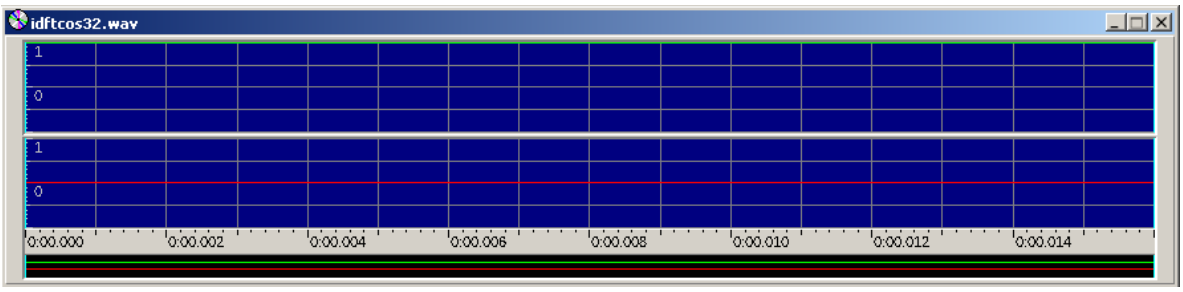

Ilustración 131 Coseno de 2000

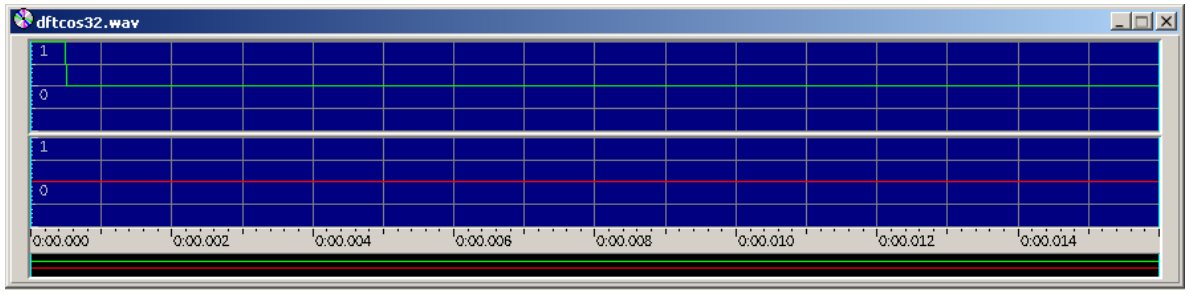

Ilustración 132 DFT del Coseno de 2000 Hz

Seno y coseno

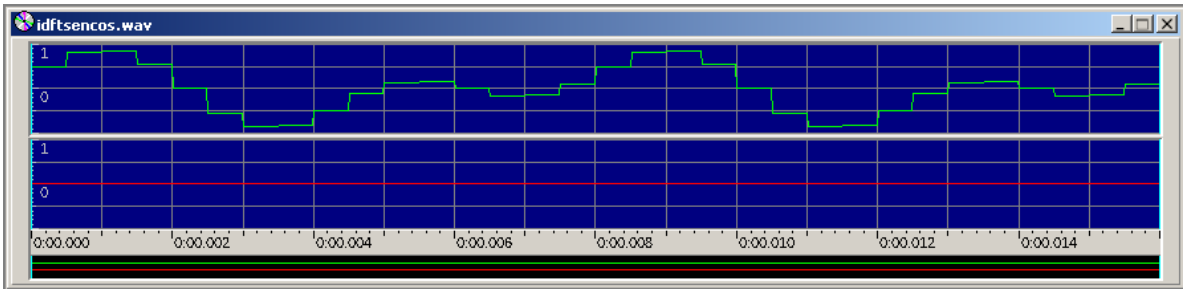

Ilustración 133 Coseno de 125 Hz más Seno de 250 Hz

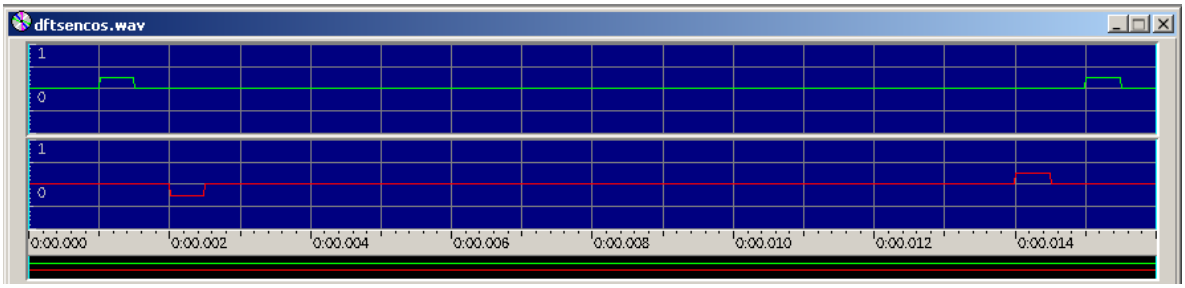

Ilustración 134 DFT del Coseno de 125 Hz más Seno de 250 Hz

Análisis Práctico

Como pudimos observar cuando regresamos al tiempo, como las señales al pasarlas a la frecuencia solo tenian valores reales, se pierde la parte imaginaria o se regresa con un valor de 0.

Es importante señalar que la función coseno es par por lo cual:

$$Cos(2 * \pi * f * t) = Cos(2 * \pi * -f * t)$$

Donde:

f: Es la frecuencia.

Dalo lo anterior una vez que no se cumple el teorema de muestreo las funciones coseno son identicas. Pero la función seno es impar por lo cual:

$$Sin(2 * \pi * f * t) \neq Sin(2 * \pi * -f * t)$$

Donde:

f: Es la frecuencia.

En consecuencia se repite la frecuencia pero la señal cambia de la frecuencia positiva a negativa y viceversa (Ver ilustación 64 e ilustación 4).

Conclusiones

Con la IDFT podimos regresar de una señal en frecuencia a una señal en el tiempo, donde las señales son iguales a antes de pasarlas a la frecuecia.

Si sacamos el promedio en la DFT nos ahorramos cómputo que si los sacamos en en la DFTI.

Y podemos decir que la Transformada Discreta de Fourier nos sirve para reconstruir señales.

Código

wav.h

```
#include <stdio.h>
#include <stdlib.h>

typedef struct CabeceraWAV{
	unsigned char ChunkID[4];
	unsigned int ChunkSize;
	unsigned char Format[4];
	unsigned char Subchunk1_ID[4];
	unsigned char Subchunk1_Size[4];
	unsigned char AudioFormat[2];
	unsigned char NumChannels[2];
	unsigned int SampleRate;
	unsigned int ByteRate;
	unsigned short BlockAlign;
	unsigned short BitsPerSample;
	unsigned char Subchunk2_ID[4];
	unsigned int Subchunk2_Size;//Catidad de bytes de informacion de la
señal
}Cabecera;

void informacionAudio(Cabecera cabeceraAudio){
	printf("%c%c%c%c\n",cabeceraAudio.ChunkID[0],cabeceraAudio.ChunkID[
1],cabeceraAudio.ChunkID[2],cabeceraAudio.ChunkID[3]);
	printf("ChunkSize: %d\n",cabeceraAudio.ChunkSize);
	printf("%c%c%c%c\n",cabeceraAudio.Format[0],cabeceraAudio.Format[1]
,cabeceraAudio.Format[2],cabeceraAudio.Format[3]);
	printf("%c%c%c%c\n",cabeceraAudio.Subchunk1_ID[0],cabeceraAudio.Sub
chunk1_ID[1],cabeceraAudio.Subchunk1_ID[2],cabeceraAudio.Subchunk1_ID[3])
;
	printf("Subchunk1_Size:%d
%d%d%d\n",cabeceraAudio.Subchunk1_Size[0],cabeceraAudio.Subchunk1_Size[1]
,cabeceraAudio.Subchunk1_Size[2],cabeceraAudio.Subchunk1_Size[3]);
	printf("Formato del audio: %d\n",cabeceraAudio.AudioFormat[0] );
	printf("Numero de canales: %d\n",cabeceraAudio.NumChannels[0] );
	printf("Muestras por segundo %d\n",cabeceraAudio.SampleRate);
	printf("Bytes por segundo: %d\n",cabeceraAudio.ByteRate);
	printf("Numero de bytes por muestra %d\n",cabeceraAudio.BlockAlign
);
	printf("Numero de bits por muestra: %d\n",
cabeceraAudio.BitsPerSample);
	printf("%c%c%c%c\n",cabeceraAudio.Subchunk2_ID[0],cabeceraAudio.Sub
chunk2_ID[1],cabeceraAudio.Subchunk2_ID[2],cabeceraAudio.Subchunk2_ID[3])
;
	printf("Numero de bytes de la informacion
%d\n",cabeceraAudio.Subchunk2_Size);
	printf("Numero de muestras
%d\n",cabeceraAudio.Subchunk2_Size/(cabeceraAudio.NumChannels[0]*(cabecer
aAudio.BitsPerSample/8)));
	printf("Bytes de sobra (informaciondesconocida):
%d\n",cabeceraAudio.ChunkSize-36-cabeceraAudio.Subchunk2_Size);
	printf("Tam total del archivo: %d\n",cabeceraAudio.ChunkSize+8);
}
```

```
//Resegra el tamaño completo de el archivo en bytes
unsigned int getFileSize(Cabecera CW){
	return CW.ChunkSize+8;
}

//Regresa el tamaño de las muestras en bytes
char getSampleSize(Cabecera CW){
	return (CW.BitsPerSample/8);
}

//Regresa el numero de canales;
char getCannalNumber(Cabecera CW){
	return CW.NumChannels[0];
}


//Regresa el numero de muestras de audio si es momo o estero regresa la
misma cantidad por lo cual en estereo se debe multiplicar por 2
unsigned int getNumberAudioSamples(Cabecera CW){
	return CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8));
}

//Regresa el tamanio total del archivo wav-8
unsigned int getNumberBytesAudioInformation(Cabecera CW){
	return CW.Subchunk2_Size;
}


//Convierte una cabecera de tipo mono a una de tipo estereo
Cabecera setAudioStereo(Cabecera CW){
	CW.ChunkSize+=CW.Subchunk2_Size;
	CW.Subchunk2_Size*=2;
	CW.NumChannels[0]=2;
	CW.ByteRate*=2;
	CW.BlockAlign*=2;
	return CW;
}


//regresa el numero de bytes del pie del archivo wav
unsigned int getNumberBytesFoot(Cabecera CW){
	return CW.ChunkSize-36-CW.Subchunk2_Size;
}

//Regresa la cabecera del archvo wav
Cabecera getHeader(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	fclose(archivo);
	return CW;
}

//Regresa las muestras de un archivo con tamanio de 8 bits
char* getAudioSamples8bits(char *archivoWavEntrada){
	Cabecera CW;
```

```
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;
	char *Samples=malloc(sizeof(char)*numeroMuestras);
	fread(Samples,numeroMuestras,1,archivo);
	fclose(archivo);
	return Samples;
}

//Regresa las muestras de audio de 16 bits mono
short* getAudioSamples16bits(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;
	short *Samples=malloc(sizeof(short)*numeroMuestras);
	fread(Samples,numeroMuestras*2,1,archivo);
	fclose(archivo);
	return Samples;
}
short* getAudioSamples16bitsStereo(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;
	numeroMuestras*=2;
	short *Samples=malloc(sizeof(short)*numeroMuestras);
	fread(Samples,numeroMuestras*2,1,archivo);
	fclose(archivo);
	return Samples;
}

//Regresa las muestras de 16bits del canal1
short* getAudioSamples16bisCanal1(char *archivoWavEntrada){
	Cabecera CW=getHeader(archivoWavEntrada);
	if(getCannalNumber(CW)==2){
		short
*muestrasAudio=getAudioSamples16bitsStereo(archivoWavEntrada);
		unsigned int tam=getNumberAudioSamples(CW);
		short *canal1=malloc(tam*sizeof(short));
		for(unsigned int i=0;i<tam;i++){
			canal1[i]=muestrasAudio[2*i];
		}
		return canal1;
	}else{
		printf("El archivo: %s no es estereo\n",archivoWavEntrada);
		return NULL;
	}
}

//Regresa las muestras de 16bits del canal2
```

```
short* getAudioSamples16bisCanal2(char *archivoWavEntrada){
	Cabecera CW=getHeader(archivoWavEntrada);
	if(getCannalNumber(CW)==2){
		short
*muestrasAudio=getAudioSamples16bitsStereo(archivoWavEntrada);
		unsigned int tam=getNumberAudioSamples(CW);
		short *canal2=malloc(tam*sizeof(short));
		for(unsigned int i=0;i<tam;i++){
				canal2[i]=muestrasAudio[2*i+1];
		}
		return canal2;
	}else{
		printf("El archivo: %s no es estereo\n",archivoWavEntrada);
		return NULL;
	}
}



//Regresa las bytes del pie del archivo wav
char* getFileFoot(char *archivoWavEntrada){
	Cabecera CW;
	FILE *archivo=fopen(archivoWavEntrada,"r");
	fread(&CW,44,1,archivo);
	unsigned int bytesSobrantes=CW.ChunkSize-CW.Subchunk2_Size-36;
	char *bytes=malloc(sizeof(char)*CW.Subchunk2_Size);
	fread(bytes,CW.Subchunk2_Size,1,archivo);
	bytes=malloc(sizeof(char)*bytesSobrantes);
	fread(bytes,bytesSobrantes,1,archivo);
	return bytes;
}
```

IDFT.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wav.h"
#define PI 3.141592653589793

void main(int argc, char *argv[]){

	Cabecera CW;
	FILE *archivoWavEntrada;
	archivoWavEntrada=fopen(argv[1],"r");

	if(archivoWavEntrada==NULL){
		printf("Error en la lectura del archivo de audio\n");
	}else{
		Cabecera CW=getHeader(argv[1]);

		if(getCannalNumber(CW)==2){
			FILE *archivoWavSalida=fopen(argv[2],"w");

			if(archivoWavSalida!=NULL){

				fwrite(&CW,44,1,archivoWavSalida);
				short
*muestrasAudioCanal1=getAudioSamples16bisCanal1(argv[1]);
				short
*muestrasAudioCanal2=getAudioSamples16bisCanal2(argv[1]);

				int
numMuestrasAudio=getNumberAudioSamples(CW);//Canal1 y canal 2
				double
*muestrasAudioSalida=malloc(sizeof(double)*numMuestrasAudio*2);
				short
*muestrasAudioFinal=malloc(sizeof(short)*numMuestrasAudio*2);//incluira
ambos canales

				/**	Transforda Discreta de Fourier Inversa
				*	El canal 1 es la parte real, el canal 2 es
la parte imaginaria.
				*
				**/

				double C=2*PI/numMuestrasAudio;
				for(int n=0;n<numMuestrasAudio;n++){
					muestrasAudioSalida[2*n]=0;
					muestrasAudioSalida[2*n+1]=0;
					for(int k=0;k<numMuestrasAudio;k++){
	//Parte Real

	muestrasAudioSalida[2*n]+=((muestrasAudioCanal1[k]*cos(C*k*n))-
(muestrasAudioCanal2[k]*sin(C*k*n)));
					}

					for(int k=0;k<numMuestrasAudio;k++){
	//Parte Imaginaria
```

```
	muestrasAudioSalida[2*n+1]+=((muestrasAudioCanal1[k]*sin(C*k*n))+(m
uestrasAudioCanal2[k]*cos(C*k*n)));
                                }
                        }

                        numMuestrasAudio*=2;
                        for(int i=0;i<numMuestrasAudio;i++){

	muestrasAudioFinal[i]=muestrasAudioSalida[i];
                        }

                        unsigned int
bytesAudio=getNumberBytesAudioInformation(CW);

                        fwrite(muestrasAudioFinal,
bytesAudio,1,archivoWavSalida);
                        char *pie=getFileFoot(argv[1]);

	fwrite(pie,getNumberBytesFoot(CW),1,archivoWavSalida);
                        fclose(archivoWavSalida);
                        fclose(archivoWavEntrada);

                }else{
                        printf("No se pudo crear el archivo de salida\n");
                }
        }else{
                printf("El archivo el monoaural\n");
        }

	}
}
```